

django



escola piloto de computação
universidade federal de são carlos



LEONARDO



LUCIANA



RAFAEL



THEODÓSIO



CAIO



Nós somos a EPiC!

Escola Piloto de Computação
Universidade Federal de São Carlos



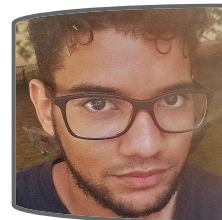
EDUARDO



JEAN



ALISSON



JHONATTAN



JACKSON



Pré-requisitos e ferramentas

- 1. Conhecimento básico na linguagem Python
- 2. Conhecimento básico em HTML
- 3. Conhecimento básico em padrões de projeto MVC
- 4. Python 3.6+
- 5. *IDE PyCharm ou algum editor de texto (sublime, atom...)*
- 6. *Terminal ou CMD*



Overview

- 1. O que é MVC?
- 2. O que é Desenvolvimento Web ?
- 3. *Ambientes Virtuais*
- 4. Estrutura de projetos Django
- 5. Executando
- 6. Rotas e URLs
- 7. Views
- 8. Templates
- 9. Models
- 10. Template Tags



Bônus

- 1. Herança de HTMLs
- 2. URLs dinâmicas
- 3. *Arquivos estáticos (img, css, js)*

1

O que é MVC?

MVC e outros padrões de projeto





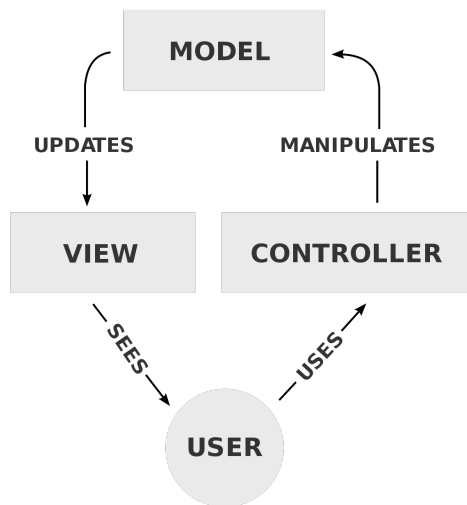
Padrões de projeto

- Organização de projetos maiores, com muitos arquivos
- Organizar de acordo com o papel daquele arquivo
- MVC, MVP, MVVM, MVW...
- Qual é o “melhor”?
- Django implementa (e cobra) corretamente o MVC



Padrão MVC

Model



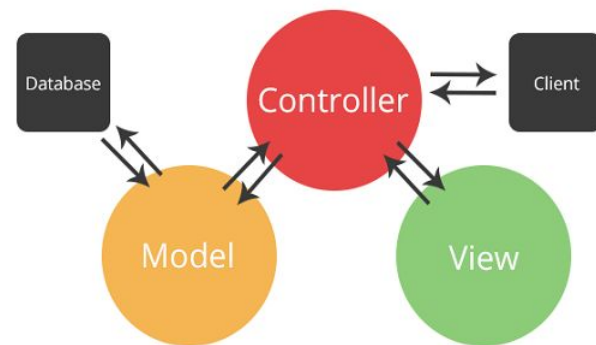
- Modelos de dados
- Banco de dados

View

- Parte visual
- Interfaces para o usuário

Controller

- Parte lógica
- Processamento
- Comunicação com BD



2

O que é desenvolvimento Web?

Back-end, Front-end, Frameworks, e outros termos esquisitos





Desenvolvimento Web

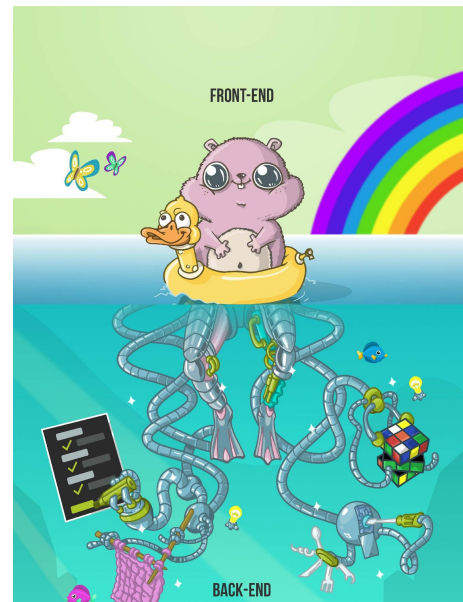
Front-end

- ❑ Parte da “frente”
- ❑ Parte visual
- ❑ UI/UX



Back-end

- ❑ Parte de “trás”
- ❑ Parte lógica
- ❑ UI/UX



3

Ambientes virtuais

Por que não no python?





Ambientes virtuais

- Temporário
- Interessante apenas ao projeto
- Compartilhável



Preparando o local

Cria a pasta do projeto
Entra na pasta do projeto

```
$ mkdir projeto_django  
$ cd projeto_django
```

Cria um ambiente virtual (windows)

```
$ python -m venv myenv
```

(Windows | Linux?)

Cria um ambiente virtual (linux)

```
$ pip install virtualenv  
$ virtualenv -p python3 myenv
```

(Linux)

Ativa o ambiente virtual

```
$ myenv\Scripts\activate  
$ source myenv/bin/activate
```

(Windows)
(Linux)

Instala o django
Verifica o que foi instalado
Lista tudo da pasta atual

```
(myenv) $ pip install django  
(myenv) $ pip freeze  
(myenv) $ dir
```

Imprime tudo que foi instalado e
já salva em um arquivo

Por fim:

```
(myenv) $ pip freeze > requirements.txt
```

4

Estruturas de projetos Django

“Bem começado, metade feito”
- Aristóteles





Preparando o projeto

Cria um projeto Django

Lista tudo da pasta atual

Entra na pasta do projeto

```
$ django-admin startproject projeto
```

```
$ dir
```

```
$ cd projeto
```



5

Executando

“Finalmente” - Vocês





Executando o projeto

ANTES:

```
$ django-admin startproject projeto  
$ dir  
$ cd projeto
```

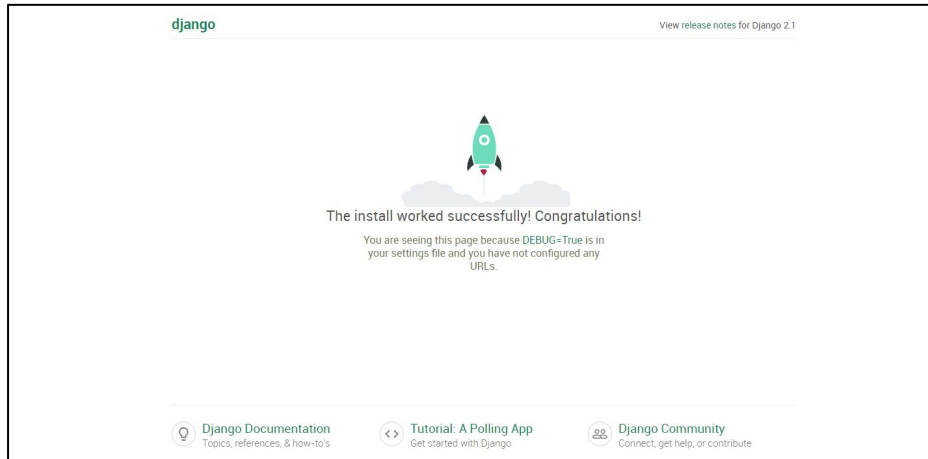
AGORA:

“Cria” o banco

```
$ python manage.py migrate
```

Executa no servidor

```
$ python manage.py runserver <porta>
```





Criando uma aplicação

- Aplicações são módulos do projeto
- Dividir o contexto de cada parte do projeto

\$ python manage.py startapp sistema Cria uma nova aplicação

```
@@ -37,6 +37,7 @@ INSTALLED_APPS = [  
    37     37     'django.contrib.sessions',  
    38     38     'django.contrib.messages',  
    39     39     'django.contrib.staticfiles',  
    +40     'sistema',  
    40     41 ]  
    41     42  
    42     43 MIDDLEWARE = [  

```

Adicionar a aplicação em INSTALLED_APPS
no arquivo “projeto/projeto/**settings.py**”



6

Rotas e URLs

Criação de novas páginas





Rotas e URLs

- URL = “Rota do navegador”
- Rota = “URL do projeto”
- Definir novas páginas
- Definir links entre páginas
- Organizar o projeto de acordo com suas funcionalidades
- Separar as URLs dentro do contexto correto



Criando uma nova Rota



Criar arquivo “projeto/sistema/urls.py” dentro da nova aplicação

```
@@ -0,0 +1,3 @@
+1  urlpatterns = [
+2
+3  ]
```



Direcionar as rotas para o arquivo urls.py correto

```
@@ -14,8 +14,9 @@ Including another URLconf
14      14      2. Add a URL to urlpatterns: path('b
15      15      """
16      16      from django.contrib import admin
-17      from django.urls import path
+17      from django.urls import path, include
18      18
19      19      urlpatterns = [
20      20          path('admin/', admin.site.urls),
+21          path('', include('sistema.urls')),
21      22      ]
```



Criando uma nova Rota

Etapas da criação de uma nova tela:



Definir a nova rota no arquivo “projeto/sistema/**urls.py**”



Definir a nova view no arquivo “projeto/sistema/**views.py**”



Criar o novo template na pasta templates

```
@@ -1,3 +1,6 @@
-1      urlpatterns = [
+1      from django.urls import path
+2      from . import views
2      3
+4      urlpatterns = [
+5          path('', views.home, name="home"),
3      6      ]
```



Criando uma nova Rota

Django e a confusão com MVC:

MVC	Função	Arquivo correspondente
Model	Representação dos dados	models.py
View	Parte visual	nome_da_tela.html
Controller	Parte lógica	views.py

- “models.py” possui as classes (tabelas) do banco
- “views.py” possui as funções e lógica da aplicação
- A pasta “templates” possui arquivos html, um para cada página, cada arquivo html desempenha o papel de view

7

Views

Lógica da aplicação






Criando uma nova view

Etapas da criação de uma nova tela:



Definir a nova rota no arquivo “projeto/sistema/**urls.py**”



Definir a nova view no arquivo “projeto/sistema/**views.py**” 



Criar o novo template na pasta templates

```
@@ -1,3 +1,6 @@
1      1  from django.shortcuts import render
2      2
3      -3  # Create your views here.
4      +3
5      +4  def home(request):
6      +5      return render(request, 'home.html',{})
7      +6
```

8

Templates

Hora de ver os resultados





Criando um novo template

Etapas da criação de uma nova tela:




Definir a nova rota no arquivo “projeto/sistema/**urls.py**”



Definir a nova view no arquivo “projeto/sistema/**views.py**”



Criar o novo template na pasta templates 

Criar a pasta “projeto/sistema/**templates**”

Baixar o arquivo do template >> <https://goo.gl/MxXbNb>

Colocar dentro da pasta templates

```
$ python manage.py runserver
```



Criando um novo template

Head:

```
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
        integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXEoaoApmYm8liuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Homepage</title>
</head>
```



Criando um novo template

Body:

```
<body>
  <header>
    <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
      <a class="navbar-brand" href="#">Sistema</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse"
        aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item active"><a class="nav-link" href="#">Home</a></li>
          <li class="nav-item"><a class="nav-link" href="#">Alunos</a></li>
        </ul>
        <form class="form-inline mt-2 mt-md-0">
          <input class="form-control mr-sm-2" type="text" placeholder="O que deseja buscar?" aria-label="Search">
          <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Buscar</button>
        </form>
      </div>
    </nav>
  </header>
  <main role="main" class="container">
    <div class="pt-5 text-center">
      <h1 class="mt-5">Seja bem-vindo!</h1>
      <p class="lead">Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar
        os próximos conceitos e realizar experimentos</p>
    </div>
  </main>
</body>
```



Criando um novo template

[Sistema](#) [Home](#) [Alunos](#)

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

9

Models

Integrando o banco de dados





Models

- Modelos representam nossos dados
 - No Django uma classe equivale a uma tabela
 - Não se usa SQL diretamente
-
- Usaremos SQLite como o banco de dados
 - Trabalharemos com migrações



Models



Criar o novo modelo em
“projeto/sistema/**models.py**”

```
@@ -1,3 +1,11 @@
1      1  from django.db import models
2      2
-3      # Create your models here.
+3
+4  class Aluno(models.Model):
+5      nome = models.CharField(max_length=100, blank=False)
+6      ra = models.PositiveIntegerField(default=123456, blank=False)
+7      trancou = models.BooleanField(null=True, blank=True, default=False)
+8
+9      def __str__(self):
+10         return self.nome
+11
```



Adicionar o novo modelo em
“projeto/sistema/**admin.py**”

```
@@ -1,3 +1,4 @@
1      1  from django.contrib import admin
+2      2  from .models import Aluno
2      3
-3      # Register your models here.
+4      admin.site.register(Aluno)
```



Models

- Acabamos de alterar o estado do banco de dados
- Precisamos dizer pro Django que uma nova versão do banco está disponível

Cria uma nova migração

```
$ python manage.py makemigrations sistema
```

Implanta essa migração

```
$ python manage.py migrate
```



Django Admin

```
$ python manage.py createsuperuser
```

- Nome: nome qualquer
- Email: email qualquer
- Senha: senha qualquer

```
(myvenv) C:\Users\Rafael\Desktop\projeto_django\projeto>python manage.py createsuperuser
Username (leave blank to use 'rafael'): Rafael
Email address: rafael@rafael.com
Password:
Password (again):
Superuser created successfully.
```

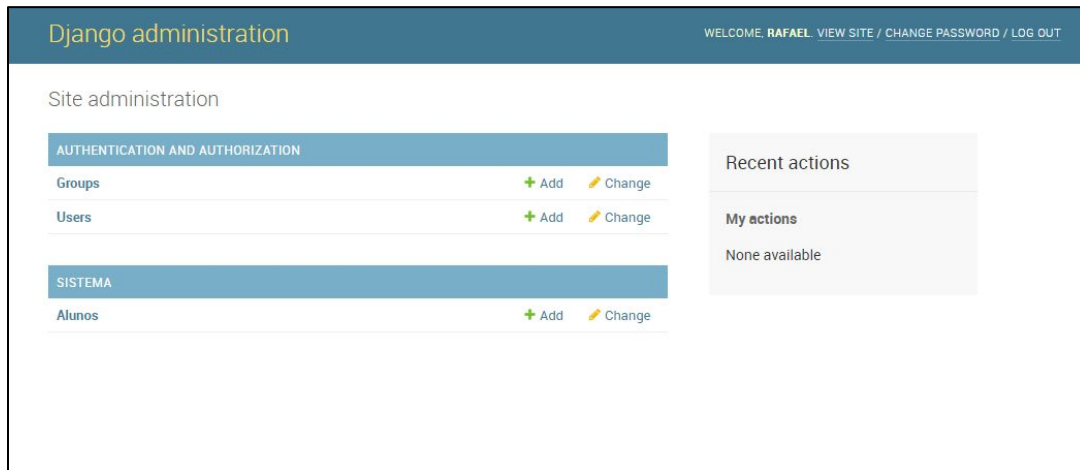


Django Admin

\$ python manage.py runserver

No fim da URL digite “/admin”

Faça o login com os seus dados





Django Admin

Cadastre manualmente 3 alunos

Django administration

WELCOME, RAFAEL. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Sistema > Alunos

✓ The aluno "Aluno 3" was added successfully.

Select aluno to change

ADD ALUNO +

Action: 0 of 3 selected

<input type="checkbox"/>	ALUNO
<input type="checkbox"/>	Aluno 3
<input type="checkbox"/>	Aluno 2
<input type="checkbox"/>	Aluno 1

3 alunos

Django administration

WELCOME, RAFAEL. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Sistema > Alunos > Aluno 1

Change aluno

Nome:

Ra:

☐ Trancou

10

Template Tags

Representando suas variáveis
no template





Template Tags

- Comunicação Python <-> HTML
- Existem diversas funcionalidades para template tags
- Lógica no HTML (if-else, for, while...)
- Acessar parâmetros passados
- Acessar arquivos estáticos (img, css, js)
- Acessar URLs



Trabalhando com template tags

Primeiro vamos alterar a nossa view e fazer ela acessar o banco de alunos

```
@@ -1,6 +1,8 @@
1      1  from django.shortcuts import render
      +2  from .models import Aluno
2
3      3
4      4
5      5  def home(request):
      -5      return render(request, 'homt.html', {})
      +6      alunos = Aluno.objects.all()
      +7      return render(request, 'home.html', {'alunos':alunos})
6      8
```

OBS: o terceiro parâmetro é um dicionário, funciona como um vetor de pares 'chave':valor

A variável alunos agora possui todos os alunos da tabela Alunos

A função render agora devolve os alunos como um parâmetro



Trabalhando com template tags

Agora vamos criar uma tabela no nosso template para representar os alunos

```
34      os próximos conceitos e realizar experimentos</p>
+35
+36      <h2 class="mt-5">Alunos</h2>
+37      <table class="table mt-5">
+38          <thead>
+39              <tr>
+40                  <th>#</th>
+41                  <th>Nome</th>
+42                  <th>RA</th>
+43                  <th>Situação</th>
+44              </tr>
+45          </thead>
+46          <tbody>
+47              <!-- Aqui serão colocados os alunos-->
+48          </tbody>
+49      </table>
50  </div>
```

OBS: insira o código dentro da div da main, logo após o </p>

Atualize a página para testar



Trabalhando com template tags

Agora queremos acessar estes alunos diretamente no HTML e, para cada aluno criar uma linha dinamicamente com seus dados

```
46      46      <tbody>
-47      -47      <!-- Aqui serão colocados os alunos-->
+47      +47      {% for aluno in alunos %}
+48      +48      <!-- Aqui serão colocados os alunos-->
+49      +49      {% endfor %}
48      50      </tbody>
```



Trabalhando com template tags

Por fim, queremos criar linhas nessa tabela acessando diretamente os campos de cada aluno

```
46      46      <tbody>
-47      -47      <!-- Aqui serão colocados os alunos-->
+47      +47      {% for aluno in alunos %}
+48      +48      <tr>
+49      +49      <td>{{aluno.pk}}</td>
+50      +50      <td>{{aluno.nome}}</td>
+51      +51      <td>{{aluno.ra}}</td>
+52      +52      <td>
+53      +53      {% if aluno.trancou == True %}
+54      +54      Trancou
+55      +55      {% else %}
+56      +56      Cursando
+57      +57      {% endif %}
+58      +58      </td>
+59      +59      </tr>
+60      +60      {% endfor %}
48      61      </tbody>
```



Trabalhando com template tags

Atualizando a página:

[Sistema](#) [Home](#) [Alunos](#)

O que deseja buscar?

Buscar

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

Alunos

#	Nome	RA	Situação
1	Aluno 1	726583	Cursando
2	Aluno 2	654321	Trancou
3	Aluno 3	123456	Cursando



1

Bônus

Herança de HTMLs

Projete uma vez, e reutilize





Herdando HTMLs

- Django utiliza blocos de HTML
- Blocos podem ser substituídos por outros blocos de mesmo nome
- Basta dizermos que em determinada página o “bloco X” corresponde a um novo conteúdo
- Assim reutilizamos aquilo que for interessante, apenas mudando trechos



Herdando HTMLs

Primeiro vamos transformar nossa main em um bloco usando template tags

Em seguida vamos criar uma nova página, repetindo todo aquele procedimento

- Criar rota
- Criar view
- Criar template

```
@@ -27,6 +27,7 @@
27      27          </div>
28      28          </nav>
29      29          </header>
30      +30          {% block content %}
31      31      <main role="main" class="container">
32      32          <div class="pt-5 text-center">
33      33              <h1 class="mt-5 ">Seja bem-vindo!</h1>

@@ -62,6 +63,7 @@
62      63          </table>
63      64          </div>
64      65          </main>
65      +66          {% endblock %}
66      67      </body>
67      68      </html>
68      69
```



Herdando HTMLs

```
@@ -3,4 +3,5 @@ from . import views
```

```
3     3
4     4     urlpatterns = [
5     5         path('', views.home, name="home"),
+6         path('adicionar', views.adicionar, name="adicionar")
6     7     ]
```

urls.py

```
@@ -6,3 +6,16 @@ def home(request):
```

```
6     6     alunos = Aluno.objects.all()
7     7     return render(request, 'homt.html', {'alunos':alunos})
8     8
+9     def adicionar(request):
+10
+11         if request.method == "POST":
+12             nome = request.POST.get('nome')
+13             ra = request.POST.get('ra')
+14             aluno = Aluno.objects.create()
+15             aluno.nome = nome
+16             aluno.ra = ra
+17             aluno.trancou = False
+18             aluno.save()
+19             return redirect('home')
+20
+21     return render(request, 'adicionar.html', {})
```

views.py

Agora vamos imaginar nosso novo template HTML. Precisamos digitar todo o código do zero?



Herdando HTMLs

Não!

Precisamos apenas indicar que estamos herdando (estendendo) um html existente.

Apenas substituindo um bloco existente por um novo com o mesmo nome

```
{% extends "home.html" %}  
  
{% block content %}  
    <!-- Nossa nova página virá aqui-->  
{% endblock %}
```

adicionar.html

Baixar o arquivo da tela de adicionar >> <https://goo.gl/MxXbNb>



Herdando HTMLs

Ao recarregar a página:

[Sistema](#) [Home](#) [Alunos](#) [Buscar](#)

Adicionar um novo aluno

Esta é a página que usaremos para adicionar novos alunos

Preencha o fomulário

Nome do aluno

RA do aluno

[Adicionar](#)



Herdando HTMLs

Um último detalhe, alterando o arquivo **home.html**

```
@@ -17,8 +17,8 @@
 17      17      </button>
 18      18      <div class="collapse navbar-collapse" id="navbarCollapse">
 19      19      <ul class="navbar-nav mr-auto">
-20          <li class="nav-item active"><a class="nav-link" href="#">Home</a></li>
-21          <li class="nav-item"><a class="nav-link" href="#">Alunos</a></li>
+20          <li class="nav-item active"><a class="nav-link" href="{% url 'home' %}">Home</a></li>
+21          <li class="nav-item"><a class="nav-link" href="{% url 'adicionar' %}">Adicionar</a></li>
 22      22      </ul>
 23      23      <form class="form-inline mt-2 mt-md-0">
```

Vamos adicionar um link para a nova página usando as template tags

Assim não precisaremos ficar digitando urls toda hora

2

Bônus

URLs Dinâmicas

One template to rule them all





URLs Dinâmicas

- Imagine que você adicionou 100 alunos
- Como fazer uma página para editar/apagar alunos?
- Uma página para cada?
- E se o usuário criar mais alunos?



URLs Dinâmicas

Queremos poder utilizar algo do tipo:

```
http://localhost:8000/aluno/1/delete  
http://localhost:8000/aluno/2/delete  
http://localhost:8000/aluno/3/delete  
http://localhost:8000/aluno/4/delete  
.  
.  
.  
http://localhost:8000/aluno/(ID)/delete
```

Da mesma maneira, queremos um template inteligente o bastante para saber lidar com diversas requisições



URLs Dinâmicas

Começaremos pela URL inteligente

```
@@ -3,5 +3,6 @@ from . import views
```

```
3      3
4      4  urlpatterns = [
5      5      path('', views.home, name="home"),
-6      path('adicionar', views.adicionar, name="adicionar")
+6      path('adicionar', views.adicionar, name="adicionar"),
+7      path('aluno/<int:id>/remove', views.remove, name="remover"),
7      8  ]
```

Aqui queremos dizer que “id” será dinâmico, e será do tipo inteiro



URLs Dinâmicas

Agora precisamos acessar esse parâmetro e apagar o aluno do banco

```
@@ -1,4 +1,4 @@
-1      from django.shortcuts import render, redirect
+1      from django.shortcuts import render, redirect, get_object_or_404
2      2      from .models import Aluno
3      3
4      4

@@ -19,3 +19,8 @@ def adicionar(request):
19      19      return redirect('home')
20      20
21      21      return render(request, 'adicionar.html', {})
+22
+23      def remover(request, id):
+24          aluno = get_object_or_404(Aluno, pk=id)
+25          aluno.delete()
+26          return redirect('home')
```

Neste caso não precisaremos de um template, já que estamos redirecionando automaticamente

Agora precisamos apenas achar um jeito de criar um jeito de ligar cada aluno com um caminho para removê-lo



URLs Dinâmicas

Podemos usar a própria lógica da tabela dinâmica. Que já passava aluno por aluno

Adicionar um novo <th>

44	44	<th>Nome</th>
45	45	<th>RA</th>
46	46	<th>Situação</th>
+47		<th>Apagar</th>

E apenas criar links <a> para acessar a URL de remoção

57	57	Cursando
58	58	{% endif %}
59	59	</td>
+60		<td>Apagar</td>
60	61	</tr>
61	62	{% endfor %}
62	63	</tbody>



URLs Dinâmicas

Recarregando a página:

[Sistema](#) [Home](#) [Adicionar](#)

O que deseja buscar? [Buscar](#)

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

Alunos

#	Nome	RA	Situação	Remover
1	Aluno 1	726583	Cursando	Apagar
2	Aluno 2	654321	Trancou	Apagar
3	Aluno 3	123456	Cursando	Apagar
4	Rafael	456789	Cursando	Apagar
5	Joao	987512	Cursando	Apagar



Exercício

Implemente um mecanismo para indicar que um aluno trancou

Sistema Home Adicionar

O que deseja buscar? Buscar

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

Alunos

#	Nome	RA	Situação	Remover	Trancar
1	Aluno 1	726583	Trancou	Apagar	Trancar
2	Aluno 2	654321	Trancou	Apagar	Trancar
3	Aluno 3	123456	Trancou	Apagar	Trancar
5	Joao	987512	Trancou	Apagar	Trancar



Exercício

Implemente um mecanismo para editar um aluno existente

[Sistema](#) [Home](#) [Adicionar](#)

Editar um aluno existente

Esta é a página que usaremos para Editar um aluno já existe

Preencha o fomulário

Nome do aluno

RA do aluno



Exercício

Implemente um mecanismo para editar um aluno existente

[Sistema](#) [Home](#) [Adicionar](#)

O que deseja buscar? [Buscar](#)

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

Alunos

#	Nome	RA	Situação	Editar	Remover	Trancar
1	Alterado1	726583	Trancou	Editar	Apagar	Trancar
2	Alterado2	654321	Trancou	Editar	Apagar	Trancar
3	Alterado3	123456	Trancou	Editar	Apagar	Trancar
5	Alterado4	987512	Trancou	Editar	Apagar	Trancar



Respostas

views.py

Trancar

```
def trancar(request, id):  
    aluno = get_object_or_404(Aluno, pk=id)  
    aluno.trancou = True  
    aluno.save()  
    return redirect('home')
```

Editar

```
def editar(request, id):  
    aluno = get_object_or_404(Aluno, pk=id)  
  
    if request.method == "POST":  
        nome = request.POST.get('nome')  
        ra = request.POST.get('ra')  
        aluno.nome = nome  
        aluno.ra = ra  
        aluno.save()  
        return redirect('home')  
  
    return render(request, 'editar.html', {'aluno':aluno})
```



Respostas

Baixar o arquivo do da tela de editar >> <https://goo.gl/MxXbNb>

urls.py

```
@@ -6,4 +6,5 @@ urlpatterns = [  
    6      6      path('adicionar', views.adicionar, name="adicionar"),  
    7      7      path('aluno/<int:id>/remove', views.remover, name="remover"),  
    8      8      path('aluno/<int:id>/trancar', views.trancar, name="trancar"),  
    +9      path('aluno/<int:id>/editar', views.editar, name="editar"),  
    9      10     ]  
    9      10     \ No newline at end of file
```

home.html

```
    +48     <th>Trancar</th>  
    +49     <th>Editar</th>  
    47     50     </tr>  
    48     51     </thead>  
    49     52     <tbody>
```

```
<td><a href="{% url 'remover' id=aluno.pk %}">Apagar</a></td>  
<td><a href="{% url 'trancar' id=aluno.pk %}">Trancar</a></td>  
<td><a href="{% url 'editar' id=aluno.pk %}">Editar</a></td>
```



3

Bônus

Arquivos estáticos

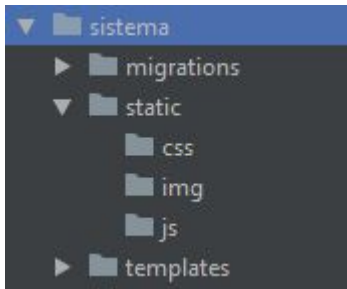
Trabalhando com CSS, JS,
imagens localmente





Trabalhando com arquivos estáticos

Primeiro vamos criar as devidas pastas e arquivos



Dentro da pasta da nossa aplicação, criaremos a pasta “projeto/sistema/**static**”.

Dentro da pasta static criaremos pastas para separar melhor nossos arquivos

```
td {  
  background-color: #666666;  
  color: #ffffff;  
}  
  
.center_div{  
  margin: 0 auto;  
  width:50%;  
}
```

Dentro da pasta css criaremos o arquivo “style.css”, para conseguirmos enxergar mudanças



Trabalhando com arquivos estáticos

Agora vamos importar nosso novo arquivo no template

```
@@ -1,9 +1,11 @@
 1      1      <!DOCTYPE html>
      +2      {% load staticfiles %}
 2      3      <html lang="en">
```

Primeiro vamos dizer ao Django que queremos usar arquivos estáticos dentro deste HTML

```
7      integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXo
+8      <link rel="stylesheet" href="{% static 'css/style.css' %}">
9      <meta name="viewport" content="width=device-width, initial-scale=1, sh
10     <title>Homepage</title>
```

Depois vamos importar nosso próprio CSS. Lembre-se de importar após importar o Bootstrap

```
@@ -119,3 +119,4 @@ USE_TZ = True
119    119    # https://docs.djangoproject.
120    120
121    121    STATIC_URL = '/static/'
+122    STATIC_RROT = 'sistema'
```

Por fim, no arquivo settings.py, criar os seguintes campos



Trabalhando com arquivos estáticos

Ao recarregar a página:

[Sistema](#) [Home](#) [Alunos](#)

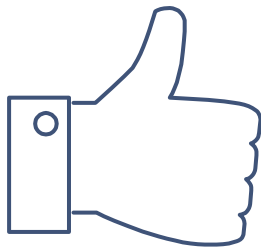
O que deseja buscar? Buscar

Seja bem-vindo!

Esta é a nova homepage da nossa aplicação, aqui iremos trabalhar os próximos conceitos e realizar experimentos

Alunos

#	Nome	RA	Situação
1	Aluno 1	726583	Cursando
2	Aluno 2	654321	Trancou
3	Aluno 3	123456	Cursando



OBRIGADO!

Escola Piloto de Computação



epic-ufscar.org



epic.ufscar@gmail.com



fb.com/epicufscar



Referências

- Documentação Django: <https://docs.djangoproject.com/pt-br/2.1/>
- Projeto no Github: <https://github.com/epicufscar/workshop-django>
- Documentação Bootstrap: <https://getbootstrap.com/>

Outras Referências & Cursos online:

- Tutorial Djangogirls <https://tutorial.djangogirls.org/pt/django/>
- Curso Alura Django <https://cursos.alura.com.br/course/introducao-ao-django>
- Curso Udemy Django <http://bit.do/django-udemy>
-