

**django**

# → O QUE É DJANGO

Django é um Framework Web de alto-nível do Python que encoraja o rápido desenvolvimento e um design limpo e pragmático.

# → PRÁTICA PIRATAS




Gerenciador de Tesouro

127.0.0.1:8000/GerenciadorTesouros

## Gerenciador de Tesouros

Adicionar

Estes são os tesouros acumulados do Barba-Ruiva em suas aventuras

Tesouro	Nome	Valor unitário	Quantidade	Valor total	Editar	
	Coroas	210	4	840.00	Remover	Modificar
	Moedas de Ouro	10	835	8350.00	Remover	Modificar
	Cálices de Ouro	4500	1	4500.00	Remover	Modificar
Total				13690.00		

Yarr Harr, marujo! Aqui é o temido Barba-Ruiva e você deve me ajudar a contabilizar os espólios das minhas aventuras!

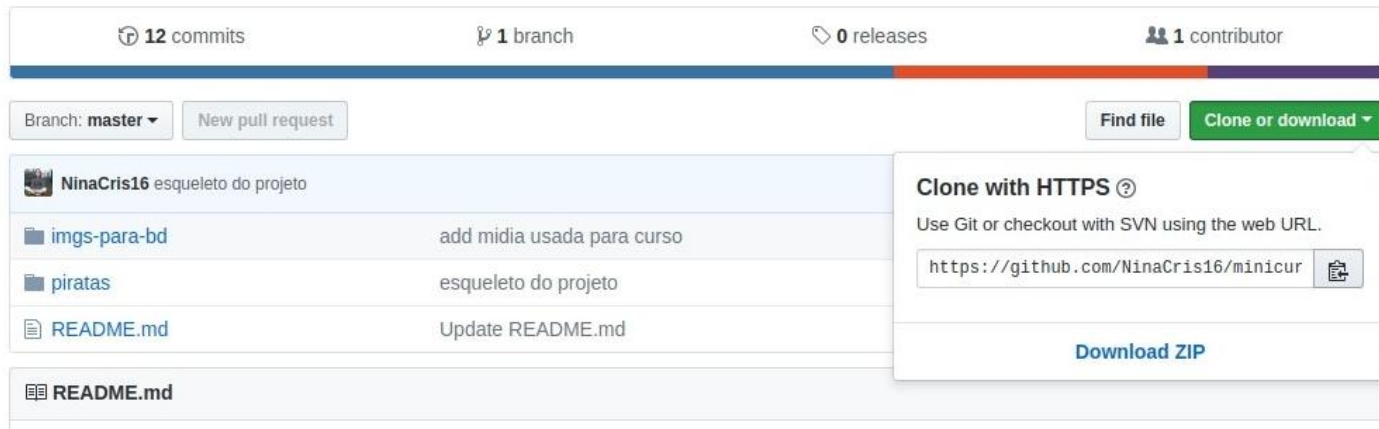
# → OBTENDO O PROJETO NO GITHUB

Acesse o GitHub:

```
https://github.com/NinaCris16/minicurso-django
```

Copie o link do repositório minicurso-django:

Dinâmica do minicurso de django



The screenshot shows the GitHub repository page for 'NinaCris16/esqueleto do projeto'. At the top, it displays repository statistics: 12 commits, 1 branch, 0 releases, and 1 contributor. Below this, there are buttons for 'Branch: master', 'New pull request', 'Find file', and 'Clone or download'. The file list shows 'imgs-para-bd' (add media usada para curso), 'piratas' (esqueleto do projeto), 'README.md' (Update README.md), and another 'README.md' at the bottom. A modal window titled 'Clone with HTTPS' is open, showing the URL 'https://github.com/NinaCris16/minicur' and a 'Download ZIP' button.

12 commits   1 branch   0 releases   1 contributor

Branch: master   New pull request   Find file   Clone or download

**NinaCris16** esqueleto do projeto

imgs-para-bd	add midia usada para curso
piratas	esqueleto do projeto
README.md	Update README.md
README.md	

**Clone with HTTPS**  
Use Git or checkout with SVN using the web URL.  
`https://github.com/NinaCris16/minicur`  
Download ZIP

## → OBTENDO O PROJETO NO GITHUB

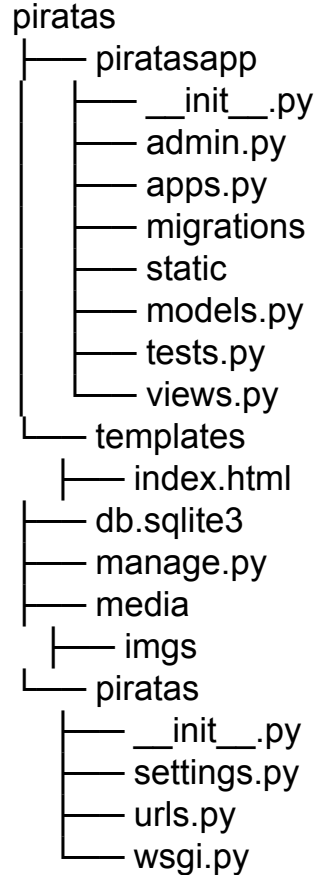
Abra o terminal (Ctrl+Alt+t) e execute:

```
$ cd Área\ de\ Trabalho/
```

```
$ git clone <link_do_git>
```

Mova a pasta piratas que se encontra dentro da pasta minicurso-django para a Área de Trabalho

# → ÁRVORE DE DIRETÓRIO



## → RODANDO A APLICAÇÃO

Abra o terminal (Ctrl+Alt+t) e entre no diretório:

```
$ cd Área\ de\ Trabalho/
```

```
$ cd piratas
```

```
$ python3 manage.py runserver
```

Acesse o endereço fornecido ( <http://127.0.0.1:8000/>) e veja o que aparece.

# → RODANDO A APLICAÇÃO

## Page not found (404)

Request Method: GET

Request URL: http://127.0.0.1:8000/

Using the URLconf defined in `piratas.urls`, Django tried these URL patterns, in this order:

1. `admin/`
2. `^media/(?P<path>.*)$`

The empty path didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Adicione o admin a url <http://127.0.0.1:8000/admin>

### Django administration

Username:

Password:

Log in



## → CRIANDO UM SUPER USUÁRIO

Para que não seja necessário abrir outro terminal ou desligar o servidor pressione (Ctrl+Shift+t) e execute:

```
$ python3 manage.py createsuperuser;
```

Adicione as informações necessárias e em seguida volte ao site do admin.

Feche a página e desligue o servidor (Ctrl+c).

## → MODELS

Um model é uma fonte de informação sobre os dados que se está armazenando.

```
from django.db import models
```

```
class Tesouro(models.Model):  
    nome = models.CharField(max_length = 45)  
    quantidade = models.IntegerField()  
    valor = models.DecimalField(max_digits=10, decimal_places=2)  
    img_tesouro = models.ImageField(upload_to="imgs")
```

# → TEMPLATE

Um template é um simples arquivo de texto que contém a parte dinâmica do site com algumas sintaxes especiais descrevendo como o conteúdo dinâmico vai ser inserido, tal como variáveis, que serão substituídas por valores quando o template for carregado no navegador, e tags que controlam a lógica do template.

# → TEMPLATE - IMPORTAÇÃO

```
<!DOCTYPE html>
{% load static %}
<head>
  <link rel="icon" href="{% static 'imgs/calice.ico'%}">
  <title>Gerenciador de Tesouros</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="{% static 'css/estilos.css'%}">
</head>
<body>
  <h1>Gerenciador de Tesouros</h1>

  <div id="myModal" class="modal">

    <div class="modal-content">
      <span class="close">&times;</span>
      <form action="" method="post" enctype="multipart/form-data">{% csrf_token %}
        {{ form.as_p }}|
        <input type="submit" value="Adicionar" />
      </form>
    </div>
  </div>
</div>
```

## → VIEWS

Uma view é uma função Python que obtém uma requisição web e retorna uma resposta web. Essa resposta pode ser o conteúdo HTML de uma página, um redirecionamento, um erro 404, uma imagem ou diversas outras coisas.

## → VIEWS - TESOUROUTIL

```
class TesouroUtil(object):

    def form_valid(self, view, form):
        lsta_tesouro = Tesouro.objects.filter(nome=form.instance.nome)
        if (form.instance.pk != None):
            lsta_tesouro = lsta_tesouro.exclude(pk=form.instance.pk)
        if len(lsta_tesouro) > 0:
            errors = form._errors.setdefault("nome", ErrorList())
            errors.append(u"nomes iguais não são permitidos")
            return False

        return True

    def get_context_data(self, context):
        lstTesouros = Tesouro.objects.annotate(total=ExpressionWrapper(F('valor') * F('quantidade'), output_field=DecimalField
(max_digits=10, decimal_places=2, blank=True)))
        context["lista_tesouro"] = lstTesouros
        soma = 0
        for tesouro in lstTesouros:
            soma += tesouro.total
        context["preco_total"] = soma
```

## → VIEWS - CREATEVIEW

```
class TesouroInsert(CreateView):

    model = Tesouro
    template_name = "index.html"
    util = TesouroUtil()

    fields=["img_tesouro", "nome", "valor", "quantidade"]

    def form_valid(self, form):
        bol_valid = TesouroInsert.util.form_valid(self, form)
        return super(CreateView, self).form_valid(form) if bol_valid else super(CreateView, self).form_invalid(form)

    def get_context_data(self, **kwargs):
        context = super(TesouroInsert, self).get_context_data(**kwargs)
        TesouroInsert.util.get_context_data(context)
        return context

    def get_success_url(self):
        return reverse('GerenciadorTesouros')

class Meta:
    labels = {
        "img_tesouro" : "Tesouro",
        "nome" : "Nome",
        "valor" : "Valor unitário",
        "quantidade" : "Quantidade"
    }
```

→ URL

Uma URL é uma forma padronizada de representação de serviços na internet, tendo um endereço único, como [www.webfeatures.com.br](http://www.webfeatures.com.br).



## → VIEWS - CREATEVIEW

```
url(r'^GerenciadorTesouros$', views.TesouroInsert.as_view(), name='GerenciadorTesouros'),
```

# → TEMPLATE - TABELA DINÂMICA

```
<tbody>
{% for objTesouro in lista_tesouro %}
  <tr>
    <td></td>
    <td>{{ objTesouro.nome }}</td>
    <td>{{ objTesouro.valor }}</td>
    <td>{{ objTesouro.quantidade }}</td>
    <td>{{ objTesouro.total }}</td>
    <td>
      <form action="" method="post">
        <button id="delete">Remover</button>
      </form>
    </td>
    <td><button data-nome="{{ objTesouro.nome }}" class="update">Modificar</button></td>
  </tr>
{% endfor %}
</tbody>
<tfoot>
  <tr>
    <td colspan="4" id="bot">Total</td>
    <td id="bot" class="soma">{{preco_total}}</td>
    <td colspan="2" id="bot"></td>
  </tr>
</tfoot>
```

## → VIEWS - UPDATEVIEW

```
class TesouroUpdate(UpdateView):
    model = Tesouro
    template_name = "index.html"
    util = TesouroUtil()

    fields=["img_tesouro", "nome", "valor", "quantidade"]

    def form_valid(self, form):
        bol_valid = TesouroUpdate.util.form_valid(self, form)
        return super(UpdateView, self).form_valid(form) if bol_valid else super(UpdateView, self).form_invalid(form)

    def get_context_data(self, **kwargs):
        context = super(TesouroUpdate, self).get_context_data(**kwargs)
        TesouroUpdate.util.get_context_data(context)
        return context

    def get_object(self):
        return Tesouro.objects.get(nome=self.kwargs["nome"])

    def get_success_url(self):
        return reverse('GerenciadorTesouros')
```

```
url(r'^GerenciadorTesouros/Update/(?P<nome>.*)$', views.TesouroUpdate.as_view(),
    name='TesouroUpdate'),
```

# → TEMPLATE - MODAL PARA ATUALIZAÇÃO

```
{% if object.id or form.errors %}
    document.getElementById("myModal").style.display = "block";

{% endif %}
var arrBtnUpdate = document.querySelectorAll(".update");
for(var i =0 ; i<arrBtnUpdate.length ; i++){
    arrBtnUpdate[i].addEventListener("click", function(e) {
        var btnTesouro = e.currentTarget;
        window.location.href = "{% url 'TesouroUpdate' '%' %}" + btnTesouro.dataset.nome;
    });
}
```

## → VIEWS - DELETEVIEW

```
class TesouroDelete(DeleteView):  
    model = Tesouro  
  
    def get_object(self):  
        return Tesouro.objects.get(nome=self.kwargs["nome"])  
  
    def get_success_url(self):  
        return reverse_lazy('GerenciadorTesouros')
```

```
url(r'^GerenciadorTesouros/(?P<nome>.*)$', views.TesouroDelete.as_view(),  
    name='TesouroDelete'),
```

## → TEMPLATE - BOTÃO DE REMOÇÃO

```
<form action="{% url 'TesouroDelete' objTesouro.nome %}" method="post">{% csrf_token %}  
    <button id="delete">Remover</button>  
</form>
```

## → NAVEGANDO EM PASTAS PELO TERMINAL

- Para acessar um diretório seguinte

```
$ cd <nome_pasta>
```

- Para acessar um diretório anterior

```
$ cd ..
```

- Para visualizar o conteúdo de uma pasta

```
$ ls
```