**ETAPA 7**

**1- primeira parte, criando os forms e views do adcionar**

primeira parte, devemos abrir a parte de forms , e vamos criar a parte de forms, para que podermos chamar eles na parte de views no caso o forms em questão é esse

class DoadorForm(forms.ModelForm):

class Meta:

model = Doador

fields = ('nome', 'cpf', 'contato', 'tipo\_sanguineo', 'rh', 'tipo\_rh\_corretos', 'situacao')

widgets = {

'tipo\_rh\_corretos': forms.CheckboxInput(attrs={'class': 'form-check-input'}),

'tipo\_sanguineo': forms.RadioSelect(attrs={'class': 'form-check-input'}),

'rh': forms.RadioSelect(attrs={'class': 'form-check-input'}),

}

# Correção: field\_order deve estar fora da classe Meta

field\_order = ('nome', 'cpf', 'contato', 'tipo\_sanguineo', 'rh', 'tipo\_rh\_corretos', 'situacao')

esse form vai setar os modulos, e criar um formulario automatizado, facilitando parte do codigo,

**1-2Views**

Nessa parte devemos ir no APP/views.py daonde vamos criar o formulario de criação nele devemos pensar que vamos inserir novos itens no codigo, no caso o formulario de criação é esse

def insertdoacao(request):

doadores = Doador.objects.all()

# Crie uma lista de tuplas (código, nome) para usar no campo de escolha

lista\_doadores = [(doador.codigo, doador.nome) for doador in doadores]

order\_forms = Doacao()

if request.method == 'POST':

forms = DoacaoForm(request.POST, request.FILES,

instance=order\_forms, prefix='main')

#codigo = int(request.POST.get("id\_main-codigo\_doador"))

#doador = Doador.objects.get(pk=codigo)

# Modifique o campo tipo\_rh\_corretos

#doador.tipo\_rh\_corretos = True

# doador.save()

if forms.is\_valid() :

teste\_instance = forms.save()

# return HttpResponseRedirect(resolve\_url('detalhe\_doador',teste\_instance.pk))

else:

forms = DoacaoForm(instance=order\_forms, prefix='main')

context = {

'forms': forms,

'lista\_doadores':lista\_doadores,

}

return render(request,'insere\_doacao.html',context)

Depois de fazer a view , devemos fazer o html que irá receber essa view, no caso o nome do html deve ser “insere\_doacao.html, nele deve conter a função {% csrf\_token %} e a {forms|bootstrap} e ambos devem estar dentro de um <forms method =”POST”,o file.html deve ser criado dentro de APP/Templates/ ,o codigo pronto em questão é esse :

{% extends 'base.html' %}

{% load bootstrap %}

{# Adding select2 css #}

{% block css %}

<link href="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.3/css/select2.min.css" rel="stylesheet"/>

{% endblock %}

{# Adding select2 js #}

{% block js %}

<script src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.3/js/select2.min.js"></script>

{% endblock %}

{% block content %}

<div class="row">

<div class="col-xs-10 col-md-12">

<form action="" method="POST">

{% csrf\_token %}

<div class="row">

<div class="col-xs-8 col-md-5">

{{ forms|bootstrap }}

<select name="codigo\_doador">

{% for codigo, nome in lista\_doadores %}

<option value="{{ codigo }}">{{ nome }}</option>

{% endfor %}

</select>

</div>

</div>

<div class="form-inline buttons">

<i class="fa fa-times"></i>

Cancelar

</a>

<button class="btn btn-primary pull-right" value="Save">

<i class="fa fa-floppy-o"></i>

Salvar

</button>

</div>

</form>

</div>

</div>

{% endblock %}

**1.2 - URL do adcionar**

Falta ainda criar a url do adcionar, então no caso vá para o APP/url.py, e insira a seguinte linha de codigo , que deve ir dentro de url patterns

path('adcionar\_doador/',insertdoador,name='insertdoador'),

**2 -url views e html do detalhe\_doador e lista doador**

depois disso , devemos criar a parte de detalhes do doador, o detalhe doador seria esse :

def detalhe\_doador(request, pk):

object = get\_object\_or\_404(Doador,pk=pk)

context = {

'object':object,

}

return render(request,'detalhe\_doador.html',context)

NO html deve ser instanciado item por item do object, que deve ser escrito assim:

{object.item}

Mas para facilitar segue o codigo html criado dentro da pasta APP/templates :

**detalhe\_doador.html:**

{% extends "base.html" %}

{% load static %}

{% block content %}

<div class="col-lg-8">

<div class="col-md-2 column">

</div>

<div class="col-md-offset-3 column">

</div>

</br>

<table class="table table-user-information">

<tbody>

<tr>

<th class="col-md-3 text-right">codigo</th>

<td>{{object.codigo}}</td>

</tr>

<tr>

<tr>

<th class="col-md-3 text-right">nome</th>

<td>{{object.nome}}</td>

</tr>

<tr>

<th class="col-md-3 text-right">cpf</th>

<td>{{object.cpf}}</td>

</tr>

<tr>

<th class="col-md-3 text-right">contato </th>

<td>{{object.contato}}</td>

</tr>

<tr>

<th class="col-md-3 text-right">tipo sanguineo</th>

<td>{{object.tipo\_sanguineo}}/</td>

</tr>

<tr>

<th class="col-md-3 text-right">rh</th>

<td>{{object.rh}}/</td>

</tr>

<tr>

<th class="col-md-3 text-right">tipo sanguineo</th>

<td>{{object.tipo\_sanguineo}}/</td>

</tr>

<tr>

<th class="col-md-3 text-right">tipo sanguineo</th>

<td>{{object.tipo\_sanguineo}}/</td>

</tr>

<tr>

<th class="col-md-3 text-right">tipo sanguineo</th>

<td>{{object.tipo\_sanguineo}}/</td>

</tr>

<tr>

</tr>

</tbody>

</table>

<a href="{{ object.get\_modifica }}">

<button id="altera\_doador" type="button" class="btn btn-primary">

<span class="glyphicon glyphicon-plus"></span> Modificar

</button>

<a href="{{ object.get\_exclui }}">

<button id="Deleta Doador" type="button" class="btn btn-primary">

<span class="glyphicon glyphicon-plus"></span> Deletar

</button>

<a href="{% url 'Doador\_list' %}">

<button id="doador\_list" type="button" class="btn btn-primary">

<span class="glyphicon glyphicon-plus"></span> Voltar

</button>

{%endblock%}

**2-1 Funções dentro de models**

Agora vamos para dentro do models antes de irmos para o urls, vamos criar uma pequena função que facilitará na parte de list, para isso , va até APP/models.py e insira essa linha logo abaixo da outra função

def get\_detalhe(self):

return f'/teste/Doador/{self.codigo}/'

deve ser lembrado que a indentação do python é obrigatória, as linhas devem estar alinhadas e com os espaços devido se não a função não vai ser lida, ou vai dar erro de indentação no caso o codigo inteiro do models fica assim : class Doador(models.Model):

options\_choices = (

('A', ' tipo A '),

('B', 'tipo B '),

('AB','tibo AB '),

('O', 'tipo O '),

)

CHOICES = (

('+', 'RH + '),

('-', 'RH - '),

)

codigo = models.AutoField("alto completo",null=False,primary\_key=True)

nome = models.CharField("Nome",blank=True,max\_length=200)

cpf = models.CharField("CPF",blank=True,max\_length=11)

contato = models.CharField("numero",blank=True,max\_length=12)

tipo\_sanguineo = models.CharField("tipo sanguineo",blank=True,null =False ,max\_length=4 ,choices = options\_choices)

rh = models.CharField("RH",blank=True,max\_length=1 ,choices= CHOICES)

tipo\_rh\_corretos = models.BooleanField("RH corretos",blank=True)

situacao = models.CharField("situcao", blank=True)

class Meta:

ordering = ['codigo']

verbose\_name = 'Doador'

def get\_detalhe(self):

return f'/teste/Doador/{self.codigo}/'

**2-2 Doador List**

Com isso so falta mesmo a lista, para a lista doador fui um pouco mais alem, para fazer a lista devemos levar em conta um sistema de filtrar os doadores por cada um de seus campus , no caso fica sendo assim :

def Doador\_list(request):

# Obtém todos os objetos da tabela Doador

object\_list = Doador.objects.all().order\_by('codigo')

# Instancia o formulário com os dados da requisição GET

form = DoadorForm(request.GET)

# Verifica se o formulário é válido

if form.is\_valid():

# Obtém os dados do formulário

nome = form.cleaned\_data.get('nome')

cpf = form.cleaned\_data.get('cpf')

contato = form.cleaned\_data.get('contato')

tipo\_sanguineo = form.cleaned\_data.get('tipo\_sanguineo')

rh = form.cleaned\_data.get('rh')

tipo\_rh\_corretos = form.cleaned\_data.get('tipo\_rh\_corretos')

situacao = form.cleaned\_data.get('situacao')

# Aplica os filtros conforme necessário

if nome:

object\_list = object\_list.filter(nome\_\_icontains=nome)

if cpf:

object\_list = object\_list.filter(cpf\_\_icontains=cpf)

if contato:

object\_list = object\_list.filter(contato\_\_icontains=contato)

if tipo\_sanguineo:

object\_list = object\_list.filter(tipo\_sanguineo\_\_icontains=tipo\_sanguineo)

if rh:

object\_list = object\_list.filter(rh\_\_icontains=rh)

if tipo\_rh\_corretos:

object\_list = object\_list.filter(tipo\_rh\_corretos\_\_exact=tipo\_rh\_corretos)

if situacao:

object\_list = object\_list.filter(situacao\_\_icontains=situacao)

# Paginação dos resultados

paginator = Paginator(object\_list, 20)

page = request.GET.get('page', 1)

try:

doador = paginator.page(page)

except PageNotAnInteger:

doador = paginator.page(1)

except EmptyPage:

doador = paginator.page(paginator.num\_pages)

context = {

'object\_list': object\_list,

'doador': doador,

'form': form

}

return render(request, 'lista\_doador.html', context)

Com isso criado está faltando criar o html ,e vamos usar a func {{ form.campus\_do\_form }} para cada um dos campos que vamos cetar para filtrar a lista dos doadores, e claro, para ver o detalhe de cada doador, é só clicar na sua chave primária :

{% extends 'base.html' %}

{% block title %}Lista de Doador{% endblock title %}

{% block subtitle %}Lista de Doador{% endblock %}

{% block content %}

<form class="navbar-form navbar-form" action="." method="get">

<!-- add -->

<a href="{% url 'insertdoador' %}">

<button id="new\_customer" type="button" class="btn btn-primary">

<span class="glyphicon glyphicon-plus"></span> Adicionar

</button>

</a>

<form method="get">

<div class="row">

<div class="col-xs-8 col-md-5">

<p>{{ form.nome.label\_tag }}</p>

{{ form.nome }}

<p>

{{ form.cpf.label\_tag }}</p>

{{ form.cpf }}

<p>

{{ form.contato.label\_tag }}</p>

{{ form.contato }}

<p>

{{ form.tipo\_sanguineo.label\_tag }}</p>

{{ form.tipo\_sanguineo }}

<p>{{ form.rh.label\_tag }}</p>

{{ form.rh }}

<p></p>{{ form.situacao.label\_tag }}</p>

{{ form.situacao }}

<button type="submit">Pesquisar</button>

</div>

</div>

</form>

</form>

{% if object\_list %}

<div>

<table class="table">

<thead>

<tr>

<th class="text-center">codigo</th>

<th class="text-center">Nome</th>

<th class="text-center">cpf</th>

<th class="text-center">contato</th>

<th class="text-center">tipo\_sanguineo</th>

<th class="text-center">rh</th>

<th class="text-center">tipo\_rh\_corretos</th>

<th class="text-center">situacao</th>

</tr>

</thead>

<tbody>

{% for doador in object\_list %}

<tr>

<td class="text-center"><a href="{{ doador.get\_detalhe }}"

title="Clique para ver os detalhes do doador selecionado">

{{ doador.codigo}}</a></td>

<td class="text-center">{{ doador.nome}}</td>

<td class="text-center">{{ doador.cpf}}</td>

<td class="text-center">{{ doador.contato }}</td>

<td class="text-center">{{ doador.tipo\_sanguineo}}</td>

<td class="text-center">{{ doador.rh}}</td>

<td class="text-center">{{ doador.tipo\_rh\_corretos}}</td>

<td class="text-center">{{ doador.situacao}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% else %}

<p class="alert alert-warning">Sem itens na lista.</p>

{% endif %}

<hr>

<div>

<h3>{{ object\_list|length }} doador{{ object\_list|length|pluralize }}</h3>

<h4><b>Total:</b> {{ page\_obj.paginator.count }} doador{{ page\_obj.paginator.count|pluralize }}</h4>

</div>

{% if object\_list.has\_other\_pages %}

<div class="row text-center">

<div class="col-lg-12">

<ul class="pagination">

{% if object\_list.has\_previous %}

<li><a href="?page={{ object\_list.previous\_page\_number }}">&laquo;</a></li>

<li><a href="?page=1">1</a></li>

{% if object\_list.previous\_page\_number != 1 %}

<li><a href="?page={{ object\_list.previous\_page\_number }}">{{ object\_list.previous\_page\_number }}</a></li>

{% endif %}

{% else %}

<li class="disabled"><span>&laquo;</span></li>

{% endif %}

<li class="active"><span>{{ object\_list.number }} <span class="sr-only">(current)</span></span></li>

{% if object\_list.has\_next %}

{% if object\_list.next\_page\_number != object\_list.paginator.num\_pages %}

<li><a href="?page={{ object\_list.next\_page\_number }}">{{ object\_list.next\_page\_number }}</a></li>

{% endif %}

<li><a href="?page={{ object\_list.paginator.num\_pages }}">{{ object\_list.paginator.num\_pages }}</a></li>

<li><a href="?page={{ object\_list.next\_page\_number }}">&raquo;</a></li>

{% endif %}

</ul>

</div>

</div>

{% endif %}

{% endblock content %}

**2-3 urls**

Mais uma vez devemos criar mais uma linha no url\_patterns, dentro de APP/urls crie essa url :

path('Doador/<int:pk>/',view=detalhe\_doador,name='detalhe\_doador'),

**3 - Altera doador**

Agora falta só o altera doador, para isso devemos voltar em APP/views.py e criar a nossa view de alteração , para realizar a alteração devemos setar a mesma pela sua pk assim podemos pegar ela e fazer a alteração que der :

def altera\_doador(request,pk):

order\_forms = Doador()

doador = get\_object\_or\_404(Doador,pk=pk)

i = 1

if request.method == 'GET':

forms = DoadorForm(instance=doador,prefix='main')

else:

forms = DoadorForm(request.POST,request.FILES,instance=doador,prefix='main')

if forms.is\_valid():

forms = forms.save(commit=False)

# apagado user.profile da linha 163 , Davi Oliveira Tito 02/05/2022

i = 0

forms.save()

return HttpResponseRedirect(resolve\_url('detalhe\_doador',forms.pk))

else:

print("erro no doador")

context = {

'forms':forms,

}

return render(request,'altera\_doador.html',context)

**3-1 Html do altera Doador**

Segue o mesmo princípio do adiciona doador,tendo somente algumas alterações, e mais uma vez , mais uma vez va para pasta APP/templates, e crie um html de nome altera\_doador.html , e insira esse codigo:

{% extends 'base.html' %}

{% load crispy\_forms\_tags %}

{% load bootstrap %}

{% block title %}Alterar Doador{% endblock %}

{% block subtitle %}Alterar Doador{% endblock %}

{% block content %}

<br>

<div class="row align-items-center">

<div class="col-sm-3">

</div>

<div class="col-sm-6">

<h4>Você pode alterar sua conta usando o formulário a seguir:</h4>

</div>

</div>

<br>

<div class="row align-items-center">

<div class="col-sm-3">

</div>

<div class="col-sm-6">

<form method="post" enctype="multipart/form-data">

{{ forms|bootstrap }}

{% csrf\_token %}

<p><input type="submit" class="btn btn-primary" value="Salvar alterações" title="Clique para salvar as alterações realizadas no perfil"></p>

</form>

</div>

</div>

{% endblock %}

Como podemos ver, ele quase não tem alteração em relação ao html de insere\_doador.html, a não ser o titulo .

**3-2 nova def dentro de models.py**

agora antes de passarmos para criar a nova url, devemos criar uma nova função dentro de models.py, para isso, vá para o file APP/models.ṕy, e crie mais uma def de nome def\_modifica, que deve parecer com a de detalhe, que seria assim :

def get\_modifica(self):

return f'/teste/altera\_doador/{self.codigo}/'

PS:NÃO ESQUECER DA INDENTAÇÃO DO PYTHON, SE NÃO VAI DAR ERRO DE INTENTAÇÃO AS LINHAS DENTRO DE UMA FUNÇÃO DEVEM ESTAR A FRENTE DELA, PARA MOSTRAR QUE ESTÃO DENTRO DA FUNÇÃO

**3-3 URLS de altera**

Agora vamos voltar para APP/urls.py, devemos criar mais uma linha para a nova view que criamos, a linha deve estar dentro de url\_patterns , a linha em questão é essa :

path('altera\_doador/<int:pk>/',view=altera\_doador,name='altera\_doador'),

**4- Deleta Doador:** Sabe\_se que, nunca devemos deletar uma linha do banco de dados, ja que isso serve mais para controle , então dentro da APP/views.py devemos modificar a “situacao”, e fixar o valor de inativo nela, para caso o cliente retornar, reativamos o cadastro , o codigo em questão é esse :

def deleta\_doador(request,pk):

print("entrou ")

doador = get\_object\_or\_404(Doador,pk=pk)

doador.situacao = 'inativo'

doador.save()

return redirect('Doador\_list')

e nesse caso não precisamos criar um html, mas sim somente uma url, e já vou colocar o codigo de urls inteiro aqui, o codigo APP/urls.py é esse :

from django.urls import include, path

from django.contrib.auth.decorators import login\_required

from django.conf import settings

from django.conf.urls.static import static

#from django.conf.urls.static import media

from rest\_framework import routers

from .views import \*

urlpatterns = [

path('Lista\_Doador/',Doador\_list,name='Doador\_list'),

path('adcionar\_doador/',insertdoador,name='insertdoador'),

path('<int:pk>/',view=detalhe\_formulario,name='detalhe\_formulario'),

path('Doador/<int:pk>/',view=detalhe\_doador,name='detalhe\_doador'), path('deleta\_Doador/<int:pk>/',view=deleta\_doador,name='deleta\_doador'),

path('altera\_doador/<int:pk>/',view=altera\_doador,name='altera\_doador'),

]

PS: as linhas de path devem estar alinhadas, elas devem ter a mesma margem de espaço, devido a um erro desse editor de texto não conseguir colocar elas.

**4-1 outra funcão dentro de models**

Agora volte para a file APP/models.py e vamos criar mais uma def, que vai deixar o codigo assim :

class Doador(models.Model):

options\_choices = (

('A', ' tipo A '),

('B', 'tipo B '),

('AB','tibo AB '),

('O', 'tipo O '),

)

CHOICES = (

('+', 'RH + '),

('-', 'RH - '),

)

codigo = models.AutoField("alto completo",null=False,primary\_key=True)

nome = models.CharField("Nome",blank=True,max\_length=200)

cpf = models.CharField("CPF",blank=True,max\_length=11)

contato = models.CharField("numero",blank=True,max\_length=12)

tipo\_sanguineo = models.CharField("tipo sanguineo",blank=True,null =False ,max\_length=4 ,choices = options\_choices)

rh = models.CharField("RH",blank=True,max\_length=1 ,choices= CHOICES)

tipo\_rh\_corretos = models.BooleanField("RH corretos",blank=True)

situacao = models.CharField("situcao", blank=True)

class Meta:

ordering = ['codigo']

verbose\_name = 'Doador'

def get\_detalhe(self):

return f'/teste/Doador/{self.codigo}/'

def get\_modifica(self):

return f'/teste/altera\_doador/{self.codigo}/'

def get\_exclui(self):

return f'/teste/deleta\_Doador/{self.codigo}/'

def \_\_str\_\_(self):

return self.nome

**5- Criando um menu**

Como um extra e para facilitar o codigo, vamos criar um menu, para isso precisamos de dois htmls , e uma nova view, com isso criamos o seguinte html de nome index.html, dentro de APP/templates :

{% load static %}

{% load bootstrap %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="">

<meta name="author" content="">

<link rel="shortcut icon" href="{% static 'img/cart.png' %}">

{% block title %}

<title>sistema doacao</title>

{% endblock title %}

<!-- Bootstrap Core CSS -->

<link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">

<!-- Custom CSS -->

<link href="{% static 'css/rgscale.css' %}" rel="stylesheet">

<!-- Custom Fonts -->

<link href="{% static 'font-awesome/css/font-awesome.min.css' %}" rel="stylesheet" type="text/css">

<link href="http://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesheet" type="text/css">

<link href="http://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">

</head>

<body id="page-top" data-spy="scroll" data-target=".navbar-fixed-top">

<!-- MENU -->

{% include "menu\_landpage.html" %}

<!-- Intro Header -->

<header class="intro">

<div class="intro-body">

<div class="container" id="icones-container">

<div class="row">

<div class="col-md-8 col-md-offset-2">

<a href="{% url 'Doador\_list' %}" class="btn btn-circle" id="cadastro" title="Cadastro"><i class="glyphicon glyphicon-user"></i></a>

<a href="#" class="btn btn-circle" id="daocao" title="Doacao"><i class="glyphicon glyphicon-tint"></i></a>

</div>

</div>

<img src="{% static './img/Entrada SIG-2000.jpg' %}" class="pull-left" id="img-entrada">

</div>

</div>

</header>

<!-- Footer -->

<footer >

<div class="container text-center">

</div>

</footer>

<!-- jQuery -->

<script src="{% static 'js/jquery.js' %}"></script>

<!-- Bootstrap Core JavaScript -->

<script src="{% static 'js/bootstrap.min.js' %}"></script>

<!-- Plugin JavaScript -->

<script src="{% static 'js/jquery.easing.min.js' %}"></script>

<!-- Google Maps API Key - Use your own API key to enable the map feature. More information on the Google Maps API can be found at https://developers.google.com/maps/ -->

<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCRngKslUGJTlibkQ3FkfTxj3Xss1UlZDA&sensor=false"></script>

<!-- Custom Theme JavaScript -->

<script src="{% static 'js/grayscale.js' %}"></script>

</body>

</html>

E o outro html é esse : <!-- Navigation -->

<nav class="navbar navbar-custom navbar-fixed-top" role="navigation">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-main-collapse">

<i class="fa fa-bars"></i>

</button>

<a class="navbar-brand page-scroll" href="/home">

<i class="fa fa-play-circle"></i> <span class="light">Sistema\_doacao</span>

</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse navbar-main-collapse">

<ul class="nav navbar-nav navbar-right">

<!-- <li>

<a class="page-scroll" href="#about">Config</a>

</li> -->

<li>

<a data-toggle="modal" data-target="#ModalSobreSistema" title="Clique para saber mais"><span class="glyphicon glyphicon-exclamation-sign"></span> Sobre</a>

</li>

<li>

<a class="page-scroll" href="#contact"><span class="glyphicon glyphicon-comment"></span> Contato</a>

</li>

<!-- <li>

<a class="page-scroll" href="#contact">Ajuda</a>

</li> -->

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container -->

</nav>

E para finalizar criamos uma pequena linha na APP/views.py para instanciarmos esse html, e depois fazemos sua url

home = TemplateView.as\_view(template\_name='index.html')

Depois disso , modificamos a APP/urls.py:

urlpatterns = [

path('',home,name="home"),

path('Lista\_Doador/',Doador\_list,name='Doador\_list'),

path('adcionar\_doador/',insertdoador,name='insertdoador'),

path('<int:pk>/',view=detalhe\_formulario,name='detalhe\_formulario'),

path('Doador/<int:pk>/',view=detalhe\_doador,name='detalhe\_doador'), path('deleta\_Doador/<int:pk>/',view=deleta\_doador,name='deleta\_doador'),

path('altera\_doador/<int:pk>/',view=altera\_doador,name='altera\_doador'),

]

Agora é só rodar esse comendo o cmd:

**python manage.py runserver 8022**

E usar a html

127.0.0.1:8022/teste/

E ir testando o codigo