

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И
ОПТИКИ»**

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Дисциплина: Основы web-программирования

Отчет
по практическому занятию №1
«Django Web Framework»

Выполнила:
Назаренко Ульяна Кирилловна,
студентка группы К3343

Преподаватель:
Говоров Антон Игоревич

Санкт-Петербург,
2020 г.

Цель работы: освоить «Django Web Framework» и научиться создавать модели.

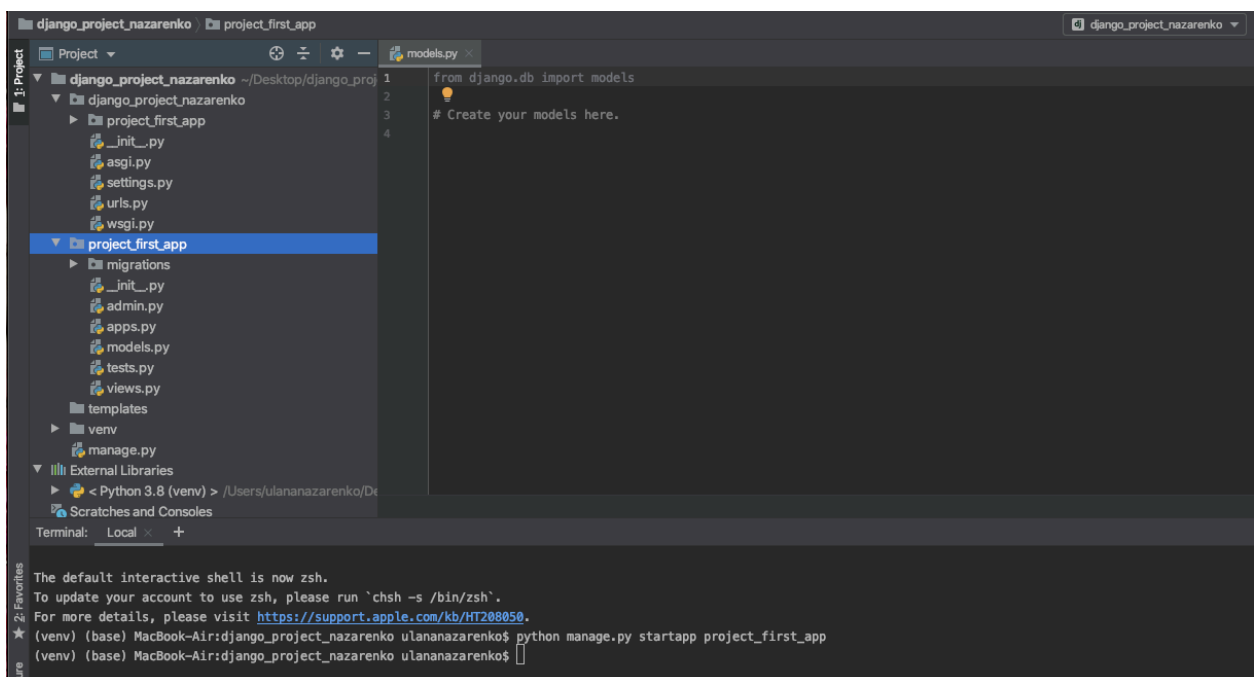
Задачи:

1. Установить Django Web Framework и создать модели Django
2. Создать админ панель для разработанной модели данных
3. Создать контроллеры для обработки данных
4. Поработать с адресацией

Выполнение работы:

Задача 1. Установка Django Web Framework и создание модели Django

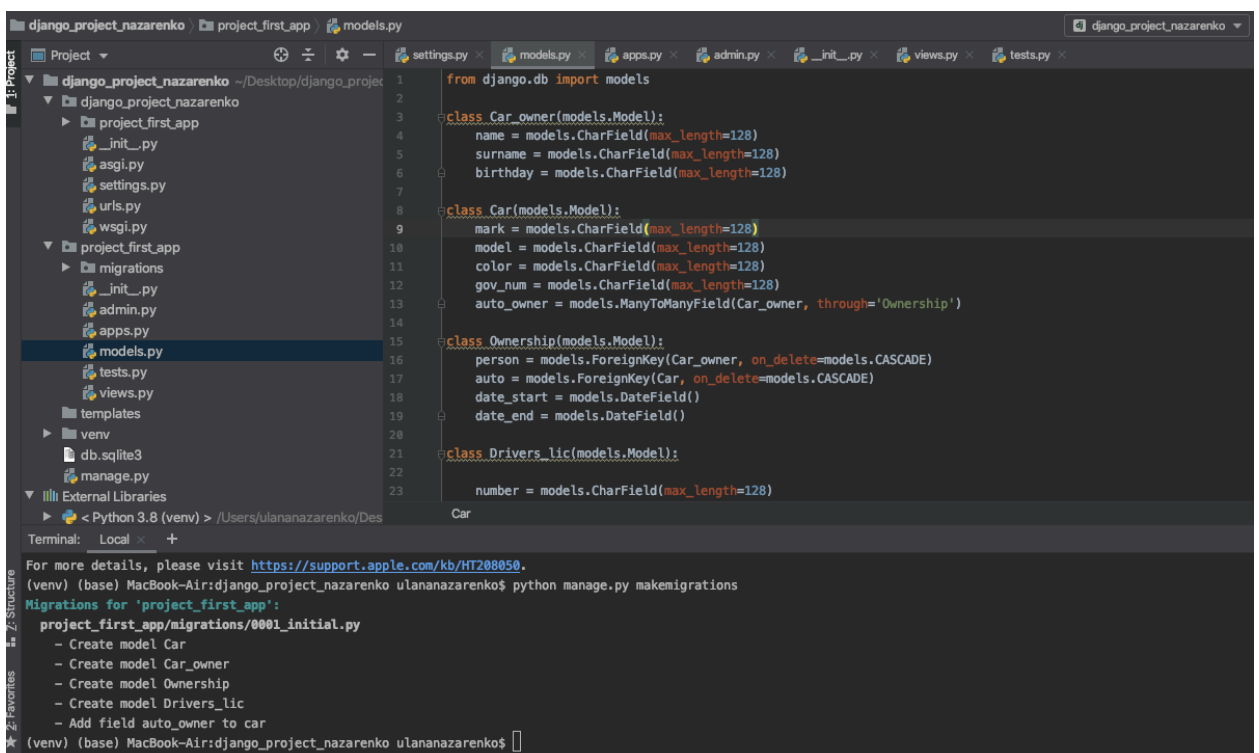
1. Устанавливаем Django Web Framework. Джанго проект называем “django_project_nazarenko”. Джанго приложение называем “project_first_app”.



2. Записываем “project_first_app” в определении приложения в файле “settings.py”.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'project_first_app',  
]
```

3. Создаем модель данных «Автосалон» Django



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows the project structure for 'django_project_nazarenko', including 'project_first_app' and its subdirectories. The main editor window displays the 'models.py' file for 'project_first_app'. The code defines three models: 'Car_owner', 'Car', and 'Ownership'. 'Car_owner' has fields for name, surname, and birthday. 'Car' has fields for mark, model, color, gov_num, and a many-to-many relationship with 'Car_owner' through 'Ownership'. 'Ownership' has foreign key relationships with 'Car_owner' and 'Car', and date fields for 'date_start' and 'date_end'. 'Drivers_lic' has a 'number' field. The terminal shows the command 'python manage.py makemigrations' and the output indicating that migrations were created for 'project_first_app', including creating models 'Car', 'Car_owner', 'Ownership', and 'Drivers_lic', and adding the 'auto_owner' field to 'Car'.

```
from django.db import models  
  
class Car_owner(models.Model):  
    name = models.CharField(max_length=128)  
    surname = models.CharField(max_length=128)  
    birthday = models.CharField(max_length=128)  
  
class Car(models.Model):  
    mark = models.CharField(max_length=128)  
    model = models.CharField(max_length=128)  
    color = models.CharField(max_length=128)  
    gov_num = models.CharField(max_length=128)  
    auto_owner = models.ManyToManyField(Car_owner, through='Ownership')  
  
class Ownership(models.Model):  
    person = models.ForeignKey(Car_owner, on_delete=models.CASCADE)  
    auto = models.ForeignKey(Car, on_delete=models.CASCADE)  
    date_start = models.DateField()  
    date_end = models.DateField()  
  
class Drivers_lic(models.Model):  
    number = models.CharField(max_length=128)  
  
Car
```

Terminal:
Local
+
For more details, please visit <https://support.apple.com/kb/HT208050>.
(venv) (base) MacBook-Air:django_project_nazarenko ulananazarenko\$ python manage.py makemigrations
Migrations for 'project_first_app':
 project_first_app/migrations/0001_initial.py
 - Create model Car
 - Create model Car_owner
 - Create model Ownership
 - Create model Drivers_lic
 - Add field auto_owner to car
(venv) (base) MacBook-Air:django_project_nazarenko ulananazarenko\$

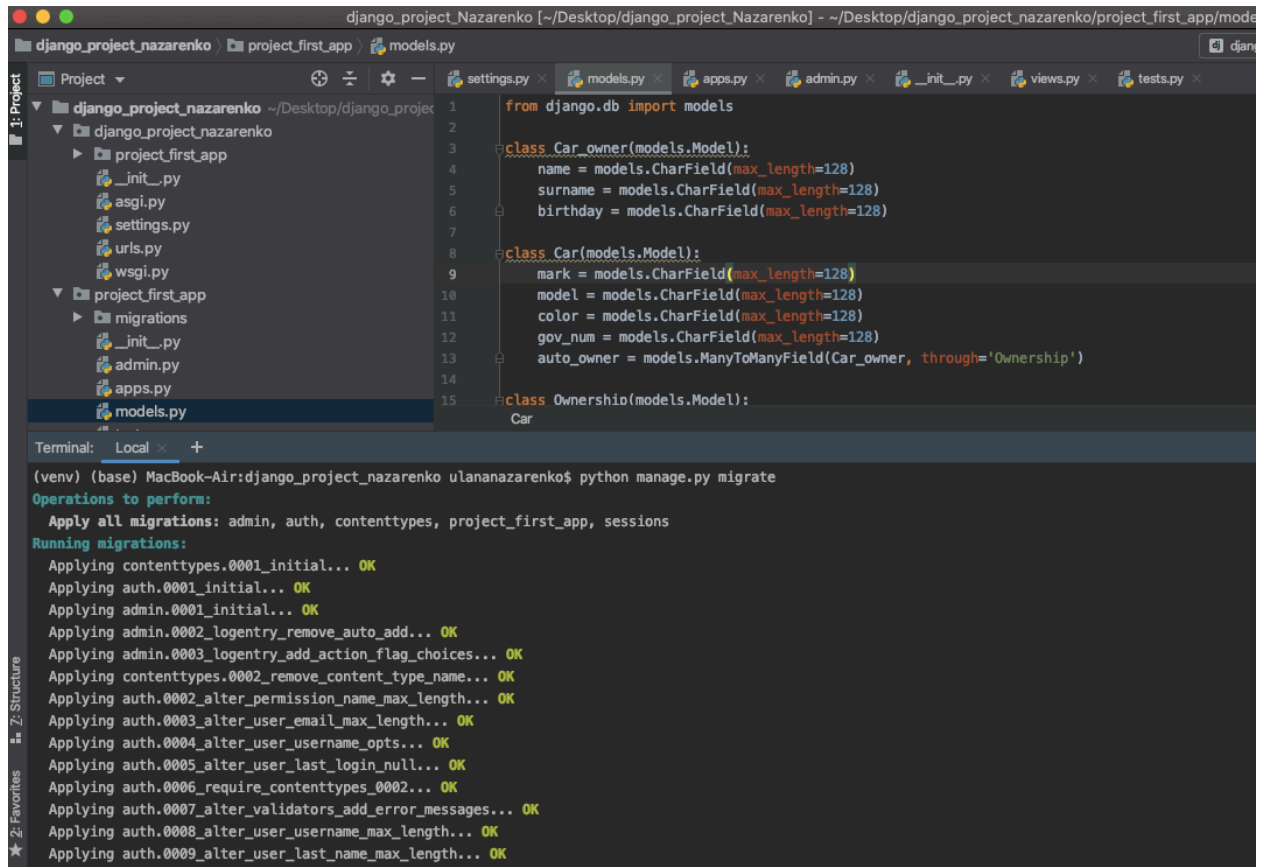
- 3.1 Применяем модель данных к базе данных в SQLite.

python manage.py – Создает миграции на основе имеющейся модели данных.

Созданные миграции доступны в файле миграций, их можно изменить вручную.

python manage.py migrate – Применяет миграции к базе данных.

Создаем миграции таким образом и применяем их к базе данных



The screenshot shows a code editor with a project structure on the left and a terminal at the bottom. The project structure includes a folder named 'project_first_app' containing files like '_init_.py', 'asgi.py', 'settings.py', 'urls.py', 'wsgi.py', and a 'migrations' folder. The main editor window displays the 'models.py' file with the following code:

```
1 from django.db import models
2
3 class Car_owner(models.Model):
4     name = models.CharField(max_length=128)
5     surname = models.CharField(max_length=128)
6     birthday = models.CharField(max_length=128)
7
8 class Car(models.Model):
9     mark = models.CharField(max_length=128)
10    model = models.CharField(max_length=128)
11    color = models.CharField(max_length=128)
12    gov_num = models.CharField(max_length=128)
13    auto_owner = models.ManyToManyField(Car_owner, through='Ownership')
14
15 class Ownership(models.Model):
16    Car
```

The terminal window shows the command 'python manage.py migrate' being executed. The output indicates that migrations for 'admin', 'auth', 'contenttypes', 'project_first_app', and 'sessions' are being applied. The migrations are applied successfully, with each step marked 'OK'.

Скачиваем DBbrowser для sqlite и открываем созданную базу.

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change
PROJECT_FIRST_APP		
Car_owners	+ Add	Change
Cars	+ Add	Change
Drivers_lics	+ Add	Change
Ownerships	+ Add	Change

Задача 2. Создание админ панели для разработанной модели данных

1. Регистрируем владельца авто в “admin.py”:

```
from django.contrib import admin

from .models import Car_owner, Car, Ownership, Drivers_lic

admin.site.register(Car_owner)
admin.site.register(Car)
admin.site.register(Ownership)
admin.site.register(Drivers_lic)
```

2. Создаем суперпользователя:

```
(venv) (base) MacBook-Air:django_project_nazarenko ulananazarenko$ python manage.py createsuperuser
Username (leave blank to use 'ulananazarenko'): Uliana
```

3. Создаем двух владельцев машин и четыре автомобиля:

Django administration

Home › Project_First_App › Car_owners › Add car_owner

Add car_owner

Name:	<input type="text" value="Tom"/>
Surname:	<input type="text" value="Smith"/>
Birthday:	<input type="text" value="1997-06-09"/> Today

Note: You are 3 hours ahead of server time.

Select car_owner to change

Action: 0 of 2 selected

<input type="checkbox"/>	CAR_OWNER
<input type="checkbox"/>	Car_owner object (3)
<input type="checkbox"/>	Car_owner object (2)

2 car_owners

Select car to change

Action: 0 of 4 selected

<input type="checkbox"/>	CAR
<input type="checkbox"/>	Car object (4)
<input type="checkbox"/>	Car object (3)
<input type="checkbox"/>	Car object (2)
<input type="checkbox"/>	Car object (1)

4 cars

Задача 3. Создание контроллеров для обработки данных

1. Создаем в файле `views.py` (находится в папке нашего приложения) контроллер, который выведет из базы данные о владельце автомобиля.

```
from django.http import Http404
from django.shortcuts import render
from .models import Car_owner

def detail(request, Car_owner_id):
    try:
        p = Car_owner.objects.get(pk=Car_owner_id)
    except Car_owner.DoesNotExist:
        raise Http404("Car owner does not exist")
    return render(request, 'owner.html', {'Car_owner': p})
```

2. Создаем страницу html-шаблона `owner.html` в папке `templates`.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Car owner</title>
</head>
<body>
    Имя: {{Car_owner.name}}, Фамилия: {{Car_owner.surname}}
</body>
</html>

```

Задача 4. Работа с адресацией

1. Создать файл `urls.py` в папке “`project_first_app`”
2. Импортировать файл юрлов приложения в проект (модифицируем файл `urls.py` в той папке, в которой хранится файл `setting.py`).

```

from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('project_first_app.urls')),
]

```

3. Описываем в файле юрл адресов приложения, такой юрл адрес, который сможет обратиться к контроллеру и вывести страницу, которая должна быть отрендерена контроллером.

```

from django.urls import path
from project_first_app.views import detail

urlpatterns = [
    path('owner/<int:Car_owner_id>', detail),
]

```


В итоге по адресу “127.0.0.1:8000/owner/1” получаем страницу с информацией о первом владельце автомобилей.

Имя: Tom, Фамилия: Smith

Выводы: в ходе выполнения данного практического задания были получены навыки работы в «Django Web Framework», а также создания модели.