**Ejemplo 2**

**Sitio web: Registro Académico**

1. Ingresamos a nuestro directorio del proyectos Django:

**cd proyectoDjango**

1. Ingresamos al ambiente virtual creado en el proyecto anterior y lo activamos

**activate**

1. Ahora volvemos al raíz de la carpeta **proyectoDjango** comenzaremos un proyecto con:

**django-admin.exe startproject academido**

1. Cambiar la configuración en el archivo setting.py para colocar el lenguaje y horario correcto:

**LANGUAGE\_CODE = 'es-cl'**

**TIME\_ZONE = 'America/Santiago'**

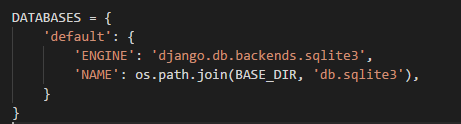
Agregar en la última línea:

**STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')**

Esto último nos servirá para agregar una ruta para los archivos estáticos.

Configurar una Base de Datos

1. La base de datos está configurada en **academico/settings.py** (por defecto sqlite3)



1. Para crear una base de datos se ejecuta (necesitamos estar en el directorio que contiene el archivo manage.py) y ejecutamos la siguiente instrucción:

**python manage.py migrate**

1. Vamos a iniciar el servidor, nuevamente debemos estar en el directorio que contiene el archivo manage.py y ejecutaremos la siguiente instrucción:

**python manage.py runserver**

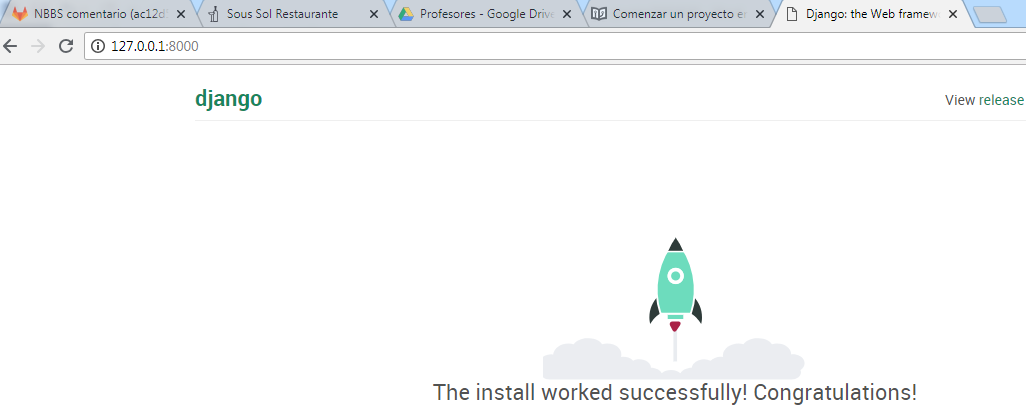
1. Para verificar si está “*en servicio*” el servidor debemos ir al browser y colocar:

<http://127.0.0.1:8000/>

Que es lo mismo que:

<http://localhost:8000/>

Se verá algo así:



El servidor web se apropiará de tu consola hasta que lo termines manualmente. Para seguir trabajando con más comandos tienes dos opciones:

1. Abres una nueva terminal
2. O frena el servidor web yendo a la consola en la que está corriendo y presionando Ctrl+C

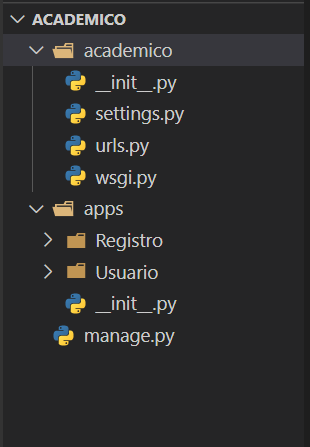
Aplicaciones del proyecto

1. El proyecto constará de dos aplicaciones:
   1. **Registro**
      1. La cual se encargará del CRUD de las tablas
      2. Consultas a tablas con filtros
   2. **Usuario**
      1. La cual se encargará del registro de usuarios del sistema
2. Antes de crear las aplicaciones le daremos un orden dentro del proyecto, por esto dentro de la carpeta raíz del proyecto crearemos la carpeta **apps en la cual irán todas las aplicaciones de nuestro sitio web.**
3. Ahora dentro de la carpeta **apps** daremos las siguiente instrucciones para crear ambas aplicaciones para nuestro sitio:

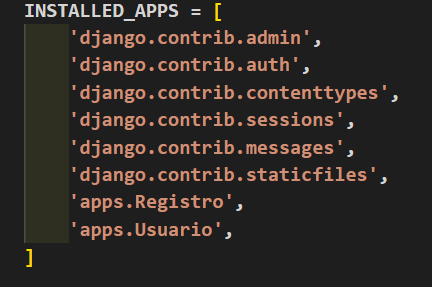
**django-admin.py startapp Registro**

**django-admin.py startapp Usuario**

1. El proyecto debería ir con esta estructura:



1. Después de crear LAS aplicaciones también necesitamos decirle a Django que debe utilizarla. Lo hacemos en el archivo **academico/settings.py**. Tenemos que encontrar **INSTALLED\_APPS** y añadir una línea que contenga 'blog', justo por encima de **)**. Debe tener este aspecto:



**Aplicación Registro**

Creando Modelos

1. En el archivo **Registro/models.py** creamos nuestros modelos y escribimos:

**from django.db import models**

**# Create your models here.**

**class Carrera(models.Model):**

**nombre = models.CharField(max\_length=20)**

**semestres = models.IntegerField()**

**mensualidad = models.IntegerField()**

**def \_\_str\_\_(self):**

**return self.nombre**

**class Alumno(models.Model):**

**carrera = models.ForeignKey(Carrera, null=True, blank=True, on\_delete=models.CASCADE)**

**rut = models.CharField(max\_length=15)**

**nombre = models.CharField(max\_length=50)**

**apellido\_paterno = models.CharField(max\_length=50)**

**apellido\_materno = models.CharField(max\_length=50)**

**edad = models.IntegerField()**

**telefono = models.CharField(max\_length=20)**

**domicilio = models.TextField()**

**fecha\_registro = models.DateTimeField(auto\_now=True)**

**fecha\_nacimiento = models.DateField()**

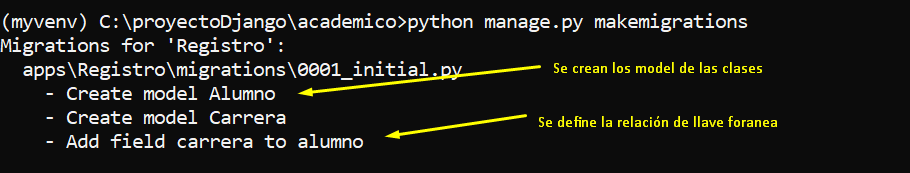
**def \_\_str\_\_(self):**

**return self.nombre**

1. El último paso es añadir nuestros modelos a la base de datos. Primero tenemos que hacer que Django sepa que tenemos algunos cambios en nuestro modelo, ejecutar:

**python manage.py makemigrations**

Noten el mensaje de la consola:

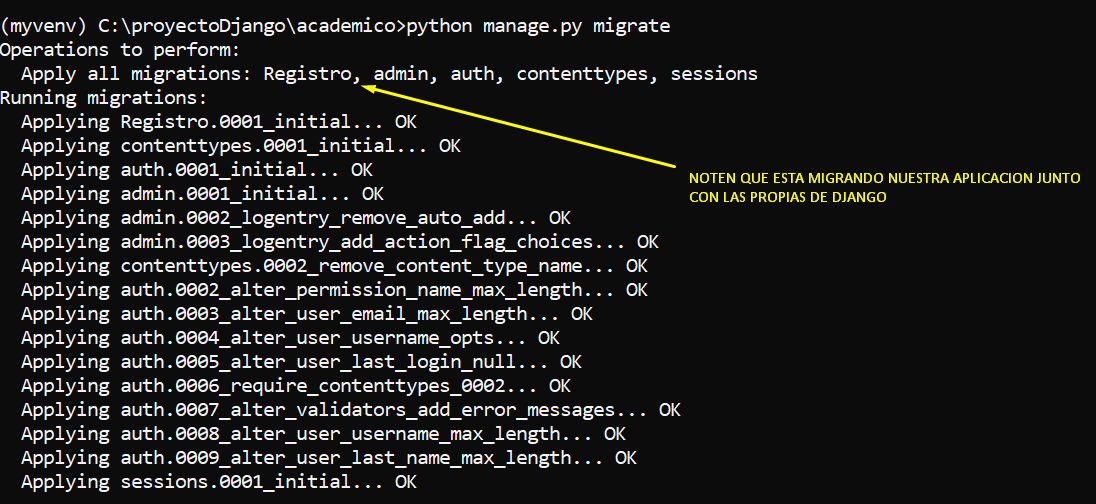


1. Ahora haremos la migración, es decir, crearemos las tablas, en este proceso se incluirán los model por defecto que trae Django (User, Session, etc)

Escribimos la instrucción:

**python manage.py migrate**

Enviando el mensaje:



Administrador de Django

1. Para agregar, editar y borrar los registros que hemos modelado, utilizaremos el administrador de Django. Abrir archivo apps/Registro/admin.py y colocar:

**from django.contrib import admin**

**from .models import Alumno, Carrera**

**# Register your models here.**

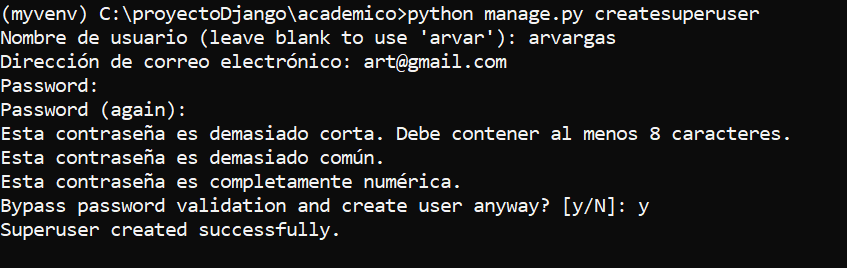
**admin.site.register(Carrera)**

**admin.site.register(Alumno)**

1. Para poder ingresar deberás crear un *superusuario* (que tiene control sobre todo lo que hay en el sitio), ejecutar:

**python manage.py createsuperuser**

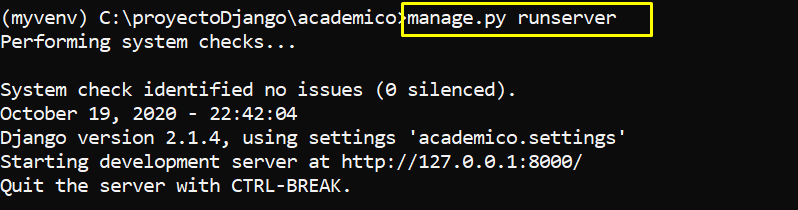
Ahora ingresa tu nombre de usuario, corrreo y password:



Observaciones:

* Como pueden ver si no colocan el nombre de usuario adopta el nombre del administrador del computador
* Pide una password de 8 caracteres, pero en mi caso puse una más corta pero le dije que la aceptará de todas formas.

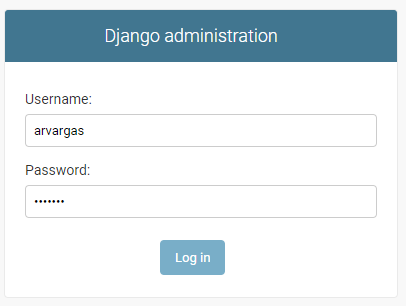
1. Arrancamos servidor:



1. Ahora en el browser nos vamos a la ruta del administrador:



1. Ingresa con las credenciales de super usuario que elegiste, ahora deberías poder ver el panel de administración de Django:



Y una vez dentro:



1. Agrega cinco o seis registros a las tablas. Ejemplo yo ingrese 3 carreras:



Ahora ingreso alumnos a las carreras:



Django urls

1. Abre el archive academico/urls.py (mapeador general) y copia el siguiente código:

**from django.contrib import admin**

**from django.urls import path, include**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('', include('apps.registro.urls')),**

**]**

1. Crea el archivo **Registro/urls.py** dentro de este archivo agrega:

**from django.urls import path, include**

**from . import views**

**urlpatterns = [**

**# listar las carreras de la bd**

**path('listarCarreras', views.listar\_carreras, name="listar\_carreras"),**

**]**

1. Y dentro de vista, es decir: **Registro/views.py** debemos agregar la función **listar\_carreras**:

**def listar\_carreras(request):**

**carreras = Carrera.objects.all()**

**return render(request, "Registro/listar\_carreras.html", {'carreras': carreras})**

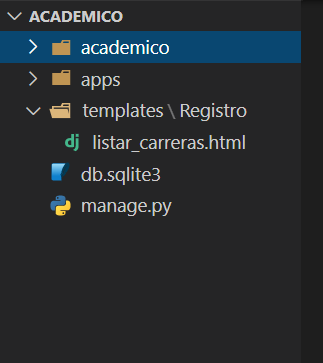
1. Falta crear la plantilla asociada, es decir, listar\_carreras.html pero daremos una nueva ruta a nuestros **templates**.

Plantillas de Django

1. Haremos un cambio en la estructura del proyecto en cuanto a las plantillas, esta vez las manejaremos fuera de las aplicaciones, por lo tanto:

Dentro de la carpeta raíz crearemos la carpeta llamada **templates** y dentro de esta la carpeta **Registro**, esta última crearemos la plantilla para el requerimiento que estamos llevando, es decir listar\_carreras.html

Debería quedar algo así:



1. Debemos volver a académico/settings.py del proyecto para indicar la ruta de los templates, puntualmente indicaremos:

**TEMPLATES = [**

**{**

**'BACKEND': 'django.template.backends.django.DjangoTemplates',**

**'DIRS': [os.path.join(BASE\_DIR, 'templates')],**

**'APP\_DIRS': True,**

**'OPTIONS': {**

**'context\_processors': [**

**'django.template.context\_processors.debug',**

**'django.template.context\_processors.request',**

**'django.contrib.auth.context\_processors.auth',**

**'django.contrib.messages.context\_processors.messages',**

**],**

**},**

**},**

**]**

1. Ahora volvamos a nuestra plantillas **listar\_carreras.html** su contenido sería:

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8" />**

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

**<meta http-equiv="X-UA-Compatible" content="ie=edge" />**

**<title>Listar carreras</title>**

**</head>**

**<body>**

**<h2>Lista de carreras</h2>**

**{% for c in carreras %}**

**<div>**

**<p> Nombre: {{ c.nombre }} </p>**

**<p>Semestres: {{ c.semestres }} </p>**

**<p>Mensualidad $ {{ c.mensualidad }} </p>**

**</div>**

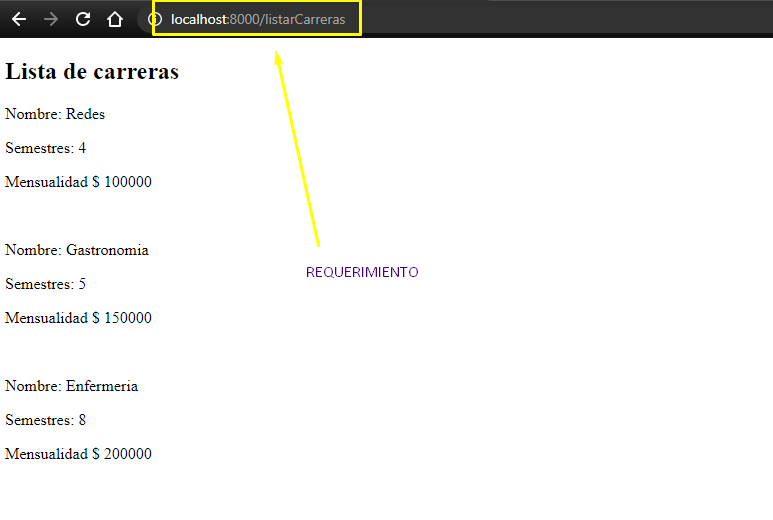
**<br>**

**{% endfor %}**

**</body>**

**</html>**

1. Ahora ejecutamos el servidor y vemos el resultado:



Creando el CRUD

1. Ahora crearemos un archivo en **Registro/forms.py** y en este tendremos:

**from django import forms**

**from .models import Carrera, Alumno**

**class CarreraForm(forms.ModelForm):**

**class Meta:**

**model = Carrera**

**fields = ['nombre', 'semestres', 'mensualidad']**

**labels = {**

**'nombre': 'Nombre',**

**'semestres': 'Semestres',**

**'mensualidad': 'Mensualidad Pesos Chilenos',**

**}**

**widgets = {**

**'nombre': forms.TextInput(attrs={'class': 'form-control'}),**

**'semestres': forms.TextInput(attrs={'class': 'form-control'}),**

**'mensualidad': forms.TextInput(attrs={'class': 'form-control'}),**

**}**

1. Ahora volvamos a vista, vamos a cargar algunas funciones, no olvidemos importar nuestra clase formulario recién creada:

**from .forms import CarreraForm**

Ahora agregaremos las siguientes funciones a nuestra vista:

**def agregar\_carrera(request):**

**if request.method == "POST":**

**form = CarreraForm(request.POST)**

**if form.is\_valid():**

**model\_instance = form.save(commit=False)**

**model\_instance.save()**

**return redirect("/agregar\_carrera")**

**else:**

**form = CarreraForm()**

**return render(request, "Registro/agregar\_carrera.html", {'form': form})**

**def borrar\_carrera(request, carrera\_id):**

**# Recuperamos la instancia de la carrera y la borramos**

**instancia = Carrera.objects.get(id=carrera\_id)**

**instancia.delete()**

**# Después redireccionamos de nuevo a la lista**

**return redirect('listar\_carreras')**

**def editar\_carrera(request, carrera\_id):**

**# Recuperamos la instancia de la carrera**

**instancia = Carrera.objects.get(id=carrera\_id)**

**# Creamos el formulario con los datos de la instancia**

**form = CarreraForm(instance=instancia)**

**# Comprobamos si se ha enviado el formulario**

**if request.method == "POST":**

**# Actualizamos el formulario con los datos recibidos**

**form = CarreraForm(request.POST, instance=instancia)**

**# Si el formulario es válido...**

**if form.is\_valid():**

**# Guardamos el formulario pero sin confirmarlo,**

**# así conseguiremos una instancia para manipular antes de grabar**

**instancia = form.save(commit=False)**

**# Podemos guardar cuando queramos**

**instancia.save()**

**# Si llegamos al final renderizamos el formulario**

**return render(request, "Registro/editar\_carrera.html", {'form': form})**

Ahora falta que el mapeador local de las url respectivas. Agregaremos al archivo Registro/urls.py.

El siguiente código:

**from django.urls import path, include**

**from . import views**

**urlpatterns = [**

**# listar las carreras de la bd**

**path('listarCarreras', views.listar\_carreras, name="listar\_carreras"),**

**# agregar una carrera**

**path('agregar\_carrera', views.agregar\_carrera, name="agregar\_carrera"),**

**# editar una carrera**

**path('editar\_carrera/<int:carrera\_id>', views.editar\_carrera ,name="editar\_carrera"),**

**# borrar una carrera**

**path('borrar\_carrera/<int:carrera\_id>', views.borrar\_carrera, name="borrar\_carrera"),**

**]**

Solo falta probar el crud……..

Vista basada en clases

1. Vamos a crear nuevas funciones en vistas, estas serán clases heredadas de la View de Django

**# --Otra forma usando clases Generics -------**

**class CarreraCreate(CreateView):**

**model = Carrera**

**form\_class = CarreraForm**

**template\_name = 'Registro/carrera\_form.html'**

**success\_url = reverse\_lazy("listar\_carreras")**

1. Importemos todas:

**from django.views.generic import ListView, CreateView, UpdateView, DeleteView**

1. Agregamos una URL para esta nueva vista, esto es nuestro mapeador local:

**path('add\_carrera', views.CarreraCreate.as\_view(), name="add\_carrera"),**

Ahora a probar si funciona, levantar servidor y hacer los requerimientos por URL

1. Aquí les dejo el resto de las funciones de vista, favor de crear plantillas heredadas para esta variante del proyecto.

Las nuevas clases de vistas serian:

**class CarreraList(ListView):**

**model = Carrera**

**template\_name = 'Registro/list\_carreras.html'**

**# paginate\_by = 4**

**class CarreraUpdate(UpdateView):**

**model = Carrera**

**form\_class = CarreraForm**

**template\_name = 'Registro/carrera\_form.html'**

**success\_url = reverse\_lazy('list\_carreras')**

**class CarreraDelete(DeleteView):**

**model = Carrera**

**template\_name = 'Registro/carrera\_delete.html'**

**success\_url = reverse\_lazy('list\_carreras')**

*A dormzZZzZzzZzzzZZZzz*