**DESPLIEGUE**

Vamos a emular como seria un deploy en produccion

Ver pag 1 Manual.

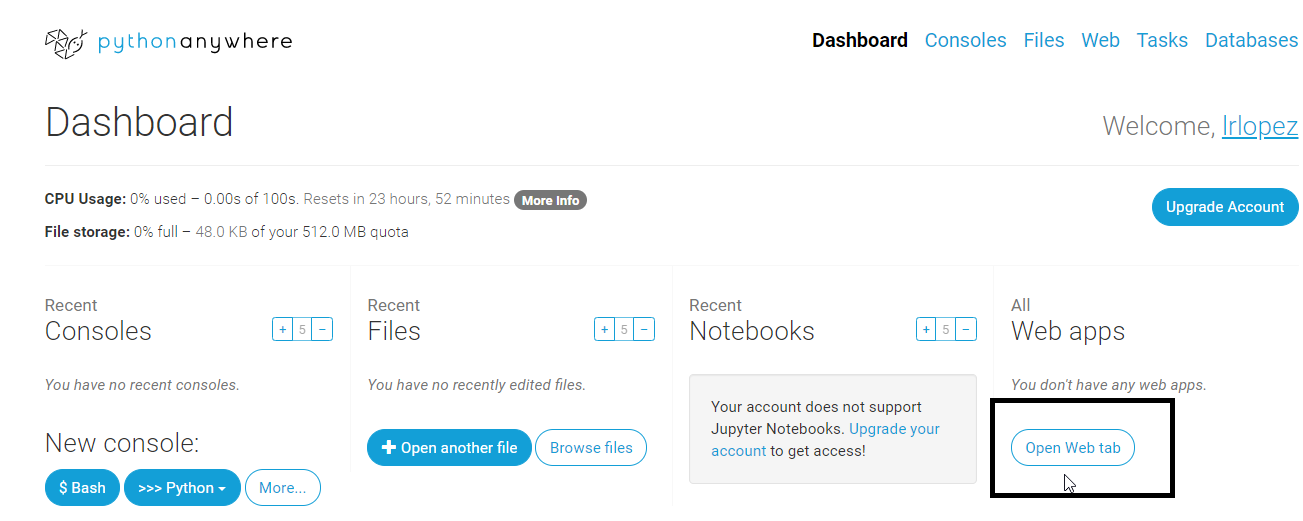


Python anywhere => <https://www.pythonanywhere.com/>

**REGISTRO Y CONFIGURACION (Subimos nuestro 1er proyecto Web)**

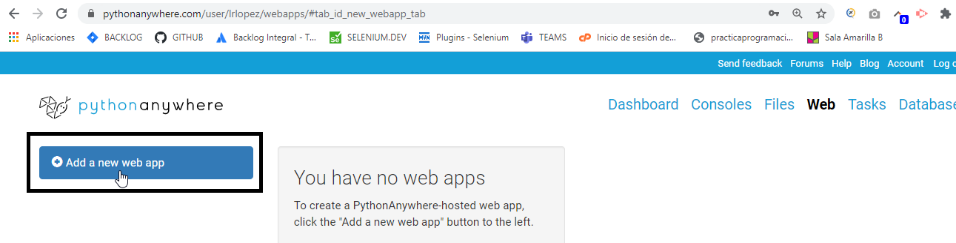
1.Crear una cuenta en Python Anywhere (https://www.pythonanywhere.com/).

2. Una vez allí nos dirigimos a “Open New tab”.

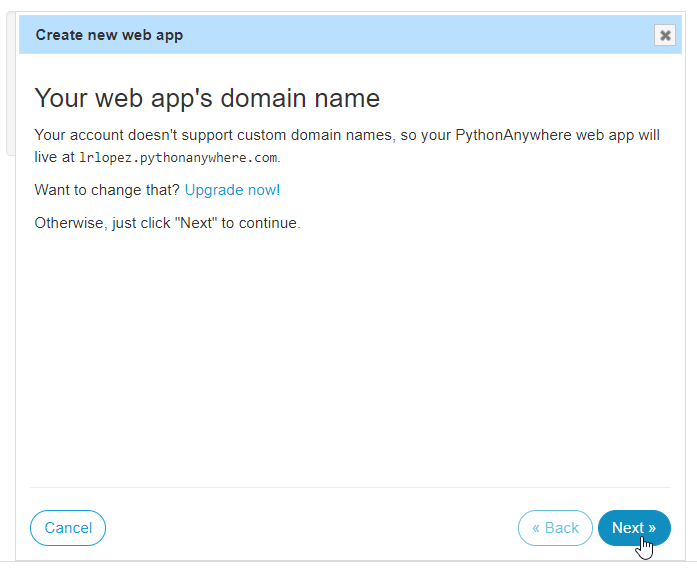


3. Y oprimimos **Add a new web app**.

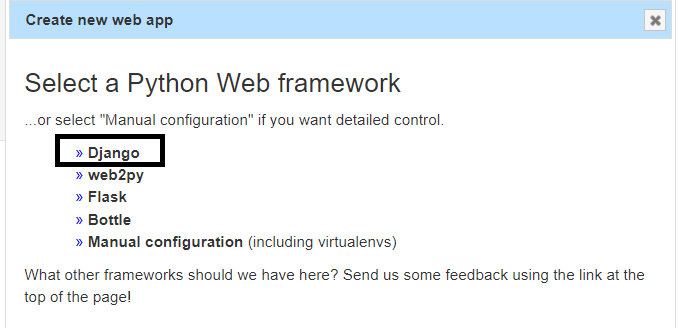
a.



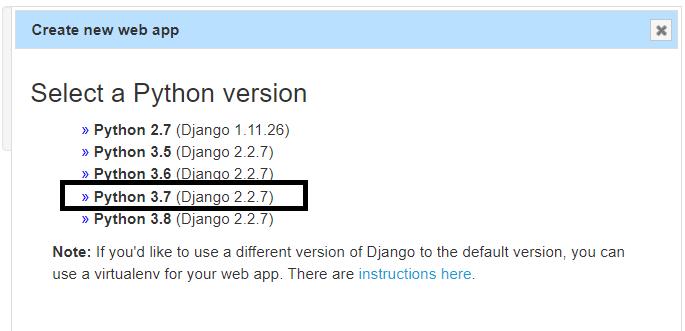
b. Oprimimos **Next**



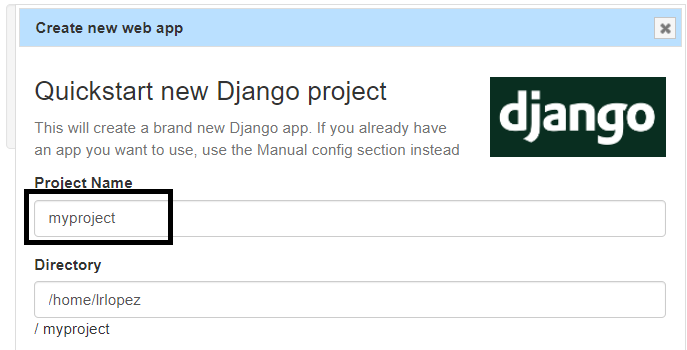
c. Elejimos el **framework** que queremos utilizar: **Django**

****

d. La **versión** de Python: **Python 3.7** (Tener en cuenta que la versión que elijamos aquí influirá sobre los comandos que ejecutemos luego, por ejemplo cuando ejecutemos “python3.7 manage.py migrate”)

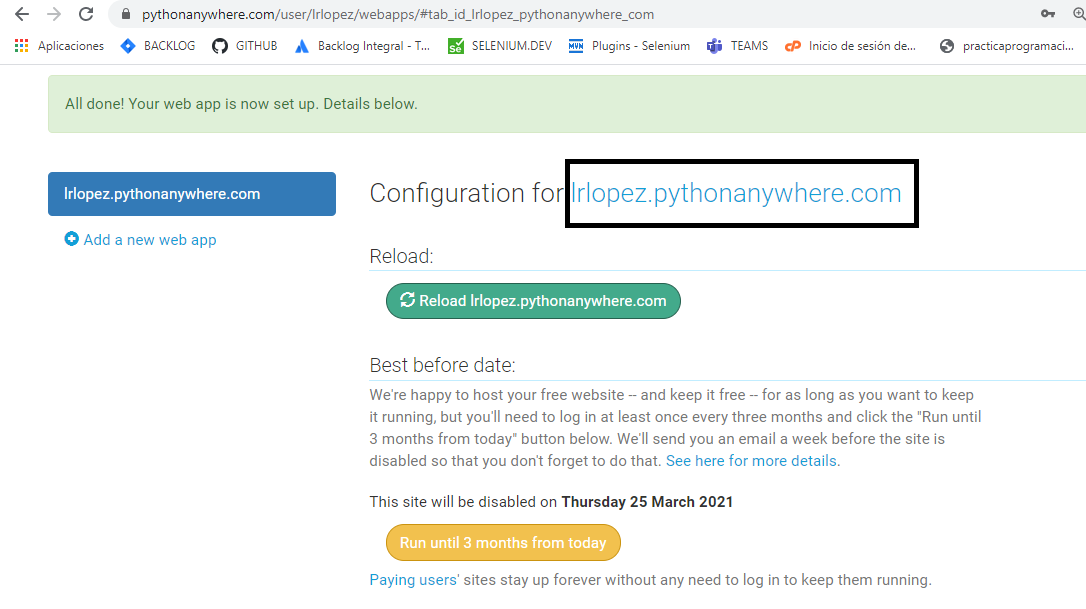


e. Colocamos el **nombre del proyecto**: **myproject**



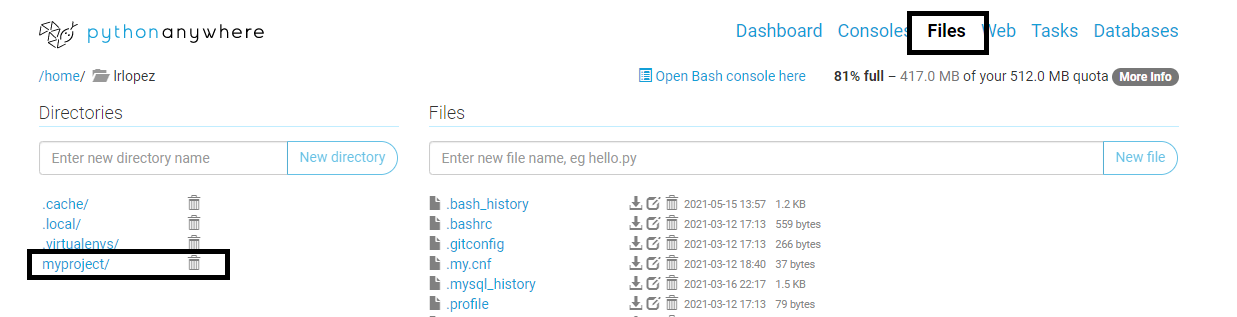
**Next**

f. Al finalizar veremos esto:



Lo señalado en el rectángulo será la URL en la que va a estar corriendo nuestra recién creada aplicación, <nombreusuario>.pythonanywhere.com. Es decir la aplicación que verán los usuarios en Produccion.

Y si nos vamos a la solapa **FILES** vemos que se ha creado la estructura de carpetas de nuestro proyecto:



**CONFIGURACION en PythonAnywhere**

Ya tenemos una aplicación corriendo en la url **http://**[**lrlopez.pythonanywhere.com**](http://lrlopez.pythonanywhere.com/).

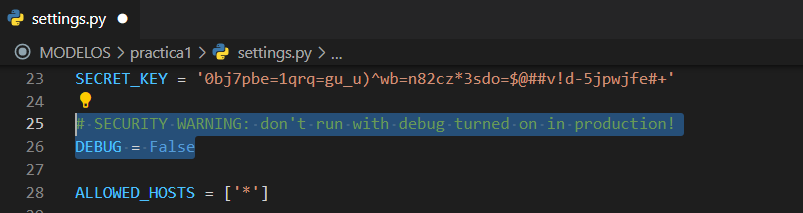
1.**Cambios en la configuración del proyecto en desarrollo (local):** los siguientes cambios que haremos para deployar en producción harán que al intentar correr nuestra app en forma local no podamos hacerlo ya que variaremos el **settings.py.** Para solucionar esto se crea un **settings\_dev.py** paralelo con las configuraciones de desarrollo (veremos mas adelante como hacerlo)

2. Partimos del siguiente código y lo abrimos en el VSC en forma local:

****

3. Vamos a **settings.py** y seteamos la variable DEBUG en **False** (estaba en True y la pasamos a False). Asi de esta manera no se podrá ver la descripción del error en prod tal como se visualiza en desarrollo.

DEBUG = False

****

4. En ese mismo archivo realizamos la siguiente configuración en **ALLOWED\_HOSTS**:

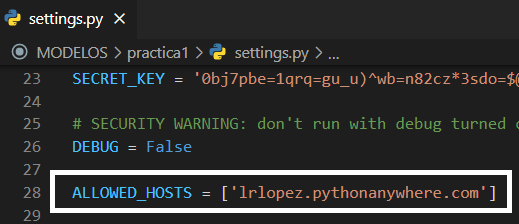
Es requisito cuando **DEBUG=False**.

Django exige indicar las direcciones de IP o dominios en los que se provee la aplicación web en producción. Sino arrojara un error.

Para mas referencia ver <https://docs.djangoproject.com/en/3.0/topics/security/#host-header-validation> (cuestiones de Seguridad en Django).

5. Colocaremos el subdominio que **PythonAnywhere** nos ha creado,

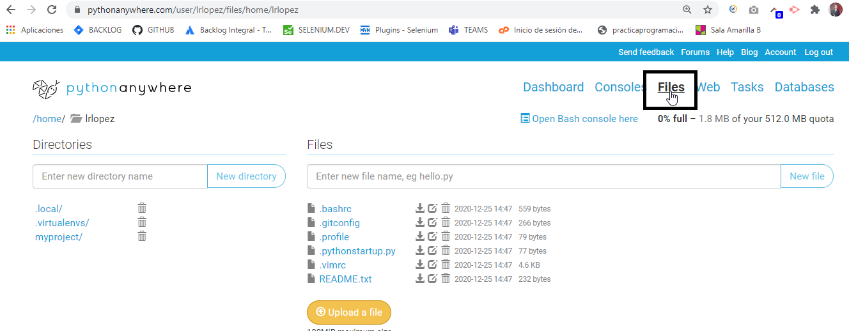
ALLOWED\_HOSTS = ['lrlopez.pythonanywhere.com']



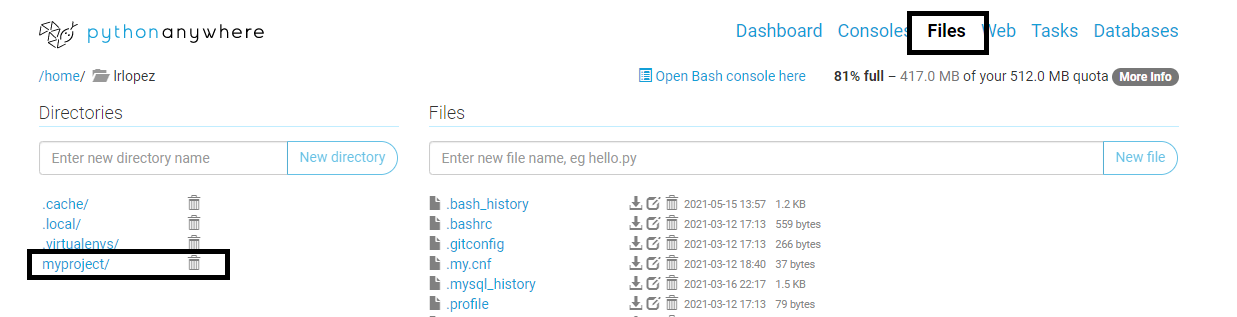
**SUBIMOS nuestro Proyecto a PythonAnywhere**

Tenemos varias maneras de subirlo. Ahora veremos la forma tradicional y luego podemos usar git y github.

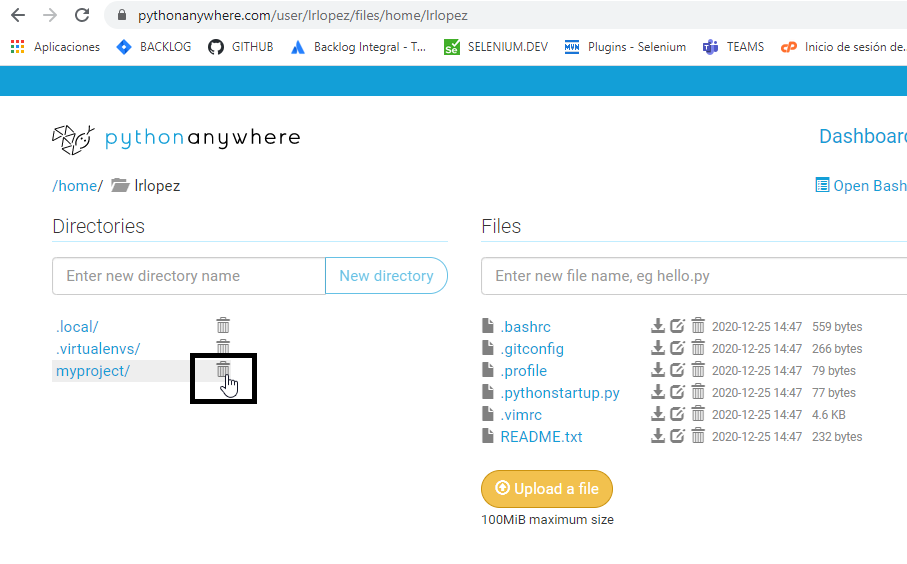
1. Vamos a **Files**:



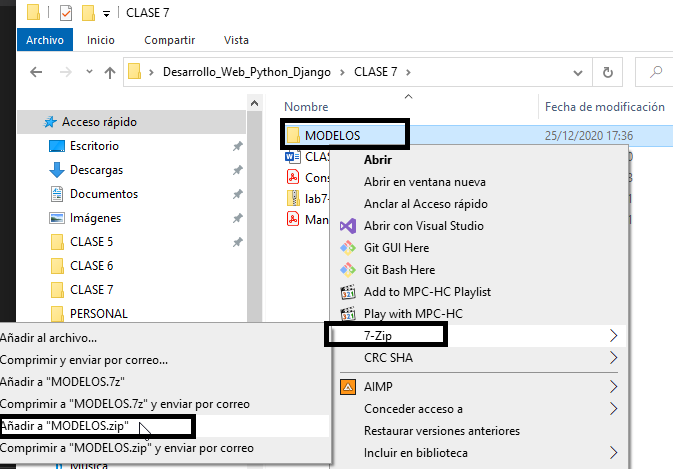
Vemos nuestra estructura de carpetas.



Como queremos subir nuestra propia carpeta lo que hacemos es eliminar el proyecto creado por defecto:

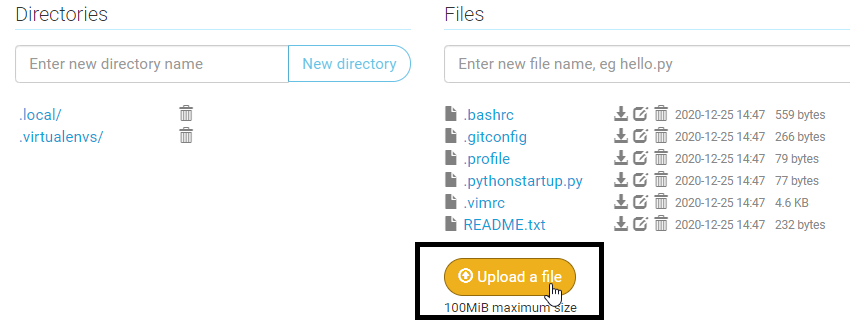
****

2. Comprimimos (.zip) nuestro proyecto local en nuestra PC.

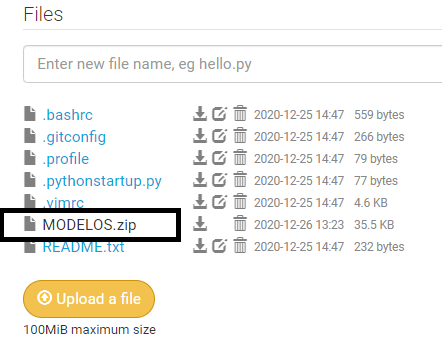




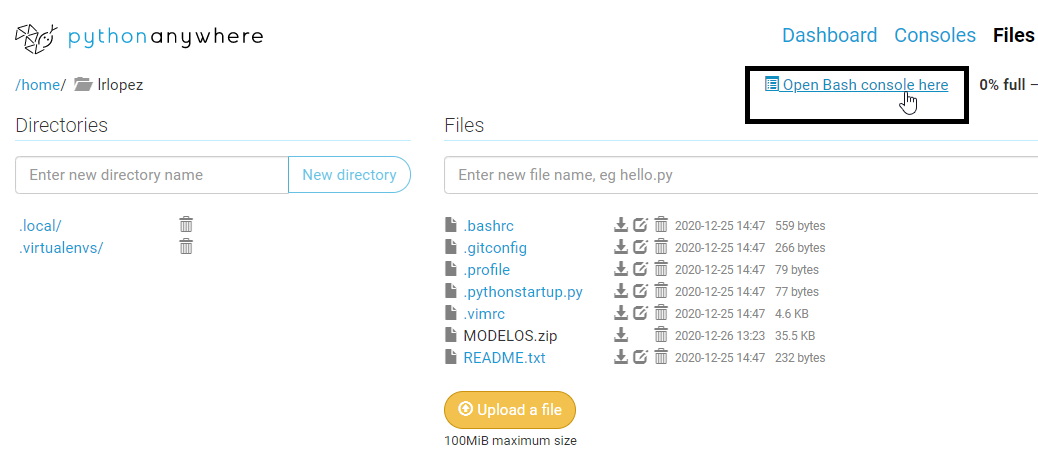
3. Vamos a “**Upload a file”** .



Y levantamos nuestro .zip

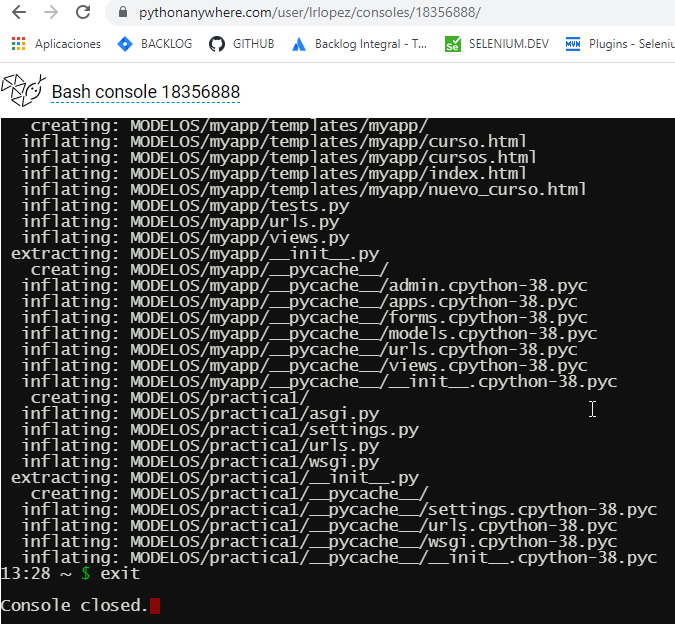


4. Ya lo tenemos subido a Python Anywhere. Ahí donde estamos nos dirigimos a la opcion **Open Bash console here** (en donde abrimos una cosola en PyAn).

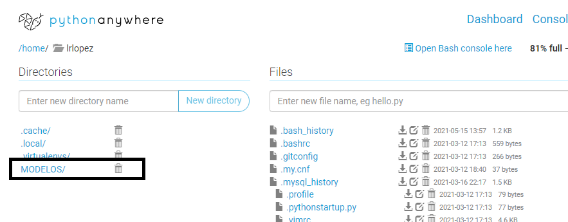


Y ejecutamos unzip MODELOS.zip

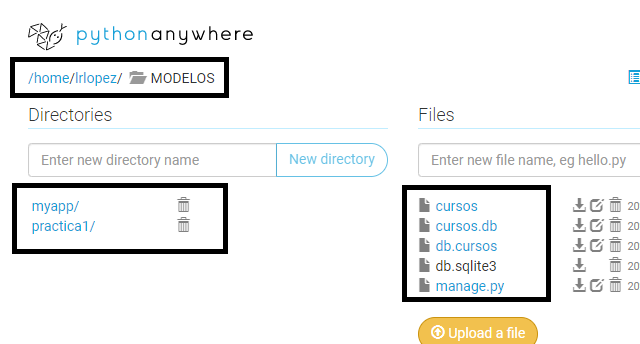
Una vez descomprimido podemos cerrar la consola via CTRL+d. O bien la podemos dejar abierta para futuros usos.



Si nos vamos a Files vemos que se creo la carpeta **MODELOS** con nuestro proyecto web:



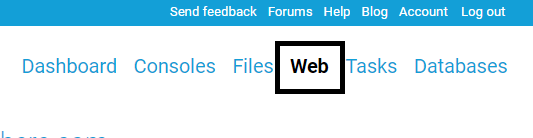
Y si entramos a la misma vemos la misma estructura de archivos y carpetas que teníamos localmente:



5. **ARCHIVOS ESTATICOS**: Hacemos un tratamiento especial de los archivos **css, js** e **imágenes**. **IMPORTANTE:** Vamos a pag 8 del manual para leer un poco sobre este tema.

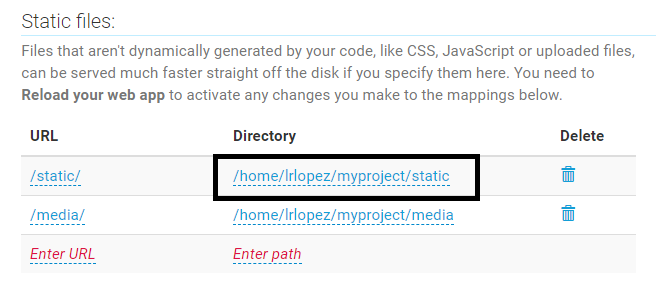
a.Python Anywhere por defecto sirve los **archivos estáticos** dentro de la carpeta **static** de nuestro proyecto.

a. Vamos a la pestaña **Web**:

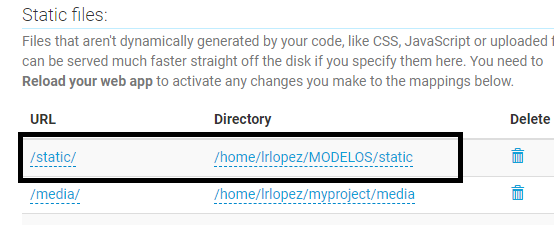


b. Buscamos la sección de **static files**:

y como nosotros usamos un proyecto preexistente (lo trajimos desde desarrollo) cambiamos esto:



Por esto:

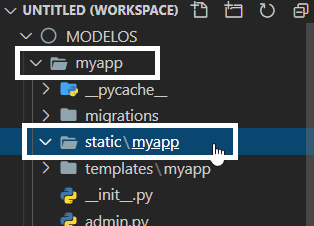


Es decir podemos poner el nombre de nuestra carpeta principal o proyecto (en mi caso es MODELOS) de forma tal que los archivos estáticos se alojaran en la siguiente ruta:

**/home/lrlopez/MODELOS/static**

Esto lo maneja asi Python Anywhere por cuestiones de seguridad y de performance.

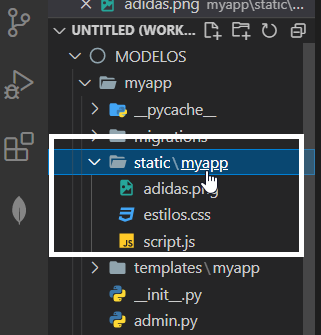
c. Creamos la carpeta **static** en desarrollo y le agregamos un css, js e imagen.



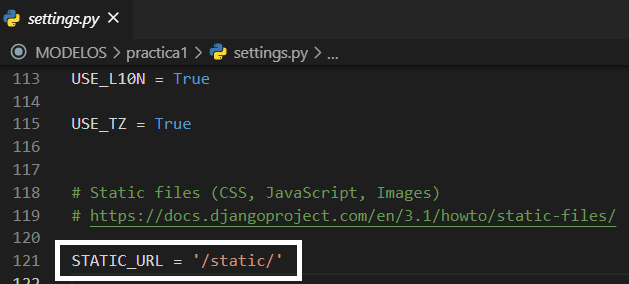
Sumamos el siguiente proyecto (de animación) a nuestra carpeta static y templates (localmente)



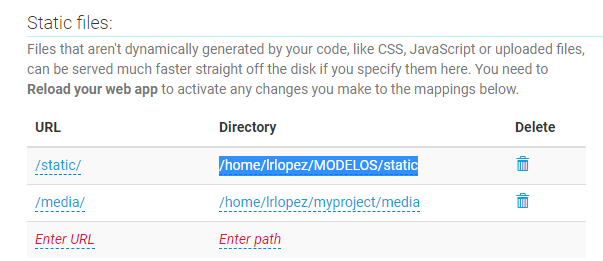
De forma tal que nos queda:



d. Configuracion LOCAL de Archivos estáticos:



e Luego, debajo de STATIC\_URL **agregamos** la variable **STATIC\_ROOT** con la configuracion de PyAny:



**settings.py**

STATIC\_ROOT= '/home/lrlopez/MODELOS/static/'

De forma tal que nos queda:

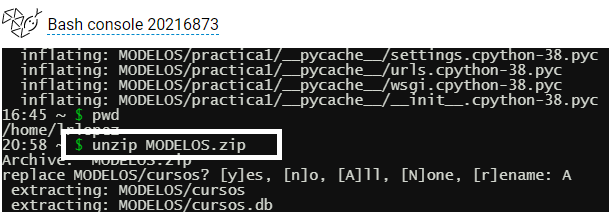
STATIC\_URL = '/static/'

STATIC\_ROOT= '/home/lrlopez/MODELOS/static/'

f. Volvemos a subir el proyecto zipeado a PythonAnywhere:



Y lo descomprimimos:



Vemos en FILES el proyecto actualizado.

g. TRATAMIENTO DE LOS STATIC FILES. Los colocamos a nivel de Proyecto.

Desde donde se encuentra **manage.py** dentro de MODELOS, vamos a la consola de Linux:



Y ejecutamos:

python3.7 manage.py collectstatic

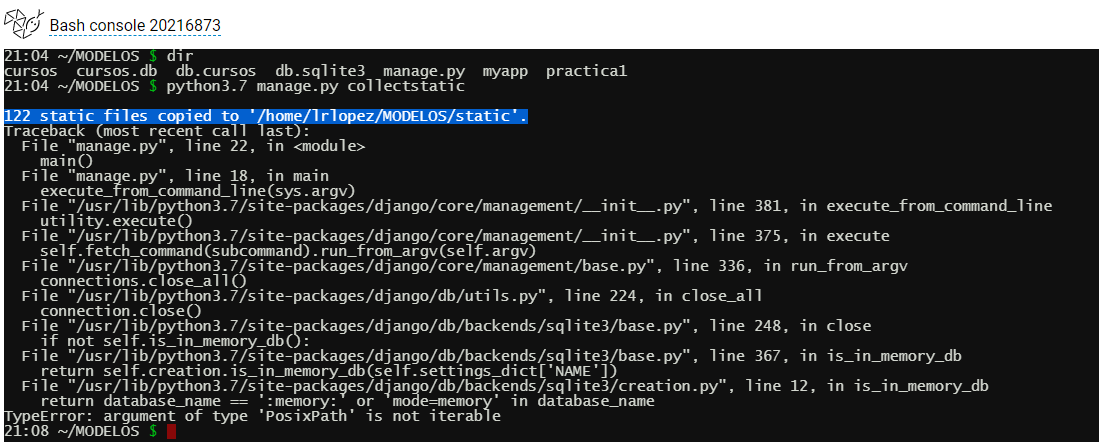
(estar atentos a la versión de Python que tengamos. Si al inicio se hizo un proyecto django 3.7 seguir con esa versión.)

**Nota (Troubleshooting 1)**: Si se llega a dar el siguiente error:

[***return database\_name == ':memory:' or 'mode=memory' in database\_name TypeError: argument of type 'PosixPath' is not iterable***](https://stackoverflow.com/questions/64185724/return-database-name-memory-or-mode-memory-in-database-name-typeerror)

[aplicar el siguiente workarround: https://stackoverflow.com/questions/64185724/return-database-name-memory-or-mode-memory-in-database-name-typeerror](file:///C:\Users\lrlopez\AppData\Roaming\Microsoft\Word\aplicar%20el%20siguiente%20workarround:%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20https:\stackoverflow.com\questions\64185724\return-database-name-memory-or-mode-memory-in-database-name-typeerror)

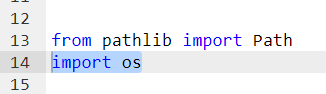
Vemos que mas allá del error, efectuo la copia:



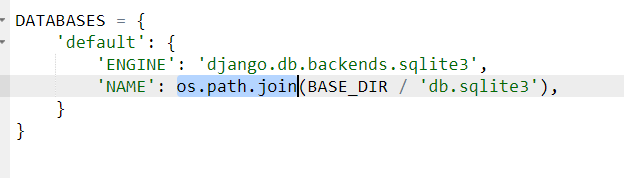
**COLLECTSTATIC** como estamos viendo, copia la carpeta **static** desde la aplicación al proyecto.

Para evitar el error previo hacemos el siguiente agregado en **settings.py** sabiendo que esto es momentaneo ya que utilizaremos la base MYSQL:

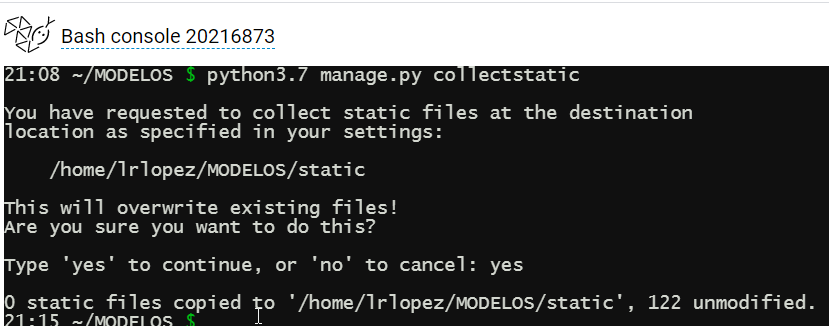
Importamos os



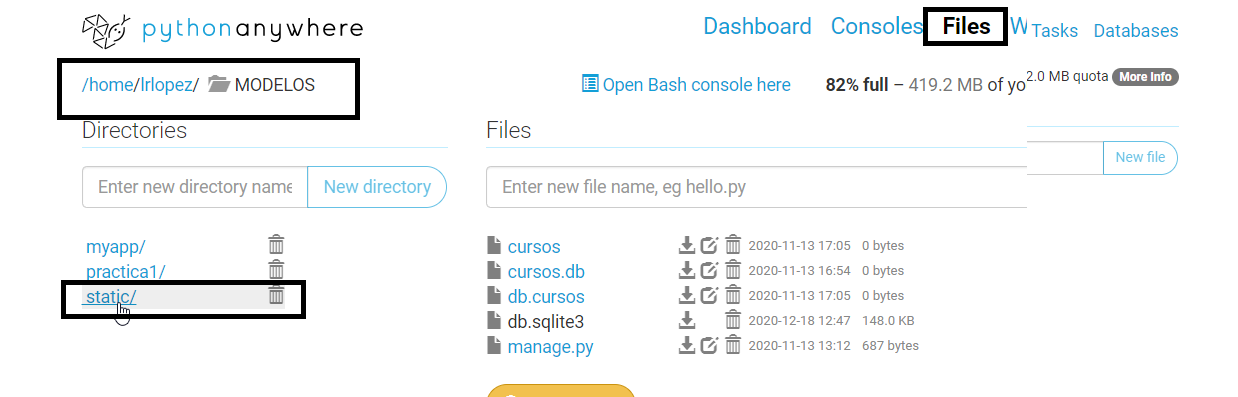
Y

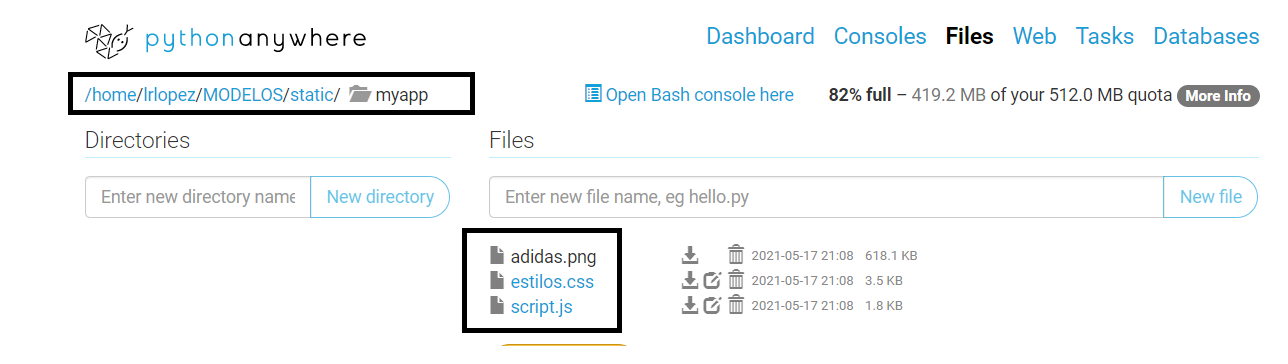


Y al ejecutar nuevamente el comando vemos que no vuelve a arrojar el error:

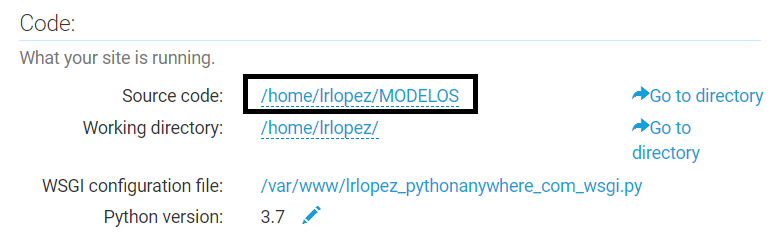


Creandose la carpeta **static** a nivel de carpeta de proyecto:



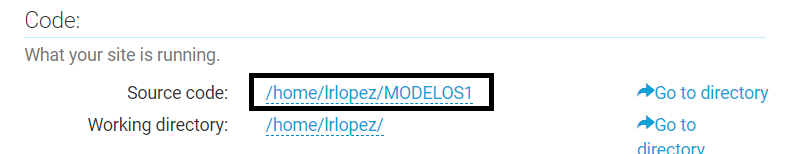


h. SOLAPA WEB, SOURCE CODE: Le indicamos a Python Anywhere donde se encuentra nuestro proyecto con el codigo fuente.



Estar atentos por si hay que efectuar alguna modificación, por ejemplo si el proyecto se llamara MODELOS1 tendriamos que hacer lo siguiente:

Home/usuario/MODELOS1 lo corregimos:



Donde “usuario” es el usuario de cada uno.

**VIRTUALEnv**

1. Creamos un entorno virtual donde le instalaremos lo necesario para correr nuestro proyecto django:

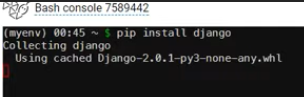
a. Abrimos una consola (Open bash console here) y nos ubicamos en el directorio /home/nombre\_usuario/. En mi caso es /home/lrlopez.

b. Y ejecutamos mkvirtualenv --python=/usr/bin/python3.7 myenv

Creara un entorno virtual.

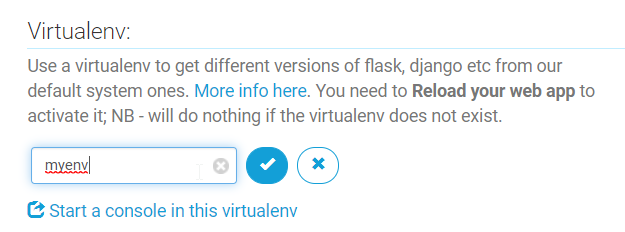
IMPORTANTE CHEQUEAR LA VERSION.

c. Instalamos django en nuestro entorno virtual **pip install django:**

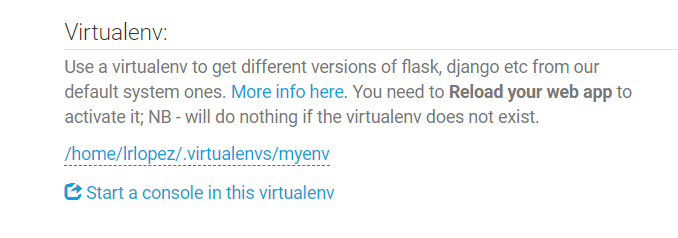
****

2. **SOLAPA WEB**, seteamos el entorno virtual.

a. Tomamos el nombre que utilizamos para nuestro entorno virtual **myenv** y lo colocamos en:

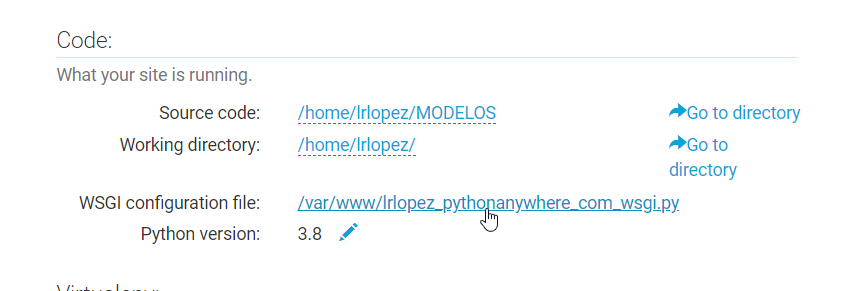


Nos queda:



Python Anywhere lo autocompleta.

**3. SOLAPA WEB –** Configuramos WSGI



1. Click sobre el link. Este archivo contiene la configuración necesaria para servir nuestra aplicación web en PythonAnywhere.
2. Borramos todo y colocamos el siguiente codigo:

import os

import sys

path = '/home/lrlopez/MODELOS' #hasta un nivel anterior de la carpeta que contiene settings.py

if path not in sys.path:

sys.path.append(path)

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'practica1.settings' #solo se pone la carpeta que contiene a settings.py

from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

import django.core.handlers.wsgi

application = django.core.handlers.wsgi.WSGIHandler()

O bien podemos utilizar el siguiente:

import os

import sys

# add your project directory to the sys.path

project\_home = '/home/lrlopez/MODELOS'

if project\_home not in sys.path:

sys.path.insert(0, project\_home)

# set environment variable to tell django where your settings.py is

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'practica1.settings'

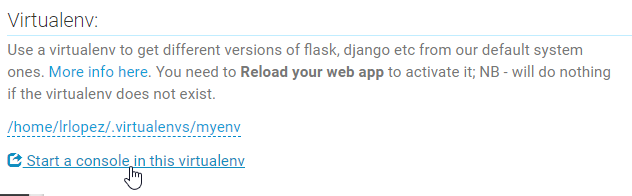
# serve django via WSGI

from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

La diferencia es que no se agregaron las ultimas 2 lineas.

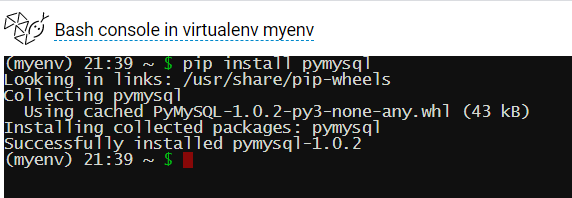
1. Solapa **WEB** -> Reload button.
2. En nuestro entorno virtual **myenv** (vamos a la consola virtual). Podemos estartearla desde la solapa **WEB**:



**Configuracion BBDD MySQL**

1. Instalamos el cliente de mysql pip install pymysql

(desde /home/usuario)



1. Vamos a **FILES -> settings.py**

Comentamos las siguientes líneas ya que no usaremos la base de SQLITE3:

DATABASES = {

'default': {

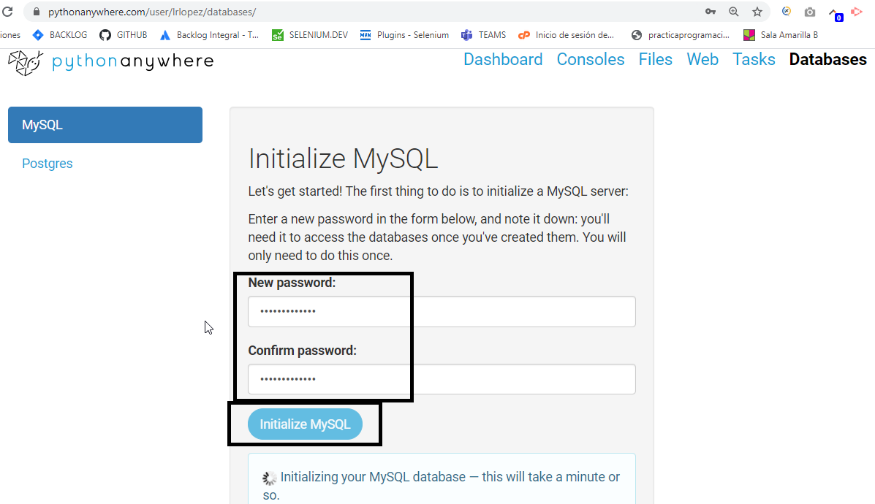
'ENGINE': 'django.db.backends.sqlite3',

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

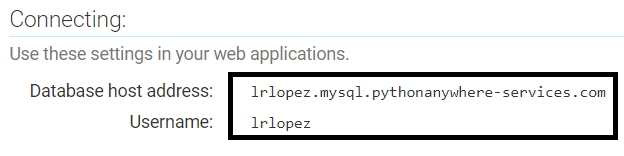
}

}

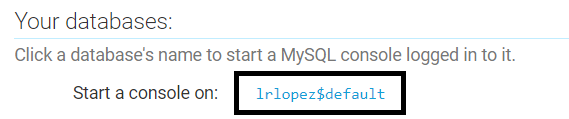
1. Creamos un servidor de BBDD **MySQL** en el hosting. Vamos a la pestaña **Databases** y elegimos una contraseña:



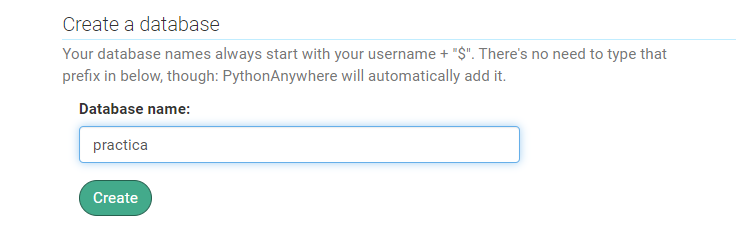
e. Las dos primeras filas indican la dirección donde se encuentra alojado nuestro servidor de BBDD:

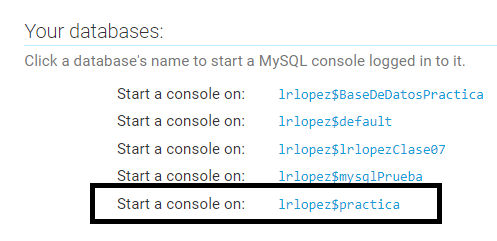


f. En la 3er fila vemos un enlace con el nombre de la BBDD creada por defecto:



O bien podemos crear una nueva:





g. FILES - SETTINGS.PY => configuramos la nueva BBDD:

DATABASES = {

    'default': {

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

        'HOST' : 'lrlopez.mysql.pythonanywhere-services.com',

        'USER' : 'lrlopez',

        'PASSWORD' : 'tar.nuf.roi-2',

        'NAME': 'lrlopez$default',

        'CHARSET' : 'utf8',

    },

}

O bien si hemos creado la BBDD practica:

DATABASES = {

'default': {

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

'HOST' : 'lrlopez.mysql.pythonanywhere-services.com',

'USER' : 'lrlopez',

'PASSWORD' : 'tar.nuf.roi-2',

'NAME': 'lrlopez$practica',

'CHARSET' : 'utf8',

},

}

1. Nos vamos al entorno virtual creado y ejecutamos:

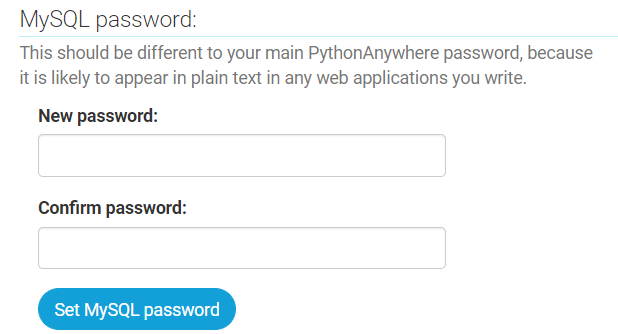
**python3.7 manage.py makemigrations**

**python3.7 manage.py migrate**

En caso de que nos aparezca el siguiente error:

*File "/usr/lib/python3.8/site-packages/MySQLdb/connections.py", line 166, in \_\_init\_\_ super(Connection, self).\_\_init\_\_(\*args, \*\*kwargs2)django.db.utils.OperationalError: (1045, "Access denied for user 'lrlopez'@'10.0.0.187' (using password: YES)")*

Resetear la password desde **Python Anywhere**, databases:



Si nos arroja el siguiente warning:

WARNINGS:?: (mysql.W002) MySQL Strict Mode is not set for database connection 'default' HINT: MySQL's Strict Mode fixes many data integrity problems in MySQL, such as data truncation upon insertion, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/en/2.2/ref/databases/#mysql-sql-mode

Debemos agregar **OPTIONS** en la configuración de la BBDD:

DATABASES = {

'default': {

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

'HOST' : 'lrlopez.mysql.pythonanywhere-services.com',

'USER' : 'lrlopez',

'PASSWORD' : 'tar.nuf.roi-2',

'NAME': 'lrlopez$default',

'CHARSET' : 'utf8',

'OPTIONS': {

'sql\_mode': 'traditional',

}

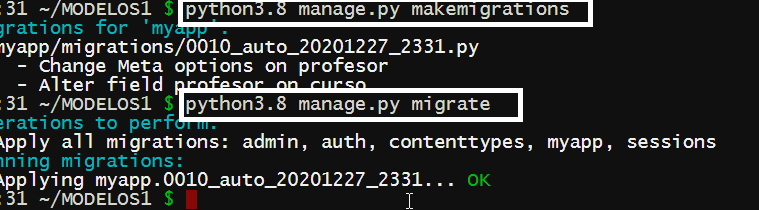
},

}

Para mayor información ir al siguiente link:

<https://docs.djangoproject.com/en/2.2/ref/databases/#mysql-sql-mode>

j. Si sale bien la migración debemos tener una salida de este tipo:



k. Por el contrario si obtenemos el siguiente error:

2021-05-19 22:02:11,079: Error running WSGI application

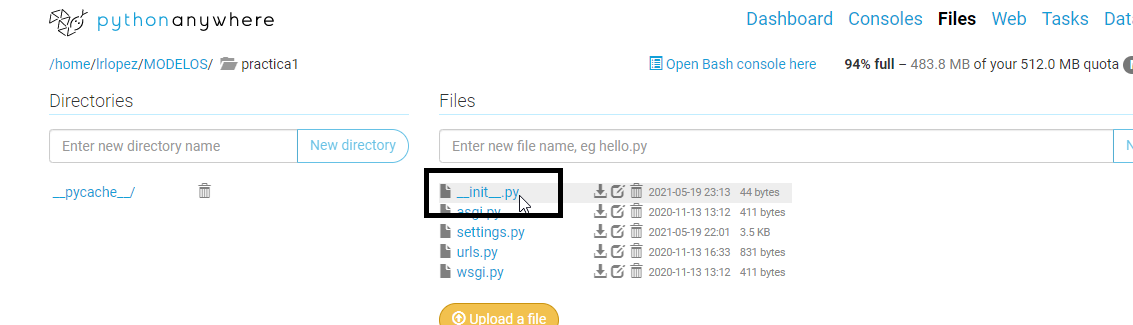
2021-05-19 22:02:11,080: django.core.exceptions.ImproperlyConfigured: Error loading MySQLdb module.

2021-05-19 22:02:11,080: Did you install mysqlclient?

2021-05-19 22:02:11,080: File "/var/www/lrlopez\_pythonanywhere\_com\_wsgi.py", line 15, in <module>

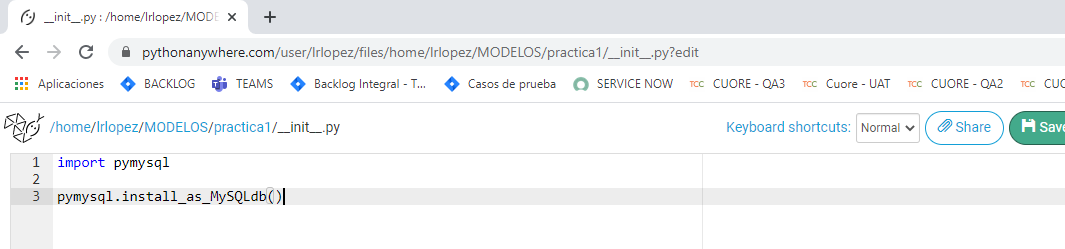
2021-05-19 22:02:11,080: application = get\_wsgi\_application()

**Solucion**: agregar las siguientes líneas al archivo \_\_init\_\_.py que se encuentra en el mismo path que **settings.py**:



import pymysql

pymysql.install\_as\_MySQLdb()



l. Luego vamos a **views.py** y hacemos la siguiente modificación:

**index**

def index ( request ):

return HttpResponse("Hola Mundo")

Sacamos el import de **reque**sts y agregamos el HttpResponse.

**L. SETTINGS.PY:** Agregamos el siguiente seteo

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

Para evitar el Warning:

myapp.Curso: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'. HINT: Configure the DEFAULT\_AUTO\_FIELD setting or the MyappConfig.default\_auto\_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.myapp.Profesor: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.

**MIGRAMOS NUEVAMENTE pero esta vez solo la aplicación myapp:**

(myenv1) 00:41 ~/MODELOS $ **python3.7 manage.py makemigrations myapp**

No changes detected in app 'myapp'

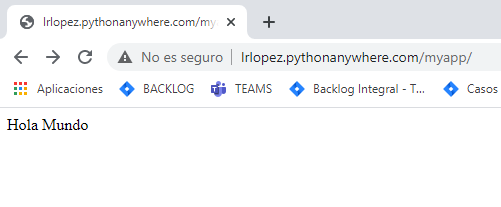
(myenv1) 00:44 ~/MODELOS $ **python3.7 manage.py migrate myapp**

Operations to perform: Apply all migrations: myappRunning migrations: Applying myapp.0001\_initial...

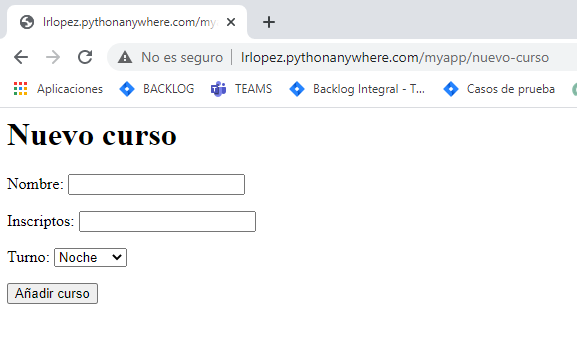
Esta operación puede tardar un poco

Recordar hacer un **Reload** en la solapa **WEB**.

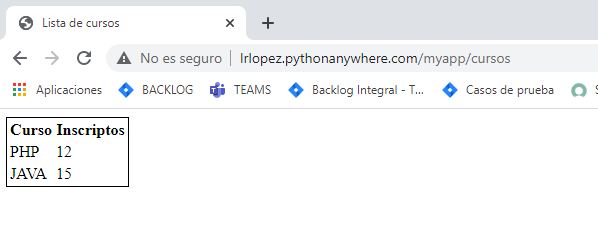
<http://lrlopez.pythonanywhere.com/myapp/>



Luego ingresamos un nuevo curso, vamos a <http://lrlopez.pythonanywhere.com/myapp/nuevo-curso>

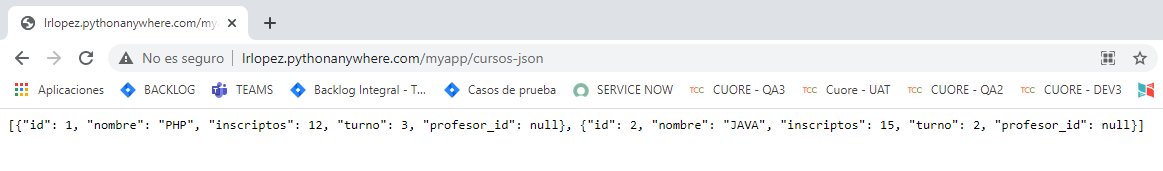


Ingresamos un curso determinado y vemos la salida de “cursos”



Y por ultimo probamos **cursos-json** <http://lrlopez.pythonanywhere.com/myapp/cursos-json>

Y obtenemos la salida correspondiente:



**INTEGRACION CON FRONTEND de Adidas**

l. Dado que nos habíamos importado un frontend elaborado con sus archivos estaticos:



Agregamos el siguiente codigo:

**models.py** agregamos

class Adidas(models.Model):

articulo = models.CharField("Articulo", max\_length=128)

talle = models.IntegerField("Talle")

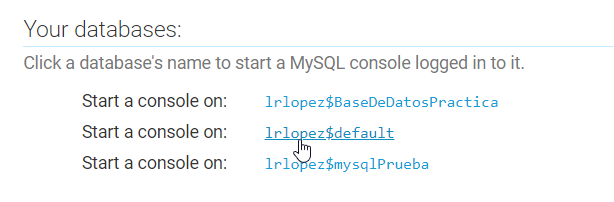
m. Migramos para que se genere la tabla en SQL.

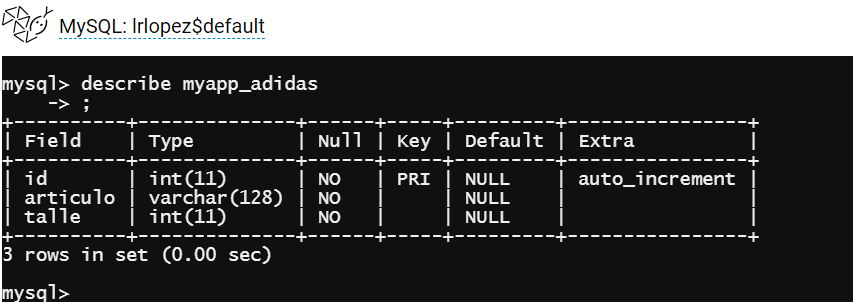
Vamos a **myenv** y escribimos:

python3.7 manage.py makemigrations

python3.7 manage.py migrate

Una vez que están OK las migraciones vamos a visualizar la tabla creada a través de la consola de SQL:





Nos muestra los 3 campos recientemente creados. Entonces ya podemos ir a la lógica.

n. Creamos el **FormularioAdidas** por medio del cual se va a poner a disposición los campos del modelo **Adidas**:

**forms.py**

from django.forms import ModelForm

from .models import Curso, Adidas

class FormularioCurso(ModelForm):

class Meta:

model = Curso

fields = ("nombre", "inscriptos", "turno")

class FormularioAdidas(ModelForm):

class Meta:

model = Adidas

fields = ("articulo", "talle")

o. Creamos una función que tome el modelo **Adidas** y por medio de la cual al seleccionar un talle y oprimir el botón **Purchase** persistan ambos datos en la BBDD (myapp\_adidas):

from django.shortcuts import render

from django.http import JsonResponse

from . import forms

from django.http import HttpResponseRedirect

from django.urls import reverse

from .models import Curso, Adidas

from django.http import Http404

…

def compraZapatillas(request):

if (request.method == 'POST'):

if(request.POST['boton1'] == '39'):

zapatilla = Adidas(articulo="ADIDAS ZX", talle = '39')

zapatilla.save()

boton1 = request.POST['boton1']

ctx = {

'boton1' : boton1,

}

return render(request, 'myapp/nuevo\_adidas.html', ctx)

if(request.POST['boton2'] == '40'):

zapatilla1 = Adidas(articulo="ADIDAS ZX", talle = '40')

zapatilla1.save()

boton2 = request.POST['boton2']

ctx1 = {

'boton2' : boton2,

}

return render(request, 'myapp/nuevo\_adidas.html', ctx1)

if(request.POST['boton3'] == '42'):

zapatilla2 = Adidas(articulo="ADIDAS ZX", talle = '42')

zapatilla2.save()

boton3 = request.POST['boton3']

ctx2 = {

'boton3' : boton3,

}

return render(request, 'myapp/nuevo\_adidas.html', ctx2)

if(request.POST['boton4'] == '44'):

zapatilla3 = Adidas(articulo="ADIDAS ZX", talle = '44')

zapatilla3.save()

boton4 = request.POST['boton4']

ctx3 = {

'boton4' : boton4,

}

return render(request, 'myapp/nuevo\_adidas.html', ctx3)

ctx = {

'boton1' : ' ',

'boton2' : ' ',

'boton3' : ' ',

'boton4' : ' '

}

return render(request, 'myapp/nuevo\_adidas.html')

def listaComprada (request):

zapatillas = Adidas.objects.all()

ctx = {"cursos": zapatillas}

return render(request, "myapp/salidaAdidas.html", ctx)

p. Agregamos la asociación correspondiente en **urls.py** de la aplicación:

from django.urls import path

from . import views

urlpatterns = [

path( "" , views.index, name = "index" ),

path("cursos", views.cursos, name="cursos"),

path("nuevo-curso", views.nuevo\_curso, name="nuevo\_curso"),

path("cursos-json", views.cursos\_json, name="cursos\_json"),

path("curso", views.curso, name="curso"),

path("adidas", views.compraZapatillas, name="compraZapatillas"),

path("salidaAdidas", views.listaComprada, name="salidaAdidas"),

]

q. Agregamos el template **nuevo\_adidas.html**:

<!DOCTYPE html>

<html lang="en">

{% load static %}

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>3d Card Effect</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500&display=swap" rel="stylesheet">

<link rel="stylesheet" type="text/css" href='{% static "myapp/estilos.css" %}'>

</head>

<body>

<div class="container">

<div class="card">

<div class="sneaker">

<!-- 0. Agregamos el circuilo que estara debajo de la imagen -->

<div class="circle"></div>

<!-- 0.1 Agregamos la zapilla que se visualizara arriba del circulo -->

<!--<img src="img/adidas.png" alt="adidas">-->

<img src='{% static "myapp/adidas.png" %}' alt="adidas">

</div>

<div class="info">

<!-- 1. Agregamos el nombre del producto -->

<h1 class="title" name="title" id="">Adidas ZX</h1>

<!-- 2. Agregamos la descripcion del producto -->

<h3>FUTURE-READY TRAINERS WITH WRAPPED BOOST FOR EXCEPTION CONFORT.</h3>

<!-- 3. Agregamos los botones para elegir los tamaños de las zapatillas -->

<div class="sizes">

<button type="submit" value="39" onclick="addToField('39', 'input1')">39</button>

<button type="submit" value="40" onclick="addToField('40', 'input2')">40</button>

<button type="submit" value="42" onclick="addToField('42', 'input3')">42</button>

<button type="submit" value="44" onclick="addToField('44', 'input4')">44</button>

</div>

<!-- 5. Agregamos el boton de compra -->

<div class="purchase">

<form action="{% url 'compraZapatillas' %}" method="post" >

{% csrf\_token %}

<input type="text" placeholder="39" name=boton1 hidden id="input1">

<input type="text" placeholder="40" name=boton2 hidden id="input2">

<input type="text" placeholder="42" name=boton3 hidden id="input3">

<input type="text" placeholder="44" name=boton4 hidden id="input4">

<button type="submit">Purchase</button>

</form>

</div>

</div>

</div>

</div>

<script>

function addToField(c, input) {

var1 = document.getElementById("input1");

var1.value = '';

var2 = document.getElementById("input2");

var2.value = '';

var3 = document.getElementById("input3");

var3.value = '';

var4 = document.getElementById("input4");

var4.value = '';

document.getElementById(input).value = c;

console.log(var1.value);

console.log(var2.value);

console.log(var3.value);

console.log(var4.value);

};

</script>

<script src='{% static "myapp/script.js" %}'></script>

</body>

</html>

Notese que tenemos dos script, uno que hace referencia al archivo externo **script.js** para darle vida al estilo. Y otro interno al template que se encarga de ponerle los valores de los botones del html a los input type text del formulario que luego se enviara al backend de Django. Todos los datos viajan en el **request**.

Ver un debug del **request** en el proyecto local:



r. El resto de los archivos deben quedar como estaban, es decir:

**script.js , estilos.css y adidas.png**

s. Si se diera el siguiente **error**:

**Error:**

django.db.utils.OperationalError: (1060, "Duplicate column name 'talle'")

**Causa:**

Seguramente al modificar un modelo se olvido migrarlo y al generar una columna nueva o hacer una nueva migracion arroja el presente error.

<https://stackoverflow.com/questions/36016485/duplicate-column-name>

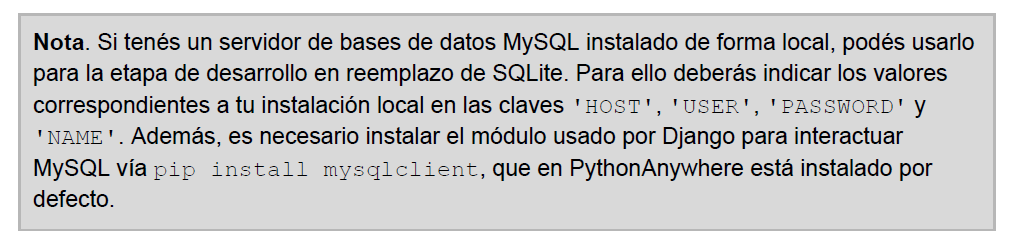
**Solucion (desde el entorno virtual):**

find . -path "\*/migrations/\*.py" -not -name "\_\_init\_\_.py" -delete

find . -path "\*/migrations/\*.pyc" -delete

remueve todas las migraciones previas y las setea a cero de forma tal que se puede migrar nuevamente.

t. **NOTA AL MARGEN:** Configuracion de MySQL para entorno de desarrollo (**NO ES PYTHON ANYWHERE)**:

****

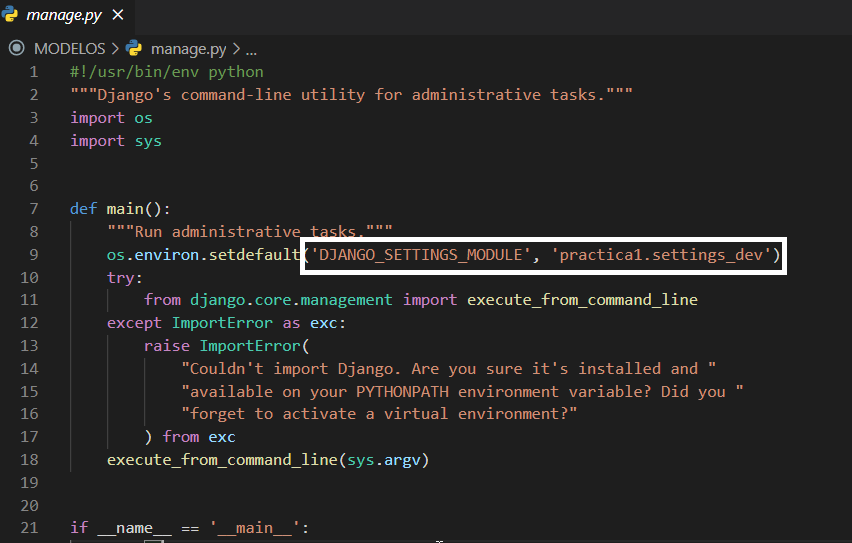
Como guía para la instalación les dejo el siguiente Word:



**CONFIGURACION ADICIONAL PARA DESARROLLO**

**MANERA 1**

Esta forma es muy sencilla, básicamente copiamos el contenido de **settings.py** en **settings\_dev.py** y en **manage.py** modificamos la variable de entorno DJANGO\_SETTINGS\_MODULEseteando el valor correspondiente:



En este caso correra con la configuración de **practica1.settings\_dev** la cual tiene

DEBUG = True

ALLOWED\_HOSTS = []

En cambio la versión de producción **settings.py** tiene

DEBUG = False

ALLOWED\_HOSTS = ['lrlopez.pythonanywhere.com']

Para mayor referencia (usando un script o un handler)

<https://stackoverflow.com/questions/10664244/django-how-to-manage-development-and-production-settings>

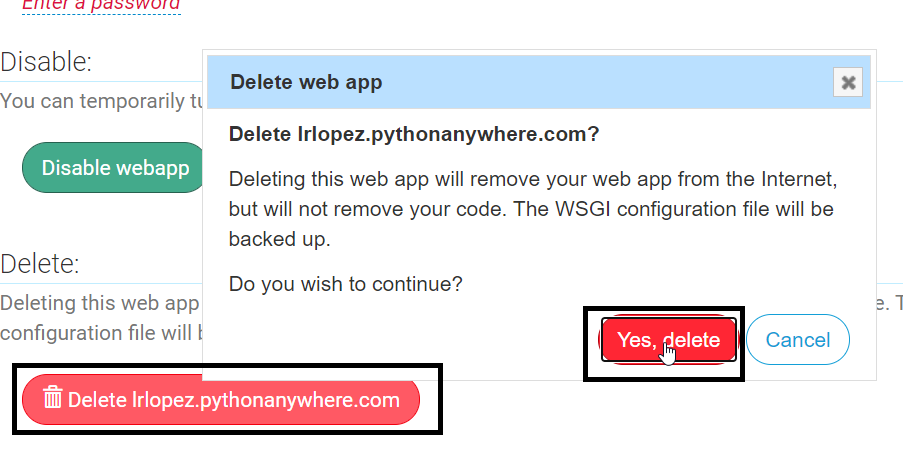
Por ultimo les dejo una ultima versión para correrlo **LOCALMENTE** y ver la salida de la lista de zapatillas en **salidaAdidas.html**.

****

**PROYECTO\_INTEGRADOR**



1. Ir a la pestaña **WEB** y presionamos el boton **Delete <nombre\_usuario>.pythonanywhere.com**:



**Nota:** Esta operación no elimina nuestro proyecto anterior de **Files**.

2. Partimos del siguiente código:



3. Cambiamos configuración en **settings.py**: DEBUG, ALLOWED\_HOSTS, DATABASES y STATIC\_ROOT

DEBUG = False

ALLOWED\_HOSTS = ['lrlopez.pythonanywhere.com']

DATABASES = {

    'default': {

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

        'HOST': 'lrlopez.mysql.pythonanywhere-services.com',

        'USER': 'lrlopez',

        'PASSWORD': '<tar.nuf.roi-2>',

        'NAME': 'lrlopez$default',

        'CHARSET': 'utf8',

    },

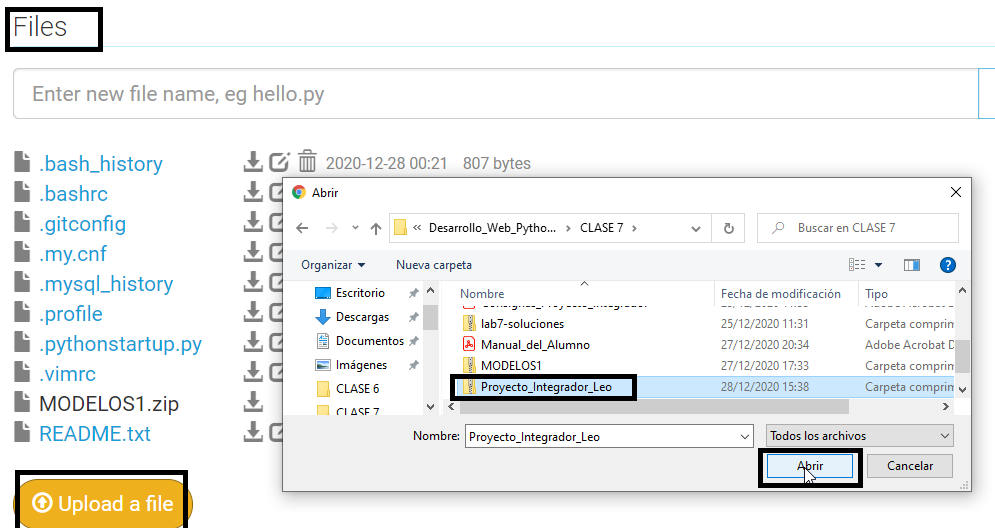
}

STATIC\_URL = '/static/'

STATIC\_ROOT = '/home/lrlopez/indumentariaonline/static'

4. Subimos nuestro proyecto a PythonAnywere:

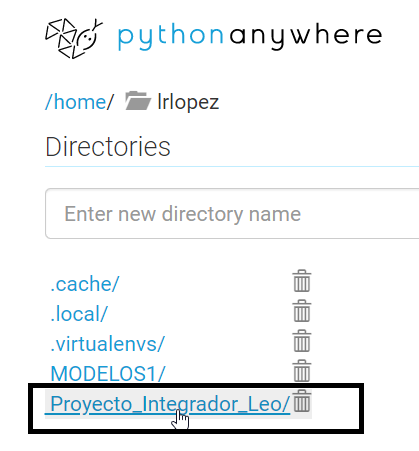
1. <https://www.pythonanywhere.com/> ingresamos
2. **Files => Upload a File** (subir el proyecto zipeado).

****

1. Vamos a la **consola** **bash** dentro de **PythonAnywhere** y descomprimimos el archivo

unzip Proyecto\_Integrador\_Leo.zip (ctrl+d para cerrar la consola)

1. Vemos como se ha creado el directorio:



5. Creamos **settings\_dev.py** para poder seguir trabajando en forma local (no se debe subir a PythonAnywhere)

DEBUG = True

ALLOWED\_HOSTS = []

DATABASES = {

    'default': {

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

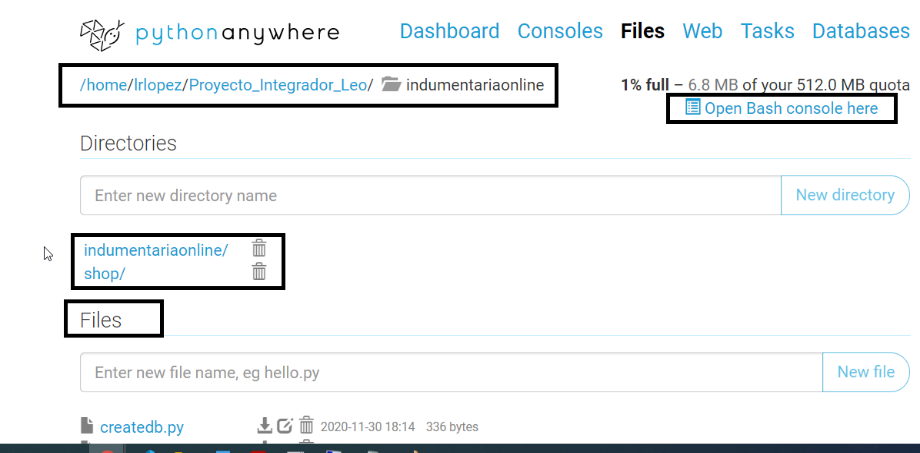
        'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

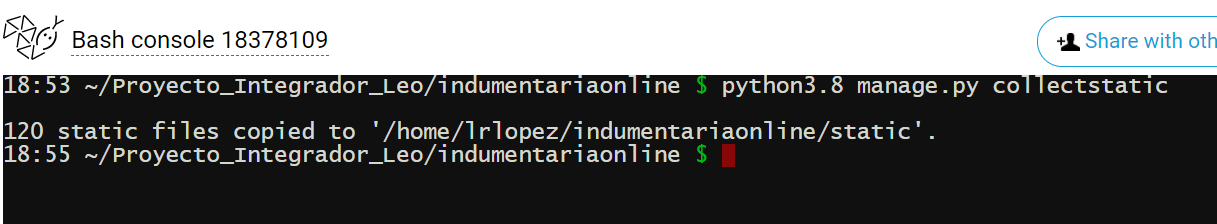
    }

}

STATIC\_ROOT = None

6. Para ejectuar **python3.8 manage.py collectstatic creamos** la carpeta **static** dentro de la carpeta general y nos ubicamos en la siguiente ruta:





7. luego efectuamos **python3.8 manage.py migrate**:

Si obtenemos este error:

super(Connection, self).\_\_init\_\_(\*args, \*\*kwargs2)django.db.utils.OperationalError: (1045, "Access denied for user 'lrlopez'@'10.0.0.187' (using password: YES)")

entonces actualizar la password en la sección de DATABASES.

Si obtienen un warning de STICKY MODE agregar las siguientes líneas a DATABASES en **settings.py:**

**'OPTIONS': {**

**'sql\_mode': 'STRICT\_TRANS\_TABLES',**

**}**

de modo tal que nos quede:

DATABASES = {

'default': {

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

'HOST': 'lrlopez.mysql.pythonanywhere-services.com',

'USER': 'lrlopez',

'PASSWORD': 'tar.nuf.roi-2',

'NAME': 'lrlopez$default',

'CHARSET': 'utf8',

'OPTIONS': {

'sql\_mode': 'STRICT\_TRANS\_TABLES',

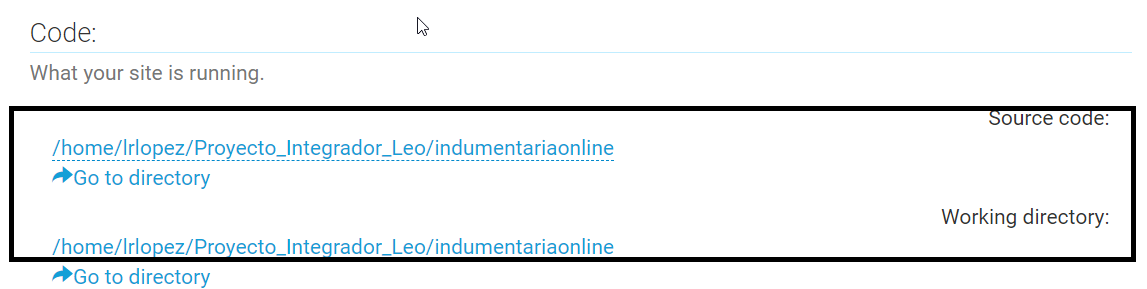
}

},

}

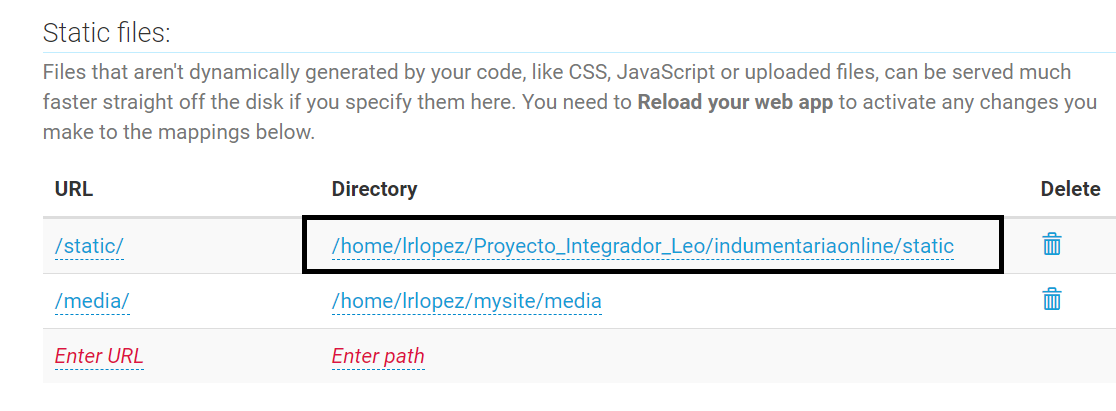
8. Hacemos las siguientes configuraciones en **WEB**:

Definimos donde esta ubicado el Source Code y en donde queremos tener nuestro Working directory.

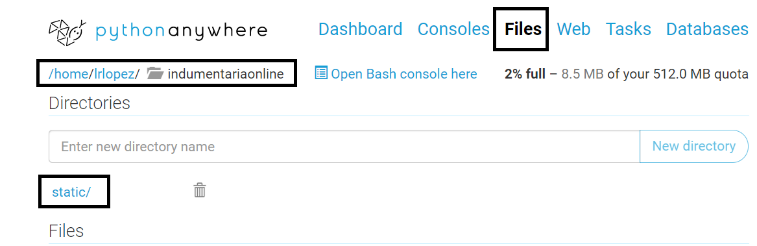


9. **ARCHIVOS ESTATICOS**:

Nos vamos al tag **WEB** y en la sección de **STATIC FILES** colocamos el siguiente path:

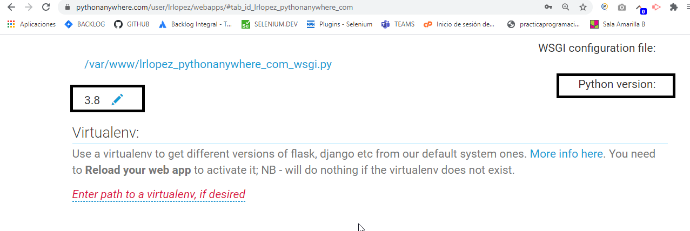


Esto lo que hace es sacar la carpeta **static** del nivel de aplicación y tenerla disponible a nivel directorios raíces (Ir a solapa FILES/indumentariaonline):



**ADICIONALES**

1.Para ver la versión de Python desde el tab **WEB** en PythonAnywhere debemos ir a:



2. Para ver los paquetes que tiene preinstalado Python en **PythonAnywhere** colocamos:

**pip3.8 list**



Probamos con las siguientes urls:

<http://lrlopez.pythonanywhere.com/>

<http://lrlopez.pythonanywhere.com/contacto>

La imagen la subimos haciendo caso de como levanta la imagen base.html.

**DEPLOYAR UNA APP DE DJANGO configurando una virtual machine**

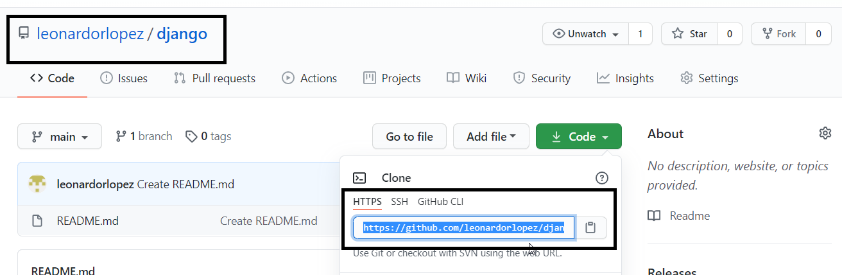
1.Vamos a **Consoles**. Clickeamos **Bash**.

2. Subimos nuestro proyecto a github.

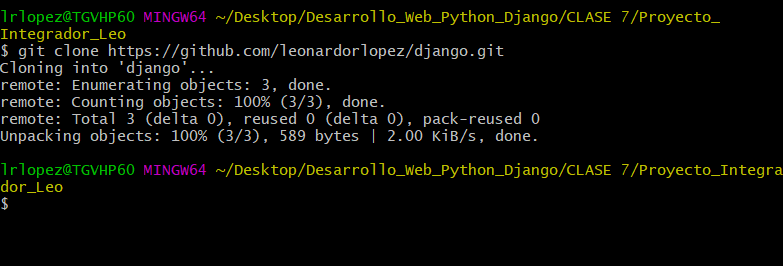
a. Nos dirijimos a la carpeta de nuestro proyecto, botón derecho, **GitBashHere**.

b. En la consola bash de git escribimos **git clone <pegamos aquí la url de github>**

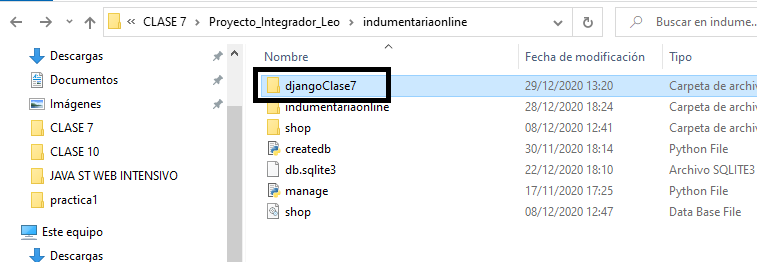
**https://github.com/leonardorlopez/adidasShoppingCart.git**

****

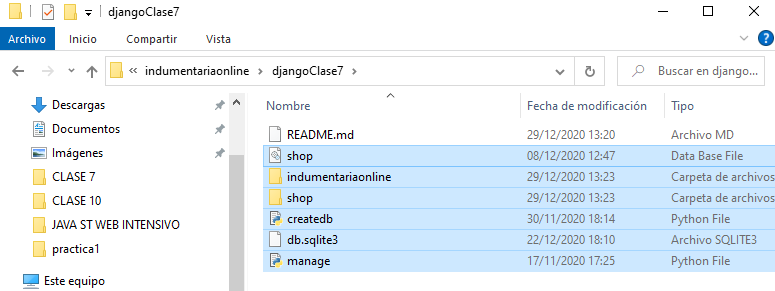
git clone https://github.com/leonardorlopez/django.git



c. Nos creo una carpeta dentro del directorio:

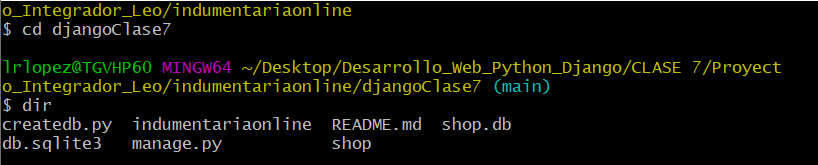


d.Entonces ahora tenemos que copiar todos los archivos del proyecto y pegarlos en **djangoClase7**:

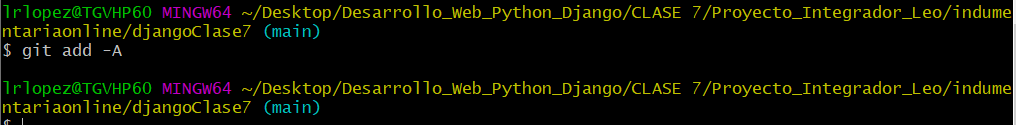


1. Nos movemos a la carpeta que se clono desde el gitbash:

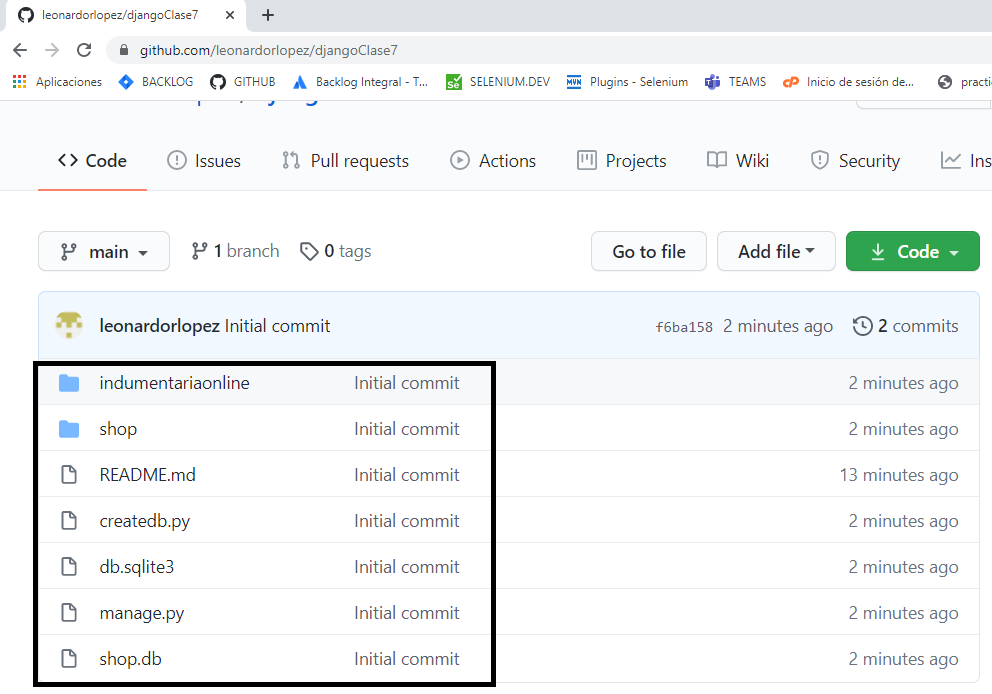
cd djangoClase7



1. **git add -A** para agregar todos los archivos:

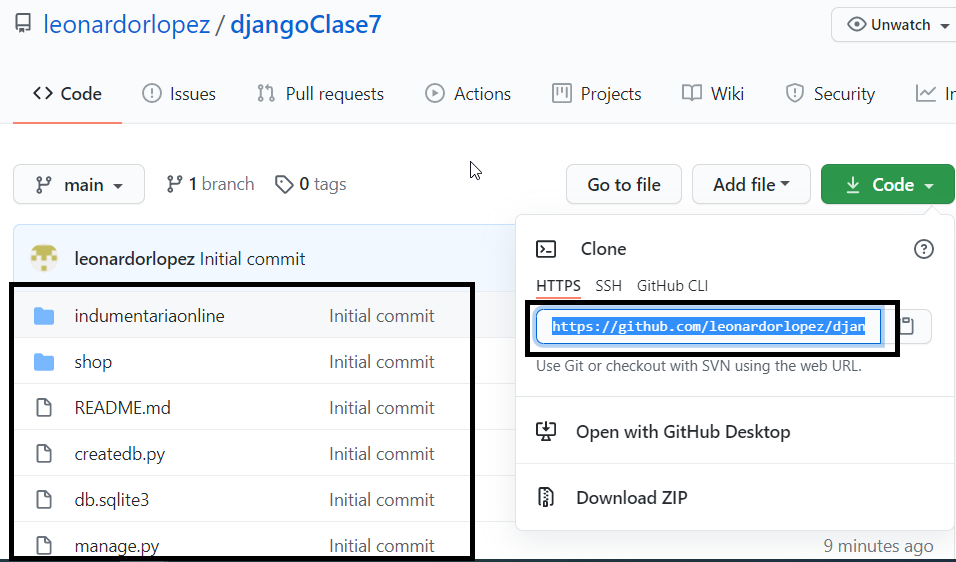


1. **git status** para ver el estado de los archivos a subir.
2. **git commit -m “Initial commit”** para crear la copia de los archivos
3. **git push** para subir los archivos.
4. Refrescamos github, <https://github.com/leonardorlopez/djangoClase7> y vemos los archivos subidos:

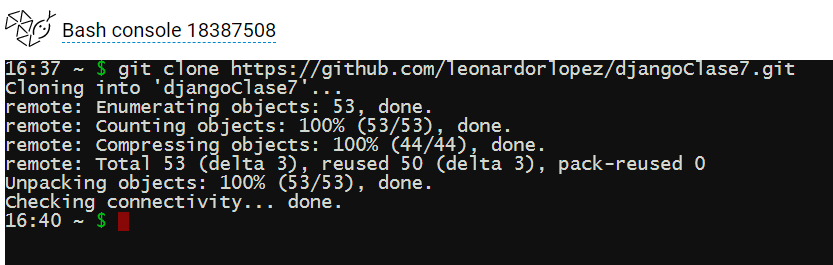


3. Subimos nuestro proyecto a PythonAnywhere:

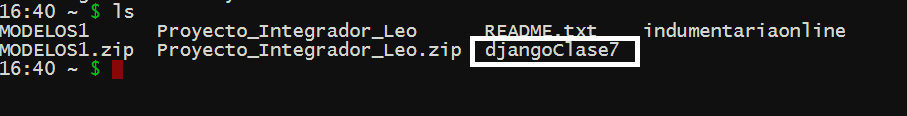
1. Vamos a **PythonAnywhere.**
2. Console => bash
3. Vamos a Github y clonamos la dirección donde tenemos nuestro proyecto. En mi caso <https://github.com/leonardorlopez/djangoClase7.git>



git clone https://github.com/leonardorlopez/djangoClase7.git



1. Con **ls** vemos el proyecto subido a **PythonAnywhere**:



1. cd djangoClase7
2. Creamos nuestro Virtual Environment.

mkvirtualenv --python=/usr/bin/python3.8 django-virtualenv

1. Activamos nuestro entorno virtual:

workon django-virtualenv

1. Una vez que nuestro entorno virtual se encuentra activado, debemos instalar Django dentro del mismo. Hay 2 formas de hacerlo. Una de ellas es preparar un requirements.txt el cual contendrá todas las bibliotecas de Django necesarias para nuestro proyecto.
2. Creamos el archivo requirements.txt el cual contendrá todas las bibliotecas que precisamos para trabajar con Django:



certifi==2020.6.20

chardet==3.0.4

defusedxml==0.6.0

Django==2.2.6

django-allauth==0.42.0

idna==2.10

cauthlib==3.1.0

python3-openid==3.2.0

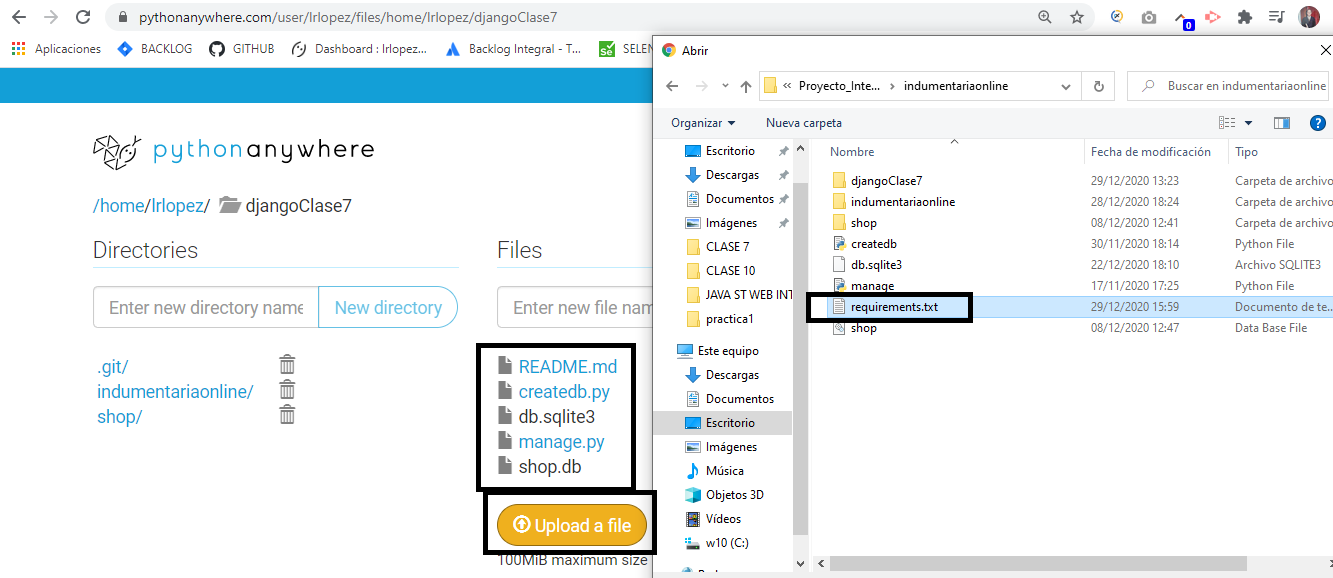
pytz==2020.1

requests==2.24.0

requests-oauthlib==1.3.0

urllib3==1.25.10

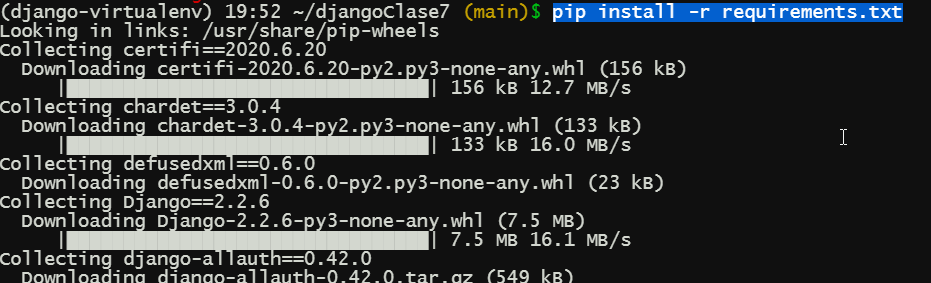
1. Lo subimos a **PythonAnywhere** al mismo nivel de **README.MD**.



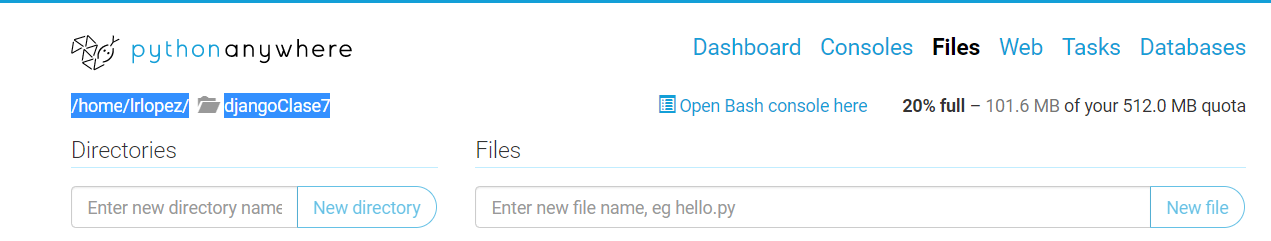
Nota: Esto lo que genera es que se deployen las librerías de Django necesarias para nuestro proyecto de PythonAnywhere.

1. Luego desde el bash de **PythonAnywhere** ejecutamos

pip install -r requirements.txt

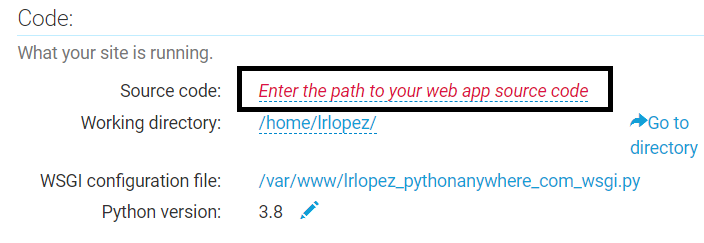


1. Una vez deployado el proyecto de Django en PythonAnywhere. Ahora es momento de setear el archivo WSGI.
2. Vamos a **PythonAnywhere**, clickeamos **Web**.
3. **Add a new web app**, **Next**. Como ya tenemos un proyecto Django en el servidor entonces seleccionamos **Manual Configuration**. Seleccionamos la versión de Python, Next. Con esto creamos nuestra Aplicación WEB.
4. Configuramos ciertos PATH del proyecto Web creado:
5. Vamos a **Files** y copiamos el siguiente path:

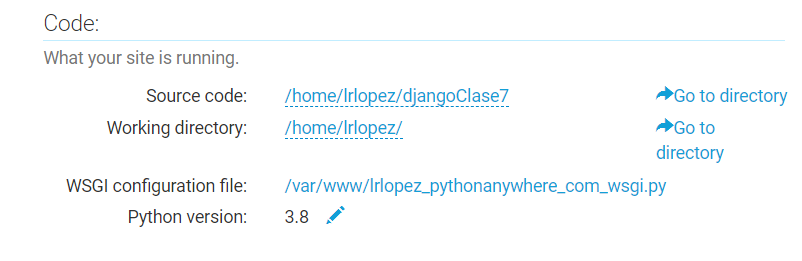


[/](https://www.pythonanywhere.com/user/lrlopez/files/)[home](https://www.pythonanywhere.com/user/lrlopez/files/home)/[lrlopez](https://www.pythonanywhere.com/user/lrlopez/files/home/lrlopez)/djangoClase7

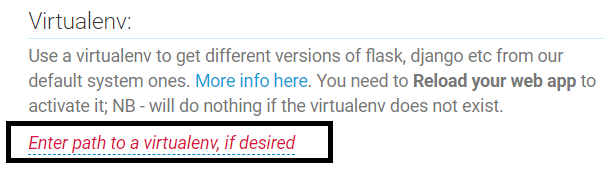
1. Volvemos a **WEB**, y pasteamos esa ruta en la sección **Code, source code**:



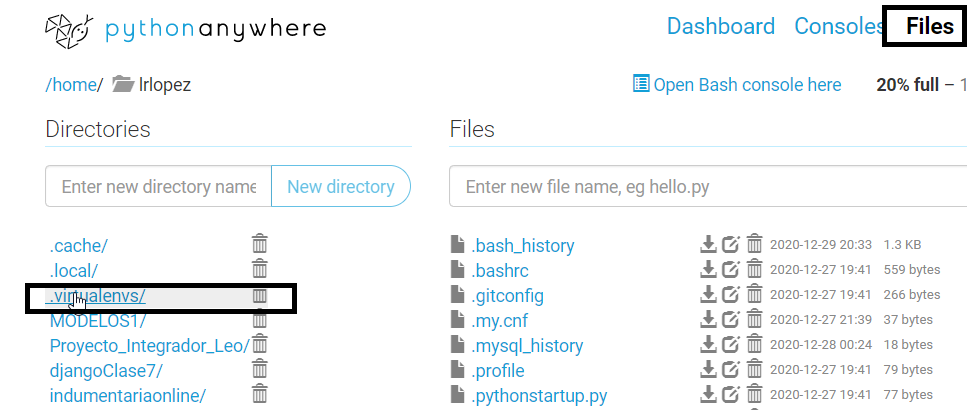
De forma tal que nos queda:



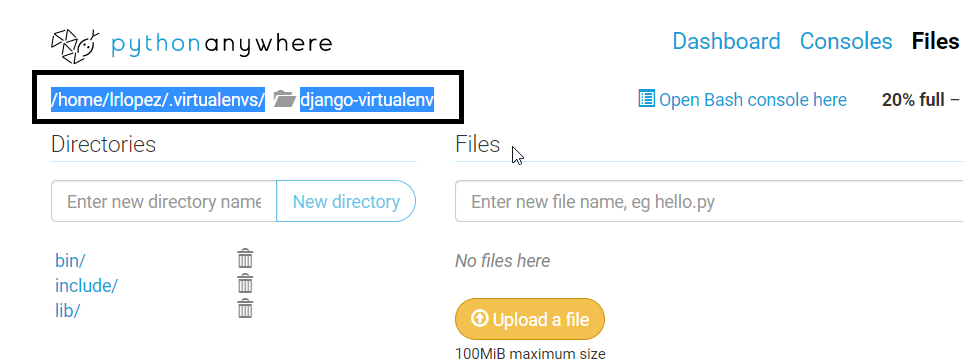
1. Para setear el **Virtualenv**, vamos a **FILES**,



1. Clickeamos en **.virtualenvs**

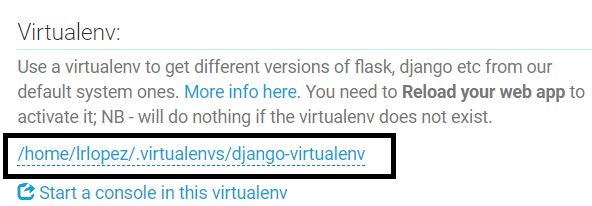


Luego **django-virtualenv/**. Y copiamos la siguiente ruta:

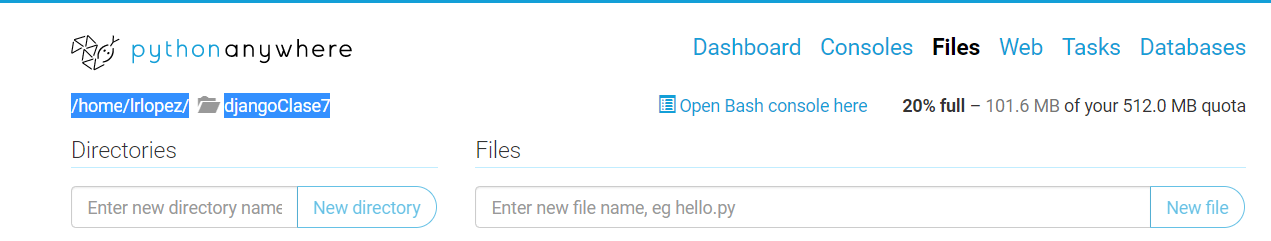


[/](https://www.pythonanywhere.com/user/lrlopez/files/)[home](https://www.pythonanywhere.com/user/lrlopez/files/home)/[lrlopez](https://www.pythonanywhere.com/user/lrlopez/files/home/lrlopez)/[.virtualenvs](https://www.pythonanywhere.com/user/lrlopez/files/home/lrlopez/.virtualenvs)/django-virtualenv

1. Vamos a **Web** y la pasteamos en **virtualenv**:



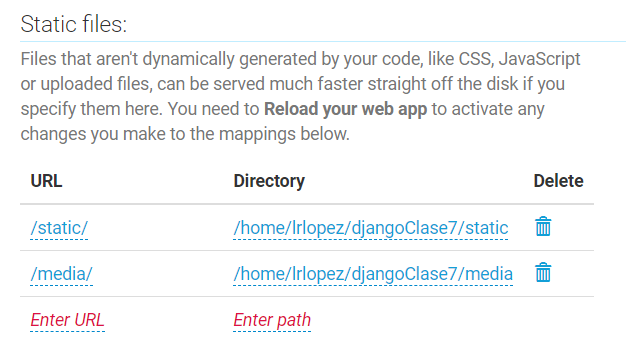
1. Agregamos el **Static Files**:
2. A la ruta del proyecto:



Le agregamos la palabra **static**.

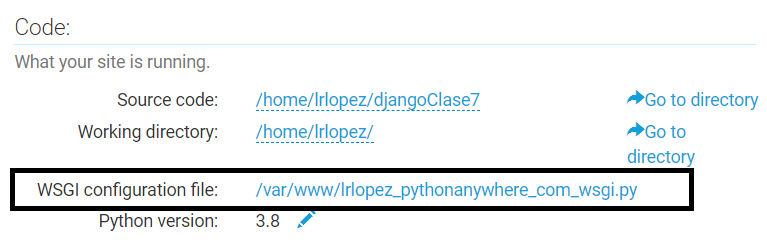
[/](https://www.pythonanywhere.com/user/lrlopez/files/)[home](https://www.pythonanywhere.com/user/lrlopez/files/home)/[lrlopez](https://www.pythonanywhere.com/user/lrlopez/files/home/lrlopez)/djangoClase7/static

De forma tal que nos queda:

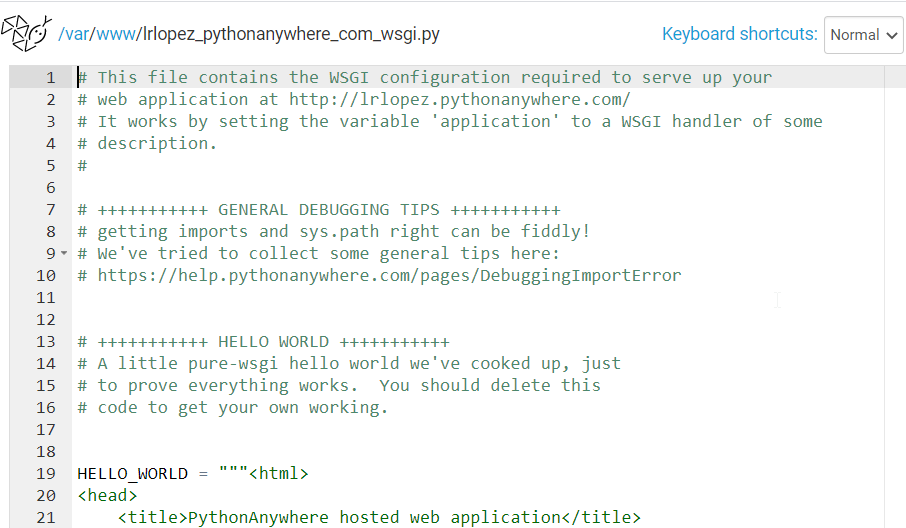


Con esto terminamos de configurar la parte Web.

1. Ahora vamos a configurar el archivo **WSGI**:
2. Vamos a la siguiente sección:



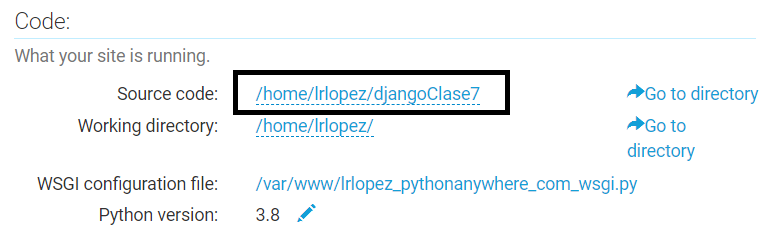
Y clickeamos sobre ese link.



1. Borramos todo el contenido del archivo:



Y ponemos el **Source code**



Y a su vez agregar el siguiente código

import os

import sys

#assuming you Django settings file is at

#'/home/myusername/mysite/mysite/settings.py'

path = '/home/lrlopez/djangoClase7'

if path not in sys.path:

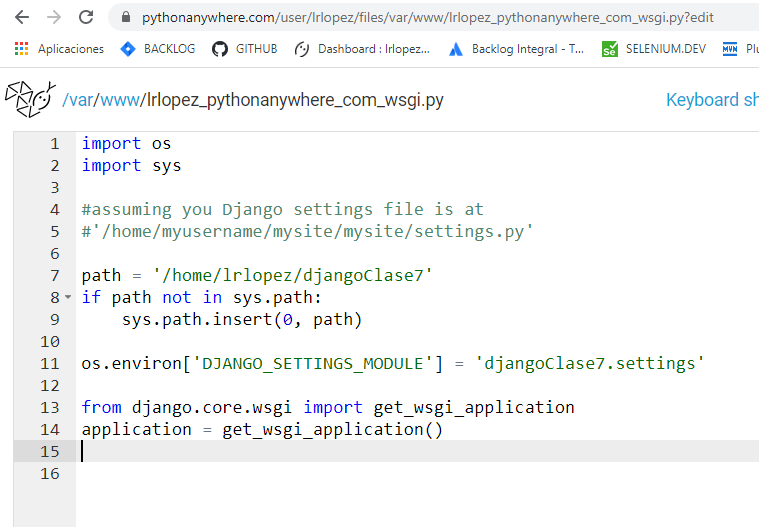
sys.path.insert(0, path)

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'djangoClase7.settings'

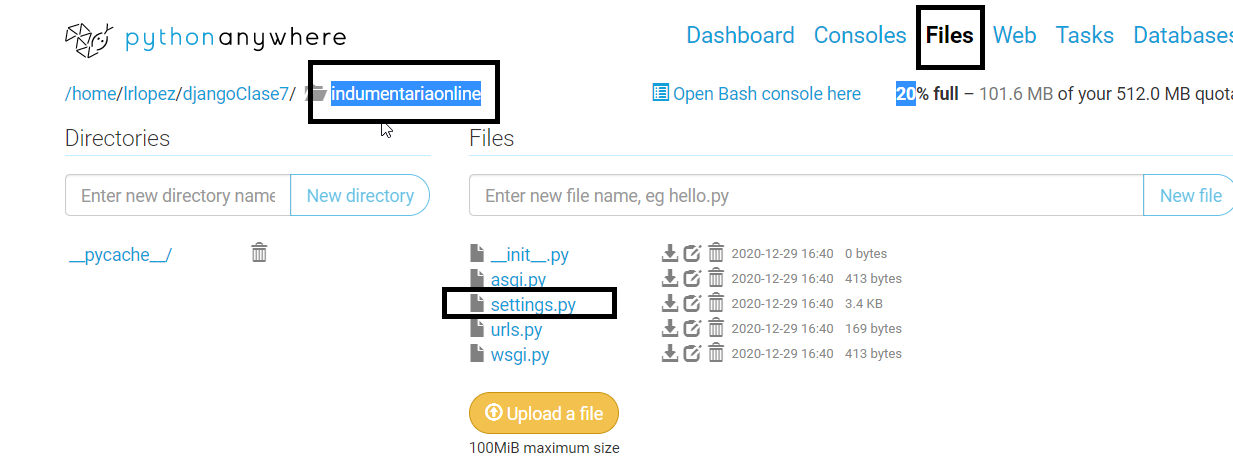
from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

de tal forma que nos queda:



A su vez, tomamos la carpeta que contiene a **settings.py**

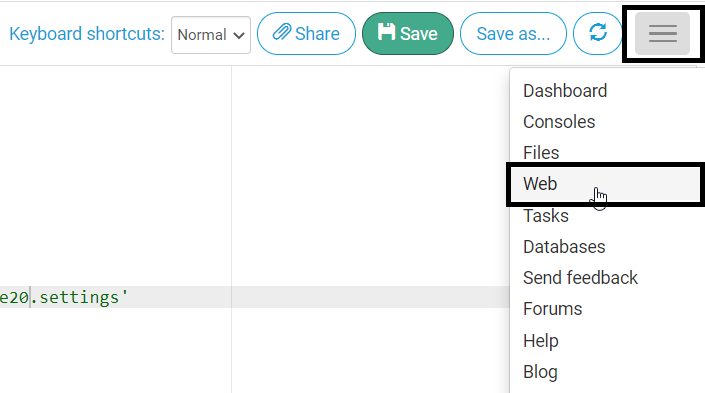


Y la colocamos en:

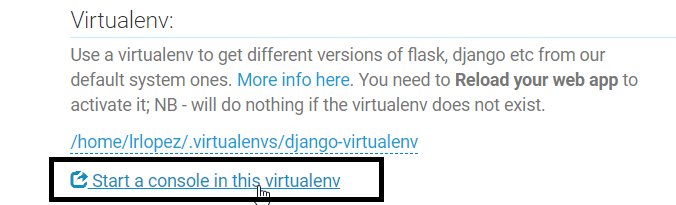


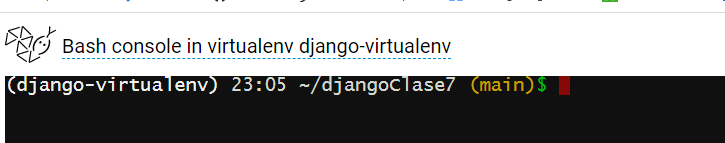
**Grabamos** el archivo.

1. Vamos a **Web** desde el menú derecho.



1. Vamos a **Console** y abrimos **bash console**.
2. Migramos nuestra información a la Base de datos. Para eso abrimos el bash de nuestro entorno virtual:



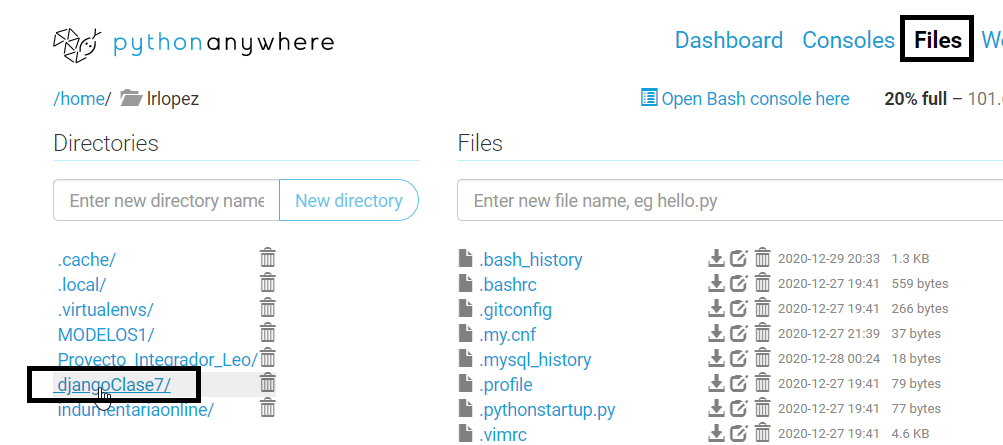


**python manage.py migrate**

The above exception was the direct cause of the following exception: Traceback (most recent call last): File "manage.py", line 22, in <module> main() File "manage.py", line 18, in main execute\_from\_command\_line(sys.argv) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/core/management/\_\_init\_\_.py", line 381, in execute\_from\_command\_line utility.execute() File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/core/management/\_\_init\_\_.py", line 357, in execute django.setup() File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/\_\_init\_\_.py", line 24, in setup apps.populate(settings.INSTALLED\_APPS) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/apps/registry.py", line 114, in populate app\_config.import\_models() File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/apps/config.py", line 211, in import\_models self.models\_module = import\_module(models\_module\_name) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/importlib/\_\_init\_\_.py", line 127, in import\_module return \_bootstrap.\_gcd\_import(name[level:], package, level) File "<frozen importlib.\_bootstrap>", line 1014, in \_gcd\_import File "<frozen importlib.\_bootstrap>", line 991, in \_find\_and\_load File "<frozen importlib.\_bootstrap>", line 975, in \_find\_and\_load\_unlocked File "<frozen importlib.\_bootstrap>", line 671, in \_load\_unlocked File "<frozen importlib.\_bootstrap\_external>", line 783, in exec\_module File "<frozen importlib.\_bootstrap>", line 219, in \_call\_with\_frames\_removed File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/contrib/auth/models.py", line 2, in <module> from django.contrib.auth.base\_user import AbstractBaseUser, BaseUserManager File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/contrib/auth/base\_user.py", line 47, in <module> class AbstractBaseUser(models.Model): File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/models/base.py", line 117, in \_\_new\_\_ new\_class.add\_to\_class('\_meta', Options(meta, app\_label)) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/models/base.py", line 321, in add\_to\_class value.contribute\_to\_class(cls, name) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/models/options.py", line 204, in contribute\_to\_class self.db\_table = truncate\_name(self.db\_table, connection.ops.max\_name\_length()) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/\_\_init\_\_.py", line 28, in \_\_getattr\_\_ return getattr(connections[DEFAULT\_DB\_ALIAS], item) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/utils.py", line 201, in \_\_getitem\_\_ backend = load\_backend(db['ENGINE']) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/utils.py", line 110, in load\_backend return import\_module('%s.base' % backend\_name) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/importlib/\_\_init\_\_.py", line 127, in import\_module return \_bootstrap.\_gcd\_import(name[level:], package, level) File "/home/lrlopez/.virtualenvs/django-virtualenv/lib/python3.8/site-packages/django/db/backends/mysql/base.py", line 17, in <module> raise ImproperlyConfigured(django.core.exceptions.ImproperlyConfigured: Error loading MySQLdb module.

Si aparece este error significa que no tenemos nuestro archivo local de configuración.

6.Vamos a **Files**:



Y entramos en **settings.py**, corroboramos que tenga la siguiente configuración activada:

DATABASES = {

'default': {

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

'NAME': os.path.join(BASE\_DIR / 'db.sqlite3'),

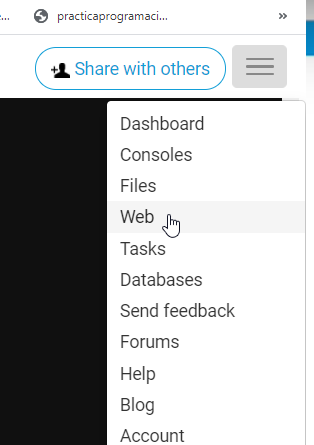
}

}

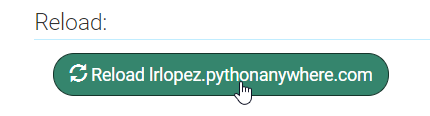
BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

Migramos nuevamente.

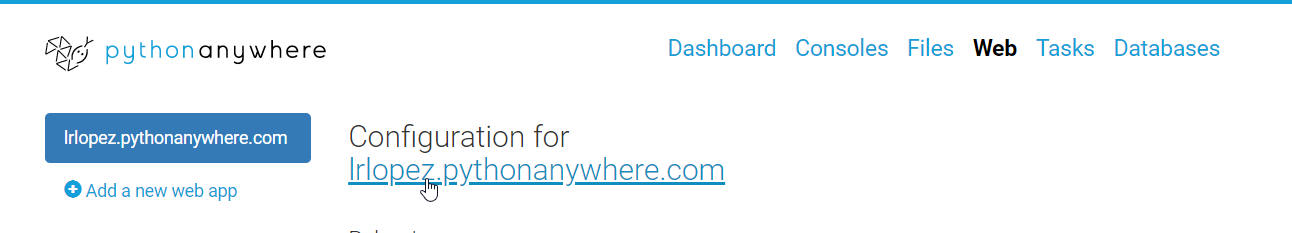
1. Vamos a **Web**.



Reload.



Y vamos a



1. Codigo utilizado:



WSGI son las siglas de **Web Server Gateway Interface**. Es una especificación que describe cómo se comunica **un servidor web**con **una aplicación web**, y cómo se pueden llegar a encadenar diferentes aplicaciones web para procesar una solicitud/petición (o request).

1. Partimos del siguiente código: Nombre del Proyecto **practica**



1. Subimos a github.

Nos ubicamos sobre la carpeta creada => git Bush here.

git clone <https://github.com/leonardorlopez/practica.git> (o bien el repositorio de GitHub en el que nos encontremos)

Vamos a la carpeta creada practica y copiamos las carpetas y archivos del proyecto dentro de esa carpeta.

cd <nombre\_carpeta>

1. **git add -A** para agregar todos los archivos:
2. **git status** para ver el estado de los archivos a subir.
3. **git commit -m “Initial commit”** para crear la copia de los archivos
4. **git push** para subir los archivos.
5. Refrescamos github y vemos los archivos subidos.

1. PythonAnywhere:
2. Bash console: git clone <dirección de github>
3. cd carpeta
4. mkvirtualenv --python=/usr/bin/python3.8 django-virtualenv
5. pip install Django
6. pip install django-import-export (instalamos los paquetes que vamos a precisar)
7. WEB: Add a new web app.
8. Manual configuration, next.
9. Virtualenv: [/home/lrlopez/.virtualenvs/django-virtualenv](https://www.pythonanywhere.com/user/lrlopez/webapps/)
10. Wsgi configuration file: damos click y pasteamos lo siguiente:

import os

import sys

path = '/home/lrlopez/practica/'

if path not in sys.path:

sys.path.append(path)

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'Ejercicio\_1\_Leo.settings'

from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

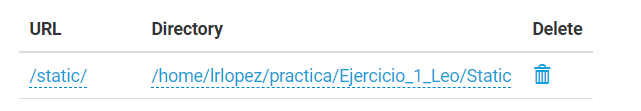
1. **Settings.py:**

ALLOWED\_HOSTS = ['lrlopez.pythonanywhere.com']

Y también debajo de STATIC agregamos

STATIC\_ROOT = '/home/lrlopez/practica/Ejercicio\_1\_Leo/Static'

1. Agregamos STATIC\_ROOT a STATIC dentro de Web.



1. **practica/myapp/views.py**=>

from django.shortcuts import render

# Create your views here.

def index(request):

return render(request, 'index.html', {})

1. **practica/myapp/urls.py**=>

from django.urls import path

from . import views

#from django.http import HttpResponse

#def okay(request):

# return HttpResponse('pretend-binary-data-here', content\_type='image/jpeg')

urlpatterns = [

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

#path('favicon.ico', okay),

]

1. Configuracion **Web**:



1. Tambien modificamos **urls.py** del Proyecto:

from django.conf.urls import url

from django.contrib import admin

from django.urls import path, include

from myapp import views

urlpatterns = [

path('myapp/', include('myapp.urls')),

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

**url('', views.index, name='index'),**

]

Y agregamos el siguiente **index.html** creando la carpeta **templates/myapp** dentro de myapp. Al html le ponemos cualquier cosa.

**PRACTICA\_2**

1.Partimos del siguiente código:



<http://127.0.0.1:8000/myapp/nuevo-curso>

1. Probamos ingresar un nuevo curso y vemos como se ve desde la base de datos.
2. Probamos las diferentes url del presente proyecto.

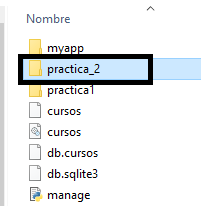
2.Migracion a Github.

a. Nos ubicamos sobre la carpeta de nuestro proyecto => git Bush here.

b. Vamos a Github y creamos un repositorio nuevo.

c. git clone <url del repositorio>

git clone <https://github.com/leonardorlopez/practica_2.git>



d.Vamos a la carpeta creada y copiamos las carpetas y archivos del

proyecto dentro de esa carpeta.

e.cd <nombre\_carpeta> entramos a la carpeta previamente creada.

cd practica\_2

1. **git add -A** para agregar todos los archivos:
2. **git status** para ver el estado de los archivos a subir.
3. **git commit -m “Initial commit”** para crear la copia de los archivos
4. **git push** para subir los archivos.
5. Refrescamos github y vemos los archivos subidos.

1. PythonAnywhere:
2. Bash console: git clone <dirección de github>

cit clone https://github.com/leonardorlopez/practica\_2.git

1. cd carpeta => cd practica\_2
2. mkvirtualenv --python=/usr/bin/python3.8 practica\_2-virtualenv
3. pip install Django
4. pip install django-import-export (instalamos los paquetes que vamos a precisar)
5. **WEB**: Add a new web app.
6. Manual configuration, next.
7. Virtualenv: [/home/lrlopez/.virtualenvs/practica\_2-virtualenv](https://www.pythonanywhere.com/user/lrlopez/webapps/) (donde practica\_2-virtualenv viene de la consola del entorno virtual creado previamente)



1. Wsgi configuration file: damos click y pasteamos lo siguiente:

import os

import sys

path = '/home/lrlopez/practica\_2/' #hasta un nivel anterior de la carpeta que contiene settings.py

if path not in sys.path:

sys.path.append(path)

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'practica1.settings' #solo se pone la carpeta que contiene a settings.py

from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

4.Archivos estáticos:

/home/lrlopez/MODELOS/static

5.CONFIGURACIONES PROPIAS DEL CODIGO DJANGO:

a. Agregar ALLOWED\_HOST y DEBUG pasarlo a False **settings.py**:

DEBUG = False

ALLOWED\_HOSTS =

#debajo de STATIC\_URL agregamos

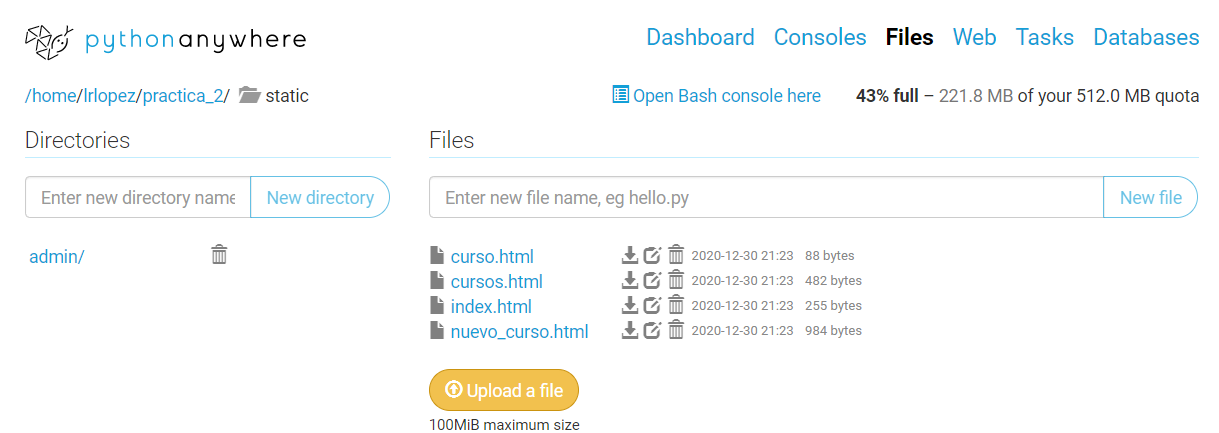
STATIC\_ROOT ='/home/lrlopez/practica\_2/static' #tiene que coincidir con la configuracion web

b.Corregimos **urls.py**:

1. Traemos los archivos estáticos a PythonAnywhere:

(practica\_2-virtualenv) 20:56 ~/practica\_2 (main)$ python3.8 manage.py collectstatic

1. Dentro de la carpeta /static/ creada en PythonAnywhere importar los html correspondientes:



1. A **urls.py** del proyecto le agregamos la siguiente línea:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path("myapp/", include("myapp.urls")),

**path('', include('myapp.urls')),**

]

1. Recordar ver el log de errores para ver de donde puede venir el error. Revisar los paquetes unused de los archivos del proyecto que pueden afectar que se vea o no la salida.
2. Configuracion WEB:



1. Probamos refrescando y las siguientes urls:

<http://lrlopez.pythonanywhere.com/nuevo-curso> (para cargar un nuevo curso)

<http://lrlopez.pythonanywhere.com/cursos> (para ver los cursos existentes)

<http://lrlopez.pythonanywhere.com/cursos-json> (para ver los cursos existentes en formato json)

**CONFIGURAR LA BASE DE DATOS:**

Abrimos una consola de mysql y colocamos lo siguiente:

mysql> CREATE TABLE cursos (

-> id serial NOT NULL PRIMARY KEY,

-> nombre varchar(200) not null,

-> inscriptos integer NOT NULL);

CREATE TABLE remera (

id serial NOT NULL PRIMARY KEY,

marca varchar(200) not null,

talle integer NOT NULL,

color varchar(200) not null,

lisa boolean,

genero integer not null);

Por ejemplo para el ejercicio integrador.

Insertamos valores

INSERT INTO cursos (id, nombre, inscriptos)

 VALUES (1, "Jose", 2);

 INSERT INTO cursos (id, nombre, inscriptos)

 VALUES (2, "Maria", 5);

 INSERT INTO cursos (id, nombre, inscriptos)

 VALUES (3, "Juana", 8);

select \* from cursos;

**settings.py bbdd**

DATABASES = {

'default': {

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

'HOST': 'lrlopez.mysql.pythonanywhere-services.com',

'USER': 'lrlopez',

'PASSWORD': 'tar.nuf.roi-2',

'NAME': 'lrlopez$default',

'CHARSET': 'utf8',

'OPTIONS': {

'sql\_mode': 'STRICT\_TRANS\_TABLES',

}

},

}

**url.py (Proyecto)**

urlpatterns = [

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

path("myapp/", include("myapp.urls")),

**path('', include('myapp.urls')),**

**]**

CREATE TABLE myapp\_curso(

id serial NOT NULL PRIMARY KEY,

nombre varchar(200) not null,

inscriptos integer NOT NULL,

turno varchar(200) not null

);