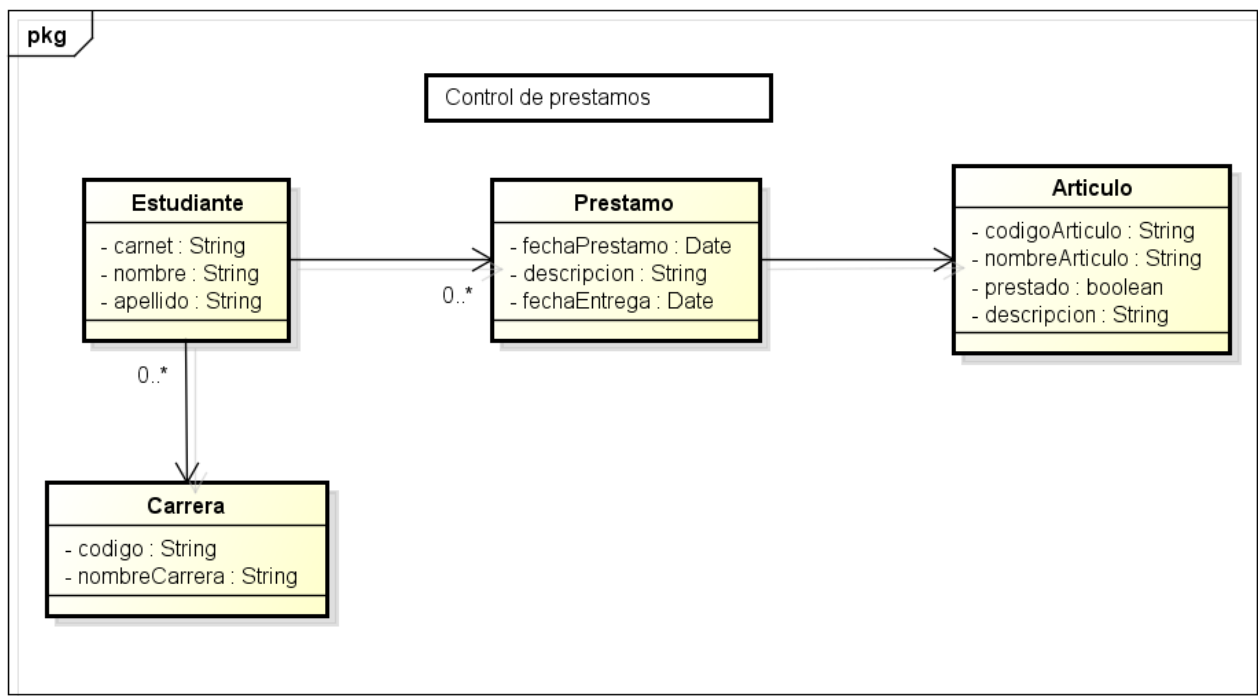


# Django



powered by Astah

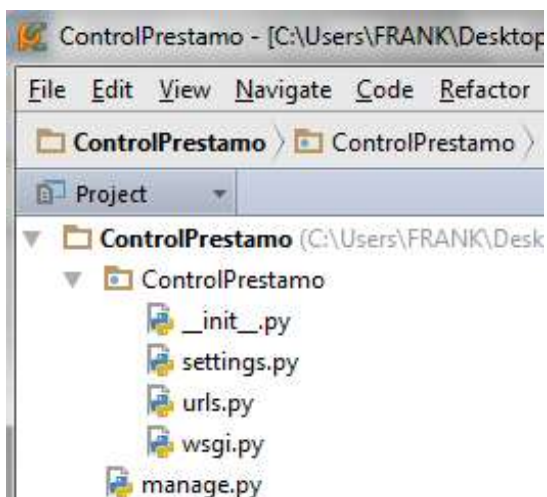
## 1) Crear Proyecto ControlPrestamo

**django-admin startproject ControlPrestamo**

Nota: probablemente en debían sea **python django-admin startproject ControlPrestamo**

```
C:\Users\FRANK\Desktop\pythonProject>django-admin startproject ControlPrestamo
C:\Users\FRANK\Desktop\pythonProject>
```

Si revisamos nos queda la siguiente estructura



**\_\_init\_\_.py**: Un archivo requerido para que Python trate a este directorio como un paquete (ejemplo un grupo de módulos).

**manage.py**: Una utilidad de línea de comandos que te deja interactuar con este proyecto de Django de varias formas.

**settings.py**: Opciones/con configuraciones para este proyecto de Django.

**urls.py**: La declaración de las URL para este proyecto de Django; una tabla de contenidos de tu sitio hecho con Django.

**wsgi.py**: Web Server Gateway Interface, se utiliza para configuración del servidor, desplegar la aplicación en un servidor.

## El servidor de desarrollo

Django incluye un servidor web ligero que puedes usar mientras estás desarrollando tu sitio. Incluimos este servidor para que puedas desarrollar tu sitio rápidamente, sin tener que lidiar con configuraciones de servidores web de producción (i.e., Apache) hasta que estés listo para la producción. Este servidor de desarrollo vigila tu código a la espera de cambios y se reinicia automáticamente, ayudándote a hacer algunos cambios rápidos en tu proyecto sin necesidad de reiniciar nada.

## Configurar la base de datos

Archivo settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Usaremos sqlite3, por tanto solo cambiaremos el nombre de la base de datos, **db** por **pretamodb**

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'pretamodb.sqlite3'),
    }
}
```

Guarda los cambios y corre el proyecto, veras que se creara la base de datos

**python manage.py runserver** (presiona Ctrl+c para detener el servidor)

**Abre el navegador en la dirección que te indica**

```

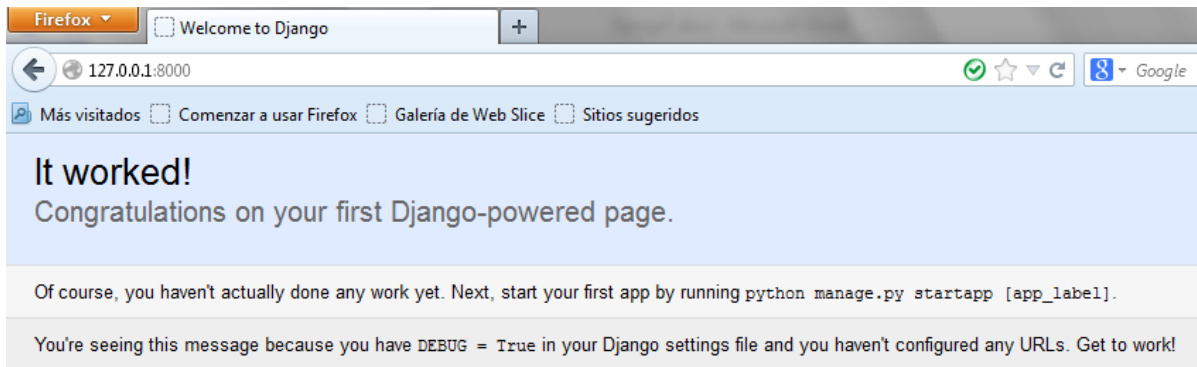
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

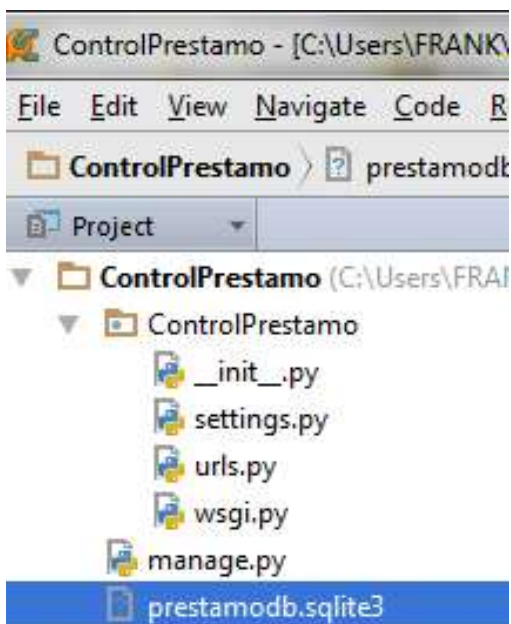
You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.
April 23, 2016 - 21:31:55
Django version 1.7.4, using settings 'ControlPrestamo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

Veras como has creado tu primer proyecto con django



Te darás cuenta como se creó la base de datos



## Crear primera app

Ingresa en la consola el siguiente comando

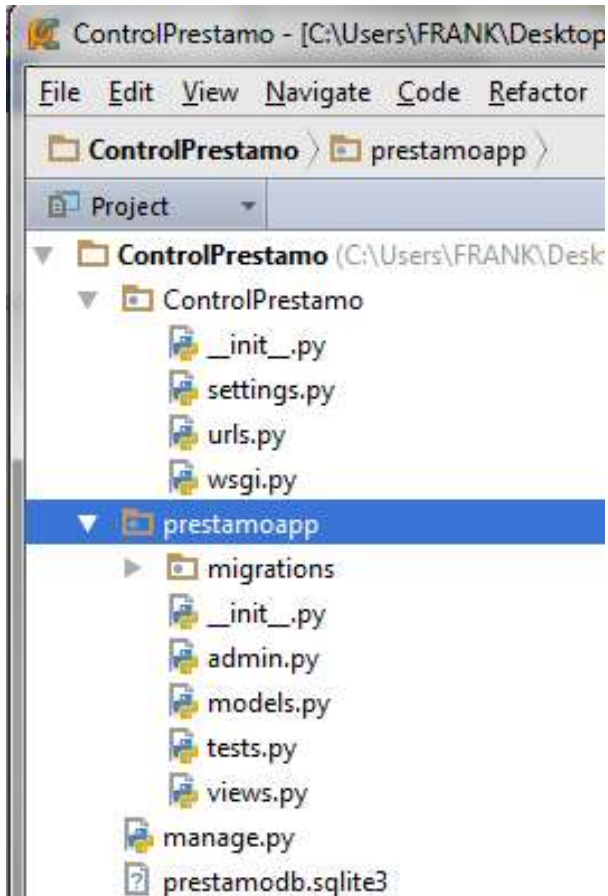
**django-admin startapp prestamoapp**

```

C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>django-admin startapp prestamoapp
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>

```

La estructura del proyecto queda de la siguiente forma



**admin.py:** Configuraciones de nuestro modelo al administrador

**models.py:** contendrá nuestros modelos (Django ORM models) para nuestra app.

**views.py:** contendrá el código de las vistas (Que datos se le pasaran al template).

**tests.py:** contiene pruebas unitarias y de integración.

Registrando nuestra app en **settings.py**: debemos de registrar nuestra aplicación.

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'prestamoapp',  
)
```

Creando nuestros modelos

Abre el archivo **models.py** y digita el siguiente código

```
from django.db import models
from django.utils import timezone

# Create your models here.

class Carrera(models.Model):
    codigo=models.CharField(max_length=6,primary_key=True)
    nombreCarrera = models.CharField(max_length=100,null=False)

class Articulo(models.Model):
    codigoArticulo=models.CharField(max_length=6,primary_key=True)
    nombreArticulo=models.CharField(max_length=100,null=False)
    descripcion = models.CharField(max_length=255)
    prestado = models.BooleanField(default=False)

class Estudiante(models.Model):
    carnet=models.CharField(max_length=7,primary_key=True)
    carrera=models.ForeignKey(Carrera,null=False)
    nombre=models.CharField(max_length=30,null=False)
    apellido=models.CharField(max_length=30,null=False)

class Prestamo(models.Model):
    articulo = models.OneToOneField(Articulo,null=False)
    estudiante=models.ForeignKey(Estudiante,null=False)
    fechaPrestamo=models.DateField(auto_now_add=timezone.now().date())
    fechaEntrega=models.DateField()
```

**guarda los cambios**

Mapeando la base de datos

Pasos:

**python manage.py makemigrations:** prepara los cambios en la base de datos.

**python manage.py validate :** valida que todo este bien y sin errores.

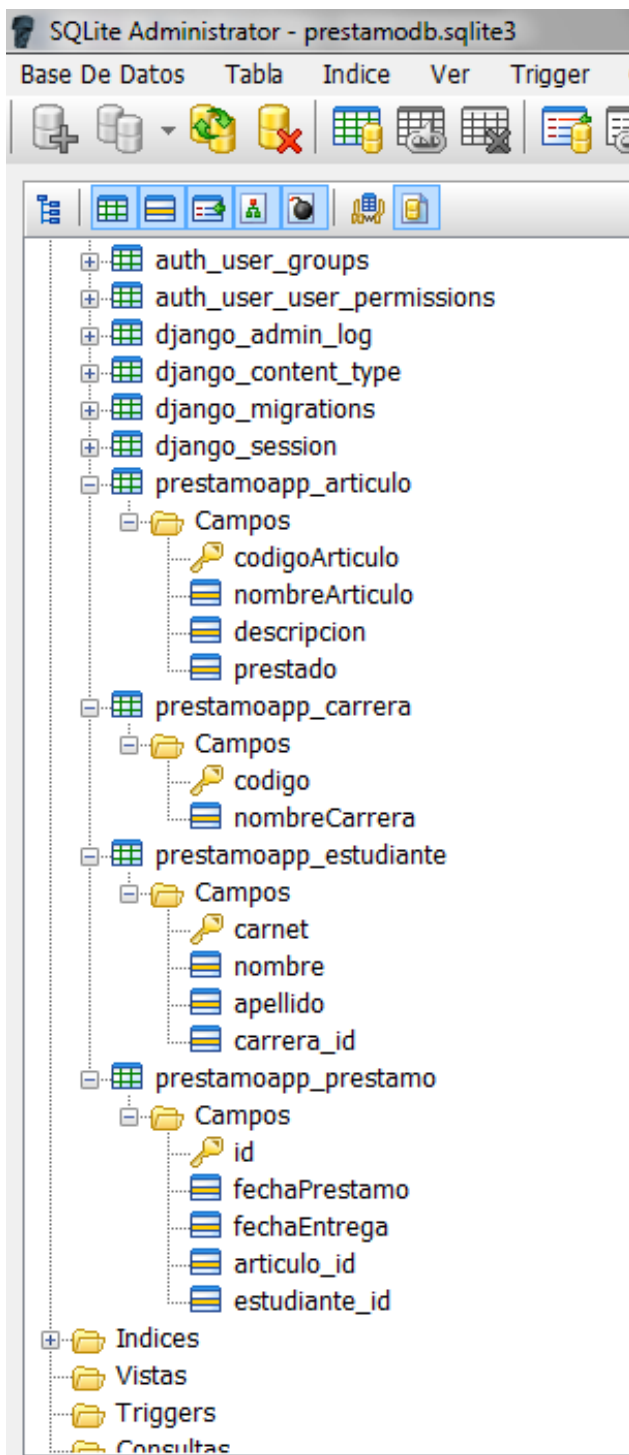
**python manage.py migrate :** realiza cambios en la base de datos.

```
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py makemigrations
Migrations for 'prestamoapp':
  0001_initial.py:
    - Create model Articulo
    - Create model Carrera
    - Create model Estudiante
    - Create model Prestamo

C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py validate
System check identified no issues (0 silenced).

C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py migrate
Operations to perform:
  Apply all migrations: prestamoapp, contenttypes, sessions, admin, auth
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying prestamoapp.0001_initial... OK
  Applying sessions.0001_initial... OK
```

Si usamos un administrador de base de datos para sqlite, en este caso **sqliteadmin** veremos como django ha mapeado la base de datos.



Vemos que los nombres de las tablas las mapeo con el siguiente formato:

**nombreapp\_nombreclase** todo en minúsculas y además nos ha generado las llaves ajenas en préstamo y también en estudiante, otra cosa que es notoria es que al préstamo no le asignamos explícitamente un identificador por lo que django le ha creado un id genérico auto incremental.

## Creando un super usuario

Para poder entrar al administrador de django debemos de tener un usuario y una contraseña, por lo que antes de pasar a ver el administrador crearemos uno.

**python manage.py createsuperuser**

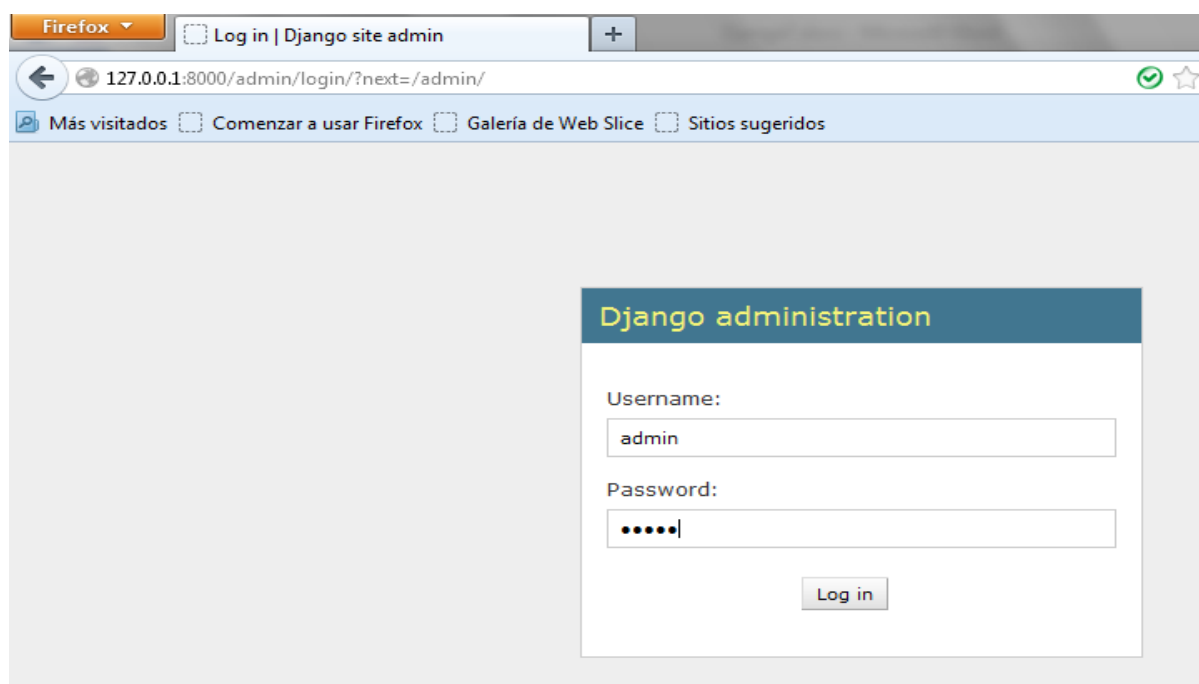
Ingresa como usuario **admin** y contraseña **admin**, además ingresa tu correo

```
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py createsuperuser
Username (leave blank to use 'frank'): admin
Email address: micorreo@yahoo.es
Password:
Password (again):
Superuser created successfully.
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>
```

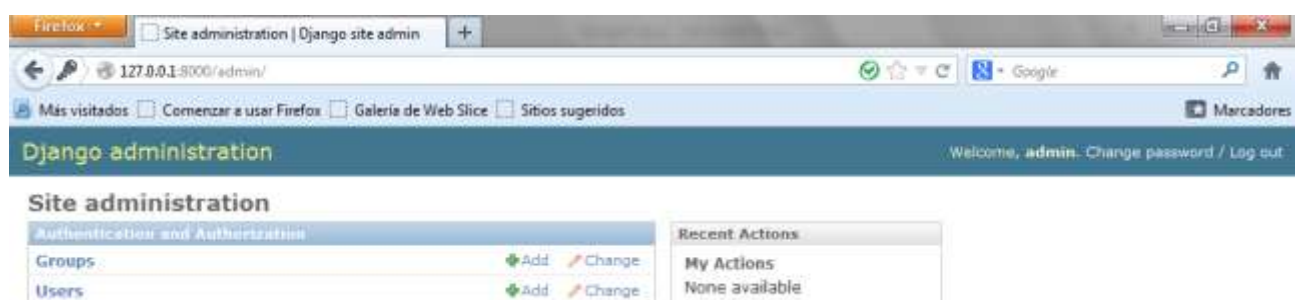
Ahora corremos la aplicación:

**Python manage.py runserver** y entra en la siguiente dirección **http://127.0.0.1:8000/admin**

Te aparecerá un formulario, ingresa tu usuario y contraseña y logueate



Te aparecerá una ventana como la siguiente



Si ves no aparecen los modelos que que necesitamos.

## Configurando el admin de django

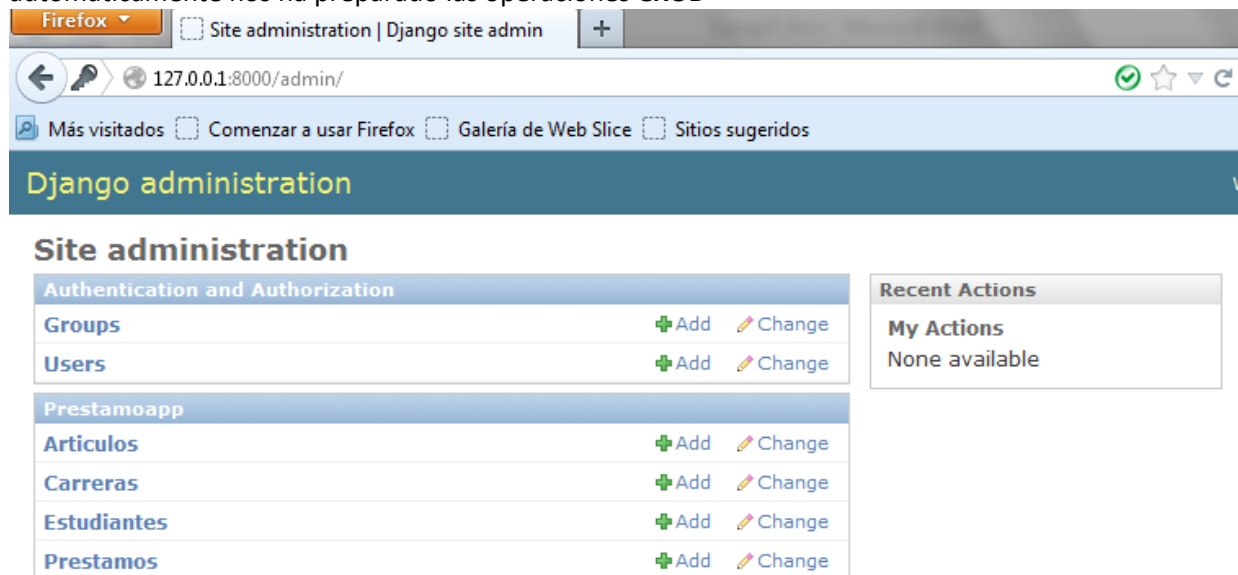
Abre el archivo **admin.py** y escribe el siguiente código para que registre los modelos en el admin

```
from django.contrib import admin
from prestamoapp.models import
Estudiante, Prestamo, Artículo, Carrera

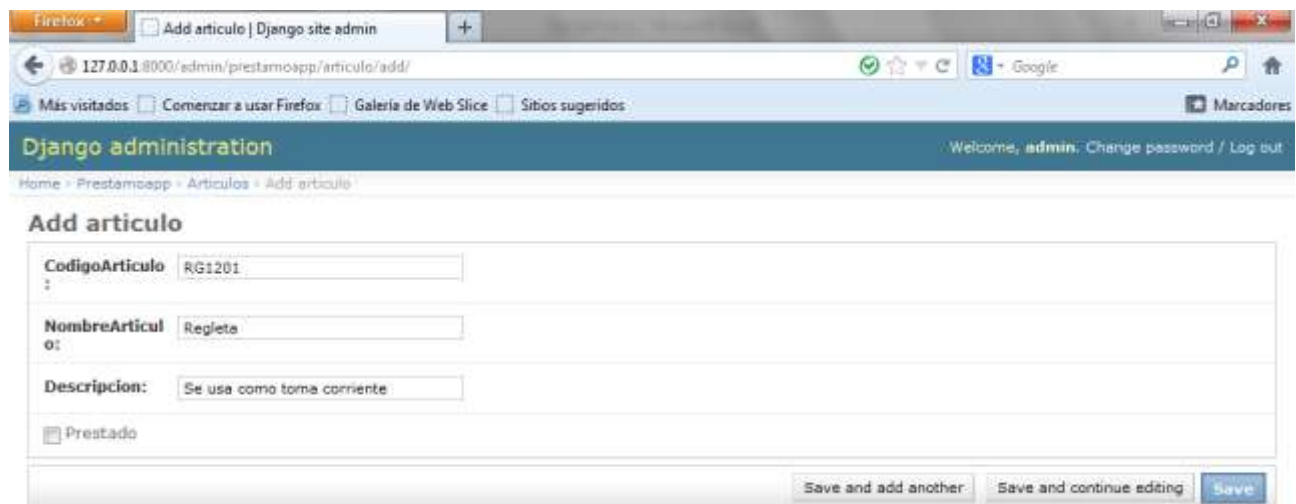
# Register your models here.

admin.site.register(Artículo)
admin.site.register(Carrera)
admin.site.register(Estudiante)
admin.site.register(Prestamo)
```

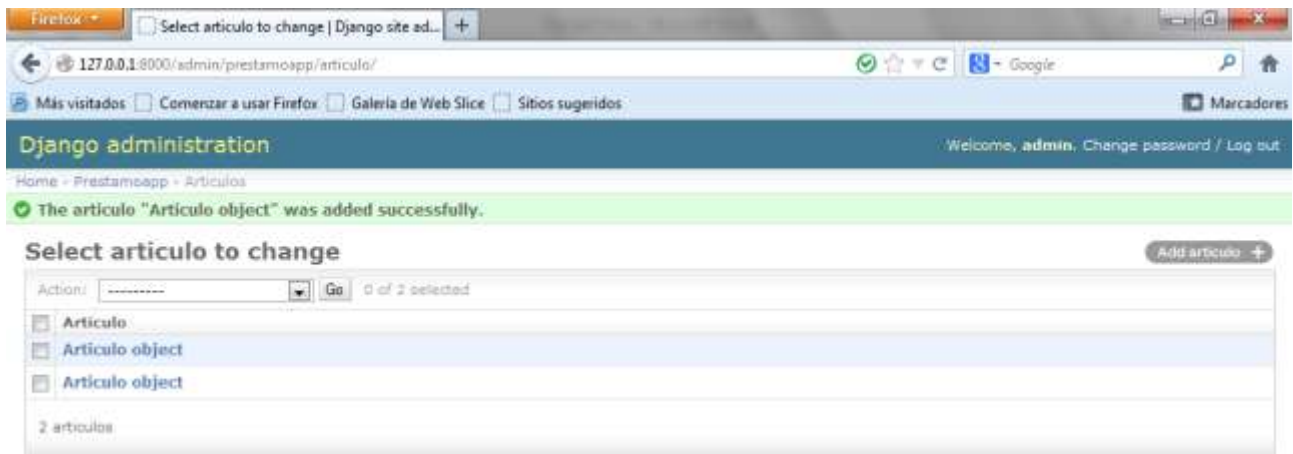
Guarda los cambios y actualiza el navegador. Veras como ya aparecen los modelos en el admin, donde automáticamente nos ha preparado las operaciones **CRUD**



Agregaremos un articulo, para ello haz clic en el signo + de Articulos, ingresa dos articulo y guarda los cambios.







Vemos como se han creado dos objetos de tipo artículo, sin embargo no podemos distinguir que artículo es, para que se vea mejor vamos a configurarlo para que nos muestre el nombre del artículo, abre el archivo **models.py** y agregale el siguiente método que retorne un identificador del objeto, en este caso el nombre del objeto:

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

configura a todas las clase para que tengan una cadena que los identifican, luego guarda los cambios y actualiza el navegador.

```
class Carrera(models.Model):
    codigo=models.CharField(max_length=6,primary_key=True)
    nombreCarrera = models.CharField(max_length=100,null=False)

    def __str__(self):
        return self.nombreCarrera

class Articulo(models.Model):
    codigoArticulo=models.CharField(max_length=6,primary_key=True)
    nombreArticulo=models.CharField(max_length=100,null=False)
    descripcion = models.CharField(max_length=255)
    prestado = models.BooleanField(default=False)

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

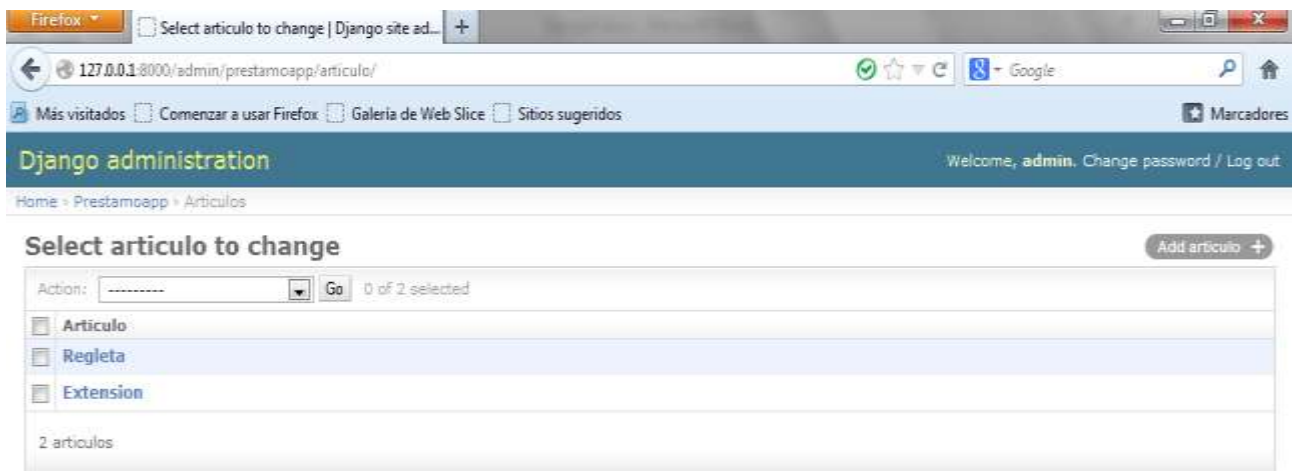
```
class Estudiante(models.Model):
    carnet=models.CharField(max_length=7,primary_key=True)
    carrera=models.ForeignKey(Carrera,null=False)
    nombre=models.CharField(max_length=30,null=False)
    apellido=models.CharField(max_length=30,null=False)

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

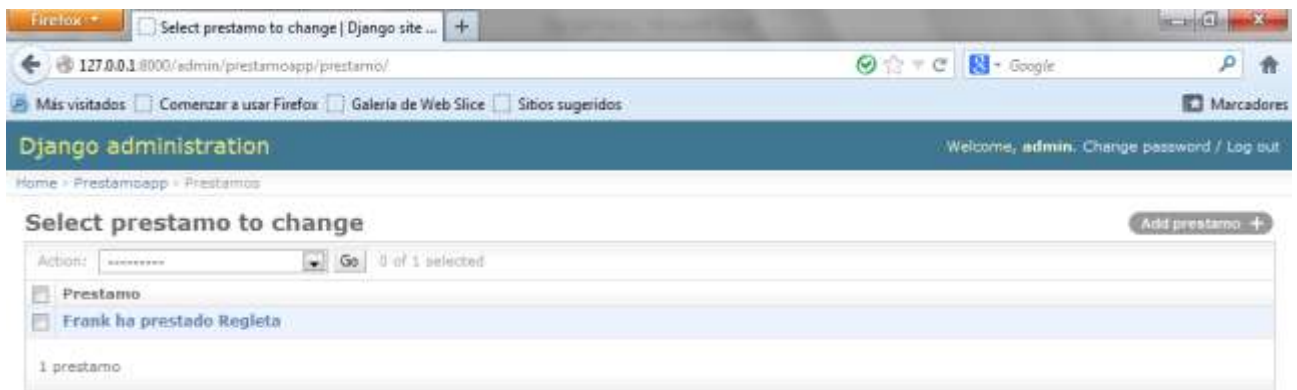
```
class Prestamo(models.Model):
    articulo = models.OneToOneField(Articulo,null=False)
    estudiante=models.ForeignKey(Estudiante,null=False)
```

```
fechaPrestamo=models.DateField(auto_now_add=timezone.now().date())  
fechaEntrega=models.DateField()
```

```
def __str__(self):  
    return self.estudiante.nombre+" ha prestado  
"+self.articulo.nombreArticulo
```



Ingresa una carrera, un estudiante y haz un prestamo



Continuaremos personalizándolo, para ello abre el archivo admin.py y agregale código hasta que te quede como el siguiente:

```
from django.contrib import admin
from prestamoapp.models import
Estudiante, Prestamo, Artículo, Carrera

# Register your models here.

class EstudianteAdmin(admin.ModelAdmin):
    list_display = ('carnet', 'nombre', 'carrera') #tupla

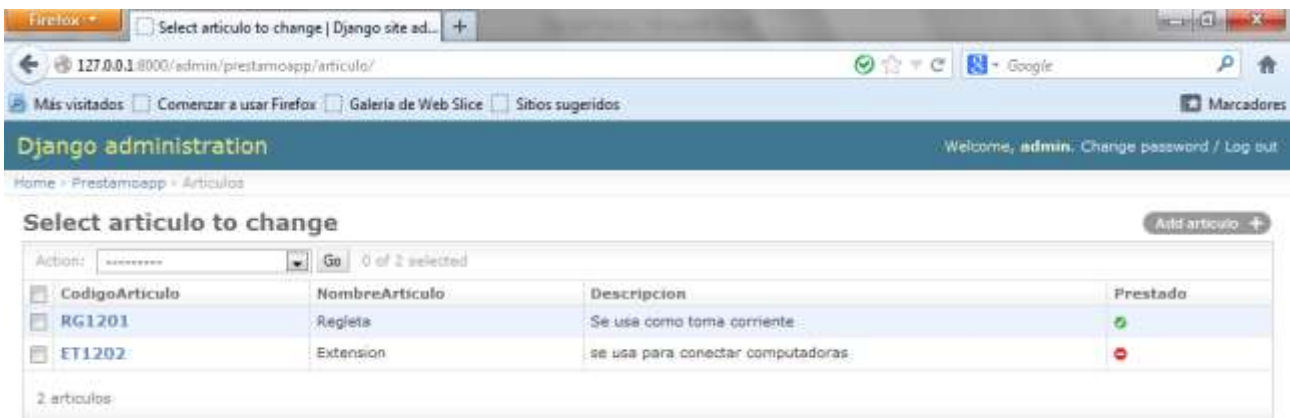
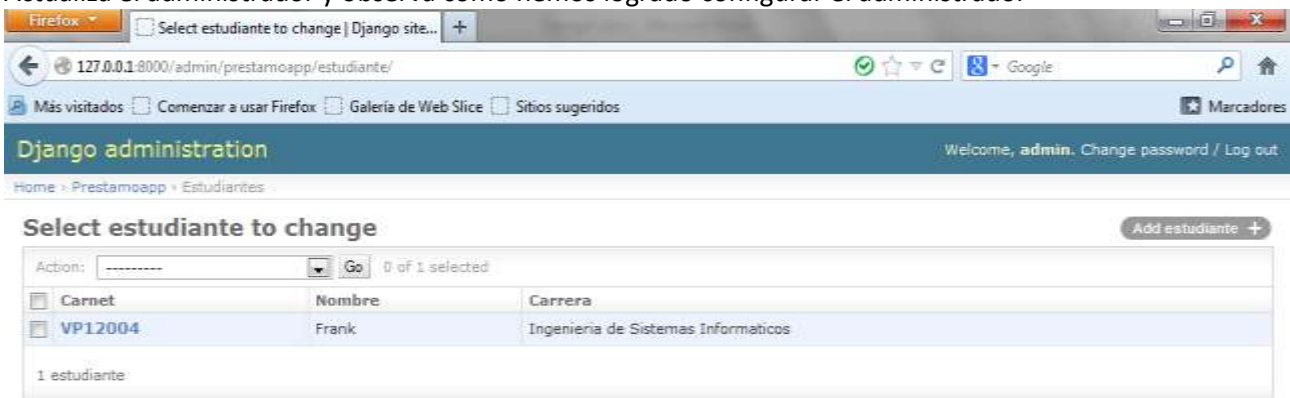
class PrestamoAdmin(admin.ModelAdmin):
    list_display =
('articulo', 'estudiante', 'fechaPrestamo', 'fechaEntrega')

class ArtículoAdmin(admin.ModelAdmin):
    list_display =
('codigoArticulo', 'nombreArticulo', 'descripcion', 'prestado')

class CarreraAdmin(admin.ModelAdmin):
    list_display = ('codigo', 'nombreCarrera')

admin.site.register(Artículo, ArtículoAdmin)
admin.site.register(Carrera, CarreraAdmin)
admin.site.register(Estudiante, EstudianteAdmin)
admin.site.register(Prestamo, PrestamoAdmin)
```

Actualiza el administrador y observa como hemos logrado configurar el administrador



# Patron MVT (Model View Template)

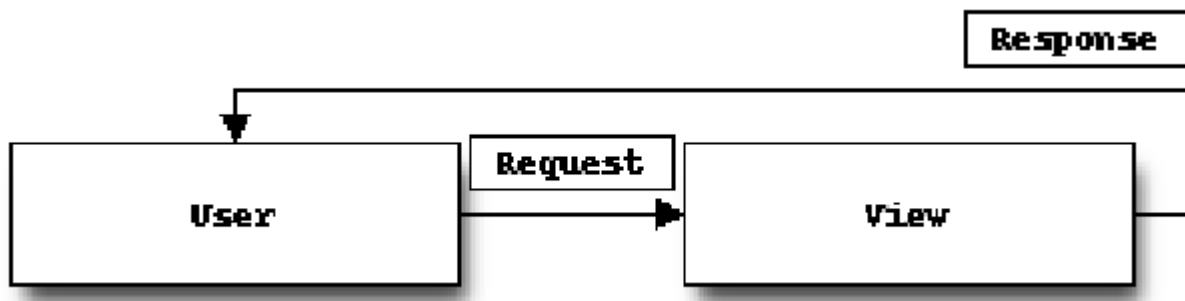
**M** significa Model (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

**T** significa Template (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web o otro tipo de documento.

**V** significa View(Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelos y las plantillas.

## Vistas

Las vistas en django toman un objeto HTTP Request (petición) y retornan un objeto HTTP Response (respuesta).



El único requisito es que esta tome el objeto HTTP Request llamado request como primer parámetro. Esto significa que escribir una vista es muy sencillo.

Abre el archivo **views.py** e ingresa el siguiente código

```
from django.http import HttpResponse

def mi_primer_vista(request):
    return HttpResponse("Bienvenido al curso de python y django")
```

Tal y como ves es muy simple, una vista no es mas que una función python que toma como parámetro obligatorio un **request**.

## Manejo de URLs

Para poder ver en que dirección ver el contenido que le mandamos de la vista al template tenemos que decirle a django donde mostrara el contenido.

Una url en djano tiene la siguiente estructura

**(expresión regular, vista [,diccionario opcional])**

Si abres el archivo **settings.py** veras que hay una parte donde está configurado el archivo que contendrá las urls

```
ROOT_URLCONF = 'ControlPrestamo.urls'
```

Abre el archivo **urls.py** e ingresa la url de nuestra vista

```
from django.conf.urls import patterns, include, url
from django.contrib import admin

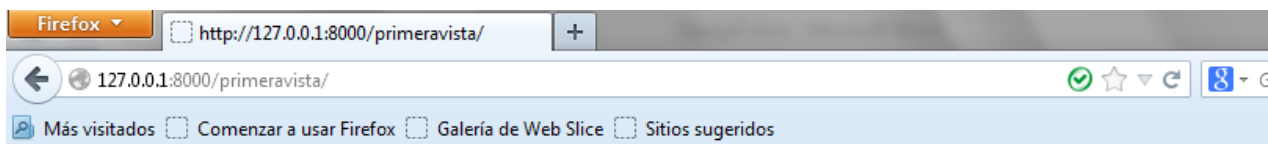
urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'ControlPrestamo.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
    url(r'^primeravista/$', 'prestamoapp.views.mi_primer_vista'),
)
```

Guarda los cambios e ingresa la siguiente dirección en el navegador

<http://127.0.0.1:8000/primeravista/>

Veras como se ha generado un template(plantilla html) genérica que nos muestra el string que le indicamos en el HTTPResponse de nuestra view.



# Bienvenido al curso de python y django

## Render

Otro tipo de vista bastante útil es usar vistas que retorne un objeto render, el cual tiene la siguiente estructura

**render(request,nombrePlantilla,{diccionario de objetos mandado al template})**

Crearemos una nueva vista, para ello abre el archivo **views.py** e digita la vista misArticulos

```
from django.shortcuts import render

from django.http import HttpResponse
from prestamoapp.models import Artículo

def mi_primer_vista(request):
    return HttpResponse("Bienvenido al curso de python y django")

def misArticulos(request):
    return
render(request,"mis_articulos.html",{ 'articulos':Articulo.objects.
all() })
```

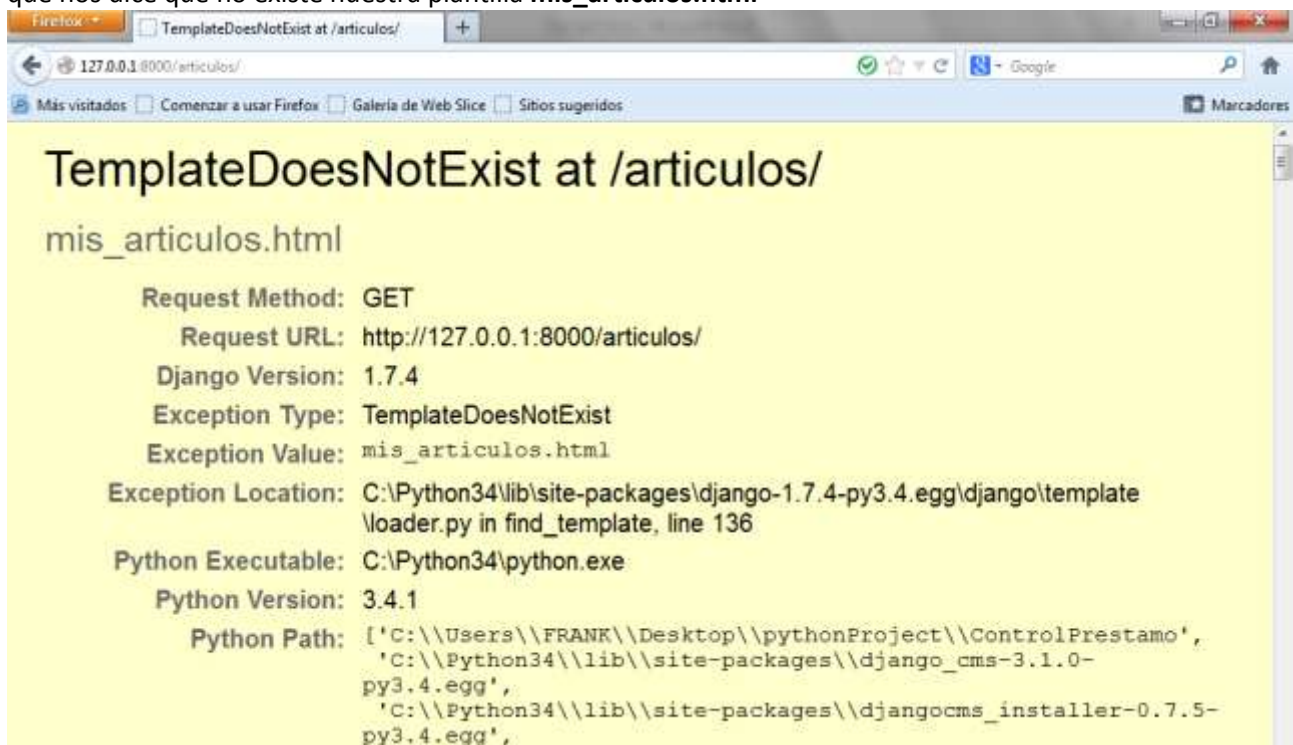
Registramos nuestra vista en el archivo **urls.py**

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
from prestamoapp.views import misArticulos

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'ControlPrestamo.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

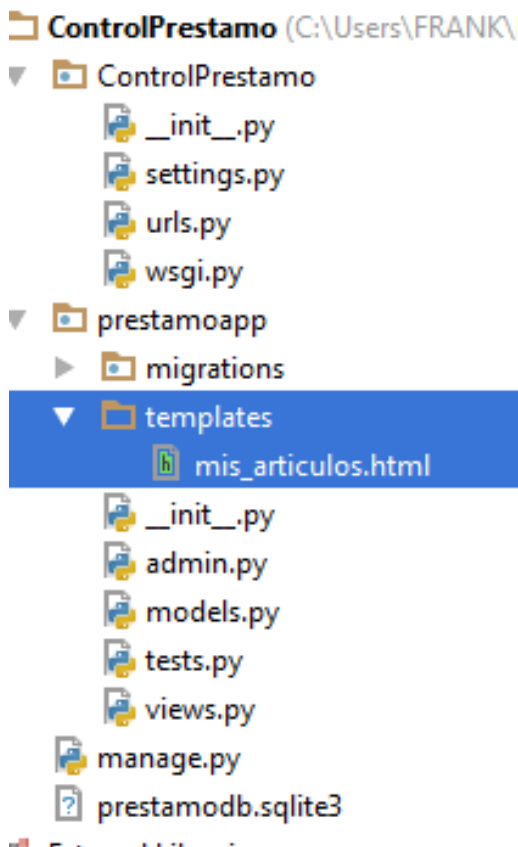
    url(r'^admin/', include(admin.site.urls)),
    url(r'^primeravista/$', 'prestamoapp.views.mi_primer_vista'),
    url(r'^articulos/$', misArticulos)
)
```

Corre el proyecto y abre la dirección <http://127.0.0.1:8000/articulos/>, y veras que nos aparecerá un error que nos dice que no existe nuestra plantilla **mis\_articulos.html**

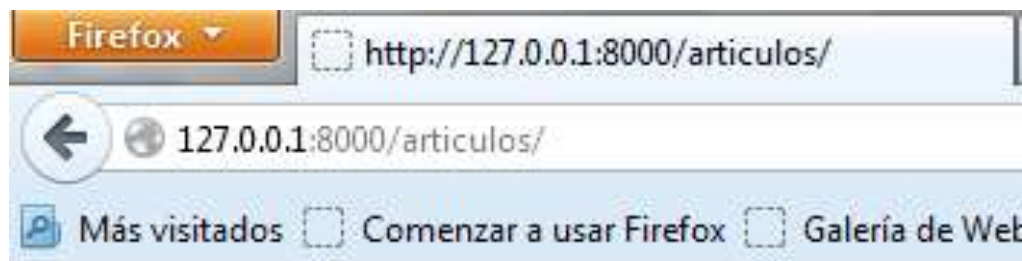


Para resolverlo, primeramente debemos saber dónde ubicar nuestras plantillas html en nuestro proyecto, django por defecto buscara las plantillas html en una carpeta llamada **templates** que se ubica dentro de nuestra aplicación. Si quieres ponerle otro nombre revisa la documentación de django, aca trabajaremos con las configuraciones por defecto de django por simplicidad y porque estamos siguiendo el estándar de django para el desarrollo de aplicaciones.

Creamos la carpeta templates y dentro de ella la plantilla mis\_articulos.html, la estructura del proyecto que da como la siguiente:



Reinicia el servidor y verifica que ya no aparece el error, pero como no hemos puesto nada, nos aparecerá la página mis\_articulos.html en blanco.



Abre el archivo mis\_articulos.html hasta que te quede como el siguiente

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>Lista de articulos</title>
</head>
<body>
  <h1>Bienvenidos</h1>
  <h2>Lista de articulos</h2>

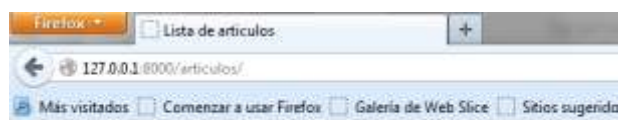
  <table>
    <thead>
      <th>Codigo</th>
      <th>Nombre</th>
      <th>Descripcion</th>
      <th>Estado</th>
    </thead>
    <tbody>
      {% for articulo in articulos %}
        <!-- articulos es el nombre en el que capturamos los datos en
        el diccionario que le enviamos de la vista al template
        {'articulos':Articulo.objects.all()}
        -->

        <tr>
          <td>{{ articulo.codigoArticulo }}</td>
          <td>{{ articulo.nombreArticulo }}</td>
          <td>{{ articulo.descripcion }}</td>
          {% if articulo.prestado == False %}
            <td>Disponible</td>
          {% endif %}
          {% if articulo.prestado == True %}
            <td>Prestado</td>
          {% endif %}
        </tr>
      {% endfor %}
    </tbody>
  </table>

</body>
</html>
```



Actualiza el navegador y tendremos el resultado siguiente



# Bienvenidos

## Lista de artículos

Codigo	Nombre	Descripcion	Estado
RG1201	Regleta		Prestado
ET1202	Extension		Disponible