

# Django

Arquivos estáticos, templates,  
exibindo e aceitando convites

Ely – [elydasilvamiranda \[at\] gmail.com](mailto:elydasilvamiranda[at]gmail.com)

1

## Arquivos estáticos e template base

- Vamos melhorar sua apresentação da nossa aplicação;
- Como vimos, isso é responsabilidade do CSS;
- Uma alternativa a criar um estilo do zero é usar estilos pré-prontos, como estilos do bootstrap;
- Como já vimos, ele é um framework CSS criado pelo pessoal do Twitter.

2

## Classes do bootstrap utilizadas

- nav: define um menu de navegação, com realce ao passar o mouse;
- nav-pills: coloca os itens de menu na horizontal;
  - <http://getbootstrap.com/components/#nav>
- pull-right: alinha um conteúdo à direita;
  - <http://getbootstrap.com/components/#dropdowns-alignment>
- text-muted: cor de texto cinza;
  - <https://v4-alpha.getbootstrap.com/utilities/colors/>
- row, content: agrupamento horizontal;
- panel: define um quadro, um painel;
- panel-default: define a separação entre título e corpo;
- panel-heading: cabeçalho de um panel;
- panel-body: corpo de um panel;
  - <http://getbootstrap.com/components/#panels>
- list-group e list-group-item: definem um agrupamento em uma lista não ordenada;
  - <https://v4-alpha.getbootstrap.com/components/list-group/>
- well e well-sm: definem bordas arredondadas e um fundo cinza em um quadro;
  - [https://www.w3schools.com/bootstrap/bootstrap\\_wells.asp](https://www.w3schools.com/bootstrap/bootstrap_wells.asp)

3

## Servindo arquivos estáticos

- A princípio, o Django exige que criemos uma rota e uma função de view que as retornam:
  - Precisariamos criar uma rota e view view para cada arquivo CSS, JS e imagens?
  - Isso é impraticável, tendo em vista que um site pode ter zilhões de imagens e scripts;
- Para contornar essa exigência, o Django permite colocar arquivos em uma pasta static:
  - Tudo que estiver dentro dela será acessível pelo navegador do usuário diretamente;
  - Assim, a pasta static é o local ideal para servir que não precisam de processamento.

4

## Importando arquivos estáticos

- Precisamos então de uma pasta com o seguinte caminho: `connectedin/perfis/static`;
- Baixe o arquivo `static.zip` e decompacte-o;
- Ao descompactar o arquivo teremos as pastas `fonts`, `img`, `scripts` e `styles`;
- Para carregar os arquivos estáticos no Django utiliza-se a tag:
  - `{% load staticfiles %}`;
  - Assim, o Django entende que os acessos a arquivos serão carregados da pasta `static`.

5

## Aplicando um estilo

```
<!-- connectedin/perfis/templates/perfis/index.html -->
{% load staticfiles %}
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>ConnectedIn</title>
    <link href="{% static 'styles/bootstrap.css' %}"
          rel="stylesheet"/>
    <link href="{% static 'styles/main.css' %}"
          rel="stylesheet"/>
  </head>
  <body>
    <!-- código omitido -->
  </body>
</html>
```



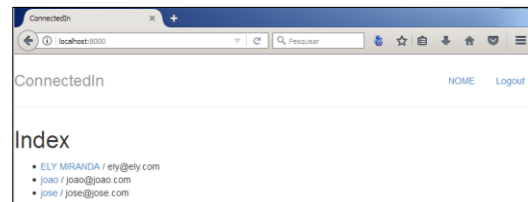
6

## Definindo um menu

```
<!-- connectedin/perfis/templates/perfis/index.html -->
{% load staticfiles %}
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title>ConnectedIn</title>
    <link href="{% static 'styles/bootstrap.css' %}" rel="stylesheet"/>
    <link href="{% static 'styles/main.css' %}" rel="stylesheet"/>
  </head>
  <body>
    <div class="header">
      <ul class="nav nav-pills pull-right">
        <li class=""><a href="{% url 'index' %}">HOME</a></li>
        <li class=""><a href="/logout/">Logout</a></li>
      </ul>
      <h3 class="text-muted">ConnectedIn</h3>
    </div>
    <!-- código posterior omitido -->
  </body>
</html>
```

7

## Definindo um menu



8

## Criando um template base

- Devemos aplicar esse estilo a cada uma das páginas? Não ficaria repetitivo?
- E se algo mudasse no estilo da aplicação? Mudar em todas as páginas?
- O ideal é termos uma página já com o menu e os arquivos de Bootstrap e que sirva como modelo;
- Assim, nos preocuparíamos apenas com o conteúdo específico de cada página;
- O Django suporta isso com o uso de templates;
- O primeiro passo é criar a página: `connectedin/perfis/templates/base.html`

9

## Criando um template base

```
<!-- connectedin/perfis/template/base.html -->
{% load staticfiles %}
<!DOCTYPE html>
<html lang="pt">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <link href="{% static 'styles/bootstrap.css' %}" rel="stylesheet">
    <link href="{% static 'styles/main.css' %}" rel="stylesheet">
    <title>ConnectedIn</title>
  </head>
  <body>
    <div class="container">
      <div class="header">
        <ul class="nav nav-pills pull-right">
          <li class=""><a href="{% url 'index' %}">Nome</a></li>
          <li class=""><a href="/logout/">Logout</a></li>
        </ul>
        <h3 class="text-muted">ConnectedIn</h3>
      </div>
      <div class="row content">
        <!-- conteúdo da página entraria aqui -->
      </div>
    </div>
  </body>
</html>
```

10

## Criando um template base

- Para definir onde estará o código personalizado de cada página, o Django permite definir blocos:

```
<!-- connectedin/perfis/template/base.html -->
{% load staticfiles %}
<!DOCTYPE html>
<html lang="pt">
  <head>
    <!-- código omitido -->
  </head>
  <body>
    <div class="container">
      <!-- código omitido -->
      <div class="row content">
        {% block body %}
        {% endblock %}
      </div>
    </div>
  </body>
</html>
```

11

## Herdando de um template

- Vamos fazer com que as páginas `index.html` e `perfil.html` herdem do template;
- Começando pela `index.html`, retirando o código já presente no arquivo base;
- Deixamos apenas o conteúdo, sem cabeçalho, menu e eventual rodapé;
- A herança do template se dá através das tags:  

```
{% extends "base.html" %}
```

```
{% block body %}
```

```
{% endblock %}
```

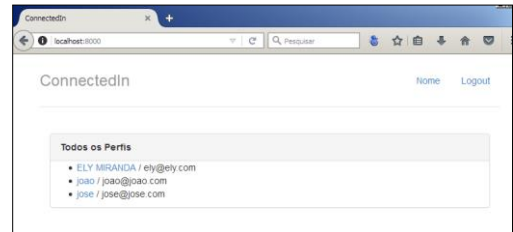
12

## Página index.html

```
<!-- connectedin/perfis/templates/index.html -->
{% extends "base.html" %}
{% block body %}
<div class="col-lg-12">
  <div class="panel panel-default">
    <div class="panel-heading">
      <strong>Todos os Perfis</strong>
    </div>
    {% if perfis %}
    <ul>
      {% for perfil in perfis %}
      <li>
        <a href="{% url 'exibir' perfil.id %}">{{ perfil.nome }}</a>
        / {{ perfil.email }}
      </li>
      {% endfor %}
    </ul>
    {% else %}
    <p>Nenhum Perfil encontrado</p>
    {% endif %}
  </div>
</div>
{% endblock %}
```

13

## Página index.html



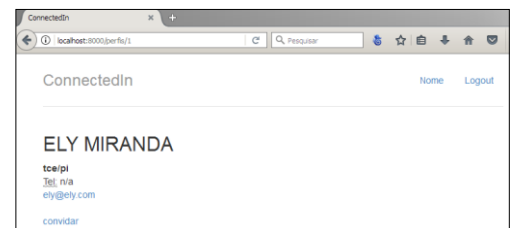
14

## Página perfil.html

```
<!-- connectedin/perfis/template/perfil.html -->
{% extends "base.html" %}
{% block body %}
<div class="row">
  <div class="col-lg-12">
    <h2 style="margin-top:0">{{ perfil.nome }}</h2>
    <address>
      <strong>{{ perfil.nome_empresa }}</strong><br/>
      <abbr title="Telefone">Tel:</abbr> {{ perfil.telefone }}<br/>
      <a href="mailto:{{ perfil.email }}">{{ perfil.email }}</a>
    </address>
    <a href="{% url 'convidar' perfil.id %}">convidar</a>
  </div>
</div>
{% endblock %}
```

15

## Página perfil.html



16

## Exibindo convites recebidos

- Criaremos um painel que exibirá todos os convites recebidos pelo usuário logado;
- Ao lado de cada convite, vamos adicionar o link "aceitar" para cada convite;
- Precisaremos também alterar a view para que disponibilize o usuário logado;

17

## Exibindo convites recebidos

```
<!-- connectedin/perfis/template/index.html -->
{% extends "base.html" %}
{% block body %}
<div class="col-lg-12">
  <!-- código omitido -->
  <div class="panel panel-default">
    {% if perfil_logado.convites_recebidos.count %}
    <div class="panel-heading">
      <strong>Convites aguardando aprovação</strong>
    </div>
    <ul class="list-group">
      {% for convite in perfil_logado.convites_recebidos.all %}
      <li class="list-group-item">
        {{ convite.solicitante.nome }}
        <a href="#" class="pull-right">aceitar</a>
      </li>
      {% endfor %}
    </ul>
    {% else %}
    <div class="panel-body">
      <p>Nenhum convite recebido :(</p>
    </div>
    {% endif %}
  </div>
</div>
{% endblock %}
```

18

## Exibindo convites recebidos

- Nota:
  - As funções count e all não precisam de parêntesis quando estão entre as tags

19

## Disponibilizando o perfil\_logado

```
# connectedin/perfis/views.py
# código anterior omitido

def index(request):
    return render(request, 'index.html',
                  {'perfil' : Perfil.objects.all(),
                   'perfil_logado' : get_perfil_logado(request)})

def exibir(request, perfil_id):
    # código comentado
    return render(request, 'perfil.html',
                  {'perfil' : perfil,
                   'perfil_logado' : get_perfil_logado(request)})

# código posterior omitido
```

20

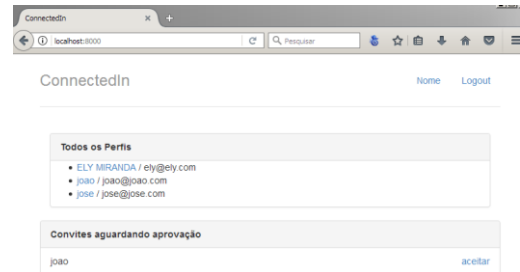
## Preparando convites

- Precisamos adicionar convites para o usuário logado (o perfil ainda fixo) ;
- Nota, preencha os ids abaixo com os ids que estiverem no seu banco:

```
python manage.py shell
>>> from perfis.models import Convite, Perfil
>>> a_convitar = Perfil.objects.get(id=1)
>>> solicitante = Perfil.objects.get(id=2)
>>> solicitante.convitar(a_convitar)
```

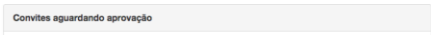
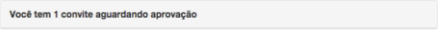
21

## Exibindo os convites



22

## Exibindo o total de convites

- Em vez de exibir:  

- Podemos exibir:  

- Para isso, basta usar a variável:  
`{{ perfil_logado.convites_recebidos.count }}`
- No caso de mais de um convite, devemos usar o recurso "pluralização" do próprio Django;
- Esses e outros filtros serão estudados posteriormente.

23

<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/#built-in-filter-reference>

## Exibindo o total de convites

```
<!-- connectedin/perfis/templates/index.html -->
{% extends "base.html" %}
{% block body %}
<div class="col-lg-12">
  <!-- código omitido -->
  <div class="panel panel-default">
    {% if perfil_logado.convites_recebidos.count %}
      <div class="panel-heading">
        <strong>
          Você tem
          {{ perfil_logado.convites_recebidos.count }}
          convite{{ perfil_logado.convites_recebidos.count|pluralize }}
          aguardando aprovação
        </strong>
      </div>
    </div>
  <!-- código omitido -->
  {% endif %}
</div>
{% endblock %}
```

<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/#pluralize>

24

## Exibindo o total de convites

- Melhorando declaração do contador com with:

```
<!-- connectedin/perfis/templates/index.html -->
{% extends "base.html" %}
{% block body %}
<div class="col-lg-12">
  <!-- código omitido -->
  <div class="panel panel-default">
    {% with total_de_convites
      = perfil_logado.Convites_recebidos.count %}
    {% if total_de_convites %}
      <div class="panel-heading">
        <strong>
          Você tem
          {{ total_de_convites }} convite{{ total_de_convites|pluralize }}
          aguardando aprovação
        </strong>
      </div>
    <!-- código omitido -->
    {% endif %}
  </div>
</div>
https://docs.djangoproject.com/en/1.11/ref/templates/builtins/#with
{% endblock %}
```

25

## Aceitando convites

- Para implementarmos o aceite de convites, precisaremos definir uma função no views.py:
- Além disso, devemos adicionar uma nova rota no urls.py:

```
# connectedin/perfis/views.py
# código comentado
def aceitar(request, convite_id):
    pass

# connectedin/perfis/urls.py
from django.conf.urls import url
from perfis import views

urlpatterns = [
    # urls omitidas

    url(r'^convite/(?P<convite_id>\d+)/aceitar$',
        views.aceitar, name='aceitar')
]
```

26

## Aceitando convites

- Atualizando o link aceitar:

```
<!-- connectedin/perfis/templates/index.html -->
<!-- código anterior comentado -->

<a href="{% url 'aceitar' convite.id %}" class="pull-right">aceitar</a>

<!-- código posterior comentado -->
```

27

## Função aceitar na view

```
# connectedin/perfis/views.py
# adicionando a classe Convite na importação já existente
from perfis.models import Perfil, Convite

# código omitido, exibindo apenas a função "aceitar"

def aceitar(request, convite_id):
    convite = Convite.objects.get(id=convite_id)
    convite.aceitar()
    return redirect('index')
```

28

## Adicionando contatos ao Perfil

```
# connectedin/perfis/models.py
from django.db import models
class Perfil(models.Model):
    nome = models.CharField(max_length=255, null=False)
    email = models.CharField(max_length=255, null=False)
    telefone = models.CharField(max_length=15, null=False)
    nome_empresa = models.CharField(max_length=255, null=False)
    # Novo atributo
    contatos = models.ManyToManyField('self')
    def convidar(self, perfil_convidado):
        Convite(solicitante=self, convidado=perfil_convidado).save()

# código da classe Convite omitido
```

29

## Implementando o aceitar na classe Convite

```
# connectedin/perfis/models.py
# código da classe Perfil omitido

class Convite(models.Model):
    solicitante = models.ForeignKey(Perfil,
                                    related_name='convites_feitos')
    convidado = models.ForeignKey(Perfil,
                                  related_name='convites_recebidos')

    def aceitar(self):
        self.convidado.contatos.add(self.solicitante)
        self.solicitante.contatos.add(self.convidado)
        self.delete()
```

30

## Sincronizando o banco

- > python manage.py makemigrations
- > python manage.py migrate

31

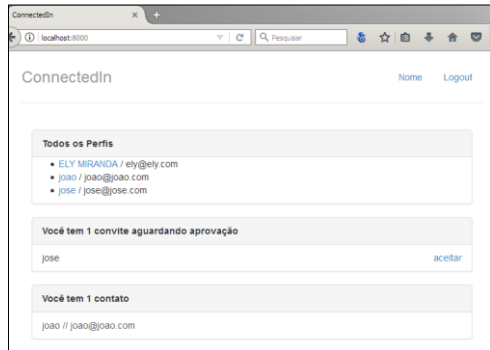
## Adicionando um painel de contatos

```
<!-- connectedin/perfis/templates/index.html -->
{% extends "base.html" %}
{% block body %}
<div class="col-lg-12">
    <!-- código omitido -->
    <div class="panel panel-default">
        {% with total_de_contatos=perfil_logado.contatos.count %}
        {% if total_de_contatos %}
            <div class="panel-heading">
                <strong>Você tem {{total_de_contatos}}
                contato{{ total_de_contatos|pluralize }}</strong>
            </div>
            <ul class="list-group">
                {% for contato in perfil_logado.contatos.all %}
                <a href="{% url 'exibir' contato.id %}"
                class="list-group-item">{{ contato.nome }} // {{ contato.email }}</a>
                {% endfor %}
            </ul>
        {% else %}
            <div class="panel-body">
                <p>Você não possui contatos no momento :(</p>
            </div>
        {% endif %}
        {% endwith %}
    </div>
</div>
{% endblock %}
```

32



## Painel de contatos



33

## Atualizando a funcionalidade exibir

- O link convidar ainda é exibido, mesmo o convite já tendo sido aceito;
- Podemos atualizar a funcionalidade de exibir para não exibir o link;
- Além disso, podemos informar que o perfil exibido já é um contato;
- Na view, podemos retornar uma variável sinalizando que um perfil já é contato:  
`perfil_logado = get_perfil_logado(request)`  
`ja_eh_contato = perfil in perfil_logado.contatos.all()`

34

## Atualizando a funcionalidade exibir

```
# connectedin/perfis/views.py

# código anterior omitido
def exibir(request, perfil_id):
    perfil = Perfil.objects.get(id=perfil_id)
    perfil_logado = get_perfil_logado(request)
    ja_eh_contato = perfil in perfil_logado.contatos.all()

    return render(request, 'perfil.html',
        {'perfil': perfil,
         'perfil_logado': get_perfil_logado(request),
         'ja_eh_contato': ja_eh_contato})

# código posterior omitido
```

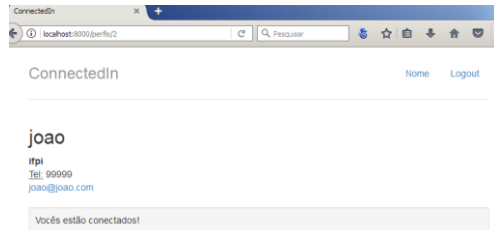
35

## Atualizando a funcionalidade exibir

```
<!-- connectedin/perfis/perfil.html -->
{% extends "base.html" %}
{% block body %}
<div class="row">
  <div class="col-lg-12">
    <h2 style="margin-top:0">{{perfil.nome}}</h2>
    <address>
      <strong>{{perfil.nome_empresa}}</strong><br/>
      <abbr title="Telefone">Tel:</abbr> {{perfil.telefone}}<br/>
      <a href="mailto:{{perfil.email}}">{{perfil.email}}</a>
    </address>
    {% if ja_eh_contato %}
      <div class="well well-sm">Você está conectado!</div>
    {% else %}
      <a href="{% url 'convidar' perfil.id %}"
        class="btn btn-success" role="button">convidar</a>
    {% endif %}
  </div>
</div>
{% endblock %}
```

36

## Atualizando a funcionalidade exibir



37

## Django

Arquivos estáticos, templates,  
exibindo e aceitando convites

Ely – [elydasilvimiranda \[at\] gmail.com](mailto:elydasilvimiranda@gmail.com)

38