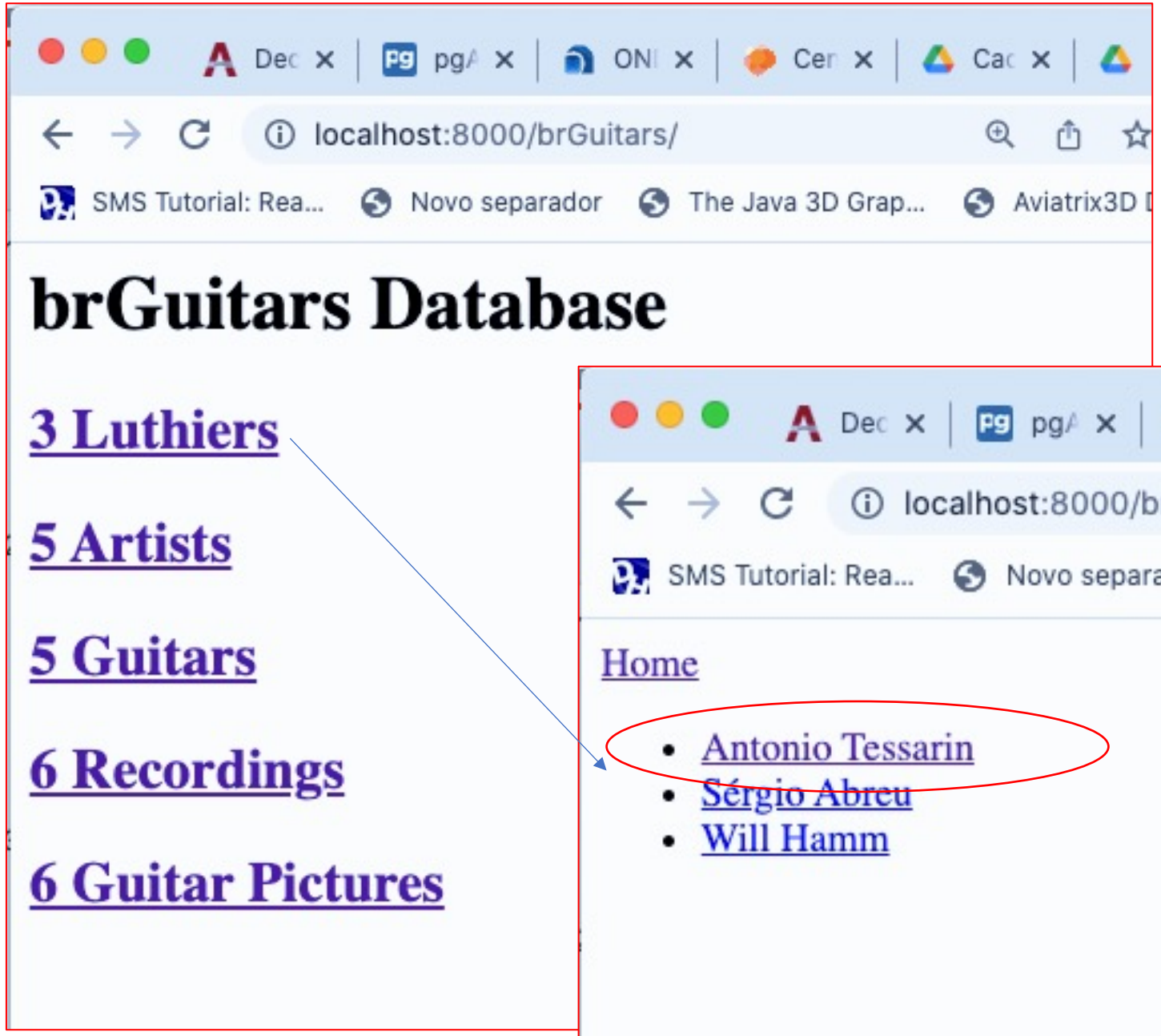
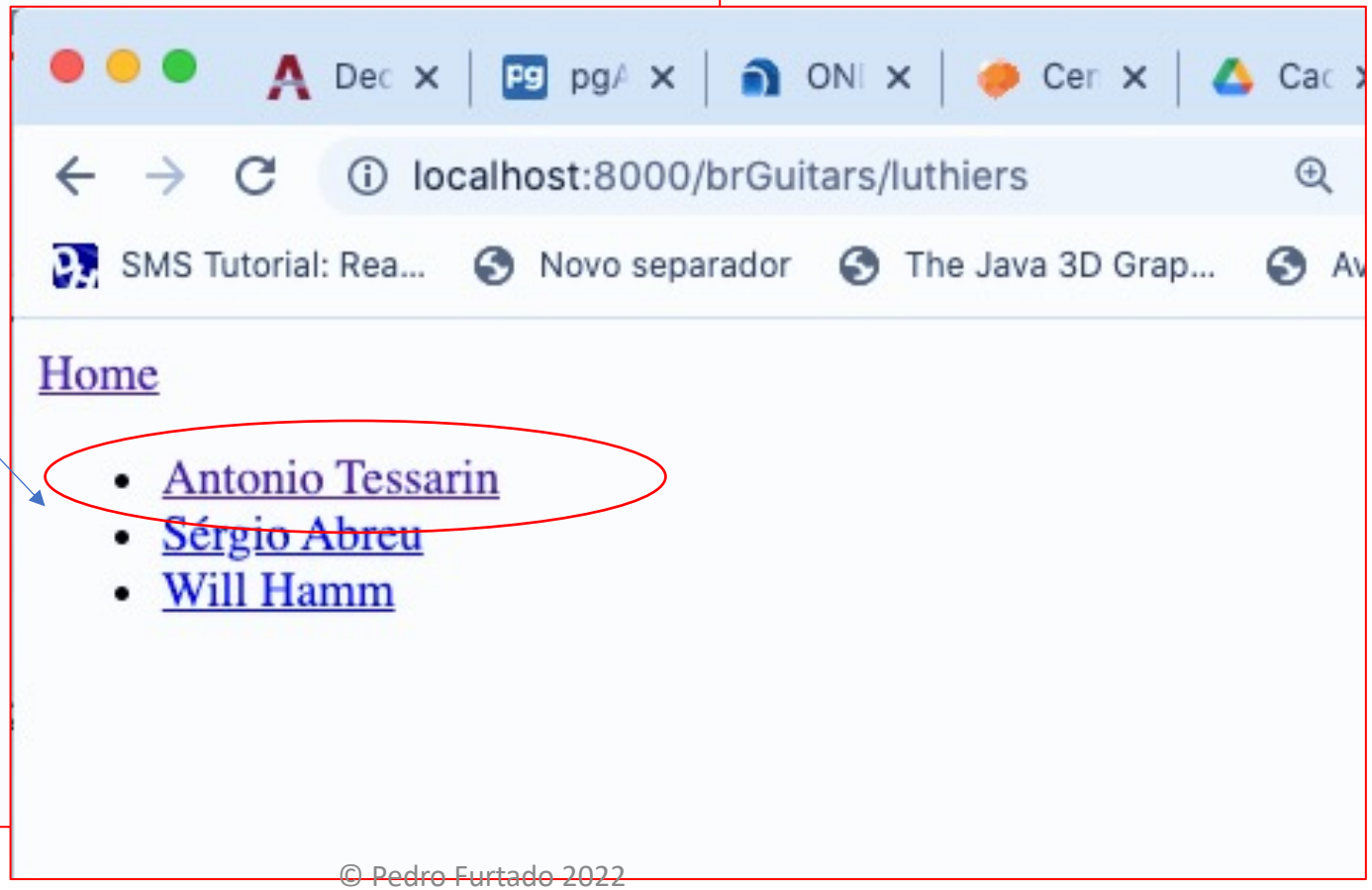
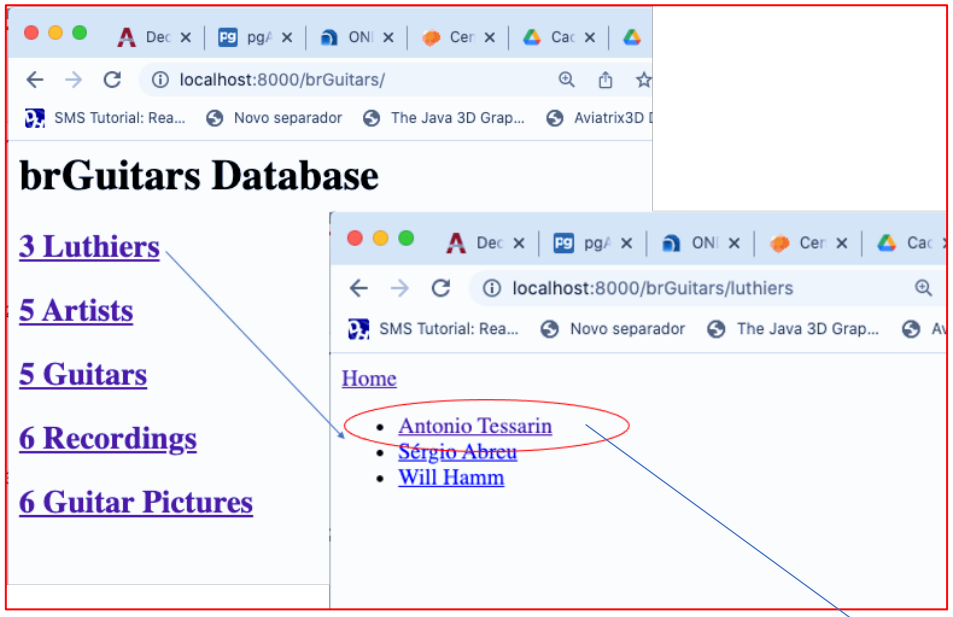
Django db programming

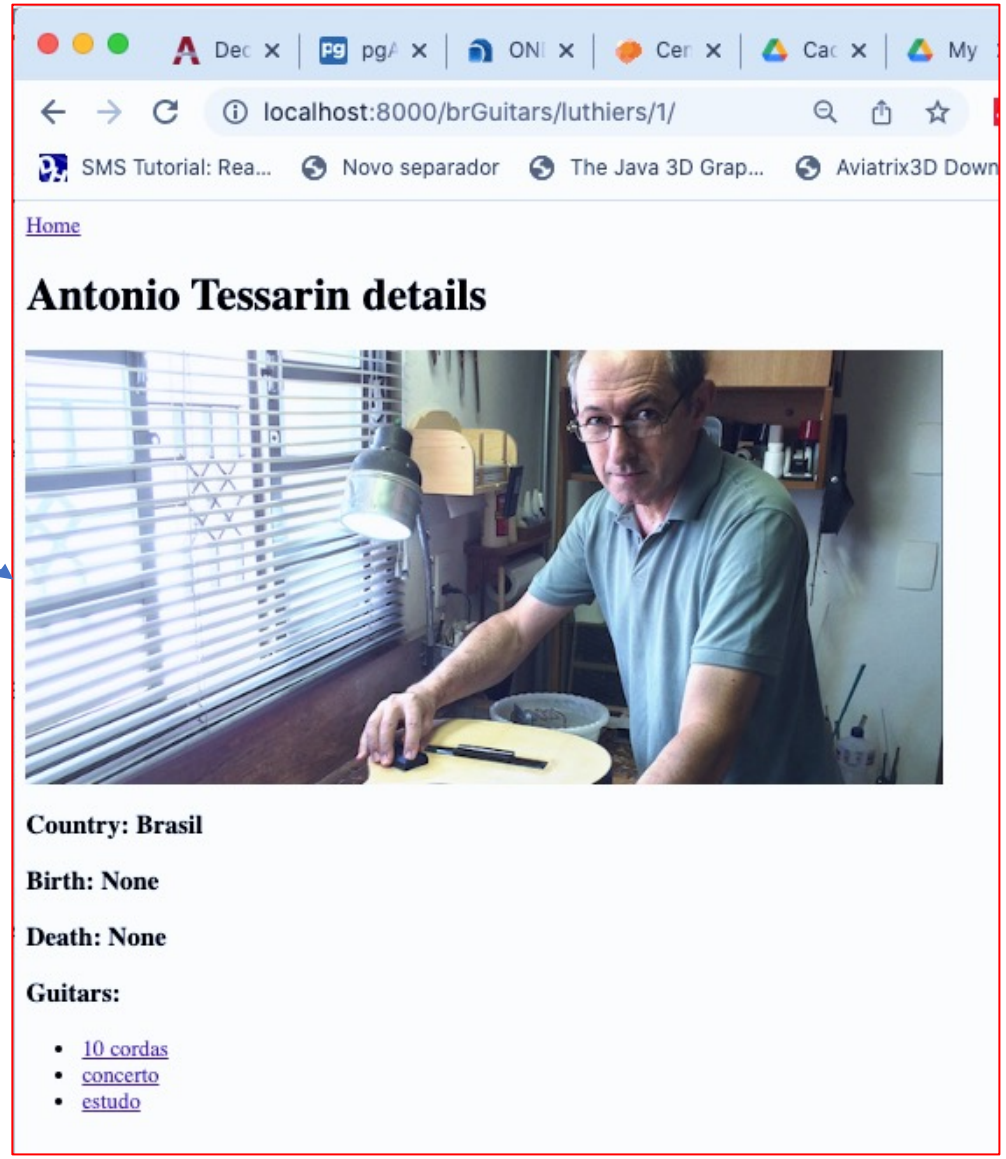
@Pedro Furtado 2022

@databases









localhost:8000/brGuitars/

brGuitars Database

3 Luthiers

5 Artists

5 Guitars

6 Recordings

6 Guitar Pictures

localhost:8000/brGuitars/guitars

Home

- Antonio Tessarin-10 cordas
- Antonio Tessarin-concerto
- Antonio Tessarin-estudo
- Sérgio Abreu-concerto
- Will Hamm-Yamandu Signature

localhost:8000/brGuitars/guitars/5/

Home

Antonio Tessarin-10 cordas

Year: None

top: abeto europeu

bottom: jacarandá bahia

neck: cedro brasileiro

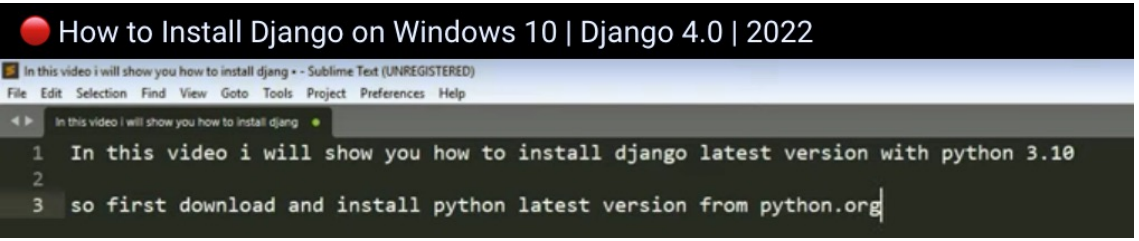
fretboard: ébano indiano

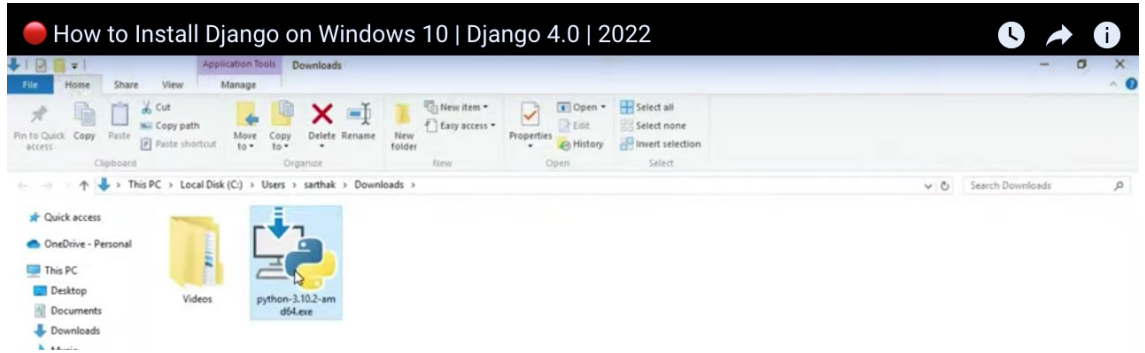
sadle: jacarandá bahia

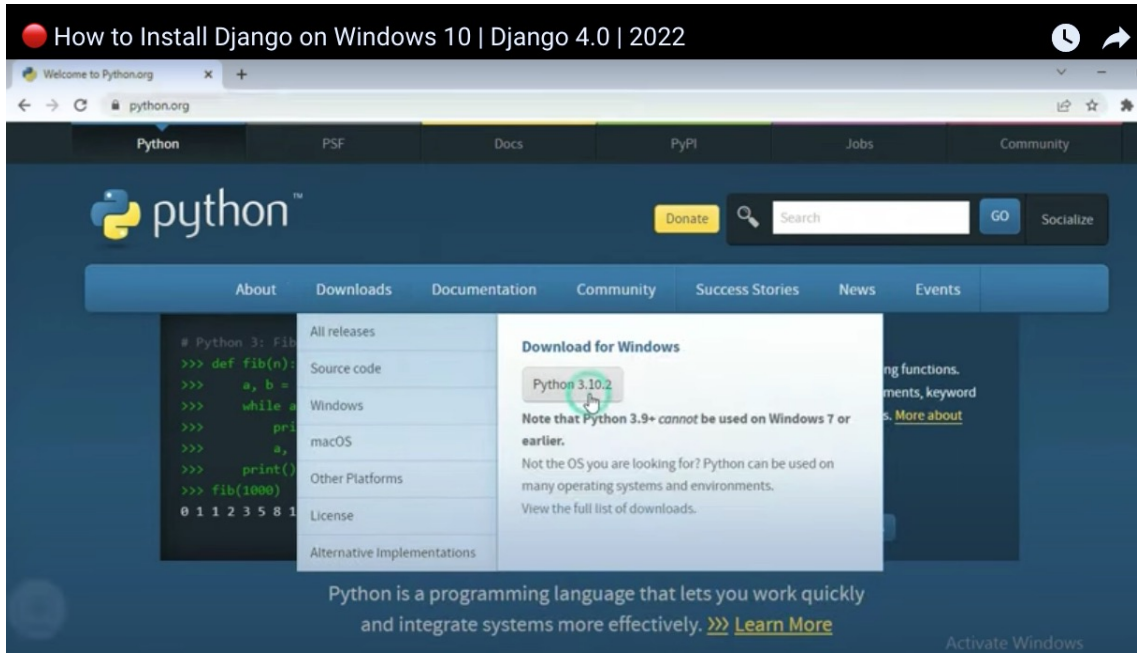
tuningMachines: Schaller

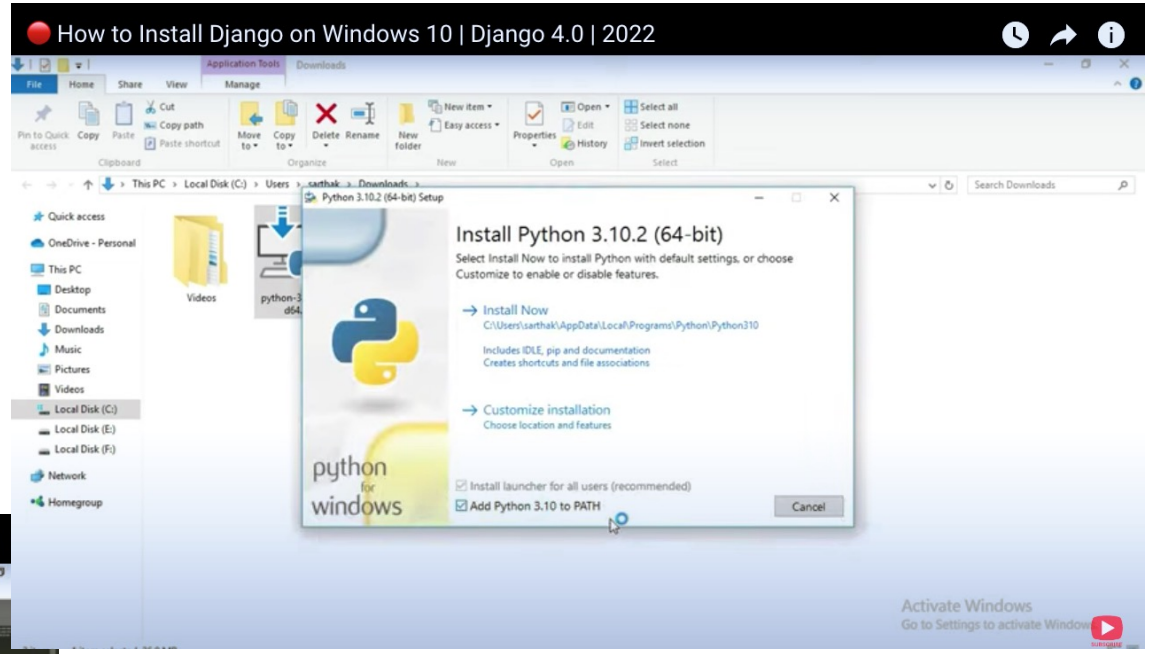
Main software parts

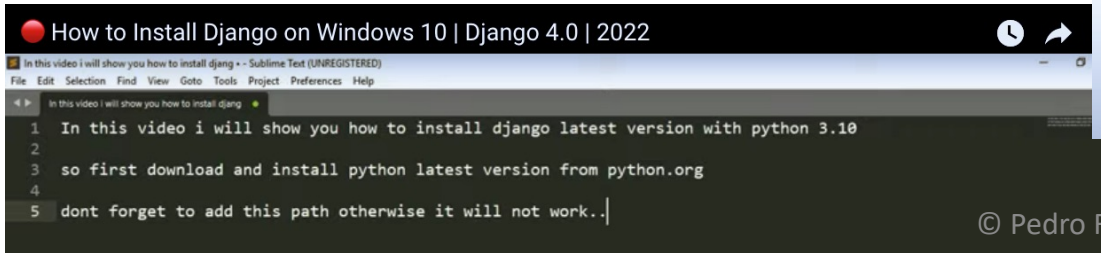
- Python = main programming language to be used
- Django = Framework programmed in python and configured to automate web-db programming

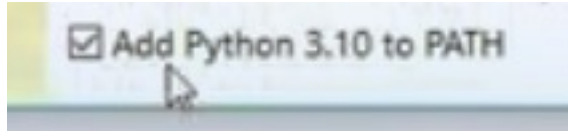


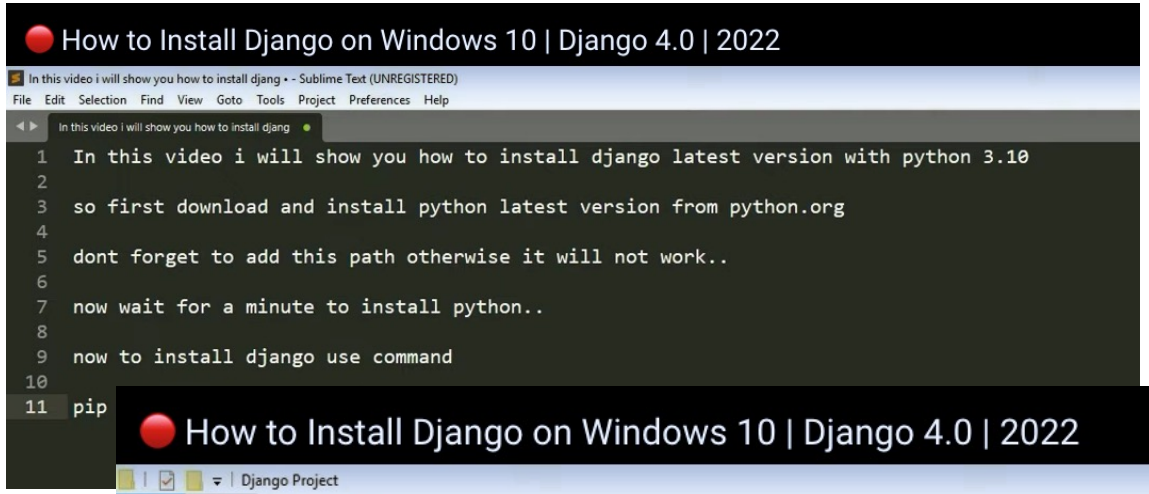


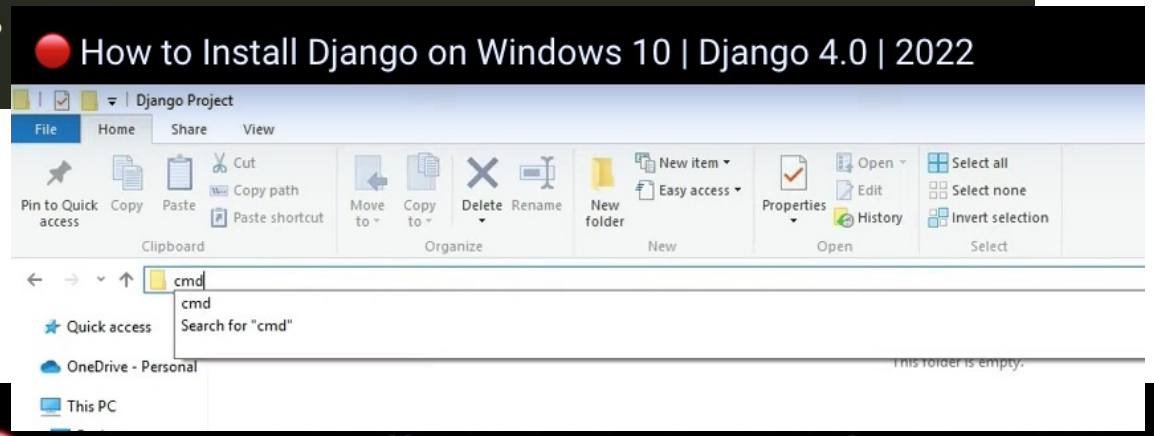


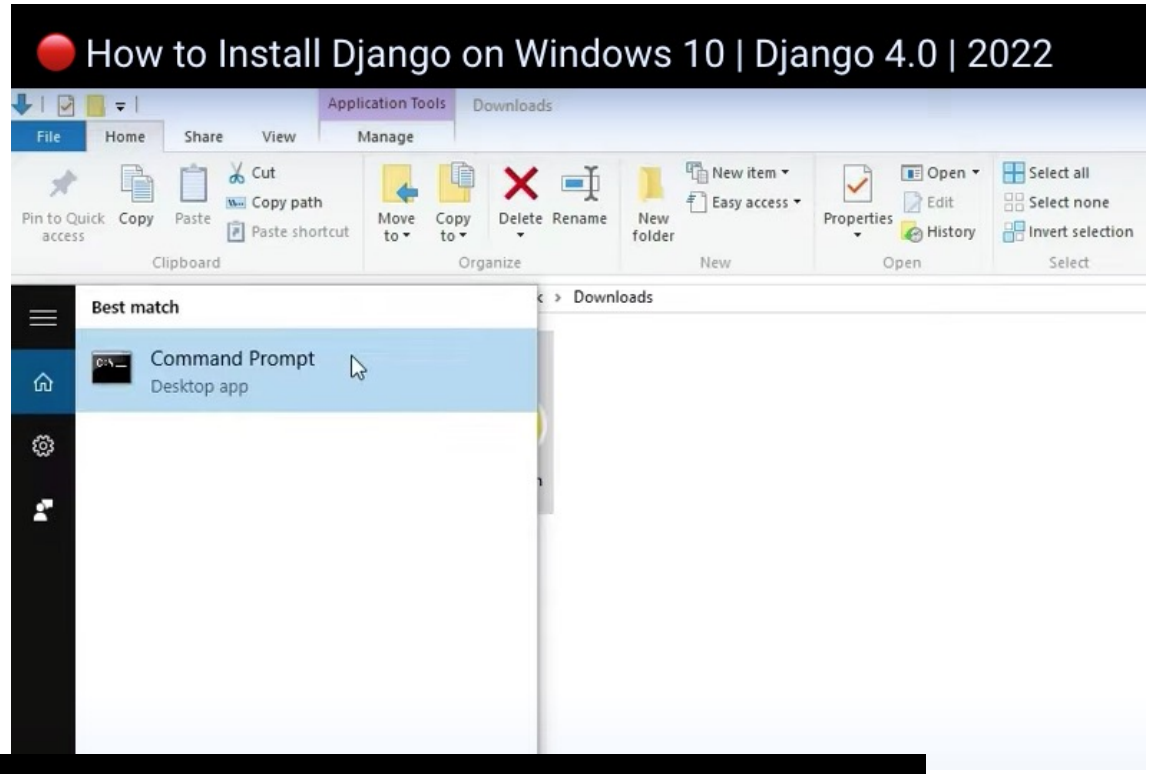


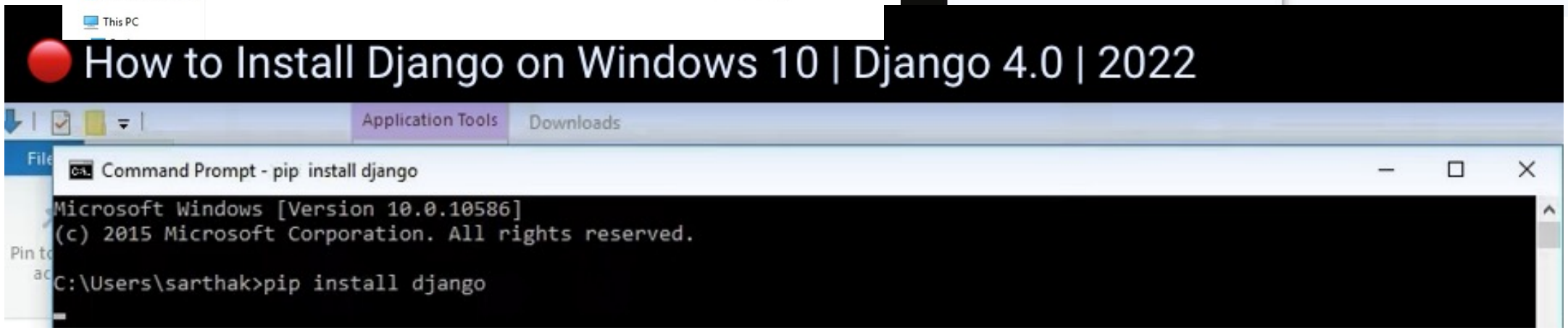












How to Install Django on Windows 10 | Django 4.0 | 2022

In this video i will show you how to install djang • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

In this video i will show you how to install djang

```
1  In this video i will show you how to install django latest version with python 3.10
2
3  so first download and install python latest version from python.org
4
5  dont forget to add this path otherwise it will not work..
6
7  now wait for a minute to install python..
8
9  now to install django use command
10
11 pip install django in cmd
12
13 you can see django latest version is successfully installed...
14
15 now i will create first project using django then i will run project on localhost..
```

How to Install Django on Windows 10 | Django 4.0 | 2022

Picture Tools | Downloads

File | Home | Share | View | Manage

Pin to Quick access | Copy | Paste | Cut | Copy path | Paste shortcut | Move to | Copy to | Delete | Rename | New folder | New item | Easy access | Properties | Open | Edit | History | Select all | Select none | Invert selection

Clipboard | Organize | New | Open | Select

This PC › Downloads ›

Quick access
OneDrive - Personal
This PC
Desktop
Documents
Downloads

Videos | python-3.10.2-amd64.exe | Django Project

How to Install Django on Windows 10 | Django 4.0 | 2022

Django Project

File | Home | Share | View

Pin to Quick access | Copy | Paste | Cut | Copy path | Paste shortcut | Move to | Copy to | Delete | Rename | New folder | New item | Easy access | Properties | Open | Edit | History | Select all | Select none | Invert selection

Clipboard | Organize | New | Open | Select

This PC › Downloads › Django Project

Quick access
OneDrive - Personal
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos

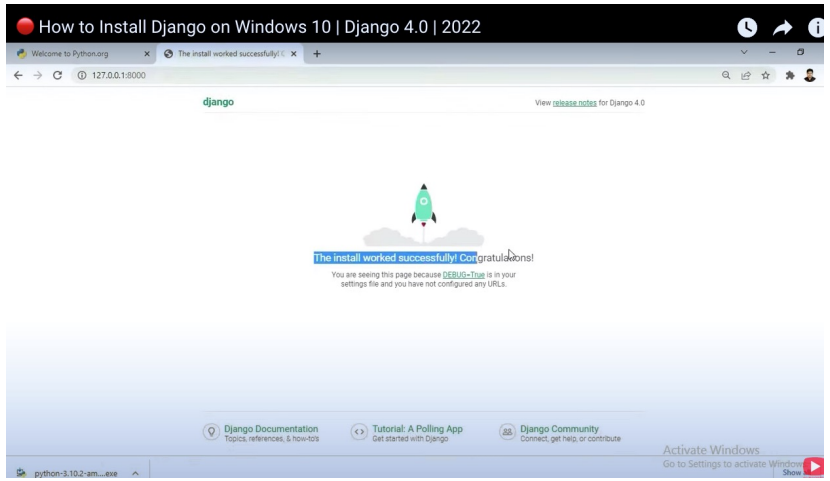
Name | Date modified | Type | Size

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\sarthak\Downloads\Django Project>django-admin startproject myproject
```

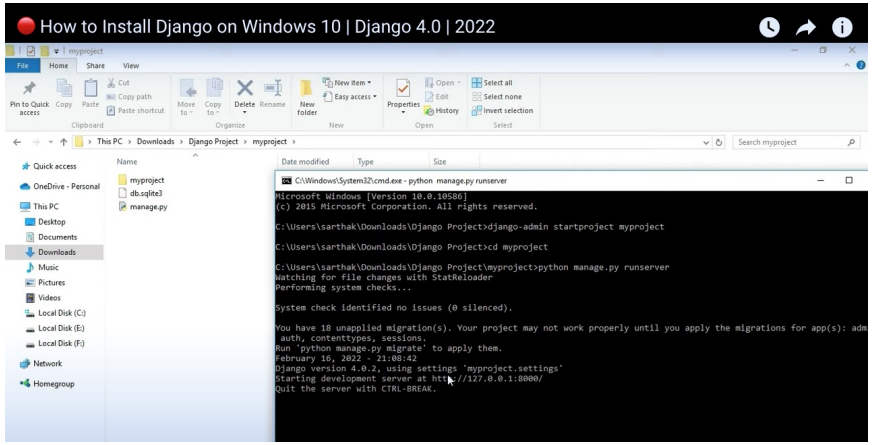
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\sarthak\Downloads\Django Project>django-admin startproject myproject

C:\Users\sarthak\Downloads\Django Project>cd myproject
```



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\sarthak\Downloads\Django Project>django-admin startproject myproject

C:\Users\sarthak\Downloads\Django Project>cd myproject

C:\Users\sarthak\Downloads\Django Project\myproject>python manage.py runserver
```

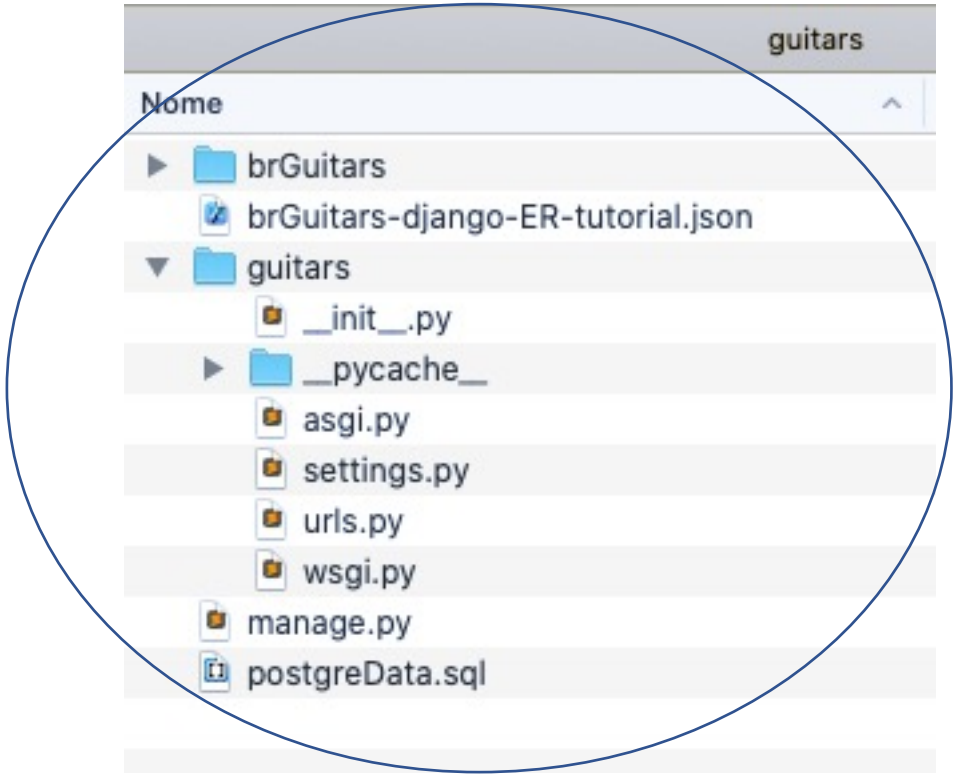


Install Python on Mac

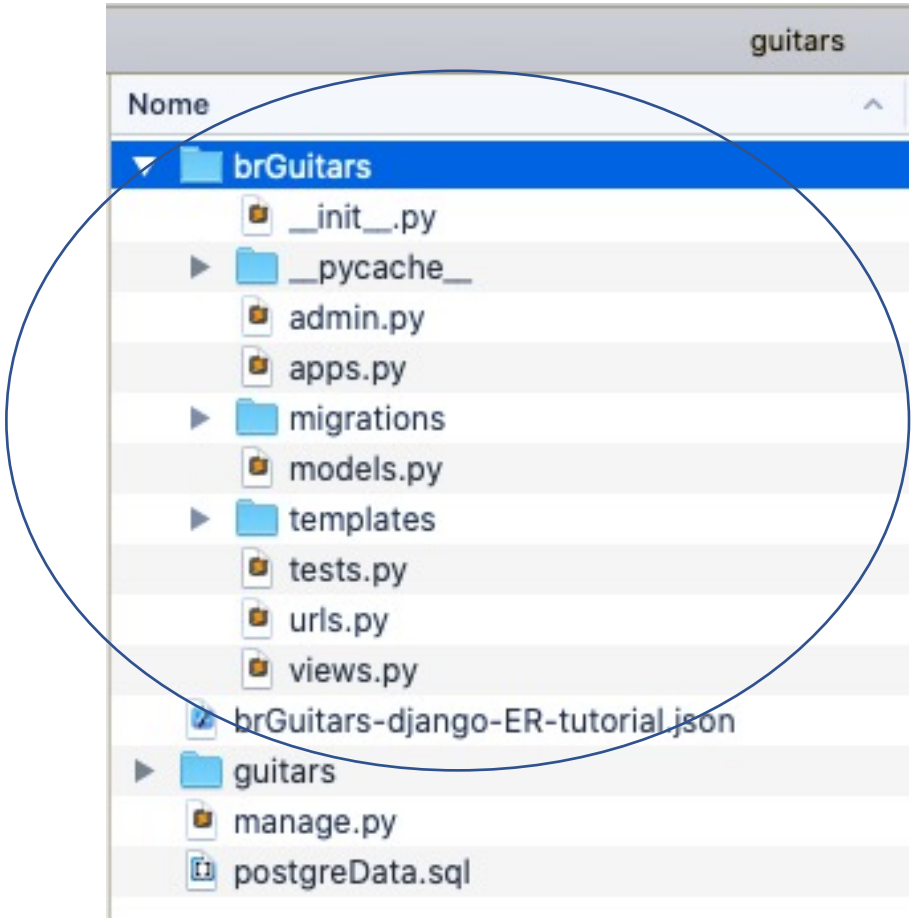
- NOTA: se nao tiver o comando brew, instale o homebrew:
- /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

```
brew install python
sudo easy_install pip
sudo pip install django
django-admin startproject thanosback
cd thanosback
python manage.py runserver
localhost:8000
```

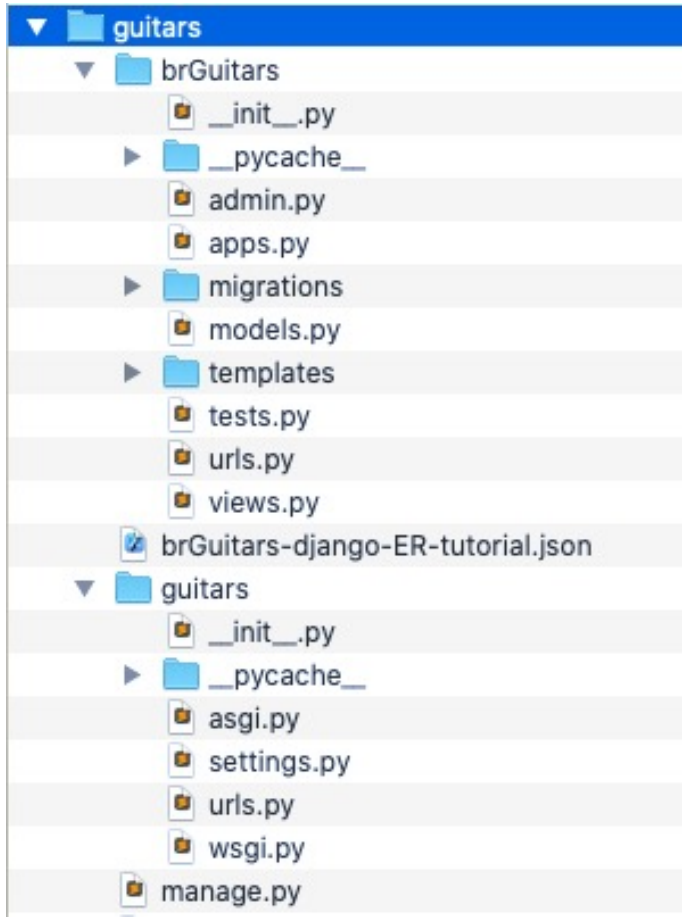
Guitars Project



brGuitars app inside guitars project



Managing the Project...



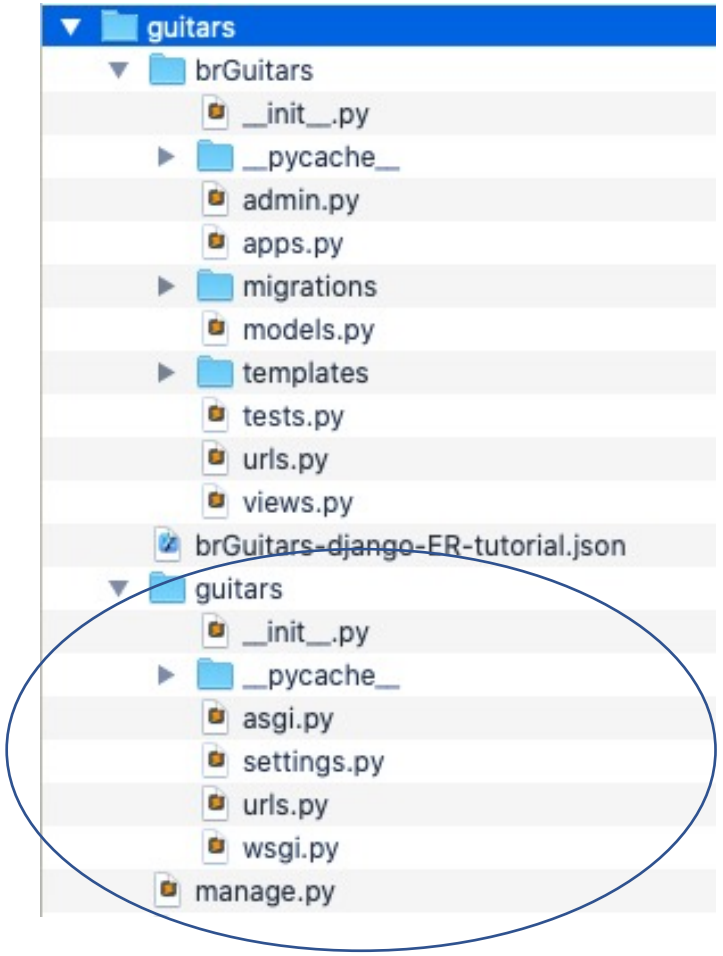
- Run important commands:

  manage.py

Such as...

- **python manage.py runserver**
- **python manage.py makemigrations brGuitars**
- **python manage.py migrate**

Main Project configurations



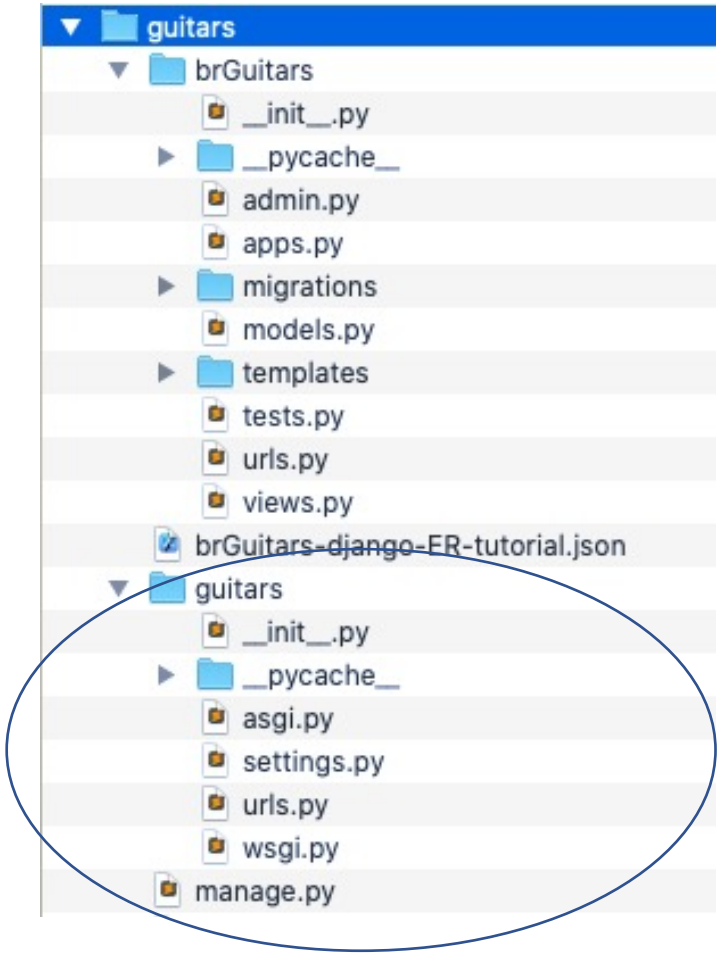
- Main project configurations

  guitars/settings.py

- Other configurations

  asgi.py, wsgi.py, admin.py, apps.py

Base project URLs

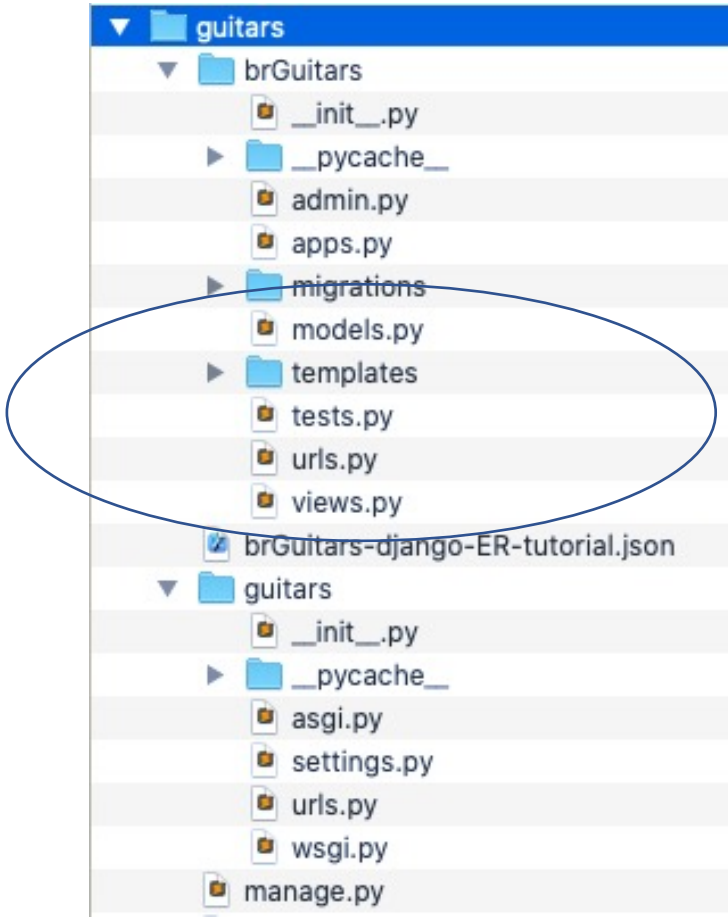


- Page addresses:

  guitars/urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('brGuitars/', include('brGuitars.urls')),
    path('admin/', admin.site.urls),
]
```

Base project URLs



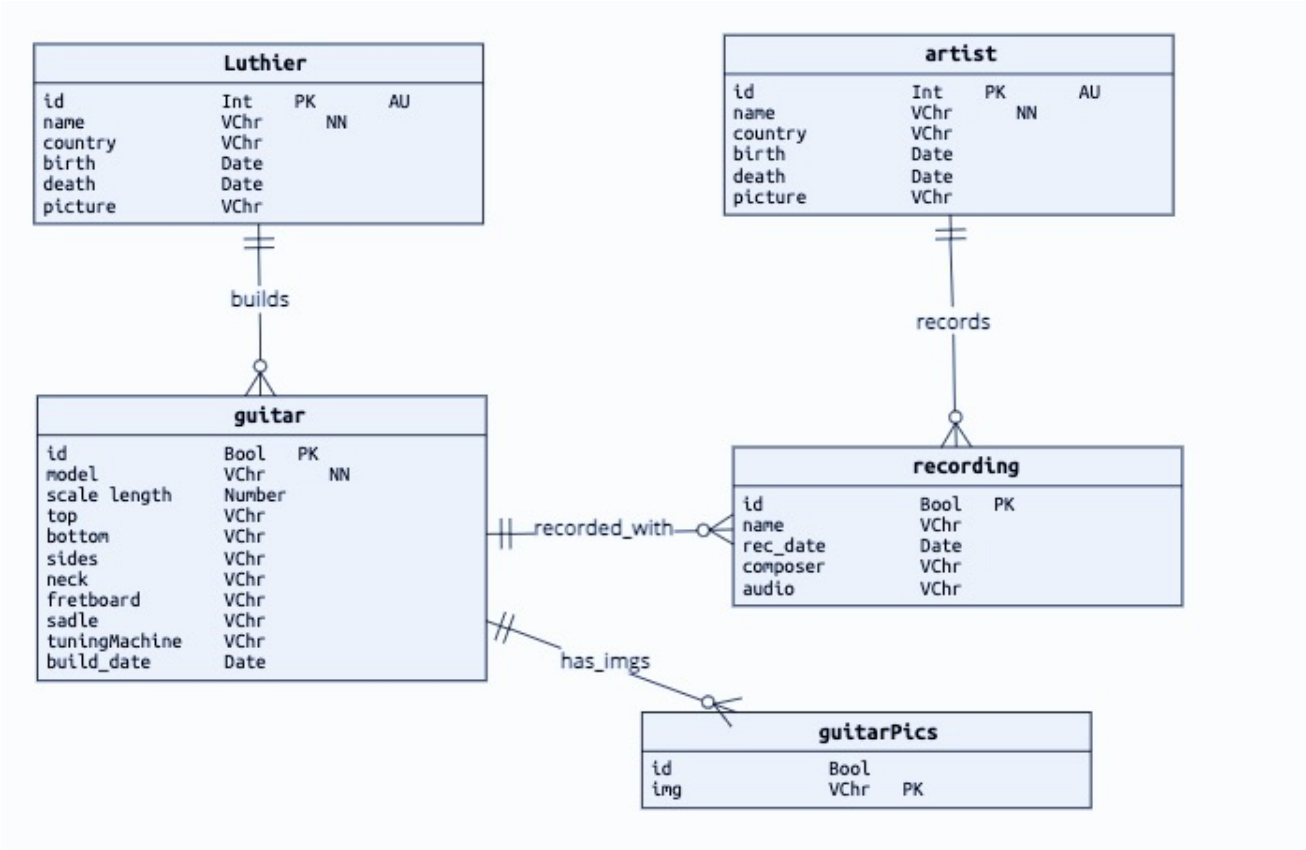
- Page addresses:

brGuitars/urls.py

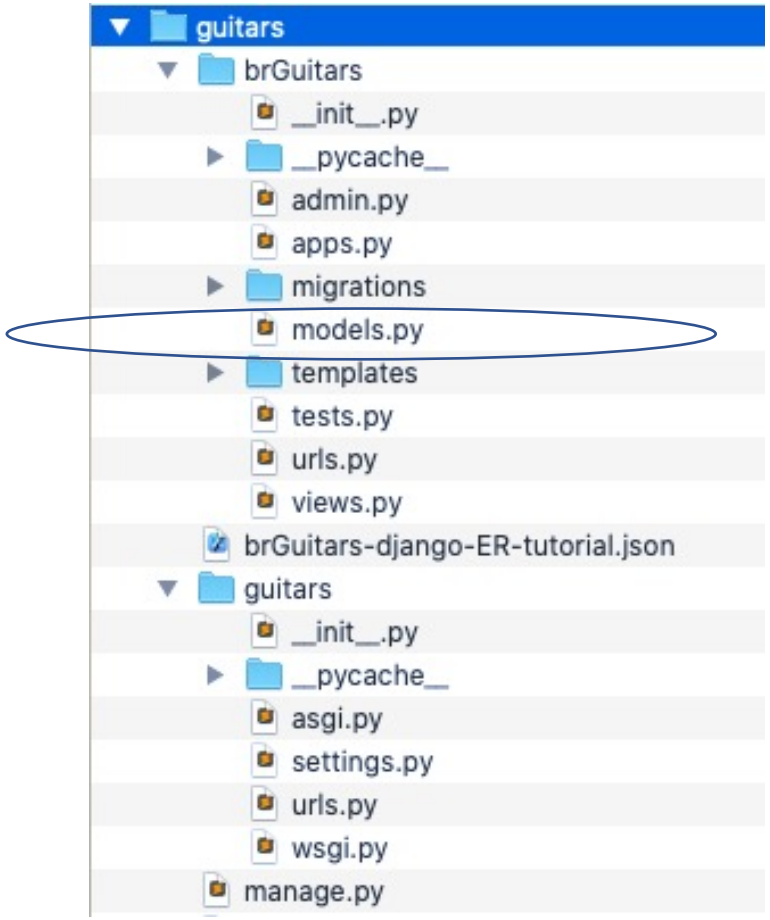
```python
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),

    #Luthiers
    path('luthiers', views.luthiers, name='luthiers'),
    path('luthiers/<int:luthier_id>/', views.luthierDetails, name='luthierDetails'),

    #artists
    path('artists', views.artists, name='artists'),
    path('artists/<int:artist_id>/', views.artistDetails, name='artistDetails'),
```

Localhost:8000/luthiers -> calls views.luthiers function

Lets have a DB schema…



Base project URLs



- We will then create the corresponding tables in the database...
- ...HOW?
- By writing a set of classes to represent those tables...
- ...then ask django to generate the tables...

A table…

```python
from django.db import models

class Luthier(models.Model):
    name = models.CharField(max_length=20, null=False, blank=False)
    country = models.CharField(max_length=200, null=True, blank=True)
    birth = models.IntegerField(null=True, blank=True)
    death = models.IntegerField(null=True, blank=True)
    pic = models.URLField(max_length=500, null=True, blank=True)
    def __str__(self):
        return self.name
    def isAlive(self):
        return death==False
```

Luthier				
id	Int	PK		AU
name	VChr		NN	
country	VChr			
birth	Date			
death	Date			
picture	VChr			

Details ...

```python
from django.db import models

class Luthier(models.Model):
    name = models.CharField(max_length=20, null=False, blank=False)
    country = models.CharField(max_length=200, null=True, blank=True)
    birth = models.IntegerField(null=True, blank=True)
    death = models.IntegerField(null=True, blank=True)
    pic = models.URLField(max_length=500, null=True, blank=True)
    def __str__(self):
        return self.name
    def isAlive(self):
        return death==False
```

Luthier			
id	Int	PK	AU
name	VChr	NN	
country	VChr		
birth	Date		
death	Date		
picture	VChr		

- Class = container with properties and functions

  e.g. Luthier

- Object = specific instances of class

  e.g. Antonio Santos, a specific luthier

Details …

```python
from django.db import models

class Luthier(models.Model):
    name = models.CharField(max_length=20, null=False, blank=False)
    country = models.CharField(max_length=200, null=True, blank=True)
    birth = models.IntegerField(null=True, blank=True)
    death = models.IntegerField(null=True, blank=True)
    pic = models.URLField(max_length=500, null=True, blank=True)
    def __str__(self):
        return self.name
    def isAlive(self):
        return death==False
```

Luthier			
id	Int	PK	AU
name	VChr	NN	
country	VChr		
birth	Date		
death	Date		
picture	VChr		

- Attributes with their data types and properties
- A Picture as a URL field = a URL
- Functions (def) to do something on the class instances

  __str__ = what to show when print(object)

Details …

```python
from django.db import models

class Luthier(models.Model):
    name = models.CharField(max_length=20, null=False, blank=False)
    country = models.CharField(max_length=200, null=True, blank=True)
    birth = models.IntegerField(null=True, blank=True)
    death = models.IntegerField(null=True, blank=True)
    pic = models.URLField(max_length=500, null=True, blank=True)
    def __str__(self):
        return self.name
    def isAlive(self):
        return death==False
```

Luthier			
id	Int	PK	AU
name	VChr	NN	
country	VChr		
birth	Date		
death	Date		
picture	VChr		

- Python specifics

  Identation is mandatory and defines blocks of code

  Class is a class

  def means a function

Details ...

```python
class Luthier(models.Model):

    def __str__(self):
        return self.name
```

Luthier			
id	Int	PK	AU
name	VChr	NN	
country	VChr		
birth	Date		
death	Date		
picture	VChr		

- Python specifics

  Identation is mandatory and defines blocks of code
  Class is a class
  def is a function

  Functions have parameters: def getNome(id)
  Functions can return values: return self.name

- No models.py repare como são definidos os tipos de dados e como são definidas as chaves estrangeiras (**foreign key**).
- Um ponto importante é que neste models.py não definimos chaves primárias, sendo que nesse caso o **django cria automaticamente chaves primárias** para cada entidade com o nome id.
- Note a palavra chave def. São definições (**def**) de funções. Uma função recebe parâmetros, faz qualquer coisa necessária e devolve resultados.
- No caso do models temos funções muito básicas. Em particular, **a função __str__** é a função que é usada sempre que alguma acção no django pretende mostrar o conteúdo de um objecto da classe.
- **definimos que quando o django pretende mostrar o conteúdo de uma guitarra, deve mostrar o nome do luthier e o modelo da guitarra**.

```
class Guitar(models.Model):
    luthier = models.ForeignKey(Luthier, on_delete=models.CASCADE)
    model = models.CharField(max_length = 200, null=False, blank=False)
    year = models.IntegerField(null=True, blank=True)
    top = models.CharField(max_length=200, null=True, blank=True)
    bottom = models.CharField(max_length=200, null=True, blank=True)
    sides = models.CharField(max_length=200, null=True, blank=True)
    neck = models.CharField(max_length=200, null=True, blank=True)
    fretboard = models.CharField(max_length=200, null=True, blank=True)
    sadle = models.CharField(max_length=200, null=True, blank=True)
    tuningMachines = models.CharField(max_length=200, null=True, blank=True)
    def __str__(self):
        return self.luthier.name+"-"+self.model

class GuitarPic(models.Model):
    img = models.URLField(max_length=500, null=False, blank=False)
    guitar = models.ForeignKey(Guitar, on_delete=models.CASCADE)
    def __str__(self):
        return self.img
```

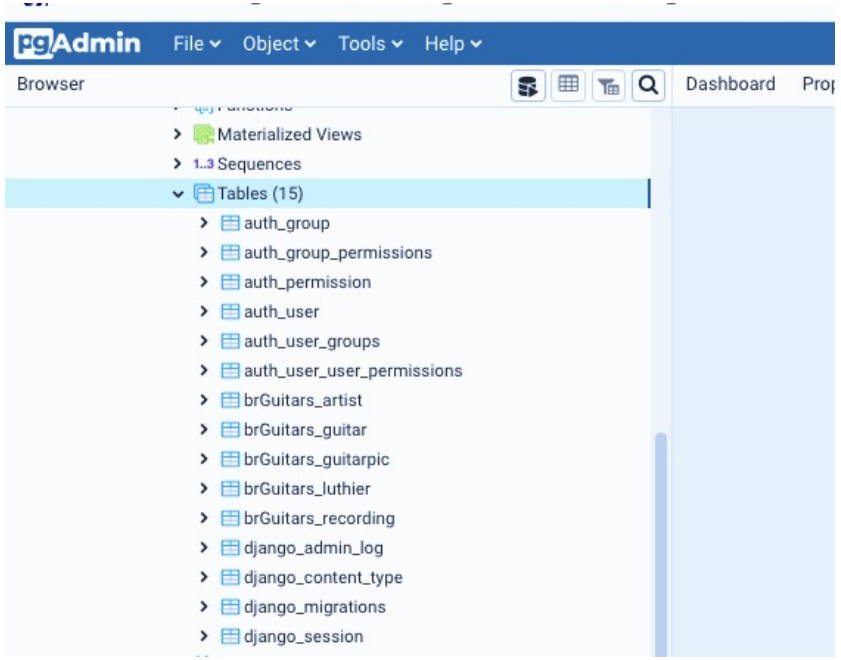
Migration to database

- Migrate = from classes generate database tables

python manage.py makemigrations brGuitars

python manage.py migrate

```
Macs-MacBook-Pro-4:guitars pedro$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, brGuitars, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying brGuitars.0001_initial... OK
  Applying sessions.0001_initial... OK
Macs-MacBook-Pro-4:guitars pedro$
```



How to develop application next?

- Tables are done and installed...
- How do I create webpages now to view tables contents and so on?

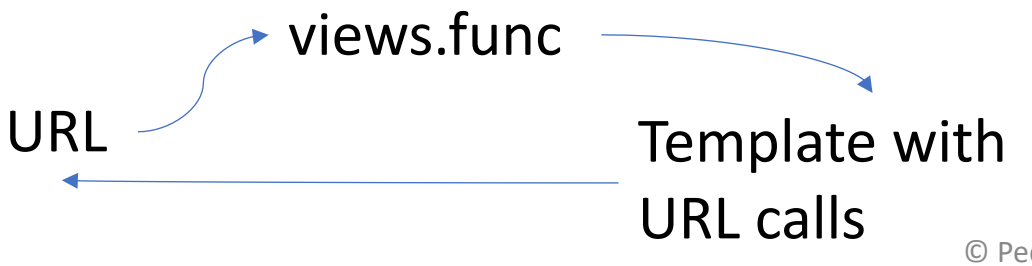
models.py: the classes as we just saw -→ migrate → the tables as we just saw

urls.py: when I specify a URLs -> cal specific view function

views.py -> view function =

what to do when called (code)

call a template = html to show contents and further link to other URLs



- **O primeiro path** indica que se não se puser nada no URL (‘’) chama a função índex do views.py. Note-se que “não pôr nada” no URL corresponde a pôr http://localhost:8000/brGuitars/, uma vez que a aplicação se chama brGuitars.

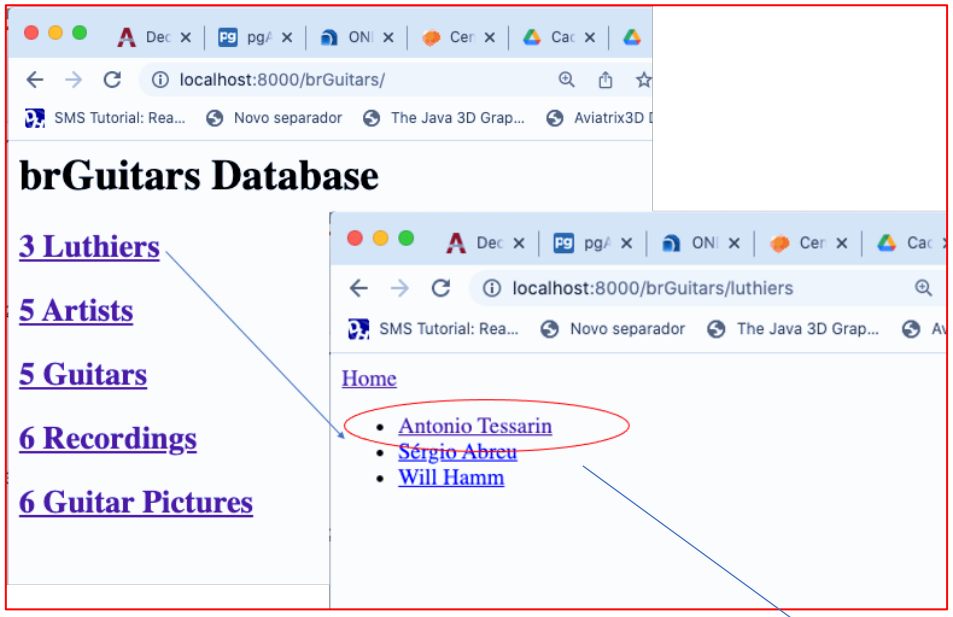
- **O segundo path** indica que se o URL for http://localhost:8000/brGuitars/luthiers/, será chamada a função views.luthiers.

- **O terceiro path** indica que se o URL for http://localhost:8000/brGuitars/luthiers/1/ será chamada a função views.luthierDetails. A ideia nesse caso será a função luthierDetails receber o valor 1 como parâmetro e por isso ir buscar o luthier com id 1 e mostrar os detalhes desse luthier.

urls.py

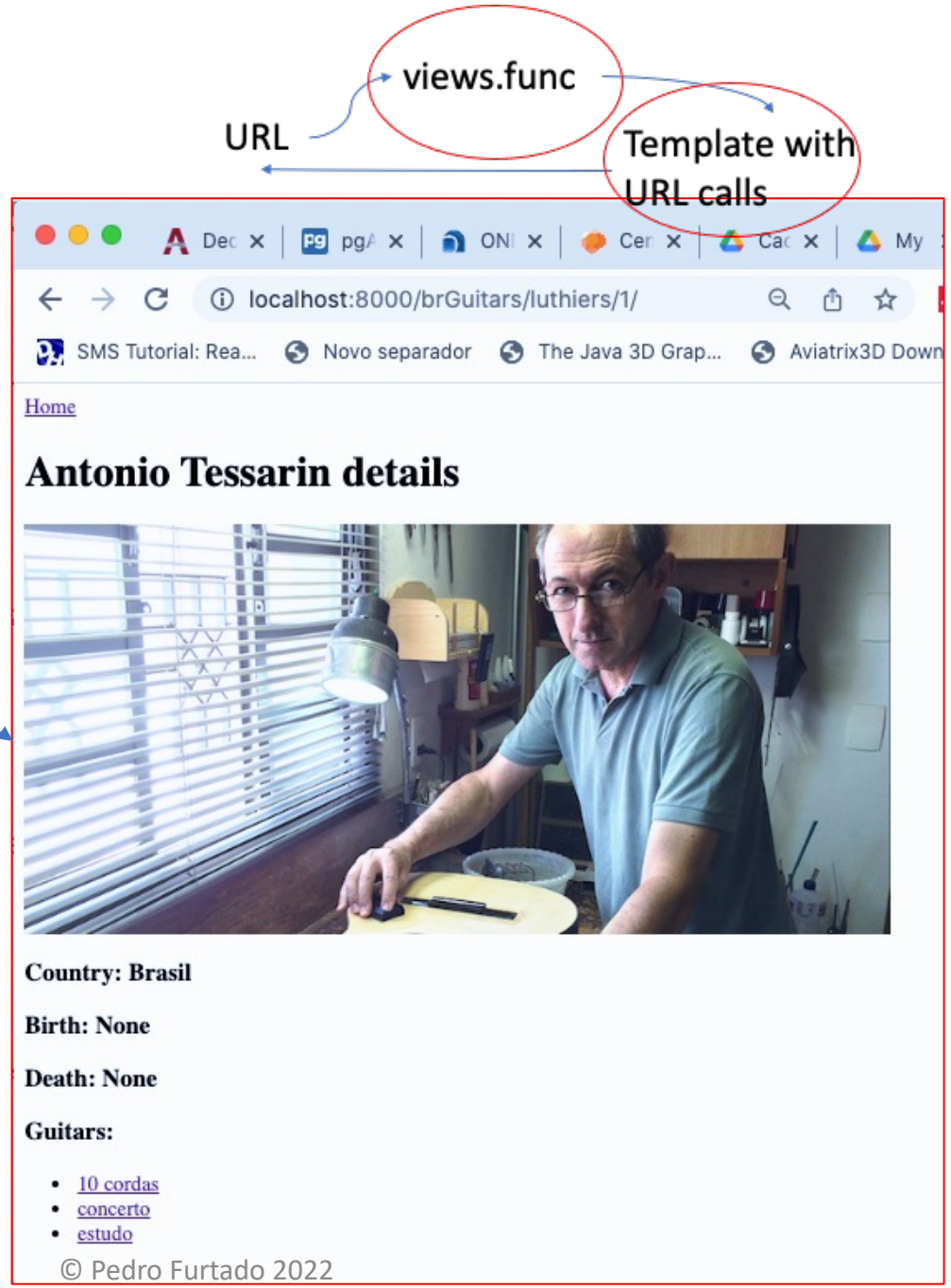
```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),

    #Luthiers
    path('luthiers', views.luthiers, name='luthiers'),
    path('luthiers/<int:luthier_id>/', views.luthierDetails, name='luthierDetails'),

    #artists
    path('artists', views.artists, name='artists'),
    path('artists/<int:artist_id>/', views.artistDetails, name='artistDetails'),

    #recordings
    path('recordings', views.recordings, name='recordings'),
    path('recordings/<int:rec_id>/', views.recordingDetails, name='recordingDetails'),

    #guitars
    path('guitars', views.guitars, name='guitars'),
    path('guitars/<int:guitar_id>/', views.guitarDetails, name='guitarDetails'),

    #guitarPics
    path('guitarPics', views.guitarPics, name='guitarPics'),
]
```

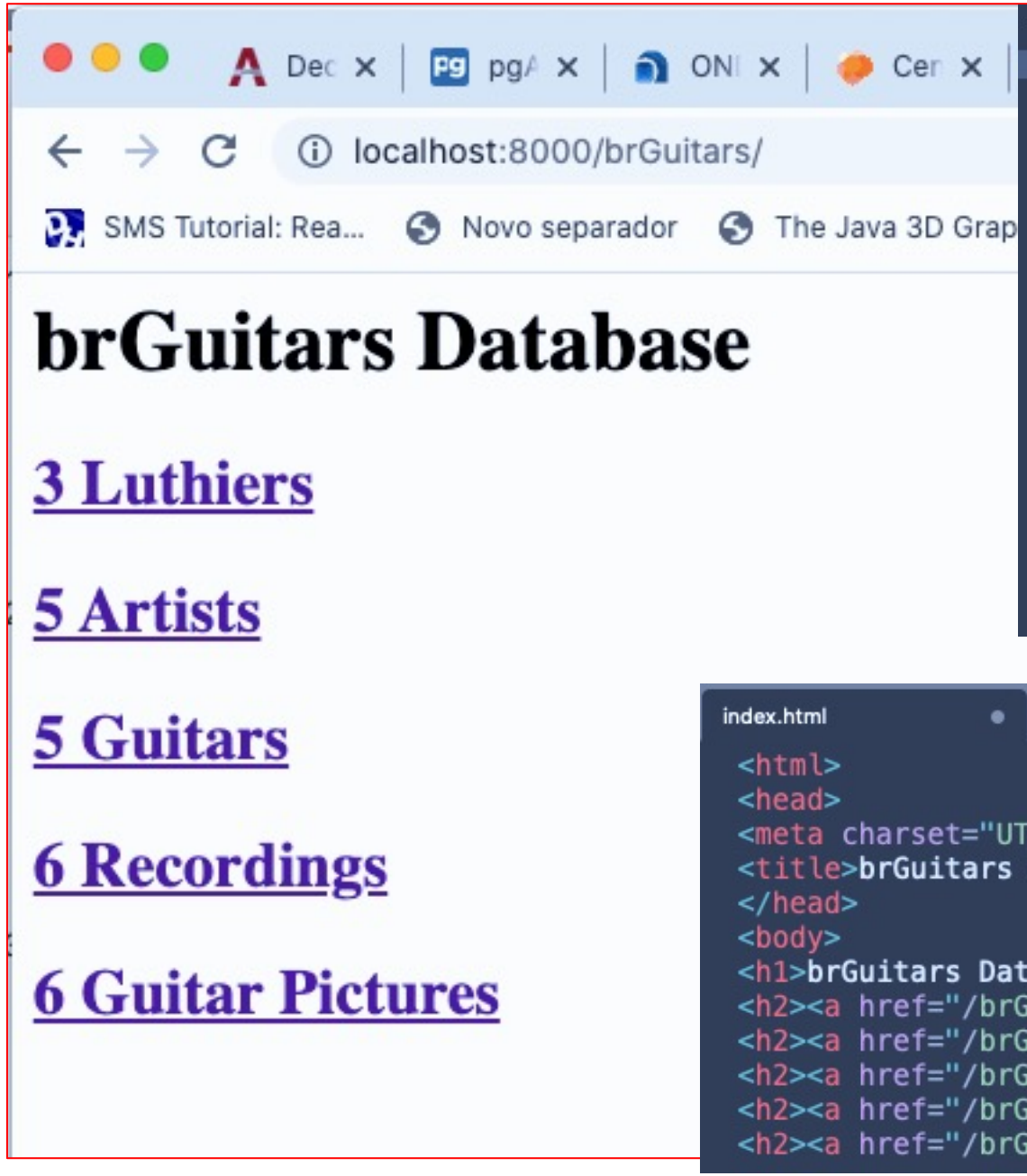
index



urls.py

```python
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),

    #Luthiers
    path('luthiers', views.luthiers, name='luthiers'),
    path('luthiers/<int:luthier_id>/', views.luthierDetails, name='luthierDetails'),

    #artists
    path('artists', views.artists, name='artists'),
    path('artists/<int:artist_id>/', views.artistDetails, name='artistDetails'),

    #recordings
    path('recordings', views.recordings, name='recordings'),
    path('recordings/<int:rec_id>/', views.recordingDetails, name='recordingDetails'),

    #guitars
    path('guitars', views.guitars, name='guitars'),
    path('guitars/<int:guitar_id>/', views.guitarDetails, name='guitarDetails'),

    #guitarPics
    path('guitarPics', views.guitarPics, name='guitarPics'),
]
```





views.py

```python
from django.http import HttpResponse
from django.shortcuts import render
from django.template import loader

from .models import Luthier
from .models import Guitar
from .models import Artist
from .models import GuitarPic
from .models import Recording

def index(request):
    template = loader.get_template('brGuitars/index.html')
    context = {
        'numLuthiers':Luthier.objects.all().count(),
        'numGuitars':Guitar.objects.all().count(),
        'numArtists':Artist.objects.all().count(),
        'numRecordings':Recording.objects.all().count(),
        'numGuitarPics':GuitarPic.objects.all().count(),
    }
    return HttpResponse(template.render(context, request))
```

index.html

```html
<html>
<head>
<meta charset="UTF-8">
<title>brGuitars Database</title>
</head>
<body>
<h1>brGuitars Database</h1>
<h2><a href="/brGuitars/luthiers"> {{numLuthiers}} Luthiers</a></h2>
<h2><a href="/brGuitars/artists"> {{numArtists}} Artists</a></h2>
<h2><a href="/brGuitars/guitars"> {{numGuitars}} Guitars</a></h2>
<h2><a href="/brGuitars/recordings"> {{numRecordings}} Recordings</a></h2>
<h2><a href="/brGuitars/guitarPics"> {{numGuitarPics}} Guitar Pictures</a></h2>
```

- A **função índex**, que como vimos atrás é chamada quando pomos http://localhost:8000/brGuitars/,

-> carrega um "template" chamado índex.html

-> **cria 5 variaveis** (tais como numLuthiers), cada uma com o número de elementos (linhas) de cada uma das tabelas definidas na base de dados, **enviando essas 5 variáveis para o template processar**.

-> Luthier.objects.all().count() significa que:

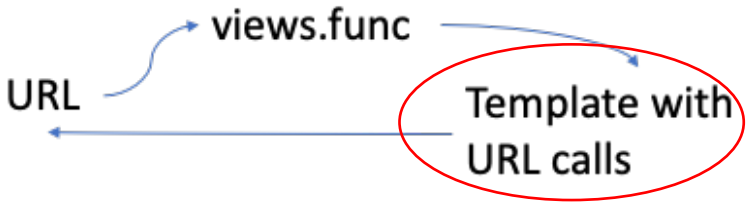
- nos referimos à **tabela de luthiers** (através da classe Luthier definida em models.py),
- objects significa que queremos os objectos, isto é, as linhas da tabela,
- all() significa que queremos todas as linhas (em vez de alguma condição que buscaria só algumas linhas)
- count() é a função que ordena que sejam contados os números de linhas devolvidos.

views.py

```
from django.http import HttpResponse
from django.shortcuts import render
from django.template import loader

from .models import Luthier
from .models import Guitar
from .models import Artist
from .models import GuitarPic
from .models import Recording

def index(request):
    template = loader.get_template('brGuitars/index.html')
    context = {
        'numLuthiers':Luthier.objects.all().count(),
        'numGuitars':Guitar.objects.all().count(),
        'numArtists':Artist.objects.all().count(),
        'numRecordings':Recording.objects.all().count(),
        'numGuitarPics':GuitarPic.objects.all().count(),
    }
    return HttpResponse(template.render(context, request))
```

Template index.html

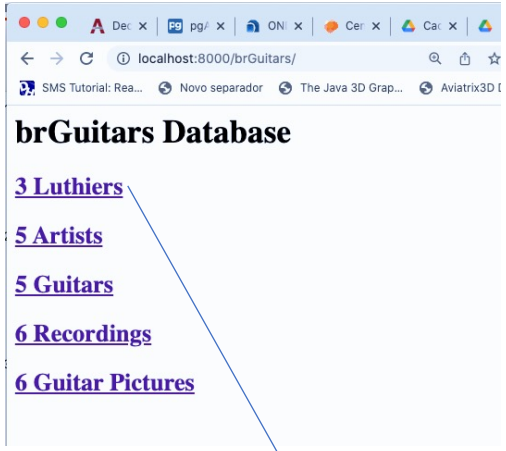
- Template índex.html receives as parameter the variables prepared in the index function of views.py (numLuthiers, numArtists, numGuitars, …).
- It is an html file which can use the typical html elements, ...
- ...but it can also use any of the variables it received as parameters, in that case enclosed by {{ to start and }} to end the reference to it.
- In this case we have the typical html tags <html>, <head>,<title> and <body>...

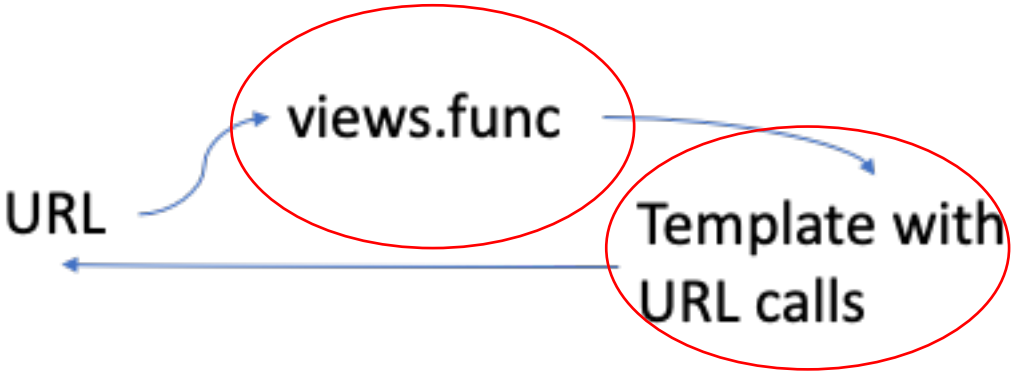


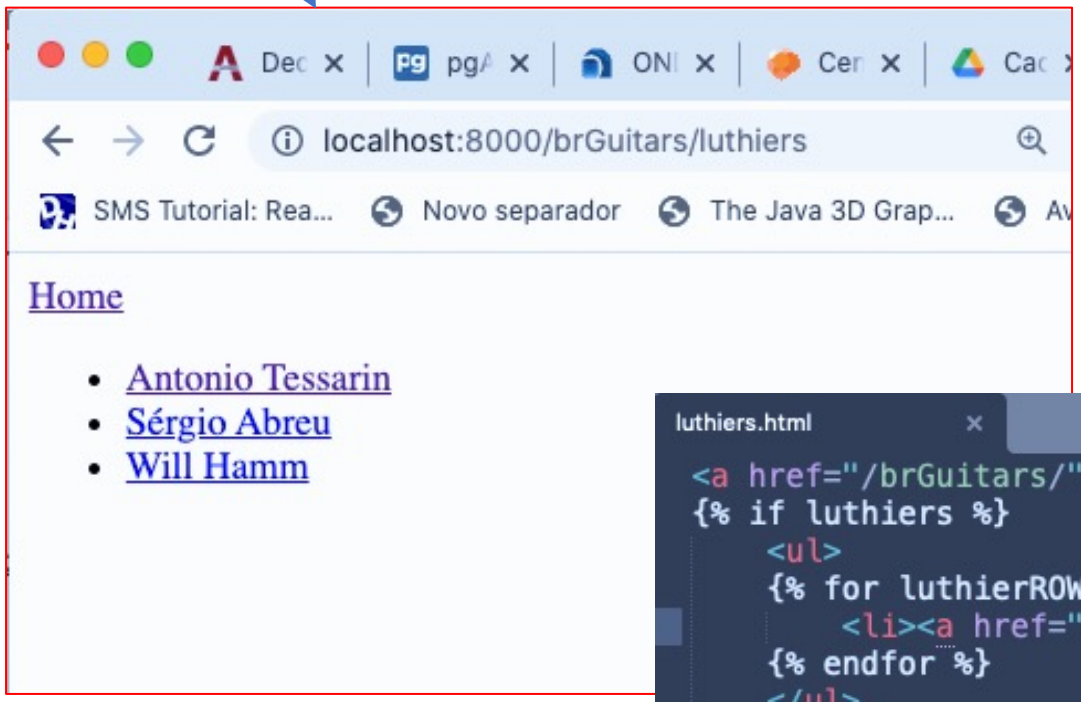
index.html

```
<html>
<head>
<meta charset="UTF-8">
<title>brGuitars Database</title>
</head>
<body>
<h1>brGuitars Database</h1>
<h2><a href="/brGuitars/luthiers"> {{numLuthiers}} Luthiers</a></h2>
<h2><a href="/brGuitars/artists"> {{numArtists}} Artists</a></h2>
<h2><a href="/brGuitars/guitars"> {{numGuitars}} Guitars</a></h2>
<h2><a href="/brGuitars/recordings"> {{numRecordings}} Recordings</a></h2>
<h2><a href="/brGuitars/guitarPics"> {{numGuitarPics}} Guitar Pictures</a></h2>
```

luthiers








```python
def luthiers(request):
    template = loader.get_template('brGuitars/luthiers.html')
    items = Luthier.objects.order_by('name')[0:]
    context = {
        'luthiers':items
    }
    return HttpResponse(template.render(context, request))
```

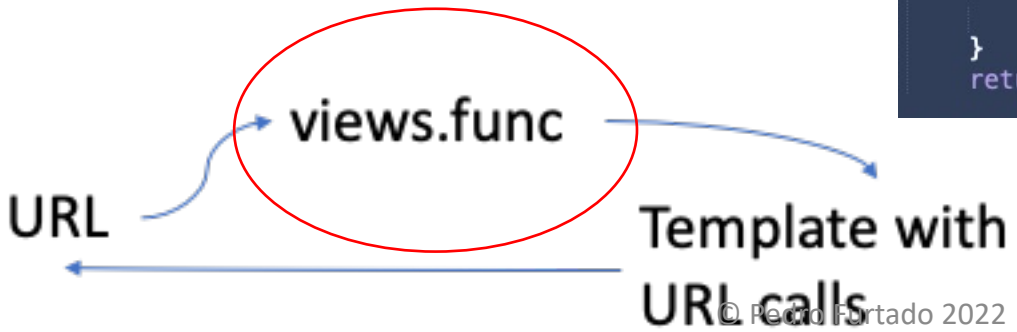
luthiers.html

```html
<a href="/brGuitars/">Home</a></p>
{% if luthiers %}
    <ul>
    {% for luthierROW in luthiers %}
        <li><a href="/brGuitars/luthiers/{{luthierROW.id}}/">{{ luthierROW.name }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No luthiers to show.</p>
{% endif %}
```

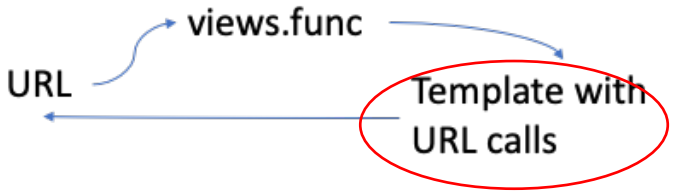
- **Função luthiers:**

- A função luthiers, que como vimos antes é chamada quando pomos http://localhost:8000/brGuitars/luthiers/,
- vai carregar o template brGuitars/luthiers.html
- envia para esse template a variável luthiers
- A variável luthiers contém todos os luthiers ordenados por nome.
  - Luthier.objects vai buscar todos os objectos do tipo luthier, isto é, as linhas da tabela luthier;
  - order_by('name') vai ordenar essas linhas buscadas por nome;
  - [0:] significa todos os elementos (linhas) começando na primeira (linha de índice 0) e acabando na ultima, já que em python um : sem nada a seguir quer dizer até ao final do conjunto.

```python
def luthiers(request):
    template = loader.get_template('brGuitars/luthiers.html')
    items = Luthier.objects.order_by('name')[0:]
    context = {
        'luthiers':items
    }
    return HttpResponse(template.render(context, request))
```



Template luthiers.html:

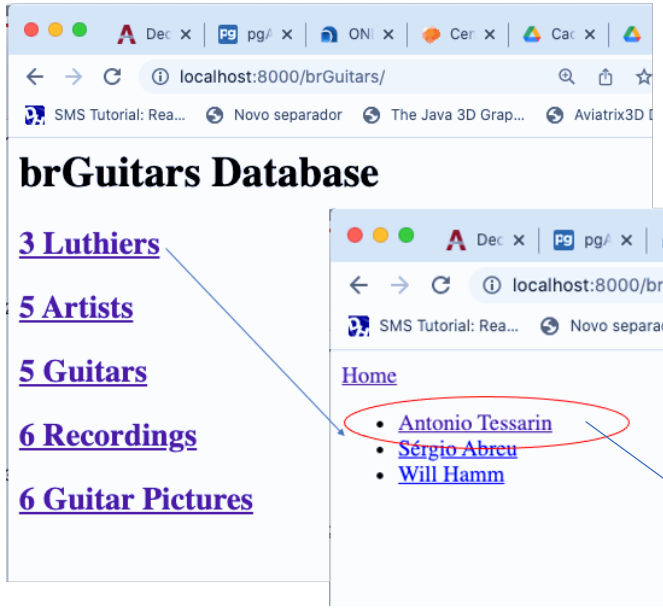


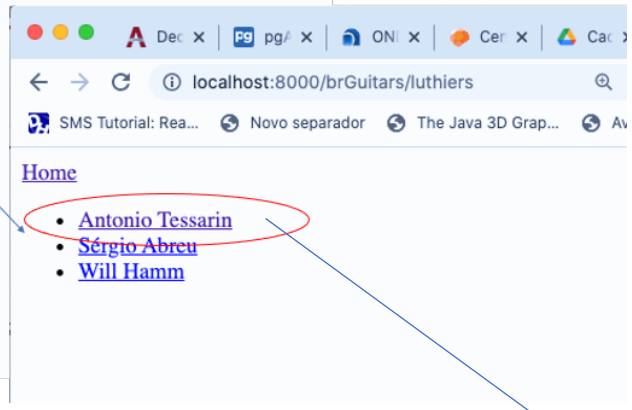
- lists the names of all luthiers.
- an anchor (a link) to the homepage Home</p>, which allows the user to go back to the index page.
- The next part of the file is html code to show a list of something.
  - The tag <ul> means the start of an unordered list,
  - and the tag closes that list.
  - The tag <li> means list item (closed by) and it shows an item of the list.

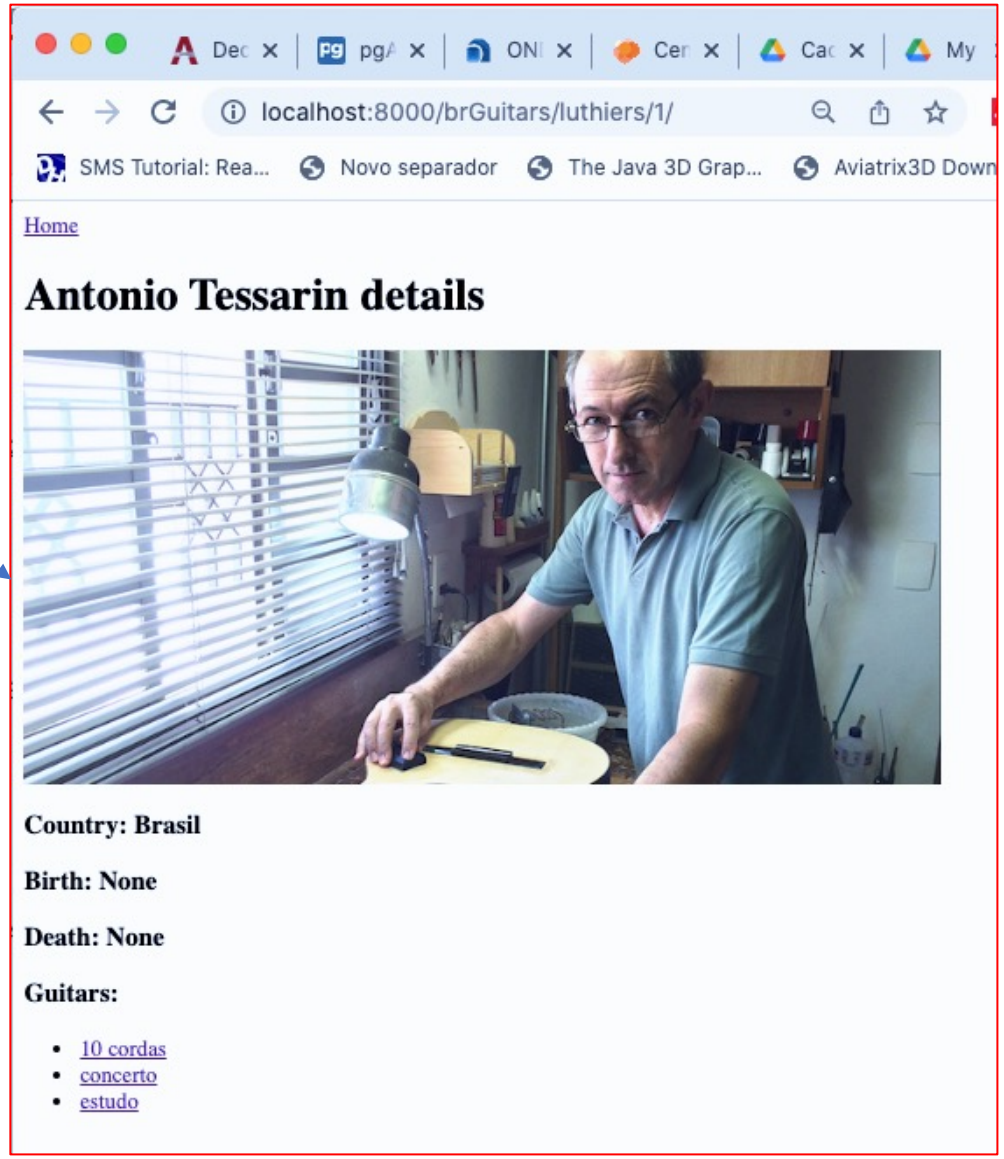
luthiers.html

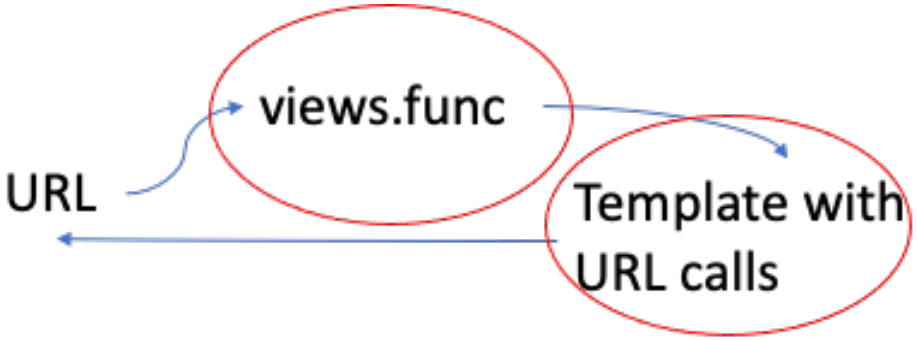
```
<a href="/brGuitars/">Home</a></p>
{% if luthiers %}
    <ul>
    {% for luthierROW in luthiers %}
        <li><a href="/brGuitars/luthiers/{{luthierROW.id}}/">{{ luthierROW.name }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No luthiers to show.</p>
{% endif %}
```

luthiersDetails









Função luthiersDetails:

- Parametro de entrada luthier_id, que corresponde por exemplo ao 1 de http://localhost:8000/brGuitars/luthiers/1/,
- Busca um so luthier baseado na chave primaria (id)

  myLuthier = Luthier.objects.get(pk=luthier_id)

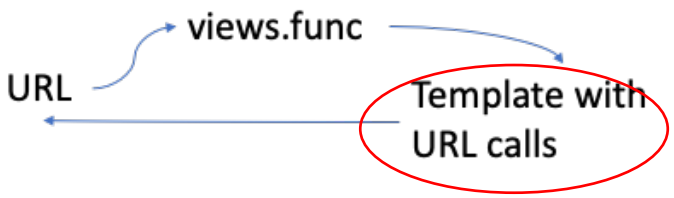
- Vou também buscar todas as guitarras desse luthier através de uma função filter sobre a chave estrangeira luthier da tabela guitar

  myLuthierGuitars=Guitar.objects.filter(luthier = luthier_id)

- Enviados para o template (brGuitars/luthierDetails.html) processar.


```python
def luthierDetails(request, luthier_id):
    template = loader.get_template('brGuitars/luthierDetails.html')
    try:
        myLuthier = Luthier.objects.get(pk=luthier_id)
        myLuthierGuitars = Guitar.objects.filter(luthier = luthier_id)
        context = {'luthier' : myLuthier, 'guitars' : myLuthierGuitars}
    except Luthier.DoesNotExist:
        raise Http404("Luthier does not exist")

    return HttpResponse(template.render(context, request))
```

Template luthierDetails.html



luthierDetails.html

```
<a href="/brGuitars/">Home</a></p>
<h1>{{luthier.name}} details</h1>
<picture>
 <source media="(min-width:320px)" srcset={{luthier.pic}}>
 <source media="(min-width:240px)" srcset={{luthier.pic}}>
 <img src={{luthier.pic}} alt={{luthier.name}}-{{luthier.pic}} style="width:auto;">
</picture>
<h3>Country: {{luthier.country}}</h3>
<h3>Birth: {{luthier.birth}}</h3>
<h3>Death: {{luthier.death}}</h3>
<h3>Guitars:</h3>
{% if guitars %}
    <ul>
    {% for guitarROW in guitars %}
        <li><a href="/brGuitars/guitars/{{guitarROW.id}}/">{{guitarROW.model}}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No guitar to show.</p>
{% endif %}
```

The end