Приложение Г. Скрипт базы данных

## АННОТАЦИЯ

В данном программном документе приведен скрипт базы данных для веб-приложения для информационной системы документооборота «DocsManage».

СОДЕРЖАНИЕ

[АННОТАЦИЯ 1](#_Toc184497536)

[1. Скрипт базы данных 3](#_Toc184497537)

[1.1. Наименование скрипта 3](#_Toc184497538)

[1.2. Область применения скрипта 3](#_Toc184497539)

[1.3. Скрипт 3](#_Toc184497540)

# Скрипт базы данных

## Наименование скрипта

Наименование - «models.py».

## Область применения скрипта

Скрипт предназначен для веб-приложения для информационной системы документооборота «DocsManage».

## Скрипт

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'