**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«РУТ (МИИТ)»**

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

Отчет по практической работе

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«ГАИ»**

Выполнил: ст. гр. ТКИ-541  
Карасев М.М.  
Проверил: Васильева М.А.

**Москва**

**2023**

**Цель работы:**

Научится использовать структуру ОРМ на любом языке программирования.

**Постановка задачи:**

При помощи языка программирования Python и фреймворка Django создать базу ГАИ.

**Выполнение работы:**

**models.py**

from django.db import models

class Owner(models.Model):

    name = models.CharField(max\_length=255)

    surname = models.CharField(max\_length=255)

    last\_name = models.CharField(max\_length=255, default='', blank=True)

    birth\_date = models.DateField()

    def \_\_str\_\_(self):

        return f"{self.name} {self.surname}"

    class Meta:

        verbose\_name = "Владелец"

        verbose\_name\_plural = "Владельцы"

class Brand(models.Model):

    brand = models.CharField(max\_length=255, verbose\_name='бренд', unique=True)

    def \_\_str\_\_(self):

        return self.brand

    class Meta:

        verbose\_name = "Бренд"

        verbose\_name\_plural = "Бренды"

class Model(models.Model):

    brand = models.ForeignKey(Brand, on\_delete=models.CASCADE)

    model = models.CharField(max\_length=255, verbose\_name='брэнд', unique=True)

    def \_\_str\_\_(self):

        return f"{self.brand} {self.model}"

    class Meta:

        verbose\_name = "Модель"

        verbose\_name\_plural = "Модели"

class Car(models.Model):

    template\_number = models.CharField(max\_length=12, verbose\_name="Номер автомобиля", unique=True)

    motor\_number = models.CharField(max\_length=255, verbose\_name="Номер двигателя", unique=True)

    model = models.ForeignKey(Model, on\_delete=models.CASCADE)

    # brand = models.CharField(max\_length=255, verbose\_name="Марка")

    # model = models.CharField(max\_length=255, verbose\_name="Модель")

    color = models.CharField(max\_length=255, verbose\_name="Цвет")

    registration\_date = models.DateField(max\_length=255, verbose\_name="Дата регистрации")

    owner = models.ForeignKey(Owner, on\_delete=models.CASCADE)

    is\_stolen = models.BooleanField(verbose\_name="Автомобиль угнан")

    prev\_ovner = models.ManyToManyField(Owner, related\_name="prev", blank=True)

    def \_\_str\_\_(self):

        return f"{self.owner} {self.model}    {'В угоне!' if self.is\_stolen else ''}"

    class Meta:

        verbose\_name = "Автомобиль"

        verbose\_name\_plural = "Автомобили"

class Crash(models.Model):

    date = models.DateField()

    car = models.ManyToManyField(Car)

    def \_\_str\_\_(self):

        return f"{self.date}"

    class Meta:

        verbose\_name = "Авария"

        verbose\_name\_plural = "Аварии"

**Описание классов:**

***Owner***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *name* | *CharField(255)* | *Да* |
| *surname* | *CharField(255)* | *Да* |
| *last\_name* | *CharField(255)* | *Нет* |
| *birth\_date* | *DateField* | *Да* |

***Brand***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *brand* | *CharField(255)* | *Да, Уникальное* |

***Model***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *brand* | *ForeignKey(Brand)* | *Да* |
| *model* | *CharField(255)* | *Да, Уникальное* |

***Car***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *template\_number* | *CharField(12)* | *Да* |
| *motor\_number* | *CharField(255)* | *Да* |
| *model* | *ForeignKey (Model)* | *Да* |
| *color* | *CharField(255)* | *Да* |
| *registration\_date* | *DateField* | *Да* |
| *owner* | *ForeignKey(Owner)* | *Да* |
| *is\_stolen* | *BooleanField* | *Да* |
| *prev\_ovner* | *ManyToMany(Owner)* | *Да* |

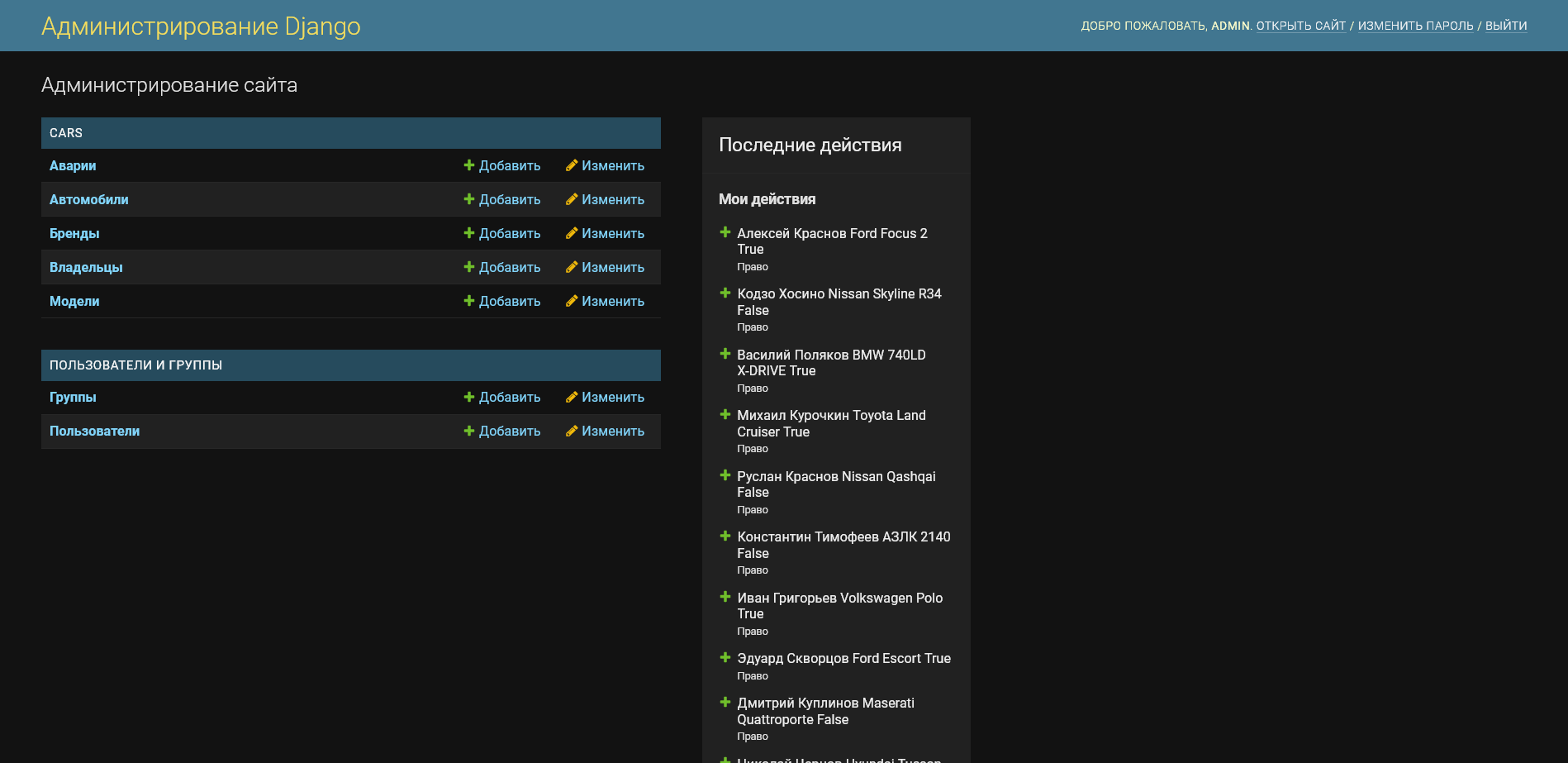
***Crash***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *date* | *DateField* | *Да* |
| *car* | *ManyToMany(Car)* | *Да* |

**Скриншоты работы**

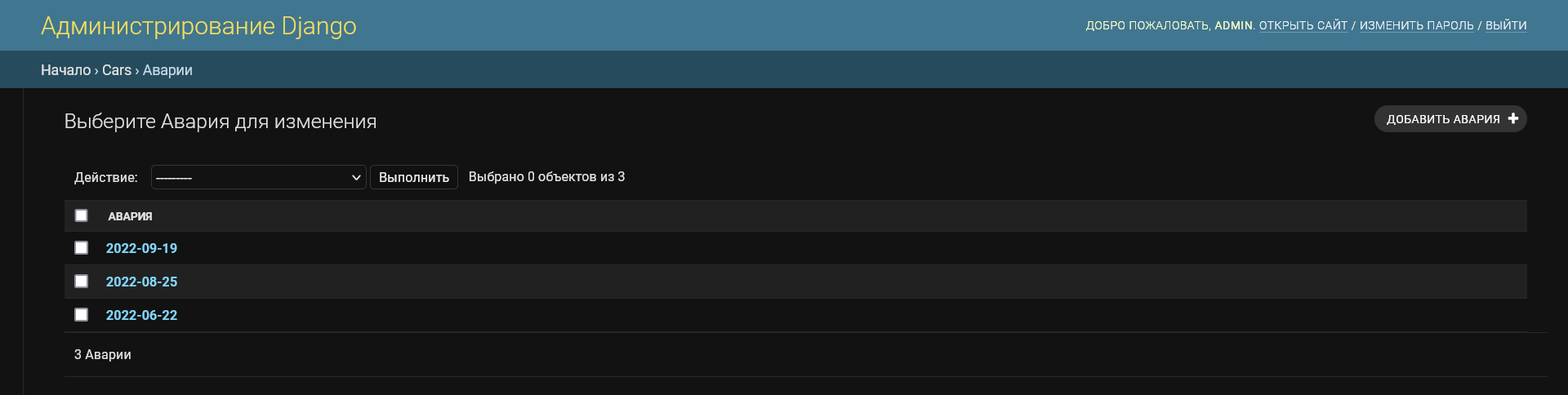
После заполнения настройки классов стало доступно редактирование базы данных в графическом интрефейсе

**Изменение данных**

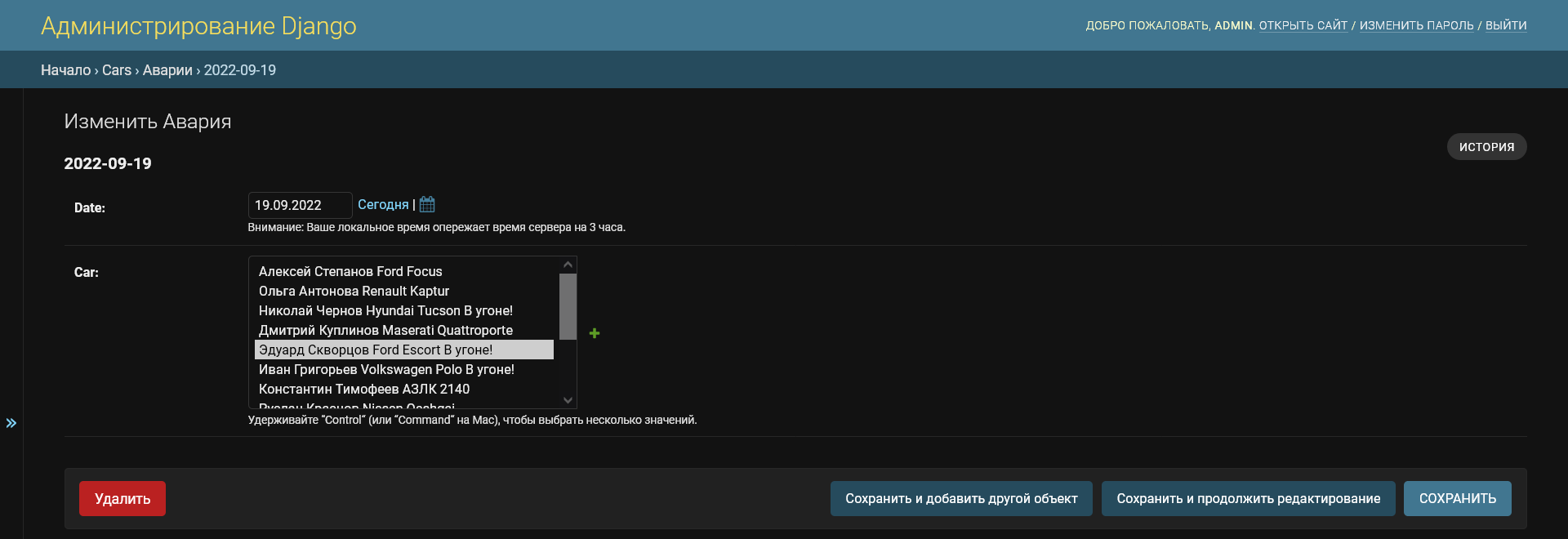


*Рисунок 1 — страница администратора*

**Изменение аварий**

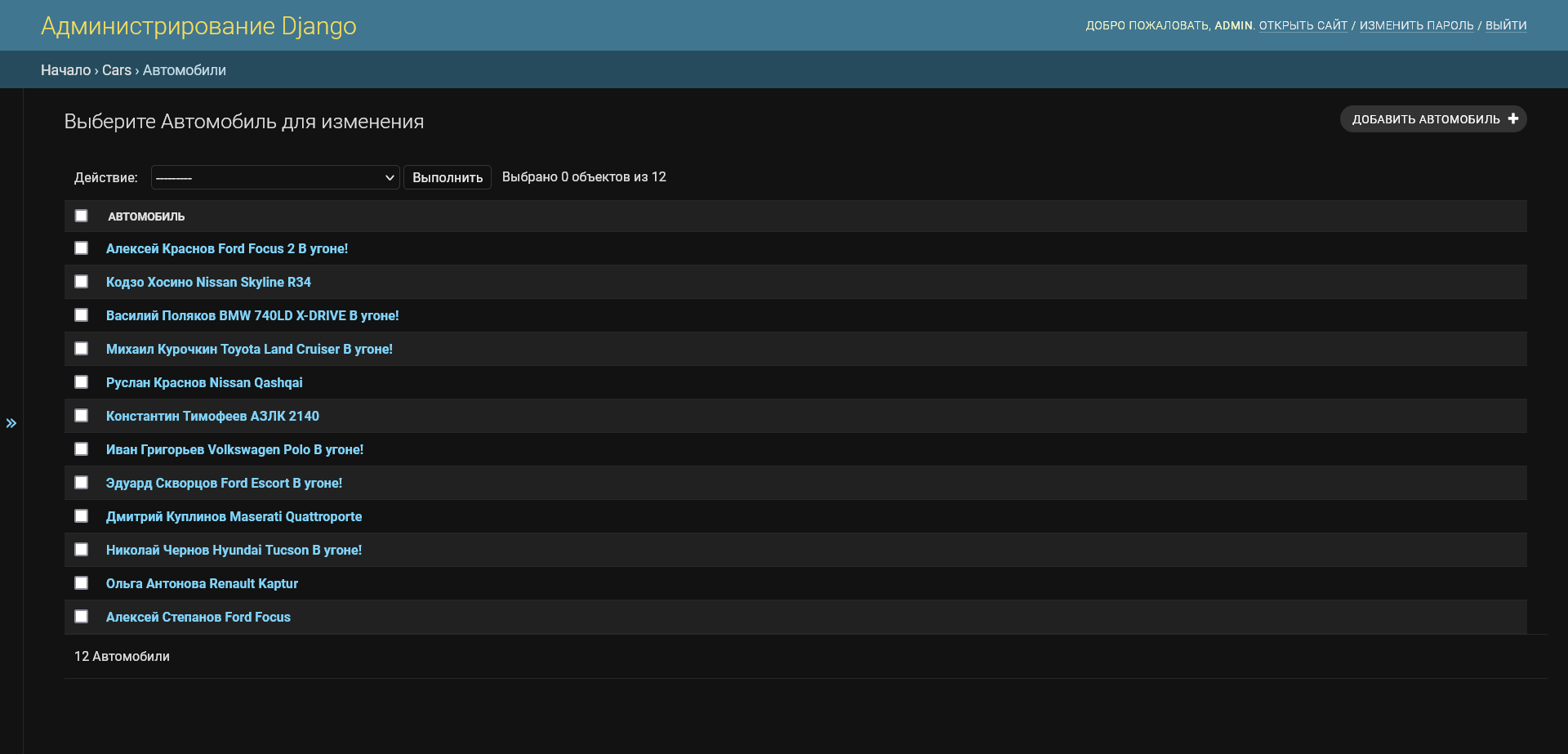


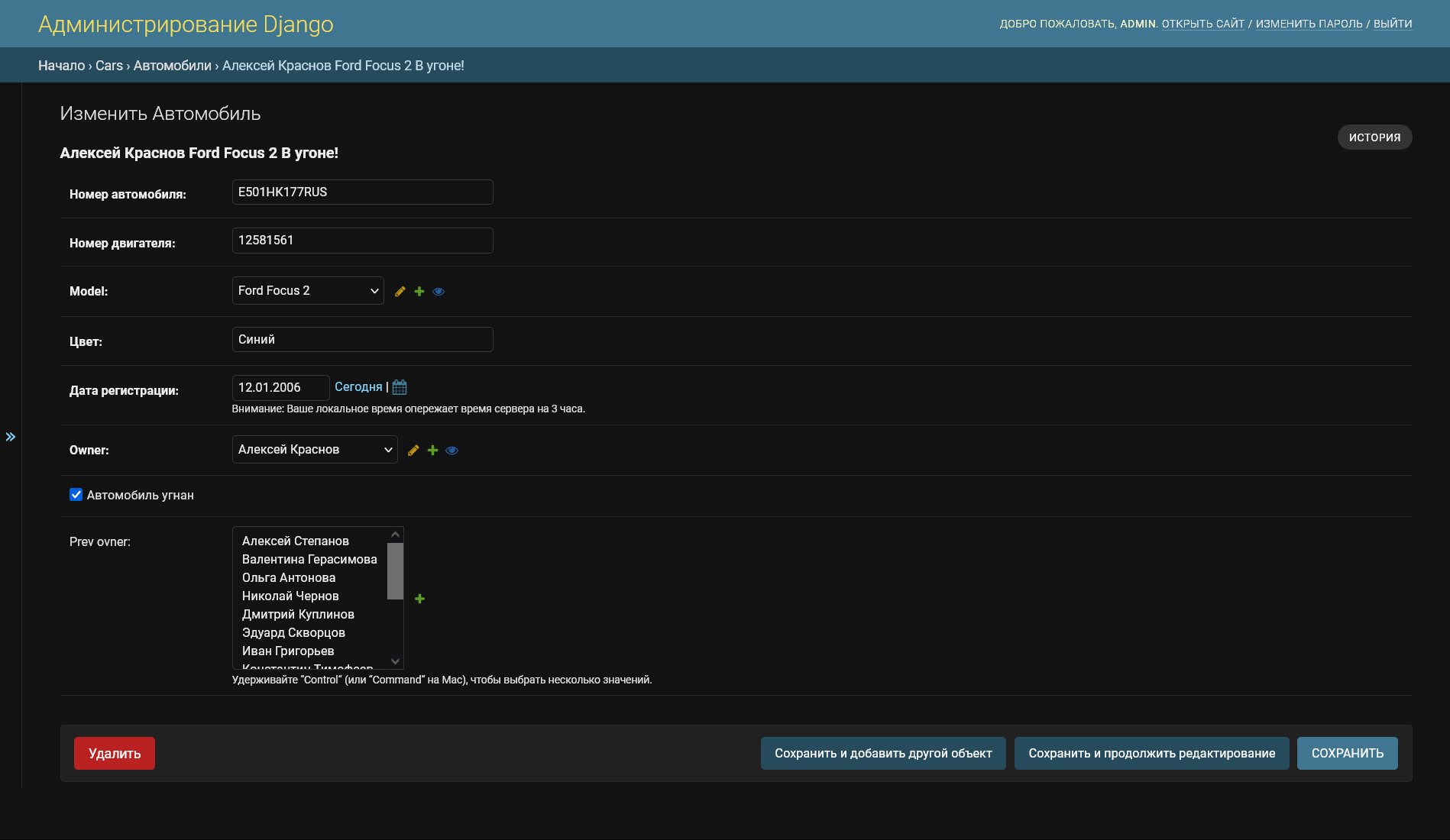
*Рисунок 2 — Страница изменения аварий*



*Рисунок 3 — страница изменения аварии*

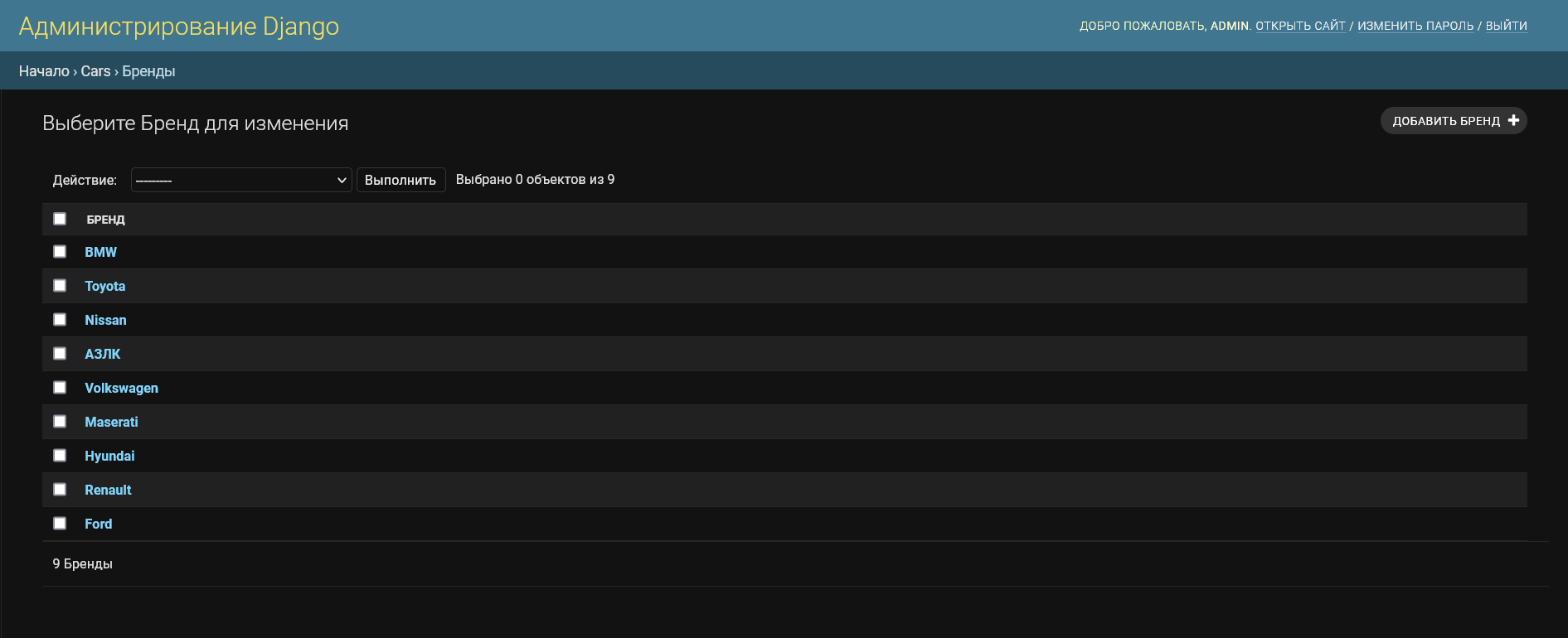
**Изменение Автомобиля**

 *Рисунок 4 — страница изменения автомобилей*



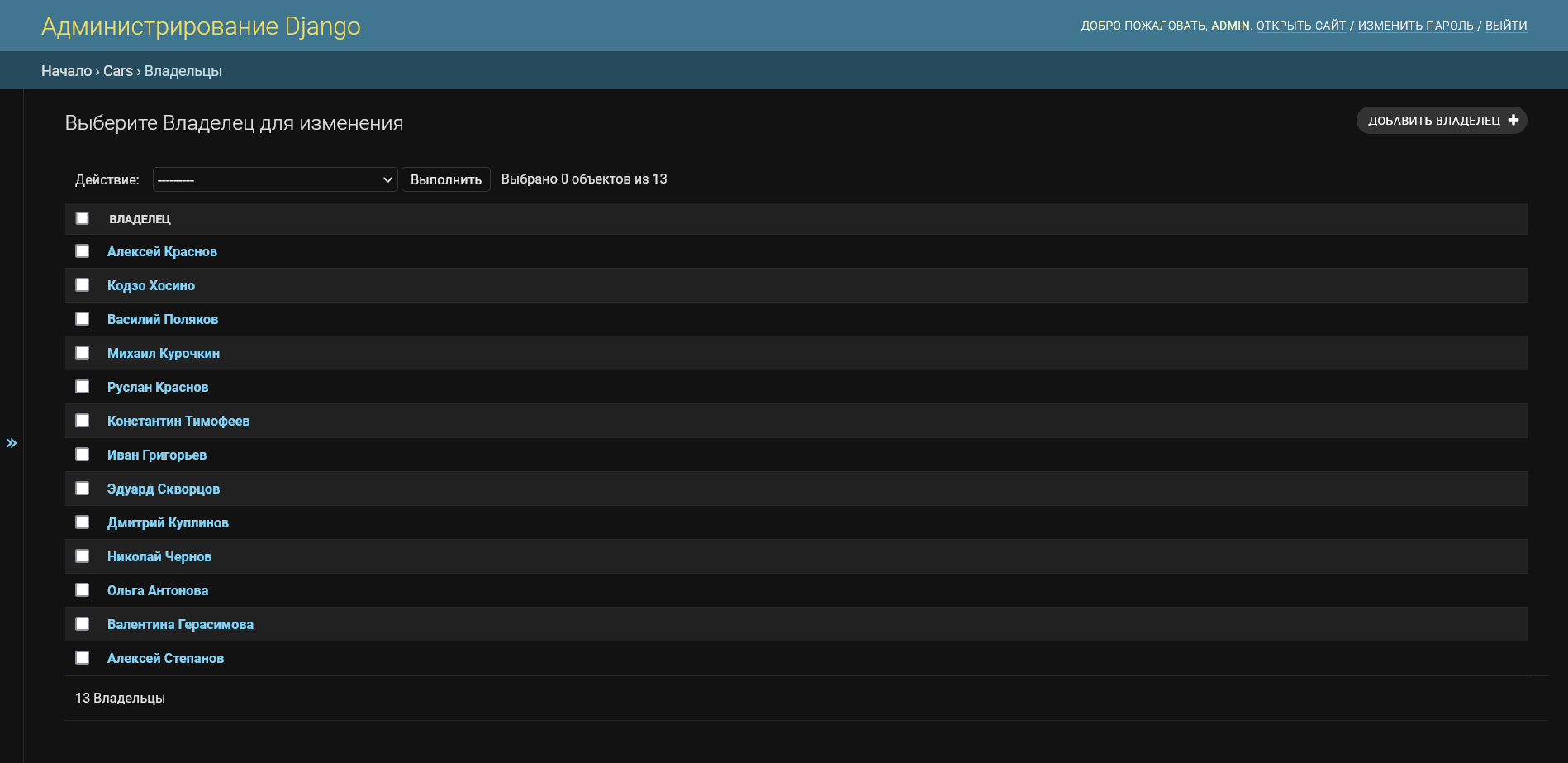
*Рисунок 5 — страница изменения автомобиля*

**Изменение Брендов**

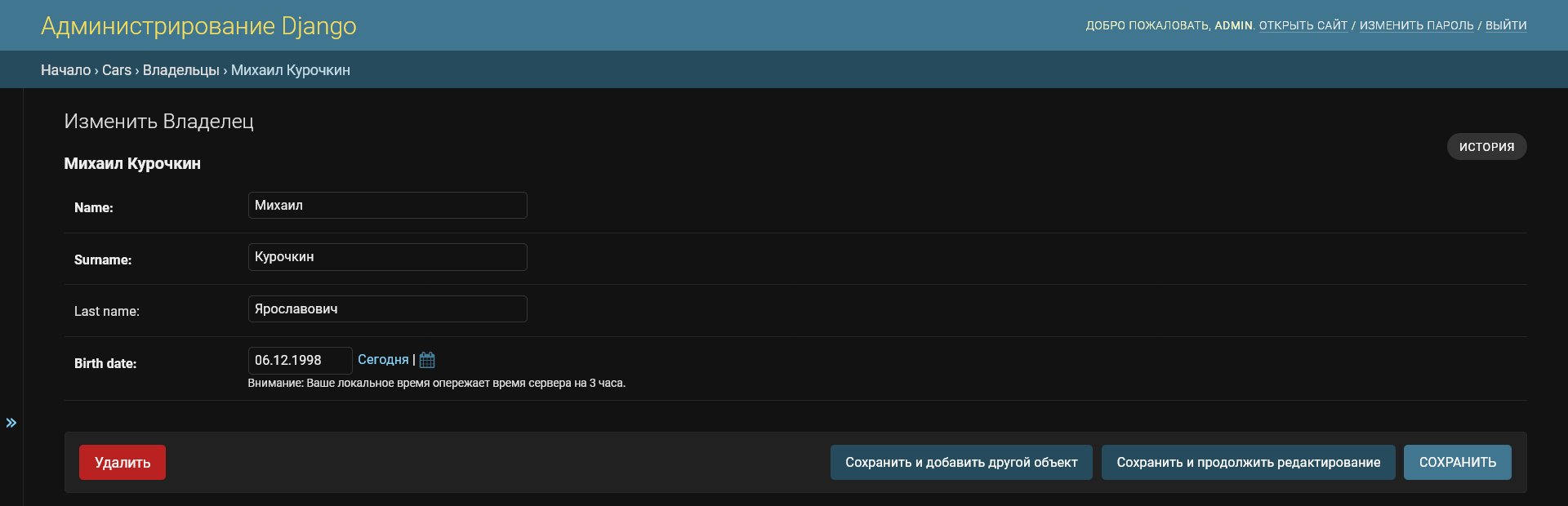


*Рисунок 6 — страница изменения записей*

**Изменение Владельцев**

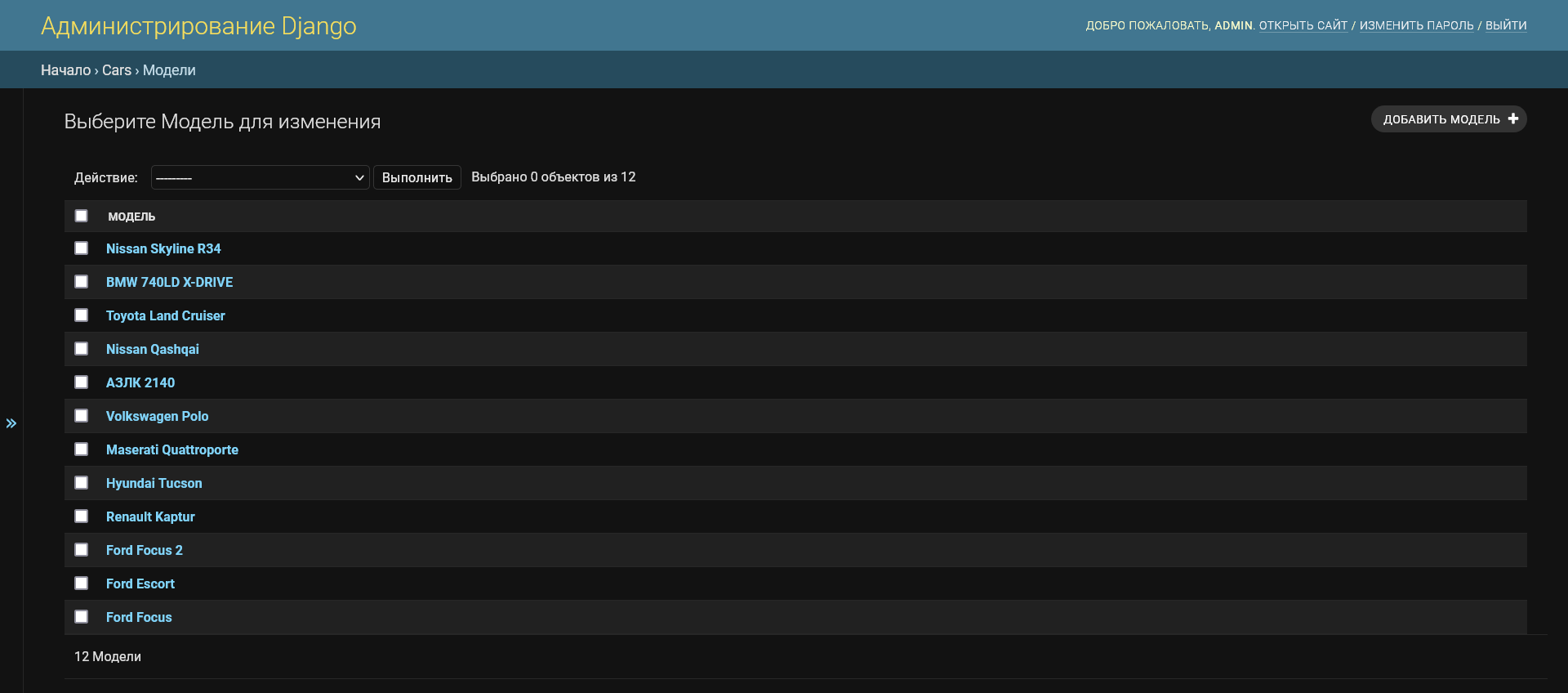


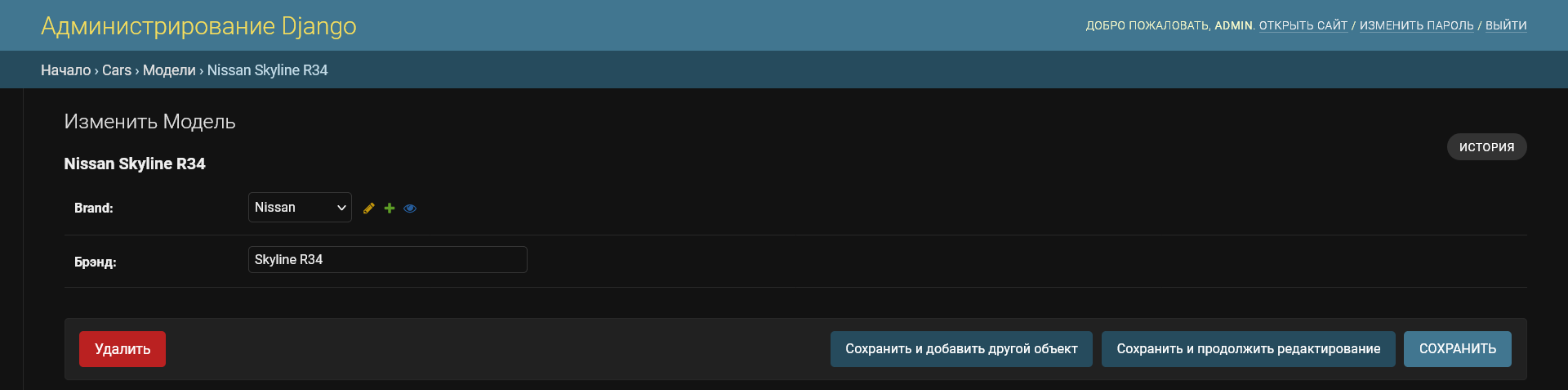
*Рисунок 7 — страница изменения владельцев*



*Рисунок 8 — страница изменения владельца*

**Изменение Моделей**

 *Рисунок 9 — страница изменения моделей*



*Рисунок 10 — страница изменения модели*

**Вывод:**

Была произведена работа по созданию базы данных ГАИ, с использованием технологии ОРМ на примере Python Django