# AULA 01

## Primeiro é necessário a criação do ambiente virtual, por isso vamos usar.

python -m venv venv

## Após isso é necessários ativar o ambiente virtual para instalarmos as bibliotecas necessárias.

#LINUX

source venv/bin/activate

#WINDOWS

.\venv\Scripts\\activate

OU

.\venv\Scripts\\activate.bat

#OU

venv\\Scripts\\Activate.ps1

## Com o ambiente ativo só nos falta instalar as bibliotecas.

pip install django pillow

## Agora precisamos criar a estrutura base do projeto Django, ou seja devemos utilizar.

django-admin startproject plataforma\_cursos .

## Vamos configurar o nosso arquivo [settings.py](http://settings.py)

"""

Django settings for plataforma\_cursos project.

Generated by 'django-admin startproject' using Django 3.2.6.

For more information on this file, see

<https://docs.djangoproject.com/en/3.2/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/3.2/ref/settings/>

"""

from pathlib import Path

import os

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See <https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/>

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-cl=$h%uvrso#=v-y)6ussue@g9w\*\_b^^5\*w!jts4!&@4oa4y5i'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'plataforma\_cursos.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR, 'templates')],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'plataforma\_cursos.wsgi.application'

# Database

# <https://docs.djangoproject.com/en/3.2/ref/settings/#databases>

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# <https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators>

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# <https://docs.djangoproject.com/en/3.2/topics/i18n/>

LANGUAGE\_CODE = 'pt-BR'

TIME\_ZONE = 'America/Sao\_Paulo'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# <https://docs.djangoproject.com/en/3.2/howto/static-files/>

STATIC\_URL = '/static/'

STATICFILES\_DIRS = (os.path.join(BASE\_DIR, 'templates/static'),)

STATIC\_ROOT = os.path.join('static')

MEDIA\_ROOT = os.path.join(BASE\_DIR,'media')

MEDIA\_URL = '/media/'

# Default primary key field type

# <https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field>

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

## Agora vamos criar o nosso app usuários, responsável pela autenticação.

python3 manage.py startapp usuarios

## Agora vamos definir uma URL para o nosso app.

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('auth/', include('usuarios.urls')),

]

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

## O próximo passo é criar as URL dentro de usuário.

from django.urls import path

from . import views

urlpatterns = [

path('cadastro/', views.cadastro, name = 'cadastro'),

path('login/', views.login, name = 'login'),

]

## Criando as respectivas funções temos:

from django.http.response import Http404

from django.shortcuts import render

from django.http import HttpResponse

def cadastro(request):

return HttpResponse('cadastro')

def login(request):

return HttpResponse('login')

## Mas agora precisamos renderizar templates HTML.

## Vamos então criar um template base.

<!doctype html>

<html lang="pt-BR">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="<https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css>">

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

{% block 'head' %}{% endblock %}

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">&nbsp&nbspPythonando</a>

</nav>

<script src="<https://code.jquery.com/jquery-3.5.1.slim.min.js>"></script>

<script src="<https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js>" ></script>

{% block 'body'%}{% endblock %}

</body>

</html>

## Feito isso podemos criar nosso HTML de cadastro

{% extends 'base.html'%}

{% block 'titulo'%}

Cadastro

{% endblock%}

{% block 'body' %}

<div class="container">

<div class="row">

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

</div>

<div class="col-md">

<br>

<h1>Cadastre-se</h1>

<hr>

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

<label>Digite seu nome:</label>

<input class="form-control" type="text" placeholder="Nome..." name = "nome">

<br>

<label>Digite seu Email:</label>

<input class="form-control" type="text" placeholder="E-mail" name = "email">

<br>

<label>Digite seu senha:</label>

<input class="form-control" type="password" placeholder="Senha" name = "senha">

<br>

<input class="btn btn-success btn-lg" type="submit">

<a href="#" class="btn btn-info btn-lg">Logar</a>

</form>

</div>

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

</div>

</div>

</div>

{% endblock %}

## Agora só precisamos alterar a view

def cadastro(request):

return render(request, 'cadastro.html')

## Já estamos renderizando uma página HTML para o cadastro, vamos repetir o processo para a página de login.

{% extends 'base.html'%}

{% block 'titulo'%}

Logar

{% endblock%}

{% block 'body' %}

<div class="container">

<div class="row">

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

</div>

<div class="col-md">

<br>

<h1>Login</h1>

<hr>

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

<label>Digite seu Email:</label>

<input class="form-control" type="text" placeholder="E-mail" name = "email">

<br>

<label>Digite seu senha:</label>

<input class="form-control" type="password" placeholder="Senha" name = "senha">

<br>

<input class="btn btn-success btn-lg" type="submit">

<a href="#" class="btn btn-info btn-lg">Cadastre-se</a>

</form>

</div>

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

</div>

</div>

</div>

{% endblock %}

def login(request):

return render(request, 'login.html')

Com o front-end pronto vamos preparar o banco de dados.

from django.db import models

class Usuario(models.Model):

nome = models.CharField(max\_length = 50)

email = models.EmailField()

senha = models.CharField(max\_length = 64)

def \_\_str\_\_(self) -> str:

return self.nome

python3 manage.py makemigrations

python3 manage.py migrate

## Feito isso temos nossa tabela no banco de dados. Vamos cadastra-la na área administrativa agora

from django.contrib import admin

from .models import Usuario

admin.site.register(Usuario)

Vamos então criar um superuser

python3 manage.py createsuperuser

Aprimorando a área administrativa.

from django.contrib import admin

from django.contrib.admin.decorators import display

from .models import Usuario

@admin.register(Usuario)

class UsuarioAdmin(admin.ModelAdmin):

list\_display = ('nome', 'email', 'senha')

search\_fields = ('nome', 'email')

readonly\_fields = ('senha',)

## Envie os dados do cadastro.html para uma determinada URL e insira o token de segurança.

<form method="POST" action="{% url 'valida\_cadastro'%}">{% csrf\_token %}

Agora precisamos criar essa URL.

path('valida\_cadastro/', views.valida\_cadastro, name = 'valida\_cadastro'),

## Criando a view:

def valida\_cadastro(request):

nome = request.POST.get('nome')

email = request.POST.get('email')

senha = request.POST.get('senha')

usuario = Usuario.objects.filter(email = email)

if len(usuario) > 0:

return redirect('/auth/cadastro/?status=1')

if len(nome.strip()) == 0 or len(email.strip()) == 0:

return redirect('/auth/cadastro/?status=2')

if len(senha) < 8:

return redirect('/auth/cadastro/?status=3')

try:

senha = hashlib.sha256(senha.encode()).hexdigest()

usuario = Usuario(nome = nome,

email = email,

senha = senha)

usuario.save()

return redirect('/auth/cadastro/?status=0')

except:

return redirect('/auth/cadastro/?status=4')

## Prepare a view para receber o parametro enviado pela URL.

def cadastro(request):

status = request.GET.get('status')

return render(request, 'cadastro.html', {'status': status})

## E agora é só renderizar as mensagens no HTML.

{% if status == '1' %}

<div class="alert alert-danger" role="alert">

E-mail já cadastrado no sistema

</div>

{% endif %}

{% if status == '2' %}

<div class="alert alert-danger" role="alert">

Nome e Email não podem ser vazios

</div>

{% endif %}

{% if status == '3' %}

<div class="alert alert-danger" role="alert">

Senha deve conter no mínimo 8 caracteres

</div>

{% endif %}

{% if status == '4' %}

<div class="alert alert-danger" role="alert">

Erro interno do sistema

</div>

{% endif %}

{% if status == '0' %}

<div class="alert alert-success" role="alert">

Cadastro confirmado com sucesso

</div>

{% endif %}

## E assim terminamos nossa primeira Aula da PyStack Week.

# AULA 02

## Nessa aula nós vamos criar o sistema de login, cadastro, alteração e exclusão de aulas.

## Primeiro vamos direcionar os dados de login.html para uma URL:

<form method="POST" action="{% url 'valida\_login'%}">{% csrf\_token %}

## Vamos criar a respectiva URL.

path('valida\_login/', views.valida\_login, name = 'valida\_login'),

## Criando a função valida\_login.

def valida\_login(request):

email = request.POST.get('email')

senha = request.POST.get('senha')

senha = hashlib.sha256(senha.encode()).hexdigest()

usuarios = Usuario.objects.filter(email = email).filter(senha = senha)

if len(usuarios) == 0:

return redirect('/auth/login/?status=1')

elif len(usuarios) > 0:

request.session['usuario'] = usuarios[0].id

return redirect('/home/')

## Prepare a view para receber o parâmetro enviado por GET.

def login(request):

status = request.GET.get('status')

return render(request, 'login.html', {'status': status})

## Não vamos esquecer dos avisos em login.html.

{% if status == '1' %}

<div class="alert alert-danger" role="alert">

E-mail ou senha inválido

</div>

{% endif %}

{% if status == '2' %}

<div class="alert alert-warning" role="alert">

Faça login antes de acessar o sistema!

</div>

{% endif %}

## Com o login realizado vamos impedir o usuário de acessar login e cadastro se já estiver logado, para isso basta acrescentar nas views:

if request.session.get('usuario'):

return redirect('/home/')

## Vamos criar uma URL para o logout.

path('sair/', views.sair, name = 'sair'),

Crie a view.

def sair(request):

request.session.flush()

return redirect('/auth/login/')

## Agora precisamos criar a área do aluno, para isso vamos criar um novo app chamado cursos.

python3 manage.py startapp cursos

Vamos criar uma URL que aponta para esse novo app.

path('home/', include('cursos.urls')),

## Agora vamos criar as nossas tabelas para salvarmos as aulas e cursos.

from django.db import models

class Cursos(models.Model):

nome = models.CharField(max\_length = 100)

descricao = models.TextField()

thumb = models.ImageField(upload\_to = "thumb\_cursos")

def \_\_str\_\_(self) -> str:

return self.nome

class Aulas(models.Model):

nome = models.CharField(max\_length = 100)

descricao = models.TextField()

aula = models.FileField(upload\_to = "aulas")

curso = models.ForeignKey(Cursos, on\_delete = models.DO\_NOTHING)

def \_\_str\_\_(self) -> str:

return self.nome

python3 manage.py makemigrations

python3 manage.py migrate

## Vamos registrar na área administrativa.

from django.contrib import admin

from .models import Cursos, Aulas

admin.site.register(Aulas)

admin.site.register(Cursos)

Crie a URL de cursos.

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name = 'home'),

]

## Crie a view home.

def home(request):

if request.session.get('usuario'):

cursos = Cursos.objects.all()

request\_usuario = request.session.get('usuario')

return render(request, 'home.html', {'cursos': cursos, 'request\_usuario': request\_usuario})

else:

return redirect('/auth/login/?status=2')

## Agora é só criar o HTML para listar os cursos para o aluno.

{% extends 'base.html'%}

{% block 'titulo'%}

Home

{% endblock%}

{% block 'body' %}

<div class="container">

<br>

<h1>Nossos cursos</h1>

<div class="row">

{% for i in cursos %}

<div style="margin-top: 20px;" class="col-md">

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="{{i.thumb.url}}" alt="Card image cap">

<div class="card-body">

<h5 class="card-title">{{i.nome}}</h5>

<p class="card-text">{{i.descricao}}</p>

<a href="#" class="btn btn-primary">Acessar</a>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

{% endblock %}

## Vamos então criar a URL para visualizar as aulas de cada curso.

path('curso/<int:id>', views.curso, name = 'curso'),

## Crie a função curso.

def curso(request, id):

if request.session.get('usuario'):

aulas = Aulas.objects.filter(curso = id)

request\_usuario = request.session.get('usuario')

return render(request, 'curso.html', {'aulas': aulas, 'request\_usuario': request\_usuario})

else:

return redirect('/auth/login/?status=2')

## Crie o curso.html.

{% extends 'base.html'%}

{% block 'titulo'%}

Curso

{% endblock%}

{% block 'body' %}

<div class="container">

<br>

<h1>Aulas do curso</h1>

<div class="row">

{% for i in aulas %}

<div style="margin-top: 20px;" class="col-md">

<div class="card" style="width: 18rem;">

<video controls>

<source src="{{i.aula.url}}" type="video/mp4">

Your browser does not support the video tag.

</video>

<div class="card-body">

<h5 class="card-title">{{i.nome}}</h5>

<p class="card-text">{{i.descricao}}</p>

<a href="#" class="btn btn-primary">Acessar</a>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

{% endblock %}

## Adicione o link para curso.html no botão

<a href="{% url 'curso' i.id%}" class="btn btn-primary">Acessar</a>

## Ainda falta a URL para visualizar a aula.

path('aula/<int:id>', views.aula, name = 'aula'),

## A view:

def aula(request, id):

if request.session.get('usuario'):

aula = Aulas.objects.get(id = id)

return render(request, 'aula.html', {'aula': aula})

else:

return redirect('/auth/login/?status=2')

## Adicione o link da aula:

<a href="{% url 'aula' i.id %}" class="btn btn-primary">Acessar</a>

## O HTML

{% extends 'base.html'%}

{% block 'titulo'%}

Aula

{% endblock%}

{% block 'body' %}

<div class="container">

<br>

<h1>{{aula.nome}}</h1>

<hr>

<video width="100%" controls>

<source src="{{aula.aula.url}}" type="video/mp4">

</video>

<hr>

<h1>Faça seu comentário: </h1>

<input id="comentario" type="text" placeholder="Comentário..." class="form-control">

<br>

<input onclick="#" type="button" value="Enviar" class="btn btn-success btn-lg">

<hr>

</div>

{% endblock %}

E assim terminamos nossa segunda aula.

# AULA 3

## Nessa aula vamos criar as funcionalidades de comentários e avaliação das aulas.

## Crie a tabela comentários no banco de dados.

class Comentarios(models.Model):

usuario = models.ForeignKey(Usuario, on\_delete = models.DO\_NOTHING)

comentario = models.TextField()

data = models.DateTimeField(default = datetime.now)

aula = models.ForeignKey(Aulas, on\_delete = models.DO\_NOTHING)

def \_\_str\_\_(self) -> str:

return self.usuario.nome

## Cadastre na área administrativa.

admin.site.register(Comentarios)

## Crie uma URL comentários.

path('comentarios/', views.comentarios, name = 'comentarios'),

## Crie a view comentarios.

def comentarios(request):

usuario\_id = int(request.POST.get('usuario\_id'))

comentario = request.POST.get('comentario')

aula\_id = int(request.POST.get('aula\_id'))

comentario\_instancia = Comentarios(usuario\_id = usuario\_id,

comentario = comentario,

aula\_id = aula\_id)

comentario\_instancia.save()

comentarios = Comentarios.objects.filter(aula = aula\_id).order\_by('-data')

somente\_nomes = [i.usuario.nome for i in comentarios]

somente\_comentarios = [i.comentario for i in comentarios]

comentarios = list(zip(somente\_nomes, somente\_comentarios))

return HttpResponse(json.dumps({'status': '1', 'comentarios': comentarios }))

em Aula.html:

{% extends 'base.html'%}

{% block 'titulo'%}

Aula

{% endblock%}

{% block 'body' %}

<div class="container">

<br>

<br>

<h1>{{aula.nome}}</h1>

<hr>

<video width="100%" controls>

<source src="{{aula.aula.url}}" type="video/mp4">

</video>

<hr>

<h1>Faça seu comentário: </h1>

<input id="comentario" type="text" placeholder="Comentário..." class="form-control">

<br>

<input onclick="envia\_comentario()" type="button" value="Enviar" class="btn btn-success btn-lg">

<hr>

<div id="comentarios">

{% for i in comentarios%}

<div style="background-color: red; width: 30px; height: 30px; border-radius: 15px; display: inline-block;"></div>

<h5 style="display: inline;">{{i.usuario}}</h5>

<p style="margin-left: 50px;">{{i.comentario}}</p>

<hr>

{% endfor %}

</div>

</div>

<script src="<https://code.jquery.com/jquery-3.4.1.js>"></script>

<script>

function envia\_comentario(){

comentario = document.getElementById('comentario').value

document.getElementById('comentario').value = ""

$.ajax({

url: "{% url 'comentarios'%}",

method: 'post',

data: {

'csrfmiddlewaretoken': '{{ csrf\_token }}',

'aula\_id': "{{aula.id}}",

'usuario\_id': "{{usuario\_id}}",

'comentario': comentario

},

success: function(resposta){

resposta = JSON.parse(resposta)

if(resposta['status'] == "1"){

div\_comentario = document.getElementById('comentarios')

div\_comentario.innerHTML = ""

for(i=0; i < resposta['comentarios'].length; i++){

console.log('teste')

div\_comentario.innerHTML += '<div style="background-color: red; width: 30px; height: 30px; border-radius: 15px; display: inline-block;"></div>'

div\_comentario.innerHTML += '<h5 style="display: inline;">'+ resposta['comentarios'][i][0] +'</h5>'

div\_comentario.innerHTML += '<p style="margin-left: 50px;">' + resposta['comentarios'][i][1] + '</p>'

div\_comentario.innerHTML += '<hr>'

}

}

}

})

}

</script>

{% endblock %}

## Agora vamos desenvolver o sistema de avaliação da aula.

## Crie a tabela NotaAulas no banco de dados.

class NotasAulas(models.Model):

choices = (

('p', 'Péssimo'),

('r', 'Ruim'),

('re', 'Regular'),

('b', 'bom'),

('o', 'Ótimo')

)

aula = models.ForeignKey(Aulas, on\_delete=models.DO\_NOTHING)

nota = models.CharField(max\_length=50, choices=choices)

usuario = models.ForeignKey(Usuario, on\_delete=models.DO\_NOTHING)

## Registre na área administrativa.

admin.site.register(NotasAulas)

Crie a seguinte URL.

path('processa\_avaliacao/', views.processa\_avaliacao, name = 'processa\_avaliacao')

## Não se esqueça da view.

def processa\_avaliacao(request):

if request.session.get('usuario'):

avaliacao = request.POST.get('avaliacao')

aula\_id = request.POST.get('aula\_id')

usuario\_id = request.session.get('usuario')

usuario\_avaliou = NotasAulas.objects.filter(aula\_id = aula\_id).filter(usuario\_id = usuario\_id)

if not usuario\_avaliou:

nota\_aulas = NotasAulas(aula\_id = aula\_id,

nota = avaliacao,

usuario\_id = usuario\_id,

)

nota\_aulas.save()

return redirect(f'/home/aula/{aula\_id}')

else:

return redirect('/auth/login/')

else:

return redirect('/auth/login/')

## Deixa e view aula dessa forma:

def aula(request, id):

if request.session.get('usuario'):

aula = Aulas.objects.get(id = id)

usuario\_id = request.session['usuario']

comentarios = Comentarios.objects.filter(aula = aula).order\_by('-data')

request\_usuario = request.session.get('usuario')

usuario\_avaliou = NotasAulas.objects.filter(aula\_id = id).filter(usuario\_id = request\_usuario)

avaliacoes = NotasAulas.objects.filter(aula\_id = id)

return render(request, 'aula.html', {'aula': aula,

'usuario\_id': usuario\_id,

'comentarios': comentarios,

'request\_usuario': request\_usuario,

'usuario\_avaliou': usuario\_avaliou,

'avaliacoes': avaliacoes})

else:

return redirect('/auth/login/?status=2')

## Agora é só preparar o front-end para enviar os dados.

## Acrescente um botão para o usuário avaliar a aula.

{% if not usuario\_avaliou %}

<div class="alert alert-primary" role="alert">

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">

CLIQUE AQUI!

</button> para avaliar essa aula

</div>

{% else %}

{% endif %}

## Adicione o modal para o usuário avaliar.

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Avalie</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<form action="{% url 'processa\_avaliacao' %}" method="POST">

{% csrf\_token %}

<input type="hidden" value="{{aula.id}}" name="aula\_id">

<select name="avaliacao" class="form-control">

<option value="p">Péssimo</option>

<option value="r">Ruim</option>

<option value="re" selected>Regular</option>

<option value="b">Bom</option>

<option value="o">Ótimo</option>

</select>

<br>

<input type="submit" class="btn btn-success" value="Avaliar">

</form>

</div>

</div>

</div>

</div>

## Agora é só exibir as avaliações.

<div id="avaliacoes">

{% for i in avaliacoes%}

<h5 style="display: inline;">

{{i.usuario}}

{% if i.nota == 'p'%}

{% for i in i.nota|return\_list:1 %}

<svg style="color: green;" xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">

<path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327 4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8 13.187l-4.389 2.256z"/>

</svg>

{% endfor %}

{% for i in i.nota|return\_list:4 %}

<svg xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">

<path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.95l-4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.12l-4.898.696c-.441.062-.612.636-.283.95l3.523 3.356-.83 4.73zm4.905-2.767-3.686 1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.745l4.052-.576a.525.525 0 0 0 .393-.288L8 2.223l1.847 3.658a.525.525 0 0 0 .393.288l4.052.575-2.906 2.77a.565.565 0 0 0-.163.506l.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>

</svg>

{% endfor %}

{% endif %}

{% if i.nota == 'r' %}

{% for i in i.nota|return\_list:2 %}

<svg style="color: green;" xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">

<path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327 4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8 13.187l-4.389 2.256z"/>

</svg>

{% endfor %}

{% for i in i.nota|return\_list:3 %}

<svg xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">

<path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.95l-4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.12l-4.898.696c-.441.062-.612.636-.283.95l3.523 3.356-.83 4.73zm4.905-2.767-3.686 1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.745l4.052-.576a.525.525 0 0 0 .393-.288L8 2.223l1.847 3.658a.525.525 0 0 0 .393.288l4.052.575-2.906 2.77a.565.565 0 0 0-.163.506l.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>

</svg>

{% endfor %}

{% endif %}

{% if i.nota == 're'%}

{% for i in i.nota|return\_list:3 %}

<svg style="color: green;" xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">

<path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327 4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8 13.187l-4.389 2.256z"/>

</svg>

{% endfor %}

{% for i in i.nota|return\_list:2 %}

<svg xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">

<path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.95l-4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.12l-4.898.696c-.441.062-.612.636-.283.95l3.523 3.356-.83 4.73zm4.905-2.767-3.686 1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.745l4.052-.576a.525.525 0 0 0 .393-.288L8 2.223l1.847 3.658a.525.525 0 0 0 .393.288l4.052.575-2.906 2.77a.565.565 0 0 0-.163.506l.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>

</svg>

{% endfor %}

{% endif %}

{% if i.nota == 'b'%}

{% for i in i.nota|return\_list:4 %}

<svg style="color: green;" xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">

<path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327 4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8 13.187l-4.389 2.256z"/>

</svg>

{% endfor %}

{% for i in i.nota|return\_list:1 %}

<svg xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">

<path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.95l-4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.12l-4.898.696c-.441.062-.612.636-.283.95l3.523 3.356-.83 4.73zm4.905-2.767-3.686 1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.745l4.052-.576a.525.525 0 0 0 .393-.288L8 2.223l1.847 3.658a.525.525 0 0 0 .393.288l4.052.575-2.906 2.77a.565.565 0 0 0-.163.506l.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>

</svg>

{% endfor %}

{% endif %}

{% if i.nota == 'o'%}

{% for i in i.nota|return\_list:5 %}

<svg style="color: green;" xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">

<path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173 6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327 4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8 13.187l-4.389 2.256z"/>

</svg>

{% endfor %}

{% for i in i.nota|return\_list:0 %}

<svg xmlns="<http://www.w3.org/2000/svg>" width="25" height="25" fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">

<path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.95l-4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.12l-4.898.696c-.441.062-.612.636-.283.95l3.523 3.356-.83 4.73zm4.905-2.767-3.686 1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.745l4.052-.576a.525.525 0 0 0 .393-.288L8 2.223l1.847 3.658a.525.525 0 0 0 .393.288l4.052.575-2.906 2.77a.565.565 0 0 0-.163.506l.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>

</svg>

{% endfor %}

{% endif %}

</h5>

<br>

<br>

{% endfor %}

<hr>

</div>

## Para finalizar vamos criar o nosso filtro return\_list

## dentro da pasta templatetags crie um arquivo chamado [filtros.py](http://filtros.py) e insira:

from django import template

register = template.Library()

@register.filter

def return\_list(value, arg):

return [i for i in range(int(arg))]

## Agora é só carregar o filtro em aula.html

{% load filtros %}