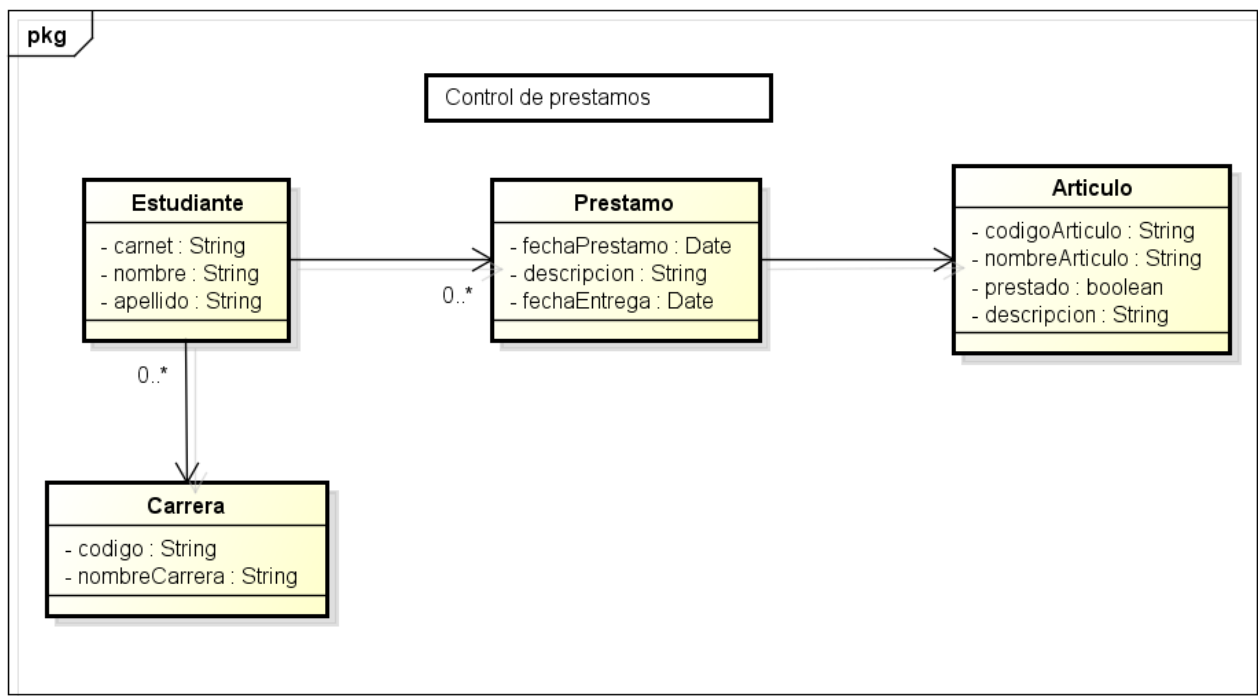


Django



powered by Astah

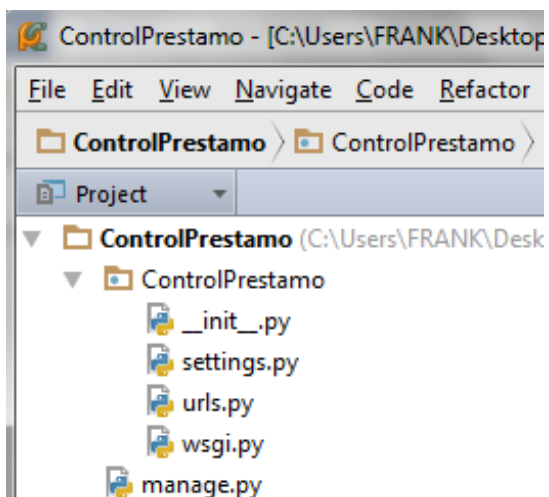
1) Crear Proyecto ControlPrestamo

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, 'prestamodb.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

```

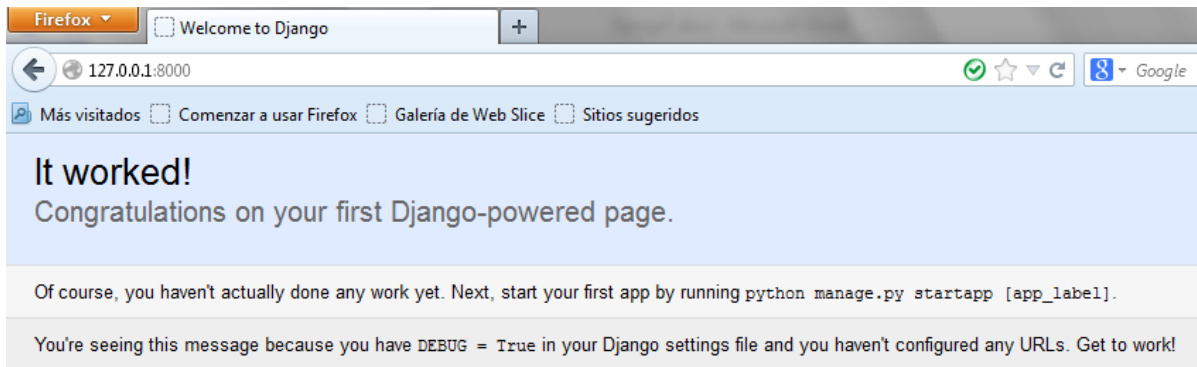
C:\Users\FRANK\Desktop\pythonProject\ControlPrestamo>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

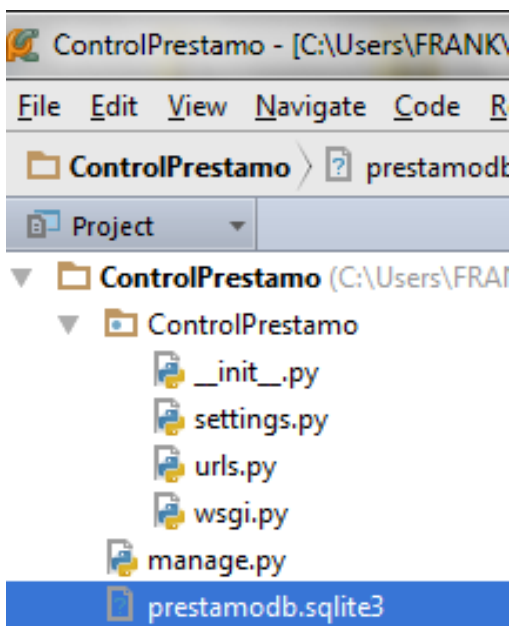
You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.
April 23, 2016 - 21:31:55
Django version 1.7.4, using settings 'ControlPrestamo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

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

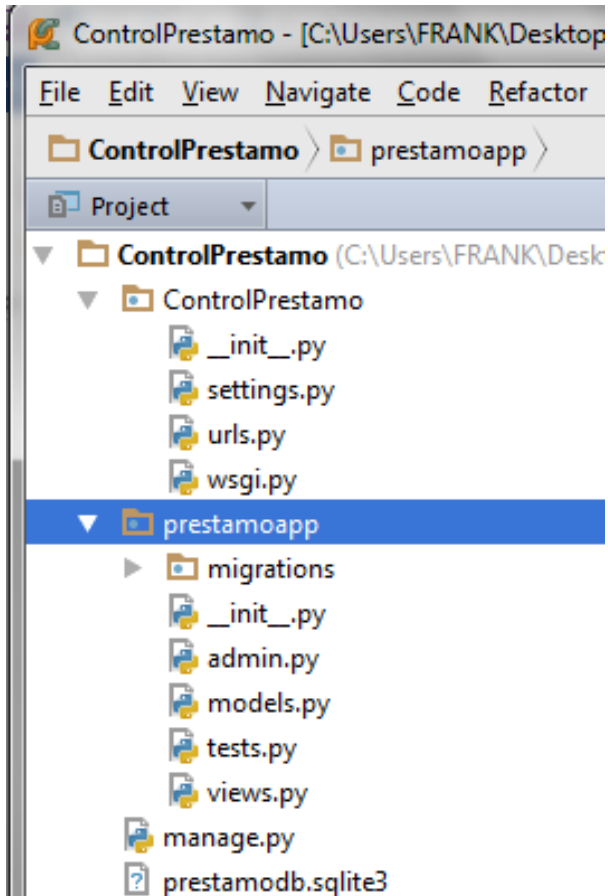
django-admin startapp prestamoapp

```

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.

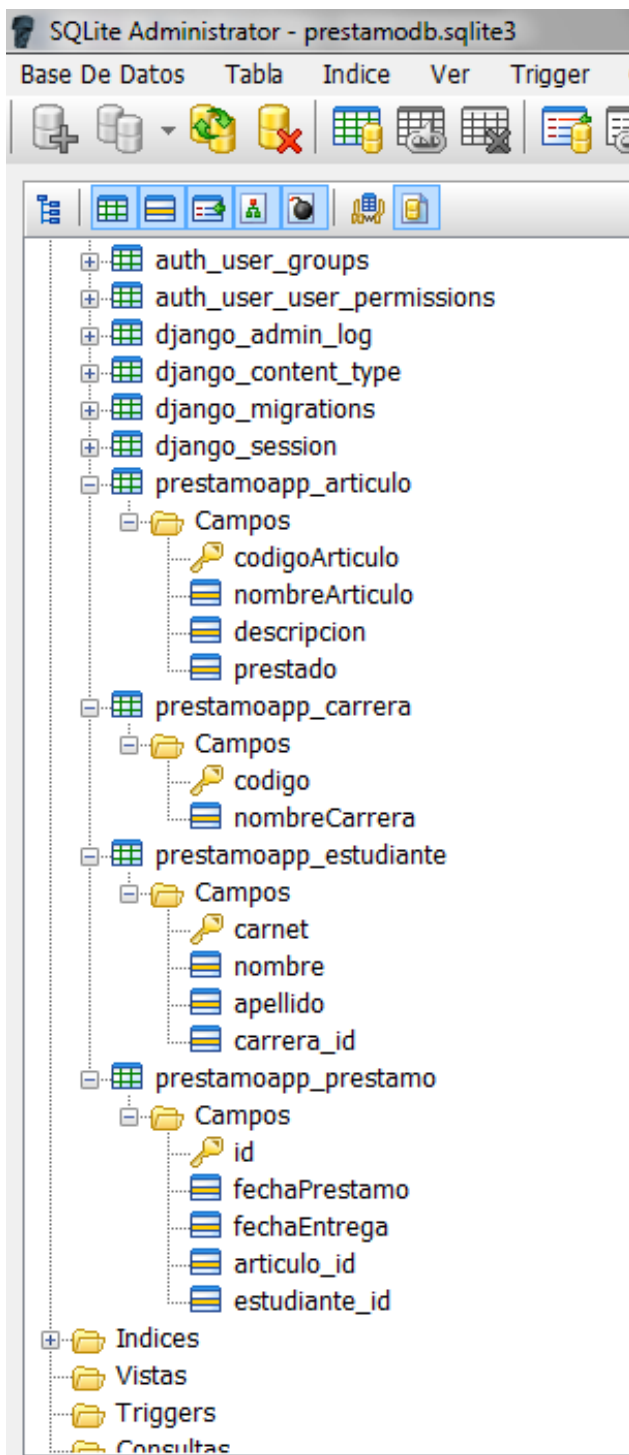
python manage.py migrate : realiza cambios en la base de datos.

```
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:

nonbreapp_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

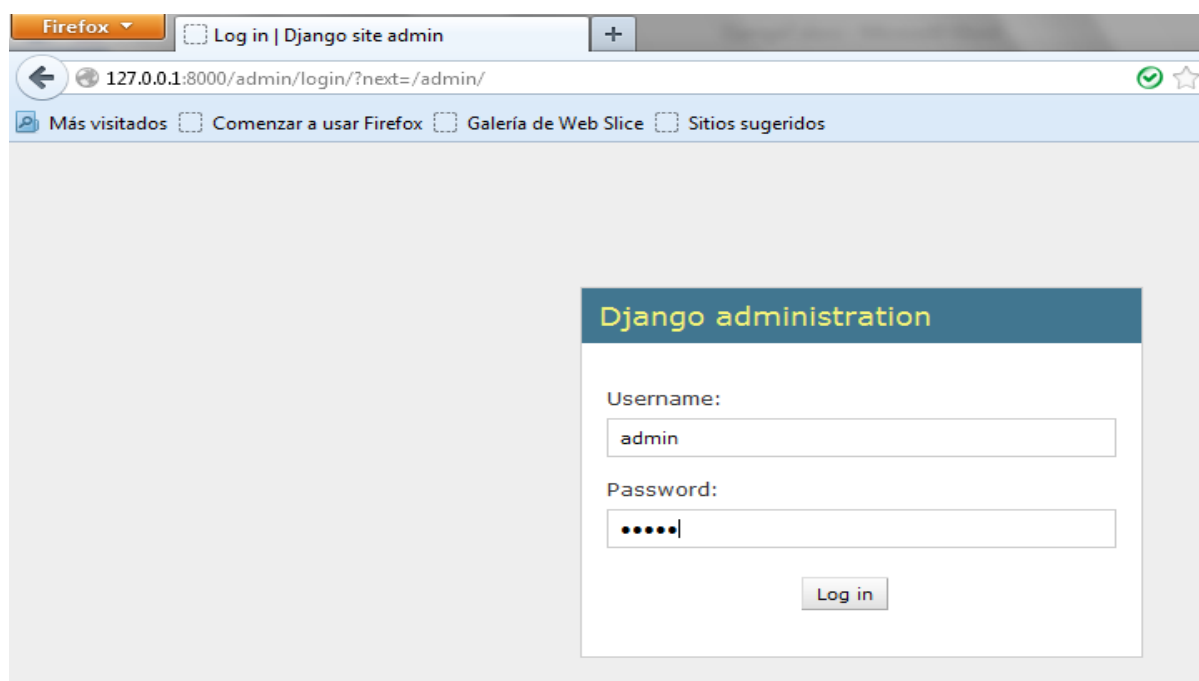
Ingresa como usuario **admin** y contraseña **admin**, además ingresa tu correo

```
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>
```

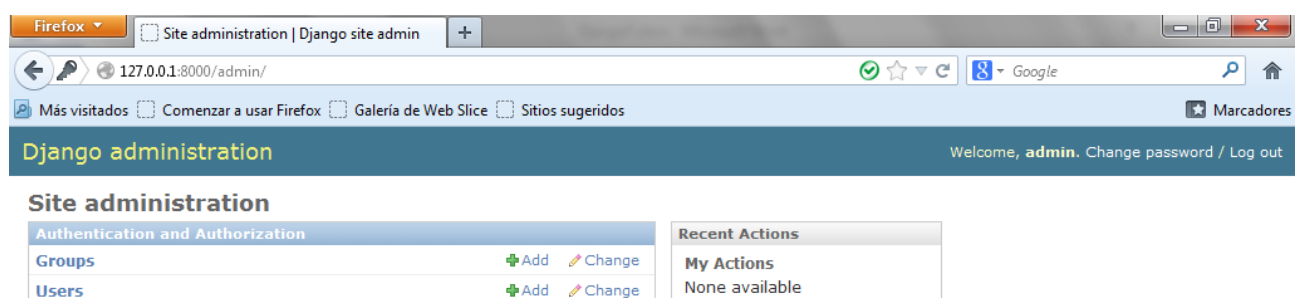
Ahora corremos la aplicación:

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



Authentication and Authorization	
Groups	Add Change
Users	Add Change

Recent Actions
My Actions
None available

Si ves no aparecen los modelos que que necesitamos.

Configurando el admin de django

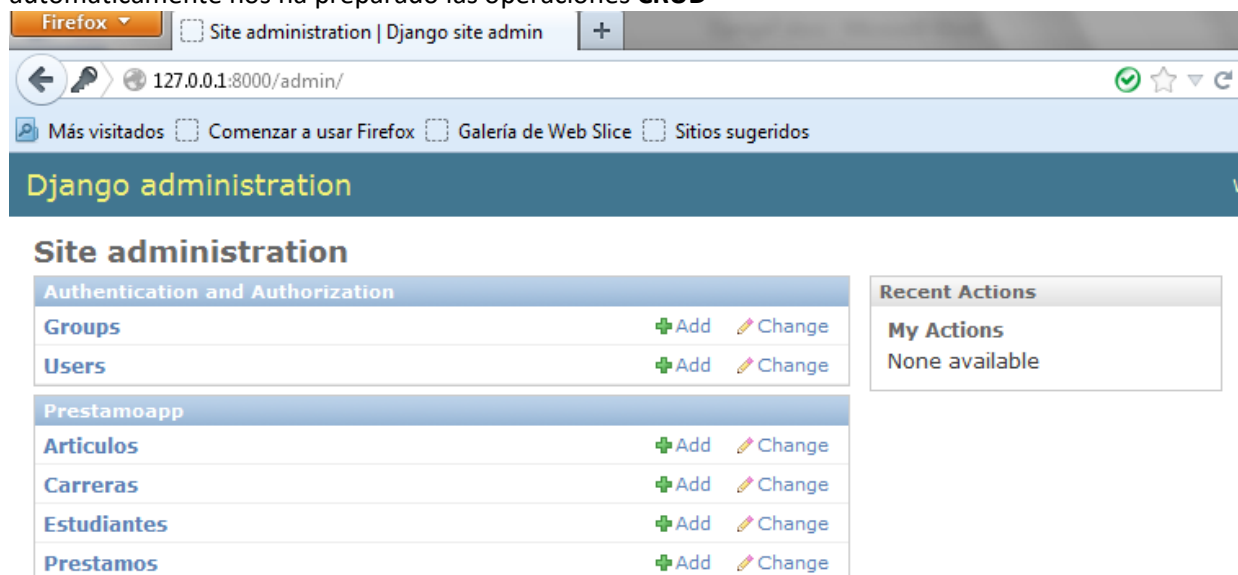
Abre el archivo **admin.py** y escribe el siguiente código para que registre los modelos en el admin

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

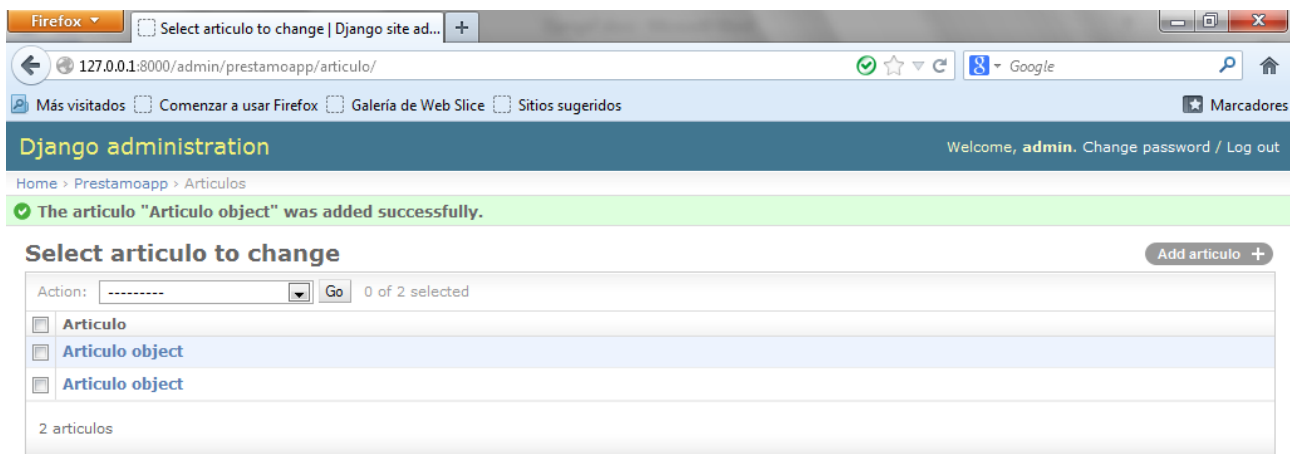
# Register your models here.

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
```

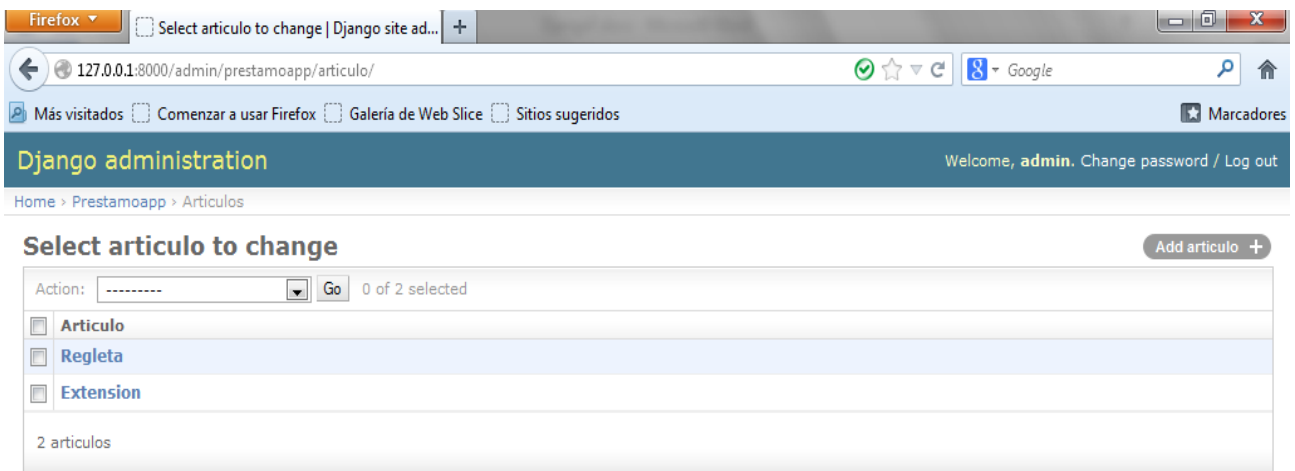
```
class Estudiante(models.Model):
    carnet=models.CharField(max_length=7,primary_key=True)
    carrera=models.ForeignKey(Carrera,null=False)
    nombre=models.CharField(max_length=30,null=False)
    apellido=models.CharField(max_length=30,null=False)

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

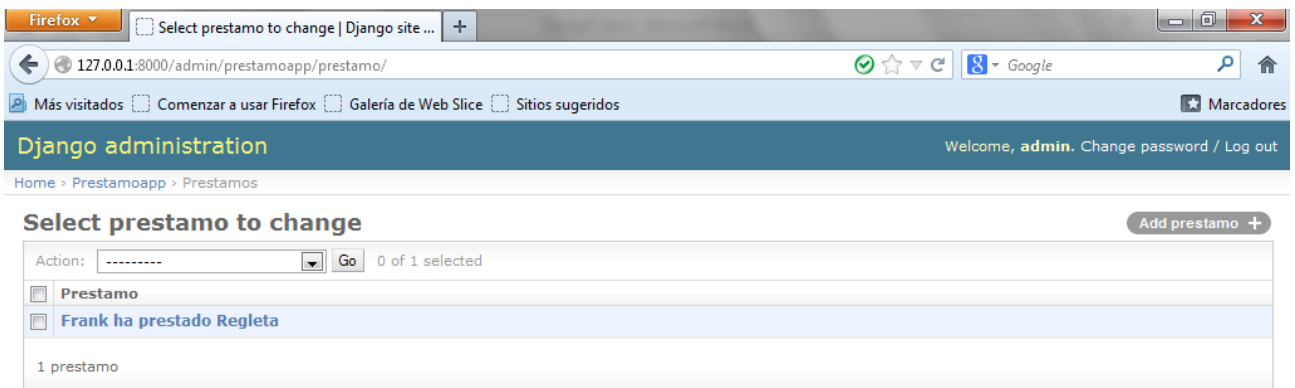
```
class Prestamo(models.Model):
    articulo = models.OneToOneField(Articulo,null=False)
    estudiante=models.ForeignKey(Estudiante,null=False)
```

```
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

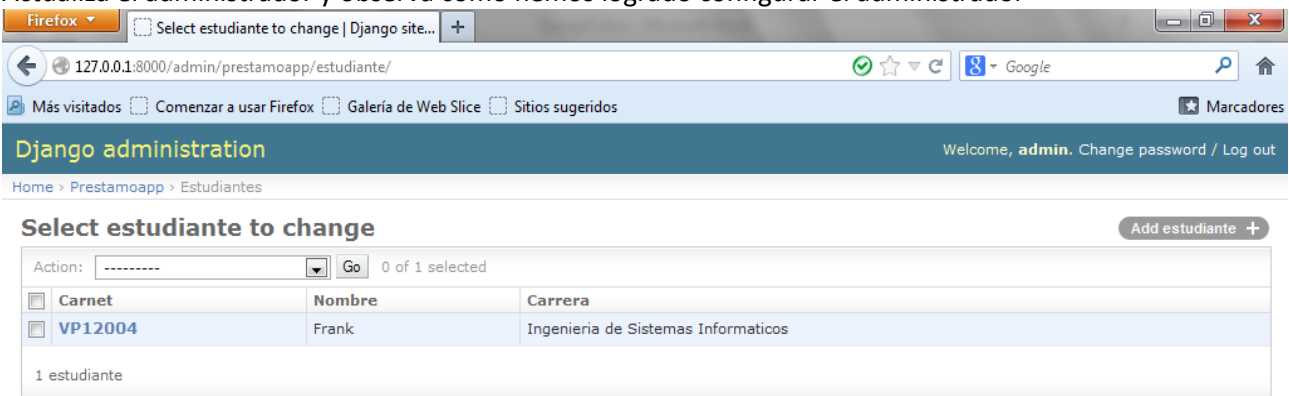
class PrestamoAdmin(admin.ModelAdmin):
    list_display =
('articulo', 'estudiante', 'fechaPrestamo', 'fechaEntrega')

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

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

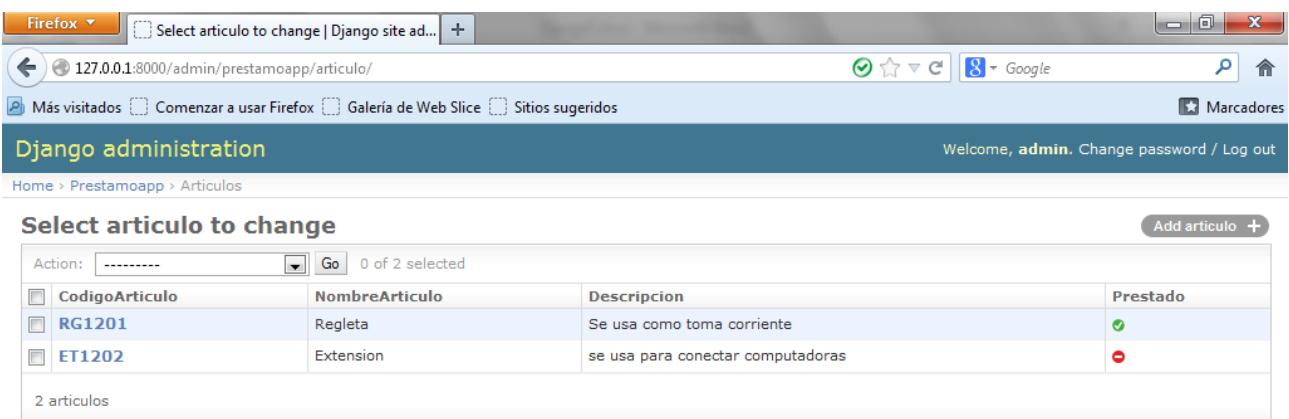
admin.site.register(Artículo, ArtículoAdmin)
admin.site.register(Carrera, CarreraAdmin)
admin.site.register(Estudiante, EstudianteAdmin)
admin.site.register(Prestamo, PrestamoAdmin)
```

Actualiza el administrador y observa como hemos logrado configurar el administrador



Carnet	Nombre	Carrera
VP12004	Frank	Ingeniería de Sistemas Informáticos

1 estudiante



CodigoArticulo	NombreArticulo	Descripcion	Prestado
RG1201	Regleta	Se usa como toma corriente	✓
ET1202	Extension	se usa para conectar computadoras	✗

2 articulos

Patron MVT (Model View Template)

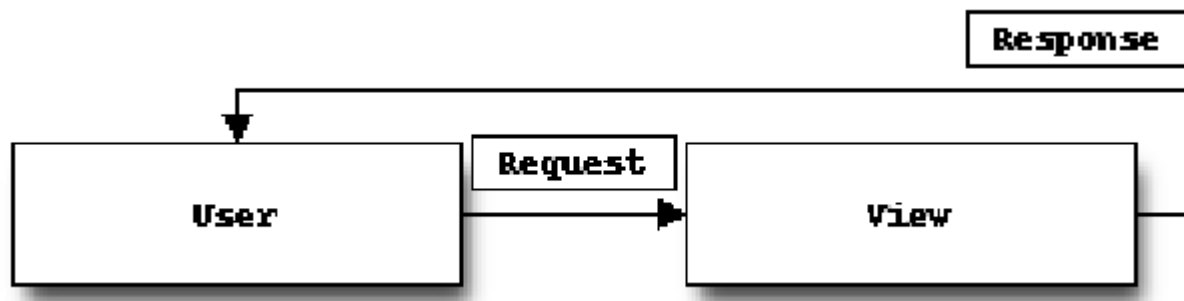
M significa Model (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

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

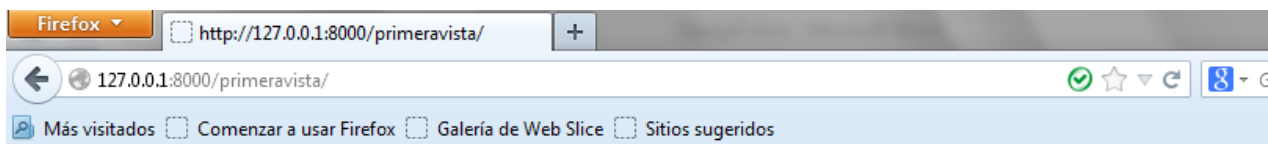
urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'ControlPrestamo.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    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() })
```

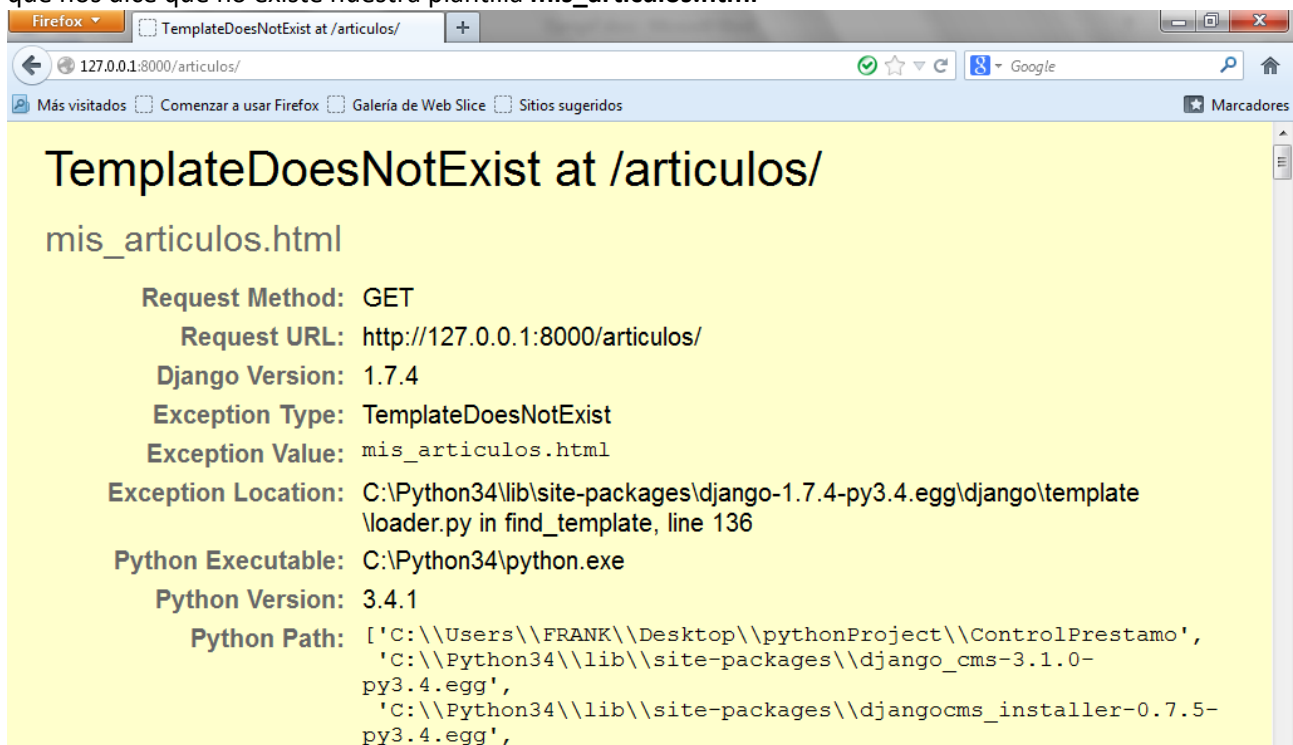
Registramos nuestra vista en el archivo **urls.py**

```
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')),

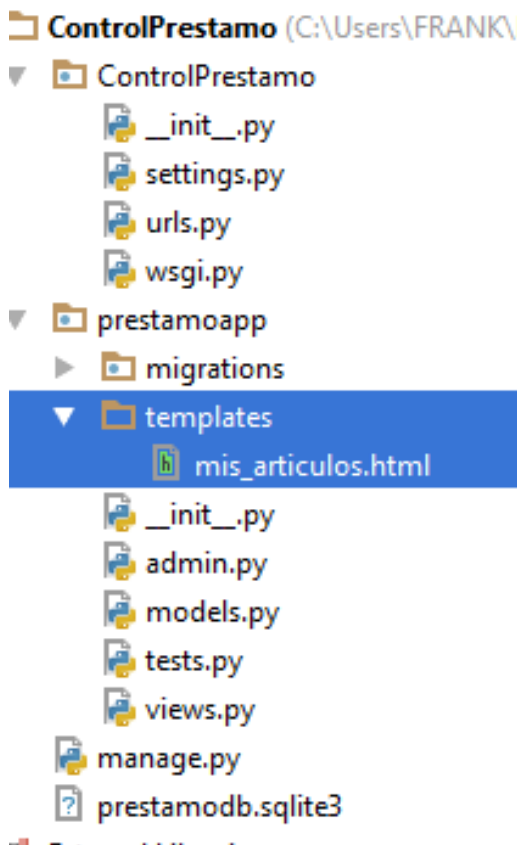
    url(r'^admin/', include(admin.site.urls)),
    url(r'^primeravista/$', 'prestamoapp.views.mi_primer_vista'),
    url(r'^articulos/$', misArticulos)
)
```

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**

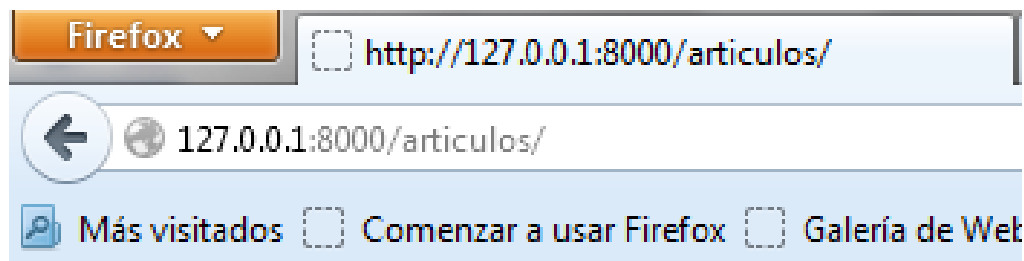


Para resolverlo, primeramente debemos saber dónde ubicar nuestras plantillas html en nuestro proyecto, django por defecto buscara las plantillas html en una carpeta llamada **templates** que se ubica dentro de nuestra aplicación. Si quieres ponerle otro nombre revisa la documentación de django, aca trabajaremos 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

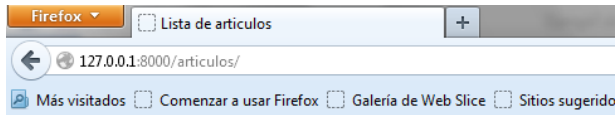
```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>Lista de articulos</title>
</head>
<body>
  <h1>Bienvenidos</h1>
  <h2>Lista de articulos</h2>

  <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



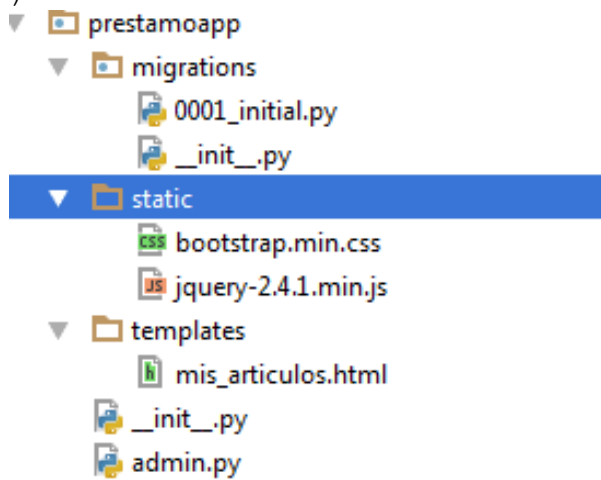
Bienvenidos

Lista de artículos

Codigo	Nombre	Descripcion	Estado
RG1201	Regleta		Prestado
ET1202	Extension		Disponible

Lo que haremos a continuación es configurar en el **settings.py** la carpeta donde debe de buscar los archivos estáticos como imágenes, archivos css, javascript, etc, además crearemos la carpeta **static** dentro de la que pondremos archivos jquery y bootstrap para la apariencia de nuestro sitio.

```
STATICFILES_DIRS = (  
    os.path.join(BASE_DIR, "prestamoapp/static"),  
)
```



De igual forma puedes configurar la carpeta para los templates, consultando a documentación si no quieres que se llame **templates**.

Herencia de plantillas

La herencia de plantillas te deja construir una plantilla base esqueleto que contenga todas las partes comunes de tu sitio y definir bloques que los hijos puedan sobrescribir.

menu	
titulo	
content	aside
footer	

Nosotros usaremos una parecida a la anterior.

Mas o mmenos quedara como la siguiente:

```

<!DOCTYPE html>

<html lang="en" >
<head>
<title></title>
</head>
<body>

{ % block menu %}{ % endblock %}

{ % block titulo %}{ % endblock %}
{ % block content %} contenido principal { % endblock %}

{% block aside %}{% endblock%}


{ % block footer %}
<hr>
<p>Gracias por visitar nuestro sitio</p>
{ % endblock %}
</body>
</html>

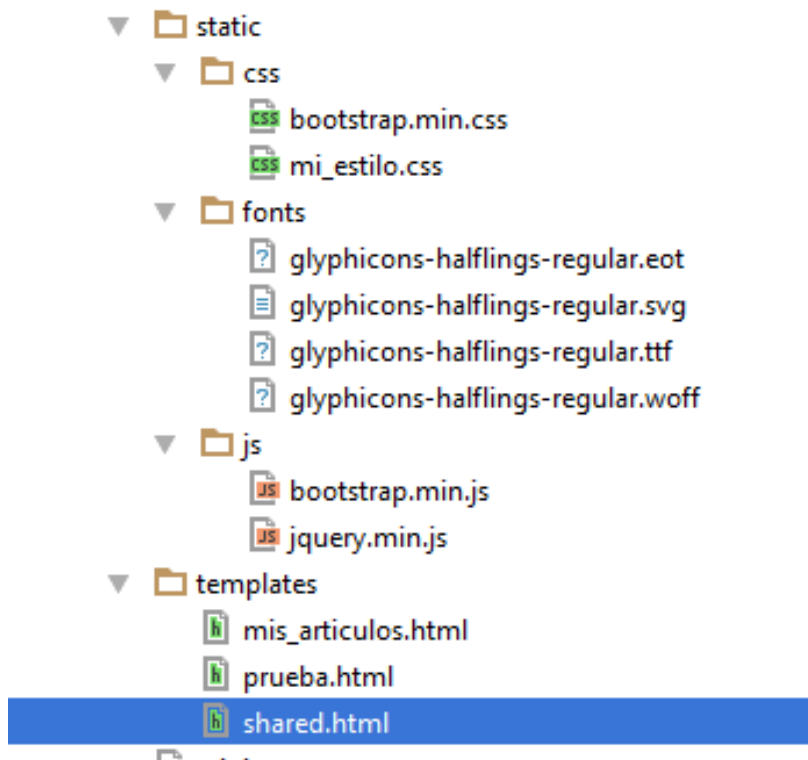
```

Esqueleto HTML simple que usaremos para todas las páginas del sitio. Es trabajo de las plantillas hijas sobrescribir, agregar, dejar vacío el contenido de los bloques.

Usamos una etiqueta de plantilla aquí que no hemos visto antes: la etiqueta **{ % block %}**. Todas las etiquetas **{ % block %}** le indican al motor de plantillas que una plantilla hijo quizás sobrescriba esa porción de la plantilla.

En la carpeta **templates** crea un archivo html llamado **shared**(compartido) y en la carpeta **static** crea dos carpetas una llamada **css** y la otra llamada **js**, en la carpeta css y js pondremos archivos de **bootstrap** y **jquery** para que las usaremos para las interfaces de usuario de nuestro proyecto, puedes descargar las bibliotecas de estos frameworks web, cuando descargues bootstrap también vendrá una carpeta de **fonts**, copia esta carpeta a tu proyecto ya que nos servirá para los iconos.

La estructura del proyecto quedara como la siguiente



El archivo **shared.html** quedara como el siguiente

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,user-
acalable=no, inicial-scale=1.0, maximun-scale=1.0, minimun-
scale=1.0"/>
  <title>Control de prestamos</title>
  <link rel="stylesheet" href="/static/css/bootstrap.min.css">
  <link rel="stylesheet" href="/static/css/mi_estilo.css">
  <script src="/static/js/jquery.min.js"></script>
  <script src="/static/js/bootstrap.min.js"></script>

</head>
<body>
  <!-- Aca se define el header de nuestro sitio con el bloque menu
  -->
  {% block menu %}
    <header>
      <nav class="navbar navbar-inverse navbar-static-top navbar-
header-background" role="navigation">
        <div class="container">
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#miNavegacion">
              <span class="sr-only">Desplegar/ocultar</span>
              <span class="icon-bar"></span>
            </div>
          </div>
        </div>
      </nav>
    </header>
  </block>
</body>
</html>
```

```

        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <a href="" class="navbar-brand">Control de
prestamos</a>
</div>
<!-- aca inicia el menu -->
<div class="collapse navbar-collapse" id="miNavegacion">
    <ul class="nav navbar-nav">

        <li class="dropdown">
            <a href="" class="dropdown-toggle" data-
toggle="dropdown" role="button">
                Categorías <span class="caret"></span>
            </a>
            <ul class="dropdown-menu" role="menu">
                <li><a href="">Carrera</a></li>
                <li><a href="">Artículo</a></li>
                <li class="divider"></li>
                <li><a href="">Estudiante</a></li>
                <li><a href="">Prestamo</a></li>
            </ul>
        </li>

        <ul class="nav navbar-nav">
            <li><a href="/">Home</a> </li>

            <li><a href="/about">About</a> </li>

            <li><a href="/help">Help</a> </li>
            <li><a href="/admin">Log in</a> </li>

        </ul>

        </ul>

        <form action="" class="navbar-form navbar-right"
role="search">
            <div class="form-group">
                <input type="text" placeholder="Buscar"
class="form-control"/>
            </div>
            <button type="submit" class="btn btn-primary"
style="">
                <span class="glyphicon glyphicon-search"></span>
            </button>
        </form>
    </div>

```

```
</div>
</nav>
</header>
{% endblock %}
```

```
<!-- aca se define el titulo de nuestra pagina -->
<div class="jumbotron">
  {% block titulo %}
    <h3>Administacion de Prestamos</h3>
    <p>Bienbenidos al asistente para la administracion de
prestamos</p>
  {% endblock %}
</div>
```

```
<div class="container">
  <div class="row">
    <section class="col-md-9">
      {% block content %}
        <h4>Este es el contenido principal del sitio</h4>
      {% endblock %}
    </section>
    <aside class="col-md-3 hidden-xs hidden-sm">
      {% block aside %}
        <h4>Categorias</h4>
        <div class="list-group">
          <a href="" class="list-group-item
active">Prestamo App</a>
          <a href="" class="list-group-
item">Articilo</a>
          <a href="" class="list-group-
item">Carrera</a>
          <a href="" class="list-group-
item">Estudiante</a>
          <a href="" class="list-group-
item">Prestamo</a>
          <a href="" class="list-group-item">Otros</a>
        </div>
        <h4>Articulos recientes</h4>
        <a href="" class="list-group-item">
          <h4 class="list-group-item-heading">Gestion de
usuarios</h4>
          <p class="list-group-item-text">
            Capacitate para la gestion de usuario
          </p>
        </a>
      {% endblock %}
    </aside>
  </div>
```

```

</div>
<footer>
  <div class="container">
    <div class="row">
      {% block footer %}
        <div class="col-xs-6">
          <p>Vasquez Perez Francisco</p>
        </div>
        <div class="col-xs-6">
          <ul class="list-inline text-right">
            <li><a href="">Home</a></li>
            <li><a href="">Capacitaciones</a></li>
            <li><a href="">Contactos</a></li>
          </ul>
        </div>
      {% endblock %}
    </div>
  </div>
</footer>
</body>
</html>

```

El archive **mi_estilo.css** contendra lo siguiente

```

input, texarea, select{
  max-width: 280px;
}

.navbar{
  margin-bottom: 1px;
}

body{

}

.jumbotron{
  margin-left: 5px;
  margin-right: 5px;
  padding-top: 5px;
  padding-bottom: 5px;
  border-radius: 10px;
  text-align: center;
}

.navbar-header-background{
  background: black;
}

```

ahora que ya tenemos una plantilla padre crearemos nuestras plantillas html y que extiendan de shared.html, crea un archivo html llamado **préstamo.html** y agregale la siguiente línea de código

```
{% extends "shared.html" %}
```

Agrega una vista de prueba el archivo **views.py**

```
def probando(request):  
    return render(request, "prestamo.html")
```

Ve al archivo **urls.py** y registra la vista

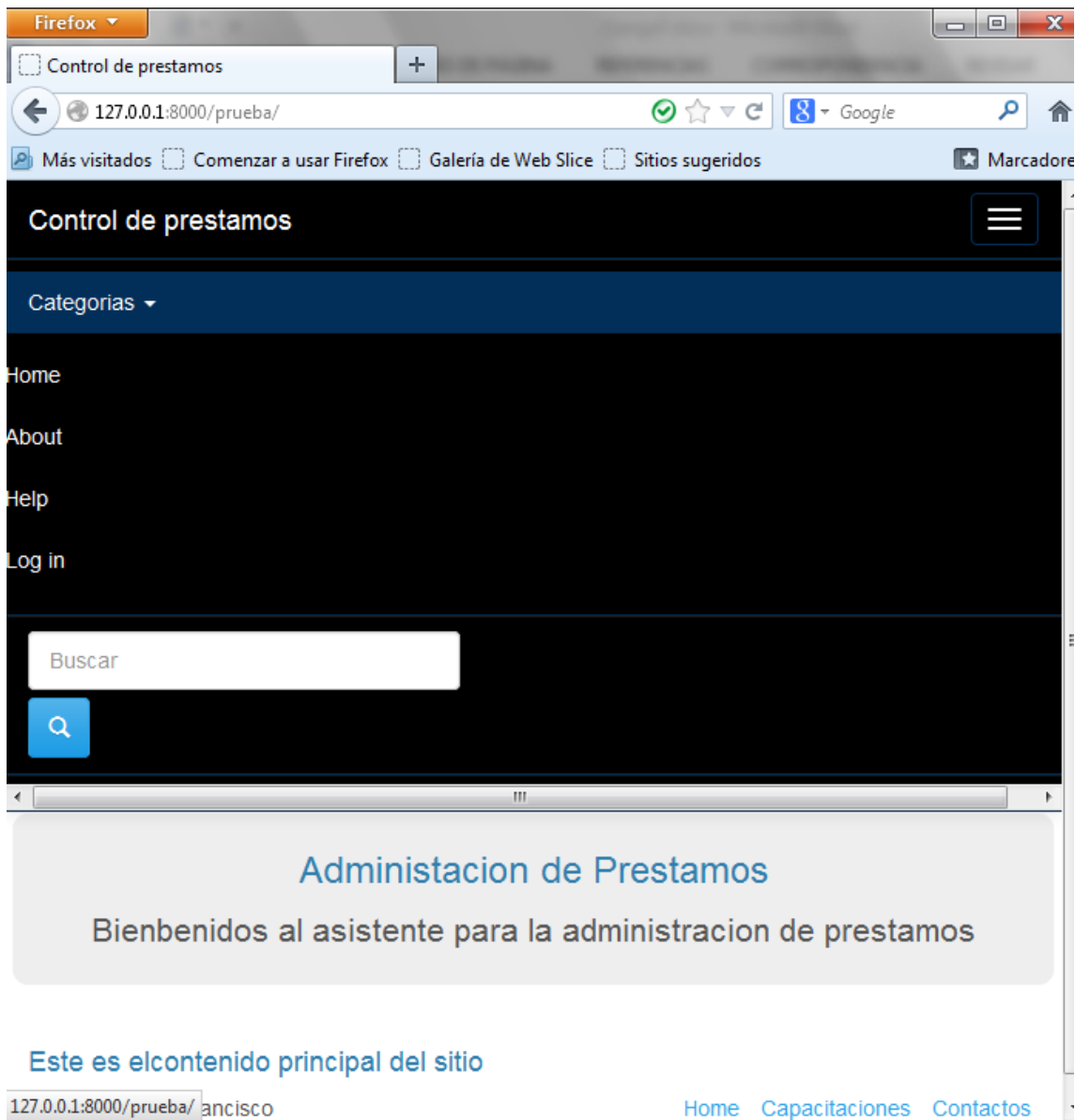
```
from django.conf.urls import patterns, include, url  
from django.contrib import admin  
from prestamoapp.views import misArticulos, probando  
  
urlpatterns = patterns('',  
    # Examples:  
    # url(r'^$', 'ControlPrestamo.views.home', name='home'),  
    # url(r'^blog/', include('blog.urls')),  
  
    url(r'^admin/', include(admin.site.urls)),  
    url(r'^primeravista/$', 'prestamoapp.views.mi_primer_vista'),  
    url(r'^articulos/$', misArticulos),  
    url(r'^prueba/$', probando),  
  
)
```

corre el servidor y abre la siguiente direccion

<http://127.0.0.1:8000/prueba/>



si reduces el tamaño de la ventana veras como se adapta al tamaño, así es como logramos la herencia de plantillas en django. Posteriormente vamos a realizar algunas modificaciones.



Vistas genéricas

Son vistas que proveen alguna funcionalidad básica como renderizar un template, crear o editar un modelo, etc.

Vistas basadas en clases

Son vistas basadas en vistas genéricas que nos dan alguna funcionalidad básica como hacer el CRUD.

Listar artículo

Abre el archivo views.py

```
from django.shortcuts import render

from django.http import HttpResponse
from prestamoapp.models import Artículo
from django.views.generic import ListView
from prestamoapp.models import Prestamo,Articulo
```

```

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()})

def probando(request):
    return render(request, "prestamo.html")

class ListArticuloView(ListView):
    model = Articulo
    template_name = "articulo_list.html"

```

Abre el archive urls.py

```

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

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),
    url(r'^prueba/$', probando),
    url(r'^$', ListArticuloView.as_view(), name="articulo-list"),
)

```

Crea el archivo articulo_list.html

```

{% extends "shared.html" %}

{% block titulo %}
<h3>Administracion de Articulos</h3>
<p>Bienvenidos al administrador de Articulos</p>
{% endblock %}

{% block content%}
<div class="table-responsive">
    <table class="table table-hover">
        <tr class="success">
            <th>codigo</th>
            <th>articulo</th>
            <th>Estado</th>
        </tr>

```

```

{% for articulo in object_list %}
    <tr>

        <td>{{ articulo.codigoArticulo}}</td>
        <td>{{ articulo.nombreArticulo}}</td>
        {% if articulo.prestado == False %}
            <td>Disponible</td>
        {% endif %}
        {% if articulo.prestado == True %}
            <td>Prestado</td>
        {% endif %}

    </tr>
{% endfor %}
</table>
</div>
{% endblock %}

```

Abre la dirección <http://127.0.0.1:8000>

Firefox Control de prestamos

127.0.0.1:8000

Más visitados Comenzar a usar Firefox Galería de Web Slice Sitios sugeridos

Control de prestamos Categorías Home About Help Log in

Buscar

Administracion de Articulos

Bienvenidos al administrador de Articulos

codigo	articulo	Estado
RG1201	Regleta	Prestado
ET1202	Extension	Disponible

Categorias

- Prestamo App
- Articulo
- Carrera
- Estudiante
- Prestamo
- Otros

Articulos recientes

Gestion de usuarios
Capacitate para la gestion de usuario

Vasquez Perez Francisco

Home Capacitaciones Contactos

Crear artículo

Abre el archivo views.py y escribe la siguiente vista

```
from django.core.urlresolvers import reverse
from django.views.generic import ListView, CreateView
from prestamoapp.models import Prestamo, Artículo

class CreateArticuloView(CreateView):
    model = Artículo
    template_name = 'edit_articulo.html'

    def get_success_url(self):
        return reverse('articulo-list')
```

registra la vista en el archive urls.py

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

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),
    url(r'^prueba/$', probando),
    url(r'^$', ListArticuloView.as_view(), name="articulo-list",),

    url(r'^newArticulo/$', CreateArticuloView.as_view(), name="articulo-
new",),
    url(r'^prestamos/$', ListPrestamoView.as_view(), name="prestamo-
list",),

)
```

Crea el archivo edit_articulo.html

```
{% extends "shared.html" %}

{% block titulo%}
{%endblock%}
```

```

{% block content %}
    <h3>Nuevo articulo</h3>
    <hr/>
    <form action="{% url 'articulo-new' %}" class="form-horizontal"
method="post">

        {% csrf_token %}

        {%for element in form %}
            <div class="form-group">
                <label class="col-md-2 control-label">{{element.label}}</label>
                <div class="col-md-7">
                    {{element}}
                </div>
            </div>
        {% endfor %}
        <div class="col-md-offset-2 col-md-7">
            <button id="save_articulo" type="submit" class="btn btn-
default">Guardar</button>
            <a href="{% url 'articulo-list' %}" class="btn btn-
default">Volver a la lista</a>
        </div>
    </form>

{%endblock%}

```

Abre el archivo articulo_list.html y agrega la siguiente línea al principio

```

{% extends "shared.html" %}

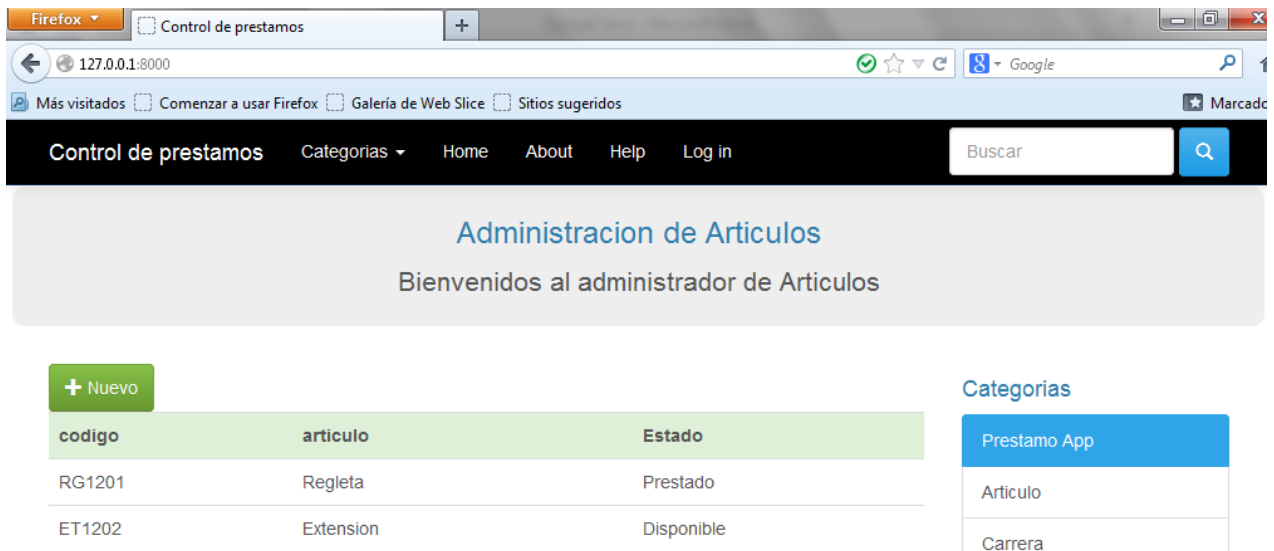
{% block titulo %}
    <h3>Administracion de Articulos</h3>
    <p>Bienvenidos al administrador de Articulos</p>
{% endblock %}

{% block content%}
    <a href="{% url 'articulo-new' %}" class="btn btn-
success"><span class="glyphicon glyphicon-plus"></span> Nuevo</a>

    . . .
{% endblock %}

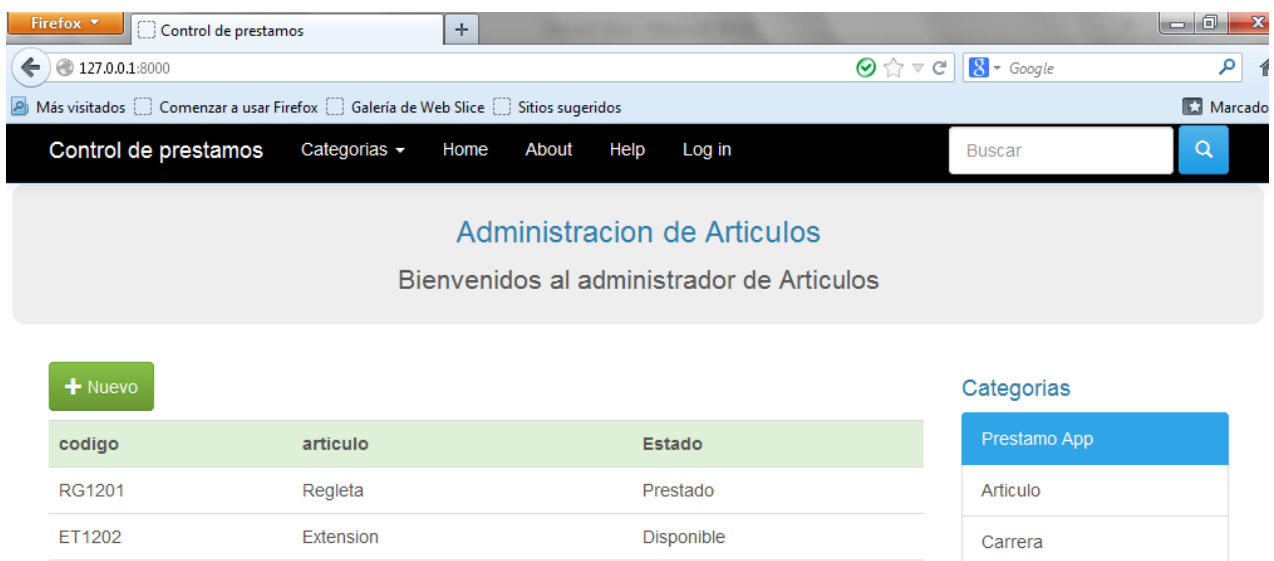
```

Si abres el listado de articulos veras como nos aparece el botón de crear un Nuevo articulo.

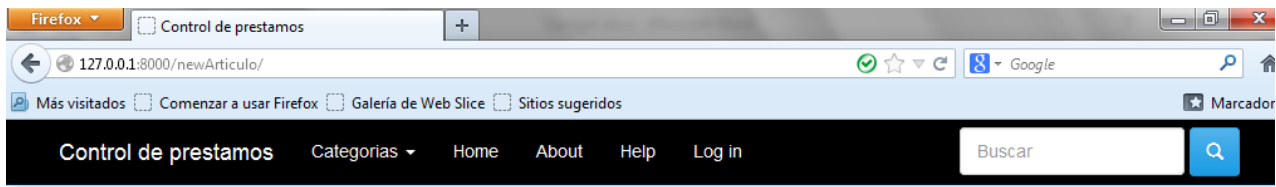


Sin embargo no se ve bien la apariencia, abre el archivo `mi_estilo.css` agrega el siguiente código para que se vea bien

```
.table{  
  margin-top: 7px;  
}
```



Pulsa el botón crear nuevo



Vasquez Perez Francisco

[Home](#) [Capacitaciones](#) [Contactos](#)

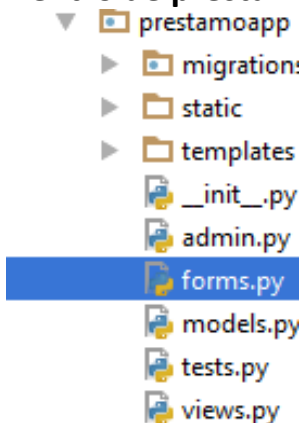
Puedes crear un nuevo artículo, ingresando los datos, recuerda que el campo descripción lo definimos como requerido por tanto es necesario que ingreses una descripción. También puedes cancelarlo dando click en el botón volver a la lista

Hasta ahora vemos ya podemos guardar, sin embargo, parte de la calidad del software se debe a que los datos sean correctos, para probar un software debe de hacerse ingresando datos incorrectos, por ejemplo nombre de una persona con números.

Validar los datos de los formularios

En el formulario anterior debemos personalizar las etiquetas de los formularios, y capturar los errores de datos incorrecto, así mismo como validar.

Dentro de prestamoapp crea otro archivo llamado forms.py



```

from django import forms
from django.core.exceptions import ValidationError
from prestamoapp.models import Artículo
import re

class ArtículoForm(forms.ModelForm):

    #Personalizamos los campos

    codigoArticulo=forms.CharField(max_length=6,required=True,label="Codigo",help_text="Codigo de Artículo")

    nombreArticulo=forms.CharField(max_length=100,required=True,label="Nombre")
    descripcion =
forms.CharField(max_length=255,required=True,label="Descripcion")
    prestado = forms.BooleanField(label="Prestado",required=False)

class Meta:
    model = Artículo

    def clean_codigoArticulo(self):
        #Capturamos el valor de codigo Artiiculo
        codigoArticulo= self.cleaned_data.get('codigoArticulo')

        #Comparamos que no tenga menos de 6 caracteres
        if len(codigoArticulo)<6:
            raise forms.ValidationError("Debe tener 6 caracteres")
        elif (re.match("[a-zA-z][a-zA-z]\d\d\d\d",codigoArticulo))
== None:
            #Validamos que las 2 primeros caracteres sean letras y
            4 numeros
            raise forms.ValidationError("error: debe ser dos
caracteres seguido de 4 numeros")

        return codigoArticulo

        #Validamos que el nombre empiece con un caracter y luego se
        alfanumerico
        # A-Z,a-z,0-9
        def clean_nombreArticulo(self):
            nombreArticulo = self.cleaned_data.get('nombreArticulo')
            #Expresion regular a-zA-Z], comprueba que el primer
            caracter sea una letra
            if(re.match("[a-zA-Z]",nombreArticulo)==None):
                raise forms.ValidationError("Debe iniciar con un
caracter")
            # "\w*" verifica que todos los caracteres sean
            alfanumericos
            # \w es lo mismo que [a-z0-9-A-Z]
            elif (re.match("\w*",nombreArticulo)) ==None:
                raise forms.ValidationError("Deben de ser caracteres

```



```

alfanumerico")
    return nombreArticulo

#validamos que la descripcion del articulo no sea menor que 4
def clean_descripcion(self):
    descripcion = self.cleaned_data.get('descripcion')
    num_words = len(descripcion.split())
    if num_words < 4:
        raise forms.ValidationError("Deben de ser minimo 4
palabras!")
    return descripción

```

con esto logramos personalizar el formulario, nota que tenemos todos los campos de nuestro modelo, la diferencia es que ahora la clase hereda ModelForm, además que los atributos ya no dependen de models, sino de forms y que algunos atributos ya no están incluidos y hemos agregados otros como required y label, Otro punto interesante es también que le indicamos el modelo al al que pertenece el formulario.

Abre el archivo views.py y realiza la siguiente modificación

```

from prestamoapp.forms import ArtículoForm

class CreateArticuloView(CreateView):
    model = Artículo
    template_name = 'edit_articulo.html'
    form_class = ArtículoForm

    def get_success_url(self):
        return reverse('articulo-list')

```

abre el archivo edit_articulo.html y agrega lo siguiente

```
{% extends "shared.html" %}
```

```
{% block titulo%}
{% endblock %}
```

```

{% block content %}
<h3>Nuevo articulo</h3>
<hr/>
<form action="{% url 'articulo-new' %}" class="form-horizontal"
method="post">

```

```
    {% csrf_token %}
```

```

    {% for element in form %}
        <div class="form-group">
            <label class="col-md-2 control-label">{{ element.label }}</label>

```

```

<div class="col-md-7">
    {{element}}
    <div class="error">{{element.errors}}</div>

</div>

</div>
{% endfor %}
<div class="col-md-offset-2 col-md-7">
<button id="save_articulo" type="submit" class="btn btn-
default">Guardar</button>
    <a href="{% url 'articulo-list' %}" class="btn btn-
default">Volver a la lista</a>
</div>
</form>

{%endblock%}

```

Si ves al div donde se captura el error tiene una clase llamada error, esto es para que el texto lo muestre en rojo, abre el archivo mi_estilo.html agrega el siguiente código

```

.error{
    color: red;
}

```

Ingresa a crear un nuevo articulo y veras los errores si lo pruebas
 Listar prestamos

Nuevo articulo

Codigo	<input type="text" value="0A1500"/>	• error: debe ser dos caracteres seguido de 4 numeros
Nombre	<input type="text" value="3articulo"/>	• Debe iniciar con un caracter
Descripcion	<input type="text" value="es un articulo"/>	• Deben de ser minimo 4 palabras!
Prestado	<input type="checkbox"/>	

Ver detalle de articulos

Abre el archive views.py

```
from django.views.generic import DetailView
```

```
class DetailArticulo(DetailView):  
    model = Articulo  
    template_name = 'detalle_articulo.html'
```

Abre el archive urls.py

```
from prestamoapp.views import CreateArticuloView, DetailArticulo
```

```
url(r'^(?P<pk>\w+)/$', DetailArticulo.as_view(), name='articulo-  
view' ),
```

Nota: ten cuidado con el tipo de datos que se le va a pasar a la expresión regular en nuestro caso como la llave primaria de nuestro modelo es string entonces la expresión debe ser `\w+`, pero si fuera un numero seria como la siguiente `\d+`

```
url(r'^(?P<pk>\d+)/$', DetailArticulo.as_view(), name='articulo-  
view' ),
```

Lo que haremos es ir al archive models.py y agregarle un metodo que nos devuelva la url con el valor de la clave primaria.

```
from django.core.urlresolvers import reverse
```

```
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 get_absolute_url(self):  
        return reverse('articulo-  
view',kwargs={'pk':self.codigoArticulo})
```

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

crea el archivo detalle_articulo.html

```
{% extends "shared.html" %}  
  
{% block titulo%}{% endblock %}  
{% block content %}  
<h3>Detalle {{ articulo }}</h3>
```

```

<hr/>
<dl class="dl-horizontal">

  <dt>Codigo: </dt>
  <dd>{{articulo.codigoArticulo }}</dd>

  <dt>Nombre: </dt>
  <dd>{{ articulo.nombreArticulo }}</dd>

  <dt>Descripcion: </dt>
  <dd>{{articulo.descripcion}}</dd>

  <dt>Estado: </dt>
  {% if articulo.prestado == False %}
    <dd>Disponible</dd>
  {% endif%}
  {% if articulo.prestado == True %}
    <dd>Prestado</dd>
  {% endif%}

</dl>
<a href="{% url 'articulo-list' %}" class="btn btn-
default">Vover a la lista</a>
{% endblock %}

```

Abre el archivo articulo_list.html y agrega el siguiente código

```

<table class="table table-hover">
  <tr class="success">

    <th>codigo</th>
    <th>articulo</th>
    <th>Estado</th>
    <th></th>

  </tr>
  {% for articulo in object_list %}
    <tr>

      <td>{{ articulo.codigoArticulo}}</td>
      <td>{{ articulo.nombreArticulo}}</td>
      {% if articulo.prestado == False %}
        <td>Disponible</td>
      {% endif%}
      {% if articulo.prestado == True %}
        <td>Prestado</td>
      {% endif%}
      <td><a href="{% articulo.get_absolute_url %}" class="btn
btn-default btn-xs"><span class="glyphicon glyphicon-th-
large"></span></a></td>

    </tr>

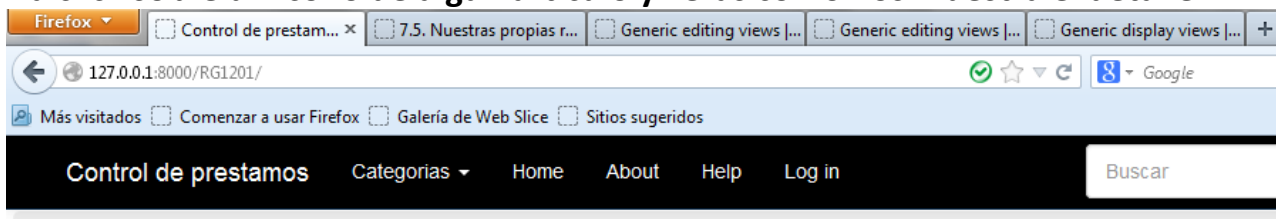
```

```
{% endfor %}
</table>
```

Actualiza el navegador



Da click sobre un icono de algún artículo y veras como nos muestra el detalle



Detalle Regleta

Codigo: RG1201
Nombre: Regleta
Descripcion: Se usa como toma corriente
Estado: Prestado

Vover a la lista

Categorías

Prestamo App

Articulo

Carrera

Estudiante

Prestamo

Eliminar Articulo

Abre el archivo views.py

```
from django.views.generic import DetailView, DeleteView
```

```
class DeleteArticulo(DeleteView):
    model = Articulo
    template_name = 'delete_articulo.html'
    def get_success_url(self):
        return reverse('articulo-list')
```

abre el archivo urls.py

```
from prestamoapp.views import DeleteArticulo
```

```
url(r'^delete/(?P<pk>\w+)/$', DeleteArticulo.as_view(),  
    name='articulo-delete'),
```

crea el archivo delete_articulo.html

```
{% extends "shared.html" %}  
{% block titulo %}{% endblock %}  
{% block content %}  
<h3>Eliminar Articulo</h3>  
<hr/>  
<p>Esta seguro que desea eliminar el articulo {{ articulo }}?</p>  
<dl class="dl-horizontal">  
  
    <dt>Codigo: </dt>  
    <dd>{{ articulo.codigoArticulo }}</dd>  
  
    <dt>Nombre: </dt>  
    <dd>{{ articulo.nombreArticulo }}</dd>  
  
    <dt>Descripcion: </dt>  
    <dd>{{ articulo.descripcion }}</dd>  
  
    <dt>Estado: </dt>  
    {% if articulo.prestado == False %}  
        <dd>Disponible</dd>  
    {% endif %}  
    {% if articulo.prestado == True %}  
        <dd>Prestado</dd>  
    {% endif %}  
  
</dl>  
<form action="{% url 'articulo-delete' pk=articulo.codigoArticulo  
%}" method="POST">  
    {% csrf token %}  
    <input type="submit" class="btn btn-default" value="Si, Eliminar."  
/>  
    <a href="{% url 'articulo-list' %}" class="btn btn-default">No,  
cancelar.</a>  
</form>  
{% endblock %}
```

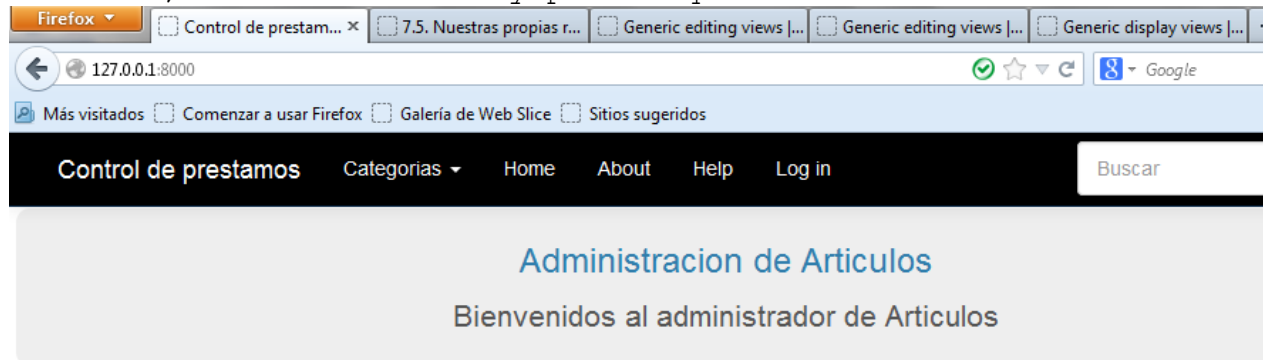
abre el archivo articulo_list.html y agrega las líneas de código

```
<th>Estado</th>
```

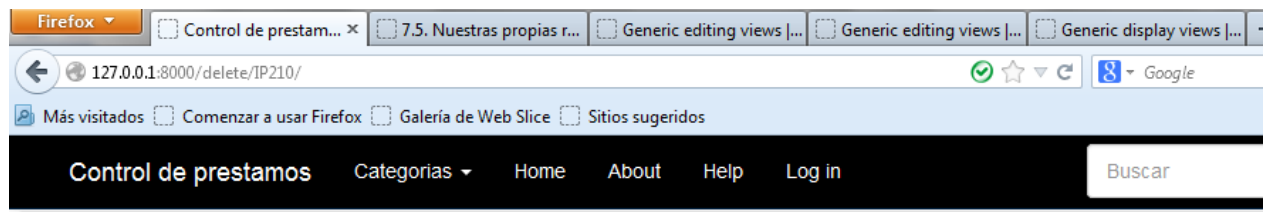
```
...
```

```
<td><a href="{ { articulo.get_absolute_url } }" class="btn btn-
default btn-xs"><span class="glyphicon glyphicon-th-
large"></span></a>
<a href="{ % url 'articulo-delete' pk=articulo.codigoArticulo % }"
class="btn btn-default btn-xs"><span class="glyphicon glyphicon-
trash"></span></a></td>
```

Actualiza el navegador que y observa que esta el icono del basurero, da click en uno y prueba que elimina



+ Nuevo					Categorias
codigo	articulo	Estado			Prestamo App
RG1201	Regleta	Prestado			Articulo
ET1202	Extension	Disponible			Carrera
IP210	Impresora	Disponible			Estudiante



Eliminar Articulo

Esta seguro que desea eliminar el articulo Impresora?

Codigo: IP210
Nombre: Impresora
Descripcion: Se usa para imprimir
Estado: Disponible

Si, Eliminar.

No, cancelar.

Categorias

Prestamo App

Articulo

Carrera

Estudiante

Prestamo

Otros

Editar un articulo

Lo que haremos es reutilizar nuestro formulario para crear edit_articulo.html, por tanto vamos a modificar un poco las vistas agregándoles un método donde la url a la que se tiene que dirigir se la mandaremos desde las vistas

Agrega el siguiente método a la vista CreateArticuloView

```
class CreateArticuloView(CreateView):
    model = Articulo
    template_name = 'edit_articulo.html'
    form_class = ArticuloForm

    def get_context_data(self, **kwargs):
        context=super(CreateArticuloView,self).get_context_data(**kwargs)
        context['action']=reverse('articulo-new')
        return context

    def get_success_url(self):
        return reverse('articulo-list')
```

Crea la vista UpdateArticuloView

```
from django.views.generic import UpdateView

class UpdateArticuloView(UpdateView):
    model = Articulo
    template_name = 'edit_articulo.html'
    form_class = ArticuloForm
    def get_context_data(self, **kwargs):
        context=super(UpdateArticuloView,self).get_context_data(**kwargs)
        context['action']=reverse('articulo-
edit',kwargs={'pk':self.get_object().codigoArticulo})
        return context

    def get_success_url(self):
        return reverse('articulo-list')
```

registra la url

```
from prestamoapp.views import DeleteArticulo, UpdateArticuloView

url(r'^update/(?P<pk>\w+)/$', UpdateArticuloView.as_view(),
    name='articulo-edit',),
```


agrega las siguientes líneas de código al archivo **edit_articulo.html**

```
{% block content %}

{% if articulo.codigoArticulo %}
<h3>Editar articulo</h3>
{% else %}
<h3>Nuevo articulo</h3>
{% endif %}

<hr/>
<form action="{{action}}" class="form-horizontal"
method="post">
```

Agrega la siguientes línea de código al archivo **articulo_list.html**






```
<td>
    <a href="{{ articulo.get_absolute_url }}" class="btn btn-
default btn-xs"><span class="glyphicon glyphicon-th-
large"></span></a>
<a href="{% url 'articulo-delete' pk=articulo.codigoArticulo %}"
class="btn btn-default btn-xs"><span class="glyphicon glyphicon-
trash"></span></a>
<a href="{% url 'articulo-edit' pk=articulo.codigoArticulo %}"
class="btn btn-default btn-xs"><span class="glyphicon glyphicon-
pencil"></span></a>
</td>
```

Prueba dando click en Nuevo

Nuevo articulo

Codigo	<input type="text" value="TB1201"/>
Nombre	<input type="text" value="Mesa"/>
Descripcion	<input type="text" value="Es usado para estudiar"/>
Prestado	<input type="checkbox"/>
<div>Guardar Volver a la lista</div>	







+ Nuevo

codigo	articulo	Estado	
RG1201	Regleta	Prestado	  
ET1202	Extension	Disponible	  
IP210	Impresora	Disponible	  
TB1201	Mesa	Disponible	  

Prueba editando el nombre de la mesa

Editar articulo

Codigo	<input type="text" value="TB1201"/>
Nombre	<input type="text" value="Mesa Circular"/>
Descripcion	<input type="text" value="Es usado para estudiar"/>
Prestado	<input type="checkbox"/>
<div>Guardar Volver a la lista</div>	

IP210	Impresora	Disponible	  
TB1201	Mesa Circular	Disponible	  

Por ultimo agregaremos funcionalidad a los menus, abre el archivo shared.html y agrega lo siguiente donde se encuentra articulo:

```
<li><a href="/">Articulo</a></li>
<a href="/" class="list-group-item">Articulo</a>
```

Listar Prestamos

Abre el archivo **views.py**

```
from django.shortcuts import render

from django.http import HttpResponse
from prestamoapp.models import Artículo
from django.views.generic import ListView
from prestamoapp.models import Artículo, Prestamo

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()})

def probando(request):
    return render(request, "prestamo.html")

class ListPrestamoView(ListView):
    model = Prestamo
    template_name = "prestamo_list.html"
```

abre **urls.py** para registrarla

```
from prestamoapp.views import
misArticulos,probando,ListPrestamoView

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),
    url(r'^prueba/$', probando),
    url(r'^$', ListPrestamoView.as_view(), name="prestamo-list"),
)
```

crea el archivo **prestamo_list.html**

```

{% extends "shared.html" %}

{% block content%}
    <div class="table-responsive">
        <table class="table table-hover">
            <tr class="success">
                <th>carnet</th>
                <th>codigo</th>
                <th>articulo</th>
                <th>Fecha prestamo</th>
            </tr>
            {% for prestamo in object_list %}
                <tr>
                    <td>{{ prestamo.estudiante.carnet}}</td>
                    <td>{{ prestamo.articulo.codigoArticulo}}</td>
                    <td>{{ prestamo.articulo.nombreArticulo}}</td>
                    <td>{{ prestamo.fechaPrestamo}}</td>
                </tr>
            {% endfor %}
        </table>
    </div>
{% endblock %}

```

Abre el navegador en la siguiente dirección <http://127.0.0.1:8000/prestamos/>

Firefox Control de prestamos

127.0.0.1:8000/prestamos/

Más visitados Comenzar a usar Firefox Galería de Web Slice Sitios sugeridos

Control de prestamos Categorías Home About Help Log in

Buscar

Administración de Prestamos

Bienvenidos al asistente para la administración de prestamos

carnet	codigo	articulo	Fecha prestamo
VP12004	RG1201	Regleta	April 23, 2016
VP12004	ET1202	Extension	May 1, 2016

Categorías

- Prestamo App
- Articulo
- Carrera
- Estudiante
- Prestamo
- Otros

Artículos recientes

Gestion de usuarios
Capacitate para la gestion de usuario

Vasquez Perez Francisco

Home Capacitaciones Contactos

Abre el archivo shared.html y busca la opción donde dice préstamo y modifícalo como el siguiente

```
<li><a href="/prestamos">Préstamo</a></li>
<a href="/prestamos" class="list-group-item">Préstamo</a>
```

Prueba en los link donde aparece préstamo y versa que funciona.