Московский Государственный Технический Университет им. Н. Э. Баумана
Факультет «Информатика и системы управления»

Отчет по лабораторной работе № 6
Курс «Разработка Интернет-приложений»

Выполнил:

студент группы ИУ5-54 _____
(подпись)

Харлашкин А. И.

"__"_____2016 г.

Проверил:
Преподаватель каф. ИУ5

_____
(подпись)

Гапанюк Ю. Е.

"__"_____2016 г.

Москва, МГТУ – 2016 г.

Описание задания лабораторной работы

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

Листинг программы

Settings.py

```
"""
Django settings for lab6 project.

Generated by 'django-admin startproject' using Django 1.10.3.

For more information on this file, see
https://docs.djangoproject.com/en/1.10/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.10/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '&14-rmry%jq3w%d)78)rud8uiki=l8m=ycn%c)1k0dl26d=ncd'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'users',
```

```
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'lab6.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'lab6.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.10/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'first_db',
        'USER': 'admin',
        'PASSWORD': '12345',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}


# Password validation
# https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.10/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.10/howto/static-files/

STATIC_URL = '/static/'
```

Models.py

```
from django.db import models


class User(models.Model):   #пользователь
    idUser = models.IntegerField(unique=True)
    email = models.EmailField(max_length=255, unique=True, null=False)
    bill = models.IntegerField()
    first_name = models.CharField(max_length=255, null=False)
    last_name = models.CharField(max_length=255, null=False)


class Bet(models.Model):  #ставка
    idBet = models.IntegerField(unique=True)
    size = models.IntegerField(null=False)
    result = models.BooleanField(null=False)


class Fight(models.Model):  #бой
    idFight = models.IntegerField(unique=True)
    date = models.DateField(null=False)
    time = models.TimeField(null=False)
    boxer1 = models.TextField(max_length=255, null=False)
    boxer2 = models.TextField(max_length=255, null=False)

    def __str__(self):
        return 'Fight {}'.format(self.title)


class Boxer(models.Model):  #Боксёр
    idBoxer = models.IntegerField(unique=True)
```

```
    title = models.TextField(max_length=255)
    boxer_first_name = models.CharField(max_length=255, null=False)
    boxer_last_name = models.CharField(max_length=255, null=False)
```

views.py

```
from django.shortcuts import render
from users.models import Fight
from django.views.generic import TemplateView


class Fights(TemplateView):
    template_name = 'index.html'

    def get_context_data(self, **kwargs):
        Fights = Fight.objects.all()
        context = dict(Fights=Fights)
        return context

class HomePageView(TemplateView):

    template_name = "index.html"

    def get_context_data(self, **kwargs):
        context = super(HomePageView, self).get_context_data(**kwargs)
        context = Fight.objects.all()
        return context
```

urls.py

```
"""lab6 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.10/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url,
include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from users.views import HomePageView
from users.views import Fights

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^fights/$', Fights.as_view()),
]
```

Admin.py

```
from django.contrib import admin
from users.models import *

admin.site.register(User)
admin.site.register(Bet)
admin.site.register(Fight)
admin.site.register(Boxer)
```

apps.py

```
from django.apps import AppConfig


class AdminConfig(AppConfig):
    name = 'users'
```

testscript.py

```
import pymysql
pymysql.install_as_MySQLdb()

db = pymysql.connect(
    host="127.0.0.1",
    user="admin",
    passwd="12345",
    db="first_db",
    charset="utf8"
)

cursor = db.cursor()

cursor.execute("""INSERT INTO Books(name, discription)VALUES(%s, %s),(%s,
%s)""",
               ( "Преступление и наказание", "Классика",
                "Три товарища", "Зарубежная литература")
               )

db.commit()

#cursor.execute("DELETE FROM Books WHERE id>1")
#db.commit()

cursor.execute("SELECT * FROM Books;")

books = cursor.fetchall()

for book in books:
    print(book)

cursor.close()
db.close()
```

connections.py

```
try:
    import pymysql
    pymysql.install_as_MySQLdb()
except ImportError:
    pass
```

User_file.py

```
import pymysql as 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
            )
    def disconnect(self):
        if self._connection:
            self._connection.close()

class Book:
    def __init__(self, db_connection, name, discription):
        self.db_connection = db_connection.connection
        self.name=name
        self.discription = discription

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO books (name, discription) VALUES (%s, %s);",
                  (self.name, self.discription))
        self.db_connection.commit()
        c.close()

con = Connection("admin", "12345", "first_db")

with con:
```

```
    book = Book(con, 'New book','Discription new book')
    book.save()
```

wsgi.py

```
"""
WSGI config for lab6 project.

It exposes the WSGI callable as a module-level variable named
``application``.

For more information on this file, see
https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "lab6-master.settings")

application = get_wsgi_application()
```

0001_inital.py

```
# -*- coding: utf-8 -*-
# Generated by Django 1.10.4 on 2016-12-17 22:04
from __future__ import unicode_literals

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Bet',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('idBet', models.IntegerField(unique=True)),
                ('size', models.IntegerField()),
                ('result', models.BooleanField()),
            ],
        ),
        migrations.CreateModel(
            name='Boxer',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('idBoxer', models.IntegerField(unique=True)),
                ('title', models.TextField(max_length=255)),
                ('boxer_first_name', models.CharField(max_length=255)),
                ('boxer_last_name', models.CharField(max_length=255)),
```

```
            ],
        ),
        migrations.CreateModel(
            name='Fight',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('idFight', models.IntegerField(unique=True)),
                ('date', models.DateField()),
                ('time', models.TimeField()),
                ('boxer1', models.TextField(max_length=255)),
                ('boxer2', models.TextField(max_length=255)),
            ],
        ),
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('idUser', models.IntegerField(unique=True)),
                ('email', models.EmailField(max_length=255, unique=True)),
                ('bill', models.IntegerField()),
                ('first_name', models.CharField(max_length=255)),
                ('last_name', models.CharField(max_length=255)),
            ],
        ),
    ]
```

Manage.py

```
#!/usr/bin/env python
import os
import sys
import lab6.connections

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "lab6.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

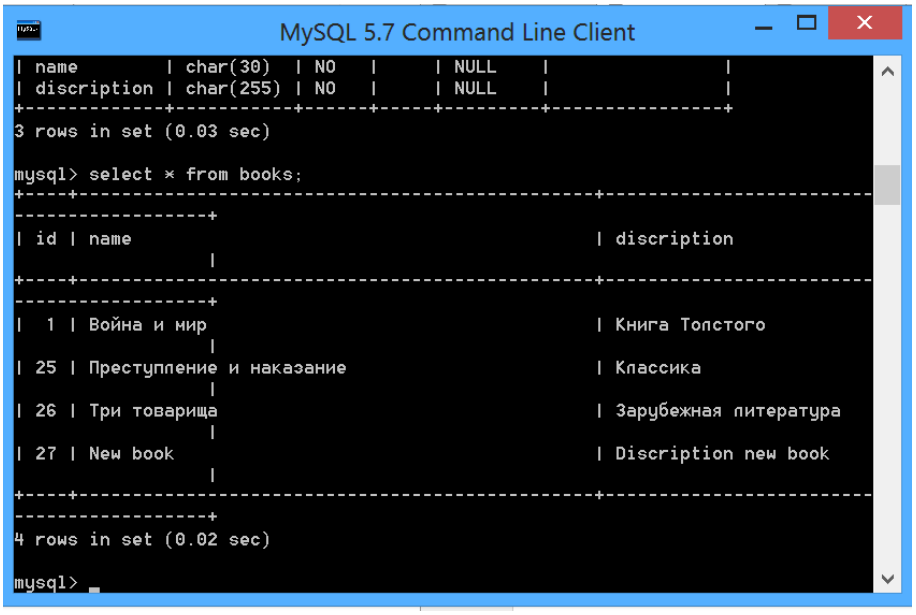
index.html

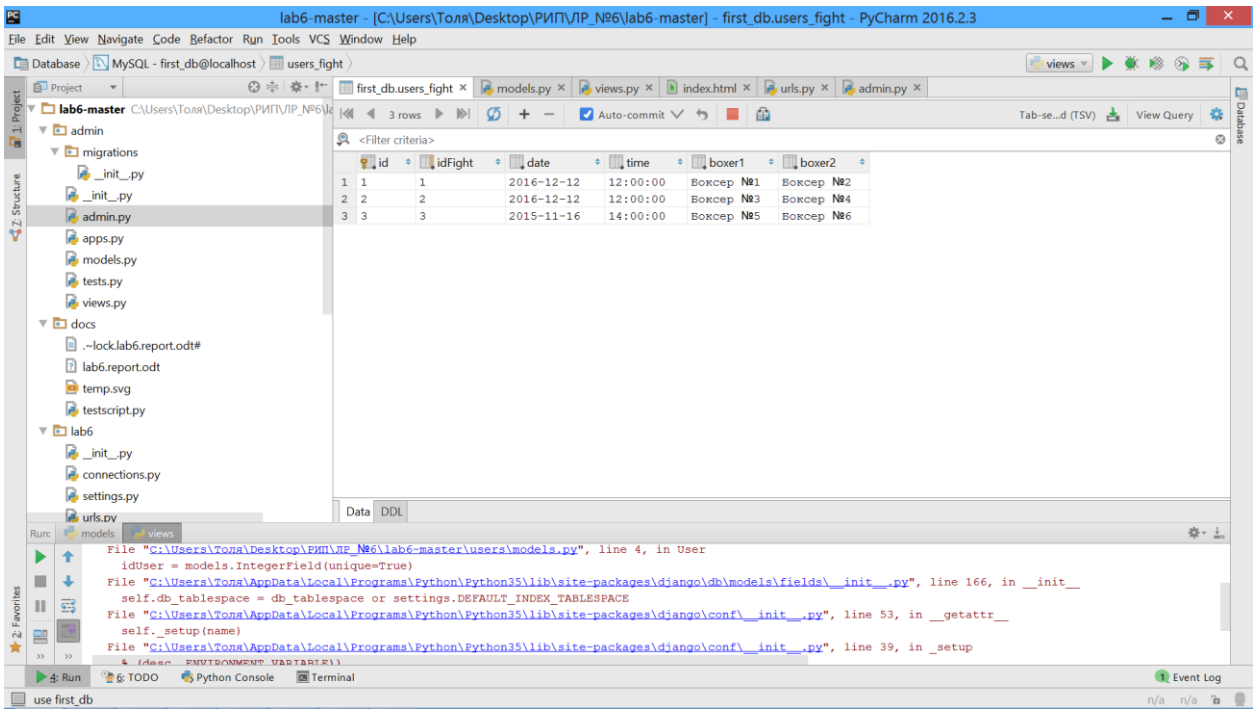
```
<html>
<body>
```

```
    {% for Fight in Fights %}
        <p>{{ Fight.boxer1 }} ___VS___ {{ Fight.boxer2 }}</p>
        <hr/>
    {% endfor %}

</body>
</html>
```

Результаты работы программы:






127.0.0.1:8000/fights/
127.0.0.1:8000/fights/
Боксер №1 ___VS___ Боксер №2
Боксер №3 ___VS___ Боксер №4
Боксер №5 ___VS___ Боксер №6