МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Белгородский Государственный Технологический Университет им. В.Г. Шухова»

Кафедра информационных технологий

Лабораторная работа №10

дисциплина: **«Управление данными»**

тема: **«Django. Административный интерфейс. Создание приложения»**

Вариант №6

Выполнил:
студент группы ИТз-212
Фокин В.О.

Принял:
ст. пр. кафедры ИТ
Шаптала В.В.

Белгород 2023

Ход работы

Установим проект *Django* с помощью команды:

```
pip install django –user
```

Создадим проект *samplesite* с помощью команды

```
django-admin startproject samplesite
```

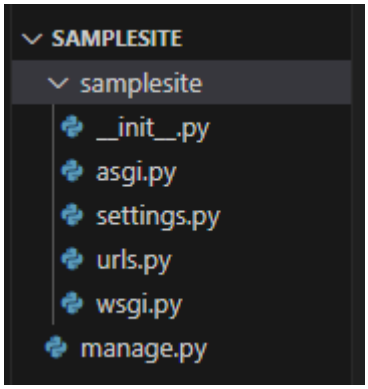

Рис. 1. Установленные файлы

Запустим веб-сервер с помощью отладочной команды

```
python manage.py runserver
```

и перейдем по адресу *http://localhost:8000/*.

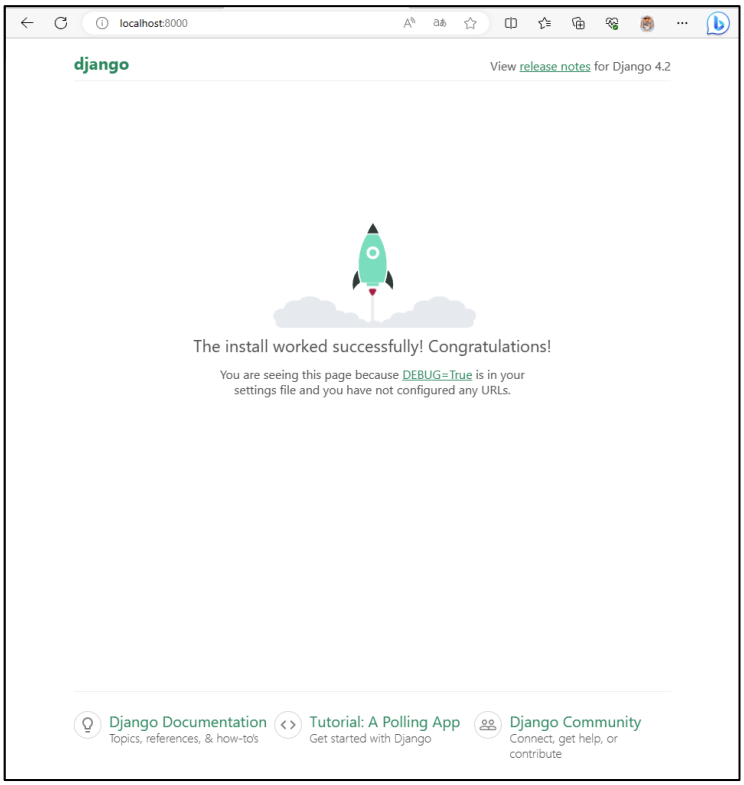

Рис. 2. Результат работы программы

Создадим приложение fboard. Для этого в папке проекта выполним команду:

```
python manage.py startapp fboard
```

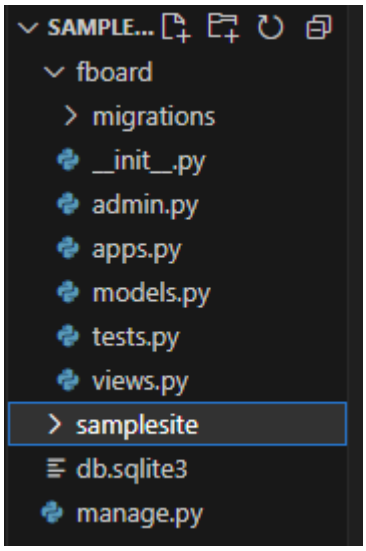

Рис. 3. Созданные файлы

Зарегистрируем созданное приложение в проекте добавим в список *INSTALLED_APPS* добавим строку

```
'fboard.aaps.FboardConfig'
```

Для описания таблицы транспортных средств, объявим модель *Vehicle* в модуле *models.py*:

```
class Vehicle(models.Model):
    seats = models.IntegerField(verbose_name="Общее количество мест")
    type = models.CharField(max_length=30, verbose_name="Тип",
primary_key=True)

    class Meta:
        verbose_name_plural = "Транспортные средства"
        verbose_name = "Транспортное средство"
```

Для описания таблицы рейсов из базы данных, объявим модель *Flight* в модуле *models.py*:

```
class Flight(models.Model):
    num = models.IntegerField(primary_key=True, verbose_name="Номер")
    fr = models.CharField(max_length=40, verbose_name="Пункт отправления")
    to = models.CharField(max_length=40, verbose_name="Пункт назначения")
    departure = models.DateTimeField(auto_now=False, db_index=True,
verbose_name="Время отправления")
    arrive = models.DateTimeField(auto_now=False, db_index=True,
verbose_name="Время прибытия")
```

```
    vehicle = models.ForeignKey(Vehicle, on_delete=models.PROTECT,
verbose_name="Тип транспортного средства")

    class Meta:
        verbose_name_plural = "Рейсы"
        verbose_name = "Рейс"
```

Для описания таблицы билетов из базы данных, объявим модель *Ticket* в модуле *models.py*:

```
class Ticket(models.Model):
    id = models.IntegerField(primary_key=True, verbose_name="Уникальный номер
пассажира")
    full_name = models.CharField(max_length=40, verbose_name="ФИО")
    departure = models.DateTimeField(auto_now=False, db_index=True,
verbose_name="Дата и время отправления")
    purchase = models.DateTimeField(auto_now=False, db_index=True,
verbose_name="Дата и время продажи")
    booking = models.DateTimeField(auto_now=False, db_index=True,
verbose_name="Дата и время бронирования")
    price = models.DecimalField(max_digits=5, decimal_places=2,
verbose_name="Стоимость")
    flight_num = models.ForeignKey(Flight, on_delete=models.PROTECT,
verbose_name="Номер рейса")
    category = models.CharField(max_length=15, verbose_name="Тип категории")

    class Meta:
        verbose_name_plural = "Билеты"
        verbose_name = "Билет"
```

Для описания таблицы категорий из базы данных, объявим модель *Category* в модуле *models.py*:

```
class Category(models.Model):
    type = models.CharField(max_length=15, verbose_name="Тип")
    seats = models.IntegerField(verbose_name="Количество мест данной
категории")
    vehicle = models.ForeignKey(Vehicle, on_delete=models.PROTECT,
verbose_name="Тип транспортного средства")

    class Meta:
        verbose_name_plural = "Категории"
        verbose_name = "Категория"
        constraints = [
            models.UniqueConstraint(
                fields=["type", "vehicle"], name="primary_key"
            )
        ]
```

Сгенерируем миграцию на основе созданной моделей и проверим sql-код, созданный миграцией с помощью команд:

```
python manage.py makemigrations fboard
```

Затем выполним миграцию с помощью команды

```
python manage.py migrate
```

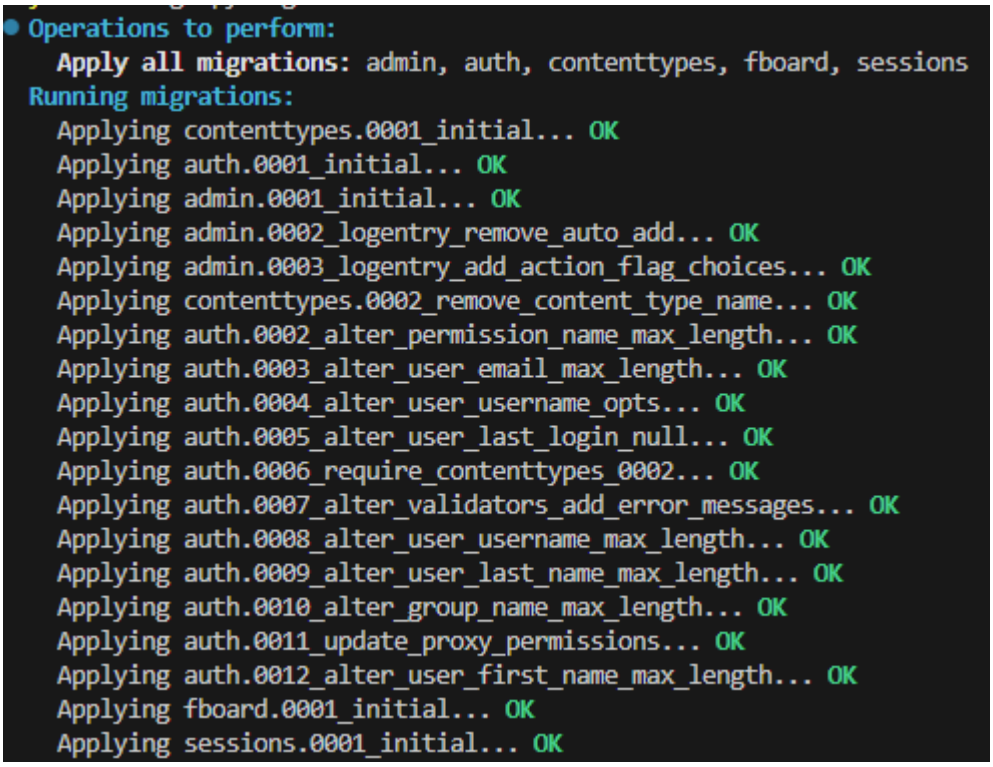

Рис. 4. Результат выполнения команды

Создадим зарегистрированного пользователя административного сайта с максимальными правами – правами суперпользователя, с помощью команды

```
python manage.py createsuperuser
```

Добавим созданные модели в модуле *admin.py*.

```
from django.contrib import admin
from .models import Vehicle, Flight, Ticket, Category

# Register your models here.
class VehicleAdmin(admin.ModelAdmin):
    list_display = ("seats", "type")

class FlightAdmin(admin.ModelAdmin):
    list_display = ("num",  "fr",  "to",  "departure", "arrive", "vehicle")
    search_fields = ("fr", "to")

class TicketAdmin(admin.ModelAdmin):
    list_display = ("id", "full_name", "departure", "purchase", "booking",
"price", "flight_num", "category")

class CategoryAdmin(admin.ModelAdmin):
    list_display = ("type", "seats", "vehicle")
```

```
admin.site.register(Vehicle, VehicleAdmin)
admin.site.register(Flight, FlightAdmin)
admin.site.register(Ticket, TicketAdmin)
admin.site.register(Category, CategoryAdmin)
```

Запустим отладочный сервер и перейдем по адресу http://localhost:8000/admin/.

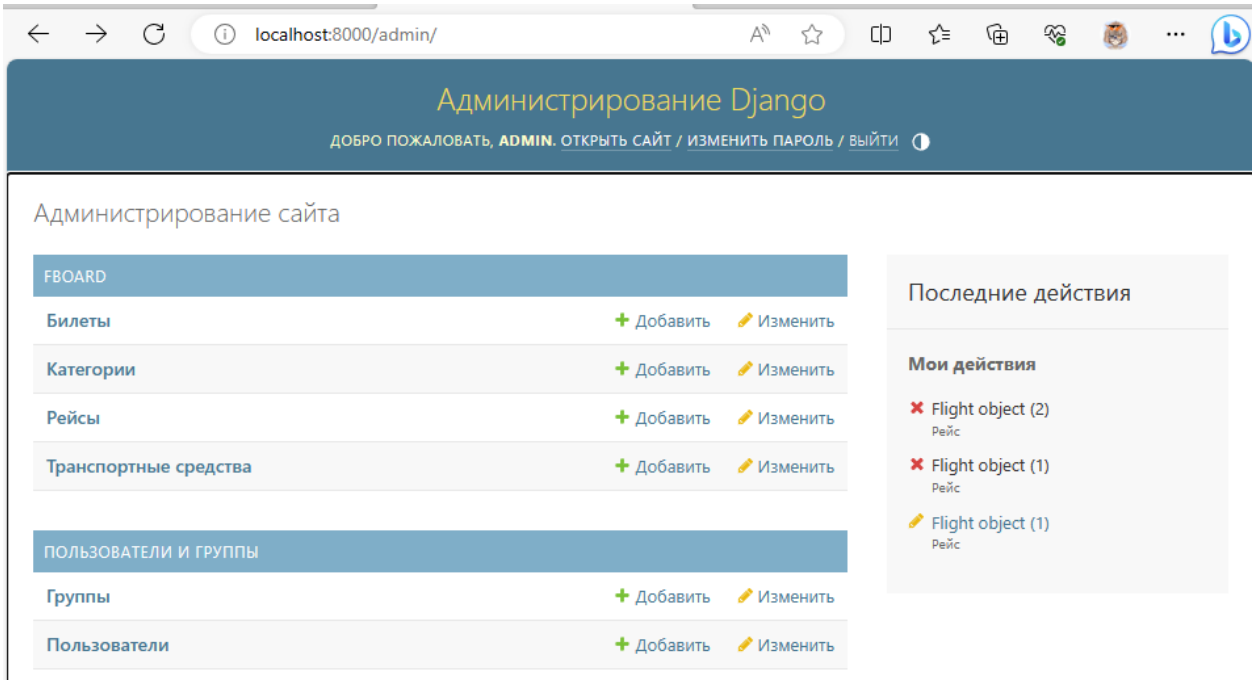

Рис. 5. Административный интерфейс сайта