

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Лабораторная работа №3
«РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ
ПРИЛОЖЕНИЯ СРЕДСТВАМИ DJANGO И
DJANGO REST FRAMEWORK»
по дисциплине:
«Web-программирование»

Выполнил:
Студент III курса ИМР и П
Группы: D33101
Ф.И.О.: Ван Цюаньюй

Проверил:
Говоров Антон Игоревич

Санкт-Петербург
2021

Цель: Овладеть практическими навыками и умениями реализации web-сервисов средствами Django.

Оборудование: компьютерный класс.

Программное обеспечение:

Python 3.6+, Django 3, Django REST Framework (DRF), PostgreSQL *.

Практическое задание:

Реализовать сайт, используя фреймворк Django 3, Django REST Framework, Djoser и СУБД PostgreSQL *, в соответствии с вариантом задания лабораторной работы.

Задание 7

Создать программную систему, ориентированную на администрацию птицефабрики и позволяющую работать с информацией о работниках фабрики и об имеющихся на ней курах.

О каждой курице должна храниться следующая информация: вес, возраст, порода, количество ежемесячно получаемых от курицы яиц, а также информация о местонахождении курицы.

Сведения о породе включают в себя: название породы, среднее количество яиц в месяц (производительность) и средний вес, номер рекомендованной и содержание диеты.

Диеты могут меняться в зависимости от сезона.

Птицефабрика имеет несколько цехов. В каждой клетке может находиться несколько кур. Код клетки, где находится курица, характеризуется номером цеха, номером ряда в цехе и номером клетки в ряду. Курицы могут пересаживаться из клетки в клетку.

Директор птицефабрики может принять или уволить работника. О работниках птицефабрики в БД должна храниться следующая информация: паспортные данные, зарплата, договор о трудоустройстве, данные об увольнении, закрепленные за работником клетки.

Не должно быть кур, не обслуживаемых ни одним работником. Количество кур может изменяться как в большую, так и в меньшую сторону, в отдельные моменты времени часть клеток может пустовать.

Какое количество яиц получают от каждой курицы данного веса, породы, возраста?

В каком цехе наибольшее количество кур определенной породы?

Среднее количество яиц, которое получает в день каждый работник от обслуживаемых им кур?

Сколько кур каждой породы в каждом цехе?

Chicken List Api

[OPTIONS](#)[GET](#)

GET /poultry/chickens/list/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "breed": "Brazilian",
    "cell": "1",
    "weight": 2,
    "age": 1,
    "egg_amount": 10
  },
  {
    "id": 2,
    "breed": "Brazilian",
    "cell": "4",
    "weight": 3,
    "age": 1,
    "egg_amount": 10
  },
  {
    "id": 3,
    "breed": "Russian",
    "cell": "4",
    "weight": 7,
    "age": 2,
    "egg_amount": 10
  },
  {
    "id": 4,
    "breed": "Chinese",
    "cell": "5",
    "weight": 7,
    "age": 1,
    "egg_amount": 5
  },
  {
    "id": 5,
    "breed": "Russian",
    "cell": "1",
    "weight": 5,
    "age": 2,
    "egg_amount": 9
  },
  {
    "id": 6,
    "breed": "Chinese",
    "cell": "2",
    "weight": 6,
    "age": 3,
    "egg_amount": 6
  }
]
```

Breed List Api

[OPTIONS](#)[GET](#)

GET /poultry/breeds/list/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "breed": "Chinese",
    "productivity": "high",
    "avg_weight": 5,
    "diet": "vegetables"
  },
  {
    "id": 2,
    "breed": "Russian",
    "productivity": "high",
    "avg_weight": 5,
    "diet": "fruits"
  },
  {
    "id": 3,
    "breed": "Brazilian",
    "productivity": "avg",
    "avg_weight": 4,
    "diet": "fruits"
  }
]
```

Chicken Nested Api

[OPTIONS](#)[GET](#)

GET /poultry/chickens/list/nested/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "breed": {
      "id": 3,
      "breed": "Brazilian",
      "productivity": "avg",
      "avg_weight": 4,
      "diet": "fruits"
    },
    "cell": {
      "id": 1,
      "cell": 1,
      "row": 1,
      "tsekh": 2
    },
    "weight": 2,
    "age": 1,
    "egg_amount": 10
  },
  {
    "id": 2,
    "breed": {
      "id": 3,
      "breed": "Brazilian",
      "productivity": "avg",
      "avg_weight": 4,
      "diet": "fruits"
    },
    "cell": {
      "id": 4,
      "cell": 4,
      "row": 4,
      "tsekh": 5
    },
    "weight": 3,
    "age": 1,
    "egg_amount": 10
  },
]
```

Cell List Api

[OPTIONS](#)[GET](#)

GET /poultry/cells/list/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "cell": 1,
    "row": 1,
    "tsekh": 2
  },
  {
    "id": 2,
    "cell": 2,
    "row": 2,
    "tsekh": 3
  },
  {
    "id": 4,
    "cell": 4,
    "row": 4,
    "tsekh": 5
  },
  {
    "id": 5,
    "cell": 5,
    "row": 5,
    "tsekh": 6
  },
  {
    "id": 6,
    "cell": 3,
    "row": 3,
    "tsekh": 7
  }
]
```

Код части:

models.py

```
poultry_app\urls.py x serializers.py x models.py x views.py x poultry\urls.py x
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3 # Create your models here.
4
5 class User(AbstractUser):
6     username = models.CharField(max_length=20, blank=True, null=True, unique=True)
7     passport = models.CharField(max_length=100, blank=True, null=True)
8     salary = models.IntegerField(default=0)
9     cell = models.ManyToManyField('Cell', through="Service")
10
11     REQUIRED_FIELDS = ['first_name', 'last_name', 'passport']
12
13     def __str__(self):
14         return self.username
15
16 class Chicken(models.Model):
17     breed = models.ForeignKey('Breed', on_delete=models.CASCADE)
18     weight = models.IntegerField(default=0)
19     age = models.IntegerField(default=0)
20     egg_amount = models.IntegerField(default=0)
21     cell = models.ForeignKey('Cell', on_delete=models.CASCADE)
22
23 class Breed(models.Model):
24     PROD = (
25         ("low", "low"),
26         ("avg", "average"),
27         ("high", "high")
28     )
29     breed = models.CharField(max_length=50)
30     productivity = models.CharField(max_length=4, choices=PROD)
31     avg_weight = models.IntegerField(default=0)
32     diet = models.TextField()
33
34     def __str__(self):
35         return self.breed
36
37 class Cell(models.Model):
38     cell = models.IntegerField(default=0)
39     row = models.ForeignKey('Row', on_delete=models.CASCADE)
40     tsekh = models.ForeignKey('Tsekh', on_delete=models.CASCADE)
```

views.py

```
poultry_app\urls.py × serializers.py × models.py × views.py × poultry\urls.py ×
1 from .serializers import *
2 from rest_framework import generics
3 from django.shortcuts import render
4 from .serializers import *
5 from rest_framework import generics
6 # Create your views here.
7 class UserListAPIView(generics.ListAPIView):
8     serializer_class = UserSerializer
9     queryset = User.objects.all()
10
11
12 class UserInfoAPIView(generics.RetrieveUpdateDestroyAPIView):
13     queryset = User.objects.all()
14     serializer_class = UserSerializer
15
16
17 class UserCreateAPIView(generics.CreateAPIView):
18     serializer_class = UserSerializer
19     queryset = User.objects.all()
20
21
22 class ChickenListAPIView(generics.ListAPIView):
23     serializer_class = ChickenRelatedSerializer
24     queryset = Chicken.objects.all()
25
26
27 class ChickenInfoAPIView(generics.RetrieveUpdateDestroyAPIView):
28     queryset = Chicken.objects.all()
29     serializer_class = ChickenRelatedSerializer
30
31
32 class ChickenCreateAPIView(generics.CreateAPIView):
33     serializer_class = ChickenSerializer
34     queryset = Chicken.objects.all()
35
36
37 class BreedListAPIView(generics.ListAPIView):
38     serializer_class = BreedSerializer
39     queryset = Breed.objects.all()
```

serializers.py

```
poultry_app\urls.py × serializers.py × models.py × views.py × apps.py × admin.py × tests.py × poultry\urls.py ×
No Python interpreter configured for the project Use C:\Users\gdhjr5233_YOLO\Desktop

1  from .models import *
2  from rest_framework import serializers
3
4  class UserSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = User
7          fields = ['first_name', 'last_name', 'username', 'passport', 'salary', 'cell']
8
9
10 class ChickenSerializer(serializers.ModelSerializer):
11     class Meta:
12         model = Chicken
13         fields = "__all__"
14
15
16 class ChickenRelatedSerializer(serializers.ModelSerializer):
17     breed = serializers.StringRelatedField(read_only=True)
18     cell = serializers.StringRelatedField(read_only=True)
19
20     class Meta:
21         model = Chicken
22         fields = "__all__"
23
24
25 class BreedSerializer(serializers.ModelSerializer):
26     class Meta:
27         model = Breed
28         fields = "__all__"
29
30
31 class ServiceSerializer(serializers.ModelSerializer):
32     class Meta:
33         model = Service
34         fields = "__all__"
35
```

urls.py

```
poultry\urls.py x poultry_app\urls.py x settings.py x
1 from .views import *
2 from django.urls import path
3
4 app_name = "poultry_app"
5
6 urlpatterns = [
7     path('users/list/', UserListAPIView.as_view()),
8     path('users/info/<int:pk>', UserInfoAPIView.as_view()),
9     path('users/create/', UserCreateAPIView.as_view()),
10
11     path('chickens/list/', ChickenListAPIView.as_view()),
12     path('chickens/info/<int:pk>', ChickenInfoAPIView.as_view()),
13     path('chickens/create/', ChickenCreateAPIView.as_view()),
14
15     path('breeds/list/', BreedListAPIView.as_view()),
16     path('breeds/info/<int:pk>', BreedInfoAPIView.as_view()),
17     path('breeds/create/', BreedCreateAPIView.as_view()),
18
19     path('service/list/', ServiceListAPIView.as_view()),
20     path('service/info/<int:pk>', ServiceInfoAPIView.as_view()),
21     path('service/create/', ServiceCreateAPIView.as_view()),
22
23     path('cells/list/', CellListAPIView.as_view()),
24     path('cells/info/<int:pk>', CellInfoAPIView.as_view()),
25     path('cells/create/', CellCreateAPIView.as_view()),
26
27     path('service/list/nested/', ServiceNestedAPIView.as_view()),
28     path('chickens/list/nested/', ChickenNestedAPIView.as_view()),
29 ]
```