# Django 4.2.2 Rest Framework 3.14.0 Datatables 1.13.4 Bootstrap 5.0.2 Jquery 3.7.0 ajax

Nombre del proyecto: **api\_vv**  
Nombre de la aplicación: **libros**

Página oficial de DRF: [Django REST framework](https://www.django-rest-framework.org/#quickstart)

#### 

#### Objetivos: desarrollar una API con Django Rest Framework que tenga:

Endpoint que nos devuelva los detalles de un libro concreto.

Endpoint para añadir un libro.

Endpoint para actualizar un libro existente.

El endpoint que borre un libro existente.

Crear test para todos los Endpoints que nos aseguren su funcionamiento en cualquier circunstancia.

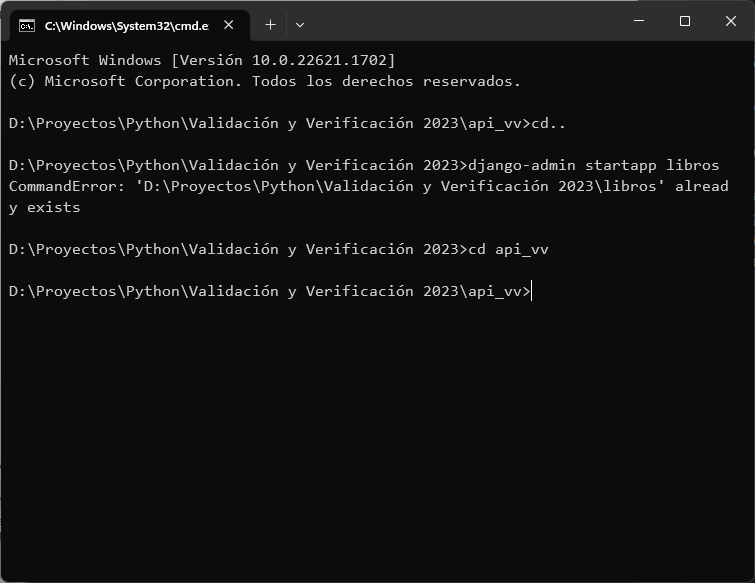
#### Actualizar pip para la instalación principal

Esto actualiza pip de la instalación principal, el que no está dentro de los entornos virtuales.

cmd: **python -m pip install –upgrade pip**

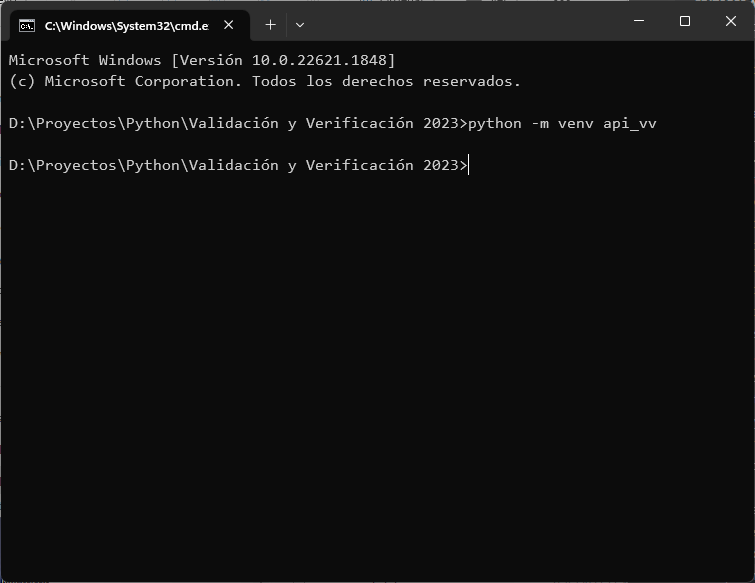
#### Iniciar un nuevo proyecto en django Abrir el cmd en la carpeta en donde se quiere crear el proyecto. Por ejemplo en la carpeta “ Validación y Verificación 2023”. cmd: Ejecutar: **django-admin startproject api\_vv**

#### Iniciar una nueva aplicación libro en el proyecto api\_vv

cmd: En la misma ventana cmd donde ejecutó el comando anterior.  
**cd api\_vv  
**

**django-admin startapp libros**

#### Crear un entorno virtual “api\_vv” [Entornos virtuales en Python](https://docs.google.com/document/u/0/d/1-58vsxhM6GF2WJTe1OiUFAk3Qx6_CoNOuRtwXB-xlmg/edit) cmd: En la misma ventana de cmd regresar a la carpeta que contiene api\_vv (“ Validación y Verificación 2023”). cd ..

cmd: Ejecuta el comando para crear el entorno virtual.  
**python -m venv api\_vv  
**

#### Crear archivo de requerimientos “requirements.txt” [Cómo instalar paquetes Python con requirements.txt](https://rukbottoland.com/blog/como-instalar-paquetes-python-con-requirementstxt/) [Crea requirements.txt en Python | Delft Stack](https://www.delftstack.com/es/howto/python/python-create-requirements.txt/)

vcode: Abrir la carpeta api\_vv.  
Crear un archivo de texto **requirements.txt** con el contenido abajo indicado.

Django # Python Web Framework

django-filter # Filtering support.

django-model-utils # Utilities for Django models.

djangorestframework # API framework

Faker # Generador de información aleatoria para los test.

Markdown # Markdown support for the browsable API.

pytest-drf # Pytest plugin for Django REST Framework.

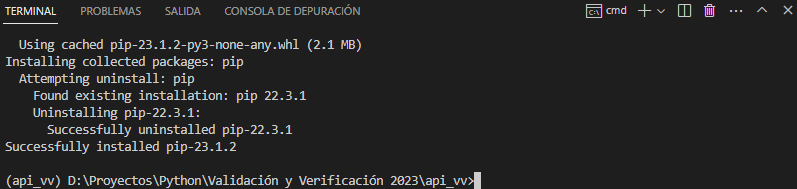
pytest-django # Integración con Django.

python-dateutil # Utilidades para trabajar con fechas.

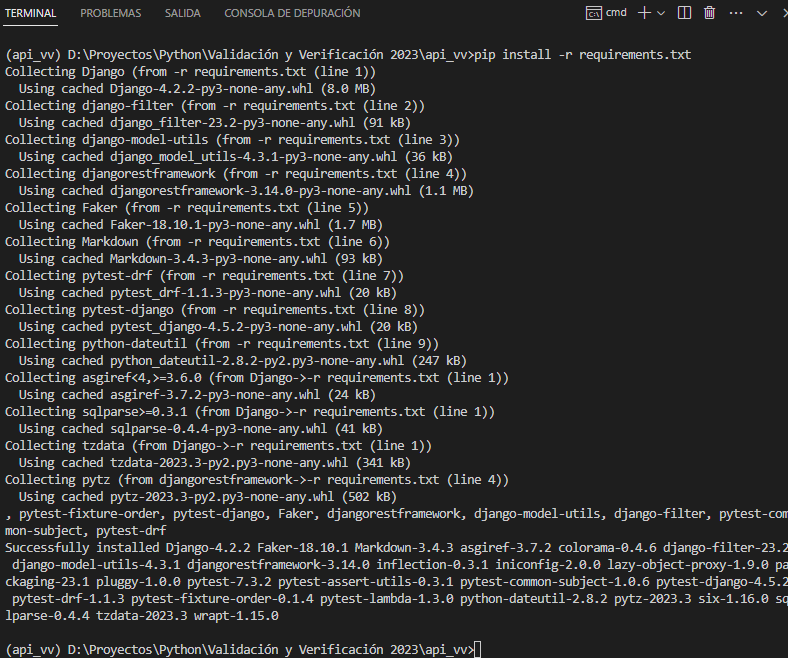
#### 

#### Activar el entorno virtual api\_vv vcode: Abrir un nuevo terminal en Visual Code y esto ya se abre con el entorno activado.

#### Actualizar pip para el entorno virtual

En el terminal de Visual Code.  
cmd: **python -m pip install –upgrade pip  
**

#### Instalar los requerimientos

cmd: En el cmd abierto de Visual Code. **pip install -r requirements.txt**(-r es para que los instale a partir de un archivo de requerimientos)  


#### Settings [Django settings](https://docs.djangoproject.com/en/4.1/topics/settings/) [Settings - Django REST framework](https://www.django-rest-framework.org/api-guide/settings/)

vcode: En el archivo **api\_vv\settings.py**, modificar la lista de aplicaciones añadiendo **rest\_framework** y **libros**:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest\_framework',

'libros',

]

Añadir al final.

REST\_FRAMEWORK = {

# Use Django's standard `django.contrib.auth` permissions,

# or allow read-only access for unauthenticated users.

'DEFAULT\_PERMISSION\_CLASSES': [

#'rest\_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',

#'rest\_framework.permissions.AllowAny',

],

'DEFAULT\_PAGINATION\_CLASS': 'rest\_framework.pagination.PageNumberPagination',

'PAGE\_SIZE': 10

}

DATE\_FORMAT = '%d/%m/%Y'

DATETIME\_FORMAT = '%d/%m/%Y %H:%M'

#### Models [Models](https://docs.djangoproject.com/es/4.1/topics/db/models/)

Reemplazar **libros\models.py** con el código de abajo para crear los modelos de nuestra aplicación, va en este orden para cuando hagamos la migración nos cree las tablas y relaciones en la base de datos de la aplicación.

from django.db import models

from django.db.models import Q

from model\_utils import Choices

ORDER\_COLUMN\_CHOICES = Choices(

('0', 'id'),

('1', 'titulo'),

('2', 'genero'),

('3', 'created\_at'),

('4', 'updated\_at'),

)

class libros(models.Model):

titulo = models.CharField(max\_length=255)

genero = models.CharField(max\_length=255)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class Meta:

db\_table = 'libros'

ordering = ("titulo",)

verbose\_name = "libro"

verbose\_name\_plural = "libros"

def \_\_str\_\_(self):

return self.titulo

def query\_libros\_by\_args(\*\*kwargs):

draw = int(kwargs.get('draw', None)[0])

length = int(kwargs.get('length', None)[0])

start = int(kwargs.get('start', None)[0])

search\_value = kwargs.get('search[value]', None)[0]

order\_column = kwargs.get('order[0][column]', None)[0]

order = kwargs.get('order[0][dir]', None)[0]

order\_column = ORDER\_COLUMN\_CHOICES[order\_column]

# django orm '-' -> desc

if order == 'desc':

order\_column = '-' + order\_column

queryset = libros.objects.all()

total = queryset.count()

if search\_value:

queryset = queryset.filter(Q(id\_\_icontains=search\_value) |

Q(titulo\_\_icontains=search\_value) |

Q(genero\_\_icontains=search\_value) |

Q(created\_at\_\_icontains=search\_value) |

Q(updated\_at\_\_icontains=search\_value))

count = queryset.count()

if length == -1:

queryset = queryset.order\_by(order\_column)

else:

queryset = queryset.order\_by(order\_column)[start: start + length]

return {

'items': queryset,

'count': count,

'total': total,

'draw': draw

}

#### 

#### Administrador (admin) de Django: [The Django admin site](https://docs.djangoproject.com/en/4.1/ref/contrib/admin/) [Administrador de Django · HonKit](https://tutorial.djangogirls.org/es/django_admin/)

Añadir el siguiente código en **libros\admin.py**

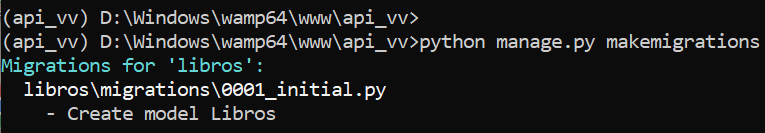
from django.contrib import admin

# Register your models here.

from .models import libros

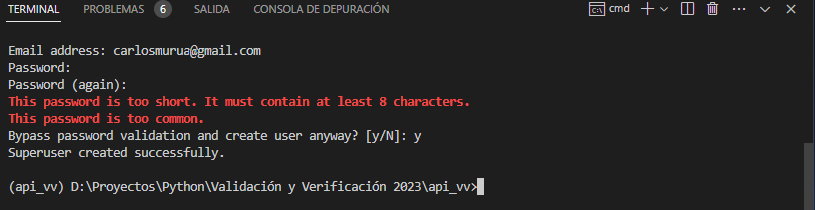
admin.site.register(libros)

#### Migrar la base de datos (esta operación migra la base de usuarios de sqlite3). [Migrations | Django documentation](https://docs.djangoproject.com/en/4.1/topics/migrations/) cmd: En el terminal de Visual Code. **python manage.py makemigrations** Busca cambios en los modelos y crea el archivo de migraciones.

****

#### **python manage.py migrate** Ejecuta la migración. [Migrations | Django documentation](https://docs.djangoproject.com/en/4.1/topics/migrations/)

#### Crear un usuario administrador

En el terminal de Visual Code.  
cdm: Crear un usuario inicial llamado **admin** con una contraseña de **pass123**.  
Nos autenticamos como ese usuario más adelante en nuestro ejemplo.  
**python manage.py createsuperuser  
**

#### Serializers (model serializer) [Serializers - Django REST framework](https://www.django-rest-framework.org/api-guide/serializers/#modelserializer) Primero vamos a definir algunos serializadores. Vamos a crear un nuevo módulo llamado libros/serializers.py que usaremos para nuestras representaciones de datos.

**from django.conf import settings**

**from rest\_framework import serializers**

**from .models import libros**

**class librosSerializer(serializers.ModelSerializer):**

**# Si su <field\_name> se declara en su serializador con el parámetro required=False**

**# Entonces este paso de validación no tendrá lugar si el campo no está incluido.**

**created\_at = serializers.DateTimeField(format=settings.DATETIME\_FORMAT, required=False, null=True, allow\_null=True)**

**updated\_at = serializers.DateTimeField(format=settings.DATETIME\_FORMAT, required=False, null=True, allow\_null=True)**

**class Meta:**

**model = libros**

**fields = "\_\_all\_\_"**

#### 

#### Views (vistas) [Viewsets - Django REST framework](https://www.django-rest-framework.org/api-guide/viewsets/#modelviewset) Abrir libros/views.py y reemplazar con el siguiente código.

from django.shortcuts import render

from rest\_framework.response import Response

from rest\_framework import viewsets, status

from .serializers import librosSerializer

from .models import libros, query\_libros\_by\_args

def index(request):

return render(request, 'index.html')

class librosViewSet(viewsets.ModelViewSet):

queryset = libros.objects.all()

serializer\_class = librosSerializer

def list(self, request, \*\*kwargs):

try:

libros = query\_libros\_by\_args(\*\*request.query\_params)

serializer = librosSerializer(libros['items'], many=True)

result = dict()

result['data'] = serializer.data

result['draw'] = libros['draw']

result['recordsTotal'] = libros['total']

result['recordsFiltered'] = libros['count']

return Response(result, status=status.HTTP\_200\_OK, template\_name=None, content\_type=None)

except Exception as e:

return Response(e, status=status.HTTP\_404\_NOT\_FOUND, template\_name=None, content\_type=None)

#### 

#### URLs (router) [URL dispatcher | Django documentation](https://docs.djangoproject.com/en/4.1/topics/http/urls/) [Routers - Django REST framework](https://www.django-rest-framework.org/api-guide/routers/#routers) En api\_vv/urls.py reemplazar con el siguiente código.

"""api\_vv URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/4.1/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('', include('libros.urls')),

]

1. En **api\_vv/libros/urls.py** crear el archivo y agregar el siguiente código.

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from libros.views import index, librosViewSet

router = DefaultRouter()

router.register(r'libros', librosViewSet, basename='libros')

urlpatterns = [

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

path('api/', include(router.urls)),

]

#### 

#### En la carpeta libros crear las carpetas:

Recomiendo realizar este procedimiento desde el Explorador de Windows y no desde Visual Code.

Crear la carpeta dentro de libros: **static**

Dentro de static crear las carpetas: **img** y **js**

#### En static/js/ crear index.js [jQuery.ajax()](https://api.jquery.com/jquery.ajax/) [7.3. Métodos Ajax de jQuery](https://uniwebsidad.com/libros/fundamentos-jquery/capitulo-7/metodos-ajax-de-jquery)

//Definición de las funciones que se ejecutan al cargar la página.

let table = $('#tablaLibros').DataTable({

"lengthMenu": [[5, 15, 20, -1], [5, 15, 20, "Todos"]],

"processing": true,

"serverSide": true,

"ajax": {

"url": "/api/libros/",

"type": "GET",

"dataType": "json"

},

"columns": [

{ "data": "id" },

{ "data": "titulo" },

{ "data": "genero" },

{ "data": "created\_at" },

{ "data": "updated\_at" },

{ "data": null,

"defaultContent": '<button type="button" class="btn btn-warning" style="--bs-btn-padding-y: .25rem; --bs-btn-padding-x: .5rem; --bs-btn-font-size: .75rem;">Modificar</button>' + '&nbsp;&nbsp' +

'<button type="button" class="btn btn-danger" style="--bs-btn-padding-y: .25rem; --bs-btn-padding-x: .5rem; --bs-btn-font-size: .75rem;">Eliminar</button>'

}

],

"language": {

url: "//cdn.datatables.net/plug-ins/1.10.19/i18n/Spanish.json"

}

});

let id = 0;

$('#tablaLibros tbody').on('click', 'button', function () {

let data = table.row($(this).parents('tr')).data();

let class\_name = $(this).attr('class');

if (class\_name == 'btn btn-warning') {

// Botón Editar

$('#title').val(data['titulo']);

$('#genre').val(data['genero']);

$('#type').val('edit');

$('#modal\_title').text('MODIFICAR');

$("#myModal").modal('show');

} else {

// Botón Eliminar

$('#modal\_title').text('ELIMINAR');

$("#confirm").modal('show');

}

id = data['id'];

});

$('form').on('submit', function (e) {

e.preventDefault();

//let $this = $(this);

let datos = {

titulo: $('#title').val(),

genero: $('#genre').val()

};

let type = $('#type').val();

let method = '';

let url = '/api/libros/';

if (type == 'new') {

// nuevo

method = 'POST';

} else {

// editar

url = url + id + '/';

method = 'PUT';

}

$.ajax({

url: url,

method: method,

data: datos,

dataType: 'json'

})

.done(function() {

location.reload();

})

.fail(function(datos, textStatus, jqXHR) {

location.reload();

});

});

$('#confirm').on('click', '#delete', function (e) {

$.ajax({

url: '/api/libros/' + id + '/',

method: 'DELETE',

dataType: 'json'

}).done(function () {

location.reload();

}).fail(function (datos, textStatus, jqXHR) {

location.reload();

});

});

$('#cancel').on('click', function (e) {

location.reload();

});

$('#new').on('click', function (e) {

$('#title').val('');

$('#genre').val('');

$('#type').val('new');

$('#modal\_title').text('NUEVO');

$("#myModal").modal('show');

});

#### 

#### En la carpeta libros crear la carpeta:

**templates**

#### En **templates crear base.html** [Templates - Al ser un marco web, Django necesita una forma conveniente de generar HTML dinámicamente - Español](https://runebook.dev/es/docs/django/topics/templates)cdn css Bootstrap: [Get started with Bootstrap · Bootstrap v5.2](https://getbootstrap.com/docs/5.2/getting-started/introduction/#cdn-links) cdn css: DataTables: [DataTables](https://datatables.net/)

{% load static %}

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Biblioteca - Libros</title>

<meta name="description" content="Libros de la biblioteca por géneros.">

<link rel="icon" href="{% static 'img/favicon.ico' %}">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.2.0/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdn.datatables.net/1.12.1/css/dataTables.bootstrap5.min.css">

</head>

<body>

<div class="container" style="font-size: .9em;">

{% block content %}{% endblock content %}

</div>

{% block extra\_js %}

<script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js" integrity="sha384-IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk" crossorigin="anonymous"></script>

<script src="https://cdn.datatables.net/1.12.1/js/jquery.dataTables.min.js"></script>

<script src="https://cdn.datatables.net/1.12.1/js/dataTables.bootstrap5.min.js"></script>

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

{% endblock extra\_js %}

</body>

<footer class="footer" style="margin-top: 25px;">

<div class="container">

<p class="text-muted text-center">

Copyright © 2023 - Alumnos de tercer año Desarrollo de Software

</p>

</div>

</footer>

</html>

#### En **templates crear index.html**

{% extends "base.html" %}

{% load static %}

{% block content %}

<div align="center">

<img src="{% static 'img/logo.png' %}" align="left" width="20%" height="20%">

<br>

<br>

<h2 align="center">DataTables - Server side processing</h2>

<br clear="all">

</div>

<div class="row align-items-center">

<!-- Nombre de la tabla -->

<div class="col-md-4" >

<h2>Libros</h2>

</div>

<!-- Botón nuevo -->

<div class="col-md-4 ms-auto" align="center">

<button type="button" data-dismiss="modal" class="btn btn-primary" id="new" style="--bs-btn-padding-y: .25rem; --bs-btn-padding-x: .5rem; --bs-btn-font-size: .75rem;">Nuevo libro</button>

</div>

<hr>

</div>

<!-- tablaLibros -->

<table id="tablaLibros" class="table table-sm table-striped row-bordered" style="width:100%">

<thead>

<tr>

<th>Id</th>

<th>Título</th>

<th>Género</th>

<th>Created\_at</th>

<th>Updated\_at</th>

<th>Action</th>

</tr>

</thead>

<tfoot>

<tr>

</tr>

</tfoot>

</table>

<!-- Nuevo y editar modal -->

<div class="modal fade" id="myModal" role="dialog">

<div class="modal-dialog">

<!-- Contenido modal -->

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="modal\_title"></h4>

</div>

<div class="modal-body">

<form role="form">

<div class="form-group">

<label for="title"><span class="glyphicon glyphicon-pencil"></span> Título</label>

<input type="text" class="form-control" id="title" name="title"

placeholder="Ingrese título" required="true">

</div>

<div class="form-group">

<label for="genre"><span class="glyphicon glyphicon-file"></span> Género</label>

<input type="text" class="form-control" id="genre" name="genre"

placeholder="Ingrese género" required="true">

</div>

<input type="hidden" id="type" name="type" value="">

<br>

<button type="submit" class="btn btn-success btn-block">

<span class="glyphicon glyphicon-ok"></span> OK

</button>

</form>

</div>

</div>

</div>

</div>

<!-- Eliminar modal -->

<div class="modal fade" id="confirm" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title">Estás seguro?</h4>

</div>

<br>

<div class="modal-body">

<button type="button" data-dismiss="modal" class="btn btn-danger" id="delete">Eliminar</button>

<button type="button" data-dismiss="modal" class="btn btn-secondary" id="cancel">Cancel</button>

</div>

</div>

</div>

</div>

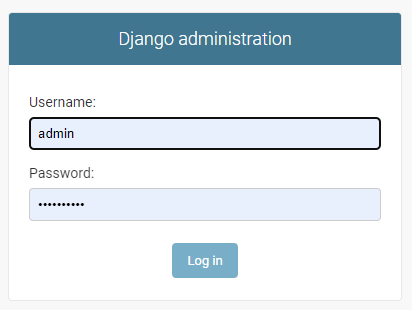
{% endblock content %}

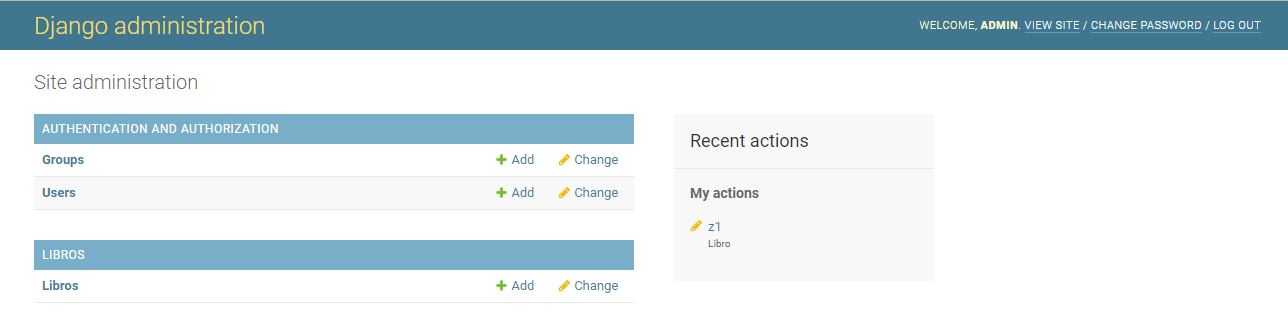
#### 

#### En static\img descargar los ico, png y psd: [Carpeta de imágenes para descargar y añadir al proyecto](https://drive.google.com/drive/folders/1fn8NA1n8ticdJorZvbyBaFsp-2JCC7Io?usp=sharing) Es posible que Drive lo baje zipeado, en ese caso hay que descomprimirlos en la misma carpeta.

#### Ejecutar el servidor web de Django. En el terminal abierto de Visual Code python manage.py runserver

#### Probar la API con el servidor de desarrollo django

En el navegador: **localhost:8000/admin/**Usuario: **admin**  
Password: **123  
  
**  
Deberás ver:



#### 

#### Ahora estamos listos para probar la API que hemos construido (para que funcione no debe estar logueado). [http://localhost:8000/](http://localhost:8000/api-alumnos/CarrerasList/)