**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка Интернет-приложений»

Отчет по домашнему заданию

по курсу РИП

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-53 |  | преподаватель каф. ИУ5 |
| Беспалова У.А. |  | Гапанюк Ю.Е. |
| Подпись и дата:  10.12.2019 |  | Подпись и дата: |

Москва, 2019 г.

1. **Содержание задания**

В рамках домашнего задания студент разрабатывает веб-приложение на основе технологий Python, Django, JavaScript, MySQL.

Предметная область домашнего задания содержит субъект, отношение и объект. Необходимые атрибуты субъекта и объекта выбираются студентом самостоятельно. Предметная область (по вариантам) представлена в следующей таблице. По согласованию с преподавателем предметная область может быть изменена.

Разрабатываемое веб-приложение должно выполнять указанные ниже функции. Функции разбиты на группы. Для каждой функции указано количество рейтинговых баллов, которые получает студент в случае реализации данной функции.

### Регистрация

1. При входе, если не залогинен, открывается регистрация (1 балл)
2. Есть рабочая кнопка выхода (1 балл)

### Авторизация

1. Автоматически логинится, редирект на главную страницу (список сущностей). (1 балл)
2. Если логин / пароль неправильный, показана ошибка (1 балл)
3. Профиль пользователя использует стороннюю модель либо переопределена основная модель юзера (1 балл)

### Список сущностей

1. Страница открывается и данные берутся из базы. (1 балл)
2. Отображаются картинки сущностей (1 балл)
3. Есть пагинация (1 балл)
4. Пагинация или бесконечный скролл реализован на AJAX. (1 балл)

### Страница добавления сущностей

1. Реализована в виде модального окна для добавления сущностей. (1 балл)
2. Добавление сущности происходит по AJAX (1 балл)
3. Есть форма с необходимыми для сущности полями (1 балл)
4. Есть возможность загрузить картинку (1 балл)
5. В случае ошибок они отображаются (1 балл)
6. Валидация основных полей формы происходит с помощью JS (1 балл)
7. Сущность создается в базе данных и редирект на страницу с сущностью (1 балл)

### Страница с сущностью

1. Отображаются данные из базы данных (1 балл)
2. При нажатии кнопки действия происходит добавление этого действия в базу, страница перезагружается, это новое действие добавляется в список действий. (1 балл)
3. Действие выполняется без перезагрузки (AJAX) (1 балл)
4. Список действий всех пользователей над этой сущностью (1 балл)

### Страница личного кабинета пользователя

1. Возможность загрузить аватарку (1 балл)
2. Возможность изменить поля пользователя (1 балл)

### Django-админка

1. Русификация. (1 балл)
2. Работа со всеми сущностями. (1 балл)
3. Фильтрация и поиск в админке основной сущности. (1 балл)

### Deploy

1. Настроен mysql (1 балл)
2. Настроен nginx (1 балл)
3. Настроен gunicorn (1 балл)
4. Проект находится в git-репозитории (1 балл)
5. Текст программы

1. Шаблоны

Базовый шаблон:

</<!DOCTYPE html>

{% load staticfiles %}

<html>

<head>

<meta charset="utf-8">

<title>AskMe</title>

<!-- my CSS -->

<link rel='stylesheet' href='ss-s.css' type='text/css'>

<script src="like.js"></script>

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cYRxT2MZw1T" crossorigin="anonymous">

<link rel='stylesheet' href='/css/bootstrap.min.css' type='text/css' media='all'>

</head>

<body>

<nav class="navbar sticky-top navbar-light bg-light">

<a class="navbar-brand lg-5" href="{% url 'index' %}">

<img src="{% static "pictures/logo.png" %}" width="200" height="50" class="d-inline-block align-top" alt=""></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li>

<form class="form-inline my-2 my-lg-0 mr-2">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-secondary mt-2 my-sm-0" type="submit">Search</button>

</form>

</li>

<li>

<div class="form-actions">

<a type="submit" class="btn btn-primary mx-lg-5 mt-lg-0 mt-sm-2 mt-2" href="{% url 'ask' %}" role="button" aria-expanded="false">&nbsp&nbspAsk!&nbsp&nbsp </a>

</div>

</li>

</ul>

{% if user.is\_authenticated %}

{% include "top\_registered.html" %}

{% else %}

{% include "top\_unregistered.html" %}

{% endif %}

</div>

</nav>

<!--Конец шапки-->

<!--Основной блок-->

<div class="container px-lg-0 my-5">

<div class="row mx-lg-n5">

<div class="col-sm-9 ">

<div>

{% block content %} {% endblock %}

</div>

</div>

<!--Конец основного блока-->

<!--БОКОВУШКИ-->

<div class="col-sm-3 mt-1 justify-content-end">

<div class="container">

<div align="left">

<h4><font color=#000080>Popular tags</font></h4>

<a href="{% url 'tag' tg='627' %}" style="color: #0000CD; font-size: 1.2em; text-decoration: none">627 &nbsp</a>

<a href="{% url 'tag' tg='bbb' %}" style="color: #4B0082; font-size: 1.1em; text-decoration: none">&nbsp&nbsp bbb &nbsp</a>

<a href="{% url 'tag' tg='four' %}" style="color: #4169E1; font-size: 1.3em; text-decoration: none"> &nbsp &nbsp four &nbsp</a>

<a href="{% url 'tag' tg='five' %}" style="color: #0000CD; font-size: 1.2em; text-decoration: none"> five &nbsp</a>

<a href="{% url 'tag' tg='six' %}" style="color: #4B0082; font-size: 1.1em; text-decoration: none"><b> &nbsp &nbsp six &nbsp </b></a>

<a href="{% url 'tag' tg='proba' %}" style="color: #4B0082; font-size: 1.1em; text-decoration: none"><b> &nbsp &nbsp proba &nbsp </b></a>

<a href="{% url 'tag' tg='one' %}" style="color: #4169E1; font-size: 1.3em; text-decoration: none"> &nbsp one &nbsp</a>

<a href="{% url 'tag' tg='aaa' %}" style="color: #0000CD; font-size: 1.2em; text-decoration: none"> &nbspaaa</a>

<!--Сделать нормально по-умному (узнать как) -->

</div>

</div>

<div class="container mt-5">

<h4><font color=#000080>Best members</font></h4>

</div>

<div class="mx-n4">

<ul class="link\_members" >

<li> <a href="....somewhere">Ivan</a> </li>

<li> <a href="....somewhere">Another Ivan</a> </li>

<li> <a href="....somewhere">Ivan the Clever</a> </li>

<li> <a href="....somewhere">Unexpected Ivan</a> </li>

</ul>

</div>

</div>

</div>

</div>

</div>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

</body>

</html>>

Главная страница:

{% extends "base.html" %}

{% block content %}

<div >

<div class="row mb-3"><h4>

<span class="col-3"><a href="{% url 'index' %}" style="color: #000080; text-decoration: none">New Questions</a></span>

<span class="col-3"><a href="{% url 'hot' %}" style="color: #0000CD; font-size: 0.6em; text-decoration: none">Hot Questions</a></span>

</h4>

</div>

{% for question in questions\_list.object\_list %}

{% include "que/question.html" %}

{% endfor %}

</div>

{% include "paginator.html" %}

{% endblock %}

Верстка одного вопроса:

{% load staticfiles %}

<!--один вопрос-->

<div class="card w-90 my-2">

<div class="row">

<img src="{% static "pictures/avatarka1.jpg" %}" width="60" height="75" class="d-inline-block align-left" alt="">

</div>

<div class="col-10 my-3 mx-md-3 mx-3">

<h5 class="card-title">

<a href="{% url 'one-question' qid=question.id %}" style="color: #0000CD; font-size: 1.1em;">

{{ question.title }}

</a> </h5>

<p class="card-text">{{ question.content }}</p>

<p class="card-text mt-n2"><small class="text-muted">Last updated 3 mins ago by {{question.author}}</small></p>

<div class="row my-2">

<div class="col-2">

<a href="{% url 'one-question' qid=question.id %}" style="color: #000CD;"> Answers ({{question.count\_answers}})</a>

</div>

<div class="col-5">

Tags:

{% for t in question.tag\_list%}

&nbsp &nbsp<a href="{% url 'tag' tg=t %}" style="color: #0000CD;"> {{t}} &nbsp</a>

{%endfor%}

</div>

<div class="col-4 d-none d-lg-block">

<div class="row justify-content-end">

<a class="like">

<button type="button" class="btn btn-outline-primary btn-sm mx-2">

&nbsp Like &nbsp

<!--<input class="qty1" name="qty1" readonly="readonly" type="text" value="0" />-->

</button>

&nbsp {{question.overall\_rating }} &nbsp&nbsp

</a>

<a class="dislike">

<button type="button" class="btn btn-outline-danger btn-sm mx-2">

Dislike

</button>

</a>

</div>

</div>

</div>

</div>

2. Подключение БД

Модели:

from django.db import models

from django.conf import settings

from django.core.exceptions import ObjectDoesNotExist

from django.contrib.contenttypes.models import ContentType

from django.dispatch import receiver

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.db.models import Sum, F

# Create your models here.

class LikeDislike (models.Model):

LIKE = 1

DISLIKE = -1

VOTES = ((LIKE, 'like'), (DISLIKE, 'dislike'))

vote = models.SmallIntegerField(choices=VOTES, default=LIKE)

author = models.ForeignKey('Author', null=True, blank=True, on\_delete=models.CASCADE)

content\_type = models.ForeignKey(ContentType, null=True, blank=True, on\_delete=models.CASCADE)

object\_id = models.PositiveIntegerField()

content\_object = GenericForeignKey('content\_type', 'object\_id')

objects = LikeDislikeManager()

class Author (models.Model):

user = models.OneToOneField(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

avatar = models.ImageField(upload\_to = "static/pictures", default='static/pictures/default.jpeg')

rating = models.IntegerField(default=0)

def \_\_str\_\_(self):

return self.user.username

class QuestionManager (models.Manager):

def get\_question(self, qid):

return self.get(pk=qid)

def hot(self):

return self.order\_by('-id')

def new(self):

return self.order\_by('-id')

def find\_tag(self, our\_tag):

return Question.objects.filter(question\_tags\_\_tag\_\_tag\_title=our\_tag)

def get\_answers (self, qid):

return Answer.objects.filter(question=qid)

class Question (models.Model):

title = models.CharField(max\_length=1)

content = models.TextField()

author = models.ForeignKey('Author', on\_delete=models.CASCADE)

rating = GenericRelation(LikeDislike, related\_query\_name='questions')

objects = QuestionManager()

def overall\_rating(self):

return LikeDislike.objects.questions().filter(object\_id=self.id).aggregate(Sum('vote')).get('vote\_\_sum') or 0

def tag\_list(self):

return list(set(Tag.objects.filter(question\_tags\_\_question=self.id)))

def count\_answers (self):

return (Answer.objects.filter(question = self.id)).count()

class Tag (models.Model):

tag\_title = models.CharField(max\_length=20)

def \_\_str\_\_(self):

return self.tag\_title

class Answer (models.Model):

content = models.TextField()

author = models.ForeignKey('Author', on\_delete=models.CASCADE)

question = models.ForeignKey('Question', null=True, blank=True, on\_delete=models.CASCADE)

correct = models.BooleanField(default=False)

rating = GenericRelation(LikeDislike, related\_query\_name='answers')

def overall\_rating(self):

return LikeDislike.objects.answers().filter(object\_id=self.id).aggregate(Sum('vote')).get('vote\_\_sum') or 0

class Question\_tags(models.Model):

question = models.ForeignKey(Question, on\_delete = models.CASCADE)

tag = models.ForeignKey(Tag, on\_delete = models.CASCADE)

Views

from django.shortcuts import render, redirect, reverse

from django.http import JsonResponse

from django.http import HttpResponse

from .models import Question

from django.contrib import auth

from django.contrib.auth import authenticate

from django.contrib.auth.decorators import login\_required

from app.models import Author

from django.contrib.auth.models import User

#from django.core.exceptions import KeyError

# Create your views here.

def paginate (request, object\_list):

current\_page = Paginator(object\_list, 5)

page = request.GET.get ('page')

try:

our\_objects\_list = current\_page.page(page)

except PageNotAnInteger:

our\_objects\_list = current\_page.page(1)

except EmptyPage:

our\_objects\_list = current\_page.page(current\_page.num\_pages)

return our\_objects\_list

def index (request):

return render (request, 'index.html' , {

'questions\_list': paginate(request, Question.objects.new()),

})

def tag\_questions (request, tg):

return render (request, 'index\_tag.html' , {'tg':tg,

'questions\_list': paginate(request, Question.objects.find\_tag(tg)),})

def question (request, qid):

if request.POST:

form = AnswerForm(qid, request.user.author, data=request.POST)

if form.is\_valid():

answer = form.save()

return redirect(reverse("one-question", kwargs={'qid': qid}))

else:

form = AnswerForm(qid)

return render(request, 'one-question.html', {

'question': Question.objects.get\_question(qid),

'answers' : Question.objects.get\_answers(qid),

'form': form })

def login (request):

if request.method == 'POST':

form = LoginForm(request.POST)

if form.is\_valid():

username = request.POST.get('login')

password = request.POST.get('password')

user = auth.authenticate(username = username, password=password)

if user is not None:

auth.login(request, user)

return redirect('index')

else:

form = LoginForm()

return render (request, 'login.html', {'form': form, 'login':form.data.get('login')})

def logout (request):

auth.logout(request)

return redirect(reverse("index"))

def signup (request):

if request.method == 'POST':

form = SignUpForm(request.POST),

files = request.FILES,

if form.is\_valid():

author = form.save(request.POST)

return redirect(reverse("login"))

else:

form = SignUpForm()

return render (request, 'signup.html', {

'form': form,

'user\_name':form.data.get('user\_name'),

'email':form.data.get('email'),

'nick\_name':form.data.get('nick\_name')})

def ask(request):

if request.POST:

form = AskForm(request.user.author, data=request.POST)

if form.is\_valid():

question = form.save()

form.save\_tags(request.POST.get('tags'), question)

return redirect(reverse("one-question", kwargs={'qid': question.pk}))

else:

form = AskForm(request.user.author)

return render(request, 'ask.html', {

'form': form,

'title':form.data.get('title'), #ты мне не нравишься

'content':form.data.get('content'),

'tags':form.data.get('tags')})

def settings (request):

saved = ""

if request.POST:

form = SettingsForm(request.user, request.POST, data=request.POST)

#form.save(request.user) #

if form.is\_valid():

form.save(request.user)

saved = "Changes saved"

else:

form = SettingsForm(request.user)

return render (request, 'settings.html', {

'user\_name':form.user\_name,

'email':form.email,

'nick\_name':form.nick\_name,

'saved':saved})

3. Настройка Django.admin

from django.contrib import admin

from app.models import Author, Question, Answer, Tag, LikeDislike, Question\_tags

# Register your models here.

class AuthorAdmin (admin.ModelAdmin):

pass

class QuestionAdmin (admin.ModelAdmin):

pass

class AnswerAdmin (admin.ModelAdmin):

pass

class TagAdmin (admin.ModelAdmin):

pass

class LikeDislikeAdmin (admin.ModelAdmin):

pass

class Question\_tagsAdmin (admin.ModelAdmin):

pass

admin.site.register (Author, AuthorAdmin)

admin.site.register (Question, QuestionAdmin)

admin.site.register (Answer, AnswerAdmin)

admin.site.register (Tag, TagAdmin)

admin.site.register (LikeDislike, LikeDislikeAdmin)

admin.site.register (Question\_tags, Question\_tagsAdmin)

1. **Экранные формы с примерами работы программы**









