

**Czech University of Life Sciences Prague**

**Faculty of Economics and Management**

**Department of Information Engineering**



**Bachelor Thesis**

**Backend development with Python/Django**

**Evgeny Nikolaev**

**© 2021 CULS Prague**

# CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

## BACHELOR THESIS ASSIGNMENT

Evgeny Nikolaev

Systems Engineering and Informatics

Informatics

Thesis title

**Back-end development with python/django**

---

### Objectives of thesis

The main objective of this bachelor thesis is to develop a website which will demonstrate possibilities of the Python/Django framework.

The partial goals of the thesis are:

- to create a literature review of web development field,
- to make a review of advantages and disadvantages over other web frameworks,
- to design a website architecture.

### Methodology

Methodology of the thesis is based on study and analysis of information resources. The practical part is focused on design website using Use Cases and UML diagrams. Then it is necessary to use the training literature to write the front-end and the back- end components. On the basis of theoretical knowledge and author s own work, the conclusion of the thesis will be formulated.

## **The proposed extent of the thesis**

30–40pages

## **Keywords**

Programming, design, front-end, back-end, python, django.

---

## **Recommended information sources**

DOWEK, Gilles. Principles of programming languages. Springer Science&BusinessMedia,2009, ISBN 9781848820326

Jacob Kaplan–Moss; Adrian Holovaty. The Definitive Guide to Django: Web Development Done Right SHEN, Alexander. Algorithms and Programming: Problems and solutions Springer Science & Business Media, 2011, ISBN9781441917485

---

## **Expected date of thesis defence**

2020/21 SS – FEM

## **The Bachelor Thesis Supervisor**

Ing. Jan Tyrychtr, Ph.D.

## **Supervising department**

Department of Information Engineering

Electronic approval: 24. 1. 2019

**Ing. Martin Pelikán, Ph.D.**

Head of department

Electronic approval: 24. 1. 2019

**Ing. Matin Pelikán, Ph.D.**

Dean

Prague on 14. 03. 2021

## **Declaration**

I declare that I have worked on my Bachelor thesis, titled “Back-end development with Python/Django,” all by myself, and I have used only the sources mentioned at the end of the thesis. As the author of this Bachelor’s thesis, I declare that the thesis does not violate the copyrights of any other person.

In Prague on 15.03.2021

---

## **Acknowledgement**

I would like to thank Ing. Jan Tyrychtr, Ph.D., for advice and support during my work on this thesis.

# Backend development with Python/Django

## Abstract

The following thesis focuses on the development of a website. It intends to show the benefits of developing a website using the Django framework and explains all the stages for developing it. The thesis covers the main stages in the process of designing the architecture, developing a website, deploying it—including the description of a platform for deployment, the techniques that are used for designing it, and the tools that are used in a project—setting up the environment, and developing the project website. Also, this thesis includes a comparison among other frameworks viz. Django, Express.js, Ruby on Rails, and Laravel. The result will be a working, hosted website, one that is fully accessible through a hosting service.

**Keywords:** Programming, design, frontend, backend, Python, Django.

# Vývoj backend s Pythonem / Django

## **Abstrakt**

Následující práce se zaměřuje na vývoj webových stránek. Má v úmyslu ukázat výhody vývoje webových stránek pomocí frameworku Django a vysvětluje všechny fáze jejich vývoje. Tato práce se zabývá hlavními fázemi procesu navrhování architektury, vývoje webových stránek, jejich nasazení – včetně popisu platformy pro nasazení, technik, které se k jejímu návrhu používají, a nástrojů, které se používají v projektu – nastavení prostředí a rozvoj webových stránek projektu. Tato práce také obsahuje srovnání mezi jinými frameworky, viz. Django, Express.js, Ruby on Rails a Laravel. Výsledkem bude fungující hostovaný web, který je plně přístupný prostřednictvím hostingové služby.

**Klíčová slova:** Programming, design, frontend, backend, Python, Django.

# Table of contents

<b>1</b>	<b>Introduction .....</b>	<b>12</b>
<b>2</b>	<b>Objectives and Methodology .....</b>	<b>13</b>
2.1	Objectives .....	13
2.2	Methodology .....	13
<b>3</b>	<b>Literature Review .....</b>	<b>14</b>
3.1	Short history of web development .....	14
3.1.1	Web development technology .....	15
3.2	Django Web Framework .....	15
3.3	Why Django? .....	16
3.4	MVC (Model-View-Controller).....	18
3.4.1	Model-View-Controller.....	18
3.4.2	Model-View-Controller.....	19
3.4.3	Model-View-Controller.....	19
3.5	Python.....	20
3.6	Comparison of frameworks .....	21
3.6.1	Introduction to frameworks .....	21
3.6.1.1	Ruby on Rails .....	21
3.6.1.2	Express.js.....	21
3.6.1.3	Laravel.....	21
3.6.2	Code comparison.....	21
3.6.2.1	Ruby on Rails .....	21
3.6.2.2	Express.js.....	22
3.6.2.3	Laravel.....	23
3.6.2.4	Django .....	23
3.6.3	Documentation and support.....	24
3.6.3.1	Ruby on Rails .....	24
3.6.3.2	Express.js.....	24
3.6.3.3	Laravel.....	24
3.6.3.4	Django .....	24
3.6.4	Popularity.....	25
3.6.4.1	Ruby on Rails .....	25
3.6.4.2	Express.js.....	25
3.6.4.3	Laravel.....	26



3.6.4.4	Django .....	26
3.6.5	Development tools and packages management system.....	26
3.6.5.1	Ruby on Rails .....	26
3.6.5.2	Express.js.....	27
3.6.5.3	Laravel.....	27
3.6.5.4	Django .....	27
3.6.6	Database .....	27
3.6.6.1	Ruby on Rails .....	28
3.6.6.2	Express.js.....	28
3.6.6.3	Laravel.....	28
3.6.6.4	Django .....	28
3.6.7	Performance.....	28
3.7	Tools for web development .....	29
3.7.1	Python environment .....	29
3.7.2	Text Editor.....	30
3.7.3	Website hosting.....	30
3.7.4	HTML.....	30
3.7.5	Bootstrap.....	31
3.7.6	CKEditor.....	31
3.7.7	Pillow .....	31
3.7.8	JavaScript.....	31
3.8	Website architecture design.....	31
3.8.1	Entity Relationship Diagram .....	32
3.8.2	Use Case diagram.....	33
<b>4</b>	<b>Practical Part .....</b>	<b>34</b>
4.1	Website architecture design.....	34
4.1.1	Entity Relationship Diagram .....	34
4.1.2	Use Cases.....	34
4.2	Setting up the development environment.....	35
4.2.1	Setting up the virtual environment.....	35
4.2.2	Starting the project .....	35
4.3	Development process .....	36
4.3.1	Thesis project folder.....	36
4.3.1.1	Settings.py .....	36
4.3.1.2	urls.py.....	37
4.3.2	Blog application .....	37
4.3.2.1	Models.py .....	37

4.3.2.2	Views.py .....	38
4.3.2.3	Templates .....	39
4.3.2.4	Forms.py .....	40
4.3.2.5	Admin.py .....	41
4.3.3	Login system .....	41
4.4	Deploying a website .....	41
4.4.1	Deployment using Heroku git .....	42
<b>5</b>	<b>Results and Discussion .....</b>	<b>43</b>
<b>6</b>	<b>Conclusion .....</b>	<b>44</b>
<b>7</b>	<b>References .....</b>	<b>45</b>
<b>8</b>	<b>Appendix .....</b>	<b>47</b>

## List of figures

Figure 1: Django versions (Foundation, 2021) .....	17
Figure 2: Technology overview for 2018 (Stack Exchange, Inc., 2018) .....	17
Figure 3: Technology overview for 2020 (Stack Exchange, Inc., 2020) .....	18
Figure 4 MVC logic (Mozilla, 2020) .....	19
Figure 5 MVT logic (George, 2019) .....	20
Figure 6 For Loop in Ruby (Hal Fulton, 2015) .....	22
Figure 7 Hello world express (Foundation, 2021) .....	22
Figure 8 For loop In JavaScript (w3schools, 2021) .....	22
Figure 9 Terminal commands (Laravel, 2021) .....	23
Figure 10 Hello world program (Laravel, 2021).....	23
Figure 11 PHP for loop (w3schools, 2021) .....	23
Figure 12 Hello world (Foundation, 2021).....	23
Figure 13 url.py (Foundation, 2021) .....	24
Figure 14 For loop in Python\Django (Lutz, 2013) .....	24
Figure 15 Ruby on Rails Usage Statistics (BuiltWith Pty Ltd, 2020).....	25
Figure 16 Express Usage Statistics (BuiltWith Pty Ltd, 2020) .....	25
Figure 17 Laravel Usage Statistics (BuiltWith Pty Ltd, 2020).....	26
Figure 18 Django Language Usage Statistics (BuiltWith Pty Ltd, 2020) .....	26
Figure 19 Top 5 databases for a year (Solid IT GmbH, 2021).....	28
Figure 20 Best plaintext responses per second (TechEmpower, Inc., 2021).....	29
Figure 21 Best JSON responses per second (TechEmpower, Inc., 2021).....	29
Figure 22 Best fortunes responses per second (TechEmpower, Inc., 2021).....	29
Figure 23 (George, 2019) .....	30
Figure 24 Representation of the HTML document as a tree diagram (Lewis Coulson, 2019) .....	30
Figure 25 A sample of Chen diagram (Wittwer, 1995).....	33
Figure 26 Use cases example (Dan Pilon, 2005).....	33
Figure 27 Class diagram.....	34
Figure 28 Use case diagram.....	34
Figure 29 Activating virtual environment and installing Django .....	35

Figure 30 Super user creation .....	36
Figure 31 List of installed apps.....	37
Figure 32 URLs from thesis project.....	37
Figure 33 Applying changes in model.py.....	38
Figure 34 Class post from models.py.....	38
Figure 35 Class example from view.py.....	39
Figure 36 add_category html file .....	40
Figure 37 PostForm from forms.py.....	40
Figure 38 Script from add_post.html .....	40
Figure 39 admin.py .....	41
Figure 40 Django administration page .....	41
Figure 41 Allowed host .....	42
Figure 42 Static root.....	42
Figure 43 Deploying a website .....	42

## List of tables

Table 1 A brief history of web development (CERN, 2021).....	14
--	----

## List of abbreviations

List...

# 1 Introduction

The world of web development has evolved with giant strides since the inception of the World Wide Web (WWW). With the development of the web industry, many different technologies have emerged for creating websites and web services. Though some of these technologies appeared quite early on, they have not gained sufficient popularity when compared with other technologies.

Before starting the development of a website, it is necessary to prepare an architecture design of it using a popular Unified Modeling Language (UML) diagram and an Entity Relationship diagram.

One of the technologies used to create a website is the one called web frameworks. Basically, a web framework is a group of packages and modules that enables developers to write websites or web applications with more comfort so that the developers have no need to handle low-level details like protocols, sockets, etc., since the framework has already collected everything inside it. There are a lot of different techniques to create a website. You can develop a website from scratch using only a programming language as a backend part and standard Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) for the frontend part, or you can use different frameworks, packages, and modules to make the development simpler and much faster. Technologies like the Django web framework are used to create websites.

The last step is to deploy a website using a hosting service.

Since there are many ways of creating a website using both simple technologies and more advanced ones, we have decided to focus only on designing the architecture of a website and mainly on developing its server side. Also, as a complement, the goal is to compare other frameworks by the type of server side while using a lot of resources.

## **2 Objectives and Methodology**

### **2.1 Objectives**

The main objective of this Bachelor's thesis is to develop a website that will demonstrate the possibilities of the Python/Django framework.

The partial goals of the thesis are:

- to create a literature review of the field of web development
- to conduct a review of the advantages and disadvantages of other web frameworks
- to design a website architecture

### **2.2 Methodology**

The methodology of this thesis is based on the study and analysis of information resources. The practical part is focused on developing a website step by step. Firstly, the aim is to design a website using Use Cases, which are a part of UML diagrams, and create Entity Relationship Diagrams. These diagrams will be created using study materials from the books related to the topic. For Use Cases, special attributes like actor, use cases, associations, and system boundary box will be used. As for the Entity Relationship Diagrams, entity symbols, relationship symbols, attribute symbols, cardinality, and ordinality will be used. It is also necessary to use the training literature and documentation to write the frontend and backend components. For the frontend, the responsible technologies are: Bootstrap, JavaScript, and HTML. The backend part will be written using Python programming language and the implemented features of Django. The website will be functional and hosted on a cloud-based service called Heroku.

### 3 Literature Review

#### 3.1 Short history of web development

The history of web development begins with the invention of the World Wide Web (WWW) by the famous British scientist Tim Berners-Lee in 1989 at Conseil Européen pour la Recherche Nucléaire (European Council for Nuclear Research) or CERN. It was invented with the aim of rapidly sharing information between scientists at universities and institutes around the world. By the end of 1990, Tim Berners-Lee had already written the web server and browser and had run it on a NeXT computer. The first webpage was written using the “hypertext document,” which was invented by Tim Berners-Lee; it was afterward called Hypertext Markup Language (HTML). These first steps toward the creation of the WWW led to one of the biggest industries in the world (CERN, 2021).

Table 1 A brief history of web development (CERN, 2021)

March 1989	Tim Berners-Lee submitted his first proposal for what became the World Wide Web
November 1990	Management proposal for a World Wide Web project
December 1990	The world’s first browser/editor, website, and server go live at CERN
March 1991	Line mode browser available at CERN
August 1991	Sir Berners-Lee announces the WWW software on the internet
December 1991	First web server outside Europe
January 1992	WWW moves from prototype to production
September 1992	A growing number of web servers and browsers emerge
January 1993	First pre-release of the Mosaic browser
April 1993	CERN puts the WWW in the public domain
December 1994	More than 10,000 web servers established around the world

Naturally, the more the web development grew, the more the different kinds of technologies that began to appear for creating websites. This is how the most popular instruments for creating website appeared viz.:

- HTML

- Cascading Style Sheet (CSS)
- JavaScript
- TypeScript
- PHP: Hypertext Preprocessor (PHP)

### 3.1.1 Web development technology

Web development is the programming that allows the developer to create website functionality just as he/she wants. Web development gives the developer a lot of opportunities to make web applications, social network applications, or even browser games.

Web development consists of:

- Frontend or Client-side coding
- Backend or Server-side coding
- Database programming

The frontend responds to the visual part of a website. It shows how the user interacts with the webpage. Most of the websites use frontend HTML, CSS, and JavaScript.

The backend responds to take care of the non-visible part of websites like hosting database, applications, etc.

## 3.2 Django Web Framework

Django is an open-sourced web framework based on Python programming language. It is a fast, secure, and easy-to-use web framework, which allows one to develop a website as quickly as possible.

Django uses the Model-View-Controller (MVC) architecture pattern. This was created by Adrian Holovaty and Simon Willison as an internal project at *The Lawrence Journal-World* newspaper in 2003. It was released officially in July 2005; in 2008, it was announced that the Django Software Foundation (DSF) would support its future development. The latest version of Django is 3.1.3; it was released on November 2, 2020.

Django has the following advantages:

- **Batteries are included**—It basically gives you a lot of stuff, which is already included in Django. This means that Django gives you some implemented instruments that you can use at any point of time.

- **It does not get in your way**—Django does not give you unnecessary functions in the Django creation application process. Therefore, everything you have is only the basic settings file, some folders, and clear starter files.
- **Built-in-admin**—Django gives you the ability to use an admin interface, which you can customize to suit your needs.
- **Scalable**—Django separates your code from media files like images, CSS, and JavaScript files.
- **Packages**—Since Django is an open-source framework, it has a large community all over the world, members of which upload their own packages for different kinds of stuff. You can always find ready-to-use apps or tools for your own project.

(George, 2019)

### 3.3 Why Django?

Why is it worth paying so much attention to this particular framework? Django—like any other tool—has its advantages and disadvantages. Here is a closer look at its positive aspects.

Since its release in 2005, more than 15 years have passed, and yet Django continues to evolve; the number of users and companies that use it in their projects continues to increase. Django has already found trust among industry mastodons like Instagram, Pinterest, Disqus, Bitbucket, Mozilla, National Geographic, Discovery, and Google. At the very least, this indicates that though Django does not occupy the highest position among retailers, it is definitely popular and has been tested over time.

If you believe that this is just an open-source project without any profits, then you are mistaken: Django is definitely not going to abandon its users. In addition to the fact that Django is constantly releasing new versions of the project, it also has a special feature viz. Long-term Support (LTS). For example, the latest version of LTS Django 2.2 will end in April 2022.



Release Series	Latest Release	End of mainstream support <sup>1</sup>	End of extended support <sup>2</sup>
3.1	3.1.5	April 2021	December 2021
3.0	3.0.11	August, 2020	April, 2021
2.2 LTS	2.2.17	December 2, 2019	April 2022
2.1	2.1.15	April 1, 2019	December 2, 2019
2.0	2.0.13	August 1, 2018	April 1, 2019
1.11 LTS <sup>3</sup>	1.11.29	December 2, 2017	April 1, 2020
1.10	1.10.8	April 4, 2017	December 2, 2017
1.9	1.9.13	August 1, 2016	April 4, 2017
1.8 LTS	1.8.19	December 1, 2015	April 1, 2018
1.7	1.7.11	April 1, 2015	December 1, 2015
1.6	1.6.11	September 2, 2014	April 1, 2015
1.5	1.5.12	November 6, 2013	September 2, 2014
1.4 LTS	1.4.22	February 26, 2013	October 1, 2015
1.3	1.3.7	March 23, 2012	February 26, 2013

Figure 1: Django versions (Foundation, 2021)

Why is it worth studying, or at least taking a closer look, at this framework? To answer this question, you need to take a closer look at the popularity of this language and the growth rate of users, according to the statistics of the popular website Stack Overflow (Figure 1, Figure 2).

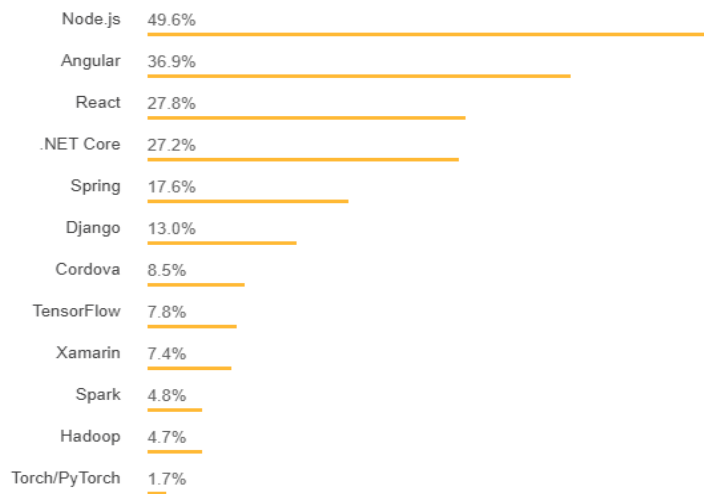


Figure 2: Technology overview for 2018 (Stack Exchange, Inc., 2018)

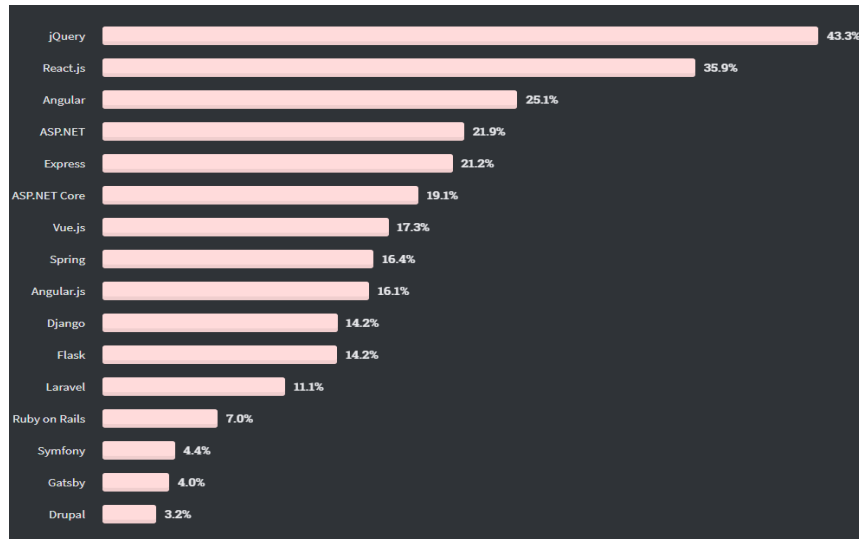


Figure 3: Technology overview for 2020 (Stack Exchange, Inc., 2020)

As can be seen, the popularity of Django in the past two years has grown from 13% of users to 14.2%. This framework has already found a place on a market, and is holding firmly on to it.

### 3.4 MVC (Model-View-Controller)

Model-View-Controller architecture was invented by Professor Trygve Reenskaug and the 1970s' programming language "smalltalk" was developed with MVC architecture as its base. MVC separates data as model, user interface as view, and data handling logic as controller. Each part included in the architecture, namely the model, view, and controller, will be explained in the following.

#### 3.4.1 Model-View-Controller

The model is an information set for each of the groups on the website called classes. That information or data is what you want to show the user. Classes, in turn, make up each part of the project.

Let us imagine a university website. On this site, there are lots of different kinds of information like teaching staff, information about faculties, and so on. The teaching staff is a class that has the attributes of a name, surname, and contact information. Model takes all this data from the database so that it can be read, updated, and deleted.

### 3.4.2 Model-View-Controller

The View is used for the all-user interface logic of the application. For example, the movie application view describes components like dropdowns, movie name, etc., which the user interacts with.

### 3.4.3 Model-View-Controller

The controller is responsible for executing requests received from the user. In a web application, the controller usually parses the parameters of the HTTP request, calls the model to get or change some data, and finally calls the View to display the result of the request. In desktop applications, the Controller is responsible for handling button clicks and other user actions.

One Controller can work with several Models, and vice versa, i.e. one Model can be used in several Controllers.

All application functionality is contained in the model. The controller and the view only provide the ability for the user to interact with the model and display data from it (Model View Controller architecture, 2009).

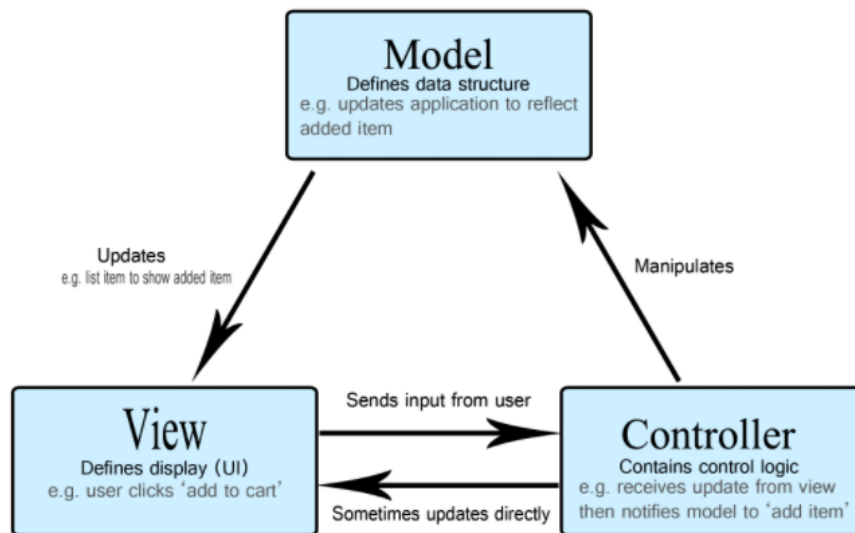


Figure 4 MVC logic (Mozilla, 2020)

Even if Django is a MVC architecture-based framework, it has a somewhat different terminology. According to Django Software Foundation, it is the Model-Template-View framework (George, 2019):

1. The model is the same.

2. The “template” represents “view”
3. The “view” represents “controller”

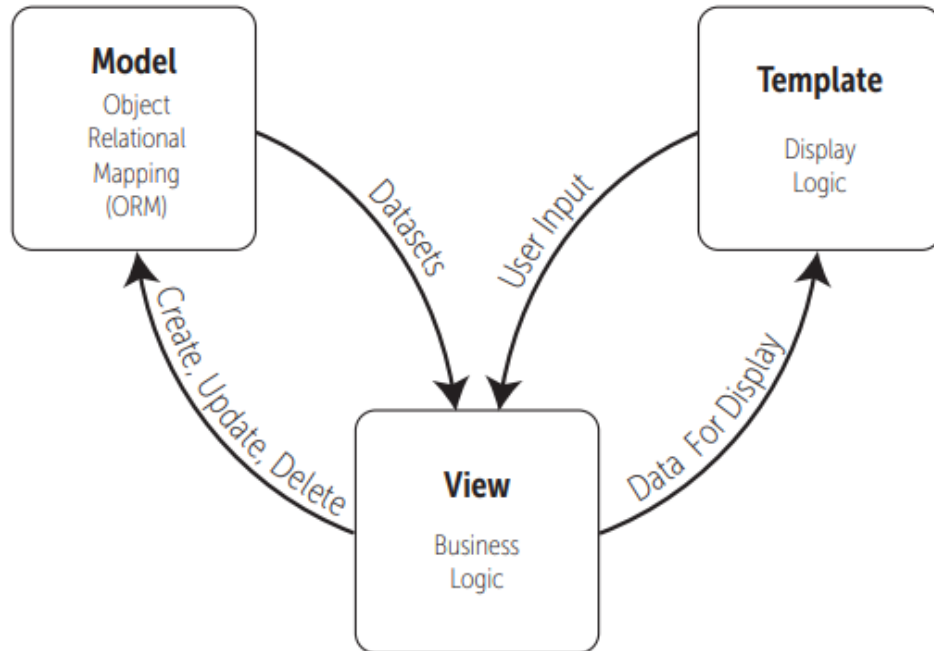


Figure 5 MVT logic (George, 2019)

### 3.5 Python

Python is an object-oriented high-level programming (OOP) language focused on improving developer productivity and code readability. It was created by Guido van Rossum and released in 1991.

It uses all the advantages associated with OOP:

- Polymorphism—which allows one to define methods in the child class with the same name as defined in their parent class;
- Operator overloading—which gives extended meaning beyond their predefined operational meaning;
- Multiple inheritance—the class can be derived from more than one base class.

The syntax is simple; its simplicity makes it easy to use, even for beginners. Python is used in a lot of areas viz. development fields of games, web, software and so on.

Advantages of Python (Lutz, 2013):

- Simple syntax.

- Large standard library for different type of tasks.

## **3.6 Comparison of frameworks**

Ruby on Rails, Express.js, Laravel (PHP), Django

### **3.6.1 Introduction to frameworks**

#### **3.6.1.1 Ruby on Rails**

The Ruby on Rails framework was released as open-source in 2004. It was created by the company 37signals; the main developer of the framework was David Heinemeier Hansson. It is based on a programming language called Ruby, which was created by Yukihiro Matsumoto (Bini, 2007).

#### **3.6.1.2 Express.js**

Express is an open-source web framework. It was created by TJ Holowaychuk specially for node.js. Its first release was on October 16, 2010. Express took its inspiration from the Ruby-based Sinatra web framework (Foundation, 2017).

#### **3.6.1.3 Laravel**

Laravel is an open-source web framework based on Symfony PHP web framework, created by Taylor Otwell, and realized on June 9, 2011. It uses a Module-View-Controller architecture (McCool, 2012).

### **3.6.2 Code comparison**

In almost every programming language or guide, the very first program is to say “hello world.” To compare all the four frameworks, it is necessary to write the same program on all of them. Also, if we take into account the fact that every scripting or ordinary programming language consists of an important part of loops, it is worth giving an example of how to use a for loop for each of the languages.

#### **3.6.2.1 Ruby on Rails**

Before writing your first “Hello world” program in rails, you need to go through some mandatory steps, such as installing the necessary programs and executing some commands in the terminal. First of all, you need to install:

- Ruby
- SQLite3
- Node.js
- Yarn

As the second step, you need to create an application using terminal and command “rails new <app name>.” The third step is to create—at a minimum—a route, a controller with an action, and a view. Finally, you can write in the html file `<h1>Hello, Rails!</h1>` and run the server using command “ruby bin\rails server” (Ruby on Rails, 2021).

For loop, as in the other programming languages, looks similar; it also performs the same task.

```
for x in list do
  print "#{x} "
end
```

Figure 6 For Loop in Ruby (Hal Fulton, 2015)

### 3.6.2.2 Express.js

Before creating simple a “hello world” program in express framework, you need to install node.js and express.js. Creating that program does not require the use of a generator; it just implements it in a single file with a simple code (Figure 5).

```
1 const express = require('express' 4.17.1 )
2 const app = express()
3 const port = 3000
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!')
7 })
8
9 app.listen(port, () => {
10  console.log(`Example app listening at http://localhost:${port}`)
11 })
```

Figure 7 Hello world express (Foundation, 2021)

Let us now examine “For loop” in JavaScript. Though it is similar to any other programming language, you need to remember that JavaScript is a dynamically typed language.

```
var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

Figure 8 For loop In JavaScript (w3schools, 2021)

### 3.6.2.3 Laravel

First of all, you need to install PHP, composer, and Laravel. After installing these, you need to create application using terminal (Figure 9).

```
laravel new example-app  
  
cd example-app  
  
php artisan serve
```

Figure 9 Terminal commands (Laravel, 2021)

To create a program, you need to write in a route file code on Figure 10.

```
use Illuminate\Support\Facades\Route;  
  
Route::get('/greeting', function () {  
    return 'Hello World!';  
});
```

Figure 10 Hello world program (Laravel, 2021)

For loop shown in Figure 11, PHP is similar to JavaScript, but variables are prefixed with a \$ symbol, which is typical for PHP language (Laravel, 2021).

```
<?php  
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}  
?>
```

Figure 11 PHP for loop (w3schools, 2021)

### 3.6.2.4 Django

To start work with Django, you need to take several steps. First, you need to install Python. As the second, you need to use a package installer pip download Django and create a project using the command “django-admin startproject <project name>” and create an application using “py manage.py startapp <app name>.”

The third step entails writing code in a views.py and creating a “hello world” program, as shown in Figure 10.

```
from django.http import HttpResponse  
  
def index(request):  
    return HttpResponse("Hello, world. You're at the polls index.")
```

Figure 12 Hello world (Foundation, 2021)

Finally, it is necessary to add the path to the application in a urls.py file (Figure 11).

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

Figure 13 url.py (Foundation, 2021)

Creating a for loop in Django is very simple. The “for loop” in Figure 12 creates a “for loop” by using a range of values between 0 to n.

```
for target in object:
    statements
    .
```

Figure 14 For loop in Python\Django (Lutz, 2013)

### 3.6.3 Documentation and support

#### 3.6.3.1 Ruby on Rails

Ruby on Rails was invented on December 2005 i.e. 15 years ago. Ruby on Rails boasts a great documentation, which makes an upgrade with notes of changes.

#### 3.6.3.2 Express.js

Express is a fairly young framework compared to Django and Ruby on Rails; it is only 10 years old. But it does give a list of recommended reading materials to study, and also has several opportunities for discussion or asking for help, which are indicated on the main site of the framework.

#### 3.6.3.3 Laravel

Laravel was released on June 9, 2011; it is the youngest framework of all four, at nine years of age. But at the same time, the programming language that it uses (PHP) is one of the oldest. This is undoubtedly one of its advantages, since a lot of training materials have been written all the time. There is also quite a large documentation on the framework itself.

#### 3.6.3.4 Django

Django is the oldest of all the four frameworks; it was released on July 15, 2005. It is now 15 years old. Django provides developers with documentation that provides a quick way to



get started with development as well as deeper things to understand the framework. Thanks to its community, there are also many teaching materials, books, and communities like Django Girls.

### 3.6.4 Popularity

#### 3.6.4.1 Ruby on Rails

According to the statistics from the BuiltWith websites, build on Ruby on Rails currently online is 1,532,730 (Figure 15). It occupies the leading position compared to other frameworks.

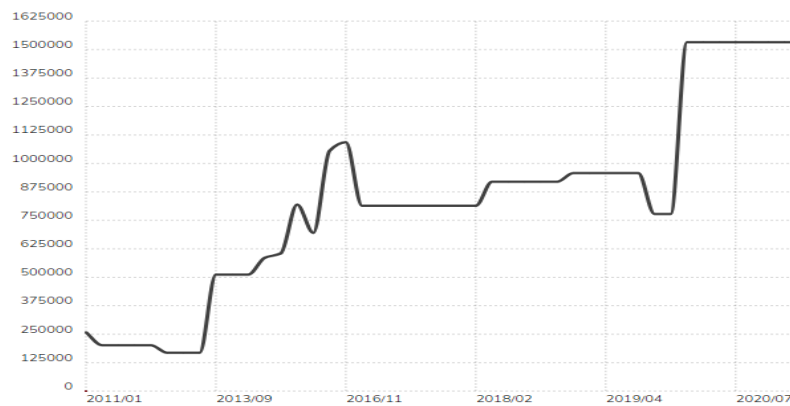


Figure 15 Ruby on Rails Usage Statistics (BuiltWith Pty Ltd, 2020)

#### 3.6.4.2 Express.js

According to statistics from the BuiltWith websites, build on Express.js currently online is 620,989 (Figure 16). It occupies the third place among the frameworks that are being compared. Also, we can see a stable growth in the number of websites.

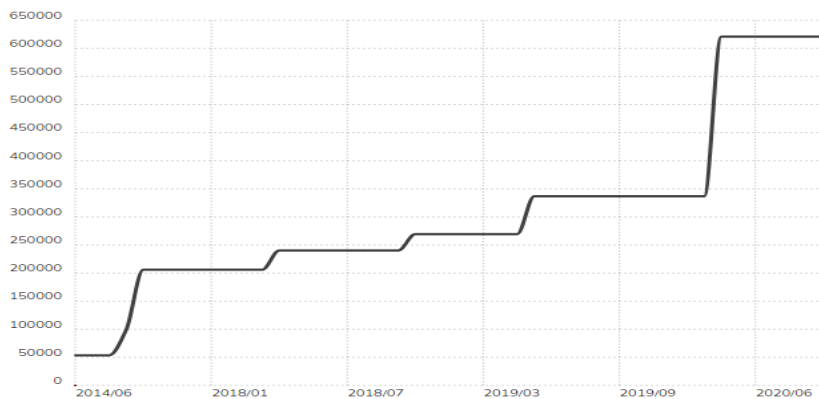


Figure 16 Express Usage Statistics (BuiltWith Pty Ltd, 2020)

### 3.6.4.3 Laravel

According to statistics from the BuiltWith websites, build on Laravel currently online is 678,361 (Figure 17). The number of online websites gives this framework the second place. However, we can observe strange behavior on the part of the graph.

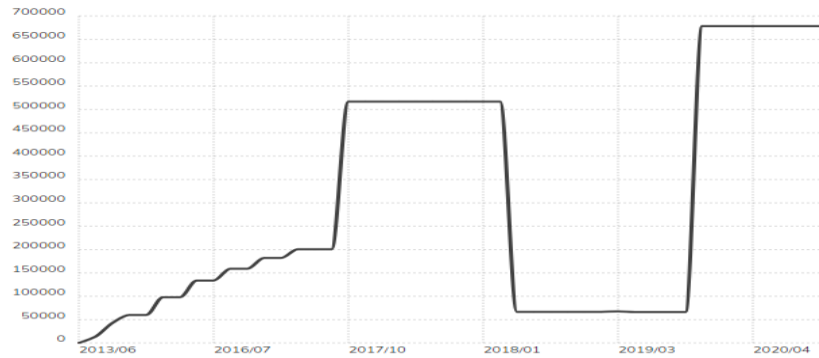


Figure 17 Laravel Usage Statistics (BuiltWith Pty Ltd, 2020)

### 3.6.4.4 Django

According to the statistics from the BuiltWith Django, the number of websites currently online is 30,016 (Figure 18). It means that Django occupies the last place compared to the other frameworks. Nevertheless, in the graph, we can see a big growth.



Figure 18 Django Language Usage Statistics (BuiltWith Pty Ltd, 2020)

## 3.6.5 Development tools and packages management system

Each framework has its own tools and packages management system. They are all similar to each other, but there are still minor differences; these will be described in this section. The current number of packages will be taken from Module Counts. It is necessary to keep in mind that the number of packages is increasing all the time.

### 3.6.5.1 Ruby on Rails

RubyGems is a package manager for Ruby and Ruby on Rails. It is installed with the installation of the Ruby programming language. All the packages are listed and described

on the RubyGems website, and currently there are 164,868 packages available. All packages can be installed through terminal (RubyGems, 2021) (DeBill, 2020).

### **3.6.5.2 Express.js**

Npm is a package manager used for Express.js and Node.js. Currently, Npm has 1,519,042 packages available and ready to use. This is the biggest number of packages among all the four frameworks. Again, packages can be installed using terminal after installing npm. A distinctive feature of the website is the sorting by:

- Optimal—taking average
- Popularity—shows the most popular packages
- Quality—shows this attribute based on package quality
- Maintenance—shows packages by upgrade activity

(npm, Inc, 2021) (DeBill, 2020)

### **3.6.5.3 Laravel**

Composer is a package manager for PHP; it is installed separately. A website called Packagist exists for searching packages. There are around 298,656 packages on the website. The special feature of this website is the assortment of tags, which helps you find what you need (Boggiano, 2021) (DeBill, 2020).

### **3.6.5.4 Django**

Django has a PyPi package manager. It installs itself automatically with Python. The number of packages is currently 291,959. The peculiarity of this website is definitely its search through the tag tree. Just like other package managers, packages are installed directly through the console (Python Software Foundation, 2021) (DeBill, 2020).

## **3.6.6 Database**

To compare the frameworks by database availability, you should take the top five databases from the db-engines website. For at least a year, there were five database management systems that occupied the top five places (Figure 19). The top five databases were: Oracle, MySQL, Microsoft SQL Server, PostgreSQL, and MongoDB (Solid IT Gmbh, 2021).

Rank			DBMS	Database Model	Score		
Feb 2021	Jan 2021	Feb 2020			Feb 2021	Jan 2021	Feb 2020
1.	1.	1.	Oracle	Relational, Multi-model	1316.67	-6.26	-28.08
2.	2.	2.	MySQL	Relational, Multi-model	1243.37	-8.69	-24.28
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1022.93	-8.30	-70.81
4.	4.	4.	PostgreSQL	Relational, Multi-model	550.96	-1.27	+44.02
5.	5.	5.	MongoDB	Document, Multi-model	458.95	+1.73	+25.62

Figure 19 Top 5 databases for a year (Solid IT GmbH, 2021)

### 3.6.6.1 Ruby on Rails

Ruby on Rails does not have a preinstalled database. All databases need to be downloaded and installed using the RubyGems package manager (Ruby on Rails, 2021).

### 3.6.6.2 Express.js

Express.js does not support any integrated database. Still, it is possible to install all databases through the package management system npm (OpenJS Foundation, 2021).

### 3.6.6.3 Laravel

Laravel has four databases integrated in PHP. The main database for PHP is MySQL. Oracle, PostgreSQL, and MongoDB can be installed through terminal using special command. As for Microsoft SQL Server, it must be installed separately by downloading it on the official Microsoft website (The PHP Group, 2021).

### 3.6.6.4 Django

Django requires the use of pip to install all the databases. However, Oracle and MySQL are already pre-installed. For Microsoft SQL Server, PostgreSQL, and MongoDB, it is necessary to install it separately (Django Software Foundation, 2021).

## 3.6.7 Performance

This segment intends to reveal the fastest among the frameworks. All the data provided for the purposes of comparison were taken from the resource Techempower Web Framework Benchmarks (TechEmpower, Inc., 2021). We can observe on Figures 20–22 how express.js leads the ranks and occupies the first place while Django, Ruby on Rails, and Laravel share the second, third, and fourth places respectively. However, Express.js has problems when it uses the mongodb database (see Figure 20).

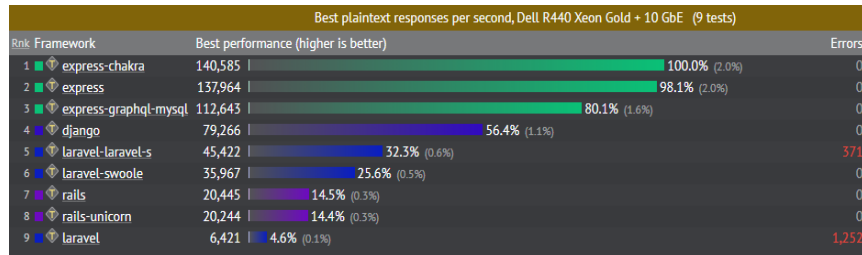


Figure 20 Best plaintext responses per second (TechEmpower, Inc., 2021)

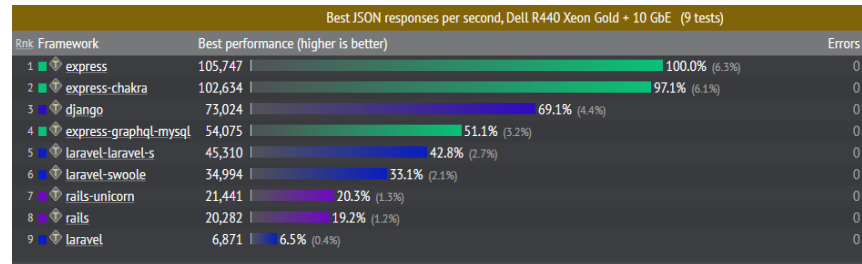


Figure 21 Best JSON responses per second (TechEmpower, Inc., 2021)

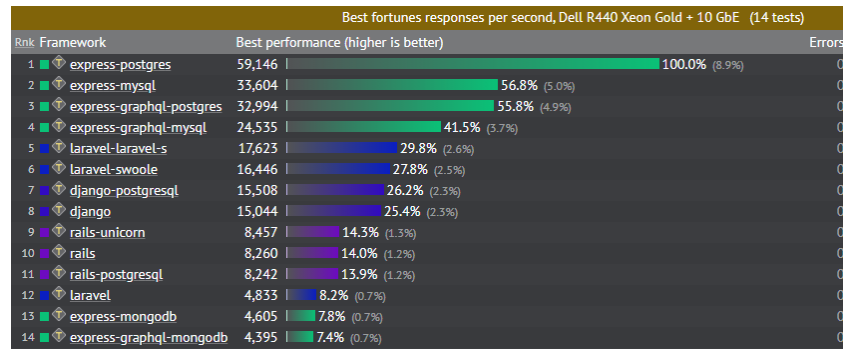


Figure 22 Best fortunes responses per second (TechEmpower, Inc., 2021)

### 3.7 Tools for web development

There are many different tools with which the developer can make his/her life easier. These tools may be necessary during development so that you do not make mistakes in future. Each developer selects his/her own tools and saves time by fully focusing on the task assigned to him/her.

#### 3.7.1 Python environment

Python environment or the venv module gives you an opportunity to create a virtual environment isolated from the system in a special folder of your choice. Each virtual environment has its own Python version. These help you, as a developer, control your Python/Django versions; hence, your project will not have conflicts with other versions of your Python/Django.

To create your virtual environment, you need to enter a special command in your terminal for Windows. Here is an example:

```
C:\Users\...\mfdw_project>python -m venv env_mfdw
```

Figure 23 (George, 2019)

### 3.7.2 Text Editor

Text editor is required to write your program. It is up to the programmer to determine the type of text editor that has to be used because it is more about preferences. You can use a simple text editor like Vim and upgrade it for your special uses, or you can use an Integrated Development Environment (IDE). I personally use PyCharm IDE because I feel comfortable using it.

*“It’s up to you what tools you use for software development. Online arguments over the best code editor or IDE are as numerous and as entertaining as arguments over the best programming language”  
(George, 2019).*

### 3.7.3 Website hosting

There are many ways to deploy a website, using different types of web hosting services. The most popular ways of web hosting for Django are: Heroku, Python Anywhere, Amazon Web Services, Microsoft Azure, and so on. In this project, Heroku will be used to deploy a project website.

Heroku is a cloud platform that is used to deploy a website (salesforce, 2021).

### 3.7.4 HTML

Hypertext Markup language (or HTML) is a markup language used to describe the structure of a webpage. No webpage can be complete without HTML. It is the main web developer tool for creating the frontend part of a website (Lewis Coulson, 2019).

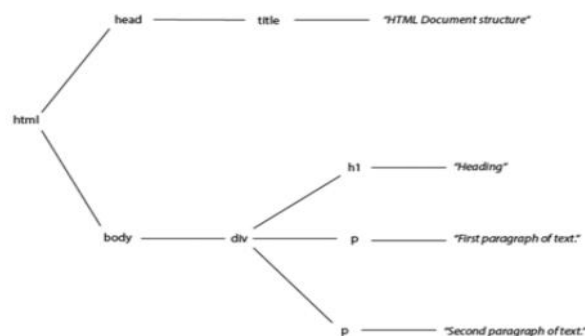


Figure 24 Representation of the HTML document as a tree diagram (Lewis Coulson, 2019)

### **3.7.5 Bootstrap**

Bootstrap is a potent front-end framework used to create modern websites and web apps. It is open-source and free to use, and has a lot of features of HTML and CSS templates for UI interface elements like buttons, navbars, and a lot of different components. Bootstrap saves time while creating a CSS and HTML features (Bootstrap team, 2021).

*“Quickly design and customize responsive mobile-first sites with Bootstrap, the world’s most popular front-end open-source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.”*

### **3.7.6 CKEditor**

CKEditor is an online WYSIWYG editor that is used to edit HTML documents (CKSource sp. z o.o. sp.k, 2021).

### **3.7.7 Pillow**

Pillow is Python Imaging Library. It is provided with Python language for image work. It is easy to use and can be downloaded using Python packet manager pip (Fredrik Lundh, 2021).

### **3.7.8 JavaScript**

JavaScript is an object-oriented language that is used in a lot of different fields like frontend, mobile development, and backend. For a frontend, it is used to create complicated and dynamic user-facing websites and cooperates with an HTML code. In a thesis project, JavaScript is responsible for the frontend part (Shute, 2019).

## **3.8 Website architecture design**

Before starting work on the creation of a website, it is necessary to provide an architecture design and UML; an Entity Relationship Diagram helps us with this task. UML helps us take an original idea to its implementation. UML can be applied in any number of ways, but is commonly used in:

- Designing Software
- Communicating software or business processes

- Capturing details about a system for requirements or analysis
- Documenting an existing system, process, or organization

As for designing software, UML gives us a different type of diagram for any level of development. Hence, to discuss requirements for the software, UML provides a Use Case diagram. For capturing what parts of the software that realize certain requirements there exist collaborations diagrams. To know and describe how parts of the system realize their requirements we have sequence and statechart diagrams. And the final Component and Deployment diagrams used to show how everything fits together and is executed (Dan Pilone, 2005).

An Entity Relationship Diagram is an excellent tool for planning and designing a database. The entity relationship gives an analyst or a developer a clear view of the data, which gives an alternative logical view of the system. It is used to design a database and to model a system's data (William S. Davis, 2019).

In this project, we will use an entity relationship diagram and a use cases diagram.

### **3.8.1 Entity Relationship Diagram**

An Entity Relationship Diagram consists of an entity that could be a person, a group, a thing, or an activity. A relationship is represented by links of those entities, and is shown by a line between them. The relationship can be represented by different styles, such as the information engineering style:

- One to one
- One to many
- Many
- One or more
- One and only one
- Zero or one
- Zero or many

It can also be represented by the Chen style, the Bachman style, and the Martin style. For project purposes, we will use the Chen style, as shown in Figure 25.



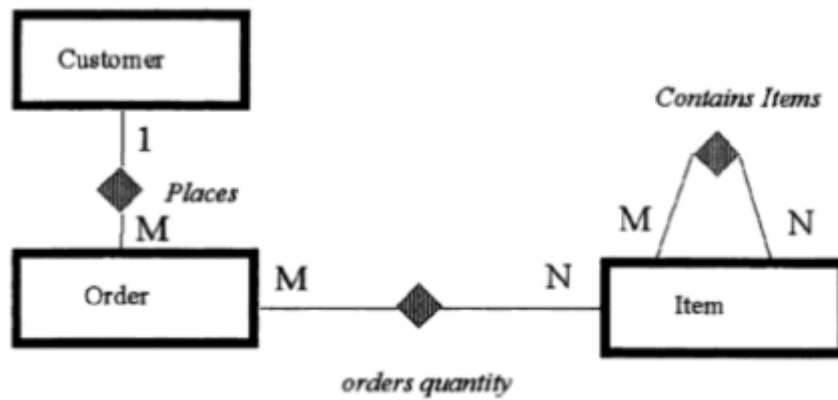


Figure 25 A sample of a Chen diagram (Wittwer, 1995)

### 3.8.2 Use Case diagram

Use Cases show a system’s functionality and requirements. Such a diagram consists of the named pieces of functionality i.e. persons who trigger an action. Example of typical Use Case diagrams consist of the actor as a “customer,” the use case as an “order item,” and an association as a line that connects the actor and the Use Case. This is shown in Figure 26.



Figure 26 Use cases example (Dan Pilone, 2005)

## 4 Practical Part

### 4.1 Website architecture design

An Entity Relationship Diagram was created and designed using the Chen style, which is described earlier in the Literature Review. Also, a Use Case diagram was created with all the attributes described earlier.

#### 4.1.1 Entity Relationship Diagram

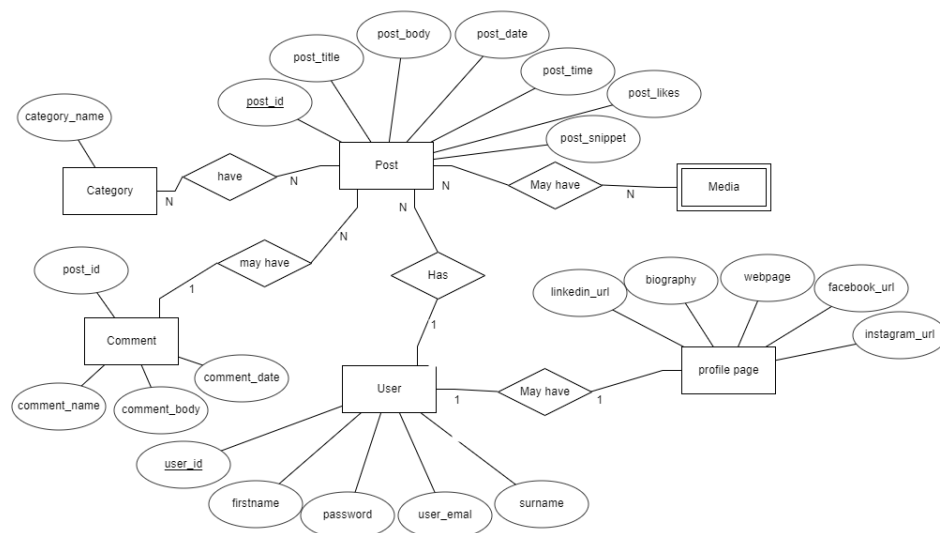


Figure 27 Class diagram

#### 4.1.2 Use Cases

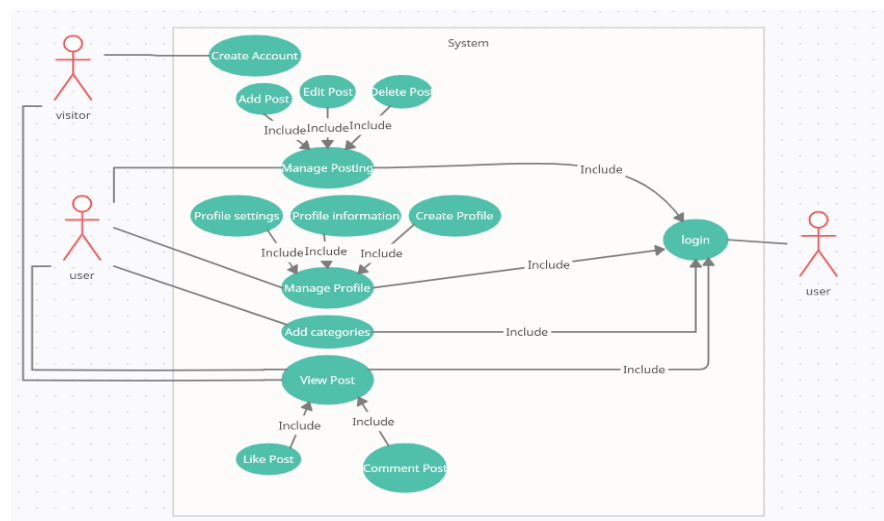


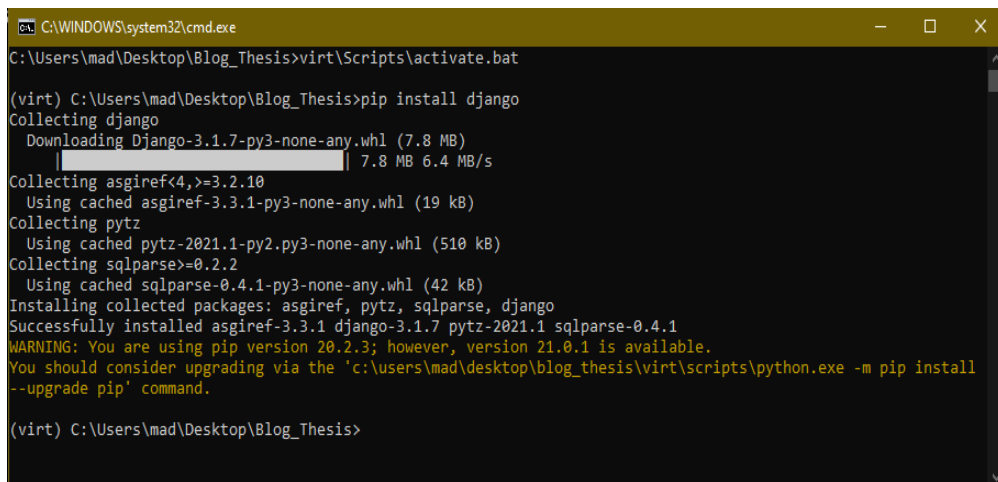
Figure 28 Use Case diagram

## 4.2 Setting up the development environment

This chapter shows the steps for setting up the development environment before actually starting the coding part. Django is known for its philosophy— “for perfectionists with deadlines”—for the development part, but it is the same when we talk about setting up the development environment. The virtual environment was deployed in Windows 10. In the next section, we will see step by step how to prepare everything to start work.

### 4.2.1 Setting up the virtual environment

The proposed website is developed using Python 3.9.1 and Django 3.1.7. Firstly, after installing Python and PyCharm, we will create a virtual environment called “venv” in a future project directory called “thesis.” Secondly, we will activate the virtual environment from the terminal by opening an “activate.bat” file. Finally, we will install Django using pip, which goes along with Python. Also, it should be noted that we will use our virtual environment all the time.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\mad\Desktop\Blog_Thesis>virt\Scripts\activate.bat

(virt) C:\Users\mad\Desktop\Blog_Thesis>pip install django
Collecting django
  Downloading Django-3.1.7-py3-none-any.whl (7.8 MB)
     |#####| 7.8 MB 6.4 MB/s
Collecting asgiref<4,>=3.2.10
  Using cached asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Installing collected packages: asgiref, pytz, sqlparse, django
Successfully installed asgiref-3.3.1 django-3.1.7 pytz-2021.1 sqlparse-0.4.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\mad\desktop\blog_thesis\virt\scripts\python.exe -m pip install
--upgrade pip' command.

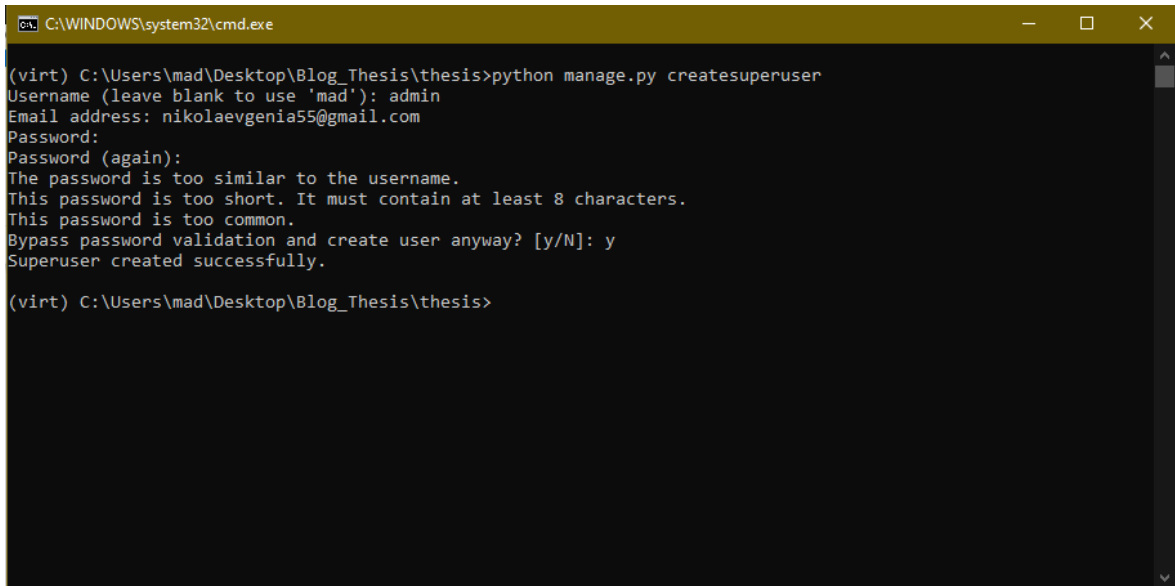
(virt) C:\Users\mad\Desktop\Blog_Thesis>
```

Figure 29 Activating the virtual environment and installing Django

### 4.2.2 Starting the project

This version of Django is the latest official version at the moment when we are working on the Bachelor’s thesis. Figure 29 demonstrates the Django version used for this thesis project.

After installing Django, the Django project—called “thesis”—was created using the command in the terminal “django-admin startproject thesis.” Also, it is necessary to create a “blog” application using the command “py manage.py startapp blog.” In the same way, we need to create a “login\_system” application. A superuser was also created (Figure 30).



```
C:\WINDOWS\system32\cmd.exe
(virt) C:\Users\mad\Desktop\Blog_Thesis\thesis>python manage.py createsuperuser
Username (leave blank to use 'mad'): admin
Email address: nikolaevgenia55@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(virt) C:\Users\mad\Desktop\Blog_Thesis\thesis>
```

Figure 30 Super user creation

Django has a standard, requiring files such as `setting.py`, `urls.py`, etc. As an addition, a template folder, files `forms.py`, and `views.py` were created in the applications.

### 4.3 Development process

This chapter will show and describe the coding part of the project. First of all, it should be noticed that in this project, we were using Bootstrap to design a webpage quickly, and make it look more beautiful. To control and add special forms of the HTML part, “forms.py” were created, which contain metadata and are integrated with Bootstrap to make different input forms look like we want them to.

#### 4.3.1 Thesis project folder

The thesis project folder is a folder with the main files of the project. Here, the main files, such as `setting.py`, are located.

##### 4.3.1.1 Settings.py

When we finished creating the applications, we need to add them to our settings file in the `INSTALLED_APPS` so that we can activate them in the project (Figure 31).

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
    'login_system',
    'ckeditor',
]

```

Figure 31 List of installed apps

### 4.3.1.2 urls.py

URLs are most common in the development of a website. Navigation in a Django website is the same as in any other website pages, so URLs were created that give us the opportunity to move to the pages we need. In Figure 32 are shown the URLs for a login page, an admin page, and a home page.

```

1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf import settings
4 from django.conf.urls.static import static
5
6 urlpatterns = [
7     path('admin/', admin.site.urls),
8     path('', include('blog.urls')),
9     path('login/', include('django.contrib.auth.urls')),
10    path('login/', include('login_system.urls')),
11
12 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Figure 32 URLs from thesis project

## 4.3.2 Blog application

This is the main application responsible for almost every part of the website. It contains:

### 4.3.2.1 Models.py

Model.py is a file where the models for application are located. It contains the essential fields and behaviors of data storing. Every time you update, delete, or make a change, it is necessary to make migrations for the database (Figure 29).

```
C:\WINDOWS\system32\cmd.exe - python manage.py runserver

(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>python manage.py makemigrations
Migrations for 'blog':
  blog\migrations\0001_initial.py
  - Create model Post

(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying blog.0001_initial... OK
```

Figure 33 Applying changes in model.py

At class, Post from model.py created a list of fields that was used in a project. The list of fields is showed in Figure 34. Also, we can see an added text editor from CKEditor package to the body section for our post.

```
class Post(models.Model):
    title = models.CharField(max_length=255)
    header_image = models.ImageField(null=True, blank=True, upload_to="images/")
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    body = RichTextField(blank=True, null=True)
    post_date = models.DateField(auto_now_add=True)
    post_time = models.TimeField(auto_now_add=True)
    category = models.CharField(max_length=255, default='django')
    snippet = models.CharField(max_length=255)
    likes = models.ManyToManyField(User, related_name='blog_post')

    def total_likes(self):
        return self.likes.count()

    def __str__(self):
        return self.title + ' | ' + str(self.author)

    def get_absolute_url(self):
        return reverse('home')
```

Figure 34 Class post from models.py

### 4.3.2.2 Views.py

Views take data information from the database and return them to a template. Views have two types—class-based views and function-based views. For each Django view, a special function is executed and has an inherited template. There are several class-based views that exist from Django generic display views, as were used in the project:

- ListView—displays a list of data objects
- DetailView—displays a single object
- DeleteView—displays a confirmation page and deletes an existing object
- CreateView—displays a form for creating an object
- UpdateView—displays a form for editing an existing project

Here is a list of classes and functions, written in this views file for a blog app:

- class HomeView
- class ArticleDetailView

- class AddPostView
- class AddCommentView
- class UpdatePostView
- class DeletePostView
- class AddCategoryView
- def category\_view
- def category\_list\_view
- def like\_view

As an example of a class-based view, we have “class HomeView,” as shown in Figure 31. HomeView takes a class Post from models.py, assigns a template called home.html, and filter posts on a home page in descending order by date and time. The function get\_context\_data grabs everything from the category class model and signs it to a variable cat\_menu. Then we push that variable to the context dictionary, signing cat\_menu to the context and returning it. We can now access cat\_menu on a home page.

```
class HomeView(ListView):
    model = Post
    template_name = 'home.html'
    ordering = ['-post_date', '-post_time']

    def get_context_data(self, *args, **kwargs):
        cat_menu = Category.objects.all()
        context = super(HomeView, self).get_context_data(*args, **kwargs)
        context["cat_menu"] = cat_menu
        return context
```

Figure 35 Class example from view.py

### 4.3.2.3 Templates

The Template folder contains every HTML file created for the project purpose. These HTML files describe a page structure. However, it does not keep only an HTML code, which has some JavaScript code and Django special template language. An example of an HTML file is shown in Figure 36. Django template language allows us to use the statement, loops, etc. inside the HTML code.

```

{% extends 'base.html' %}
{% block content %}
{% if user.is_authenticated %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<h1> Add category</h1>
  <div class="form-group">
    <form method="POST">
      {{ csrf_token }}
      {{ form.as_p }}
      <button class="btn btn-secondary">Add category</button>
    </form>
  </div>
</body>
</html>
{% endif %}
{% endblock content %}

```

Figure 36 add\_category html file

A total of 17 html files were created for the project; most of them are similar to each other. However, there are HTML files that should be specially noted. Base.html is an html file that contains a template from bootstrap exactly from that file; all the other html files inherit the components of the bootstrap template.

#### 4.3.2.4 Forms.py

Forms is an important part of the project. Forms is similar to the ModelForm class, which creates a form by using a model, but it does not require a Model. Figure 37 demonstrates the PostForm fields that are taken from Post Models (Figure 34). Also, fields can be taken selectively. Some of a field could be hidden, such as the author form.

```

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ('title', 'author', 'category', 'body', 'snippet', 'header_image')

    widgets = {
        'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Write a title here'}),
        'author': forms.TextInput(attrs={'class': 'form-control', 'value': '', 'id': 'user_id', 'type': 'hidden'}),
        'category': forms.Select(choices=choice_list, attrs={'class': 'form-control'}),
        'body': forms.Textarea(attrs={'class': 'form-control', 'placeholder': 'Write your text here'}),
        'snippet': forms.Textarea(attrs={'class': 'form-control'}),
    }

```

Figure 37 PostForm from forms.py

Here, in an author's widget, was added a value with pure space—an ID that was called "user\_id." The type of the widget was set as hidden. It was made to automatically set an author when creating a post. A script written using JavaScript is shown in Figure 38, creating a variable name. It is assigned to a user id value; it finds an ID by an id name "user\_id" and assigns it to a name which is an actual user ID.

```

<script>
  let name = "{ { user.id } }"
  document.getElementById("user_id").value = name;
</script>

```

Figure 38 Script from add\_post.html



### 4.3.2.5 Admin.py

The admin file was created to register all models in a Django site administration. After importing them into the admin site, it is possible to see and change every comment, category, post, and profile (Figures 39–40).

```
from django.contrib import admin
from .models import Post, Category, Profile, Comment

admin.site.register(Post)
admin.site.register(Category)
admin.site.register(Profile)
admin.site.register(Comment)
```

Figure 39 admin.py

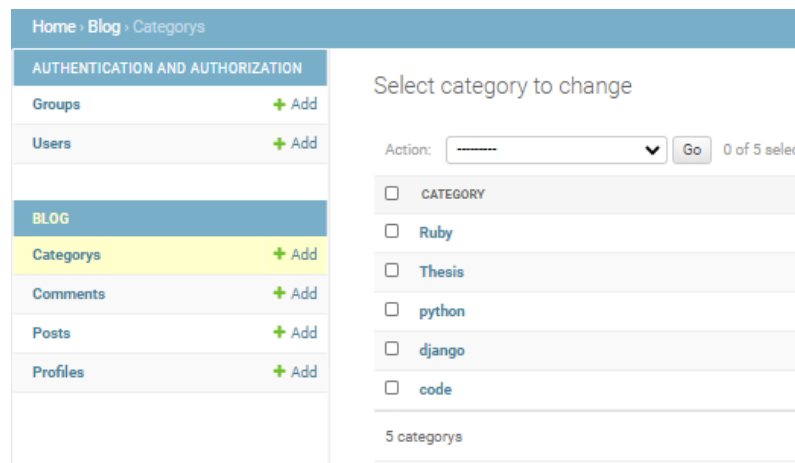


Figure 40 Django administration page

### 4.3.3 Login system

The login system application contains views, URLs, templates, and form files. This application is responsible for creating the login-logout systems for users. Also, it gives users the opportunity to create a profile page, where users can write something about themselves. Basically, the login system app complements the main blog app.

## 4.4 Deploying a website

To deploy a website project using Heroku, we need to install git and Heroku Command Line Interface (CLI) from the Heroku official website. There are three different ways to host a website:

- To deploy it through Heroku git CLI,
- Deploy it using git, or
- Deploy it using Container Registry.

In our project, we used the first variant of deployment.

#### 4.4.1 Deployment using Heroku git

First of all, we need to install two python packages—whitenoise and gunicorn—using command “pip install whitenoise gunicorn.” After the installation, we need to create a text file with a list of all the packages we use. To do so, we need to write a command “pip freeze > requirements.txt.” It is necessary to call that text file a “requirements” file, because Heroku needs to find all the packages to install them on the server. Our next step would be to create a “runtime.txt” file with writhed Python version used in a project. Also, a file should be created without any extension, called “Procfile.” Inside it, we will have to write “web: gunicorn thesis.wsgi --log-file -.” Finally, we need to add to the settings file an allowed host (Figure 41).

```
ALLOWED_HOSTS = ['thesis-madwizard.herokuapp.com', '127.0.0.1']
```

Figure 41 Allowed host

We need to add a static root with a static URL, as is shown in Figure 42.

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')  
STATIC_URL = '/static/'
```

Figure 42 Static root

Everything is ready now. We can start deploying the project website. It is necessary to create a Heroku git repository using the “heroku create” command, or just create it on an official Heroku webpage. In my project, it was already created; so we just need to log in to the Heroku, set an existing Heroku git repository, telling git that we want to include updates to a file in the next commit. Finally, we make a commit, as is shown in Figure 43.

```
(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>pip freeze > requirements.txt  
(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>heroku login  
heroku: Press any key to open up the browser to login or q to exit:  
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/886f0717-a332-4f96-9914-982b967c0bb4?requestor=SFMyNzY2QAA4xNDcuMzIuMTA3LjIzN24GAEMuS053AWIAAVGA.yyxIwvs5vOAKUfzJJdwG6KHyyIkUqI1M101Yqdm7cw  
Logging in... done  
Logged in as nikolaevgenia55@gmail.com  
(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>heroku git:remote -a thesis-madwizard  
set git remote heroku to https://git.heroku.com/thesis-madwizard.git  
(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>git add .  
(venv) C:\Users\mad\Desktop\Blog_Thesis\thesis>git commit -am "heroku push"  
[master c80e642] heroku push  
35 files changed, 1427 insertions(+)  
create mode 100644 venv\Lib\site-packages\whitenoise-5.2.0.dist-info\INSTALLER  
create mode 100644 venv\Lib\site-packages\whitenoise-5.2.0.dist-info\LICENSE  
create mode 100644 venv\Lib\site-packages\whitenoise-5.2.0.dist-info\METADATA  
create mode 100644 venv\Lib\site-packages\whitenoise-5.2.0.dist-info\RECORD
```

Figure 43 Deploying a website

## **5 Results and Discussion**

The results of this Bachelor's thesis compare four server-side frameworks and the fully designed and working website, which can be accessed through the Heroku cloud hosting platform "thesis-madwizard.herokuapp.com."

Django, Ruby on Rail, Laravel, and Express.js were compared, using a lot of websites with a lot of data about them, such as the number of packages and the number of requests per second that the framework can provide, books, and official documentations.

The website has a login possibility, registration, and reading articles. It shows the category list, shows articles by categories, adds a category, creates a profile page, edits profile settings, and updates the information of the user.

The source code for the website project is attached to this thesis. It consists of the main project and two applications for that project. The first application, called the "blog," is responsible for the main functionality of the website and contains all models. The second application, called the "login system," is responsible for login, logout, and user's profile.

For the development of this website, a lot of different tools were used. It is powered by the Python programming language, has a Bootstrap template as a main base for all HTML files, and a bunch of packages like CKEditor that helped make the website better.

After developing the website, it was hosted on a Heroku cloud hosting platform, which allows it to be available all the time.

## **6 Conclusion**

This thesis was meant to design and develop a website using the Django web framework. It was supposed to show that even though Django is not that popular, it deserves to be noticed. The thesis was expected to show the differences with other frameworks; these were shown in a literature review. The literature review explained the basics of Django and Python. Also, other frameworks were described, so that we could see the differences with other frameworks, based on the different programming languages. As an addition to the literature review was described the website architecture design and a brief history of the web development. In the last parts of the literature review were described all the technologies and packages used in the project.

To show how the technologies and tools can be used, the practical part focused on the development and deployment of the website. It shows how tools can be added and a website can be created using only Python and Django for the backend part of the project and HTML, bootstrap, and JavaScript for the frontend part. The practical part contains a step-by-step way to design a website, develop it, and deploy it, while describing each file in a closer way.

There is no doubt that Django justifies its advantages in rapid website development.

## 7 References

- Bini, Ola. 2007.** *Practical JRuby on Rails Web 2.0 Projects: Bringing Ruby on Rails to Java*. New York : Apress, 2007. 1430204192.
- Boggiano, Jordi. 2021.** Packagist The PHP Package Repository. *Packagist*. [Online] 2021. <https://packagist.org/>.
- Bootstrap team . 2021.** Build fast, responsive sites with Bootstrap. *getbootstrap*. [Online] Bootstrap team, 2021. [Cited: January 12, 2021.] <https://getbootstrap.com/>.
- Brown, Ethan. 2019.** *Web Development with Node and Express: Leveraging the JavaScript Stack*. Sebastopol: O'Reilly Media, Inc, 2019. 1492053481.
- BuiltWith Pty Ltd. 2020.** Django Language Usage Statistics. *builtwith*. [Online] 2020. <https://trends.builtwith.com/framework/Django-Language>.
- . 2020. Express Usage Statistics. *Builtwith*. [Online] 2020. <https://trends.builtwith.com/framework/Express>.
- . 2020. Laravel Usage Statistics. *Builtwith*. [Online] 2020. <https://trends.builtwith.com/framework/Laravel>.
- . 2020. Ruby on Rails Usage Statistics. *BuiltWith*. [Online] 2020. <https://trends.builtwith.com/framework/Ruby-on-Rails>.
- CERN. 2021.** A short history of the Web. *home.cern*. [Online] CERN, 2021. <https://home.cern/science/computing/birth-web/short-history-web>.
- CKSource sp. z o.o. sp.k. 2021.** CKEditor Ecosystem. *ckeditor*. [Online] CKSource sp. z o.o. sp.k, 2021. <https://ckeditor.com/>.
- Dan Pilone, Neil Pitman. 2005.** *UML 2.0 in a Nutshell*. Sebastopol: O'Reilly Media, Inc, 2005. 0596007957.
- DeBill, Erik. 2020.** Module Counts. *Modulecounts*. [Online] 2020. <http://www.modulecounts.com/>.
- Django Software Foundation. 2021.** Databases. *Django documentation*. [Online] Django Software Foundation, 2021. <https://docs.djangoproject.com/en/3.1/ref/databases/>.
- Foundation, Django Software. 2021.** Documentation. *Django*. [Online] Django Software Foundation, 2021. <https://docs.djangoproject.com/en/3.1/intro/tutorial01/>.
- Foundation, OpenJS. 2017.** Glossary. *Express*. [Online] OpenJS Foundation, 2017. <https://expressjs.com/en/resources/glossary.html>.
- . 2021. Hello world example. *Express “Hello World” example*. [Online] OpenJS Foundation, 2021. <https://expressjs.com/en/starter/hello-world.html>.
- Fredrik Lundh, Alex Clark. 2021.** Pillow 8.1.0. *PyPi*. [Online] 2021. <https://pypi.org/project/Pillow/>.
- George, Nigel. 2019.** *Build a website with Django 3 a complete introduction to django 3*. Hamilton: GNW Independent Publishing, 2019. 978-0-9946168-9-0.
- Hal Fulton, André Arko. 2015.** *The Ruby Way: Solutions and Techniques in Ruby Programming. Looping and Branching*. s.l.: Addison-Wesley Professional, 2015.
- Laravel. 2021.** Installation. *Laravel*. [Online] 2021. <https://laravel.com/docs/8.x/installation>.
- Lewis Coulson, Brett Jephson, Rob Larsen, Matt Park, Marian Zburlea. 2019.** *Introduction to HTML and CSS. The The HTML and CSS Workshop: Learn to build your own websites and kickstart your career as a web designer or developer*. Birmingham: Packt Publishing Ltd, 2019.

**Lutz, Mark. 2013.** *Learning Python: Powerful Object-Oriented Programming*. Sebastopol: O'Reilly Media, Inc., 2013. 1449355714.

**McCool, Shawn. 2012.** *Laravel Starter*. Birmingham : Packt Publishing Ltd, 2012. 1782160914.

*model view controller architecture.* **Deacon, John. 2009.** May 2009.

**Mozilla. 2020.** MVC. *developer.mozilla*. [Online] Mozilla, 2020. <https://developer.mozilla.org/en-US/docs/Glossary/MVC>.

**npm, Inc. 2021.** npm. *Build amazing things*. [Online] 2021. <https://www.npmjs.com/>.

**OpenJS Foundation. 2021.** Database integration. *Express js*. [Online] OpenJS Foundation, 2021. <https://expressjs.com/en/guide/database-integration.html>.

**Python Software Foundation. 2021.** PyPI—the Python Package Index. *PyPI—the Python Package Index*. [Online] Python Software Foundation, 2021. <https://pypi.org/>.

**Ruby on Rails. 2021.** Getting Started with Rails. *Rails Guides*. [Online] 2021. [Cited: January 20, 2021.] [https://guides.rubyonrails.org/getting\\_started.html](https://guides.rubyonrails.org/getting_started.html).

**RubyGems. 2021.** Find, install, and publish. *RubyGems*. [Online] 2021. <https://rubygems.org/>.

**salesforce. 2021.** What. *Heroku*. [Online] salesforce.com, inc, 2021. <https://www.heroku.com/what>.

**Shute, Zachary. 2019.** *Advanced JavaScript: Speed up web development with the powerful features and benefits of JavaScript*. s.l.: Packt Publishing Ltd, 2019. 1789803896.

**Solid IT GmbH. 2021.** DB-Engines Ranking. *DB-Engines*. [Online] Solid IT GmbH, February 2021. <https://db-engines.com/en/ranking>.

**Stack Exchange. 2020.** Developer Survey. *stackoverflow*. [Online] Stack Exchange Inc, 2020. <https://insights.stackoverflow.com/survey/2020/#technology>.

**Stack Exchange Inc. 2018.** Developer Survey Results 2018. *stackoverflow*. [Online] Stack Exchange, 2018. <https://insights.stackoverflow.com/survey/2018/#technology>.

**TechEmpower, Inc. 2021.** Round 20. *Techempower Web Framework Benchmarks*. [Online] TechEmpower, Inc., February 8, 2021. [Cited: February 15, 2021.] <https://www.techempower.com/benchmarks/#section=data-r20>.

**The PHP Group. 2021.** Database Extensions. *PHP*. [Online] The PHP Group, 2021. <https://www.php.net/manual/en/refs.database.php>.

**w3schools. 2021.** JavaScript For Loop. *w3schools*. [Online] 2021. [https://www.w3schools.com/js/js\\_loop\\_for.asp](https://www.w3schools.com/js/js_loop_for.asp).

—. 2021. PHP for Loop. *w3schools*. [Online] 2021. [https://www.w3schools.com/php/php\\_looping\\_for.asp](https://www.w3schools.com/php/php_looping_for.asp).

**William S. Davis, David C. Yen. 2019.** *The Information System Consultant's Handbook: Systems Analysis and Design*. Ohio : CRC Press, 2019. 1420049100.

**Wittwer, Sylvan H. 1995.** *Food, Climate, and Carbon Dioxide: The Global Environment and World Food Production*. Boca Raton: CRC Press, 1995. 0873717961.

## **8 Appendix**

List of Supplements...