1862

RIGA TECHNICAL UNIVERSITY

Benedikt Görgei
Studnet ID: 220AEM115

Telecommunication Software
Practical Exercise 5 (Python cloud full stack development)

submitted to

Riga Technical University
Faculty of Electronics and Telecommunications

Supervisor
Tianhua Chen

Riga, January 8, 2023

Contents

1 Introduction

This is a university exercise report for the course ”Telecommunication Software RAE411”. The exercises considers the principals of full stack development with the python Django framework. The source code is available at `https://github.com/Bennys-Quarter/DjangoWebSite.git`

2 Example: Hello World

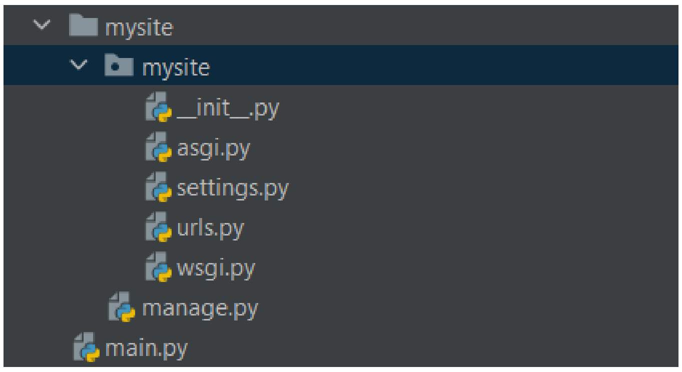
This section explains how to get started with django and creates a simple ”Hello World!” webpage. Django’s primary goal is to ease the creation of complex, database-driven websites. The framework can be easily installed with the pip package manager.

```
python -m pip install Django
```

Before starting to program the website it is recommended to create a initial folder structure.

```
django-admin startproject mysite
```

The newly created mysite Django project has the following structure:



```
mysite
    mysite
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
    manage.py
main.py
```

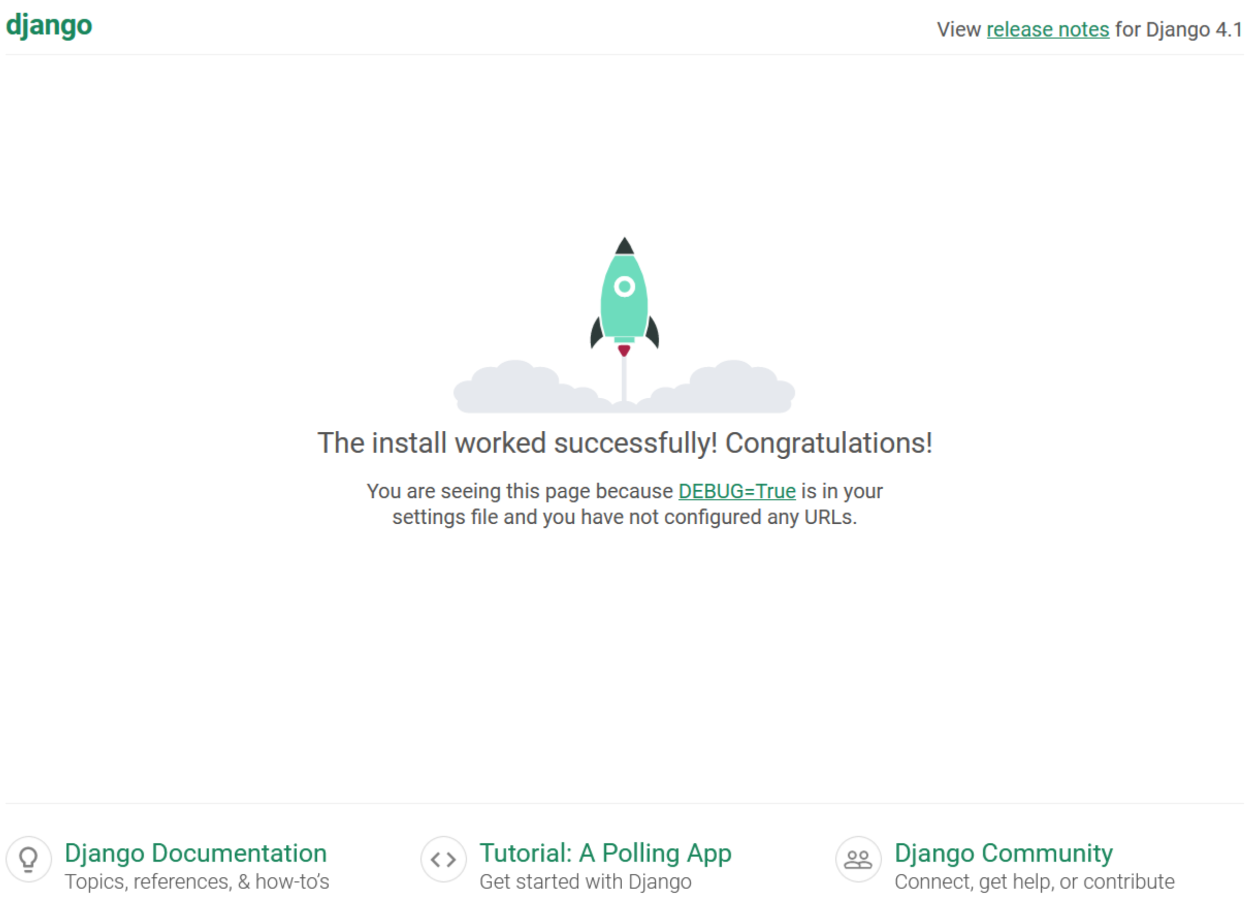
Figure 1: Initial folder structure

After changing to the outer mysite folder the initial webpage server can be started with:

```
python manage.py runserver
```

The shows that the server starts on localhost on port 8000.

```
January 06, 2023 - 23:37:49
Django version 4.1.5, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK
```


django
View release notes for Django 4.1

The install worked successfully! Congratulations!

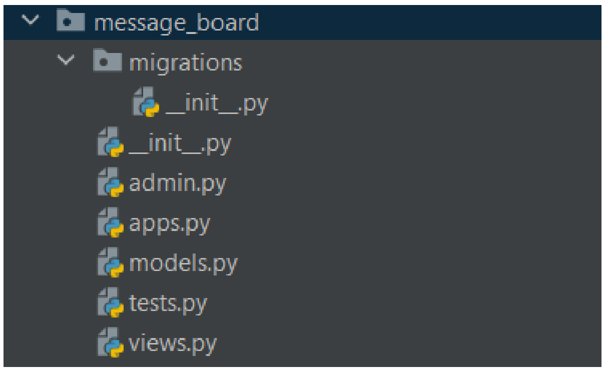
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

Figure 2: Initial Webpage

To venture further, an app is created with the name *message_board*. A folder is created with the same name and necessary python scripts.

```
python manage.py startapp message_board
```



```
message_board
    migrations
        __init__.py
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
```

Figure 3: App folder structure

The frontend webpages that the user can see and interact with are implemented in views.py script.

```
from django.shortcuts import render
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world!")
```

These views can be accessed through the correct url path. These have to be defined in the urls.py scripts.

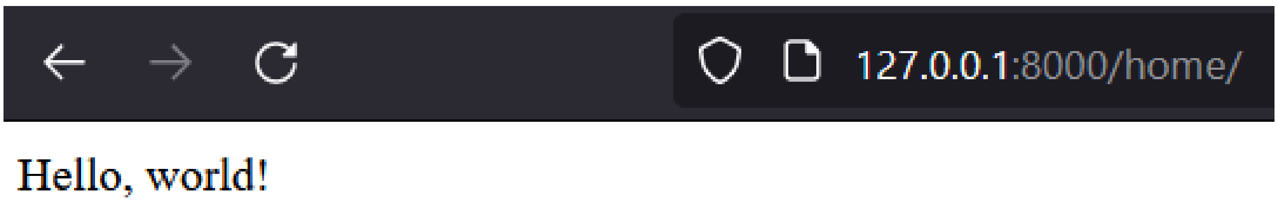
In the app:

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

included in the about project urls.py:

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

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

New view accessible at `http://127.0.0.1:8000/home/`

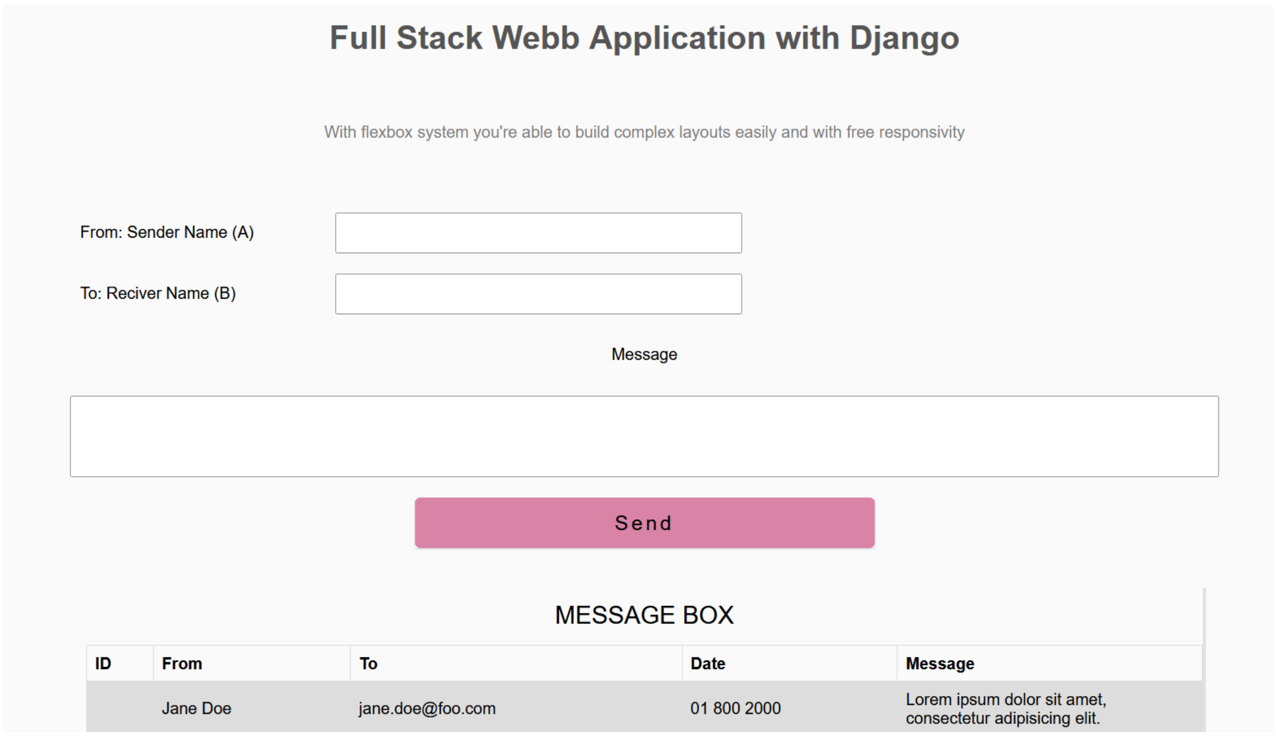


Hello, world!

Figure 4: Hello World!

3 Example: Webpage Design

The design of the webpage uses html and css files and uses the Model View Template (MVT) of Django.



Full Stack Webb Application with Django

With flexbox system you're able to build complex layouts easily and with free responsivity

From: Sender Name (A)

To: Reciver Name (B)

Message

Send

MESSAGE BOX

ID	From	To	Date	Message
	Jane Doe	jane.doe@foo.com	01 800 2000	Lorem ipsum dolor sit amet, consectetur adipisicing elit.

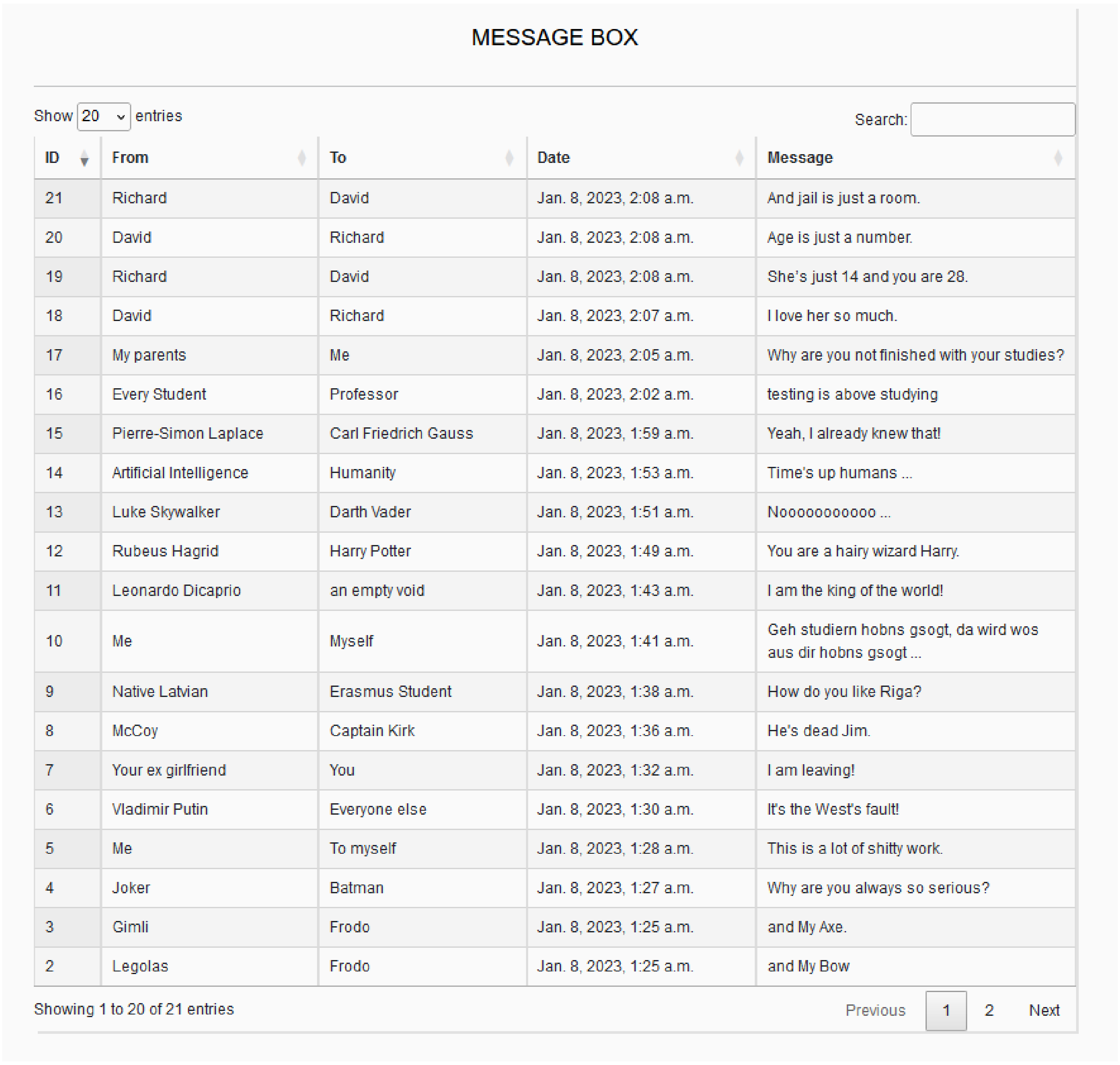
Figure 5: Webpage basic design

```
def index(request):
    form = NameForm()
    if request.method == "POST":
        form = NameForm(request.POST)
        if form.is_valid():
            print(request.POST)
            print(form['name_a'].value())
            print(form['name_b'].value())
            print(form['msg'].value())
            new_entry = Entry(name_A=str(form['name_a'].value()),
                              name_B=str(form['name_b'].value()),
                              pub_date=datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
                              message=str(form['msg'].value()))
            new_entry.save()
    messages = Entry.objects.all()
    context = {'m_table': messages, }
    return render(request, 'message_board/index.html', context)
```

The MVT is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data.

The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template. `https://www.javatpoint.com/django-mvt`

The views are implemented in the views.py file. To interact with the webpage input values need to be passed to the SQL database. Models manage the access to the database in Django and Forms are used to get data from a POST request.



MESSAGE BOX

ID	From	To	Date	Message
21	Richard	David	Jan. 8, 2023, 2:08 a.m.	And jail is just a room.
20	David	Richard	Jan. 8, 2023, 2:08 a.m.	Age is just a number.
19	Richard	David	Jan. 8, 2023, 2:08 a.m.	She's just 14 and you are 28.
18	David	Richard	Jan. 8, 2023, 2:07 a.m.	I love her so much.
17	My parents	Me	Jan. 8, 2023, 2:05 a.m.	Why are you not finished with your studies?
16	Every Student	Professor	Jan. 8, 2023, 2:02 a.m.	testing is above studying
15	Pierre-Simon Laplace	Carl Friedrich Gauss	Jan. 8, 2023, 1:59 a.m.	Yeah, I already knew that!
14	Artificial Intelligence	Humanity	Jan. 8, 2023, 1:53 a.m.	Time's up humans ...
13	Luke Skywalker	Darth Vader	Jan. 8, 2023, 1:51 a.m.	Nooooooooooo ...
12	Rubeus Hagrid	Harry Potter	Jan. 8, 2023, 1:49 a.m.	You are a hairy wizard Harry.
11	Leonardo Dicaprio	an empty void	Jan. 8, 2023, 1:43 a.m.	I am the king of the world!
10	Me	Myself	Jan. 8, 2023, 1:41 a.m.	Geh studiern hobns gsogt, da wird wos aus dir hobns gsogt ...
9	Native Latvian	Erasmus Student	Jan. 8, 2023, 1:38 a.m.	How do you like Riga?
8	McCoy	Captain Kirk	Jan. 8, 2023, 1:36 a.m.	He's dead Jim.
7	Your ex girlfriend	You	Jan. 8, 2023, 1:32 a.m.	I am leaving!
6	Vladimir Putin	Everyone else	Jan. 8, 2023, 1:30 a.m.	It's the West's fault!
5	Me	To myself	Jan. 8, 2023, 1:28 a.m.	This is a lot of shitty work.
4	Joker	Batman	Jan. 8, 2023, 1:27 a.m.	Why are you always so serious?
3	Gimli	Frodo	Jan. 8, 2023, 1:25 a.m.	and My Axe.
2	Legolas	Frodo	Jan. 8, 2023, 1:25 a.m.	and My Bow

Showing 1 to 20 of 21 entries

Figure 6: Message Box with the 20 most recent messages

models.py

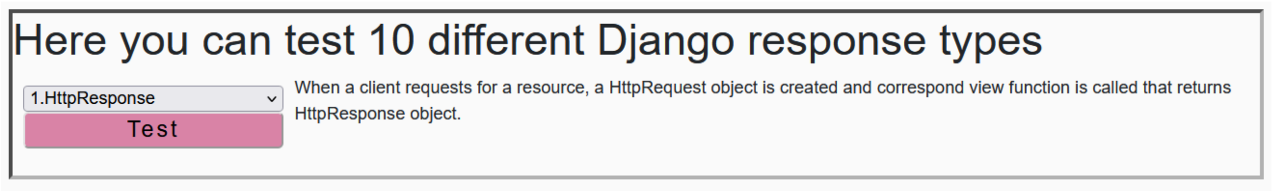
```
class Entry(models.Model):
    name_A = models.CharField(max_length=50)
    name_B = models.CharField(max_length=50)
    pub_date = models.DateTimeField(blank=True, null=True)
    message = models.CharField(max_length=500)

    def __str__(self):
        return self.name_A + " " + self.pub_date
```

forms.py

```
class NameForm(forms.Form):
    name_a = forms.CharField(label='your_name', max_length=100)
    name_b = forms.CharField(label='receiver_name', max_length=100)
    msg = forms.CharField(label="message_name", max_length=500, widget=forms.TextInput(
    attrs={'placeholder': 'Name', 'style': 'width: 300px;'}))
```

4 Example: Response Types

On the webpage is a section to explore different response types.



Here you can test 10 different Django response types

When a client requests for a resource, a HttpRequest object is created and correspond view function is called that returns HttpResponse object.

Figure 7: Section to choose a response

When a page is requested, Django creates an HttpRequest object that contains metadata about the request. Then Django loads the appropriate view, passing the HttpRequest as the first argument to the view function. Each view is responsible for returning an HttpResponse object.

10 Different Responses are implemented on the Website:

- HttpRequest
  When a client requests for a resource, a HttpRequest object is created and correspond view function is called that returns HttpResponse object.
- HttpResponseDirect
  HttpResponseRedirect is a subclass of HttpResponse (source code) in the Django web framework that returns the HTTP 302 status code, indicating the URL resource

was found but temporarily moved to a different URL. This class is most frequently used as a return object from a Django view.

- HttpResponseNotFound
  Acts just like HttpResponse but uses a 404 status code.
- HttpResponseForbidden
  Acts just like HttpResponse but uses a 403 status code.
- HttpResponseServerError
  Acts just like HttpResponse but uses a 500 status code.
- JsonRespons
  An HttpResponse subclass that helps to create a JSON-encoded response.
- StreamingHttpResponse
  The StreamingHttpResponse class is used to stream a response from Django to the browser. StreamingHttpResponse should only be used in situations where it is absolutely required that the whole content isn't iterated before transferring the data to the client. Because the content can't be accessed, many middleware can't function normally. For example the ETag and Content-Length headers can't be generated for streaming responses.
- FileResponse
  FileResponse is a subclass of StreamingHttpResponse optimized for binary files. It uses wsgi.file_wrapper if provided by the wsgi server, otherwise it streams the file out in small chunks.
- HttpResponseGone
  Acts just like HttpResponse but uses a 410 status code.
- HttpResponseNotModified
  The constructor doesn't take any arguments and no content should be added to this response. Use this to designate that a page hasn't been modified since the user's last request (status code 304).

5 Conclusion

Django is a powerful backend tool that may be chosen as framework for full stack development. Strong community support and documenting makes it attractive. Simple Websides can be created in a matter of days, however styling and fronted programming for details and functionality is still required. Django is not a silver bullet, but simple enough to grasp the fundamental functionalities quickly and complex enough to allow larger and more involved projects.