1. **Criar novo projeto**

**FILE > NEW PROJECT >**

**Location: D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2**

**VENV location:**

**D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2\venv**

1. **Instalar os pacotes e dependências**

**(Venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install django**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install whitenoise**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install gunicorn**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install django-bootstrap4**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install django-stdimage**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install MySQL**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip freeze**

1. **Gerar o arquivo requirements.txt**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip freeze > requirements.txt**

*asgiref==3.4.1*

*beautifulsoup4==4.10.0*

*Django==3.2.7*

*django-bootstrap4==3.0.1*

*django-stdimage==5.3.0*

*gunicorn==20.1.0*

*Pillow==8.3.2*

*pytz==2021.1*

*soupsieve==2.2.1*

*sqlparse==0.4.2*

*whitenoise==5.3.0*

1. **Criar o Projeto**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>django-admin startproject django2 .**

1. **Criar a aplicação principal**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>django-admin startapp core**

1. **Configuração inicial do arquivo: django2/settings.py**

# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = *True*ALLOWED\_HOSTS = ['\*']

INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'core',  
 'bootstrap4',  
 'stdimage',  
]

MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 #'whitenoise.middleware.WhiteNoiseMiddleware',  
 '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',  
]

**Comentar a linha do WhiteNoise, que somente será usado no deploy e publicação**

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': ['templates'],

# Database  
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'django2',  
 'USER': 'admin',  
 'PASSWORD': '1234',  
 'HOST': 'localhost',  
 'PORT': '3306',  
 }  
}

# Internationalization  
# https://docs.djangoproject.com/en/3.2/topics/i18n/  
  
LANGUAGE\_CODE = 'pt-br'  
  
TIME\_ZONE = 'America/Sao\_Paulo'

# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/3.2/howto/static-files/  
  
STATIC\_URL = '/static/'  
STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'django2',  
 'USER': 'root',  
 'PASSWORD': 'MySQLadmin00!',  
 'HOST': 'localhost',  
 'PORT': '3306',  
 }  
}

1. **Configuração inicial do arquivo: core/views.py**

*from* django.shortcuts *import* render  
  
*def* index(*request*):  
 *return* render(*request*, 'index.html')  
  
*def* contato(*request*):  
 *return* render(*request*, 'index.html')  
  
*def* produto(*request*):  
 *return* render(*request*, 'index.html')

1. **Criar o diretório ‘templates’ no app core e criar os arquivos index.html, contato.html e produto.html**

**<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Index</title>  
</head>  
<body>  
  
</body>  
</html>**

1. **Criar o diretório ‘static’ no app core; e no diretório criar as pastas para os arquivos: css, js e images**
2. **Editar o arquivo: django2/urls.py para adicionar a rota “ ” apontando para core.urls;**

*from* django.contrib *import* admin  
*from* django.urls *import* path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('core.urls')),  
]

1. **Criar o arquivo core/urls.py**

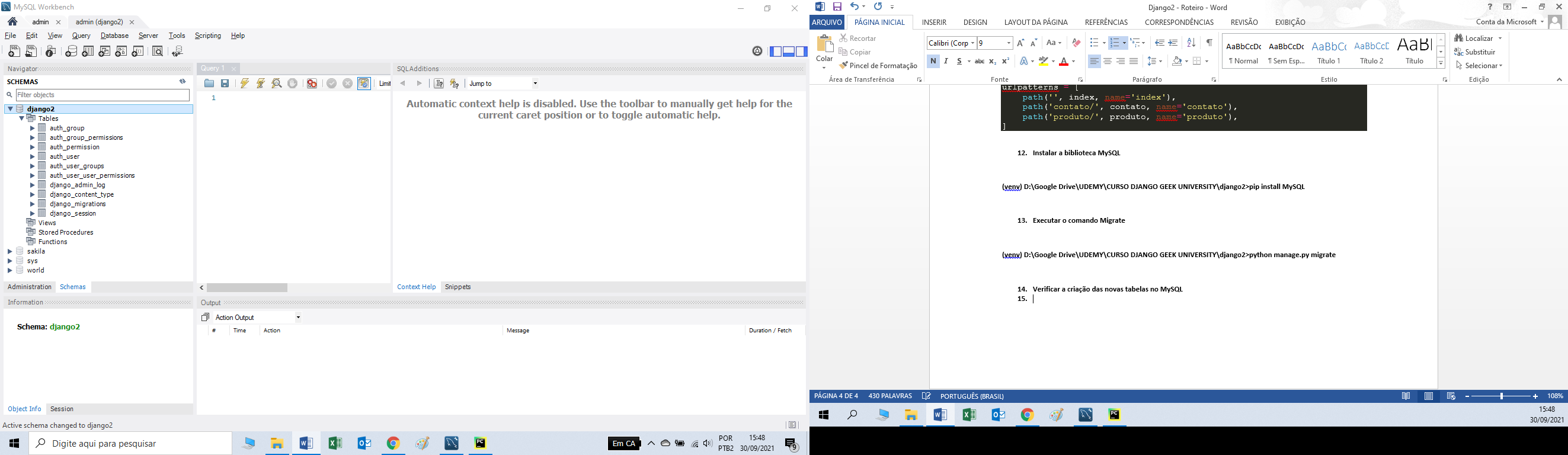
*from* django.urls *import* path  
*from* .views *import* index, contato, produto  
  
  
urlpatterns = [  
 path('', index, name='index'),  
 path('contato/', contato, name='contato'),  
 path('produto/', produto, name='produto'),  
]

1. **Instalar a biblioteca MySQL**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>pip install MySQL**

1. **Executar o comando Migrate**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py migrate**

1. **Verificar a criação das novas tabelas no MySQL**
2. **Criar o SUPERUSUARIO**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py createsuperuser**

**Usuário (leave blank to use 'olavo'): admin**

**Endereço de email: admin@admin.com.br**

**Password: 1234**

**Password (again): 1234**

**Esta senha é muito curta. Ela precisa conter pelo menos 8 caracteres.**

**Esta senha é muito comum.**

**Esta senha é inteiramente numérica.**

**Bypass password validation and create user anyway? [y/N]: y**

**Superuser created successfully.**

1. **Executar o servidor interno, para testar o projeto**

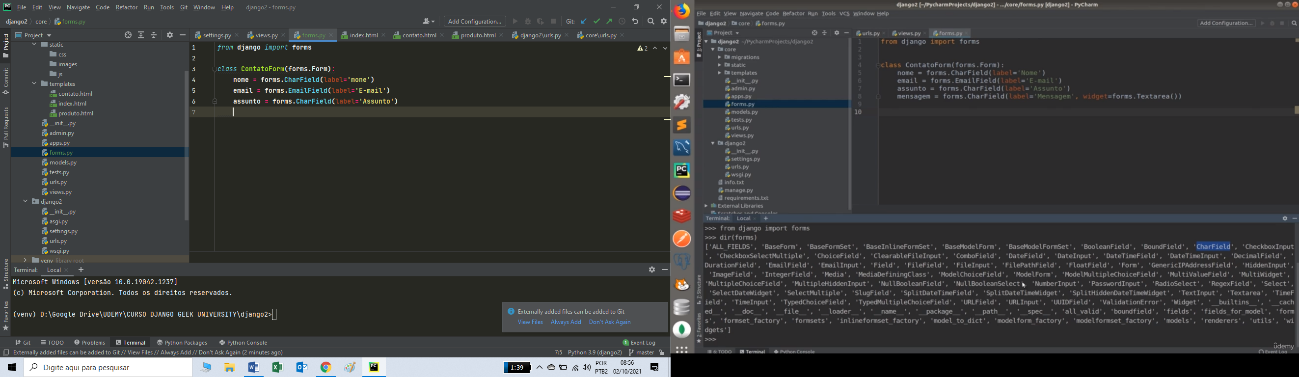
**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py runserver**

**\_\_\_\_\_\_\_\_\_\_\_+++++++++++++++\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_+++++++++++++++++\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aula 39**

1. **Criando formulários. Criar o arquivo forms.py no core**

*from* django *import* forms  
  
*class* ContatoForm(forms.Form):  
 nome = forms.CharField(label='None', max\_length=100)  
 email = forms.EmailField(label='E-mail', max\_length=100)  
 assunto = forms.CharField(label='Assunto', max\_length=100)  
 mensagem = forms.CharField(label='Mensagem', max\_length=100)



**Importar o forms.py em views.py:**

*from* django.shortcuts *import* render  
  
from .forms import ContatoForm  
  
*def* index(*request*):  
 *return* render(*request*, 'index.html')  
  
*def* contato(*request*):  
 form = ContatoForm()  
  
 context = {  
 'form': form  
 }  
 *return* render(*request*, 'contato.html', context)  
  
*def* produto(*request*):  
 *return* render(*request*, 'produto.html')

1. **Editar o contato.html; Incluir o Bootstrap**

{% load bootstrap4 %} **<!DOCTYPE html>  
<html lang="pt-br">  
<head>  
 <meta charset="UTF-8">  
 <title>Contato</title>** {% bootstrap\_css %}  
**</head>  
<body>  
 <h1>Contato</h1>**{% bootstrap\_javascript jquery='full' %}  
**</body>  
</html>**

1. **Editar o <body> do contato.html**

**<body>  
 <div class="container">  
 <h1>Contato</h1>  
 {% bootstrap\_messages %}  
  
 <form action="{% url 'contato' %}" method="post" class="form" autocomplete="off">  
 {% csrf\_token %}  
 {% bootstrap\_form form %}  
 {% buttons %}  
 <button type="submit" class="btn btn-primary">Enviar</button>  
 {% endbuttons %}  
 </form>  
 </div>   
{% bootstrap\_javascript jquery='full' %}  
</body>**

1. **Editar o views.py**

*from* django.shortcuts *import* render  
from django.contrib import messages  
  
*from* .forms *import* ContatoForm

*def* contato(*request*):  
 form = ContatoForm(request.POST or None)  
  
 if str(request.method) == 'POST':  
 if form.is\_valid():  
 nome = form.cleaned\_data['nome']  
 email = form.cleaned\_data['email']  
 assunto = form.cleaned\_data['assunto']  
 mensagem = form.cleaned\_data['mensagem']  
  
 print('Mensagem enviada')  
 print(f'Nome: {nome}')  
 print(f'E-mail: {email}')  
 print(f'Assunto: {assunto}')  
 print(f'Mensagem: {mensagem}')  
  
 messages.success(request, 'E-mail enviado com sucesso!')  
 form = ContatoForm()  
 else:  
 messages.error(request, 'Erro ao enviar o E-mail!')  
  
 context = {  
 'form': form  
 }  
 *return* render(*request*, 'contato.html', context)

1. **Aula 40 – Enviando E-mails**

#configurações de e-mail  
EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'  
  
"""  
no deploy:   
EMAIL\_HOST = 'localhost'  
EMAIL\_HOST\_USER = 'no-reply@kwpsolucoes.com.br'  
EMAIL\_PORT = 587  
EMAIL\_USER\_TSL = True  
EMAIL\_HOST\_PASSWORD = <dua senha>  
"""

**Adicionar a class ContatoForm, no forms.py:**

*from* django *import* forms  
from django.core.mail.message import EmailMessage  
  
*class* ContatoForm(forms.Form):  
 nome = forms.CharField(label='None', max\_length=100)  
 email = forms.EmailField(label='E-mail', max\_length=100)  
 assunto = forms.CharField(label='Assunto', max\_length=100)  
 mensagem = forms.CharField(label='Mensagem', max\_length=100)  
  
 def send\_mail(self):  
 nome = self.cleaned\_data['nome']  
 email = self.cleaned\_data['email']  
 assunto = self.cleaned\_data['assunto']  
 mensagem = self.cleaned\_data['assunto']  
  
 conteudo = f'Nome: {nome}\n E-Mail: {email}\nAssunto: {assunto}\nMensagem: {mensagem}'  
  
 mail = EmailMessage(  
 subject='E-mail enviado pelo sistema django2',  
 body=conteudo,  
 from\_email='contato@kwpsolucoes.com.br',

to=['contato@dominio.com.br','outro@email.com.br'],

headers={'Reply-To': email}  
 )  
 mail.send()

**[02/Oct/2021 11:06:38] "GET /contato/ HTTP/1.1" 200 2174**

**Content-Type: text/plain; charset="utf-8"**

**MIME-Version: 1.0**

**Content-Transfer-Encoding: 7bit**

**Subject: E-mail enviado pelo sistema django2**

**From: contato@kwpsolucoes.com.br**

**To: contato@dominio.com.br, outro@email.com.br**

**Date: Sat, 02 Oct 2021 14:06:47 -0000**

**Message-ID: <163318360722.21720.13552123501994252892@Note\_Olavo>**

**Reply-To: olavo.melo.eng@gmail.com**

**Nome: Olavo Melo**

**E-Mail: olavo.melo.eng@gmail.com**

**Assunto: cdfdd**

**Mensagem: cdfdd**

**-------------------------------------------------------------------------------**

**[02/Oct/2021 11:06:47] "POST /contato/ HTTP/1.1" 200 2178**

1. **Aula 41 – Edição do arquivo models.py / model Produto**

**Criação da classe produto**

*from* django.db *import* models  
*from* stdimage.models *import* StdImageField  
  
# SIGNALS  
  
*from* django.db.models *import* signals  
*from* django.template.defaultfilters *import* slugify  
  
*class* Base(models.Model):  
 criado = models.DateField('data de Criação', auto\_now\_add=*True*)  
 modificado = models.DateField('data de Atualização', auto\_now\_add=*True*)  
 ativo = models.BooleanField('ativo?', default=*True*)  
  
 *class* Meta:  
 abstract = *True  
  
class* Produto(Base):  
 nome = models.CharField('Nome', max\_length=100)  
 preco = models.DecimalField('Preco', max\_digits=8, decimal\_places=2)  
 estoque = models.IntegerField('Estoque')  
 imagem = StdImageField('Imagem', upload\_to='produtos', variations={'thumb': (124,124)})  
 slug = models.SlugField('Slug', max\_length=100, blank=*True*, editable=*False*)  
  
 *def \_\_str\_\_*(self):  
 *return* self.nome  
  
*def* produto\_pre\_save(signal, *instance*, sender, \*\*kwargs):  
 *instance*.slug = slugify(*instance*.none)  
  
signals.pre

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py makemigrations**

**Migrations for 'core':**

**core\migrations\0001\_initial.py**

**- Create model Produto**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py migrate**

**Operations to perform:**

**Apply all migrations: admin, auth, contenttypes, core, sessions**

**Running migrations:**

**Applying core.0001\_initial... OK**

**Editar o arquivo admin.py**

*from* django.contrib *import* admin  
  
*from* .models *import* Produto  
  
@admin.register(Produto)  
*class* ProdutoAdmin(admin.ModelAdmin):  
 list\_display = ('nome','preco','estoque','slug', 'criado', 'modificado', 'ativo')

**criar o super usuário**

**(venv) D:\Google Drive\UDEMY\CURSO DJANGO GEEK UNIVERSITY\django2>python manage.py createsuperuser**

**Usuário (leave blank to use 'olavo'): admin2**

**Endereço de email: admin2@uol.com.br**

**Password:**

**Password (again):**

**Esta senha é muito curta. Ela precisa conter pelo menos 8 caracteres.**

**Esta senha é muito comum.**

**Esta senha é inteiramente numérica.**

**Bypass password validation and create user anyway? [y/N]: y**

**Superuser created successfully.**

1. **AULA 42. CRIANDO UM FORMULÁRIO QUE ESCREVE NO BD**

**EDITAR O FORMS.PY:**

*from* .models *import* Produto

*class* ProdutoModelForm(forms.ModelForm):  
  
 *class* Meta:  
 model = Produto  
 fields = ['nome','preco','estoque','imagem']

**editar o views.py**

*from* .forms *import* ContatoForm, ProdutoModelForm

*def* produto(*request*):  
 *if* str(*request*.method) == 'POST':  
 form = ProdutoModelForm(*request*.POST, *request*.FILES)  
 *if* form.is\_valid():  
 prod = form.save(commit=*False*)  
  
 print(f'Nome: {prod.nome}')  
 print(f'Preço: {prod.preco}')  
 print(f'Estoque: {prod.estoque}')  
 print(f'Imagem: {prod.image}')  
  
 messages.success(*request*, 'Produto salvo com sucesso!')  
 form = ProdutoModelForm()  
 *else*:  
 messages.error(*request*,'Erro ao salvar o produto')  
 *else*:  
 form = ProdutoModelForm()  
 context = {  
 'form':form  
 }  
 *return* render(*request*, 'produto.html', context)

**editar o produto.html:**

**{% load bootstrap4 %}  
<!DOCTYPE html>  
<html lang="pt-br">  
<head>  
 <meta charset="UTF-8">  
 <title>Produto</title>  
 {% bootstrap\_css %}  
</head>  
<body>  
 <div class="container">  
 <h1>Produto</h1>  
 {% bootstrap\_messages %}  
  
 <form action="{% url 'produto' %}" method="post" class="form" autocomplete="off" enctype="multipart/form-data">  
 {% csrf\_token %}  
 {% bootstrap\_form form %}  
 {% buttons %}  
 <button type="submit" class="btn btn-primary">Cadastrar</button>  
 {% endbuttons %}  
 </form>  
 </div>  
{% bootstrap\_javascript jquery='full' %}  
</body>  
</html>**

**Runserver**

**Starting development server at http://127.0.0.1:8000/**

**Quit the server with CTRL-BREAK.**

**[02/Oct/2021 14:25:03] "GET / HTTP/1.1" 200 144**

**Not Found: /favicon.ico**

**[02/Oct/2021 14:25:03] "GET /favicon.ico HTTP/1.1" 404 2614**

**[02/Oct/2021 14:25:07] "GET /contato/ HTTP/1.1" 200 1951**

**[02/Oct/2021 14:25:11] "GET /produto/ HTTP/1.1" 200 1944**

**Nome: ps4.1**

**Preço: 1999**

**Estoque: 123**

**Imagem: ps4.jpg**

**[02/Oct/2021 14:25:25] "POST /produto/ HTTP/1.1" 200 2170**

1. **Aula 43... continuação salvar dados no BD**

**Editar o arquivo settings. Py**

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

**Editar o arquivo urls.py:**

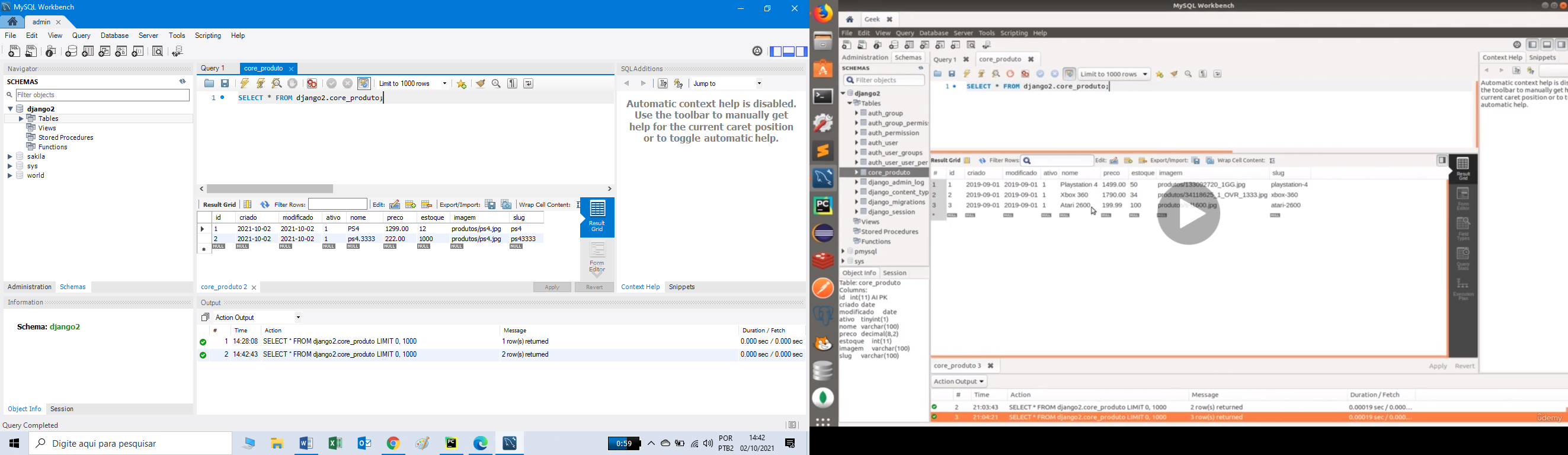
*from* django.contrib *import* admin  
*from* django.urls *import* path, include  
  
from django.conf.urls.static import static  
from django.conf import settings  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('core.urls')),  
] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**editar o views.py para inserir o comando form.save()**

*def* produto(*request*):  
 *if* str(*request*.method) == 'POST':  
 form = ProdutoModelForm(*request*.POST, *request*.FILES)  
 *if* form.is\_valid():  
 # prod = form.save(commit=False)  
  
 # print(f'Nome: {prod.nome}')  
 # print(f'Preço: {prod.preco}')  
 # print(f'Estoque: {prod.estoque}')  
 # print(f'Imagem: {prod.imagem}')  
 form.save()  
 messages.success(*request*, 'Produto salvo com sucesso!')  
 form = ProdutoModelForm()  
 *else*:

**>>> Deletar o arquivo produtos criado automaticamente no diretório raíz do projeto**

**Verificar o cadastro do produto no banco de dadosMySQL Workbench:**



**Verificar o diretório media criado e os aquivos de imagens cadastrados no BD**

1. **Aula 44 ... exibindo a lista de produtos no index.html**

**Editar o views.py**

*from .*models *import* Produto  
  
*def* index(*request*):  
 context = {  
 'produtos': Produto.objects.all()  
 }  
 *return* render(*request*, 'index.html', context)

**editar o index.html**

**{% load bootstrap4 %}  
<!DOCTYPE html>  
<html lang="pt-br">  
<head>  
 <meta charset="UTF-8">  
 <title>Index</title>  
 {% bootstrap\_css %}  
</head>  
<body>  
 <div class="container">  
 <h1>Index</h1>  
 <table class="table table-dark">  
 <thead>  
 <tr>  
 <th scope="col">#</th>  
 <th scope="col">Produto</th>  
 <th scope="col">Preço</th>  
 <th scope="col">Estoque</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for produto in produtos %}  
 <tr>  
 <td scope="row">{{ produto.id }}</td>  
 <td scope="row">{{ produto.nome }}</td>  
 <td scope="row">{{ produto.preco }}</td>  
 <td scope="row">{{ produto.estoque }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
{% bootstrap\_javascript jquery='full' %}  
</body>  
</html>**

**Inserir uma tela tipo Modal com a imagem do produto, inserir um link no id do produto para chamar a telamodal.**

**Editar o index.html**

**{% for produto in produtos %}  
 <tr>  
 <td scope="row">{{ produto.id }}</td>** <td scope="row"><a href="#modal{{produto.id}}" data-toggle="modal">{{ produto.nome }}</a></td>  
 **<td scope="row">{{ produto.preco }}</td>  
 <td scope="row">{{ produto.estoque }}</td>  
 </tr>  
{% endfor %}**

**<tbody>  
 {% for produto in produtos %}  
 <tr>  
 <td scope="row">{{ produto.id }}</td>  
 <td scope="row"><a href="#modal{{produto.id}}" data-toggle="modal">{{ produto.nome }}</a></td>  
 <td scope="row">{{ produto.preco }}</td>  
 <td scope="row">{{ produto.estoque }}</td>  
 </tr>** <div class="modal fade bd-example-modal-lg show" id="modal{{ produto.id }}" role="dialog">  
 <div class="modal-dialog">  
 <div class="modal-content">  
 <div class="modal-header">  
 <button type="button" class="close" data-dismiss="modal">  
 <span aria-hidden="true">&times;</span>  
 </button>  
 </div>  
 <div class="modal-body" id="dynamic-content">  
 <img src="{{ produto.imagem.url }}" class="img-fluid" alt="{{ produto.nome }}"/>   
 </div>  
 </div>  
 </div>  
 </div>  
 **{% endfor %}  
</tbody>**

**Deixar o LINK do item novamente na cor branca, editar o arquivo styles.css:**

a {  
 color: #fff;  
}

**alterar os arquivos html:**

**{% load bootstrap4 %}**{% load static %}  
**<!DOCTYPE html>**

**<head>  
 <meta charset="UTF-8">  
 <title>Index</title>  
 {% bootstrap\_css %}** <link href="{% static 'css/styles.css' %}" rel="stylesheet"> **</head>**

**Aula 45 – LOGIN secções de usuários**

from django.shortcuts import redirect

if str(request.user) != 'AnonymousUser':  
 *if* str(*request*.method) == 'POST':  
 form = ProdutoModelForm(*request*.POST, *request*.FILES)  
 *if* form.is\_valid():  
 # prod = form.save(commit=False)  
  
 # print(f'Nome: {prod.nome}')  
 # print(f'Preço: {prod.preco}')  
 # print(f'Estoque: {prod.estoque}')  
 # print(f'Imagem: {prod.imagem}')  
 form.save()  
 messages.success(*request*, 'Produto salvo com sucesso!')  
 form = ProdutoModelForm()  
 *else*:  
 messages.error(*request*,'Erro ao salvar o produto')  
 *else*:  
 form = ProdutoModelForm()  
 context = {  
 'form':form  
 }  
 *return* render(*request*,'produto.html',context)  
else:  
 return redirect('index')

**aula 46 – Deploy**