**1 criando os models**

primeiro deve se criar um models que vai ser o nosso objeto, antes de tudo, dentro da pasta APP, crie um file chamado models.py , logo após crie a class Teste(models.Model), colocando os seguintes atributos :

models.py

import uuid

import re

#from stdimage.models import StdImageField

from django.utils import timezone

from pathlib import Path, os

from django.db.models import signals

from django.core.exceptions import ValidationError

from django.db import models

class Teste(models.Model):

options\_choices = (

('1', 'option 1 '),

('F', 'option 2 '),

)

CHOICES = (

('1', 'Opção 1'),

('2', 'Opção 2'),

('3', 'Opção 3'),

('4', 'Opção 4'),

)

id = models.AutoField("id",primary\_key=True, null=False)

texto = models.TextField("texto",null = True)

interio = models.IntegerField("inteiro",blank=True)

Boolean = models.BooleanField("booelan",blank=True,null=True ,default=False)

lista = models.CharField("escolha", choices = options\_choices)

escolha\_radio = models.CharField("escolharadio",null=True,max\_length=30)

def get\_detalhe(self):

return f'/teste/{self.id}'

**2 Criando o Forms**

Depois de Criar a classe Teste, devemos instanciar um Forms, que serve justamente para facilitar o transporte de dados da parte front para a parte back-end, antes de tudo, dentro da pasta “APP” deve se criar um novo File.py chamado forms.py , e depois disso dentro dele coloque o seguinte codigo :

from django import forms

from .models import \*

#from rest\_framework import serializers

class TesteForm(forms.ModelForm):

class Meta:

model = Teste

fields = ('texto','interio','Boolean','lista','escolha\_radio')

widgets = {

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

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

}

field\_order = ('texto','interio','Boolean','lista','escolha\_radio')

Esse Codigo instancia quais partes devem aparecer no Front sem termos que colocar-las manualmente na parte html

**3 criando um novo metodo**

Agora vamos criar um novo metodoChamado inserttest para conseguir pegar todas as variaveis dentro da classe Teste

def inserttest(request):

order\_forms = Teste()

if request.method == 'POST':

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

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

# ESCOLHA = request.POST.get("opcaoRadio",False)

#print(ESCOLHA)

# order\_forms.escolha\_radio =str(ESCOLHA)

# print(order\_forms.escolha\_radio)

if forms.is\_valid() :

teste\_instance = forms.save()

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

else:

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

context = {

'forms': forms,

}

return render(request,'teste2.html',context)

VIEWS.py

**4 - Criando os htmls**

Antes de criar os principais htmls, vamos criar uma html que servirá de base para todas as paginas htmls, que serviria somente para a parte estetica, devo pedir que dentro da Pasta App e colocar a pasta static que vai estar dentro do nosso github, so precisa abrir nosso codigo, e procurar trabalho/APP/static , depois crie dentro de templates Base.html

{% load static %}

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">

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

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

<title>{% block title %}{% endblock title %}</title>

<!-- Font Awesome -->

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

<!-- Bootstrap core CSS -->

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

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

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

{% block css %}{% endblock %}

<!-- Bootstrap core JavaScript -->

<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>

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

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

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

{% block js %}{% endblock %}

</head>

<body>

<!-- MENU -->

<!-- CABEÇALHO -->

<div class="container-cabec">

<div class="row-cabec">

</div>

<!--<div class="col" id="img-cliente">

<img src="{{ request.user.user\_profile.setor\_usu.fundo\_cabec\_imagem\_usu.cabec.url }}">

</div>-->

<div class="col" id="subtitulo-pagina">

<h1>{% block subtitle %}{% endblock subtitle %}</h1>

</div>

<div class="col" id="img-cecotein">

<p id="numero-tela" title="Código do sistema - Número da tela">{% block numeroPagina %}{% endblock %}</p>

</div>

</div>

</div>

<div class="container">

<!-- MENSAGENS -->

<!-- CORPO -->

{% block content %}{% endblock content %}

<script>

if (!window.Global) {

window.Global = [];

}

Global.angular\_dependencies = {% block angulardependencies %}[

]{% endblock angulardependencies %};

</script>

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

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

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

</body>

</html>

Depois de criar toda a base.html. Agora vamos criar uma nova html chamada teste2.hml, e coloque esse codigo nela, ele deve colocar todo o Forms em uma indentação automatica

{% 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 }}

<label>Opções do Radio:</label><br>

</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 %}

teste2.html

**5 - Criando a parte de detalhamento dos dados**

Agora vamos fazer um novo metodo que vai ler os dados e imprimi-los direto em uma nova pagina, para isso devemos criar um novo metodo dentro de views.py, apra o file views.py, e crie o codigo def detalhe\_formulario(request,pk):

ef detalhe\_formulario(request, pk):

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

context = {

'object':object,

}

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

Ele vai pegar os dados do objeto Teste, e colocar-los diretamente na tela .

**6 - criar um html para detalhe**

Depois crie um novo html chamado detalhe\_formulario.html, e coloque o seguinte codigo:

{% load bootstrap %}

{% 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">Texto</th>

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

</tr>

<tr>

<tr>

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

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

</tr>

<tr>

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

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

</tr>

<tr>

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

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

</tr>

<tr>

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

<td>{{object.escolha\_radio}}/</td>

</tr>

<tr>

</tr>

</tbody>

</table>

{%endblock%}

detalhe\_formulario.html

**7- URL**

Agora voltamos para a parte url.py, devemos adcionar duas novas partes , o path(‘conc2/’,inserttest,name=’formu’), e o path(‘<int:pk>/’,view =detalhe\_formulario,name=’detalhe\_formulario),

depois disso rode os seguintes comandos no terminal do codigo

python manage.py makemigrations APP

python manage.py migrate APP

lembre-se que isso deve ser rodado antes de dar o runserver

urlpatterns = [

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

path('conc2/',inserttest,name='formu'),

path('conc/',teste2,name='teste2'),

path('mar6/',mar6,name='mar6'),

path('men6/',men6,name='men6'),

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

]

URL.py