

Universidade Federal de Goiás
Instituto de Informática
Prof. Ronaldo Martins da Costa





Importando e Exportando Dados



- Algunas vezes é necessário trabalhar com arquivos txt. Vamos construir um exemplo para importar um txt para o DB

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <title>Página11.html</title>
  </head>

  <body>
    <h1>Upload / Import Arquivo!</h1>
    <p>bdpratico\exemplo01\templates\pagina11.html</p>
    <form method="post" enctype="multipart/form-data">
      {% csrf_token %}
      <input type="file" name="arq_upload">
      <button type="submit">Upload</button>
    </form>
  </body>

</html>
```



Importando e Exportando Dados



🕒 Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index_alias'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path('pagina4', views.pagina4, name='pagina4_alias'),
    path('pagina5', views.pagina5, name='pagina5_alias'),
    path('pagina6', views.pagina6, name='pagina6_alias'),
    path('pagina7', views.pagina7, name='pagina7_alias'),
    path('pagina8', views.pagina8, name='pagina8_alias'),
    path('pagina9', views.pagina9, name='pagina9_alias'),
    path('pagina10', views.pagina10, name='pagina10_alias'),
    path('pagina11', views.pagina11, name='pagina11_alias'),
    path('pessoa_menu', views.pessoa_menu.as_view(), name='pessoa_menu_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(), name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(), name='pessoa_delete_alias'),
```



Importando e **Exportando** Dados



- Criando um model no DB para receber os valores que serão importados

```
class exame(models.Model):  
    valor = models.FloatField(null=True, blank=True, default=None,  
                               verbose_name='Valor')  
  
    def __str__(self):  
        return self.valor
```



Importando e **Exportando** Dados



● Atualizando o modelo no DB

```
python manage.py makemigrations exemplo01
```

```
python manage.py migrate
```



Importando e **Exportando** Dados

- Se os ajustes no model foram realizados corretamente, você verá uma tela como esta

```
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro bdpratico % python manage.py makemigrations exemplo01
Migrations for 'exemplo01':
  exemplo01/migrations/0007_exame.py
    - Create model exame
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro bdpratico % python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, exemplo01, sessions
Running migrations:
  Applying exemplo01.0007_exame... OK
```





Importando e Exportando Dados



● Criando a view pagina11

```
def pagina11(request):
    from .models import exame

    import os
    from django.core.files.storage import FileSystemStorage

    if request.method == 'POST' and request.FILES['arq_upload']:
        fss = FileSystemStorage()
        upload = request.FILES['arq_upload']
        file1 = fss.save(upload.name, upload)
        file_url = fss.url(file1)

        print("upload", upload)
        print("file1", file1)
        print("file_url", file_url)

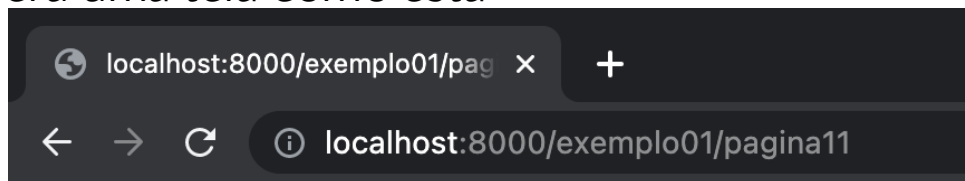
        file2 = open(file1, 'r')
        for row in file2:
            colunas = row.replace("(", "").replace(")", "").split(",")
            exame.objects.create(valor=float(colunas[8]))
        file2.close()
        os.remove(file_url.replace("/", ""))

    return HttpResponse("Arquivo Importado")
return render(request, 'pagina11.html')
```



Importando e **Exportando** Dados

- Se os ajustes foram realizados corretamente, após a execução da página11 você verá uma tela como esta



Arquivo Importado

e a tabela **exame** no DB está preenchida com os valores do arquivo txt fornecido





Plotando Gráficos



- Existem diversas API para construção de gráficos para o python/Django
- Vamos utilizar uma chamada **plotly**. A documentação desta API pode ser encontrada em

<https://plotly.com/graphing-libraries/>



Plotando Gráficos



● Instalando a plotly

```
pip install plotly
```



Plotando Gráficos



● Criando o template pagina12.html

```
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Página12.html</title>
  </head>
  <body>
    <h1>Plotando Gráficos</h1>
    <p>bdpratico\exemplo01\templates\pagina12.html</p>
    <div class="container">
      <div class="col-md-12 col-sm-12 col-xs-12">
        <center>
          <div>{{ grafico|safe }}</div>
        </center>
      </div>
    </div>
  </body>
</html>
```



Plotando Gráficos



🟡 Vamos criar uma nova rota em nosso arquivo urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index_alias'),
    path('pagina0', views.pagina0, name='pagina0_alias'),
    path('pagina1', views.pagina1, name='pagina1_alias'),
    path('pagina2', views.pagina2, name='pagina2_alias'),
    path('pagina3', views.pagina3, name='pagina3_alias'),
    path('pagina4', views.pagina4, name='pagina4_alias'),
    path('pagina5', views.pagina5, name='pagina5_alias'),
    path('pagina6', views.pagina6, name='pagina6_alias'),
    path('pagina7', views.pagina7, name='pagina7_alias'),
    path('pagina8', views.pagina8, name='pagina8_alias'),
    path('pagina9', views.pagina9, name='pagina9_alias'),
    path('pagina10', views.pagina10, name='pagina10_alias'),
    path('pagina11', views.pagina11, name='pagina11_alias'),
    path('pagina12', views.pagina12, name='pagina12_alias'),
    path('pessoa_menu', views.pessoa_menu.as_view(), name='pessoa_menu_alias'),
    path("pessoa_list/", views.pessoa_list.as_view(), name='pessoa_list_alias'),
    path("pessoa_create/", views.pessoa_create.as_view(), name='pessoa_create_alias'),
    path("pessoa_update/<int:pk>/", views.pessoa_update.as_view(), name='pessoa_update_alias'),
    path('pessoa_delete/<int:pk>/', views.pessoa_delete.as_view(), name='pessoa_delete_alias'),
```



Plotando Gráficos



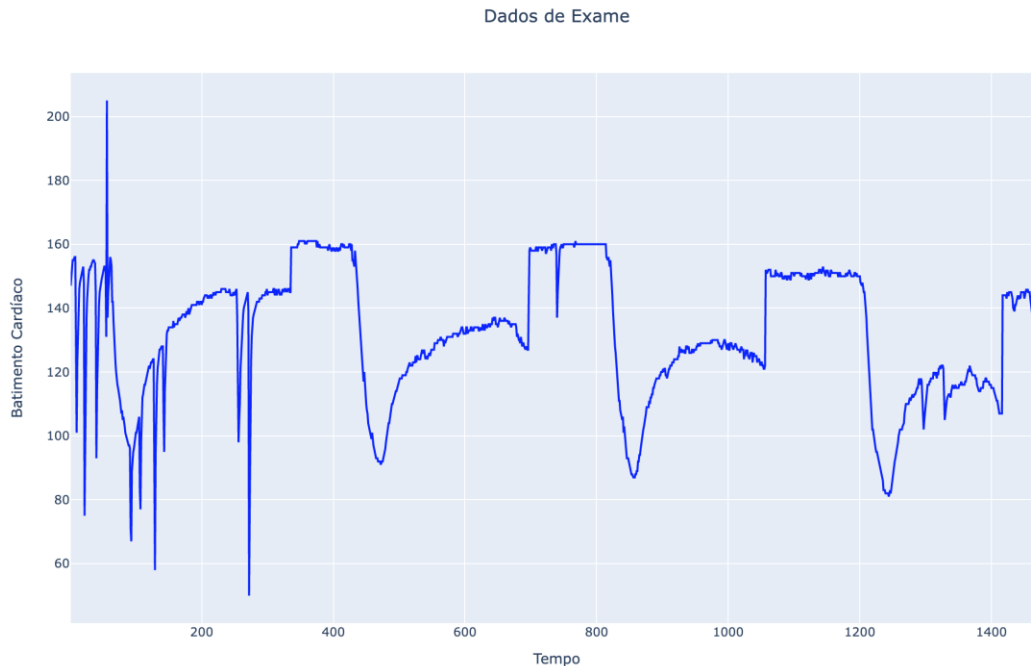
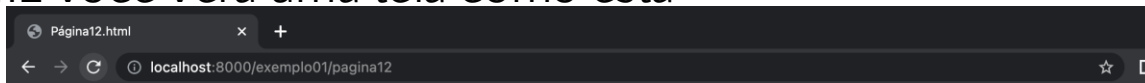
● Criando a view pagina12

```
def pagina12(request):
    from .models import exame
    import plotly.graph_objs as go
    from plotly.offline import plot
    exame_tmp = exame.objects.all()
    eixo_x = []
    eixo_y = []
    i = 0
    for e in exame_tmp:
        i += 1
        eixo_x.append(i)
        eixo_y.append(e.valor)
    figura = go.Figure()
    figura.add_trace(go.Scatter(x=eixo_x, y=eixo_y,
                                mode='lines',
                                line_color='rgb(0, 0, 255)'))
    figura.update_layout(title="Dados de Exame",
                          title_x=0.5,
                          xaxis_title='Tempo',
                          yaxis_title='Batimento Cardíaco')
    plot_div = plot(figura, output_type='div')
    dicionario = {}
    dicionario['grafico'] = plot_div
    return render(request, 'pagina12.html', dicionario)
```



Plotando Gráficos

- Se os ajustes foram realizados corretamente, após a execução da página12 você verá uma tela como esta

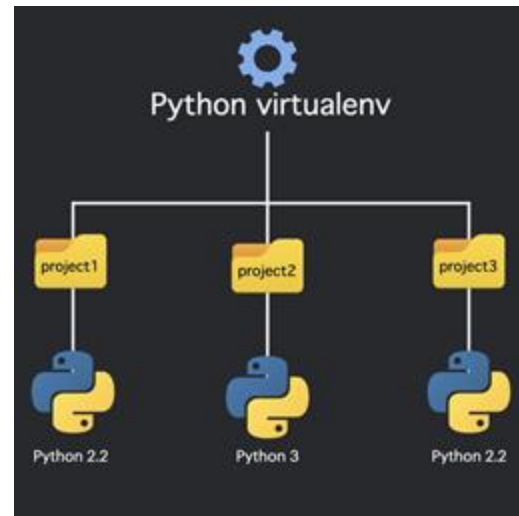




Ambiente Virtual

- Copiando o ambiente virtual criado para outra máquina
- O primeiro passo é gerar uma lista com todas as API's instaladas no ambiente virtual de origem

```
python -m pip freeze > requirements.txt
```





Ambiente Virtual

- A tela exibe o arquivo requirements.txt gerado

```
requirements.txt
1 asgiref==3.6.0
2 beautifulsoup4==4.11.1
3 Django==4.1.5
4 django-bootstrap-v5==1.0.11
5 django-tables2==2.5.1
6 numpy==1.24.1
7 pandas==1.5.3
8 plotly==5.13.0
9 python-dateutil==2.8.2
10 pytz==2022.7.1
11 six==1.16.0
12 soupsieve==2.3.2.post1
13 sqlparse==0.4.3
14 tenacity==8.2.0
15
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro BigData-env % ls
bdpratico  etc      lib      share
bin        include  pyvenv.cfg
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro BigData-env % python -m pip freeze > requirements.txt
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro BigData-env %
```





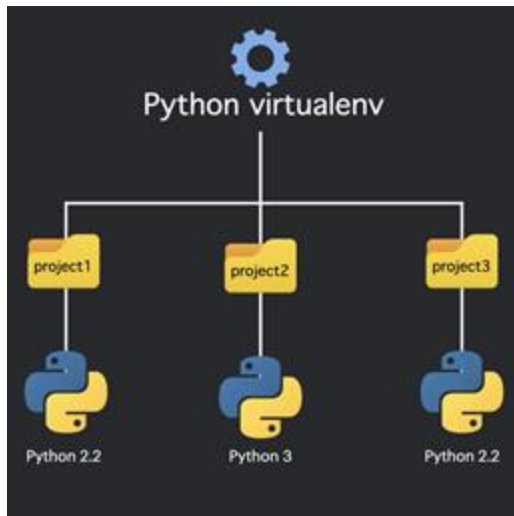
Ambiente Virtual

- O próximo passo é criar o ambiente virtual de destino na outra máquina

```
python -m venv BigData-Bak-env
```

- A seguir, copie o arquivo requirements.txt para dentro do ambiente virtual de destino
- Depois, com o ambiente virtual de destino “ativo”, execute o comando a seguir para instalar todas as API's

```
python -m pip install -r requirements.txt
```





Ambiente Virtual

- A tela exibe o resultado da cópia para a máquina destino

```
bin include lib
(BigData-Bak-env) MacBookAir2:BigData-Bak-env Eliana$ python -m pip install -r requirements.txt
Collecting asgiref==3.6.0
  Downloading asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting beautifulsoup4==4.11.1
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
Collecting Django==4.1.5
  Downloading Django-4.1.5-py3-none-any.whl (8.1 MB)
Collecting django-bootstrap-v5==1.0.11
  Downloading django_bootstrap_v5-1.0.11-py3-none-any.whl (24 kB)
Collecting django-tables2==2.5.1
  Downloading django_tables2-2.5.1-py2.py3-none-any.whl (94 kB)
Collecting numpy==1.24.1
  Downloading numpy-1.24.1.tar.gz (10.9 MB)
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing wheel metadata ... done
Collecting pandas==1.5.3
  Downloading pandas-1.5.3.tar.gz (5.2 MB)
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing wheel metadata ... done
Collecting plotly==5.13.0
  Downloading plotly-5.13.0-py2.py3-none-any.whl (15.2 MB)
Collecting python-dateutil==2.8.2
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting pytz==2022.7.1
  Downloading pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
Collecting six==1.16.0
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting soupsieve==2.3.2.post1
  Downloading soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Collecting sqlparse==0.4.3
  Downloading sqlparse-0.4.3-py3-none-any.whl (42 kB)
Collecting tenacity==8.2.0
  Downloading tenacity-8.2.0-py3-none-any.whl (24 kB)
Collecting backports.zoneinfo; python_version < "3.9"
  Downloading backports.zoneinfo-0.2.1.tar.gz (74 kB)
```



- Para finalizar, basta copiar a pasta bdpratico para dentro do ambiente virtual de destino que foi criado



Universidade Federal de Goiás
Instituto de Informática
Prof. Ronaldo Martins da Costa

