Согласовано:
Гапанюк Ю.Е.

"__"_____2016 г.

Утверждаю:
Гапанюк Ю.Е.

«__»__

Лабораторная работа №6 по курсу

Разработка интернет приложений

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-53
Семенова Е.В.

_____

"18" ноября 2016г.

Москва - 2016

Задание и порядок выполнения

В этой лабораторной работе вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также вам нужно будет дополнить свои классы предметной области, связав их с созданной базой. После этого вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей и ClassBasedViews.

Для сдачи вы должны иметь:

1. Скрипт с подключением к БД и несколькими запросами.
2. Набор классов вашей предметной области с привязкой к СУБД (класс должен уметь хотя бы получать нужные записи из БД и преобразовывать их в объекты этого класса)
3. Модели вашей предметной области
4. View для отображения списка ваших сущностей

Отчет по лабораторной работе№5 «Шаблонизация»

«urls.py»

```
from django.conf.urls import url
from django.contrib import admin
from my_app.views import index
from my_app.views import post
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url( r'^post/([0-9]+)/$', post, name = 'post'),
    url( r'', index, name = 'index'),
]
```

«views.py»

```
from django.shortcuts import render
from .models import User, Bank, Transaction
from django.http import HttpResponse
def index(request):
    banks = Bank.objects.all()
    return render(request, "index.html", {'banks': banks})
def post(request, id):
    bank = Bank.objects.get(id=id)
    transactions = Transaction.objects.select_related('user').filter(bank=bank)
    return render(request, "post.html", {'bank': bank, 'transactions':
transactions})
```

Шаблоны

«_base.html»

```
<!DOCTYPE html>
<!-- saved from url=(0050)http://getbootstrap.com/examples/starter-template/
-->
<html lang="en"><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head
content must come *after* these tags -->
```

```
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Index</title>
    <!-- Bootstrap core CSS -->
    {% load static %}
    <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
    {% load static %}
    <!-- Custom styles for this template -->
    <link href="{% static 'css/starter-template.css' %}" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container">
        <div class="navbar-header">
          {%  url 'index' as main_url %}
          <a class="navbar-brand" href="{{ main_url }}">Posts</a>
        </div>
      </div>
    </nav>
    <div class="container">
        {% block list %}
        {% endblock %}
    </div><!-- /.container -->
</body></html>
```

«_list_item.html»

```
<div class="jumbotron">
    <h2>
        {{ bank.name }}
        <br>
        <small>{{  bank.address }}</small>
        <br>
        <br>
        {%  with id=bank.id %}
        <a class="btn btn-lg btn-primary" href="{%  url 'post' id %}"
role="button">View</a>
        {% endwith %}
    </h2>
</div>
```

«index.html»

```
{% extends '_base.html' %}
{% block list %}
    <br>
    <div class="row">
        {%  for bank in banks %}
            <div class="col-lg-4">
                {% include '_list_item.html' %}
            </div>
        {%  endfor %}
    </div>
{%  endblock %}
```

«post.html»

```
{% extends '_base.html' %}
{%  block list %}
    <h1>{{ bank.name }}</h1>
    {%  for transaction in transactions %}
        <div class="jumbotron">
            <h2>
                <small>
                    {{ transaction.type }}
                    {{ transaction.count }}
                    <br>
                    {{ transaction.user.last_name }}
                    {{ transaction.user.first_name }}
                    {{ transaction.user.middle_name }}
                    {{ transaclion.user.document_number }}
                </small>
                <br>
                <br>
                {%  if transaction.count < 50000 %}
                    <span class="label label-success">Ordinary</span>
                {% elif transaction.count < 500000 %}
                    <span class="label label-warning">Important</span>
                {%  else %}
                    <span class="label label-danger">Very Important</span>
                {% endif %}
            </h2>
        </div>
    {%  endfor %}
{%  endblock %}
```

«models.py»

```
from django.db import models
class User(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    middle_name = models.CharField(max_length=50)
    document_number = models.IntegerField()
    birthday = models.DateField()
class Bank(models.Model):
    name = models.CharField(max_length=70)
    address = models.CharField(max_length=100)
class Transaction(models.Model):
    user = models.ForeignKey('User')
    bank = models.ForeignKey('Bank')
    type = models.CharField(max_length=60)
    count = models.IntegerField()
```

«test_data.py»

```
import MySQLdb
class Connection:
    def __init__(self, user, password, db, host='localhost'):
        self.user = user
        self.host = host
        self.password = password
        self.db = db
```

```
        self._connection = None
    @property
    def connection(self):
        return self._connection
    def __enter__(self):
        self.connect()
    def __exit__(self, exc_type, exc_val, exc_tb):
        self.disconnect()
    def connect(self):
        if not self._connection:
            self._connection = MySQLdb.connect(
                host=self.host,
                user=self.user,
                passwd=self.password,
                db=self.db,
                charset = "utf8"
            )
    def disconnect(self):
        if self._connection:
            self._connection.close()
class User:
    def __init__(self, db_connection, first_name, last_name, middle_name,
document_number, birthday):
        self.db_connection = db_connection.connection
        self.first_name = first_name
        self.last_name = last_name
        self.middle_name = middle_name
        self.document_number = document_number
        self.birthday = birthday
    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO my_app_user (first_name, last_name, middle_name,
document_number, birthday) VALUES (%s, %s, %s, %s, %s);",
                  (self.first_name, self.last_name, self.middle_name,
self.document_number, self.birthday))
        self.db_connection.commit()
        c.close()
    def get(self):
        c = self.db_connection.cursor()
        c.execute("SELECT * FROM my_app_user;")
        users = []
        for row in c.fetchall():
            u = User
            u.first_name = row[1]
            u.last_name = row[2]
            u.middle_name = row[3]
            u.document_number = row[4]
            u.birthday = row[5]
            users.append(u)
        return users
class Bank:
    def __init__(self, db_connection, name, address):
        self.db_connection = db_connection.connection
        self.name = name
        self.address = address
    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO my_app_bank (name, address) VALUES (%s, %s);",
                  (self.name, self.address))
        self.db_connection.commit()
        c.close()
class Transaction:
```

```
    def __init__(self, db_connection, type, count, usr_id, bank_id):
        self.db_connection = db_connection.connection
        self.type = type
        self.count = count
        self.usr_id = usr_id
        self.bank_id = bank_id
    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO my_app_transaction (type, count, user_id,
bank_id) VALUES (%s, %s, %s, %s);",
                  (self.type, self.count, self.usr_id, self.bank_id))
        self.db_connection.commit()
        c.close()
con = Connection("kate", "123", "db_rip")
with con:
    user = User(con, 'Екатерина'.encode('utf-8'), 'Семенова'.encode('utf-8'),
'Владимировна'.encode('utf-8'), '4510'.encode('utf-8'), '1996-12-02')
    user.save()
    bank = Bank(con, 'Стандарт', 'Адрес!')
    bank.save()
    bank = Bank(con, 'Стандарт2', 'Адрес2')
    bank.save()
    for i in range(2, 10):
        tr = Transaction(con, 'perevod', '1000', i/2, i)
        tr.save()
```