Структура базы данных  
  
from django.db import models  
from django.contrib.auth.base\_user import BaseUserManager  
from django.contrib.auth.models import AbstractUser  
  
  
class UserManager(BaseUserManager):  
 *"""Define a model manager for User model with no username field."""* def \_create\_user(self, phone, password=None, \*\*extra\_fields):  
 *"""Create and save a User with the given phone and password."""* if not phone:  
 raise ValueError("The given phone number must be set")  
 user = self.model(phone=phone, \*\*extra\_fields)  
 user.set\_password(password)  
 user.save(using=self.\_db)  
 return user  
  
 def create\_user(self, phone, password=None, \*\*extra\_fields):  
 extra\_fields.setdefault("is\_staff", False)  
 extra\_fields.setdefault("is\_superuser", False)  
 return self.\_create\_user(phone, password, \*\*extra\_fields)  
  
 def create\_superuser(self, phone, password=None, \*\*extra\_fields):  
 *"""Create and save a SuperUser with the given phone and password."""* extra\_fields.setdefault("is\_staff", True)  
 extra\_fields.setdefault("is\_superuser", True)  
  
 if extra\_fields.get("is\_staff") is not True:  
 raise ValueError("Superuser must have is\_staff=True.")  
 if extra\_fields.get("is\_superuser") is not True:  
 raise ValueError("Superuser must have is\_superuser=True.")  
  
 return self.\_create\_user(phone, password, \*\*extra\_fields)  
  
  
class User(AbstractUser):  
 groups = models.ManyToManyField(  
 'users.Group',  
 related\_name='custom\_user\_set',  
 blank=True,  
 help\_text='The groups this user belongs to.',  
 verbose\_name='groups',  
 )  
  
 user\_permissions = models.ManyToManyField(  
 'users.Permission',  
 related\_name='custom\_user\_permissions',  
 blank=True,  
 help\_text='Specific permissions for this user.',  
 verbose\_name='user permissions',  
 )  
  
 first\_name = models.CharField(max\_length=50, verbose\_name="Имя")  
 phone = models.CharField(max\_length=13, unique=True, null=False, verbose\_name='Номер телефона')  
 user\_phone = models.CharField(max\_length=13, unique=True, blank=True, null=True, verbose\_name='Номер телефон пользователя')  
 email = models.EmailField(unique=True, null=True, blank=True)  
 code = models.CharField(max\_length=50, null=True, verbose\_name='Код', blank=True)  
 last\_sms\_date = models.DateTimeField(null=True, verbose\_name='Дата отправки кода', blank=True)  
 is\_confirm = models.BooleanField(default=False, blank=True, verbose\_name='Подтверждение почты')  
 created\_at = models.DateField(auto\_now\_add=True, verbose\_name='Дата создание')  
 objects = UserManager()  
 last\_name = None  
 username = None  
  
 USERNAME\_FIELD = "phone"  
 REQUIRED\_FIELDS = []  
  
 def \_\_str\_\_(self):  
 return str(self.phone)  
  
 class Meta:  
 verbose\_name = 'Пользователь'  
 verbose\_name\_plural = 'Пользователи'

from django.db import models  
  
  
class ClientType(models.TextChoices):  
 SENDER = 'sender', 'Отправитель'  
 RECEIVER = 'receiver', 'Получатель'  
 PARTNER = 'partner', 'Партнёр'  
  
  
class Client(models.Model):  
 name = models.CharField(max\_length=255, verbose\_name="Имя")  
 phone = models.CharField(max\_length=20, verbose\_name="Телефон")  
 email = models.EmailField(blank=True, null=True, verbose\_name="Email")  
 address = models.TextField(verbose\_name="Адрес")  
 client\_type = models.CharField(max\_length=20, choices=ClientType.choices, verbose\_name="Тип клиента")  
  
 class Meta:  
 verbose\_name = "Клиент"  
 verbose\_name\_plural = "Клиенты"  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Driver(models.Model):  
 full\_name = models.CharField(max\_length=255, verbose\_name="ФИО")  
 phone = models.CharField(max\_length=20, verbose\_name="Телефон")  
 license\_number = models.CharField(max\_length=50, verbose\_name="Номер водительского удостоверения")  
 is\_available = models.BooleanField(default=True, verbose\_name="Доступен")  
  
 class Meta:  
 verbose\_name = "Водитель"  
 verbose\_name\_plural = "Водители"  
  
 def \_\_str\_\_(self):  
 return self.full\_name  
  
  
class Vehicle(models.Model):  
 number\_plate = models.CharField(max\_length=20, verbose\_name="Госномер")  
 brand = models.CharField(max\_length=100, verbose\_name="Марка")  
 capacity\_kg = models.FloatField(verbose\_name="Грузоподъемность (кг)")  
 volume\_m3 = models.FloatField(verbose\_name="Объем (м³)")  
 is\_available = models.BooleanField(default=True, verbose\_name="Доступна")  
  
 class Meta:  
 verbose\_name = "Транспорт"  
 verbose\_name\_plural = "Транспорты"  
  
 def \_\_str\_\_(self):  
 return f"{self.number\_plate} ({self.brand})"  
  
  
class OrderStatus(models.TextChoices):  
 CREATED = 'created', 'Создан'  
 IN\_TRANSIT = 'in\_transit', 'В пути'  
 DELIVERED = 'delivered', 'Доставлен'  
 CANCELLED = 'cancelled', 'Отменен'  
  
  
class Order(models.Model):  
 client\_from = models.ForeignKey(Client, on\_delete=models.CASCADE, related\_name='orders\_sent', verbose\_name="Отправитель")  
 client\_to = models.ForeignKey(Client, on\_delete=models.CASCADE, related\_name='orders\_received', verbose\_name="Получатель")  
 cargo\_description = models.TextField(verbose\_name="Описание груза")  
 weight = models.FloatField(verbose\_name="Вес (кг)")  
 volume = models.FloatField(verbose\_name="Объем (м³)")  
 vehicle = models.ForeignKey(Vehicle, on\_delete=models.SET\_NULL, null=True, blank=True, verbose\_name="Машина")  
 driver = models.ForeignKey(Driver, on\_delete=models.SET\_NULL, null=True, blank=True, verbose\_name="Водитель")  
 status = models.CharField(max\_length=20, choices=OrderStatus.choices, default=OrderStatus.CREATED, verbose\_name="Статус")  
 created\_at = models.DateTimeField(auto\_now\_add=True, verbose\_name="Дата создания")  
 delivery\_date = models.DateField(null=True, blank=True, verbose\_name="Дата доставки")  
  
 class Meta:  
 verbose\_name = "Заказ"  
 verbose\_name\_plural = "Заказы"  
  
 def \_\_str\_\_(self):  
 return f"Заказ #{self.pk} от {self.client\_from} к {self.client\_to}"

Структура Административной части  
from django.contrib import admin  
  
from apps.logistika.models import Client, Driver, Vehicle, Order  
  
admin.site.register(Client)  
admin.site.register(Driver)  
admin.site.register(Vehicle)  
admin.site.register(Order)

Настройки проекта  
*"""  
Django settings for main project.  
  
Generated by 'django-admin startproject' using Django 5.2.1.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/5.2/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/5.2/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/5.2/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-2awfl3dsngzgkq)tg+\*f53al93i=-pvwxl+jo46#&36#y9l4ww'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = ["\*"]  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'admin\_star.apps.AdminStarConfig',  
 'django.contrib.admin',  
 'django.contrib.users',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'apps.logistika',  
 'apps.user',  
]  
  
ADMIN2\_THEME\_DIRECTORY = "djadmin2theme\_bootstrap3"  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.users.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'main.urls'  
  
AUTH\_USER\_MODEL = "user.User"  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.request',  
 'django.contrib.users.context\_processors.users',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'main.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.users.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.users.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.users.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.users.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/5.2/topics/i18n/  
  
LANGUAGE\_CODE = 'ru'  
  
TIME\_ZONE = 'Asia/Bishkek'  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/5.2/howto/static-files/  
  
STATIC\_URL = '/static/'  
STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')  
  
STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, 'static'),  
]  
  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')  
MEDIA\_URL = '/media/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'