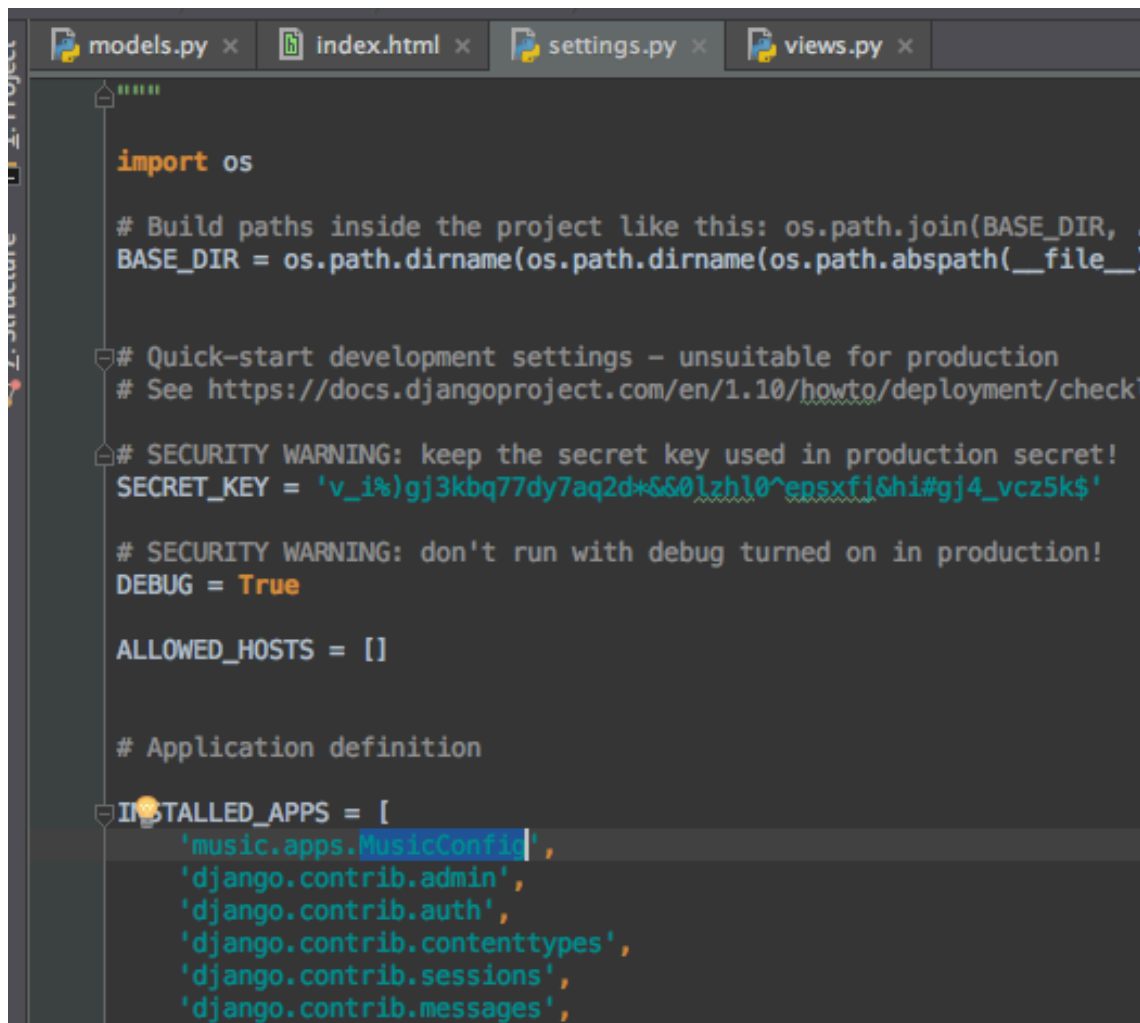


- django -> criar o backend de websites, atraves do python
- download:
 - sudo pip install django
 - para verificar as versoes do python e do django -> pip freeze
- comecar um projeto:
 - django-admin startproject nome_do_projeto(django_try)
- manage.py -> um programa que auxilia a acessar a db, criar usuarios, runserver, entreoutros
- help -> python manage.py -h
- in django_try:
 - __init__.py -> mostra que o projeto criado eh um pacote python
 - settings.py -> configuracoes gerais do site criado
 - urls.py -> table of contents of the website, onde voce coloca as urls que acessam os apps do django_try; user requested and give a response
 - wsgi.py -> webserver gateway interface, basic overview
- runserver -> python manage.py runserver, ctrl C para parar de rodar o server
- conectando/sincronizando o workspace com o db:
 - python manage.py migrate
 - obs -> sincroniza os installed apps (settings.py) com o db
- create super user -> python manage.py createsuperuser
- app -> slack n eh um app, eh um programa de computador; app, sim, eh cada parte de um website, tal como em um canal do youtube que possui videos, forum, isso sao apps -> cada app deve ter uma funcao
- criar um app -> python manage.py startapp app_name(music)
- in django_try /music:
 - migrations -> conecta os codigos do app com o db
 - __init__.py -> mostra que o projeto eh um pacote python
 - admin.py -> deleta users, posts; sudo
 - apps.py -> configuracoes do app
 - models.py -> a forma de como armazenar a data para esse app(templates)
 - tests.py
 - views.py -> pegam um request do usuario e retorno uma resposta(pode ser html response por exemplo, ou um download)
- instalando o app:
 - em django_try /setting.py -> adicionar o app:

A screenshot of a code editor with four tabs: models.py, index.html, settings.py, and views.py. The settings.py file is open, showing Django configuration. The code includes imports, path definitions, development settings, a security warning, a secret key, debug settings, allowed hosts, and a list of installed apps. The 'music.apps.MusicConfig' entry in the INSTALLED_APPS list is highlighted with a blue selection box.

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'v_i%)gj3kbq77dy7aq2d*&&0lzh10^epsxfj&hi#gj4_vcz5k$'

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

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'music.apps.MusicConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
```

python manage.py makemigrations music

python manage.py migrate

obs -> toda mudança no app, executar as duas linhas acima no terminal

-montando um app:

em music/models.py:

```
admin.py × apps.py × models.py × music/urls.py ×
from django.db import models

class Band(models.Model):
    band_name = models.CharField(max_length=100)
    genre = models.CharField(max_length=50)

    def __str__(self):
        return self.band_name

class Album(models.Model):
    band = models.ForeignKey(Band, on_delete=models.CASCADE)
    album_title = models.CharField(max_length=100)
    album_date_release = models.DateField()

    def __str__(self):
        return self.album_title

class Song(models.Model):
    album = models.ForeignKey(Album, on_delete=models.CASCADE)
    song_title = models.CharField(max_length=100)
    song_extension = models.CharField(max_length=10)

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

, esse é um exemplo de mudança
- obs: ver as funções de models
para instanciar objetos nessas classes:

```
MacBook-Pro-de-Tarik:django_try TarikBauer$ python manage.py shell
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from music.models import Band, Album, Song
>>> a = Band(band_name='Metallica', genre='Metal')
>>> a.save()
>>> a.id
1
```

```

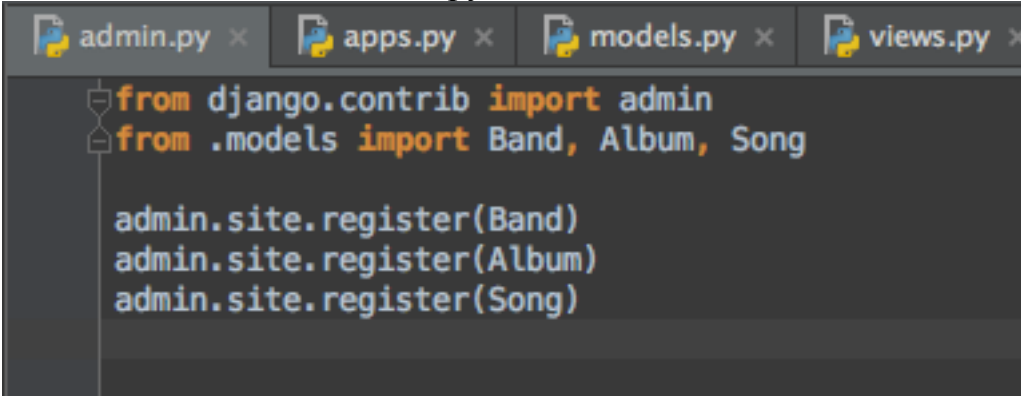
>>> exit()
MacBook-Pro-de-Tarik:django_try TarikBauer$ python manage.py shell
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from music.models import Band, Album, Song
>>> Band.objects.all()
<QuerySet [<Band: Metallica>]>
>>> Band.objects.get(id=1)
<Band: Metallica>
>>> a = Band.objects.get(id=1)

>>> b = Album(band=a, album_title='Ride the Lightning', album_date_release='1984-07-27')
>>> b.save()
>>> b.id
1
>>> c = Album.objects.get(id=1)
>>> d = Song(album=c, song_title='Fade to Black', song_extension='.mp3')
>>> d.save()
>>> Song.objects.all()
<QuerySet [<Song: Fade to Black>]>
>>> exit()

```

ou:

em music/admin.py:



```

from django.contrib import admin
from .models import Band, Album, Song

admin.site.register(Band)
admin.site.register(Album)
admin.site.register(Song)

```

e editar diretamente na url -> .../admin/music

- criando urls para cada app, nesse caso teremos:

127.0.0.1:8000/music/url_criada:

criar um python file (urls.py) no app criado (music)

em django_try/urls.py:

```
django_try/urls.py x admin.py x apps.py x models.py x
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^music/', include('music.urls')),
]
```

em music/urls.py:

```
django_try/urls.py x admin.py x apps.py x mo
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
]
```

obs -> essa url criada significa o main page do app (music), o seu view se dá no views.index

- criando um view:

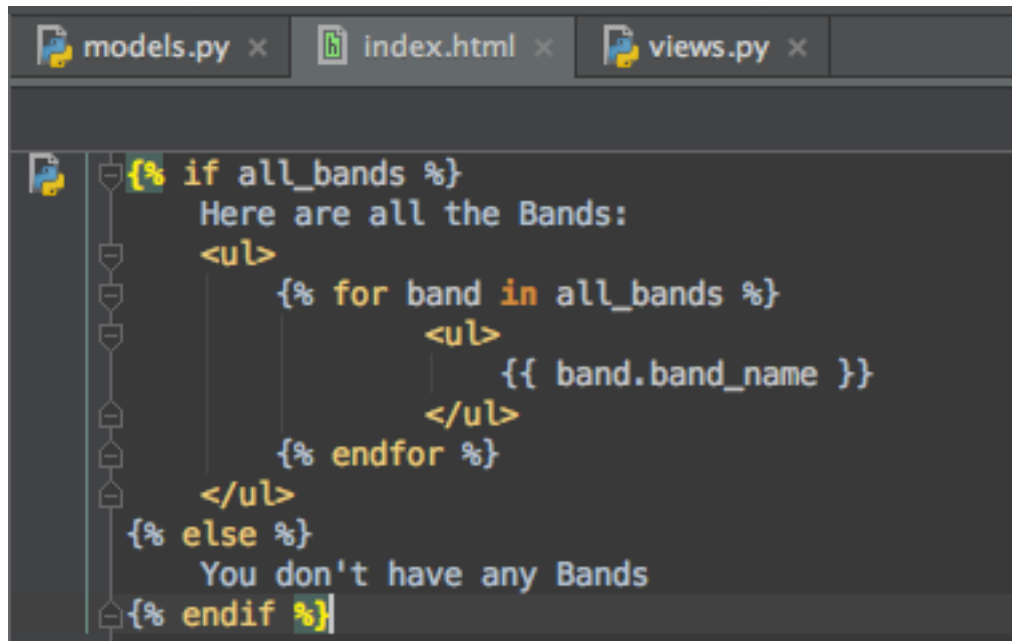
criar em musica um diretório templates, dentro de templates
criar um diretorio music, dentro de music criar index.html

em views:

```
models.py x index.html x views.py x
from django.http import HttpResponse, Http404
from django.shortcuts import render
from .models import Band

def index(request):
    all_bands = Band.objects.all()
    context = {
        'all_bands': all_bands,
    }
    return render(request, 'music/index.html', context)
```

em index.html:

A screenshot of a code editor with three tabs: 'models.py', 'index.html', and 'views.py'. The 'index.html' tab is active, showing Django template code. The code uses curly braces for control structures and HTML tags for output. A vertical toolbar on the left side of the editor contains icons for file operations and code navigation.

```
{% if all_bands %}
    Here are all the Bands:
    <ul>
        {% for band in all_bands %}
            <ul>
                {{ band.band_name }}
            </ul>
        {% endfor %}
    </ul>
{% else %}
    You don't have any Bands
{% endif %}
```

esse é o básico!