Raíssa Azevedo

Como gerar gráficos utilizando Django?

Usando recursos de uma biblioteca chamada **Chart.js.**

O novo projeto será: *charts*

```
pip3 install django django-chartjs django-bootstrap
django-admin startproject chrts .
django-admin startapp core
```

Fazer todas as configurações necessárias do Settings.py.

Criar um arquivo de rotas (urls) na aplicação core:

```
from django.urls import path

from .views import IndexView, DadosJSONView

urlpatterns = [
    path('', IndexView.as_view(), name='index'),
    path('dados/', DadosJSONView.as_view(), name='dados'),
]
```

Na url do Projeto, indicar as rotas criadas:

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

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

Será necessária a criação dos templates:

```
{% load static %}
{% load bootstrap4 %}
<!doctype html>
<html lang="pt-br">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
      {% bootstrap_css %}

    <title>Charts!</title>
  </head>
  <body>
  <div class="container">
      <h1 class="text-primary">Charts!</h1>
  </div>
  <div class="container">
      <canvas id="grafico" width="500" height="400"></canvas>
  </div>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    {% bootstrap_javascript jquery='full' %}
    <script type="text/javascript" src="{% static 'js/Chart.min.js' %}"></script>
    <script type="text/javascript">
        $.get('{% url 'dados' %}', function(data){
          var ctx = $("#grafico").get(0).getContext("2d");
          new Chart(ctx,{
              type: 'line', data: data
            });
          });
    </script>
  </body>
</html>
```

Obs: Nesse projeto não se faz necessário a criação da pasta Static com os arquivos min.js, porque o Django entende que é pra buscar dentro da biblioteca Static indicada no inicio da página HTML.

No Views.py:

```
from random import randint
from django.views.generic import TemplateView
from chartjs.views.lines import BaseLineChartView

class IndexView(TemplateView):
    template_name = 'index.html'

class DadosJSONView(BaseLineChartView):

    def get_label(self):
        '''Retorna 12 labels para a apresentação do x'''
        labels = [
            'Janeiro',
            'Fevereiro',
            'Março',
            'Abril',
            'Maio',
            'Junho',
            'Julho',
            'Agosto',
            'Setembro',
            'Outubro',
            'Novembro',
            'Dezembro'
        ]
        return labels
    def get_providers(self):
        '''Retorna os nomes dos datassets'''
        datasets = [
            'Programação para Leigos',
            'Algoritmos e Lógica de Programaçãp',
            'Programação em C#',
            'Programação em Python',
            'Banco de Dados'
        ]
        return datasets
    def get_data(self):
        '''Retorna 6 datasets para plotar o Gráfico
        E cada linha representa um dataset,
        Cada coluna representa um label
        A quantidade de dados precisa ser igual aos datasets/labels'''
        dados = []
        for l in range(6):
            for c in range(12):
                dado = [
                    randint(1, 200), # Jan
                    randint(1, 200), # Fev
                    randint(1, 200), # Mar
                    randint(1, 200), # Abr
                    randint(1, 200), # Mai
                    randint(1, 200), # Jun
                    randint(1, 200), # Jul
                    randint(1, 200), # Ago
                    randint(1, 200), # Set
                    randint(1, 200), # Out
                    randint(1, 200), # Nov
                    randint(1, 200)  # Dez
                ]
            dados.append(dado)
        return dados
```

O Randint irá gerar dados aleatórios de preenchimento para o gráfico. Gerando um número inteiro de 1 a 199.

Em seguida, executar o migrate:

```
python3 manage.py migrate
```

Em seguida, fazer as configurações para publicação:

```
heroku login
git init
git status
git add .
git commit -m ‘Projeto Finalzado’
```