Приложение А. Текст программы

АННОТАЦИЯ

В данном программном документе приведен текст программы веб-приложения «DocsManage».

Текст программы включает в себя все необходимые компоненты для полноценного функционирования системы.

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc195005590)

[1.1. Наименование программы 3](#_Toc195005591)

[1.2. Область применения программы 3](#_Toc195005592)

[1.3. Модули 3](#_Toc195005593)

### ТЕКСТ ПРОГРАММЫ

### Наименование программы

#### Наименование – «DocsManage».

### Область применения программы

Приложение «DocsManage» предназначено для организации и управления внутренним документооборотом, обеспечивая удобную платформу для создания, согласования, подписания и отслеживания документов в учреждении.

### Модули

В таблице 1 представлено описание всех модулей приложения, количество строк верстки или кода и размер файла.

Таблица 1 - модули приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Модуль | Описание | Количество строк | Размер (Кб) |
| /Docs/admin.py | Регистрирует модели приложения Docs для админ-панели Django. | 6 | 0.21 |
| /Docs/apps.py | Содержит конфигурацию приложения Docs. | 6 | 0.14 |
| /Docs/forms.py | Определяет формы для создания и редактирования документов. | 96 | 05.фев |
| /Docs/migrations/0001\_initial.py | Содержит начальную миграцию для создания таблиц в базе данных. | 95 | 06.май |
| /Docs/migrations/\_\_init\_\_.py | Инициализирующий файл для пакета миграций. | 0 | 0.0 |
| /Docs/models.py | Определяет модели данных приложения Docs. | 130 | 06.фев |
| /Docs/pipelines.py | Содержит pipeline-функции для авторизации через Google OAuth2. | 17 | 0.75 |
| /Docs/serializers.py | Предназначен для сериализаторов, но не используется. | 27 | 0.91 |
| /Docs/views.py | Содержит представления приложения Docs для обработки запросов. | 540 | 21.27 |
| /Docs/\_\_init\_\_.py | Инициализирующий файл для пакета Docs. | 0 | 0.0 |
| /DocsManage/asgi.py | Определяет настройки ASGI для проекта. | 16 | 0.4 |
| /DocsManage/manage.py | Главный скрипт управления проектом Django. | 22 | 0.67 |
| /DocsManage/settings.py | Содержит настройки проекта (база данных, приложения, OAuth2). | 164 | апр.78 |
| /DocsManage/urls.py | Определяет маршруты проекта. | 37 | янв.78 |
| /DocsManage/wsgi.py | Определяет настройки WSGI для проекта. | 16 | 0.4 |
| /DocsManage/\_\_init\_\_.py | Инициализирующий файл для пакета DocsManage. | 0 | 0.0 |
| /TelegramBot/admin.py | Предназначен для регистрации моделей в админ-панели, но не используется. | 3 | 0.06 |
| /TelegramBot/apps.py | Содержит конфигурацию приложения TelegramBot. | 6 | 0.16 |
| /TelegramBot/bot.py | Реализует логику Telegram-бота (привязка аккаунта, уведомления). | 115 | май.25 |
| /TelegramBot/management/commands/runbot.py | Содержит команду runbot для запуска Telegram-бота. | 11 | 0.37 |
| /TelegramBot/management/commands/\_\_init\_\_.py | Инициализирующий файл для пакета команд. | 0 | 0.0 |
| /TelegramBot/management/\_\_init\_\_.py | Инициализирующий файл для пакета management. | 0 | 0.0 |
| /TelegramBot/migrations/\_\_init\_\_.py | Инициализирующий файл для пакета миграций. | 0 | 0.0 |
| /TelegramBot/models.py | Предназначен для моделей, но не используется. | 3 | 0.06 |
| /TelegramBot/tests.py | Предназначен для тестов, но не используется. | 3 | 0.06 |
| /TelegramBot/views.py | Предназначен для представлений, но не используется. | 3 | 0.06 |
| /TelegramBot/\_\_init\_\_.py | Инициализирующий файл для пакета TelegramBot. | 0 | 0.0 |
| /templates/base.html | Базовый HTML-шаблон с общей структурой страницы. | 50 | янв.89 |
| /templates/bind\_telegram.html | Шаблон страницы привязки Telegram-аккаунта. | 11 | 0.44 |
| /templates/documents/document\_detail.html | Шаблон страницы детального просмотра документа. | 122 | апр.67 |
| /templates/documents/document\_edit.html | Шаблон страницы редактирования документа. | 69 | 03.фев |
| /templates/documents/document\_form.html | Шаблон формы создания документа. | 46 | 02.июл |
| /templates/documents/document\_list.html | Шаблон страницы списка документов. | 139 | июл.22 |
| /templates/edit\_photo.html | Шаблон страницы редактирования фото профиля. | 13 | 0.52 |
| /templates/index.html | Шаблон главной страницы приложения. | 69 | 5.0 |
| /templates/login\_disabled.html | Шаблон страницы ошибки авторизации. | 11 | 0.53 |
| /templates/profile.html | Шаблон страницы профиля пользователя. | 71 | 03.мар |
| /templates/templated\_document/create\_templated\_document.html | Шаблон страницы создания документа по шаблону. | 30 | янв.19 |
| /templates/templated\_document/document\_created.html | Шаблон страницы после создания документа по шаблону. | 15 | 0.75 |
| /templates/templated\_document/select\_document\_type.html | Шаблон страницы выбора типа шаблона документа. | 26 | янв.16 |

/Docs/admin.py

from django.contrib import admin  
from .models import Document, DocumentFile, Profile  
# Register your models here.  
admin.site.register(Profile)  
admin.site.register(DocumentFile)  
admin.site.register(Document)

/Docs/apps.py

from django.apps import AppConfig  
  
  
class DocsConfig(AppConfig):  
 default\_auto\_field = 'django.db.models.BigAutoField'  
 name = 'Docs'

/Docs/forms.py

from django import forms  
from .models import Document, DocumentFile, Profile, DocumentComment  
from django.contrib.auth.models import User  
  
class ActDocumentForm(forms.Form):  
 act\_number = forms.CharField(label='Номер акта', max\_length=100)  
 city = forms.CharField(label='Город', max\_length=100)  
 date = forms.DateField(label='Дата', widget=forms.DateInput(attrs={'type': 'date'}))  
 title = forms.CharField(label='Заголовок', max\_length=200)  
 customer = forms.CharField(label='Заказчик', max\_length=100)  
 executor = forms.CharField(label='Исполнитель', max\_length=100)  
 work\_description = forms.CharField(label='Описание работы', widget=forms.Textarea)  
 work\_result = forms.CharField(label='Результат работы', widget=forms.Textarea)  
 commission\_members = forms.CharField(label='Члены комиссии', widget=forms.Textarea, help\_text="Список членов комиссии, разделенных запятыми.")  
  
class ContractDocumentForm(forms.Form):  
 contract\_number = forms.CharField(label='Номер договора', max\_length=100)  
 city = forms.CharField(label='Город', max\_length=100)  
 contract\_date = forms.DateField(label='Дата договора', widget=forms.DateInput(attrs={'type': 'date'}))  
 executor\_name = forms.CharField(label='Исполнитель', max\_length=100)  
 signature\_executor = forms.CharField(label='Подпись исполнителя', max\_length=100)  
 customer\_name = forms.CharField(label='Заказчик', max\_length=100)  
 signature\_customer = forms.CharField(label='Подпись заказчика', max\_length=100)  
 subject = forms.CharField(label='Предмет договора', widget=forms.Textarea)  
 price = forms.DecimalField(label='Стоимость услуг', max\_digits=10, decimal\_places=2)  
 payment\_terms = forms.CharField(label='Условия оплаты', widget=forms.Textarea)  
 contract\_term = forms.DateField(label='Срок действия договора', widget=forms.DateInput(attrs={'type': 'date'}))  
 executor\_details = forms.CharField(label='Реквизиты исполнителя', widget=forms.Textarea)  
 customer\_details = forms.CharField(label='Реквизиты заказчика', widget=forms.Textarea)  
  
class OrderDocumentForm(forms.Form):  
 document\_number = forms.CharField(label='Номер приказа', max\_length=100)  
 city = forms.CharField(label='Город', max\_length=100)  
 date = forms.DateField(label='Дата приказа', widget=forms.DateInput(attrs={'type': 'date'}))  
 title = forms.CharField(label='Заголовок', max\_length=200)  
 basis = forms.CharField(label='На основании', widget=forms.Textarea)  
 items = forms.CharField(label='Пункты приказа', widget=forms.Textarea, help\_text="Список пунктов приказа, разделенных запятыми.")  
 responsible\_person = forms.CharField(label='Ответственный за исполнение', max\_length=100)  
 director\_name = forms.CharField(label='Имя директора', max\_length=100)  
  
  
class DocumentForm(forms.ModelForm):  
 creators = forms.ModelMultipleChoiceField(  
 queryset=User.objects.none(),  
 widget=forms.CheckboxSelectMultiple,  
 required=False,  
 label='Инициаторы (кроме вас)',  
 )  
 signers = forms.ModelMultipleChoiceField(  
 queryset=User.objects.none(),  
 widget=forms.CheckboxSelectMultiple,  
 required=False,  
 label='Подписанты (кроме вас)',  
 )  
  
 class Meta:  
 model = Document  
 fields = ['title', 'description', 'creators', 'signers']  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 self.current\_user = kwargs.pop('current\_user')  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
  
 # Исключаем текущего пользователя из выбора  
 self.fields['creators'].queryset = User.objects.exclude(id=self.current\_user.id)  
 self.fields['signers'].queryset = User.objects.exclude(id=self.current\_user.id)  
  
  
class DocumentFileForm(forms.ModelForm):  
 class Meta:  
 model = DocumentFile  
 fields = ['file', 'name']  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['file'].required = True  
  
  
class ProfilePhotoForm(forms.ModelForm):  
 class Meta:  
 model = Profile  
 fields = ['photo'] # Только фото  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['photo'].widget.attrs.update({  
 'class': 'block w-full text-sm text-gray-500 file:mr-4 file:py-2 file:px-4 file:rounded-full file:border-0 file:text-sm file:font-semibold file:bg-blue-50 file:text-blue-700 hover:file:bg-blue-100'  
 })  
  
class DocumentCommentForm(forms.ModelForm):  
 class Meta:  
 model = DocumentComment  
 fields = ['message']  
 widgets = {  
 'message': forms.Textarea(attrs={'class': 'border p-2 rounded w-full', 'placeholder': 'Напишите комментарий...'})  
 }

/Docs/migrations/0001\_initial.py

# Generated by Django 5.2 on 2025-04-07 17:48  
  
import django.db.models.deletion  
from django.conf import settings  
from django.db import migrations, models  
  
  
class Migration(migrations.Migration):  
  
 initial = True  
  
 dependencies = [  
 migrations.swappable\_dependency(settings.AUTH\_USER\_MODEL),  
 ]  
  
 operations = [  
 migrations.CreateModel(  
 name='DocumentTemplate',  
 fields=[  
 ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),  
 ('name', models.CharField(max\_length=255, verbose\_name='Название шаблона')),  
 ('document\_type', models.CharField(choices=[('order', 'Приказ'), ('contract', 'Договор'), ('act', 'Акт')], max\_length=50, verbose\_name='Тип документа')),  
 ('template\_file', models.FileField(upload\_to='document\_templates/', verbose\_name='Файл шаблона')),  
 ('description', models.TextField(blank=True, verbose\_name='Описание')),  
 ('created\_at', models.DateTimeField(auto\_now\_add=True)),  
 ('updated\_at', models.DateTimeField(auto\_now=True)),  
 ('fields\_metadata', models.JSONField(default=dict, help\_text='JSON структура с описанием полей для заполнения', verbose\_name='Метаданные полей')),  
 ],  
 options={  
 'verbose\_name': 'Шаблон документа',  
 'verbose\_name\_plural': 'Шаблоны документов',  
 },  
 ),  
 migrations.CreateModel(  
 name='Document',  
 fields=[  
 ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),  
 ('title', models.CharField(max\_length=255, verbose\_name='Название документа')),  
 ('description', models.CharField(blank=True, max\_length=500, null=True, verbose\_name='Описание документа')),  
 ('status', models.CharField(choices=[('accept', 'Принят'), ('signed', 'Подписан'), ('pending', 'На подписании'), ('revision', 'На доработке')], default='pending', max\_length=50, verbose\_name='Статус документа')),  
 ('created\_at', models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')),  
 ('updated\_at', models.DateTimeField(auto\_now=True, verbose\_name='Дата изменения')),  
 ('creators', models.ManyToManyField(related\_name='created\_documents', to=settings.AUTH\_USER\_MODEL, verbose\_name='Создатели')),  
 ('signed\_by', models.ManyToManyField(blank=True, related\_name='signed\_documents', to=settings.AUTH\_USER\_MODEL, verbose\_name='Подписавшие')),  
 ('signers', models.ManyToManyField(related\_name='documents\_to\_sign', to=settings.AUTH\_USER\_MODEL, verbose\_name='Подписанты')),  
 ],  
 options={  
 'verbose\_name': 'Документ',  
 'verbose\_name\_plural': 'Документы',  
 },  
 ),  
 migrations.CreateModel(  
 name='DocumentComment',  
 fields=[  
 ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),  
 ('message', models.TextField(verbose\_name='Сообщение')),  
 ('created\_at', models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')),  
 ('is\_system', models.BooleanField(default=False)),  
 ('document', models.ForeignKey(on\_delete=django.db.models.deletion.CASCADE, related\_name='comments', to='Docs.document', verbose\_name='Документ')),  
 ('user', models.ForeignKey(on\_delete=django.db.models.deletion.CASCADE, to=settings.AUTH\_USER\_MODEL, verbose\_name='Пользователь')),  
 ],  
 options={  
 'verbose\_name': 'Комментарий к документу',  
 'verbose\_name\_plural': 'Комментарии к документам',  
 },  
 ),  
 migrations.CreateModel(  
 name='DocumentFile',  
 fields=[  
 ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),  
 ('file', models.FileField(upload\_to='documents/%Y/%m/%d/', verbose\_name='Файл')),  
 ('name', models.CharField(max\_length=255, verbose\_name='Имя файла')),  
 ('uploaded\_at', models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата загрузки')),  
 ('status', models.CharField(choices=[('uploaded', 'Загружен'), ('deleted', 'Удален')], default='uploaded', max\_length=50, verbose\_name='Статус файла')),  
 ('document', models.ForeignKey(on\_delete=django.db.models.deletion.CASCADE, related\_name='files', to='Docs.document', verbose\_name='Документ')),  
 ],  
 options={  
 'verbose\_name': 'Файл документа',  
 'verbose\_name\_plural': 'Файлы документов',  
 },  
 ),  
 migrations.CreateModel(  
 name='Profile',  
 fields=[  
 ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),  
 ('first\_name', models.CharField(max\_length=100, verbose\_name='Имя')),  
 ('last\_name', models.CharField(blank=True, max\_length=100, null=True, verbose\_name='Фамилия')),  
 ('photo', models.ImageField(blank=True, null=True, upload\_to='profile\_pics/', verbose\_name='Фото профиля')),  
 ('role', models.CharField(choices=[('employee', 'Сотрудник'), ('admin', 'Администратор'), ('director', 'Директор'), ('deputy\_director', 'Заместитель директора'), ('secretary', 'Секретарь')], default='employee', max\_length=50)),  
 ('telegram\_id', models.CharField(blank=True, max\_length=100, null=True)),  
 ('telegram\_token', models.CharField(blank=True, max\_length=100, null=True)),  
 ('user', models.OneToOneField(on\_delete=django.db.models.deletion.CASCADE, related\_name='profile', to=settings.AUTH\_USER\_MODEL)),  
 ],  
 ),  
 ]

/Docs/migrations/\_\_init\_\_.py

/Docs/models.py

import secrets  
from django.contrib.auth.models import User  
from django.db import models  
  
  
class Profile(models.Model):  
 # Связь с моделью пользователя  
 user = models.OneToOneField(User, on\_delete=models.CASCADE, related\_name='profile')  
 first\_name = models.CharField(max\_length=100, verbose\_name='Имя')  
 last\_name = models.CharField(max\_length=100, blank=True, null=True, verbose\_name='Фамилия')  
 photo = models.ImageField(upload\_to='profile\_pics/', blank=True, null=True, verbose\_name='Фото профиля')  
  
 # Роль пользователя  
 ROLE\_CHOICES = [  
 ('employee', 'Сотрудник'),  
 ('admin', 'Администратор'),  
 ('director', 'Директор'),  
 ('deputy\_director', 'Заместитель директора'),  
 ('secretary', 'Секретарь'),  
 ]  
 role = models.CharField(max\_length=50, choices=ROLE\_CHOICES, default='employee')  
  
 # Telegram ID (может быть пустым)  
 telegram\_id = models.CharField(max\_length=100, blank=True, null=True)  
 telegram\_token = models.CharField(max\_length=100, blank=True, null=True) # Новое поле для токена  
  
 def generate\_telegram\_token(self):  
 self.telegram\_token = secrets.token\_urlsafe(16) # Генерация случайного токена  
 self.save()  
  
 def clear\_telegram\_token(self):  
 self.telegram\_token = None  
 self.save()  
  
 def \_\_str\_\_(self):  
 return f"{self.user.username} - {self.role}"  
  
  
class Document(models.Model):  
 # Статусы документа  
 STATUS\_CHOICES = [  
 ('accept', 'Принят'),  
 ('signed', 'Подписан'),  
 ('pending', 'На подписании'),  
 ('revision', 'На доработке'),  
 ]  
  
 # Основные поля  
 title = models.CharField(max\_length=255, verbose\_name='Название документа')  
 description = models.CharField(max\_length=500, verbose\_name='Описание документа', blank=True, null=True)  
 status = models.CharField(max\_length=50, choices=STATUS\_CHOICES, default='pending', verbose\_name='Статус документа')  
 created\_at = models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')  
 updated\_at = models.DateTimeField(auto\_now=True, verbose\_name='Дата изменения')  
  
 # Создатели документа (может быть несколько)  
 creators = models.ManyToManyField(User, related\_name='created\_documents', verbose\_name='Создатели')  
  
 # Подписанты (пользователи, которые должны подписать документ)  
 signers = models.ManyToManyField(User, related\_name='documents\_to\_sign', verbose\_name='Подписанты')  
  
 # Пользователи, которые уже подписали документ  
 signed\_by = models.ManyToManyField(User, related\_name='signed\_documents', verbose\_name='Подписавшие', blank=True)  
  
 def \_\_str\_\_(self):  
 return self.title  
  
 class Meta:  
 verbose\_name = 'Документ'  
 verbose\_name\_plural = 'Документы'  
  
class DocumentFile(models.Model):  
 # Статусы файла  
 FILE\_STATUS\_CHOICES = [  
 ('uploaded', 'Загружен'),  
 ('deleted', 'Удален'),  
 ]  
  
 # Основные поля  
 document = models.ForeignKey(Document, on\_delete=models.CASCADE, related\_name='files', verbose\_name='Документ')  
 file = models.FileField(upload\_to='documents/%Y/%m/%d/', verbose\_name='Файл')  
 name = models.CharField(max\_length=255, verbose\_name='Имя файла')  
 uploaded\_at = models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата загрузки')  
 status = models.CharField(max\_length=50, choices=FILE\_STATUS\_CHOICES, default='uploaded', verbose\_name='Статус файла')  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 class Meta:  
 verbose\_name = 'Файл документа'  
 verbose\_name\_plural = 'Файлы документов'  
  
class DocumentComment(models.Model):  
 document = models.ForeignKey(Document, on\_delete=models.CASCADE, related\_name='comments', verbose\_name='Документ')  
 user = models.ForeignKey(User, on\_delete=models.CASCADE, verbose\_name='Пользователь')  
 message = models.TextField(verbose\_name='Сообщение')  
 created\_at = models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')  
 is\_system = models.BooleanField(default=False)  
  
 def \_\_str\_\_(self):  
 return f"{self.user.get\_full\_name()} - {self.created\_at}"  
  
 class Meta:  
 verbose\_name = 'Комментарий к документу'  
 verbose\_name\_plural = 'Комментарии к документам'  
  
  
class DocumentTemplate(models.Model):  
 DOCUMENT\_TYPES = [  
 ('order', 'Приказ'),  
 ('contract', 'Договор'),  
 ('act', 'Акт'),  
 ]  
 name = models.CharField(max\_length=255, verbose\_name="Название шаблона")  
 document\_type = models.CharField(max\_length=50, choices=DOCUMENT\_TYPES, verbose\_name="Тип документа")  
 template\_file = models.FileField(upload\_to='document\_templates/', verbose\_name="Файл шаблона")  
 description = models.TextField(blank=True, verbose\_name="Описание")  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
 updated\_at = models.DateTimeField(auto\_now=True)  
 fields\_metadata = models.JSONField(  
 verbose\_name="Метаданные полей",  
 help\_text="JSON структура с описанием полей для заполнения",  
 default=dict  
 )  
  
 def \_\_str\_\_(self):  
 return f"{self.get\_document\_type\_display()} - {self.name}"  
  
 class Meta:  
 verbose\_name = 'Шаблон документа'  
 verbose\_name\_plural = 'Шаблоны документов'

/Docs/pipelines.py

from django.contrib.auth.models import User  
from .models import Profile  
  
def create\_user\_profile(backend, user, response, \*args, \*\*kwargs):  
 if backend.name == 'google-oauth2':  
 profile, created = Profile.objects.get\_or\_create(user=user)  
  
 if created:  
 profile.first\_name = response.get('given\_name', user.first\_name)  
 profile.last\_name = response.get('family\_name', user.last\_name)  
 profile.role = 'employee' # Роль по умолчанию - "сотрудник"  
  
 picture = response.get('picture')  
 if picture:  
 profile.photo = picture # Здесь лучше использовать сохранение изображения  
  
 profile.save()

/Docs/serializers.py

from rest\_framework import serializers  
from django.contrib.auth.models import User  
from .models import Profile, Document, DocumentFile  
  
  
class ProfileSerializer(serializers.ModelSerializer):  
 user = serializers.StringRelatedField() # Отображаем имя пользователя  
  
 class Meta:  
 model = Profile  
 fields = '\_\_all\_\_'  
  
  
class DocumentSerializer(serializers.ModelSerializer):  
 creators = serializers.PrimaryKeyRelatedField(many=True, queryset=User.objects.all())  
 signers = serializers.PrimaryKeyRelatedField(many=True, queryset=User.objects.all())  
 signed\_by = serializers.PrimaryKeyRelatedField(many=True, queryset=User.objects.all(), required=False)  
  
 class Meta:  
 model = Document  
 fields = '\_\_all\_\_'  
  
  
class DocumentFileSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = DocumentFile  
 fields = '\_\_all\_\_'

/Docs/views.py

import secrets  
from django.contrib.auth import logout  
from django.views import View  
from .forms import \*  
from django.shortcuts import get\_object\_or\_404, redirect  
from django.contrib.auth.decorators import login\_required, user\_passes\_test  
from django.contrib import messages  
from django.core.paginator import Paginator  
from .forms import DocumentForm  
from django.shortcuts import render  
from docx import Document as DocxDocument  
from django.conf import settings  
import os  
from django.urls import reverse  
from urllib.parse import unquote  
from datetime import datetime  
from reportlab.lib.pagesizes import A4  
from reportlab.lib import colors  
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle  
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle  
from reportlab.pdfbase.ttfonts import TTFont  
from reportlab.pdfbase import pdfmetrics  
from django.http import HttpResponse  
from .models import Document  
class CustomLogoutView(View):  
 def get(self, request):  
 logout(request)  
 return redirect('login') # или куда нужно  
  
def login\_disabled\_view(request):  
 return render(request, 'login\_disabled.html')  
  
def home(request):  
 return render(request, 'index.html')  
  
  
# Функция для проверки роли "Секретарь"  
def is\_secretary(user):  
 return user.profile.role == 'secretary'  
  
  
def export\_documents\_pdf(request):  
 # Проверка роли  
 if not is\_secretary(request.user):  
 return HttpResponse('У вас нет прав для доступа к этому ресурсу.', status=403)  
  
 # Ответ с PDF  
 response = HttpResponse(content\_type='application/pdf')  
 response['Content-Disposition'] = 'attachment; filename="documents.pdf"'  
  
 # Создаем буфер  
 doc = SimpleDocTemplate(response, pagesize=A4)  
  
 # Регистрируем шрифт Times New Roman  
 pdfmetrics.registerFont(TTFont('TimesNewRoman', 'C:/Windows/Fonts/times.ttf'))  
  
 # Получаем все документы  
 documents = Document.objects.all().order\_by('-created\_at')  
  
 # Данные для таблицы  
 data = [['ID', 'Название', 'Статус', 'Создан']]  
  
 # Заполняем таблицу данными  
 for doc\_item in documents:  
 data.append([  
 str(doc\_item.id),  
 doc\_item.title, # Используем поле title  
 doc\_item.status, # Используем поле status  
 doc\_item.created\_at.strftime('%d.%m.%Y')  
 ])  
  
 # Стиль для таблицы и текста  
 style = getSampleStyleSheet()  
 style['Normal'].fontName = 'TimesNewRoman'  
 style['Normal'].fontSize = 12  
  
 # Создаем таблицу  
 table = Table(data, colWidths=[40, 150, 80, 80])  
 table.setStyle(TableStyle([  
 ('BACKGROUND', (0, 0), (-1, 0), colors.grey),  
 ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),  
 ('ALIGN', (0, 0), (-1, -1), 'CENTER'),  
 ('FONTNAME', (0, 0), (-1, 0), 'TimesNewRoman'),  
 ('BOTTOMPADDING', (0, 0), (-1, 0), 10),  
 ('BACKGROUND', (0, 1), (-1, -1), colors.beige),  
 ('GRID', (0, 0), (-1, -1), 1, colors.black),  
 ]))  
  
 # Добавляем таблицу в документ  
 elements = [table]  
 doc.build(elements)  
  
 return response  
  
  
@login\_required  
def bind\_telegram(request):  
 # Генерируем токен  
 token = secrets.token\_urlsafe(16)  
  
 profile = request.user.profile  
 profile.telegram\_token = token  
 profile.save()  
  
 bot\_username = "Shool654Bot"  
 deep\_link = f"https://t.me/{bot\_username}?start={token}"  
  
 return render(request, 'bind\_telegram.html', {  
 'deep\_link': deep\_link  
 })  
  
  
@login\_required  
def select\_document\_type(request):  
 # Показываем страницу с выбором типа документа  
 return render(request, 'templated\_document/select\_document\_type.html')  
  
@login\_required  
def create\_templated\_document(request, document\_type):  
 if document\_type == 'act':  
 form\_class = ActDocumentForm  
 elif document\_type == 'contract':  
 form\_class = ContractDocumentForm  
 elif document\_type == 'order':  
 form\_class = OrderDocumentForm  
 else:  
 return HttpResponse('Тип документа не найден', status=400)  
  
 if request.method == 'POST':  
 form = form\_class(request.POST)  
 if form.is\_valid():  
 data = form.cleaned\_data  
  
 # Определение шаблона и имени файла  
 if document\_type == 'act':  
 template\_path = os.path.join(settings.BASE\_DIR, 'templates', 'act\_template.docx')  
 identifier = data['act\_number']  
 elif document\_type == 'contract':  
 template\_path = os.path.join(settings.BASE\_DIR, 'templates', 'contract\_template.docx')  
 identifier = data['contract\_number']  
 elif document\_type == 'order':  
 template\_path = os.path.join(settings.BASE\_DIR, 'templates', 'order\_template.docx')  
 identifier = data['document\_number']  
  
 # Открытие шаблона  
 doc = DocxDocument(template\_path)  
  
 # Замена плейсхолдеров  
 for para in doc.paragraphs:  
 text = para.text  
  
 if document\_type == 'act':  
 members = [m.strip() for m in data.get('commission\_members', '').split(',')]  
 members\_list = "\n".join(f"- {m}" for m in members)  
 member\_signatures = "\n".join(f"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / {m} /" for m in members)  
  
 replacements = {  
 '{{ act\_number }}': data['act\_number'],  
 '{{ city }}': data['city'],  
 '{{ date }}': str(data['date']),  
 '{{ title }}': data['title'],  
 '{{ customer }}': data['customer'],  
 '{{ executor }}': data['executor'],  
 '{{ work\_description }}': data['work\_description'],  
 '{{ work\_result }}': data['work\_result'],  
 '{{ commission\_members\_list }}': members\_list,  
 '{{ commission\_signatures }}': member\_signatures,  
 }  
  
 elif document\_type == 'contract':  
 replacements = {  
 '{{ contract\_number }}': data['contract\_number'],  
 '{{ city }}': data['city'],  
 '{{ contract\_date }}': str(data['contract\_date']),  
 '{{ executor\_name }}': data['executor\_name'],  
 '{{ signature\_executor }}': data['signature\_executor'],  
 '{{ customer\_name }}': data['customer\_name'],  
 '{{ signature\_customer }}': data['signature\_customer'],  
 '{{ subject }}': data['subject'],  
 '{{ price }}': str(data['price']),  
 '{{ payment\_terms }}': data['payment\_terms'],  
 '{{ contract\_term }}': str(data['contract\_term']),  
 '{{ executor\_details }}': data['executor\_details'],  
 '{{ customer\_details }}': data['customer\_details'],  
 }  
  
 elif document\_type == 'order':  
 items = [i.strip() for i in data.get('items', '').split(',')]  
 items\_list = "\n".join(f"{index + 1}. {item}" for index, item in enumerate(items))  
  
 replacements = {  
 '{{ document\_number }}': data['document\_number'],  
 '{{ city }}': data['city'],  
 '{{ date }}': str(data['date']),  
 '{{ title }}': data['title'],  
 '{{ basis }}': data['basis'],  
 '{{ items\_list }}': items\_list,  
 '{{ responsible\_person }}': data['responsible\_person'],  
 '{{ director\_name }}': data['director\_name'],  
 }  
  
 for key, value in replacements.items():  
 text = text.replace(key, value)  
 para.text = text  
  
 # Сохраняем файл  
 today = datetime.today()  
 year = today.year  
 month = today.month  
 day = today.day  
 filename = f"{identifier}\_{today.strftime('%Y-%m-%d')}.docx"  
 save\_dir = os.path.join(settings.MEDIA\_ROOT, 'documents', str(year), str(month), str(day))  
 os.makedirs(save\_dir, exist\_ok=True)  
 save\_path = os.path.join(save\_dir, filename)  
 doc.save(save\_path)  
  
 file\_url = f"/media/documents/{year}/{month}/{day}/{filename}"  
 return redirect(f"{reverse('document\_created')}?file\_url={file\_url}")  
  
 else:  
 form = form\_class()  
  
 return render(request, 'templated\_document/create\_templated\_document.html', {'form': form})  
  
  
@login\_required  
def document\_created(request):  
 file\_url = request.GET.get('file\_url')  
 if file\_url:  
 file\_url = unquote(file\_url)  
 return render(request, 'templated\_document/document\_created.html', {'file\_url': file\_url})  
  
  
@login\_required  
def create\_document(request):  
 if request.method == 'POST':  
 form = DocumentForm(request.POST, current\_user=request.user)  
 files = request.FILES.getlist('files')  
  
 if form.is\_valid():  
 document = form.save(commit=False)  
 document.save()  
  
 # Получаем списки из формы  
 creators = list(form.cleaned\_data['creators'])  
 signers = list(form.cleaned\_data['signers'])  
  
 # Добавляем текущего пользователя, если его нет  
 if request.user not in creators:  
 creators.append(request.user)  
  
 # Добавляем всех инициаторов в подписанты  
 for creator in creators:  
 if creator not in signers:  
 signers.append(creator)  
  
 # Сохраняем  
 document.creators.set(creators)  
 document.signers.set(signers)  
  
 # Загрузка файлов  
 for file in files:  
 DocumentFile.objects.create(  
 document=document,  
 file=file,  
 name=file.name  
 )  
  
 return redirect('document\_detail', pk=document.pk)  
 else:  
 form = DocumentForm(current\_user=request.user)  
  
 return render(request, 'documents/document\_form.html', {'form': form})  
  
  
@login\_required  
def edit\_document(request, pk):  
 document = get\_object\_or\_404(Document, pk=pk)  
  
 if request.user not in document.creators.all():  
 messages.error(request, "Вы не являетесь инициатором этого документа.")  
 return redirect('document\_detail', pk=pk)  
  
 if request.method == 'POST':  
 form = DocumentForm(request.POST, instance=document, current\_user=request.user)  
 files = request.FILES.getlist('files')  
 comment\_form = DocumentCommentForm(request.POST)  
  
 # Проверка на кнопку "Отправить на доработку"  
 if 'send\_to\_revision' in request.POST:  
 # Если добавлены или изменены файлы, отправляем на доработку  
 document.status = 'revision'  
 document.signed\_by.clear()  
 messages.info(request, "Документ отправлен на доработку, подписи сброшены.")  
  
 if form.is\_valid():  
 document = form.save(commit=False)  
  
 # Применяем изменения, если статус не изменен на "revision"  
 if document.status != 'revision' and not files:  
 document.status = 'pending'  
  
 # Сохраняем документ  
 document.save()  
  
 # Обновление инициаторов и подписантов  
 creators = list(form.cleaned\_data['creators'])  
 signers = list(form.cleaned\_data['signers'])  
  
 if request.user not in creators:  
 creators.append(request.user)  
  
 for creator in creators:  
 if creator not in signers:  
 signers.append(creator)  
  
 document.creators.set(creators)  
 document.signers.set(signers)  
  
 # Сохранение новых файлов, если они есть  
 for file in files:  
 DocumentFile.objects.create(  
 document=document,  
 file=file,  
 name=file.name  
 )  
  
 # Системный комментарий, если отправлено на доработку  
 if 'send\_to\_revision' in request.POST:  
 DocumentComment.objects.create(  
 document=document,  
 user=request.user,  
 message="Документ отправлен на доработку",  
 is\_system=True  
 )  
  
 # Сохраняем комментарий от пользователя, если форма отправлена  
 if comment\_form.is\_valid():  
 comment = comment\_form.save(commit=False)  
 comment.document = document  
 comment.user = request.user  
 comment.save()  
  
 messages.success(request, "Документ успешно обновлён.")  
 return redirect('document\_detail', pk=document.pk)  
 else:  
 messages.error(request, "Пожалуйста, исправьте ошибки в форме.")  
 else:  
 form = DocumentForm(instance=document, current\_user=request.user)  
 comment\_form = DocumentCommentForm()  
  
 files = document.files.filter(status='uploaded')  
 is\_creator = request.user in document.creators.all()  
  
 return render(request, 'documents/document\_edit.html', {  
 'form': form,  
 'files': files,  
 'document': document,  
 'comment\_form': comment\_form,  
 'is\_creator': is\_creator,  
 })  
  
  
@login\_required  
def document\_detail(request, pk):  
 document = get\_object\_or\_404(Document, pk=pk)  
 user = request.user  
  
 is\_signer = user in document.signers.all()  
 has\_signed = user in document.signed\_by.all()  
 can\_sign = is\_signer and not has\_signed  
  
 is\_director = hasattr(user, 'profile') and user.profile.role == 'director'  
 all\_signed = document.signers.count() == document.signed\_by.count()  
  
 # Обработка добавления комментария  
 comment\_form = DocumentCommentForm(request.POST or None)  
 if request.method == 'POST':  
 # Подписание документа  
 if 'sign' in request.POST and can\_sign:  
 document.signed\_by.add(user)  
 # Добавляем системное сообщение о подписании  
 DocumentComment.objects.create(  
 document=document,  
 user=user,  
 message=f"Документ подписан пользователем {user.get\_full\_name()}",  
 is\_system=True  
 )  
 messages.success(request, "Документ подписан.")  
  
 # Принятие документа директором  
 elif 'accept' in request.POST and is\_director and document.status == 'signed':  
 document.status = 'accept'  
 document.save()  
 # Добавляем системное сообщение о принятии  
 DocumentComment.objects.create(  
 document=document,  
 user=user,  
 message=f"Документ принят директором {user.get\_full\_name()}",  
 is\_system=True  
 )  
 messages.success(request, "Документ принят директором.")  
  
 # Обновляем статус, если все подписали, но не директор ещё  
 if document.status != 'accept' and document.signers.count() == document.signed\_by.count():  
 document.status = 'signed'  
 document.save()  
 # Добавляем системное сообщение о статусе  
 DocumentComment.objects.create(  
 document=document,  
 user=user,  
 message="Все подписанты подписали документ. Статус обновлён на 'Подписан'.",  
 is\_system=True  
 )  
  
 # Обработка отправки комментария  
 if comment\_form.is\_valid():  
 comment = comment\_form.save(commit=False)  
 comment.document = document  
 comment.user = user  
 comment.save()  
 messages.success(request, "Комментарий успешно добавлен.")  
 return redirect('document\_detail', pk=pk)  
  
 # Получаем все комментарии к документу  
 comments = document.comments.all()  
  
 return render(request, 'documents/document\_detail.html', {  
 'document': document,  
 'is\_signer': is\_signer,  
 'has\_signed': has\_signed,  
 'can\_sign': can\_sign,  
 'is\_director': is\_director,  
 'all\_signed': all\_signed,  
 'comment\_form': comment\_form,  
 'comments': comments,  
 })  
  
  
  
@login\_required  
def delete\_file(request, file\_id):  
 file = get\_object\_or\_404(DocumentFile, id=file\_id)  
 file.status = 'deleted'  
 file.save()  
 return redirect('edit\_document', pk=file.document.pk)  
  
  
@login\_required  
def document\_list(request):  
 status\_filter = request.GET.get('status', 'all')  
 academic\_year\_filter = request.GET.get('academic\_year', '')  
 creator\_filter = request.GET.get('creator', '')  
 signer\_filter = request.GET.get('signer', '')  
  
 # Все документы, без фильтрации по пользователю  
 documents = Document.objects.all()  
  
 if status\_filter != 'all':  
 documents = documents.filter(status=status\_filter)  
  
 # Фильтрация по учебному году  
 if academic\_year\_filter:  
 start\_year, end\_year = map(int, academic\_year\_filter.split('-'))  
 documents = documents.filter(created\_at\_\_year\_\_gte=start\_year, created\_at\_\_year\_\_lte=end\_year)  
  
 # Фильтрация по инициатору  
 if creator\_filter:  
 documents = documents.filter(creators\_\_id=creator\_filter)  
  
 # Фильтрация по подписанту  
 if signer\_filter:  
 documents = documents.filter(signers\_\_id=signer\_filter)  
  
 # Пагинация  
 paginator = Paginator(documents, 10) # 10 документов на страницу  
 page\_number = request.GET.get('page') # Получаем номер страницы из GET-запроса  
 page\_obj = paginator.get\_page(page\_number)  
  
 # Список учебных годов  
 years = [(year, year + 1) for year in range(2020, 2025)]  
  
 # Получаем всех пользователей, которых можно выбрать для фильтрации  
 creators = User.objects.all()  
 signers = User.objects.all()  
  
 return render(request, 'documents/document\_list.html', {  
 'documents': page\_obj,  
 'status\_filter': status\_filter,  
 'years': years,  
 'creators': creators,  
 'signers': signers,  
 })  
  
  
def profile\_view(request):  
 """  
 Отображение профиля пользователя.  
 """  
 user = request.user  
 profile = user.profile  
 documents = Document.objects.filter(creators=user)  
  
 context = {  
 'user': user,  
 'profile': profile,  
 'documents': documents,  
 }  
  
 return render(request, 'profile.html', context)  
  
  
@login\_required  
def profile\_edit(request):  
 profile = request.user.profile  
  
 if request.method == 'POST':  
 form = ProfilePhotoForm(request.POST, request.FILES, instance=profile)  
 if form.is\_valid():  
 form.save()  
 return redirect('profile') # или куда нужно  
 else:  
 form = ProfilePhotoForm(instance=profile)  
  
 return render(request, 'edit\_photo.html', {'form': form})  
  
  
@login\_required  
def unbind\_telegram(request):  
 """Отвязываем Telegram аккаунт от профиля пользователя."""  
 profile = request.user.profile  
  
 if profile.telegram\_id:  
 profile.telegram\_id = None # Убираем Telegram ID  
 profile.save()  
 messages.success(request, "Ваш Telegram аккаунт успешно отвязан.")  
 else:  
 messages.warning(request, "Ваш Telegram аккаунт не был привязан.")  
  
 return redirect('profile')

/Docs/\_\_init\_\_.py

/DocsManage/asgi.py

"""  
ASGI config for DocsManage project.  
  
It exposes the ASGI callable as a module-level variable named ``application``.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/5.1/howto/deployment/asgi/  
"""  
  
import os  
  
from django.core.asgi import get\_asgi\_application  
  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'DocsManage.settings')  
  
application = get\_asgi\_application()

/DocsManage/manage.py

#!/usr/bin/env python  
"""Django's command-line utility for administrative tasks."""  
import os  
import sys  
  
  
def main():  
 """Run administrative tasks."""  
 os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'DocsManage.settings')  
 try:  
 from django.core.management import execute\_from\_command\_line  
 except ImportError as exc:  
 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?"  
 ) from exc  
 execute\_from\_command\_line(sys.argv)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

/DocsManage/settings.py

"""  
Django settings for DocsManage project.  
  
Generated by 'django-admin startproject' using Django 5.1.7.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/5.1/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/5.1/ref/settings/  
"""  
  
from pathlib import Path  
import os  
# Build paths inside the project like this: BASE\_DIR / 'subdir'.  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-^6=y!u9i0e+-j&\_#rbj1ez^radz$i@)-ex-yj@0ffqd\*ak9owp'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_KEY='565895263494-0pu97nictl5e9sdds11r3572obbl8bcp.apps.googleusercontent.com'  
SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_SECRET = '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'  
SOCIAL\_AUTH\_GOOGLE\_OAUTH2\_REDIRECT\_URI = 'http://localhost:8000/complete/google-oauth2/'  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'social\_django', # Add this Line  
 'Docs', # Add your application  
 'TelegramBot',  
 'rest\_framework',  
 'drf\_yasg', # Добавляем drf\_yasg  
 'tailwind',  
 'psycopg2',  
]  
  
AUTHENTICATION\_BACKENDS = [  
 'social\_core.backends.google.GoogleOAuth2',  
 'django.contrib.auth.backends.ModelBackend',  
]  
  
SOCIAL\_AUTH\_PIPELINE = (  
 'social\_core.pipeline.social\_auth.social\_details',  
 'social\_core.pipeline.social\_auth.social\_uid',  
 'social\_core.pipeline.social\_auth.auth\_allowed',  
 'social\_core.pipeline.social\_auth.social\_user',  
 'social\_core.pipeline.user.get\_username',  
 'social\_core.pipeline.user.create\_user',  
 'social\_core.pipeline.social\_auth.associate\_user',  
 'social\_core.pipeline.social\_auth.load\_extra\_data',  
 'social\_core.pipeline.user.user\_details',  
 'Docs.pipelines.create\_user\_profile',  
)  
  
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',  
 'social\_django.middleware.SocialAuthExceptionMiddleware',  
]  
  
ROOT\_URLCONF = 'DocsManage.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [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',  
 'social\_django.context\_processors.backends',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'DocsManage.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'NAME': 'DocsManage',  
 'USER': 'postgres',  
 'PASSWORD': 'postgres',  
 'HOST': 'localhost',  
 'PORT': '5432',  
 }  
}  
  
  
  
# Password validation  
# https://docs.djangoproject.com/en/5.1/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/5.1/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/5.1/howto/static-files/  
  
STATIC\_URL = 'static/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'  
  
MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')  
LOGOUT\_REDIRECT\_URL = '/'  
LOGIN\_URL = '/'

/DocsManage/urls.py

from django.contrib import admin  
from django.urls import path, include  
from django.contrib.auth import views as auth\_views  
from Docs import views  
from Docs.views import bind\_telegram  
from django.conf import settings  
from django.conf.urls.static import static  
from django.urls import path, include  
from django.http import HttpResponse  
docspatterns = [  
 path('', views.document\_list, name='document\_list'),  
 path('create/', views.create\_document, name='create\_document'),  
 path('select\_document\_type/', views.select\_document\_type, name='select\_document\_type'),  
 path('create\_templated/<str:document\_type>/', views.create\_templated\_document, name='create\_templated\_document'),  
 path('edit/<int:pk>/', views.edit\_document, name='edit\_document'),  
 path('file/delete/<int:file\_id>/', views.delete\_file, name='delete\_file'),  
 path('<int:pk>/', views.document\_detail, name='document\_detail'),  
 path('document\_created/', views.document\_created, name='document\_created'),  
 path('export/pdf/', views.export\_documents\_pdf, name='export\_documents\_pdf'),  
]  
  
urlpatterns = [  
 path('profile/', views.profile\_view, name='profile'),  
 path('profile/edit/', views.profile\_edit, name='edit\_profile'),  
 path("docs/", include(docspatterns)),  
 path("bind\_telegram/", bind\_telegram, name="bind\_telegram"),  
 path('unbind\_telegram/', views.unbind\_telegram, name='unbind\_telegram'),  
 path('admin/', admin.site.urls),  
 path('logout/', views.CustomLogoutView.as\_view(), name='logout'),  
 path('social-auth/', include('social\_django.urls', namespace='social')),  
 path("", views.home, name='index'),  
 path('login/', views.login\_disabled\_view, name='login'),  
  
]  
  
if settings.DEBUG:  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

/DocsManage/wsgi.py

"""  
WSGI config for DocsManage 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/5.1/howto/deployment/wsgi/  
"""  
  
import os  
  
from django.core.wsgi import get\_wsgi\_application  
  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'DocsManage.settings')  
  
application = get\_wsgi\_application()

/DocsManage/\_\_init\_\_.py

/TelegramBot/admin.py

from django.contrib import admin  
  
# Register your models here.

/TelegramBot/apps.py

from django.apps import AppConfig  
  
  
class TelegrambotConfig(AppConfig):  
 default\_auto\_field = 'django.db.models.BigAutoField'  
 name = 'TelegramBot'

/TelegramBot/bot.py

import logging  
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup  
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, ContextTypes  
from asgiref.sync import sync\_to\_async  
from Docs.models import Profile  
  
# Настройка логирования  
logging.basicConfig(  
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',  
 level=logging.INFO  
)  
logger = logging.getLogger(\_\_name\_\_)  
  
BOT\_TOKEN = "8021983986:AAEAd83O01vWEGJBVsp0\_DA4tNxvpG9Tdp8"  
  
@sync\_to\_async  
def get\_profile\_by\_code(code: str):  
 try:  
 return Profile.objects.get(telegram\_token=code)  
 except Profile.DoesNotExist:  
 return None  
  
@sync\_to\_async  
def get\_profile\_by\_telegram\_id(telegram\_id: str):  
 try:  
 return Profile.objects.get(telegram\_id=telegram\_id)  
 except Profile.DoesNotExist:  
 return None  
  
@sync\_to\_async  
def save\_profile(profile: Profile):  
 profile.save()  
  
async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 telegram\_id = str(update.effective\_user.id)  
 username = update.effective\_user.username  
 logger.info(f"Пользователь {telegram\_id} (@{username}) использовал /start")  
  
 profile = await get\_profile\_by\_telegram\_id(telegram\_id)  
  
 if profile:  
 keyboard = [[InlineKeyboardButton("Отвязать Telegram", callback\_data='unbind\_telegram')]]  
 reply\_markup = InlineKeyboardMarkup(keyboard)  
 await update.message.reply\_text(  
 "Ваш Telegram аккаунт уже привязан. Вы можете его отвязать.",  
 reply\_markup=reply\_markup  
 )  
 else:  
 await update.message.reply\_text(  
 "Ваш Telegram аккаунт не привязан.\nПерейдите по ссылке на сайте, чтобы выполнить привязку."  
 )  
  
async def bind\_telegram(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 telegram\_id = str(update.effective\_user.id)  
 username = update.effective\_user.username  
 args = context.args  
  
 if args:  
 code = args[0]  
 logger.info(f"Пользователь {telegram\_id} (@{username}) пытается привязаться с кодом {code}")  
 profile = await get\_profile\_by\_code(code)  
  
 if profile:  
 if profile.telegram\_id:  
 await update.message.reply\_text("Этот код уже был использован.")  
 logger.warning(f"Попытка повторной привязки к уже использованному коду: {code}")  
 return  
  
 # Проверка, не привязан ли кто-то уже с таким ID  
 existing = await get\_profile\_by\_telegram\_id(telegram\_id)  
 if existing:  
 await update.message.reply\_text("Этот Telegram уже привязан к другому аккаунту.")  
 logger.warning(f"Telegram ID {telegram\_id} уже привязан к другому профилю.")  
 return  
  
 profile.telegram\_id = telegram\_id  
 profile.telegram\_token = None  
 await save\_profile(profile)  
  
 await update.message.reply\_text("✅ Привязка выполнена успешно!")  
 logger.info(f"Telegram ID {telegram\_id} привязан к пользователю {profile.user.username}")  
 else:  
 await update.message.reply\_text("⛔ Неверный код привязки или он уже использован.")  
 logger.warning(f"Неверный код привязки: {code}")  
 else:  
 await update.message.reply\_text("Используйте специальную ссылку с кодом, чтобы привязать аккаунт.")  
 logger.info(f"Пользователь {telegram\_id} не передал код.")  
  
async def unbind\_telegram(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 telegram\_id = str(update.effective\_user.id)  
 username = update.effective\_user.username  
 profile = await get\_profile\_by\_telegram\_id(telegram\_id)  
  
 if profile:  
 profile.telegram\_id = None  
 await save\_profile(profile)  
 await update.callback\_query.answer("Telegram отвязан.")  
 await update.callback\_query.edit\_message\_text("Ваш Telegram аккаунт успешно отвязан.")  
 logger.info(f"Пользователь {telegram\_id} (@{username}) отвязал Telegram")  
 else:  
 await update.callback\_query.answer("Telegram не был привязан.")  
 logger.warning(f"Попытка отвязки несуществующего Telegram ID: {telegram\_id}")  
  
def start\_bot\_logic():  
 logger.info("🚀 Telegram-бот запускается...")  
 application = Application.builder().token(BOT\_TOKEN).build()  
  
 application.add\_handler(CommandHandler("start", bind\_telegram))  
 application.add\_handler(CallbackQueryHandler(unbind\_telegram, pattern='^unbind\_telegram$'))  
  
 logger.info("✅ Бот успешно запущен и слушает обновления.")  
 application.run\_polling()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 start\_bot\_logic()

/TelegramBot/management/commands/runbot.py

# TelegramBot/management/commands/runbot.py  
  
from django.core.management.base import BaseCommand  
from TelegramBot.bot import start\_bot\_logic  
  
class Command(BaseCommand):  
 help = 'Run the Telegram bot'  
  
 def handle(self, \*args, \*\*options):  
 # Просто вызываем функцию, которая запускает бота  
 start\_bot\_logic()

/TelegramBot/management/commands/\_\_init\_\_.py

/TelegramBot/management/\_\_init\_\_.py

/TelegramBot/migrations/\_\_init\_\_.py

/TelegramBot/models.py

from django.db import models  
  
# Create your models here.

/TelegramBot/tests.py

from django.test import TestCase  
  
# Create your tests here.

/TelegramBot/views.py

from django.shortcuts import render  
  
# Create your views here.

/TelegramBot/\_\_init\_\_.py

/templates/base.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>{% block title %}Документы{% endblock %}</title>  
 <script src="https://cdn.tailwindcss.com"></script>  
</head>  
<body class="bg-gray-50 text-gray-800">  
  
 <!-- Header -->  
 <header class="bg-white shadow">  
 <div class="max-w-7xl mx-auto px-4 py-4 flex justify-between items-center">  
 <a href="{% url 'index' %}" class="text-xl font-bold text-gray-900 hover:text-blue-600">  
 📄 Документооборот  
 </a>  
 <nav class="flex space-x-8">  
 <a href="{% url 'document\_list' %}" class="text-lg text-gray-900 hover:text-blue-600 {% if '/docs' in request.path %}font-semibold{% endif %}">Все документы</a>  
 </nav>  
 <div>  
 {% if user.is\_authenticated %}  
 <a href="{% url 'profile' %}" class="mr-4 text-lg text-gray-900 hover:text-blue-600">  
 👤 {{ user.get\_full\_name|default:user.username }}  
 </a>  
 <a href="{% url 'logout' %}" class="text-blue-600 hover:underline">Выйти</a>  
 {% else %}  
 <a href="{% url 'social:begin' 'google-oauth2' %}?next={{ request.path }}" class="text-blue-600 hover:underline">Войти</a>  
 {% endif %}  
 </div>  
 </div>  
 </header>  
  
 <!-- Content -->  
 <main class="min-h-screen max-w-5xl mx-auto py-8 px-4">  
 {% for message in messages %}  
 <div class="mb-4 px-4 py-3 rounded bg-{{ message.tags }}-100 text-{{ message.tags }}-800">  
 {{ message }}  
 </div>  
 {% endfor %}  
  
 {% block content %}{% endblock %}  
 </main>  
  
 <!-- Footer -->  
 <footer class="bg-white border-t mt-10 py-4 text-center text-gray-500 text-sm">  
 &copy; {{ now.year }} Документооборот. Все права защищены.  
 </footer>  
  
</body>  
</html>

/templates/bind\_telegram.html

{% extends 'base.html' %}  
  
{% block content %}  
 <div class="text-center mt-10">  
 <h2 class="text-2xl font-semibold mb-4">Привязка Telegram</h2>  
 <p class="mb-4">Чтобы привязать Telegram-аккаунт, нажмите кнопку ниже:</p>  
 <a href="{{ deep\_link }}" class="bg-blue-500 hover:bg-blue-600 text-white px-6 py-2 rounded-lg">  
 Привязать Telegram  
 </a>  
 </div>  
{% endblock %}

/templates/documents/document\_detail.html

{% extends 'base.html' %}  
{% load static %}  
  
{% block content %}  
<div class="max-w-3xl mx-auto p-8 bg-white rounded-lg shadow-lg mt-8">  
  
 <!-- Заголовок документа -->  
 <h1 class="text-3xl font-semibold text-gray-900 mb-6">{{ document.title }}</h1>  
  
 <!-- Описание документа -->  
 <p class="text-gray-600 mb-6"><strong>Описание:</strong> {{ document.description|default:"—" }}</p>  
  
 <!-- Статус документа -->  
 <p class="text-gray-600 mb-6"><strong>Статус:</strong>  
 <span class="inline-block px-4 py-2 rounded-full text-sm  
 {% if document.status == 'signed' %}bg-green-200 text-green-800  
 {% elif document.status == 'accept' %}bg-blue-200 text-blue-800  
 {% elif document.status == 'revision' %}bg-yellow-200 text-yellow-800  
 {% else %}bg-gray-200 text-gray-800{% endif %}">  
 {{ document.get\_status\_display }}  
 </span>  
 </p>  
  
 <!-- Инициаторы -->  
 <div class="mb-6">  
 <h2 class="text-xl font-semibold text-gray-700 mb-2">Инициаторы:</h2>  
 <ul class="list-disc pl-6 space-y-2 text-gray-600">  
 {% for creator in document.creators.all %}  
 <li>{{ creator.get\_full\_name|default:creator.username }}</li>  
 {% endfor %}  
 </ul>  
 </div>  
  
 <!-- Подписанты -->  
 <div class="mb-6">  
 <h2 class="text-xl font-semibold text-gray-700 mb-2">Подписанты:</h2>  
 <ul class="list-disc pl-6 space-y-2 text-gray-600">  
 {% for signer in document.signers.all %}  
 <li class="flex justify-between items-center">  
 <span>{{ signer.get\_full\_name|default:signer.username }}</span>  
 {% if signer in document.signed\_by.all %}  
 <span class="text-green-500 text-sm">✔ Подписал</span>  
 {% endif %}  
 </li>  
 {% endfor %}  
 </ul>  
 </div>  
  
 <!-- Файлы документа -->  
 <div class="mb-6">  
 <h2 class="text-xl font-semibold text-gray-700 mb-2">Файлы:</h2>  
 {% if document.files.all %}  
 <ul class="list-disc pl-6 space-y-2 text-blue-600">  
 {% for file in document.files.all %}  
 {% if file.status == 'uploaded' %}  
 <li>  
 <a href="{{ file.file.url }}" class="hover:underline" target="\_blank">{{ file.name }}</a>  
 </li>  
 {% endif %}  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p class="text-gray-500">Нет прикреплённых файлов.</p>  
 {% endif %}  
 </div>  
  
 <!-- Действия с документом -->  
 <div class="mt-8 flex gap-6">  
 {% if can\_sign %}  
 <form method="post">  
 {% csrf\_token %}  
 <button type="submit" name="sign" class="bg-green-600 text-white px-6 py-3 rounded-lg hover:bg-green-700 transition">  
 ✍ Подписать документ  
 </button>  
 </form>  
 {% endif %}  
  
 {% if is\_director and document.status == 'signed' %}  
 <form method="post">  
 {% csrf\_token %}  
 <button type="submit" name="accept" class="bg-blue-600 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition">  
 ✅ Принять документ  
 </button>  
 </form>  
 {% endif %}  
  
 {% if is\_creator %}  
 <a href="{% url 'edit\_document' document.pk %}" class="bg-yellow-500 text-white px-6 py-3 rounded-lg hover:bg-yellow-600 transition">  
 ✏️ Редактировать  
 </a>  
 {% endif %}  
 </div>  
  
 <!-- Комментарии к документу -->  
 <div class="mt-8">  
 <h3 class="text-xl font-semibold text-gray-700 mb-4">📝 Комментарии:</h3>  
  
 {% for comment in comments %}  
 <div class="border-b py-4 mb-4">  
 <p class="font-semibold text-sm text-gray-800">{{ comment.user.get\_full\_name }} ({{ comment.created\_at|date:"H:i" }}):</p>  
 <p class="text-sm text-gray-700">{{ comment.message }}</p>  
 {% if comment.is\_system %}  
 <p class="text-xs text-gray-500">(Системное сообщение)</p>  
 {% endif %}  
 </div>  
 {% empty %}  
 <p class="text-sm text-gray-500">Комментариев ещё нет.</p>  
 {% endfor %}  
 </div>  
  
 <!-- Форма для оставления комментария -->  
 <div class="mt-8">  
 <h3 class="text-xl font-semibold text-gray-700 mb-4">Оставить комментарий:</h3>  
 <form method="post" class="space-y-4">  
 {% csrf\_token %}  
 {{ comment\_form.as\_p }}  
 <button type="submit" class="bg-green-600 text-white px-6 py-3 rounded-lg hover:bg-green-700 transition">Оставить комментарий</button>  
 </form>  
 </div>  
  
</div>  
{% endblock %}

/templates/documents/document\_edit.html

{% extends 'base.html' %}  
{% block title %}Редактирование документа{% endblock %}  
  
{% block content %}  
<div class="bg-white p-8 rounded-lg shadow-lg max-w-3xl mx-auto mt-8">  
 <h2 class="text-3xl font-semibold mb-6 text-gray-800">✏️ Редактирование документа</h2>  
  
 <form method="post" enctype="multipart/form-data" class="space-y-6">  
 {% csrf\_token %}  
  
 <p class="text-sm text-gray-700 font-semibold">  
 Вы автоматически добавлены как инициатор и подписант.  
 </p>  
  
 <!-- Формы полей документа -->  
 <div class="space-y-6">  
 {% for field in form %}  
 <div class="space-y-2">  
 <label class="block font-medium text-gray-700">{{ field.label }}</label>  
 <div class="block w-full p-3 border border-gray-300 rounded-lg shadow-sm focus:ring-2 focus:ring-blue-600 focus:outline-none">  
 {{ field }}  
 </div>  
 {% if field.help\_text %}  
 <p class="text-sm text-gray-500">{{ field.help\_text }}</p>  
 {% endif %}  
 {% for error in field.errors %}  
 <p class="text-red-500 text-sm">{{ error }}</p>  
 {% endfor %}  
 </div>  
 {% endfor %}  
 </div>  
  
 <!-- Добавление файлов с кастомным стилем -->  
 <h3 class="font-semibold text-lg mt-6 text-gray-800">📎 Добавить файлы</h3>  
 <div>  
 <label class="inline-block mb-2 text-sm text-gray-700">Выберите файлы:</label>  
 <input type="file" name="files" multiple class="block w-full text-sm text-gray-700 file:border file:border-gray-300 file:bg-blue-50 file:px-4 file:py-2 file:rounded-lg file:text-blue-700 file:cursor-pointer hover:file:bg-blue-100 focus:ring-2 focus:ring-blue-600 focus:outline-none" />  
 </div>  
  
 <!-- Загруженные файлы -->  
 <h3 class="font-semibold text-lg mt-6 text-gray-800">Загруженные файлы:</h3>  
 <ul class="list-disc pl-6 space-y-2 text-gray-700">  
 {% for file in files %}  
 <li class="flex justify-between items-center">  
 <span>{{ file.name }}</span>  
 <a href="{% url 'delete\_file' file.id %}" class="text-red-500 hover:underline">Удалить</a>  
 </li>  
 {% empty %}  
 <li class="text-gray-500">Нет файлов</li>  
 {% endfor %}  
 </ul>  
  
 <!-- Кнопки отправки формы -->  
 <div class="flex gap-6 pt-6">  
 <button type="submit"  
 class="bg-blue-600 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition">  
 💾 Сохранить изменения  
 </button>  
  
 {% if is\_creator %}  
 <button type="submit" name="send\_to\_revision" value="1"  
 class="bg-yellow-500 text-white px-6 py-3 rounded-lg hover:bg-yellow-600 transition">  
 🔄 Отправить на доработку  
 </button>  
 {% endif %}  
 </div>  
 </form>  
</div>  
{% endblock %}

/templates/documents/document\_form.html

{% extends 'base.html' %}  
{% block title %}Создание документа{% endblock %}  
  
{% block content %}  
<div class="bg-white p-8 rounded-lg shadow-lg max-w-3xl mx-auto mt-8">  
 <h2 class="text-3xl font-semibold mb-6 text-gray-800">➕ Создание нового документа</h2>  
  
 <form method="post" enctype="multipart/form-data" class="space-y-6">  
 {% csrf\_token %}  
  
 <p class="text-sm text-gray-700 font-semibold">  
 Вы автоматически добавлены как инициатор и подписант.  
 </p>  
  
 <!-- Формы полей документа -->  
 <div class="space-y-6">  
 {% for field in form %}  
 <div class="space-y-2">  
 <label class="block font-medium text-gray-700">{{ field.label }}</label>  
 <div class="block w-full p-3 border border-gray-300 rounded-lg shadow-sm focus:ring-2 focus:ring-blue-600 focus:outline-none">  
 {{ field }}  
 </div>  
 {% if field.help\_text %}  
 <p class="text-sm text-gray-500">{{ field.help\_text }}</p>  
 {% endif %}  
 {% for error in field.errors %}  
 <p class="text-red-500 text-sm">{{ error }}</p>  
 {% endfor %}  
 </div>  
 {% endfor %}  
 </div>  
  
 <!-- Файлы для добавления -->  
 <h3 class="font-semibold text-lg mt-6 text-gray-800">📎 Файлы:</h3>  
 <div>  
 <label class="inline-block mb-2 text-sm text-gray-700">Выберите файлы:</label>  
 <input type="file" name="files" multiple class="block w-full text-sm text-gray-700 file:border file:border-gray-300 file:bg-blue-50 file:px-4 file:py-2 file:rounded-lg file:text-blue-700 file:cursor-pointer hover:file:bg-blue-100 focus:ring-2 focus:ring-blue-600 focus:outline-none" />  
 </div>  
  
 <!-- Кнопка создания документа -->  
 <button type="submit" class="bg-green-600 text-white px-6 py-3 rounded-lg hover:bg-green-700 transition mt-6">  
 Создать документ  
 </button>  
 </form>  
</div>  
{% endblock %}

/templates/documents/document\_list.html

{% extends 'base.html' %}  
{% block title %}Все документы{% endblock %}  
  
{% block content %}  
<div class="bg-white p-6 rounded shadow-md">  
 <div class="flex justify-between items-center mb-4">  
 <h2 class="text-xl font-semibold">📚 Все документы</h2>  
 {% if user.profile.role == 'secretary' %}  
 <a href="{% url 'export\_documents\_pdf' %}" class="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700">  
 📥 Экспорт в PDF  
 </a>  
 {% endif %}  
 <a href="{% url 'create\_document' %}" class="bg-green-600 text-white px-4 py-2 rounded hover:bg-green-700 transition">  
 ➕ Новый документ  
 </a>  
  
 </div>  
  
 <!-- Форма фильтрации -->  
 <form method="get" class="space-x-4 mb-6">  
  
 <select name="academic\_year" class="border border-gray-300 rounded p-2">  
 <option value="">Выберите учебный год</option>  
 {% for year in years %}  
 <option value="{{ year.0 }}-{{ year.1 }}" {% if request.GET.academic\_year == year.0|stringformat:"s"|add:"-"|add:year.1|stringformat:"s" %}selected{% endif %}>  
 {{ year.0 }}-{{ year.1 }}  
 </option>  
 {% endfor %}  
 </select>  
  
 <select name="creator" class="border border-gray-300 rounded p-2">  
 <option value="">Выберите инициатора</option>  
 {% for creator in creators %}  
 <option value="{{ creator.id }}" {% if creator\_filter == creator.id|stringformat:"s" %}selected{% endif %}>{{ creator.get\_full\_name }}</option>  
 {% endfor %}  
 </select>  
  
 <select name="signer" class="border border-gray-300 rounded p-2">  
 <option value="">Выберите подписанта</option>  
 {% for signer in signers %}  
 <option value="{{ signer.id }}" {% if signer\_filter == signer.id|stringformat:"s" %}selected{% endif %}>{{ signer.get\_full\_name }}</option>  
 {% endfor %}  
 </select>  
  
 <button type="submit" class="bg-blue-600 text-white px-6 py-2 rounded hover:bg-blue-700 transition">Фильтровать</button>  
 </form>  
  
 <!-- Таблица документов -->  
 <div class="mb-4 border-b border-gray-200">  
 <nav class="-mb-px flex space-x-8">  
 <a href="?status=all" class="tab-link {% if not request.GET.status or request.GET.status == 'all' %}border-blue-500 text-blue-600{% else %}border-transparent text-gray-500{% endif %}">Все</a>  
 <a href="?status=accept" class="tab-link {% if request.GET.status == 'accept' %}border-blue-500 text-blue-600{% else %}border-transparent text-gray-500{% endif %}">Принятые</a>  
 <a href="?status=signed" class="tab-link {% if request.GET.status == 'signed' %}border-blue-500 text-blue-600{% else %}border-transparent text-gray-500{% endif %}">Подписанные</a>  
 <a href="?status=pending" class="tab-link {% if request.GET.status == 'pending' %}border-blue-500 text-blue-600{% else %}border-transparent text-gray-500{% endif %}">На подписании</a>  
 <a href="?status=revision" class="tab-link {% if request.GET.status == 'revision' %}border-blue-500 text-blue-600{% else %}border-transparent text-gray-500{% endif %}">На доработке</a>  
 </nav>  
 </div>  
  
 {% if documents %}  
 <div class="overflow-x-auto">  
 <table class="min-w-full divide-y divide-gray-200">  
 <thead class="bg-gray-50">  
 <tr>  
 <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Название</th>  
 <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Дата создания</th>  
 <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Подписанты</th>  
 <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Действия</th>  
 </tr>  
 </thead>  
 <tbody class="bg-white divide-y divide-gray-200">  
 {% for doc in documents %}  
 <tr>  
 <td class="px-6 py-4 whitespace-nowrap text-sm font-medium text-blue-600">  
 <a href="{% url 'document\_detail' doc.pk %}" class="hover:underline">{{ doc.title }}</a>  
 </td>  
 <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">  
 {{ doc.created\_at|date:"d.m.Y H:i" }}  
 </td>  
 <td class="px-6 py-4 whitespace-nowrap text-sm">  
 <div class="flex flex-wrap gap-2">  
 {% for signer in doc.signers.all %}  
 {% if signer in doc.signed\_by.all %}  
 <span class="bg-green-100 text-green-800 px-2 py-1 rounded text-xs">{{ signer.get\_full\_name|default:signer.username }}</span>  
 {% else %}  
 <span class="bg-yellow-100 text-yellow-800 px-2 py-1 rounded text-xs">{{ signer.get\_full\_name|default:signer.username }}</span>  
 {% endif %}  
 {% endfor %}  
 </div>  
 </td>  
 <td class="px-6 py-4 whitespace-nowrap text-sm flex gap-2">  
 <a href="{% url 'document\_detail' doc.pk %}" class="text-blue-600 hover:underline">👁 Просмотр</a>  
 {% if user in doc.creators.all %}  
 <a href="{% url 'edit\_document' doc.pk %}" class="text-yellow-600 hover:underline">✏️ Редактировать</a>  
 {% endif %}  
 {% if user in doc.signers.all and user not in doc.signed\_by.all %}  
 <form method="post" action="{% url 'document\_detail' doc.pk %}">  
 {% csrf\_token %}  
 <button type="submit" name="sign" class="text-green-600 hover:underline">✍ Подписать</button>  
 </form>  
 {% endif %}  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
  
 <!-- Пагинация -->  
 <div class="mt-4 flex justify-between items-center">  
 <div>  
 {% if documents.has\_previous %}  
 <a href="?status={{ status\_filter }}&page=1" class="text-blue-600 hover:underline">« Первая</a>  
 <a href="?status={{ status\_filter }}&page={{ documents.previous\_page\_number }}" class="ml-2 text-blue-600 hover:underline">‹ Назад</a>  
 {% endif %}  
 </div>  
 <div>  
 <span class="text-gray-600 text-sm">Страница {{ documents.number }} из {{ documents.paginator.num\_pages }}</span>  
 </div>  
 <div>  
 {% if documents.has\_next %}  
 <a href="?status={{ status\_filter }}&page={{ documents.next\_page\_number }}" class="ml-2 text-blue-600 hover:underline">Вперед ›</a>  
 <a href="?status={{ status\_filter }}&page={{ documents.paginator.num\_pages }}" class="ml-2 text-blue-600 hover:underline">Последняя »</a>  
 {% endif %}  
 </div>  
 </div>  
 {% else %}  
 <p class="text-gray-500">Нет документов в этой категории.</p>  
 {% endif %}  
</div>  
  
<style>  
 .tab-link {  
 padding-bottom: 0.5rem;  
 font-weight: 500;  
 border-b-2;  
 transition: all 0.3s;  
 }  
</style>  
{% endblock %}

/templates/edit\_photo.html

{% extends 'base.html' %}  
{% block content %}  
<div class="max-w-xl mx-auto mt-10">  
 <h2 class="text-2xl font-bold mb-4">Изменить фото профиля</h2>  
 <form method="post" enctype="multipart/form-data" class="bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit" class="mt-4 bg-blue-600 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">  
 Сохранить  
 </button>  
 </form>  
</div>  
{% endblock %}

/templates/index.html

{% extends 'base.html' %}  
  
{% block title %}Главная страница{% endblock %}  
  
{% block content %}  
<div class="max-w-6xl mx-auto py-16 px-4">  
 <h1 class="text-4xl font-extrabold text-center mb-8 text-gray-900">Добро пожаловать в систему документооборота</h1>  
  
 <p class="text-xl text-gray-700 mb-12 text-center">  
 Это ваша центральная платформа для управления документами и их подписанием в организации. Легко создавайте, отслеживайте и подписывайте важные документы в любое время.  
 </p>  
  
 <div class="grid grid-cols-1 md:grid-cols-2 gap-8">  
 <!-- Кнопки действия -->  
 <div class="flex flex-col items-center space-y-6">  
 <a href="{% url 'create\_document' %}" class="bg-green-600 text-white px-8 py-4 rounded-lg shadow-lg hover:bg-green-700 transition-all duration-300 text-xl text-center w-full sm:w-auto flex items-center justify-center space-x-2">  
 <span>➕ Создать новый документ</span>  
 </a>  
 <a href="{% url 'select\_document\_type' %}" class="bg-yellow-600 text-white px-8 py-4 rounded-lg shadow-lg hover:bg-green-700 transition-all duration-300 text-xl text-center w-full sm:w-auto flex items-center justify-center space-x-2">  
 <span>➕ Создать документ по шаблону</span>  
 </a>  
 <a href="{% url 'profile' %}" class="bg-blue-600 text-white px-8 py-4 rounded-lg shadow-lg hover:bg-blue-700 transition-all duration-300 text-xl text-center w-full sm:w-auto flex items-center justify-center space-x-2">  
  
 <span>👤 Перейти в профиль</span>  
 </a>  
 </div>  
  
 <!-- Информационные карточки -->  
 <div class="space-y-6">  
 <div class="bg-white p-8 rounded-lg shadow-lg hover:shadow-xl transition-shadow duration-300">  
 <div class="flex items-center mb-4 space-x-4">  
 <img src="https://img.icons8.com/ios-filled/50/000000/document.png" class="h-12 w-12" alt="Документы">  
 <h3 class="text-2xl font-semibold text-gray-900">Управление документами</h3>  
 </div>  
 <p class="text-lg text-gray-700">  
 Система позволяет эффективно управлять всеми документами внутри организации. Вы можете отслеживать статус каждого документа, управлять файлами и подписи, назначать подписантов и многое другое.  
 </p>  
 </div>  
  
 <div class="bg-white p-8 rounded-lg shadow-lg hover:shadow-xl transition-shadow duration-300">  
 <div class="flex items-center mb-4 space-x-4">  
 <img src="https://img.icons8.com/ios-filled/50/000000/pen.png" class="h-12 w-12" alt="Подписывать">  
 <h3 class="text-2xl font-semibold text-gray-900">Подписывать документы</h3>  
 </div>  
 <p class="text-lg text-gray-700">  
 Все документы можно подписывать в электронной форме. Подписанты могут легко подписать документы без необходимости физических подписей, ускоряя процесс работы.  
 </p>  
 </div>  
  
 <div class="bg-white p-8 rounded-lg shadow-lg hover:shadow-xl transition-shadow duration-300">  
 <div class="flex items-center mb-4 space-x-4">  
 <img src="https://img.icons8.com/ios-filled/50/000000/teamwork.png" class="h-12 w-12" alt="Роли пользователей">  
 <h3 class="text-2xl font-semibold text-gray-900">Роли пользователей</h3>  
 </div>  
 <p class="text-lg text-gray-700">  
 В системе предусмотрены различные роли для пользователей: Инициатор, Подписант, Администратор, Директор. В зависимости от роли, пользователи могут создавать, редактировать и подписывать документы.  
 </p>  
 </div>  
 </div>  
 </div>  
  
 <div class="mt-12 text-center">  
 <h3 class="text-2xl font-semibold text-gray-900 mb-6">Как это работает?</h3>  
 <p class="text-lg text-gray-700 max-w-3xl mx-auto">  
 Система документооборота позволяет организовать все этапы работы с документами, от их создания до подписания. Вы можете управлять доступом и ролями, отслеживать статус документов в режиме реального времени и получать уведомления о изменениях.  
 </p>  
 </div>  
</div>  
{% endblock %}

/templates/login\_disabled.html

{% extends 'base.html' %}  
  
{% block content %}  
<div class="max-w-xl mx-auto bg-white shadow-md rounded px-8 pt-6 pb-8 mt-10">  
 <h2 class="text-2xl font-semibold text-red-600 mb-4">Авторизация отключена</h2>  
 <p class="text-gray-700 mb-4">  
 Вход через сайт временно отключён. Используйте вход через Google или Telegram.  
 </p>  
 <a href="/" class="text-blue-500 hover:underline">Вернуться на главную</a>  
</div>  
{% endblock %}

/templates/profile.html

{% extends 'base.html' %}  
{% block title %}Профиль{% endblock %}  
  
{% block content %}  
<div class="max-w-3xl mx-auto py-10 px-4">  
 <h1 class="text-3xl font-bold text-center mb-6 text-gray-900">Ваш профиль</h1>  
  
 <div class="bg-white p-6 rounded-lg shadow-md">  
 <div class="flex justify-between items-center mb-6">  
 <div class="flex items-center">  
 {% if request.user.profile.photo %}  
 <img src="{{ request.user.profile.photo.url }}" alt="Фото профиля" class="w-32 h-32 rounded-full">  
 {% else %}  
 <div class="w-20 h-20 bg-gray-300 rounded-full mr-4"></div>  
 {% endif %}  
 <div>  
 <h2 class="text-xl font-semibold text-gray-900">{{ user.get\_full\_name|default:user.username }}</h2>  
 <p class="text-gray-700">{{ user.profile.role|capfirst }}</p>  
 <p class="text-gray-500">{{ user.email }}</p>  
 </div>  
 </div>  
  
 <a href="{% url 'edit\_profile' %}" class="bg-blue-600 text-white px-6 py-3 rounded-lg shadow hover:bg-blue-700 transition">  
 ✏️ Редактировать  
 </a>  
 </div>  
  
 <div class="mt-8">  
 <h3 class="text-xl font-semibold text-gray-900 mb-4">Информация о Telegram</h3>  
 <div class="mt-6">  
 {% if user.profile.telegram\_id %}  
 <p class="text-green-600 font-semibold">Telegram привязан ✅</p>  
 {% else %}  
 <a href="{% url 'bind\_telegram' %}"  
 class="inline-block bg-blue-600 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded transition">  
 🔗 Привязать Telegram  
 </a>  
 {% endif %}  
 </div>  
 {% if user.profile.telegram\_id %}  
 <p class="text-gray-700">Telegram ID: <span class="font-semibold">{{ user.profile.telegram\_id }}</span></p>  
 {% else %}  
  
 <p class="text-gray-500">Telegram ID не привязан.</p>  
 {% endif %}  
 {% if user.profile.telegram\_id %}  
 <form method="post" action="{% url 'unbind\_telegram' %}">  
 {% csrf\_token %}  
 <button type="submit" class="inline-block bg-red-600 hover:bg-red-700 text-white font-bold py-2 px-4 rounded transition">Отвязать Telegram</button>  
 </form>  
 {% endif %}  
 </div>  
  
 <div class="mt-8">  
 <h3 class="text-xl font-semibold text-gray-900 mb-4">Файлы профиля</h3>  
 {% if user.profile.document\_set.all %}  
 <ul class="list-disc pl-5 space-y-2">  
 {% for doc in user.profile.document\_set.all %}  
 <li class="flex justify-between items-center">  
 <span>{{ doc.title }}</span>  
 <a href="{% url 'document\_detail' doc.pk %}" class="text-blue-500 hover:underline">Подробнее</a>  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p class="text-gray-500">У вас нет документов.</p>  
 {% endif %}  
 </div>  
 </div>  
</div>  
{% endblock %}

/templates/templated\_document/create\_templated\_document.html

{% extends 'base.html' %}  
  
{% block title %}Создание документа{% endblock %}  
  
{% block content %}  
<div class="max-w-2xl mx-auto p-6 bg-white shadow-md rounded-lg mt-10">  
 <h1 class="text-3xl font-semibold text-center text-indigo-600 mb-6">Заполните данные для создания {{ document\_type }}</h1>  
  
 <form method="post" class="space-y-6">  
 {% csrf\_token %}  
 {% for field in form %}  
 <div class="mb-4">  
 <label for="{{ field.id\_for\_label }}" class="block text-sm font-medium text-gray-700">{{ field.label }}</label>  
 <div class="mt-1">  
 {{ field }}  
 {% if field.help\_text %}  
 <p class="text-xs text-gray-500 mt-1">{{ field.help\_text }}</p>  
 {% endif %}  
 </div>  
 </div>  
 {% endfor %}  
  
 <div class="flex justify-center">  
 <button type="submit" class="px-6 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-600 transition duration-300">  
 Создать документ  
 </button>  
 </div>  
 </form>  
</div>  
{% endblock %}

/templates/templated\_document/document\_created.html

{% extends 'base.html' %}  
  
{% block title %}Документ создан{% endblock %}  
  
{% block content %}  
<div class="max-w-2xl mx-auto p-6 bg-white shadow-md rounded-lg mt-10">  
 <h1 class="text-3xl font-semibold text-center text-indigo-600 mb-6">Документ успешно создан</h1>  
 <p class="text-center text-gray-700 mb-4">Документ был успешно создан и сохранен. Вы можете скачать его по следующей ссылке:</p>  
 <div class="text-center">  
 <a href="{{ file\_url }}" class="px-6 py-2 bg-green-500 text-white rounded-lg hover:bg-green-600 transition duration-300">  
 Скачать документ  
 </a>  
 </div>  
</div>  
{% endblock %}

/templates/templated\_document/select\_document\_type.html

{% extends 'base.html' %}  
  
{% block title %}Выбор типа документа{% endblock %}  
  
{% block content %}  
<div class="max-w-2xl mx-auto p-6 bg-white shadow-md rounded-lg mt-10">  
 <h1 class="text-3xl font-semibold text-center text-indigo-600 mb-6">Выберите тип документа</h1>  
 <ul class="space-y-4">  
 <li>  
 <a href="{% url 'create\_templated\_document' document\_type='act' %}" class="block p-4 bg-blue-500 text-white rounded-lg text-center hover:bg-blue-600 transition duration-300">  
 АКТ  
 </a>  
 </li>  
 <li>  
 <a href="{% url 'create\_templated\_document' document\_type='contract' %}" class="block p-4 bg-green-500 text-white rounded-lg text-center hover:bg-green-600 transition duration-300">  
 ДОГОВОР  
 </a>  
 </li>  
 <li>  
 <a href="{% url 'create\_templated\_document' document\_type='order' %}" class="block p-4 bg-yellow-500 text-white rounded-lg text-center hover:bg-yellow-600 transition duration-300">  
 ПРИКАЗ  
 </a>  
 </li>  
 </ul>  
</div>  
{% endblock %}