

# Referencia Rápida de Django

Modelos, vistas, plantillas, ORM, formularios, admin, auth

## Configuración del proyecto

### Crear proyecto y app

```
pip install django
django-admin startproject mysite
cd mysite
python manage.py startapp blog
```

### Comandos comunes

<b>runserver</b>	Iniciar servidor de desarrollo en el puerto 8000
<b>makemigrations</b>	Generar archivos de migración desde cambios en modelos
<b>migrate</b>	Aplicar migraciones a la base de datos
<b>createsuperuser</b>	Crear superusuario de administración
<b>shell</b>	Shell interactivo de Python con Django
<b>test</b>	Ejecutar suite de pruebas

### Estructura del proyecto

<b>manage.py</b>	Punto de entrada de la CLI
<b>settings.py</b>	Configuración del proyecto
<b>urls.py</b>	Configuración de URLs raíz
<b>wsgi.py / asgi.py</b>	Puntos de entrada del servidor
<b>apps/models.py</b>	Modelos de base de datos
<b>apps/views.py</b>	Manejadores de solicitudes

## Modelos

### Definir un modelo

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    body = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
    published = models.BooleanField(default=False)
```

### Tipos de campo

<b>CharField(max_length=N)</b>	Texto corto (max_length requerido)
<b>TextField()</b>	Texto largo (sin límite)
<b>IntegerField()</b>	Valor entero
<b>FloatField()</b>	Número de punto flotante
<b>BooleanField()</b>	Verdadero / Falso
<b>DateTimeField()</b>	Fecha y hora
<b>EmailField()</b>	Email con validación
<b>FileField(upload_to='')</b>	Carga de archivos

### Relaciones

```
author = models.ForeignKey(
    User, on_delete=models.CASCADE
)
tags = models.ManyToManyField(Tag, blank=True)
profile = models.OneToOneField(User, on_delete=models.CASCADE)
```

### Meta y métodos

```
class Meta:
    ordering = ['-created']
    verbose_name_plural = 'posts'

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

## Vistas

### Vista basada en función

```
from django.shortcuts import render, get_object_or_404

def post_list(request):
    posts = Post.objects.filter(published=True)
    return render(request, 'blog/list.html', {'posts': posts})
```

### Vistas basadas en clases

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

class PostListView(ListView):
    model = Post
    template_name = 'blog/list.html'
    context_object_name = 'posts'
    paginate_by = 10
```

### CBVs comunes

<b>ListView</b>	Mostrar lista de objetos
<b>DetailView</b>	Mostrar un solo objeto
<b>CreateView</b>	Formulario para crear objeto
<b>UpdateView</b>	Formulario para editar objeto
<b>DeleteView</b>	Confirmar y eliminar objeto
<b>TemplateView</b>	Renderizar plantilla (sin modelo)

### Respuesta JSON

```
from django.http import JsonResponse

def api_posts(request):
    data = list(Post.objects.values('id', 'title'))
    return JsonResponse(data, safe=False)
```

## Plantillas

### Sintaxis de plantilla

```
{{ variable }}
{{ post.title|truncatewords:30 }}
{% if user.is_authenticated %}
    <p>Welcome, {{ user.username }}!</p>
{% endif %}
```

### Bucles y condiciones

```
{% for post in posts %}
    <h2>{{ post.title }}</h2>
    {% if forloop.last %}<hr>{% endif %}
{% empty %}
    <p>No posts yet.</p>
{% endfor %}
```

### Herencia de plantillas

```
{# base.html #}
<html>
<body>{% block content %}{% endblock %}</body>
</html>

{# child.html #}
{% extends "base.html" %}
{% block content %}<h1>Hello</h1>{% endblock %}
```

### Filtros comunes

<b> date:"Y-m-d"</b>	Formatear fecha
<b> default:"N/A"</b>	Valor por defecto para vacíos
<b> length</b>	Contar elementos en lista
<b> truncatewords:N</b>	Limitar a N palabras
<b> safe</b>	Marcar como HTML seguro (sin escapado)
<b> slugify</b>	Cadena en minúsculas apta para URL

## URLs

### Patrones de URL

```
from django.urls import path, include

urlpatterns = [
    path('', views.index, name='index'),
    path('post/<int:pk>/', views.detail, name='detail'),
    path('blog/', include('blog.urls')),
]
```

### Conversores de path

<b>&lt;int:pk&gt;</b>	Entero (ej. 42)
<b>&lt;str:slug&gt;</b>	Cadena sin barras
<b>&lt;slug:slug&gt;</b>	Slug (letras, números, guiones)
<b>&lt;uuid:id&gt;</b>	Formato UUID
<b>&lt;path:rest&gt;</b>	Ruta completa incluyendo barras

### URLs inversas

```
from django.urls import reverse
url = reverse('detail', kwargs={'pk': 1})
# En plantillas: {% url 'detail' pk=post.pk %}
```

## Formularios

### ModelForm

```
from django import forms

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'body', 'published']
```

### Procesar formulario en vista

```
def create_post(request):
    form = PostForm(request.POST or None)
    if form.is_valid():
        post = form.save(commit=False)
        post.author = request.user
        post.save()
        return redirect('detail', pk=post.pk)
    return render(request, 'blog/form.html', {'form': form})
```

### Formulario en plantilla

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
</form>
```

### Validación

```
def clean_title(self):
    title = self.cleaned_data['title']
    if len(title) < 5:
        raise forms.ValidationError("Title too short.")
    return title
```

## Admin

### Registrar modelo

```
from django.contrib import admin
from .models import Post

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
    list_display = ['title', 'author', 'created', 'published']
    list_filter = ['published', 'created']
    search_fields = ['title', 'body']
```

# Referencia Rápida de Django

## Opciones de admin

<b>list_display</b>	Columnas en la vista de lista
<b>list_filter</b>	Opciones de filtro en la barra lateral
<b>search_fields</b>	Campos buscables
<b>prepopulated_fields</b>	Autocompletar (ej. slug desde título)
<b>readonly_fields</b>	No editables en el admin
<b>ordering</b>	Orden de clasificación por defecto

## Consultas ORM

### Consultas básicas

Post.objects.all()	# todos los registros
Post.objects.get(pk=1)	# único (lanza error si falta)
Post.objects.filter(published=True)	# queryset
Post.objects.exclude(draft=True)	# excluir coincidencias
Post.objects.count()	# total de registros

### Lookups de campo

<b>field__exact</b>	Coincidencia exacta (por defecto)
<b>field__icontains</b>	Contiene sin distinguir mayúsculas
<b>field__gt / __lt</b>	Mayor / menor que
<b>field__gte / __lte</b>	Mayor/menor o igual que
<b>field__in=[1,2,3]</b>	Valor en la lista
<b>field__isnull=True</b>	Es NULL
<b>field__startswith</b>	Empieza con cadena
<b>field__range=(a,b)</b>	Entre a y b inclusive

## Encadenamiento y agregación

```
from django.db.models import Q, Count, Avg

Post.objects.filter(
    Q(title__icontains='django') | Q(body__icontains='django')
).order_by('-created')[:10]

Post.objects.aggregate(
    avg_views=Avg('views'))
```

## Crear, actualizar, eliminar

```
post = Post.objects.create(title='New', body='...')
post.title = 'Updated'
post.save()
Post.objects.filter(draft=True).update(published=False)
post.delete()
```

## Autenticación

### Login / Logout

```
from django.contrib.auth import authenticate, login, logout

user = authenticate(request, username='admin', password='pw')
if user is not None:
    login(request, user)
```

### Proteger vistas

```
from django.contrib.auth.decorators import login_required

@login_required
def dashboard(request):
    return render(request, 'dashboard.html')
```

### URLs de auth

```
# urls.py
path('accounts/', include('django.contrib.auth.urls'))
# Provee: login, logout, password_change, password_reset
```

## Auth en plantilla

```
{% if user.is_authenticated %}
<p>Hi, {{ user.username }}</p>
<a href="{% url 'logout' %}">Logout</a>
{% else %}
<a href="{% url 'login' %}">Login</a>
{% endif %}
```

## Configuración

### Ajustes clave

<b>DEBUG</b>	<b>True</b> para desarrollo, <b>False</b> para producción
<b>ALLOWED_HOSTS</b>	Lista de nombres de host válidos
<b>SECRET_KEY</b>	Clave de firma criptográfica (mantener en secreto)
<b>DATABASES</b>	Motor, nombre, host y credenciales de BD
<b>INSTALLED_APPS</b>	Lista de apps registradas
<b>STATIC_URL</b>	Prefijo de URL para archivos estáticos
<b>MEDIA_URL / MEDIA_ROOT</b>	Rutas para archivos subidos por usuarios

### Configuración de base de datos

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'mydb',
        'USER': 'dbuser',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

### Archivos estáticos

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / 'static']
# En plantillas: {% load static %}
# <link href="{% static 'css/style.css' %}">
```