Московский государственный технический университет им. Н.Э.Баумана

Защищено:

Гапанюк Ю.Е.

"__"_____2016 г.

Демонстрация ЛР:

Гапанюк Ю.Е.

"__"_____2016 г.

Отчет по лабораторной работе №7
по курсу РИП

Вариант № <26>

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-52

_____

(подпись)

Чекулина М.Ю.

"__"_____2016 г.

Москва, МГТУ - 2016

1. Задание

Основная цель данной лабораторной работы – научиться обрабатывать веб-формы на стороне приложения, освоить инструменты, которые предоставляет Django, по работе с формами. Также в этой лабораторной работе вы освоите инструменты Django по работе с авторизацией и реализуете простейшую авторизацию. Напоследок, вы познакомитесь с инструментом администрирования Django – как в несколько строчек кода сделать панель администратора сайта.
1. Создайте view, которая возвращает форму для регистрации.
Поля формы:
• Логин
• Пароль
• Повторный ввод пароля
• Email
• Фамилия
• Имя
2. Создайте view, которая возвращает форму для авторизации.
Поля формы:
• Логин
• Пароль
3. При отправке формы регистрации во view проверять каждый параметр по правилам валидации, если валидация всех полей пройдена, то создавать пользователя и делать перенаправление на страницу логина, а ошибки, если они есть, выводить над формой.
Правила валидации:
• Логин не меньше 5 символов
• Пароль не меньше 8 символов
• Пароли должны совпадать
• Все поля должны быть заполнены
• Логин – уникален для каждого пользователя
4. При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать.
5. Переписать view регистрации с использованием Django Form, правила валидации удалить из view, использовать встроенный механизм валидации полей.
6. Во view авторизации реализовать логин при POST запросе. При успешной авторизации должен происходить переход на страницу успешной авторизации.
7. Страница успешной авторизации должна проверять, что пользователь авторизован. Иначе делать перенаправление на страницу авторизации.
8. Реализовать view для выхода из аккаунта.
9. Заменить проверку на авторизацию на декоратор login_required
10. Добавить superuser’a через комманду manage.py
11. Подключить django.contrib.admin и войти в панель администрирования.
12. Зарегистрировать все свои модели в django.contrib.admin
13. Для выбранной модели настроить страницу администрирования:
• Настроить вывод необходимых полей в списке
• Добавить фильтры
• Добавить поиск
• Добавить дополнительное поле в список.

2. Код

urls.py

```
from django.conf.urls import include, url
from django.conf.urls.static import static
from django.contrib import admin

from App import settings
from lab.views import registration, main, login_success, authorization, logout_view,
error_auth

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'', include('lab.urls')),
```

```
    url(r'^registration/', registration, name='registration'),
    url(r'^authorization/', authorization, name='authorization'),
    url(r'^success/', login_success),
    url(r'^error/', error_auth, name='error_auth'),
    url(r'^logout/', logout_view, name='logout'),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

```
from django.shortcuts import render
from lab.models import Car, Address, Sales, RegistrationForm, AuthorizationForm
from lab.models import User1
from django.views.generic import View
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger


# Create your views here.
# def base(request):
#     return render(request, 'lab/base.html', {})

def main(request):
    return render(request, 'lab/main.html', {})


class CarView(View):
    def get(self, request):
        model_row = Car.objects.all()
        page = request.GET.get('page')
        paginator = Paginator(model_row, 1)
        try:
            model_row = paginator.page(page)
        except PageNotAnInteger:
            model_row = paginator.page(1)
        except EmptyPage:
            model_row = paginator.page(paginator.num_pages)
        return render(request, 'lab/model_row.html', {'model_row': model_row})


class AddressView(View):
    def get(self, request):
        address = Address.objects.all()
        return render(request, 'lab/address.html', {'address': address})


class SalesView(View):
    def get(self, request):
        sales = Sales.objects.all()
        return render(request, 'lab/sales.html', {'sales': sales})


class UserView(View):
    def get(self, request):
        users = User1.objects.all()
        return render(request, 'lab/users.html', {'users': users})


def error_auth(request):
    return render(request, 'lab/logout.html')


def registration(request):
    if request.method == 'POST':
```

```
        form = RegistrationForm(request.POST)
        if form.is_valid():
            user = form.save()
            if user is not None:
                login(request, user)
            return HttpResponseRedirect('/success/')
    else:
        form = RegistrationForm()
    return render(request, 'lab/registration.html', {'form': form})


@login_required(login_url='/error/')
def login_success(request):
    # if not request.user.is_authenticated():
        # return HttpResponseRedirect('/')
    return render(request, 'lab/login.html')


def authorization(request):
    if request.method == 'POST':
        form = AuthorizationForm(request.POST)
        if form.is_valid():
            data = form.cleaned_data
            user = authenticate(username=data.get('username'),
password=data.get('password'))
            if user is not None:
                login(request, user)
                return HttpResponseRedirect('/success/')
    else:
        form = AuthorizationForm()
    return render(request, 'lab/authorization.html', {'form': form})


def logout_view(request):
    logout(request)
    return HttpResponseRedirect('/error/')
```

models.py

```
from django.db import models
from django import forms
from django.contrib.auth import authenticate
from django.contrib.auth.hashers import make_password
from django.contrib.auth.models import User
import pymysql
pymysql.install_as_MySQLdb()

# Create your models here.


class Car(models.Model):
    name = models.CharField(max_length=255, verbose_name='Название модели автомобиля')
    price = models.FloatField(verbose_name='Цена')
    country = models.CharField(max_length=100, verbose_name='Страна-производитель')
    # image_car = models.ImageField(upload_to='lab/static/lab/images',
verbose_name='Изображение модели')
    image_car = models.ImageField(verbose_name='Изображение модели')
    length = models.CharField(max_length=20, verbose_name='Длина автомобиля')
    vehicle_clearance = models.CharField(max_length=15, verbose_name='Клиренс
автомобиля')
    max_speed = models.CharField(max_length=15, verbose_name='Максимальная скорость
автомобиля')
    average_fuel_consumption = models.CharField(max_length=15, verbose_name='Средний
расход топлива в смешанном режиме')
    weight = models.CharField(max_length=15, verbose_name='Масса автомобиля')
    type_of_transmission = models.CharField(max_length=100, verbose_name='Коробка
передач')
    volume = models.CharField(max_length=20, verbose_name='Объем двигателя')
```

```
    description = models.CharField(max_length=1500, verbose_name='Описание')

    def __str__(self):
        return self.name


class Address(models.Model):
    name = models.CharField(max_length=255, verbose_name='Название')
    adr = models.CharField(max_length=255, verbose_name='Адрес')
    time = models.CharField(max_length=255, verbose_name='Время работы')
    phone = models.CharField(max_length=255, verbose_name='Телефон')
    image_adr = models.ImageField(verbose_name='Схема проезда:')


class Sales(models.Model):
    name = models.CharField(max_length=255, verbose_name='Название')
    date = models.CharField(max_length=50, verbose_name='Сроки акции')
    text = models.CharField(max_length=255, verbose_name='Описание')


class User1 (models.Model):
    dop_id = models.IntegerField()
    first_name = models.CharField(max_length=255, verbose_name='Имя')
    last_name = models.CharField(max_length=255, verbose_name='Фамилия')
    age = models.IntegerField(verbose_name='Возраст')
    phone = models.CharField(max_length=20, verbose_name='Контактный телефон')
    email = models.EmailField(verbose_name='Электронная почта')


def __str__(self):
    return self.first_name


#форма для регистрации
class RegistrationForm(forms.Form):
    username = forms.CharField(min_length=5, label='Логин:')
    password = forms.CharField(min_length=8, widget=forms.PasswordInput,
label='Пароль:')
    password2 = forms.CharField(min_length=8, widget=forms.PasswordInput,
label='Повторите ввод:')
    email = forms.EmailField(label='Email:')
    first_name = forms.CharField(max_length=30, label='Введите имя:')
    last_name = forms.CharField(max_length=30, label='Введите фамилию:')

    def clean_username(self):
        username = self.cleaned_data.get('username')
        try:
            user = User.objects.get(username=username)
            raise forms.ValidationError('Логин уже занят')
        except User.DoesNotExist:
            return username

    def clean_password2(self):
        pass1 = self.cleaned_data['password']
        pass2 = self.cleaned_data['password2']
        if pass1 != pass2:
            raise forms.ValidationError('Пароли не совпадают, введите одинаковые
пароли')

    def save(self):
        user = User()
        data = self.cleaned_data
        user.username = data.get('username')
        user.password = make_password(data.get('password'))
        user.email = data.get('email')
        user.first_name = data.get('first_name')
        user.last_name = data.get('last_name')
```

```
        user.is_active = True
        user.is_superuser = False
        user.save()
        return authenticate(username=user.username, password=user.password)

#форма для авторизации
class AuthorizationForm(forms.Form):
    username = forms.CharField(min_length=5, label='Логин пользователя:')
    password = forms.CharField(min_length=8, widget=forms.PasswordInput, label='Пароль
пользователя:')

    def clean(self):
        data = self.cleaned_data
        user = authenticate(username=data.get('username'),
password=data.get('password'))
        if user is not None:
            if user.is_active:
                data['user'] = user
            else:
                raise forms.ValidationError('Пользователь неактивен')
        else:
            raise forms.ValidationError('Неверный логин или пароль')
```

authorization.html

```
{% extends 'lab/base.html' %}

{% block title %}Авторизация{% endblock %}

{% block header %}<h2>Авторизация</h2>{% endblock %}

{% block body %}
    <form action="{% url 'authorization' %}" method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button class="btn-primary" type="submit">Авторизироваться</button>
    </form>

{% endblock %}
```

registration.html

```
{% extends 'lab/base.html' %}

{% block title %}Регистрация{% endblock %}

{% block header %}<h2 class="text-left">Регистрация нового пользователя</h2>{% endblock
%}

{% block body %}
    <form action="{% url 'registration' %}" method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button class="btn-primary" type="submit">Зарегистрироваться</button>
    </form>
{% endblock %}
```

login.html

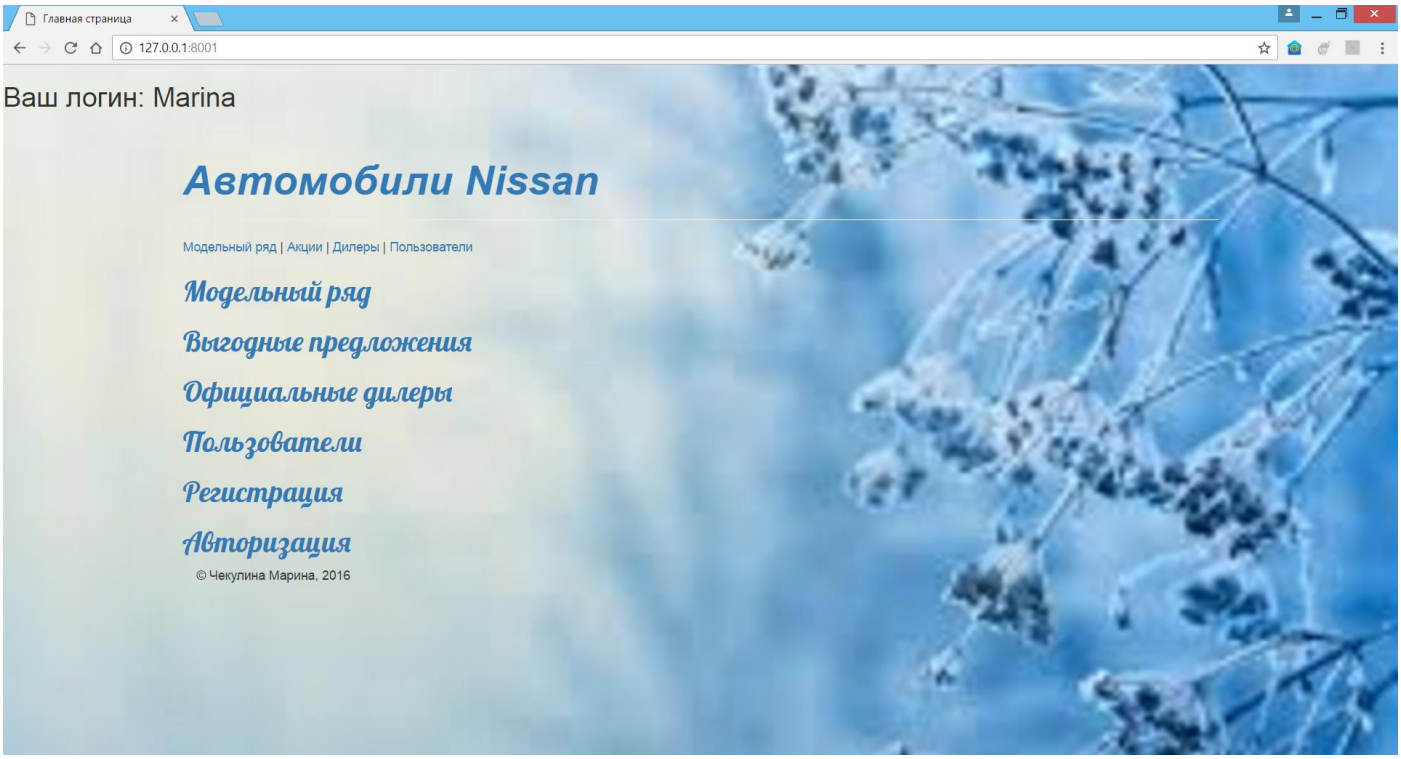
```
{% extends 'lab/base.html' %}

{% block title %}Успешный вход!{% endblock %}


{% block header %}<h2>Вы успешно вошли в систему,{{ user }}!</h2>{% endblock %}

{% block body %}
    <button type="button" class="btn-default"><a href="{% url 'logout'
```

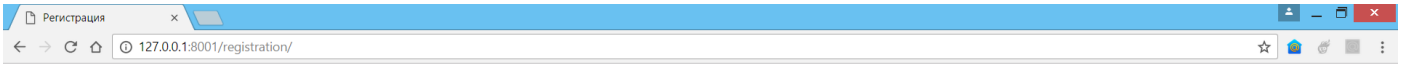
```
%}">Выйти</a></button>
{% endblock %}
```

logout.html

```
{% extends 'lab/base.html' %}

{% block title %}Ошибка!{% endblock %}

{% block header %}<h2>Вы не вошли в систему!</h2>{% endblock %}

{% block body %}
    <button type="button" class="btn-default"><a href="{% url 'registration'
%}">Регистрация</a></button>
    <button type="button" class="btn-default"><a href="{% url 'authorization'
%}">Авторизация</a></button>
{% endblock %}
```

3. Результаты





Ваш логин: Marina

Автомобили Nissan

Регистрация нового пользователя

Логин:

Пароль:

Повторите ввод:

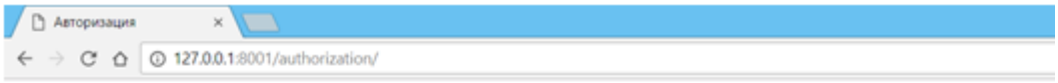
Email:

Введите имя:

Введите фамилию:

Зарегистрироваться

© Чекулина Марина, 2016



Ваш логин: Marina

Автомобили Nissan

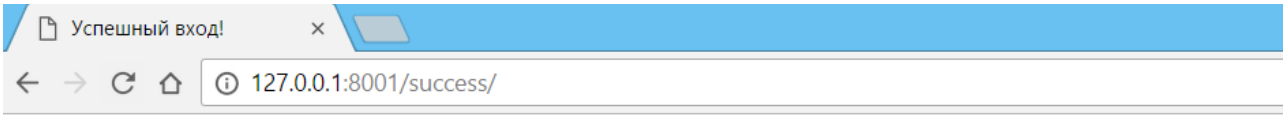
Авторизация

Логин пользователя: user1

Пароль пользователя: ••••••••

Авторизироваться

© Чекулина Марина, 2016



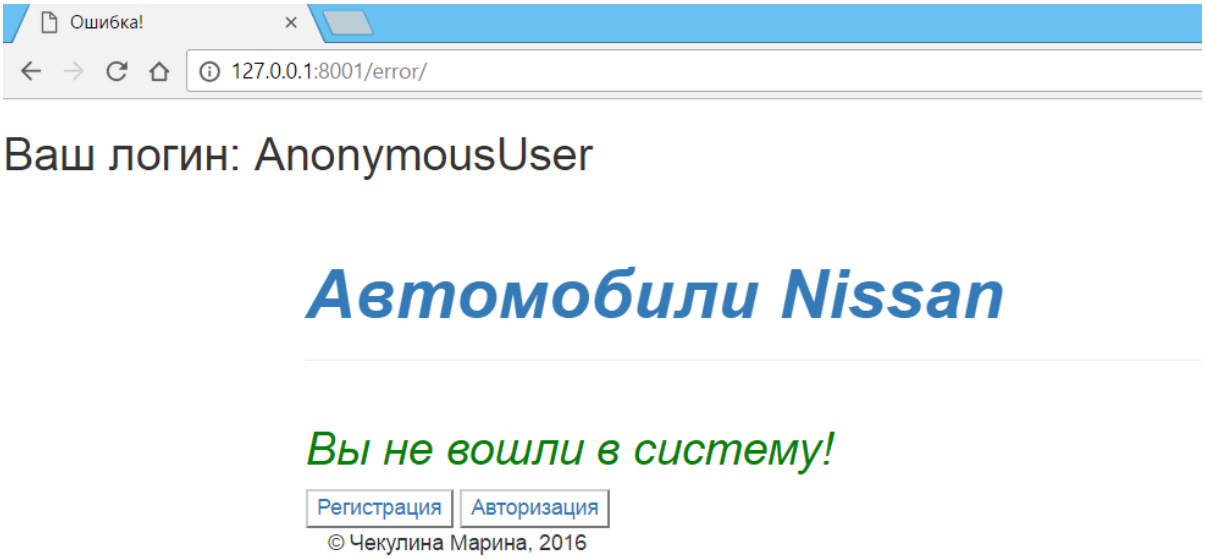
Ваш логин: user1

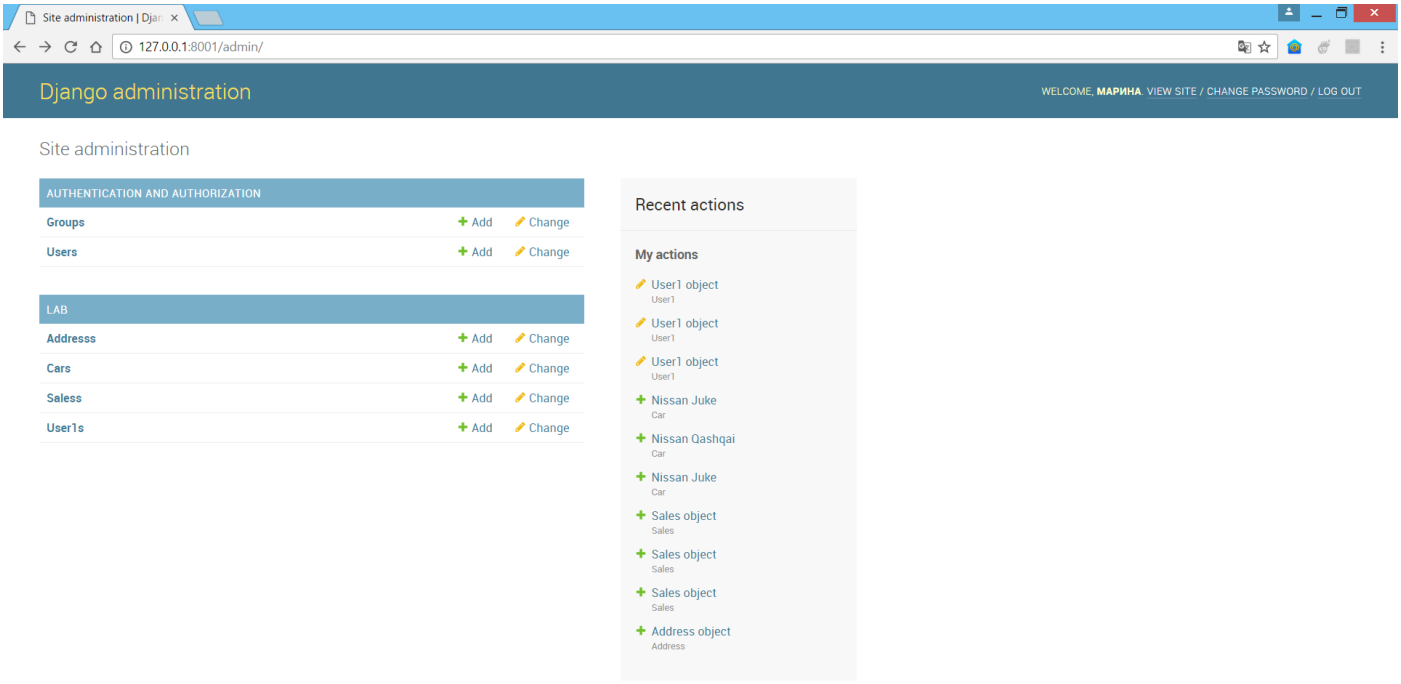
Автомобили Nissan

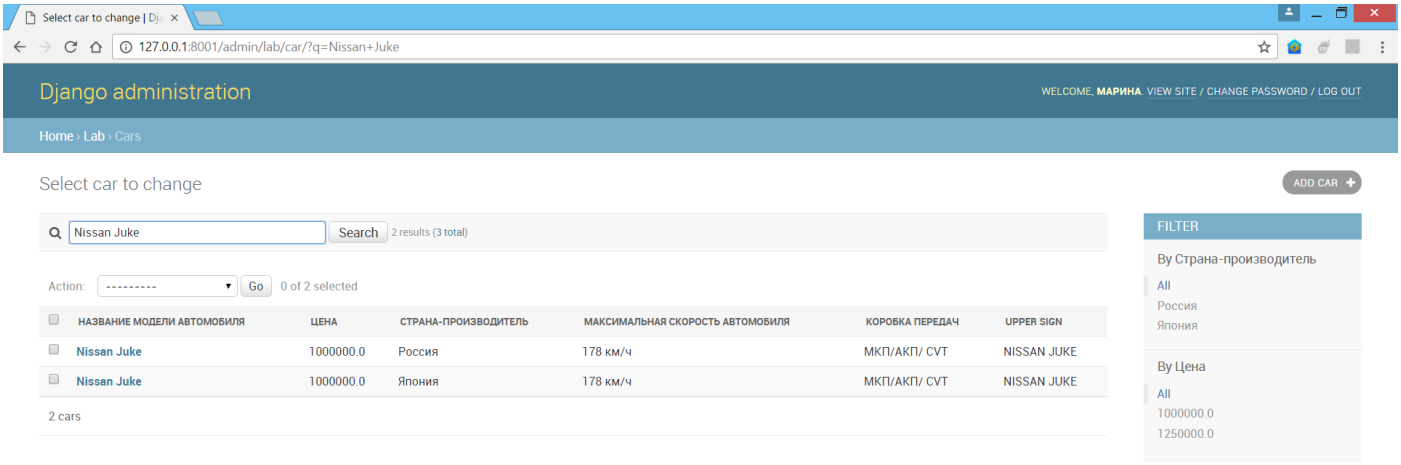
Вы успешно вошли в систему,user1!

Выйти

© Чекулина Марина, 2016


Ошибка!
127.0.0.1:8001/error/
Ваш логин: AnonymousUser
Автомобили Nissan
Вы не вошли в систему!
Регистрация
Авторизация
© Чекулина Марина, 2016


127.0.0.1:8001/admin/
Django administration
WELCOME, МАРИНА. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
LAB
Addresss + Add Change
Cars + Add Change
Saless + Add Change
User1s + Add Change
Recent actions
My actions
User1 object
User1
User1 object
User1
User1 object
User1
Nissan Juke
Car
Nissan Qashqai
Car
Nissan Juke
Car
Sales object
Sales
Sales object
Sales
Sales object
Sales
Address object
Address


Select car to change | Dj
127.0.0.1:8001/admin/lab/car/?q=Nissan+Juke
Django administration
WELCOME, МАРИНА. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Lab › Cars
Select car to change
ADD CAR
Nissan Juke
Search
2 results (3 total)
Action:

Go
0 of 2 selected
НАЗВАНИЕ МОДЕЛИ АВТОМОБИЛЯ
ЦЕНА
СТРАНА-ПРОИЗВОДИТЕЛЬ
МАКСИМАЛЬНАЯ СКОРОСТЬ АВТОМОБИЛЯ
КОРОБКА ПЕРЕДАЧ
UPPER SIGN
Nissan Juke
1000000.0
Россия
178 км/ч
МКП/АКП/ CVT
NISSAN JUKE
Nissan Juke
1000000.0
Япония
178 км/ч
МКП/АКП/ CVT
NISSAN JUKE
2 cars
FILTER
By Страна-производитель
All
Россия
Япония
By Цена
All
1000000.0
1250000.0