

Universidade Federal de Goiás
Instituto de Informática
Prof. Ronaldo Martins da Costa





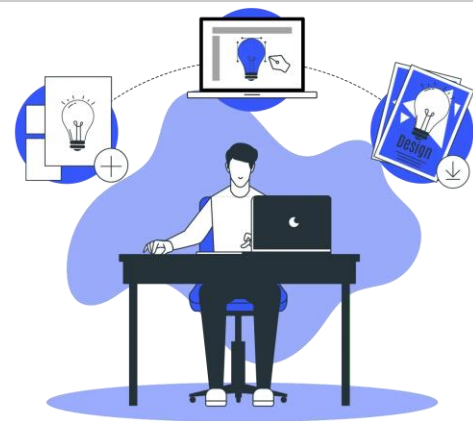
Criando suas APP's

- É possível construirmos nossas próprias aplicações no django
- Até o momento, além do APP de administração do django vimos apenas como apresentar algumas mensagens no navegador

```
def index(request):  
    return HttpResponse("EXEMPLO 01.")
```

```
def index(request):  
    return HttpResponse("AGORA EH O EXEMPLO 02.")
```

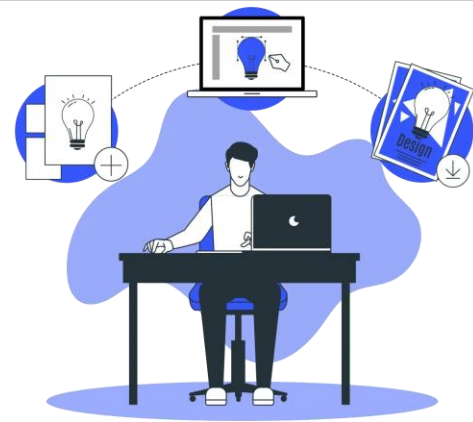
- No entanto, é possível construir saídas para o navegador bem mais sofisticadas. Para tal iremos explorar mais os recursos disponíveis nas views(`views.py`) e os templates em nosso projeto





Criando suas APP's

- Uma view é o lugar onde nós colocamos a "lógica" da nossa aplicação
- Ela vai extrair informações do model que você criou e entregá-las a um template
- Views são apenas funções Python





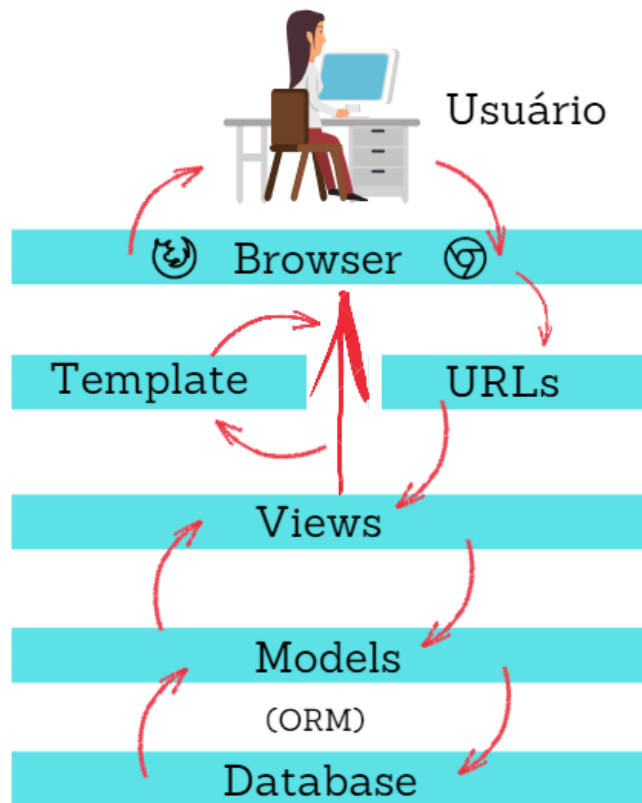
Renderizando páginas HTML



- No Django, os templates são páginas para visualização de dados
- As páginas HTML são chamadas de templates
- É possível configurar nossa APP para exibir páginas HTML e, como veremos futuramente, inserir em um arquivo HTML o que django chama de linguagem de template



O MTV do Django





Renderizando páginas HTML



- Para exibir/renderizar uma página html em nossa aplicação primeiramente devemos criar uma pasta com o nome **templates** onde serão armazenadas todas as páginas html, que o django chama de “Template”

```
bdpratico
  bdpratico
    exemplo01
      migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      admin.py
      urls.py
      views.py
    manage.py
```



Renderizando páginas HTML



- Dentro desta pasta templates, vamos criar um arquivo chamado **pagina0.html**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Página0.html</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>Minha primeira página renderizada no Django.</p>
    <p>bdpratico\exemplo01\templates\pagina0.html</p>
  </body>
</html>
```



Renderizando páginas HTML



- A estrutura de pastas do projeto ficará da seguinte forma

```
bdpratico
  bdpratico
    exemplo01
      migrations
      templates
        pagina0.html
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      admin.py
      urls.py
      views.py
    manage.py
```




Renderizando páginas HTML

- Agora vamos criar uma nova rota em nosso arquivo urls.py
- Desta forma quando for digitado
http://localhost:8000 ou
http://localhost:8000/exemplo01
será executada a view → index
- E quando for digitado
http://localhost:8000/exemplo01/pagina0
será executada a view → pagina0

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0'),
]
```





Renderizando páginas HTML



- Finalmente, altere o código contido em views.py

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("EXEMPLO 01.")

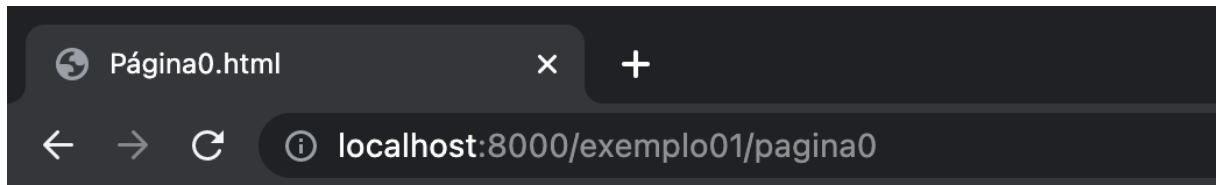
def pagina0(request):
    return render(request, 'pagina0.html')
```





Renderizando páginas HTML

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Hello World!

Minha primeira página renderizada no Django.

bdpratico\exemplo01\templates\pagina0.html



Inserindo imagens no Template

- A alteração apresentada seria suficiente para exibir uma imagem em nossa página html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Página0.html</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>Minha primeira página renderizada no Django.</p>
    <p>bdpratico\exemplo01\templates\pagina0.html</p>
    <p></p>
  </body>
</html>
```

Não no Django



``



Inserindo imagens no Template

- Agora vamos criar um arquivo html com o nome **pagina1.html** com os ajustes necessários

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Página0.html</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>Minha primeira página renderizada no Django.</p>
    <p>bdpratico\exemplo01\templates\pagina1.html</p>
    <p></p>
  </body>
</html>
```



``



Renderizando páginas HTML



- A estrutura de pastas do projeto ficará da seguinte forma

```
bdpratico
  bdpratico
  exemplo01
    migrations
    templates
      pagina0.html
      pagina1.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    admin.py
    urls.py
    views.py
  manage.py
```



Renderizando páginas HTML

- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0'),
    path('pagina1', views.pagina1, name='pagina1'),
]
```





Inserindo imagens no Template

- O django trata as imagens como “Arquivos estáticos” que devem todos ser armazenados em uma pasta
- O nome desta pasta é definido no arquivo settings.py, por default ela é chamada de “**static**”

```
STATIC_URL = 'static/'
```



```

```




Inserindo imagens no Template

- O projeto agora ficará com a seguinte estrutura

```
bdpratico
  bdpratico
  exemplo01
    migrations
    static
      pagina1.png
    templates
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    admin.py
    urls.py
    views.py
  manage.py
```



```

```



Inserindo imagens no Template

- Finalmente, vamos criar uma nova view para a nova rota criada em urls.py

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("EXEMPLO 01.")

def pagina0(request):
    return render(request, 'pagina0.html')

def pagina1(request):
    return render(request, 'pagina1.html')
```



``



Inserindo imagens no Template

- Alterações feitas nos arquivos e pastas estáticos do projeto necessitam que o servidor seja reiniciado

```
CONTROL + C  
python manage.py runserver
```



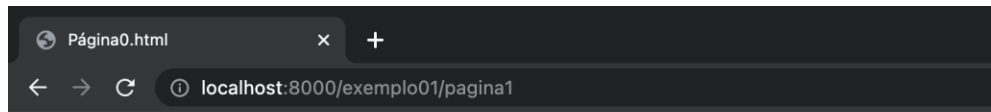
```

```



Inserindo imagens no Template

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Hello World!

Minha primeira página renderizada no Django.

bdpratico\exemplo01\templates\pagina1.html

**"Hello
World"**



Acessando o DB

- Neste ponto, vamos aprender mais algumas instruções que o django chama de linguagem de template e também alguns comandos para executar queries do *Object-relational mapping* (ORM) do django
- Iniciaremos com frases simples e futuramente veremos em mais detalhes o ORM e a linguagem de template do django





Inserindo imagens no Template

- Agora vamos criar um arquivo html com o nome **pagina2.html**



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Página2.html</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>bdpratico\exemplo01\templates\pagina2.html</p>
    {% for regs in pessoas %}
      <p>{{ regs.nome }} - {{ regs.email }} - {{ regs.celular }}</p>
    {% endfor %}
  </body>
</html>
```



Renderizando páginas HTML

- A estrutura de pastas do projeto ficará da seguinte forma

```
bdpratico
  bdpratico
  exemplo01
    migrations
    templates
      pagina0.html
      pagina1.html
      pagina2.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    admin.py
    urls.py
    views.py
  manage.py
```





Acessando o DB

- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0'),
    path('pagina1', views.pagina1, name='pagina1'),
    path('pagina2', views.pagina2, name='pagina2'),
]
```





Renderizando páginas HTML

- Finalmente, altere o código contido em views.py



```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("EXEMPLO 01.")

def pagina0(request):
    return render(request, 'pagina0.html')

def pagina1(request):
    return render(request, 'pagina1.html')

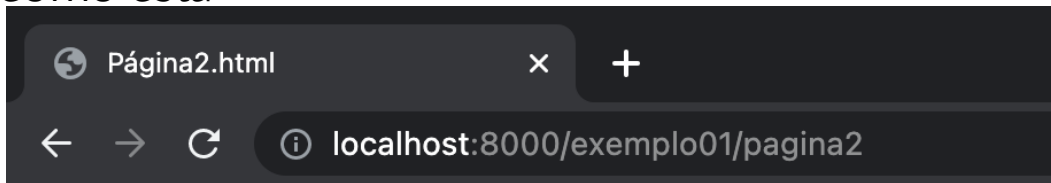
def pagina2(request):
    from .models import pessoa
    dicionario = {}
    registros = pessoa.objects.all()
    dicionario['pessoas'] = registros
    return render(request, 'pagina2.html', dicionario)
```





Renderizando páginas HTML

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Hello World!

bdpratico\exemplo01\templates\pagina2.html

Fulano de Tal - fulanotal@gmail.com - (98) 76543-2109

Joao da Silva - joaosilva@gmail.com - (12) 34567-8901

Joao da Silva - joaosilva2@gmail.com - (98) 76543-2109

Ronaldo Martins da Costa - ronaldocosta@ufg.br - (62) 98271-6303



Melhorando o Visual da APP

- Bootstrap é um framework web com código-fonte aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web, usando HTML, CSS e JavaScript, baseado em modelos de design para a tipografia, melhorando a experiência do usuário em um site amigável e responsivo





Melhorando o Visual da APP

- A instalação do Bootstrap é bastante simples com o instalador de pacotes pip

```
pip install django-bootstrap-v5
```

- O próximo passo é colocar o bootstrap na lista de APP's do projeto

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'exemplo01',  
    'exemplo02',  
    'bootstrap5',
```

```
]
```





Melhorando o Visual da APP



- Agora vamos criar um arquivo html com o nome **pagina3.html**

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Página3.html</title>
</head>
<body>
  
  <p class="h2">Colocando Estilos na Página</p>
  <p>bdpratico\exemplo01\templates\pagina3.html</p>
  <div class="container">
    <div class="row" style="line-height: 1;">
      <div class="col-md-12 col-sm-12 col-xs-12">
```

Carregando o Bootstrap no HTML

Tags do Bootstrap

Tags do Bootstrap

Continua...



Melhorando o Visual da APP

- Agora vamos criar um arquivo html com o nome **pagina3.html**



Continuação...

```
<table class="table table-striped table-hover">
  <th>Nome</th>
  <th>eMail</th>
  <th>Celular</th>
  {% for regs in pessoas %}
  <tr>
    <td>{{ regs.nome }}</td>
    <td>{{ regs.email }}</td>
    <td>{{ regs.celular }}</td>
  </tr>
  {% endfor %}
</table>
</div>
</div>
</div>
</body>
</html>
```

Tags do Bootstrap



Melhorando o Visual da APP

- A estrutura de pastas do projeto ficará da seguinte forma

```
bdpratico
  bdpratico
  exemplo01
    migrations
    static
      pagina1.png
      pagina3.png
    templates
      pagina0.html
      pagina1.html
      pagina2.html
      pagina3.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    admin.py
    urls.py
    views.py
  manage.py
```





Melhorando o Visual da APP

- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0'),
    path('pagina1', views.pagina1, name='pagina1'),
    path('pagina2', views.pagina2, name='pagina2'),
    path('pagina3', views.pagina3, name='pagina3'),
]
```





Melhorando o Visual da APP

- Finalmente, altere o código contido em views.py

```
from django.shortcuts import render
from django.http import HttpResponse

...

...

...

def pagina3(request):
    from .models import pessoa
    dicionario = {}
    registros = pessoa.objects.all()
    dicionario['pessoas'] = registros
    return render(request, 'pagina3.html', dicionario)
```





Melhorando o Visual da APP

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



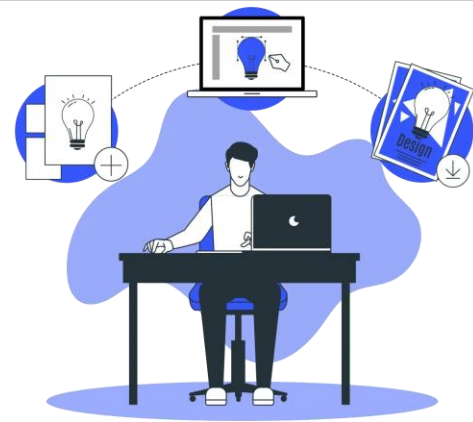
The screenshot shows a web browser window with the address bar displaying 'localhost:8000/exemplo01/pagina3'. The page title is 'Colocando Estilos na Página'. Below the title, the file path 'bdpratico\exemplo01\templates\pagina3.html' is visible. The main content is a table with three columns: 'Nome', 'eMail', and 'Celular'. The table contains four rows of data.

Nome	eMail	Celular
Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109
Joao da Silva	joaosilva@gmail.com	(12) 34567-8901
Joao da Silva	joaosilva2@gmail.com	(98) 76543-2109
Ronaldo Martins da Costa	ronaldocosta@ufg.br	(62) 98271-6303



Class Based Views

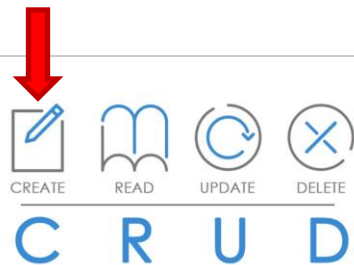
- O que vimos até o momento foi como receber requests e responder respostas com páginas html contendo diversas informações. Isso é bem legal para começar a entender o processo que o Django faz: recebe requests e devolve templates
- As **Class Based Views** agrega as funções básicas das views dentro de classes como métodos. Classes que já estão “pré-prontas” e que a sua classe pode herdar
- A partir daí as alterações que precisam ser feitas são mínimas





Class Based Views - Create

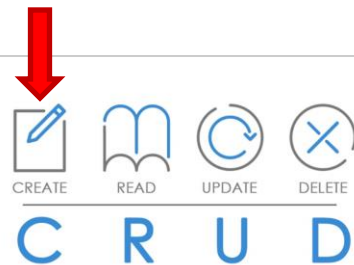
- O Django possui uma classe chamada **CreateView** que facilita a criação da funcionalidade de **incluir** registro em uma tabela com poucas linhas de programação.





Class Based Views - Create

- Agora vamos criar um arquivo html com o nome
bdpratico\exemplo01\templates\exemplo01\pessoa_form.html



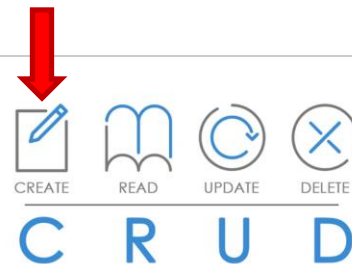
```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!doctype html>
<html>
<head>
    <title>Form de Pessoas</title>
</head>
<body>
    <div class="container">
        <div class="col-md-12 col-sm-12 col-xs-12">
            <center>
                <p>bdpratico\exemplo01\templates\exemplo01\pessoa_form.html</p>
                

                <form method="POST" enctype="multipart/form-data">
                    {% csrf_token %}
                    <div class="col-md-12 col-sm-12 col-xs-12">
                        <div class="row" style="line-height: 1;">
                            <div class="col-md-1 col-sm-1 col-xs-1"></div>
                            <div class="col-md-8 col-sm-12 col-xs-12">
                                {% bootstrap_field form.nome placeholder=' ' %}
```



Class Based Views - Create

- Agora vamos criar um arquivo html com o nome `bdpratico\exemplo01\templates\exemplo01\pessoa_form.html`



```
</div>
</div>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-8 col-sm-12 col-xs-12">
    {% bootstrap_field form.email placeholder='' style='height:15px;' %}
  </div>
</div>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-4 col-sm-4 col-xs-4">
    {% bootstrap_field form.celular placeholder='' %}
  </div>
  <div class="col-md-4 col-sm-4 col-xs-4">
    {% bootstrap_field form.funcao placeholder='' %}
  </div>
</div>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-4 col-sm-4 col-xs-4">
    {% bootstrap_field form.nascimento placeholder='' %}
  </div>
  <div class="col-md-1 col-sm-1 col-xs-1"><br><br>
    {% bootstrap_field form.ativo placeholder='' %}
  </div>
</div>
```



Class Based Views - Create

- Agora vamos criar um arquivo html com o nome `bdpratico\exemplo01\templates\exemplo01\pessoa_form.html`



```
<div class="col-md-1 col-sm-1 col-xs-1"><br><br>
</div>
<div class="col-md-1 col-sm-1 col-xs-1"><br>
    <button type="submit" class="btn btn-primary">Salvar</button>
</div>
</div>
</div>
</form>
</center>
</div>
</div>
</body>
</html>
```



Class Based Views - Create

- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias')
]
```





Class Based Views - Create



CREATE

C



READ

R



UPDATE

U



DELETE

D

- Finalmente, altere o arquivo views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from django.views.generic import ListView
from .models import pessoa
...

from django.urls import reverse_lazy
from django.views.generic.edit import CreateView
class pessoa_create(CreateView):
    from .models import pessoa
    model = pessoa
    fields = ['nome', 'email', 'celular', 'funcao', 'nascimento', 'ativo']
    def get_success_url(self):
        return reverse_lazy('pagina3_alias')
```



Class Based Views - Create


- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Form de Pessoas

localhost:8000/emplo01/pessoa_create/

bdpratico\emplo01\templates\emplo01\pessoa_form.html



Nome

EEmail

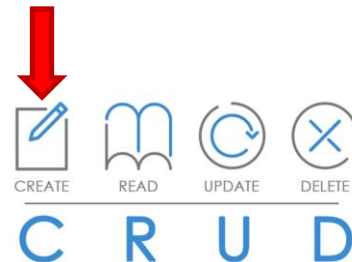
Celular

Funcao

Nascimento

☒ Ativo

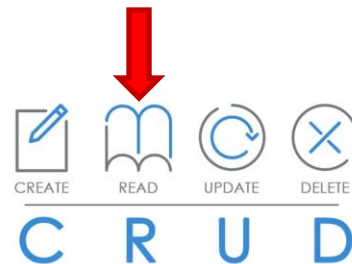
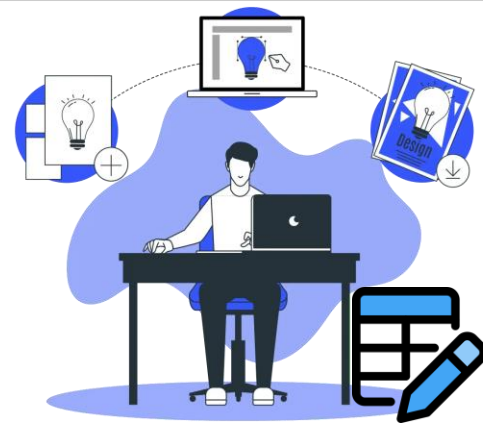
Salvar





Tabelas Dinâmicas

- Outra funcionalidade interessante para utilizarmos em nossos projetos são as “tabelas dinâmicas”
- Estas são “visualizações” dinâmicas dos registros de um DB de maneira bastante ágil e eficiente





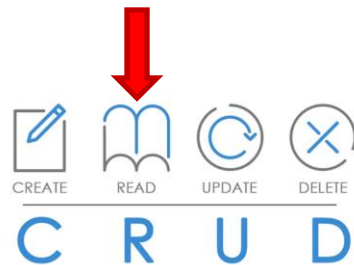
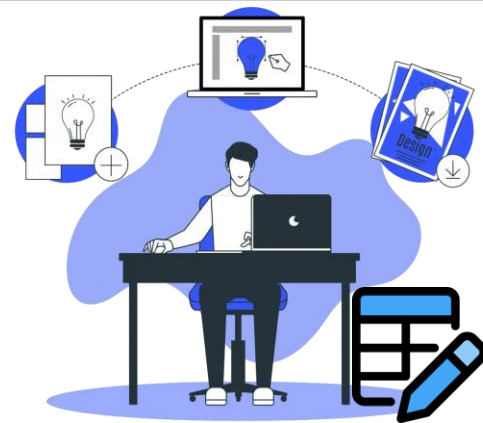
Tabelas Dinâmicas

- A instalação é feita utilizando o pip

```
pip install django-tables2
```

- Depois de instalado, assim como as demais APP's, esta também deve ser inserida no INSTALLED_APPS

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'exemplo01',  
    'exemplo02',  
    'bootstrap5',  
    'django_tables2',  
]
```

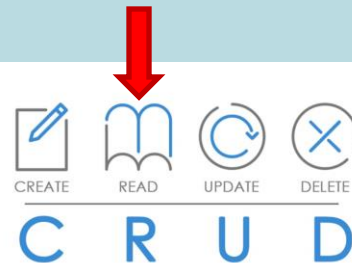
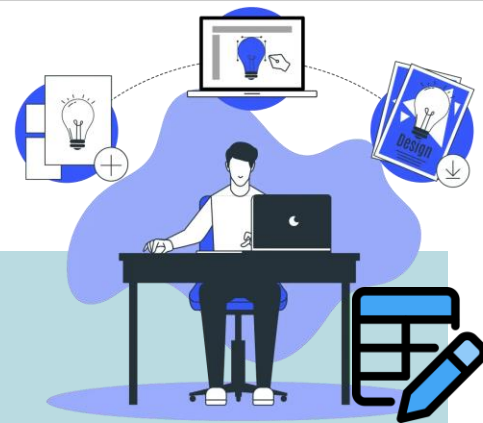




Tabelas Dinâmicas

- Agora vamos criar um arquivo html com o nome `bdpratico\exemplo01\templates\exemplo01\pessoa_list.html`

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
{% load render_table from django_tables2 %}
<!doctype html>
<html>
<head>
  <title>Lista de Pessoas</title>
</head>
<body>
  <div class="container">
    <div class="col-md-12 col-sm-12 col-xs-12">
      <center>
        <p>bdpratico\exemplo01\templates\exemplo01\pessoa_list.html</p>
        
        {% render_table object_list %}
      </center>
    </div>
  </div>
</body>
</html>
```





Tabelas Dinâmicas

- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
]
```





Tabelas Dinâmicas

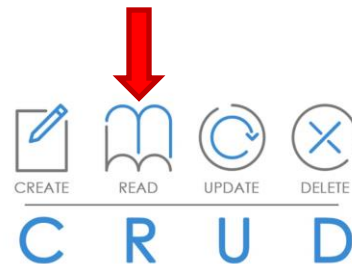
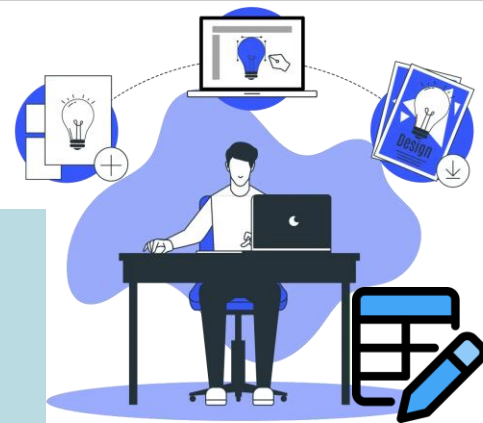
- Finalmente, altere o código contido em views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from .models import pessoa

...

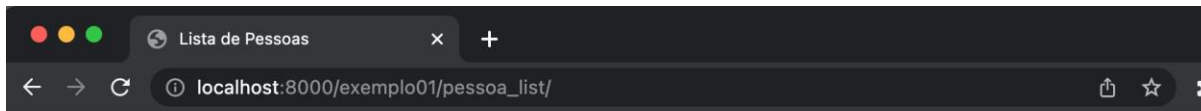
from django.views.generic import ListView
class pessoa_list(ListView):
    from .models import pessoa
    model = pessoa
```



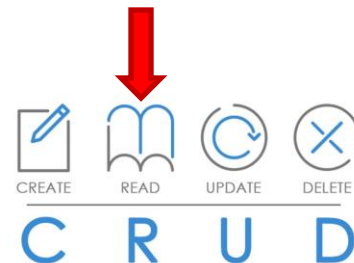


Tabelas Dinâmicas

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



<u>ID</u>	<u>Nome</u>	<u>EMail</u>	<u>Celular</u>	<u>Funcao</u>	<u>Nascimento</u>	<u>Ativo</u>
2	Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Aluno	12/30/1998	✓
1	Joao da Silva	joaosilva@gmail.com	(12) 34567-8901	Diretor	05/06/2002	✓
4	Joao da Silva	joaosilva2@gmail.com	(98) 76543-2109	Duplicidade	07/09/1975	✗
3	Ronaldo Martins da Costa	ronaldocosta@ufg.br	(62) 98271-6303	Professor	08/27/1971	✓





Tabelas Dinâmicas - Filtros

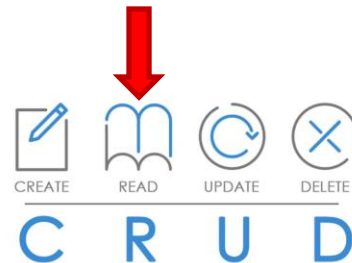
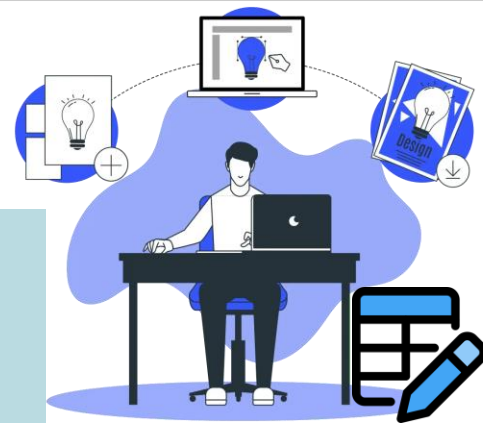
- Finalmente, altere o código contido em views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from django.views.generic import ListView
from .models import pessoa

...

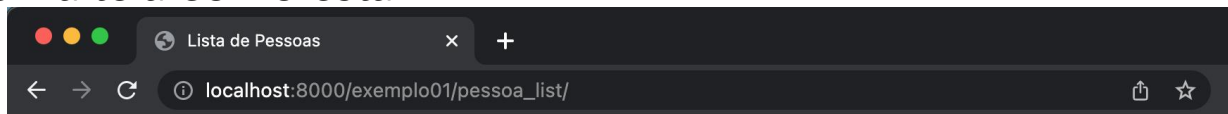
class pessoa_list(ListView):
    from .models import pessoa
    model = pessoa
    queryset = pessoa.objects.filter(ativo=True)
```





Tabelas Dinâmicas - Filtros

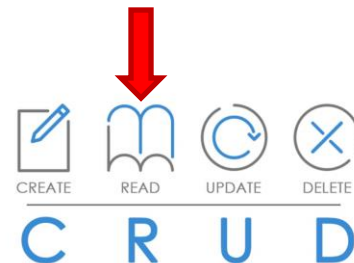
- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



bdpratico\exemplo01\templates\exemplo01\pessoa_list.html



<u>ID</u>	<u>Nome</u>	<u>EEmail</u>	<u>Celular</u>	<u>Funcao</u>	<u>Nascimento</u>	<u>Ativo</u>
2	Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Aluno	12/30/1998	✓
1	Joao da Silva	joaosilva@gmail.com	(12) 34567-8901	Diretor	05/06/2002	✓
3	Ronaldo Martins da Costa	ronaldocosta@ufg.br	(62) 98271-6303	Professor	08/27/1971	✓





Class Based Views - Create

- O Django possui uma classe chamada **UpdateView** que facilita a criação da funcionalidade de **atualizar** registro em uma tabela com poucas linhas de programação.





Class Based Views - Update

- ☉ Vamos utilizar o mesmo template html já criado
`bdpratico\exemplo01\templates\exemplo01\pessoa_form.html`



```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!doctype html>
<html>
<head>
    <title>Form de Pessoas</title>
</head>
<body>
    <div class="container">
        <div class="col-md-12 col-sm-12 col-xs-12">
            <center>
                <p>bdpratico\exemplo01\templates\exemplo01\pessoa_form.html</p>
                

                <form method="POST" enctype="multipart/form-data">
                    {% csrf_token %}
                    <div class="col-md-12 col-sm-12 col-xs-12">
                        <div class="row" style="line-height: 1;">
                            <div class="col-md-1 col-sm-1 col-xs-1"></div>
                            <div class="col-md-8 col-sm-12 col-xs-12">
                                {% bootstrap_field form.nome placeholder=' ' %}
```



Class Based Views - Update

- Vamos criar uma nova rota em nosso arquivo urls.py



```
from django.contrib import admin
from django.urls import path
from . import views
from .views import pessoa_list

urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(),
    name='pessoa_update_alias'),
]
```



Class Based Views - Update

- Finalmente, altere o arquivo views.py



```
from django.shortcuts import render
from django.http import HttpResponse

from django.views.generic import ListView
from .models import pessoa
...

from django.views.generic.edit import UpdateView
class pessoa_update(UpdateView):
    from .models import pessoa
    model = pessoa
    fields = ['nome', 'email', 'celular', 'funcao', 'nascimento', 'ativo']
    def get_success_url(self):
        return reverse_lazy('pessoa_list_alias')
```



Class Based Views - Update

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



bdpratico\exemplo01\templates\exemplo01\pessoa_form.html



Nome

Joao da Silva

EEmail

joaosilva@gmail.com

Celular

(12) 34567-8901

Funcao

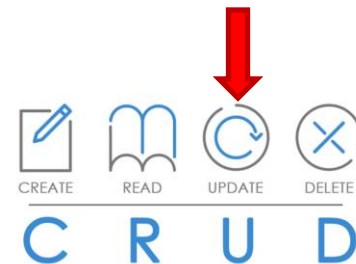
Diretor

Nascimento

2002-05-06

☒ Ativo

Salvar





Class Based Views - Delete

- O Django possui uma classe chamada **DeleteView** que facilita a criação da funcionalidade de **excluir** registro em uma tabela com poucas linhas de programação.





Class Based Views - Delete

- Agora vamos criar um arquivo html com o nome
bdpratico\exemplo01\templates\exemplo01\pessoa_delete.html



```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!doctype html>
<html>
<head>
    <title>Excluir Pessoas</title>
</head>
<body>
    <div class="container">
        <div class="col-md-12 col-sm-12 col-xs-12">
            <center>
                <p>bdpratico\exemplo01\templates\exemplo01\pessoa_delete.html</p>
                
                <form method="POST">
                    {% csrf_token %}
                    <p>{{object.name}}</p>
                    <button type="submit" class="btn btn-danger">Sim, exclua!</button> &nbsp;
                </form>
            </center>
        </div>
    </div>
</body>
</html>
```



Class Based Views - Delete

- Vamos criar uma nova rota em nosso arquivo urls.py



```
from django.contrib import admin
from django.urls import path
from . import views
from .views import pessoa_list

urlpatterns = [
    path('', views.index, name='index'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(),
    name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(),
name='pessoa_delete_alias'),
]
```



Class Based Views - Delete

- Finalmente, altere o arquivo views.py



```
from django.shortcuts import render
from django.http import HttpResponse

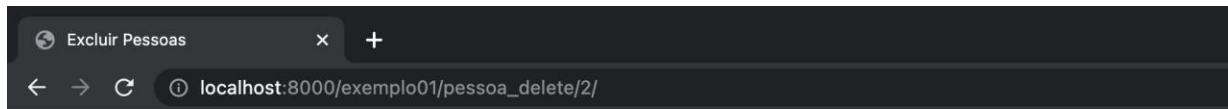
from django.views.generic import ListView
from .models import pessoa
...

from django.views.generic.edit import DeleteView
class pessoa_delete(DeleteView):
    from .models import pessoa
    model = pessoa
    fields = ['nome', 'email', 'celular', 'funcao', 'nascimento', 'ativo']
    template_name_suffix = '_delete'
    def get_success_url(self):
        return reverse_lazy('pessoa_list_alias')
```



Class Based Views - Delete

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



bdpratico\exemplo01\templates\exemplo01\pessoa_delete.html



Fulano de Tall

Sim, exclua!





Enviando Dados - POST

- Para que uma aplicação se comunique com um servidor de serviços é necessário enviar os dados para processamento
- O servidor processa os dados e envia uma resposta ao usuário
- Isso parece simples, mas é importante manter algumas coisas em mente para garantir que os dados não danifiquem o servidor ou causem problemas para seus usuários
- Veremos como realizar esta tarefa no Django e construiremos uma página de login com senha, para tal veremos alguns detalhes da classe `django.contrib.auth`



Enviando Dados - POST

- Vamos construir um template para entrada de dados
`bdpratico\exemplo01\templates\pagina4.html`

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!doctype html>
<html>

<head>
  <title>Form POTS</title>
</head>

<body>
  <div class="container">
    <div class="col-md-12 col-sm-12 col-xs-12">
      <center>
        <p>bdpratico\exemplo01\templates\pagina4.html</p>
        
        <br>
        <form method="POST" enctype="multipart/form-data">
          {% csrf_token %}
          <div class="col-md-12 col-sm-12 col-xs-12">
            <br>
```



Enviando Dados - POST

submit

- Vamos construir um template para entrada de dados
bdpratico\exemplo01\templates\pagina4.html

```
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-8 col-sm-12 col-xs-12">
    <label for="nome">Nome</label>
    <input type="text" name="nome" id="nome" placeholder="Digite o nome da pessoa">
  </div>
</div>
<br>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-8 col-sm-12 col-xs-12">
    <label for="email">eMail</label>
    <input type="text" name="email" id="email" placeholder="Digite o eMail da pessoa">
  </div>
</div>
<br>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-4 col-sm-4 col-xs-4">
    <label for="celular">Celular</label>
    <input type="text" name="celular" id="celular" placeholder="Digite o celular da pessoa">
  </div>
</div>
```



Enviando Dados - POST

submit

- Vamos construir um template para entrada de dados
bdpratico\exemplo01\templates\pagina4.html

```
<div class="col-md-4 col-sm-4 col-xs-4">
  <label for="funcao">Função</label>
  <input type="text" name="funcao" id="funcao" placeholder="Digite a função da pessoa">
</div>
</div>
<br>
<div class="row" style="line-height: 1;">
  <div class="col-md-1 col-sm-1 col-xs-1"></div>
  <div class="col-md-4 col-sm-4 col-xs-4">
    <label for="nascimento">Nascimento</label>
    <input type="date" name="nascimento" id="nascimento">

  </div>
  <div class="col-md-1 col-sm-1 col-xs-1"><br><br>
    <label for="ativo">Ativo</label>
    <input type="checkbox" name="ativo" id="ativo">
  </div>
  <div class="col-md-1 col-sm-1 col-xs-1"><br><br>
</div>
  <div class="col-md-1 col-sm-1 col-xs-1"><br>
    <button type="submit" class="btn btn-primary">Salvar</button>
  </div>
</div>
```




Enviando Dados - POST

- Vamos construir um template para entrada de dados
bdpratico\exemplo01\templates\pagina4.html

```
        </div>
      </form>
    </center>
  </div>
</div>
</body>
</html>
```



Enviando Dados - POST



- Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from .views import pessoa_list

urlpatterns = [
    path('', views.index, name='index_alias'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path('pagina4', views.pagina4, name='pagina4_alias'),
    path('menu', views.pessoa_menu, name='menu_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(),
    name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(),
    name='pessoa_delete_alias'),
]
```



Enviando Dados - POST

submit

- Finalmente, altere o arquivo views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from django.views.generic import ListView
from .models import pessoa
...
...

def pagina4(request):
    nome = request.POST.get('nome')
    email = request.POST.get('email')
    celular = request.POST.get('celular')
    funcao = request.POST.get('funcao')
    nascimento = request.POST.get('nascimento')
    ativo = request.POST.get('ativo')
    print("Nome:", nome)
    print("eMail:", email)
    print("Celular:", celular)
    print("Funcao:", funcao)
    print("Nascimento:", nascimento)
    print("ativo:", ativo)
    return render(request, 'pagina4.html')
```



Enviando Dados - POST

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta ao preencher o form e clicar no botão salvar



```
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro bdpratico % python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 26, 2023 - 11:33:45
Django version 4.1.5, using settings 'bdpratico.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
Nome: None
eMail: None
Celular: None
Funcao: None
Nascimento: None
ativo: None
[26/Jan/2023 11:34:00] "GET /exemplo01/pagina4 HTTP/1.1" 200 3955
Nome: aaaaa
eMail: bbbbbb
Celular: ccccccc
Funcao: dddddddd
Nascimento: 2023-01-01
ativo: on
[26/Jan/2023 11:34:32] "POST /exemplo01/pagina4 HTTP/1.1" 200 3955
```

submit



Autenticação



- Agora vamos construir uma tela de autenticação
- Veremos algumas funcionalidade da classe `django.contrib.auth` e o conceito de session
- A sessão é um conjunto de interações que ocorrem no seu site em um determinado período
- Uma única sessão pode ter várias exibições de página ou de tela, eventos, interações sociais e transações de comércio eletrônico, ela é o “depósito” das ações realizadas por um usuário no seu site



Autenticação



- Vamos construir um template para os dados de autenticação password `bdpratico\exemplo01\index.html`

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!doctype html>
<html>
<head>
    <title>Pessoas</title>
</head>
<body>
    <center>
        <br><br><br><br>
        
        <br><br>
    </center>
    <form method="POST" action="{% url 'index_alias' %}">
        {% csrf_token %}
```



Autenticação



- Vamos construir um template para os dados de autenticação password **bdpratico\exemplo01\index.html**

```
<center>
  <p style="width:300px;">Bem Vindo ao Site de Pessoas</p>
  <div style="border-style: ridge; border-radius: 10px; width: 300px;">
    <br>
    <p><input type="text" name="username" class="border-top-0" placeholder="User Name" style="width:250px;">
    </p>
    <p></p><input type="password" name="password" class="border-top-0" placeholder="Password"
      style="width:250px;">
    </p>
    <p></p><input type="submit" value="Acessar" class="btn btn-info"></p>
  </div>
</center>
</form>
</body>
<br><br>
```



Autenticação



- ⦿ Vamos utilizar a mesma rota para o index já existente em urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from .views import pessoa_list

urlpatterns = [
    path('', views.index, name='index_alias'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path('pagina4', views.pagina4, name='pagina4_alias'),
    path('menu', views.pessoa_menu, name='menu_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(),
    name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(),
    name='pessoa_delete_alias'),
]
```




Autenticação



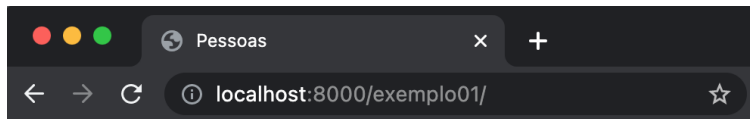
- Finalmente, altere o arquivo a view index anterior pela seguinte:

```
from django.shortcuts import render
from django.http import HttpResponse
from django.views.generic import ListView
from .models import pessoa
...
...
from django.contrib.auth import authenticate, login, logout
def index(request):
    print("else")
    usuario = request.POST.get('username')
    senha = request.POST.get('password')
    user = authenticate(username=usuario, password=senha)
    if (user is not None):
        login(request, user)
        request.session['username'] = usuario
        request.session['password'] = senha
        request.session['usernamefull'] = user.get_full_name()
        print(request.session['username'])
        print(request.session['password'])
        print(request.session['usernamefull'])
        from django.shortcuts import redirect
        return redirect('menu_alias')
    else:
        return render(request, 'index.html')
```



Autenticação

- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta



Bem Vindo ao Site de Pessoas

Acessar





Um Menu para CRUD

- Vamos construir um template para o menu
`bdpratico\exemplo01\templates\exemplo01\pessoa_menu.html`



```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
{% load render_table from django_tables2 %}
<!doctype html>
<html>
<head>
  <title>Menu de Pessoas</title>
</head>
<body>
  <div class="container">
    <div class="col-md-12 col-sm-12 col-xs-12">
      <center>
        <p>bdpratico\exemplo01\templates\exemplo01\pessoa_menu.html</p>
        
        <br><br>
        {% render_table table %}
        <br>
        <a href="{% url 'pessoa_create_alias' %}">
          <input type="button" class="btn btn-primary" value="Incluir novas pessoas">
        </a>
      </center>
    </div>
  </div>
</body>
</html>
```



Um Menu para CRUD



● Verificando a rota para o menu crud em urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from .views import pessoa_list

urlpatterns = [
    path('', views.index, name='index_alias'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path('pagina4', views.pagina4, name='pagina4_alias'),
    path('menu', views.pessoa_menu, name='menu_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(),
    name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(),
    name='pessoa_delete_alias'),
]
```



Um Menu para CRUD

- Criando a view para o menu do crud de pessoas no views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from django.views.generic import ListView
from .models import pessoa
...
...
from django_tables2 import SingleTableView
class pessoa_menu(SingleTableView):
    from .models import pessoa
    from .tables import pessoa_table
    model = pessoa
    table_class = pessoa_table
    template_name_suffix = '_menu'
    table_pagination = {"per_page": 5}
    template_name = 'exemplo01/pessoa_menu.html'
```





Um Menu para CRUD



- É necessário criar um arquivo chamado tables.py que customizará a tabela e funcionará como ponto de chamada para as funcionalidade Update e Delete

```
import django_tables2 as tables
from django_tables2.utils import A
from django.utils.html import format_html
from .models import pessoa
class pessoa_table(tables.Table):
    nome = tables.LinkColumn("pessoa_update_alias", args=[A("pk")])
    email = tables.LinkColumn("pessoa_update_alias", args=[A("pk")])
    celular = tables.LinkColumn("pessoa_update_alias", args=[A("pk")])
    funcao = tables.LinkColumn("pessoa_update_alias", args=[A("pk")])
    nascimento = tables.LinkColumn("pessoa_update_alias", args=[A("pk")])
    ativo = tables.Column()
    id = tables.LinkColumn("pessoa_delete_alias", args=[A("pk")],
        verbose_name="Excluir")
    class Meta:
        model = pessoa
        attrs = {"class": "table thead-light table-striped table-hover"}
        template_name = "django_tables2/bootstrap4.html"
        fields = ('nome', 'email', 'celular', 'funcao', 'nascimento', 'ativo')
```

Localização do arquivo tables.py
\\bdpratico\\exemplo01\\tables.py



Um Menu para CRUD

- Cada função do CRUD deve “retornar” para o Menu, sendo assim altere as funções `get_success_url` para:



```
def get_success_url(self):  
    return reverse_lazy('pessoa_menu_alias')
```



Um Menu para CRUD


- Se os ajustes no projeto foram realizados corretamente, você verá uma tela como esta:



Menu de Pessoas x +

localhost:8000/exemplo01/pessoa_menu

bdpratico\exemplo01\templates\exemplo01\pessoa_menu.html



Nome	EEmail	Celular	Funcao	Nascimento	Ativo	Excluir
Fulano de Tal	fulanotal@gmail.com	(98) 76543-2109	Aluno	1998-12-30	True	2
Joao da Silva	joaosilva@gmail.com	(12) 34567-8901	Diretor	2002-05-06	True	1
Joao da Silva	joaosilva2@gmail.com	(98) 76543-2109	Duplicidade	1975-07-09	False	4
Nova Pessoa	novapessoa@gmail.com	2238729387	Nov Função	2023-01-01	True	6
Outra Nova	outranova@gmail.com	7373463748	78643876	2023-01-23	True	7

1 2 next »

Incluir novas pessoas

Universidade Federal de Goiás
Instituto de Informática
Prof. Ronaldo Martins da Costa

