Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)____

О Т Ч Е Т

по домашнему заданию

по дисциплине: Разработка Интернет-приложений_____

на тему:___Разработка веб-приложения на основе технологий Python, Django, JavaScript,MySQL_____
_____
_____
_____
_____
_____
_____

Студент __ИУ5-54Б_____ _____ _____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ _____
(Подпись, дата) (И.О.Фамилия)

2019 г.

Задание лабораторной работы

Разработать веб сервис на базе технологий: Python, Django, JS, MySQL.
Предметная область:
Субъект – **Пользователь**
Отношение – **Подписка**
Объект – **Онлайн-курс**

Требования

1. Регистрация

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

2. 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)

3. 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)

4. 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)

5. 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)

6. Страница личного кабинета пользователя

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

7. Django-админка

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

8. Deploy

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

Листинги исходных модулей

- Urls.py:

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.conf.urls import include, url
from django.urls import path
from django.contrib.auth.views import LoginView, LogoutView, PasswordChangeView,
PasswordChangeDoneView

from books import views
namespace ='course'
app_name ='course'

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^course/', include('courses.urls.urls'), name='course'),
    path('logout/', LogoutView.as_view(), name='logout'),
    path('login/', LoginView.as_view(), name='login', ),
    url(r'^password-change/$', PasswordChangeView.as_view(),
name='password_change'),
    url(r'^password-change/done/$', PasswordChangeDoneView.as_view(),
name='password_change_done'),
    url(r'^register/$', views.register, name='register'),
    url(r'^edit/$', views.edit, name='edit'),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

- Admin.py:

```
from django.contrib import admin
from .models import Profile
# Register your models here.
from books.models import Book, Comment


# отображение полей в списке объектов
class PostAdmin(admin.ModelAdmin):
```

```
    list_display = ('title', 'slug', 'teacher', 'start_date', 'end_date',
'status', 'groups')
    list_filter = ('status', 'start_date', 'groups', 'teacher')
    search_fields = ('title', 'teacher')
    prepopulated_fields = {'slug': ('title',)}
    date_hierarchy = 'start_date'
    ordering = ['status', 'teacher']


admin.site.register(Book, PostAdmin)


class CommentAdmin(admin.ModelAdmin):
    list_display = ('name', 'course', 'created', 'active', 'imagin')
    list_filter = ('active', 'created')
    search_fields = ('name', 'body')
    raw_id_fields = ('course', 'name')


admin.site.register(Comment, CommentAdmin)


class ProfileAdmin(admin.ModelAdmin):
    list_display = ['user', 'photo']


admin.site.register(Profile, ProfileAdmin)
```

- Apps.py:

```
from django.apps import AppConfig


class BooksConfig(AppConfig):
    name = 'courses'
```

- Forms.py:

```
from django import forms
from django.contrib.auth.models import User

from books.models import Comment, Profile


class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ('body', 'imagin')


class UserRegistrationForm(forms.ModelForm):
    password = forms.CharField(label='Password', widget=forms.PasswordInput)
    password2 = forms.CharField(label='Repeat password',
widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ('username', 'first_name', 'email')

    def clean_password2(self):
        cd = self.cleaned_data
        if cd['password'] != cd['password2']:
            raise forms.ValidationError('Passwords don\'t match.')
        return cd['password2']
```

```
class UserEditForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ('first_name', 'last_name', 'email')


class ProfileEditForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ('photo',)
```

- Models.py:

```
from django.conf import settings
from django.contrib.auth.models import User
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.urls import reverse
from django.utils import timezone


# Create your models here.
class Book(models.Model):
    STATUS_CHOICES = (
        ('o', 'Opened'),
        ('c', 'Closed'),
    )
    id = models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250, unique_for_date='start_date')
    teacher = models.CharField(max_length=250)
    desc = models.TextField()
    start_date = models.DateTimeField(default=timezone.now)
    end_date = models.DateTimeField(auto_now_add=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES,
default='draft')
    image = models.ImageField(upload_to='books/%Y/%m/%d', blank=True)


class Meta:
    ordering = ('-start_date',)


def __str__(self):
    # Стандартное представление объекта
    return self.title


class Comment(models.Model):
    course = models.ForeignKey(Book, related_name='comments',
on_delete=models.CASCADE)
    teacher = models.ForeignKey(User, related_name='users',
on_delete=models.CASCADE)
    body = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
    active = models.BooleanField(default=True)
    imagin = models.ImageField(upload_to='courses/%Y/%m/%d', blank=True)

    class Meta:
        ordering = ('created',)

    def __str__(self):
```

```
        return 'Course by {} on {}'.format(self.name, self.book)


class Profile(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, primary_key=True)
    photo = models.ImageField(upload_to='courses/%Y/%m/%d', blank=True,
default='courses/default.png')


@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)


@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()

    def __str__(self):
        return 'Profile for user {}'.format(self.user.username)
```

- Views.py:

```
import json

from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.models import User
from django.core import serializers
from django.core.exceptions import ValidationError
from django.http import HttpResponseRedirect, JsonResponse, HttpResponse
from django.shortcuts import get_object_or_404, render, redirect
from django.utils.decorators import method_decorator
from django.views.generic import ListView
from django.utils.translation import gettext as _
from el_pagination.views import AjaxListView

from books.forms import CommentForm, UserRegistrationForm, UserEditForm,
ProfileEditForm
from books.models import Book
from .models import Profile


@login_required
def course_detail(request, ident):
    course = get_object_or_404(Course, id=ident)
    teacher = get_object_or_404(Teacher, id=request.user.id)
    profile = get_object_or_404(Profile, user=name)
    return render(request, 'detail.html', {'course': course})


class CourseListView(LoginRequiredMixin, AjaxListView):
    context_object_name = 'courses'
    template_name = 'list.html'
    page_template = 'entry_list_page.html'
    def get_queryset(self):
        return Course.objects.all()


def register(request):
    if request.method == 'POST':
        user_form = UserRegistrationForm(request.POST)
```

```
        if user_form.is_valid():
            user_form.save()
            return render(request, 'registration/register_done.html')
    else:
        user_form = UserRegistrationForm()
    return render(request, 'registration/register.html', {'user_form':
user_form})

@login_required
def edit(request):
    if request.method == 'POST':
        user_form = UserEditForm(instance=request.user, data=request.POST)
        profile_form = ProfileEditForm(instance=request.user.profile,
data=request.POST, files=request.FILES)
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            messages.success(request, _('Ваш профиль был успешно обновлен!'))
            return redirect('/course/')
        else:
            raise ValidationError(_('Invalid date - renewal in past'))
    else:
        user_form = UserEditForm(instance=request.user)
        profile_form = ProfileEditForm(instance=request.user.profile)
        return render(request,
                      'edit.html',
                      {'user_form': user_form,
                       'profile_form': profile_form})
```