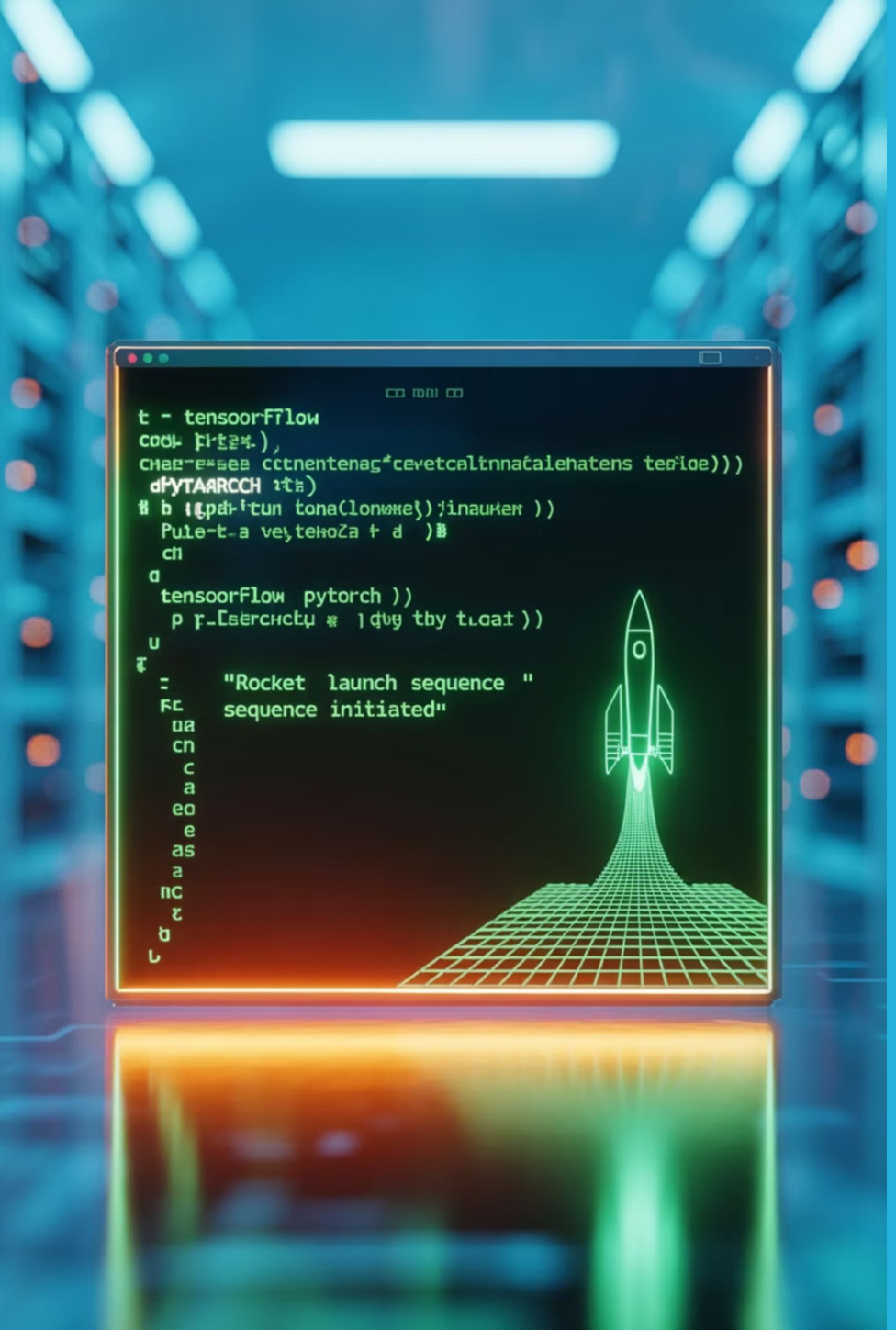
Introducción a Django y sus herramientas administrativas

Bienvenidos al módulo sobre la creación y configuración de proyectos web con Django. Aprenderemos a instalar, configurar y manejar un proyecto Django desde cero, utilizando sus herramientas administrativas.

Django es uno de los frameworks web más poderosos de Python, diseñado para facilitar el desarrollo rápido y limpio de aplicaciones web robustas y escalables.

¿Qué es pip?

Pip es el gestor de paquetes de Python. Permite instalar, actualizar y manejar librerías externas, incluyendo Django. Es fundamental para cualquier desarrollador Python.
Con pip puedes acceder a miles de paquetes disponibles en el Python Package Index (PyPI), lo que amplía enormemente las capacidades de tus proyectos.



Instalación de pip

En la mayoría de las versiones modernas de Python, pip viene preinstalado. Si no, se puede instalar manualmente con:

```
python -m ensurepip --upgrade
```

Este comando asegura que tengas la versión más reciente de pip disponible para tu instalación de Python.

Importancia del entorno virtual

Aislamiento

Evita conflictos entre dependencias de diferentes proyectos

Control de versiones

Mantiene versiones específicas de paquetes por proyecto

Reproducibilidad

Facilita la replicación del entorno en otros sistemas

Antes de instalar Django, es recomendable usar un entorno virtual. Esto evita conflictos entre dependencias de diferentes proyectos y mantiene tu sistema limpio y organizado.

Crear un entorno virtual

Para crear un entorno virtual usamos:

```
python -m venv mi_entorno
```

Esto genera una carpeta con Python y pip independientes del sistema global. El entorno virtual contendrá su propia instalación de Python y un directorio separado para instalar paquetes.

Activar el entorno virtual

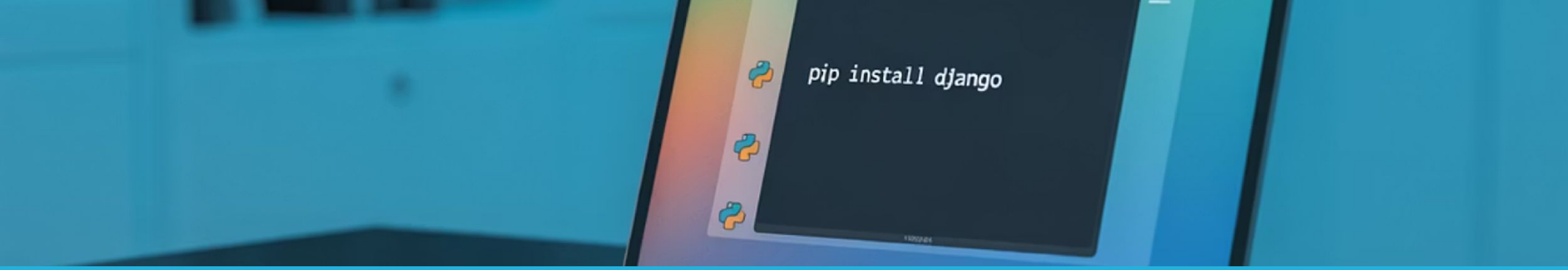
macOS/Linux

```
source mi_entorno/bin/activate
```

Windows

```
mi_entorno\Scripts\activate
```

Verás el nombre del entorno en la terminal, indicando que está activo. Una vez activado, todos los paquetes que instales se guardarán únicamente en este entorno.



Instalar Django en el entorno virtual

Una vez activado el entorno, instalamos Django con:

```
pip install django
```

Esto garantiza que la instalación esté aislada y controlada. Django se instalará junto con todas sus dependencias necesarias dentro del entorno virtual activo.

Verificar la instalación de Django

Para comprobar la instalación:

```
django-admin --version
```

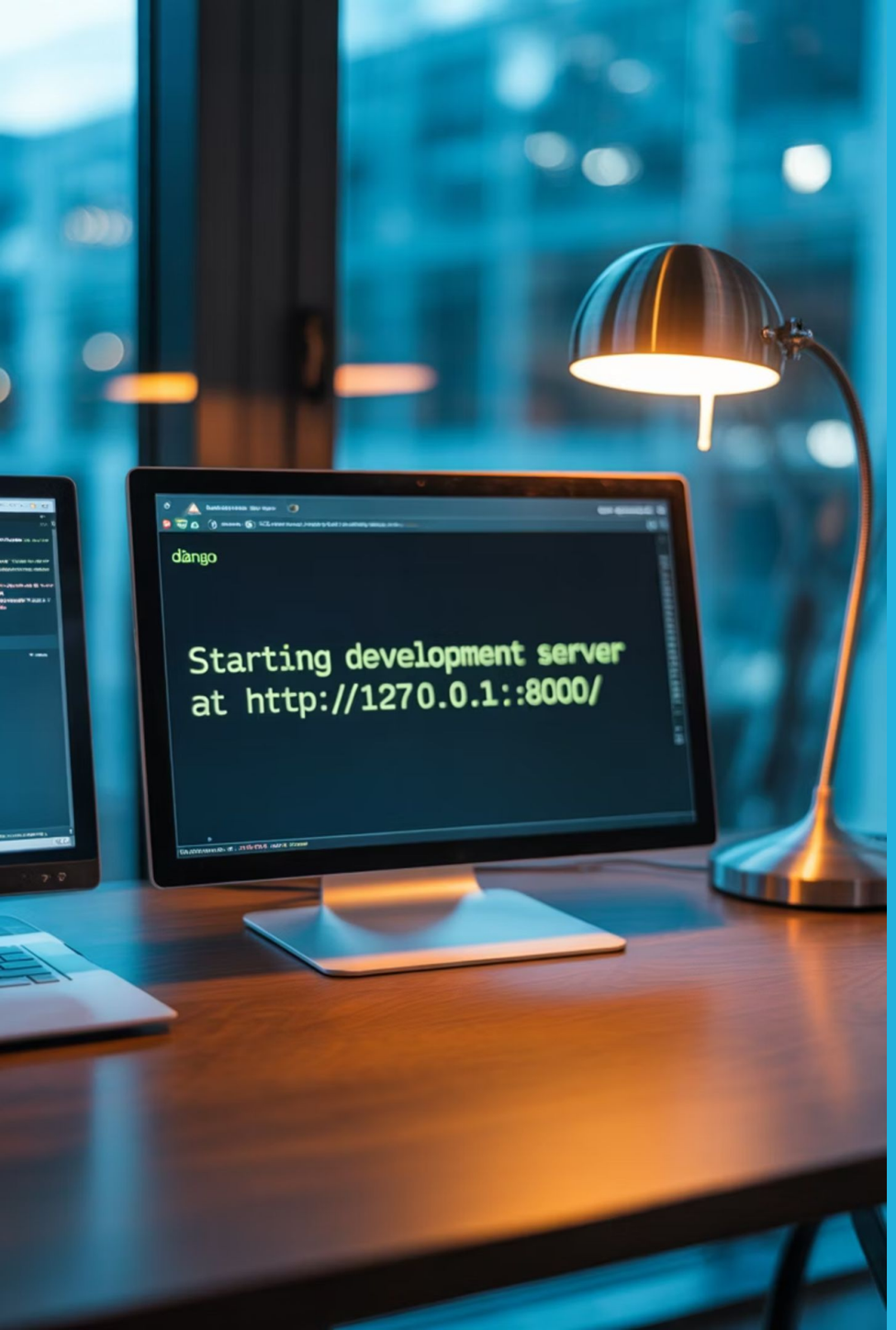
Debería mostrar la versión instalada, por ejemplo: 4.1.2

Crear un proyecto Django

Django facilita la creación de proyectos con el comando:

```
django-admin startproject mi_sitio
```

Esto genera la estructura básica necesaria para comenzar a desarrollar tu aplicación web. El comando crea todos los archivos de configuración y directorios fundamentales.



Navegar al proyecto y ejecutar el servidor

01

Navegar al directorio

```
cd mi_sitio
```

02

Ejecutar el servidor

```
python manage.py runserver
```

Esto inicia el servidor de desarrollo. Podrás ver tu proyecto funcionando en el navegador accediendo a la dirección local proporcionada.

Introducción a manage.py

manage.py es una herramienta clave que permite ejecutar comandos administrativos: runserver, createsuperuser, y más.

Gestión del servidor

Inicia y controla el servidor de desarrollo

Administración de base de datos

Ejecuta migraciones y maneja la estructura de datos

Creación de usuarios

Permite crear superusuarios y gestionar permisos

Diferencia entre django-admin y manage.py

1

django-admin

Útil para administración global y creación de proyectos

2

manage.py

Interactúa con el proyecto específico y sus configuraciones

Ambas herramientas son complementarias y esenciales en el desarrollo con Django, cada una con su propósito específico en el flujo de trabajo.

Estructura de carpetas de Django

Un proyecto Django incluye: manage.py, carpeta del proyecto con settings.py, urls.py, y __init__.py. Cada archivo tiene un propósito específico.

manage.py

Herramienta de línea de comandos para interactuar con el proyecto

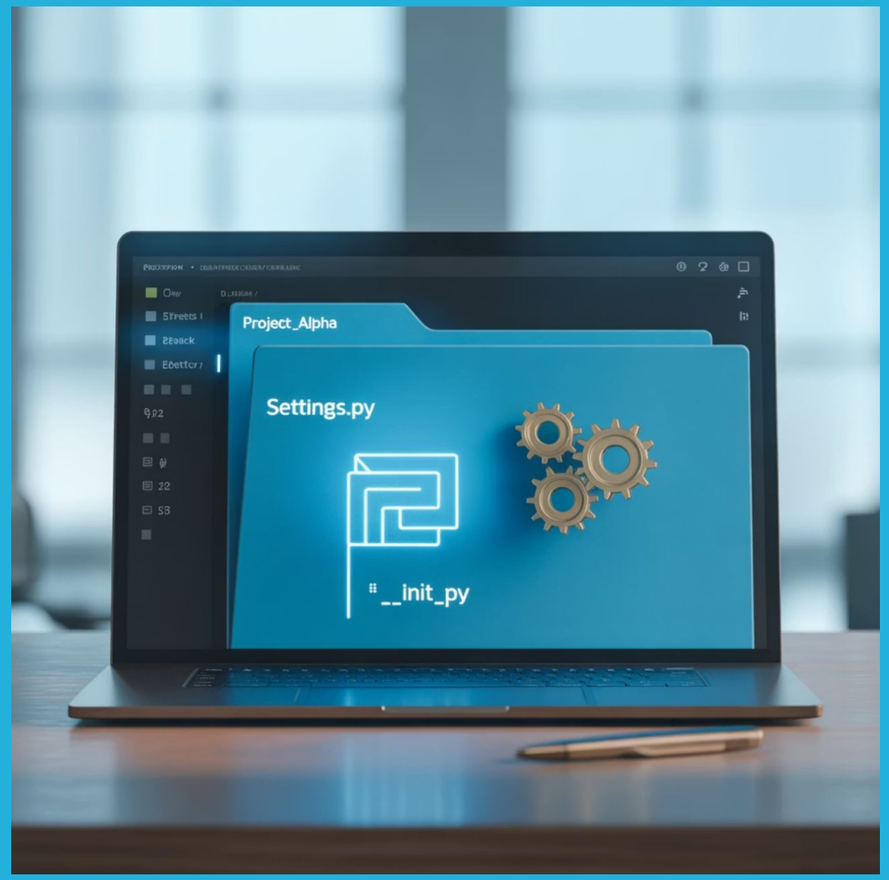
settings.py

Configuración principal del proyecto Django

urls.py

Configuración de rutas y URLs del proyecto

Archivos de configuración



__init__.py

Convierte la carpeta en un paquete Python, permitiendo importar módulos desde el directorio

settings.py

Núcleo de la configuración del proyecto, contiene todas las variables de configuración importantes

Configuración de templates y paths

Configurar la carpeta de plantillas y paths permite organizar vistas dinámicas y gestionar archivos de manera eficiente.

1

Definir TEMPLATES

Configurar la ruta de las plantillas HTML

2

Establecer STATIC_URL

Definir la ruta para archivos estáticos

3

Configurar MEDIA

Gestionar archivos subidos por usuarios

Configuración de urls.py

urls.py define cómo se enlazan las URLs a las vistas y aplicaciones dentro del proyecto. Es clave para la navegación web.

URL solicitada

El usuario accede a una dirección específica

Patrón coincidente

Django busca el patrón que coincida en urls.py

Respuesta generada

Se devuelve la respuesta al navegador

Vista ejecutada

Se ejecuta la vista asociada al patrón

Introducción a aplicaciones y MVC en Django

Django sigue el patrón MVC (Modelo-Vista-Controlador). Las aplicaciones se crean para separar responsabilidades y organizar el proyecto.

Modelo

Gestiona los datos y la lógica de negocio

Vista

Procesa las solicitudes y devuelve respuestas

Controlador

Coordina la interacción entre modelo y vista

Crear aplicaciones en Django

Comando:

```
python manage.py startapp nombre_app
```

Esto genera la estructura de la aplicación dentro del proyecto. Cada aplicación es un módulo Python que agrupa funcionalidades relacionadas.

Estructura creada

Se genera la carpeta con archivos básicos

Registrar en settings

Agregar la app a INSTALLED_APPS

Desarrollar funcionalidad

Crear modelos, vistas y plantillas

Componentes esenciales de Django

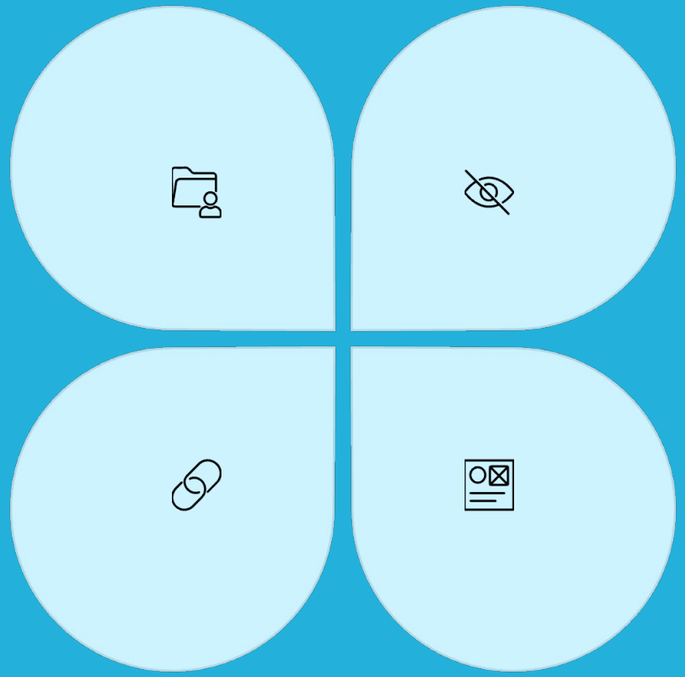
Modelos, vistas, plantillas y URL routing son los pilares de cualquier proyecto Django.

Modelos

Definen la estructura de datos

Vistas

Procesan lógica de negocio



URL Routing

Conecta URLs con vistas

Plantillas

Presentan la interfaz de usuario

Comandos de ayuda y flujo de trabajo

Usa `python manage.py help` para obtener asistencia con comandos. Mantener un flujo de trabajo claro facilita el desarrollo y depuración del proyecto.

Documentación integrada

Django incluye ayuda detallada para cada comando disponible

Flujo organizado

Seguir buenas prácticas mejora la productividad del desarrollo

Depuración eficiente

Las herramientas administrativas facilitan la resolución de problemas

¡Felicidades! Ahora tienes las bases para crear y gestionar proyectos Django profesionales.