Программный код основных модулей программы продукта «Сервис создания уникальных профилей групп «VEDA»».

Views.py

from VEDA\_application.templatetags.veda\_extras import get\_avg\_user, get\_count\_n, get\_count\_n\_user

from django.dispatch.dispatcher import receiver

from django.shortcuts import render, HttpResponseRedirect, get\_object\_or\_404

from django.contrib.auth.models import User

from django.contrib.auth import authenticate, logout, login

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

from django.core.mail import send\_mail, EmailMessage

from django.urls import reverse

from . import models

from VEDA\_application.models import Admin\_post, Client, Discipline, Group, Group\_post, Lesson, Note, Notice, Notification, Parent, Task

import json

import openpyxl

from datetime import datetime

def index(request):

""" Основная страница сайта, где расположены

группы пользователя, их новости и блог разработчика """

try:

notes = Note.objects.filter(receiver\_\_id=request.user.client.id)

avg = get\_avg\_user(notes, request.user.client)

count\_n = get\_count\_n\_user(notes, request.user.client)

count\_less\_than\_4 = 0

for note in notes:

try:

if int(note.value) < 4:

count\_less\_than\_4 += 1

except:

pass

except:

avg = 0

count\_n = 0

count\_less\_than\_4 = 4

if request.method == 'GET':

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

'admin\_posts': models.Admin\_post.objects.all(), # посты от администратора

'l\_groups': models.Group.objects.filter(listeners\_\_user\_\_username=request.user.username), # группы, где есть текущий пользователь

'm\_groups': models.Group.objects.filter(moderators\_\_user\_\_username=request.user.username), # группы, где есть текущий пользователь

'group\_posts\_listener': models.Group\_post.objects.filter(group\_\_listeners\_\_user\_\_username=request.user.username), # посты в группах, в которых находится пользователь

'group\_posts\_moderator': models.Group\_post.objects.filter(group\_\_moderators\_\_user\_\_username=request.user.username), # посты в группах, в которых находится пользователь

'notifications': models.Notification.objects.filter(receiver\_\_user\_\_username=request.user.username), # уведомления для пользователя

'tasks': models.Task.objects.filter(receiver\_\_user\_\_username=request.user.username), # задачи из группа для пользователя

'avg': avg,

'count\_n': count\_n,

'count\_less\_than\_4': count\_less\_than\_4

})

def logout\_view(request):

""" View для выхода из аккаунта авторизированного пользователя, в любом случае

(даже если не авторизирован пользователь) - возвращает на основную страницу """

logout(request)

return HttpResponseRedirect(reverse('index'))

def login\_view(request):

""" Страница входа в учетную запись пользователя, должна обрабатывать

сохранение, вход и переходы на главную страницу, восстановление пароля

и регистрацию пользователя """

if request.method == 'GET':

return render(request, 'login.html', context={

})

if request.method == "POST":

user = authenticate(username=request.POST.get('username'), password=request.POST.get('password'))

if user is not None:

login(request, user)

return HttpResponseRedirect(reverse('index'))

else:

return render(request, 'login.html', context={

'error': True

})

def register(request):

""" Страница регистрации, обрабатывает выдачу формы для ввода данных

нового пользователя, а также принимает POST запрос на создание нового пользователя """

if request.method == "GET":

return render(request, 'register.html', context={

})

if request.method == "POST":

try:

user = User.objects.create\_user(username=request.POST.get('username'),

email=request.POST.get('email'),

first\_name=request.POST.get('firstname'),

last\_name=request.POST.get('secondname'),

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

# user.client.adress = request.POST.get('address')

# user.client.hobbies = request.POST.get('hobbies')

user.client.day\_of\_birthday = request.POST.get('dob')

user.client.gender = request.POST.get('gender')

user.client.status = ""

user.save()

if user is not None:

login(request, user)

send\_mail('Регистрация',

f'Поздравляем, вы зарегистрировались на нашей платформе, ваш логин: {user.username}! Приятного использования!',

'labark.group@gmail.com',

[user.email])

return HttpResponseRedirect(reverse('index'))

except Exception as e:

print(e)

return render(request, 'register.html', context={

'error': e

})

def profile(request, pk):

""" Отдает страницу профиля, и, в случае, если это профиль пользователя, то может принять

UPDATE запрос на изменение данных профиля """

if request.method == "GET":

client = get\_object\_or\_404(models.Client, pk=pk)

groups = models.Group.objects.filter(listeners\_\_user\_\_username=client.user.username)

parents\_access = False

for group in groups:

if request.user.client in group.moderators.all():

parents\_access = True

return render(request, 'profile.html', context={

'client': client,

'groups': groups,

'parents\_access': parents\_access

})

def post(request):

""" Страница конкретного поста блога (как группы, так и разработчика), может быть POST,

тогда создает новый пост в области создания """

if request.method == "GET":

return render(request, 'post.html', context={

})

if request.method == "POST":

if request.user.is\_staff:

newPost = Admin\_post(author=request.user.client, article=request.POST.get('article'), text=request.POST.get('content'))

newPost.save()

return HttpResponseRedirect(reverse('index'))

def group(request, pk):

""" Страница, отображающая всю основную информацию о комнате группы, принимает POST,

чтобы создать страницу группы, UPDATE для изменения данных """

if request.method == "GET":

group = get\_object\_or\_404(models.Group, pk=pk)

listeners = group.listeners.all()

moderators = group.moderators.all()

notes = Note.objects.filter(group\_\_id=group.id)

sorted\_users = list(set(listeners | moderators))

sorted\_users = sorted(sorted\_users, key=lambda user: -get\_avg\_user(notes, user))

return render(request, 'group.html', context={

'group': group,

'listeners': listeners,

'moderators': moderators,

'sorted\_users': sorted\_users,

'notices': models.Notice.objects.filter(group\_\_id=group.id),

'tasks': models.Task.objects.filter(group\_\_id=group.id),

'posts': models.Group\_post.objects.filter(group\_\_id=group.id),

'sheet': models.Lesson.objects.filter(group\_\_id=group.id),

'disciplines': models.Discipline.objects.filter(group\_\_id=group.id),

'days': range(1, 32),

'notes': notes,

'months': [

(1, 'Январь'),

(2, 'Февраль'),

(3, 'Март'),

(4, 'Апрель'),

(5, 'Май'),

(6, 'Июнь'),

(7, 'Июль'),

(8, 'Август'),

(9, 'Сентябрь'),

(10, 'Октябрь'),

(11, 'Ноябрь'),

(12, 'Декабрь'),

],

})

@login\_required

def create\_group(request):

""" Страница создания комнаты группы """

if request.method == "GET":

return render(request, 'create\_group.html', context={

})

elif request.method == "POST":

data = json.loads(request.body)

group = Group.objects.create(

name = data['name'],

tag = data['tag'],

hNotices = data['hNotices'],

hTasks = data['hTasks'],

hSheet = data['hSheet'],

)

group.moderators.add(request.user.client)

group.save()

notification = Notification.objects.create(

receiver=request.user.client,

message=f'Вы создали группу {group.name}',

priority=2

)

disciplines = []

for lmo in data['mo']:

if lmo not in disciplines:

disciplines.append(lmo)

lesson = Lesson(

group=group,

discipline=lmo,

day=1,

)

lesson.save()

for lt in data['tu']:

if lt not in disciplines:

disciplines.append(lt)

lesson = Lesson(

group=group,

discipline=lt,

day=2

)

lesson.save()

for lwe in data['we']:

if lwe not in disciplines:

disciplines.append(lwe)

lesson = Lesson(

group=group,

discipline=lwe,

day=3

)

lesson.save()

for lth in data['th']:

if lth not in disciplines:

disciplines.append(lth)

lesson = Lesson(

group=group,

discipline=lth,

day=4

)

lesson.save()

for lfr in data['fr']:

if lfr not in disciplines:

disciplines.append(lfr)

lesson = Lesson(

group=group,

discipline=lfr,

day=5

)

lesson.save()

for lsa in data['sa']:

if lsa not in disciplines:

disciplines.append(lsa)

lesson = Lesson(

group=group,

discipline=lsa,

day=6

)

lesson.save()

for lsu in data['su']:

if lsu not in disciplines:

disciplines.append(lsu)

lesson = Lesson(

group=group,

discipline=lsu,

day=7

)

lesson.save()

for discipline in disciplines:

new\_discipline = Discipline.objects.create(

group = group,

name = discipline

)

return HttpResponseRedirect(reverse('group', args=[group.id]))

@login\_required

def set\_lesson(request, pk):

""" Вью для обновления значения урока в расписании """

if request.method == "POST":

lesson = get\_object\_or\_404(Lesson, pk=pk)

new\_value = request.POST.get('discipline')

if len(new\_value) > 0:

lesson.discipline = new\_value

lesson.save()

else:

lesson.delete()

return HttpResponseRedirect(reverse('group', args=[lesson.group.id]))

@login\_required

def add\_lesson(request, pk):

""" Вью для добавления урока в расписание, рк - ключ группы, остальное -- в запрсое """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

lesson = Lesson.objects.create(

group = group,

day = request.POST.get('day'),

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

)

for moderator in group.moderators.all():

notification = Notification.objects.create(

receiver = moderator,

message = f"В группе {group.name} обновлено расписание!",

priority = 4

)

for listener in group.listeners.all():

notification = Notification.objects.create(

receiver = listener,

message = f"В группе {group.name} обновлено расписание!",

priority = 4

)

return HttpResponseRedirect(reverse('group', args=[group.id]))

def add\_discipline(request, pk):

""" Вью для создания предмета для ведения ведомости """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

discipline = Discipline.objects.create(

group = group,

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

)

return HttpResponseRedirect(reverse('group', args=[pk]))

def remove\_discipline(request, pk):

""" Вью для удаления предмета для ведения ведомости """

if request.method == "POST":

discipline = get\_object\_or\_404(Discipline, pk=pk)

discipline.delete()

return HttpResponseRedirect(reverse('group', args=[discipline.group.id]))

def group\_enter(request, pk):

""" Вью для входа в группу конкретного пользователя в группу под РК """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

group.listeners.add(request.user.client)

group.save()

moderators = group.moderators.all()

for moderator in moderators:

notification = Notification.objects.create(

receiver = moderator,

message = f'@{request.user.username} зашел в группу #{group.name}',

priority = 2

)

return HttpResponseRedirect(reverse('index'))

def group\_out(request, pk):

""" Вью для выхода пользователя из группы """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

if request.user.client in group.listeners.all():

group.listeners.remove(request.user.client)

elif request.user.client in group.moderators.all():

group.moderators.remove(request.user.client)

group.save()

moderators = group.moderators.all()

for moderator in moderators:

notification = Notification.objects.create(

receiver = moderator,

message = f'@{request.user.username} вышел из группы #{group.name}',

priority = 2

)

return HttpResponseRedirect(reverse('index'))

def group\_post(request, pk):

""" Вью для создания постов в группе """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

group\_post = Group\_post.objects.create(

group = group,

article = request.POST.get('article'),

text = request.POST.get('text'),

author = request.user.client,

)

try:

file = request.FILES['file']

if file:

group\_post.post\_file\_field = file

group\_post.save()

except:

pass

for listener in group.listeners.all():

notification = Notification.objects.create(

receiver = listener,

message = f'В группе #{group.name} новый пост!',

priority = 4

)

for moderator in group.moderators.all():

notification = Notification.objects.create(

receiver = moderator,

message = f'В группе #{group.name} новый пост!',

priority = 4

)

return HttpResponseRedirect(reverse('group', args=[pk]))

def get\_results(request, pk):

""" Получение на почту ведомости группы """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

wb = openpyxl.Workbook()

wb.create\_sheet(title = 'Ведомость', index = 0)

sheet = wb['Ведомость']

max\_count\_notes = 0

notes = Note.objects.filter(group\_\_id=group.id)

notes\_dict = {}

for note in notes:

if note.receiver.id not in notes\_dict.keys():

notes\_dict[note.receiver.id] = []

notes\_dict[note.receiver.id].append(note.value)

else:

notes\_dict[note.receiver.id].append(note.value)

cell = sheet.cell(1,1)

cell.value = f'Ведомость группы {group.name} за 12 месяцев'

iterator = 3

for key in notes\_dict.keys():

receiver = get\_object\_or\_404(Client, pk=key)

cell = sheet.cell(iterator, 1)

cell.value = receiver.user.first\_name + " " + receiver.user.last\_name

if max\_count\_notes < len(notes\_dict[key]):

max\_count\_notes = len(notes\_dict[key])

col\_iterator = 2

for note in notes\_dict[key]:

cell = sheet.cell(iterator, col\_iterator)

cell.value = note

col\_iterator += 1

iterator += 1

iterator = 3

col = max\_count\_notes + 2

cell = sheet.cell(iterator - 1, col)

cell.value = 'Средний балл'

cell = sheet.cell(iterator - 1, col + 1)

cell.value = 'Количество пропусков'

for key in notes\_dict.keys():

cell = sheet.cell(iterator, col)

cell.value = get\_avg\_user(notes, get\_object\_or\_404(Client, pk=key ))

cell = sheet.cell(iterator, col + 1)

cell.value = get\_count\_n\_user(notes, get\_object\_or\_404(Client, pk=key ))

iterator += 1

wb.save(f'Ведомость группы.xlsx')

mail = EmailMessage('Ваша ведомость группы!',

"Мы собрали для вас ведомость группы в прикрепленном файле!",

"labark.group@gmail.com",

[request.user.email])

with open('Ведомость группы.xlsx', 'rb') as report:

mail.attach('Ведомость группы.xlsx', report.read(), 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')

res = mail.send()

return HttpResponseRedirect(reverse('group', args=[group.id]))

@login\_required

def group\_search(request):

""" Вью для поиска группы по идентификтору из буков """

if request.method == "GET":

group = get\_object\_or\_404(Group, tag=request.GET.get('tag'))

return HttpResponseRedirect(reverse('group', args=[group.id]))

@login\_required

def update\_status\_group(request, pk):

""" Вью для обновления статуса между модератором и слушателем в группе """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

client = get\_object\_or\_404(Client, pk=request.POST.get('user'))

if client in group.moderators.all():

group.moderators.remove(client)

group.listeners.add(client)

elif client in group.listeners.all():

group.listeners.remove(client)

group.moderators.add(client)

group.save()

return HttpResponseRedirect(reverse('group', args=[pk]))

@login\_required

def kick\_user(request, pk):

""" Вью для обновления статуса между модератором и слушателем в группе """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk)

client = get\_object\_or\_404(Client, pk=request.POST.get('user'))

if client in group.listeners.all():

group.listeners.remove(client)

group.save()

return HttpResponseRedirect(reverse('group', args=[pk]))

def faq(request):

""" Страница ответов на основные вопросы к проекту """

if request.method == "GET":

return render(request, 'faq.html', context={

})

def settings(request):

""" Страница просмотра и изменения настроек пользователя """

if request.method == "GET":

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

})

def update\_user(request, pk):

""" Вью для обновления настроек и личной информации пользователя """

if request.method == "POST":

user = get\_object\_or\_404(User, pk=pk)

user.first\_name = request.POST.get('first\_name')

user.last\_name = request.POST.get('last\_name')

user.client.status = request.POST.get('status')

user.client.hobbies = request.POST.get('hobbies')

user.client.adress = request.POST.get('adress')

user.client.vk = request.POST.get('vk')

user.client.instagram = request.POST.get('instagram')

user.save()

return HttpResponseRedirect(reverse('settings'))

@login\_required

def update\_password(request, pk):

""" Вью для изменения пароля пользователя """

if request.method == "POST":

user = get\_object\_or\_404(User, pk=pk)

if (user.check\_password(request.POST.get('password'))):

user.set\_password(request.POST.get('new\_password'))

user.save()

user = authenticate(username=user.username, password=request.POST.get('new\_password'))

if user is not None:

login(request, user)

return HttpResponseRedirect(reverse('settings'))

else:

return render(request, 'login.html', context={

'error': True

})

else:

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

'password\_error': True

})

def update\_father(request):

""" Вью для изменения информации о отце """

if request.method == "POST":

try:

parent = request.user.client.father

parent.surname = request.POST.get('surname')

parent.name = request.POST.get('name')

parent.middlename = request.POST.get('middlename')

parent.job = request.POST.get('job')

parent.dob = request.POST.get('dob')

parent.phone = request.POST.get('phone')

parent.save()

except:

parent = Parent()

parent.surname = request.POST.get('surname')

parent.name = request.POST.get('name')

parent.middlename = request.POST.get('middlename')

parent.job = request.POST.get('job')

parent.dob = request.POST.get('dob')

parent.phone = request.POST.get('phone')

parent.save()

request.user.client.father = parent

request.user.save()

return HttpResponseRedirect(reverse('settings'))

def update\_mother(request):

""" Вью для изменения информации о отце """

if request.method == "POST":

try:

parent = request.user.client.mother

parent.surname = request.POST.get('surname')

parent.name = request.POST.get('name')

parent.middlename = request.POST.get('middlename')

parent.job = request.POST.get('job')

parent.dob = request.POST.get('dob')

parent.phone = request.POST.get('phone')

parent.save()

except:

parent = Parent()

parent.surname = request.POST.get('surname')

parent.name = request.POST.get('name')

parent.middlename = request.POST.get('middlename')

parent.job = request.POST.get('job')

parent.dob = request.POST.get('dob')

parent.phone = request.POST.get('phone')

parent.save()

request.user.client.mother = parent

request.user.save()

return HttpResponseRedirect(reverse('settings'))

def update\_userpic(request):

""" Вью для изменения аватарки пользователя """

if request.method == "POST":

# try:

user = request.user

user.client.userpic = request.FILES['file']

user.save()

return HttpResponseRedirect(reverse('settings'))

# except Exception as e:

# print(e)

return HttpResponseRedirect(reverse('index'))

def recover(request):

""" Страница восстановления пароля пользователя """

if request.method == "GET":

return render(request, 'recover.html', context={

})

def add\_note(request, pk\_group, pk\_user):

""" Вью для добавления оценки в группу """

if request.method == "POST":

group = get\_object\_or\_404(Group, pk=pk\_group)

receiver = get\_object\_or\_404(Client, pk=pk\_user)

date\_of\_create = request.POST.get('date')

discipline = get\_object\_or\_404(Discipline, pk=request.POST.get('discipline'))

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

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

note = Note.objects.create(

group = group,

receiver = receiver,

date\_of\_receive = date\_of\_create,

value = value,

discipline = discipline,

description = description

)

notification = Notification.objects.create(

receiver = receiver,

message = f'Вы получили оценку {note.value} в группе {group.name}',

priority = 1

)

print(receiver)

print(group)

print(pk\_user)

print(pk\_group)

return HttpResponseRedirect(reverse('group', args=[group.id]))

def remove\_note(request, pk):

""" Вью для удаления конкретной оценки """

if request.method == "POST":

note = get\_object\_or\_404(Note, pk=pk)

note.delete()

return HttpResponseRedirect(reverse('group', args=[note.group.id]))

@login\_required

def add\_task(request, pk):

""" Вью для добавления задачи пользователю в конкретной группе """

if request.method=="POST":

group = get\_object\_or\_404(Group, pk=pk)

user = get\_object\_or\_404(Client, pk=request.POST.get('user'))

task = Task.objects.create(

group = group,

receiver = user,

content = request.POST.get('content'),

status = 0

)

notification = Notification.objects.create(

receiver = user,

message = f"Вам пришла новая задача в группе #{group.name}",

priority = 3

)

return HttpResponseRedirect(reverse('group', args=[pk]))

@login\_required

def update\_task(request, pk):

""" Вью для изменения статуса выполнения задачи в конкретной группе """

if request.method == "POST":

task = get\_object\_or\_404(Task, pk=pk)

if task.status == 0:

task.status = 1

else:

task.status = 0

task.date\_of\_completion = datetime.now()

task.save()

return HttpResponseRedirect(reverse('group', args=[task.group.id]))

@login\_required

def remove\_task(request, pk):

""" Вью для удаления задачи из конкретной группы """

pass

@login\_required

def remove\_notification(request, pk):

""" Вьюшка для удаления уведомления """

if request.method == "POST":

notification = get\_object\_or\_404(Notification, pk=pk)

notification.delete()

return HttpResponseRedirect(reverse('index'))

@login\_required

def add\_notice(request, pk):

""" Вьюшка для добавления уведомления """

if request.method == "POST":

group = get\_object\_or\_404(models.Group, pk=pk)

notice = models.Notice(group=group, author=request.user.client, message=request.POST.get('message'))

notice.save()

for listener in notice.group.listeners.all():

notification = Notification.objects.create(

receiver = listener,

message = f'Новое объявление в группе {group.name}',

priority = 4

)

for moderator in notice.group.moderators.all():

notification = Notification.objects.create(

receiver = moderator,

message = f'Новое объявление в группе {group.name}',

priority = 4

)

return HttpResponseRedirect(reverse('group', args=[group.id]))

@login\_required

def remove\_notice(request, pk):

""" Вьюшка для удаления уведомления """

if request.method == "POST":

notice = get\_object\_or\_404(Notice, pk=pk)

group = get\_object\_or\_404(Group, pk=notice.group.id)

notice.delete()

return HttpResponseRedirect(reverse('group', args=[group.id]))

Urls.py