Python 2023

Diego Saavedra

Dec 2, 2023

Table of contents

1 Bienvenida **3**
- 1.1 ¿Qué es este Curso? . . . 3
- 1.2 ¿A quién está dirigido? . . . 3
- 1.3 ¿Cómo contribuir? . . . 4

I Unidad 0: Git y Github **5**

2 Módulo 0: Git/Github **6**
- 2.1 Introducción a Git y Control de Versiones. . . . 6
- 2.2 Configuración de un Repositorio en GitHub . . . 6
- 2.3 Uso Básico de Comandos de Git. . . . 6
- 2.4 Colaboración en un Proyecto Utilizando GitHub . . . 7

3 Ejercicio práctico: **8**
- 3.1 Configurar un Repositorio y Realizar Cambios . . . 8
- 3.2 Resolución del Ejercicio Práctico . . . 8

II Unidad 1: Introducción a Django **10**

4 Módulo 1: Introducción a Django. **11**
- 4.1 ¿Qué es Django y por qué utilizarlo? . . . 11
- 4.2 MVC vs MTV. . . . 11
- 4.3 Instalación de Django 4.2.3 y Configuración del Entorno de Desarrollo . . . 11
- 4.4 Creación de un Proyecto en Django. . . . 12
  - 4.4.1 Estructura de directorios generada: . . . 12
  - 4.4.2 Estructura de Directorios de un Proyecto Django. . . . 12
- 4.5 Creación de una Aplicación en Django . . . 14

5 Ejemplo Práctico: **15**
- 5.1 Creación de un Proyecto de E-commerce en Django . . . 15

6 Actividad Práctica. **17**
- 6.1 Creación de Un Blog . . . 17
- 6.2 Resolución de la Actividad Práctica: . . . 17

1 Bienvenida

¡Bienvenidos al Curso Completo de Python, analizaremos desde los fundamentos hasta aplicaciones prácticas!

1.1 ¿Qué es este Curso?

Este curso exhaustivo te llevará desde los fundamentos básicos de la programación hasta la creación de aplicaciones prácticas utilizando el lenguaje de programación Python. A través de una combinación de teoría y ejercicios prácticos, te sumergirás en los conceptos esenciales de la programación y avanzarás hacia la construcción de proyectos reales. Desde la instalación de herramientas hasta la creación de una API con Django Rest Framework, este curso te proporcionará una comprensión sólida y práctica de Python y su aplicación en el mundo real.

1.2 ¿A quién está dirigido?

Este curso está diseñado para principiantes y aquellos con poca o ninguna experiencia en programación. No importa si eres un estudiante curioso, un profesional que busca cambiar de carrera o simplemente alguien que desea aprender a programar: este curso es para ti. Desde adolescentes hasta adultos, todos son bienvenidos a participar y explorar el emocionante mundo de la programación a través de Python.

1.3 ¿Cómo contribuir?

Valoramos tu participación en este curso. Si encuentras errores, deseas sugerir mejoras o agregar contenido adicional, ¡nos encantaría escucharte! Puedes contribuir a través de nuestra plataforma en línea, donde puedes compartir tus comentarios y sugerencias. Juntos, podemos mejorar continuamente este recurso educativo para beneficiar a la comunidad de estudiantes y entusiastas de la programación.

Este libro ha sido creado con el objetivo de brindar acceso gratuito y universal al conocimiento. Estará disponible en línea para que cualquiera, sin importar su ubicación o circunstancias, pueda acceder y aprender a su propio ritmo.

¡Esperamos que disfrutes este emocionante viaje de aprendizaje y descubrimiento en el mundo de la programación con Python!

Part I

Unidad 0: Git y Github

2 Módulo 0: Git/Github

2.1 Introducción a Git y Control de Versiones.

Git es un sistema de control de versiones distribuido que permite rastrear cambios en el código de forma eficiente.

El control de versiones es esencial para mantener un historial de los cambios realizados en un proyecto y facilitar la colaboración en equipo.

2.2 Configuración de un Repositorio en GitHub

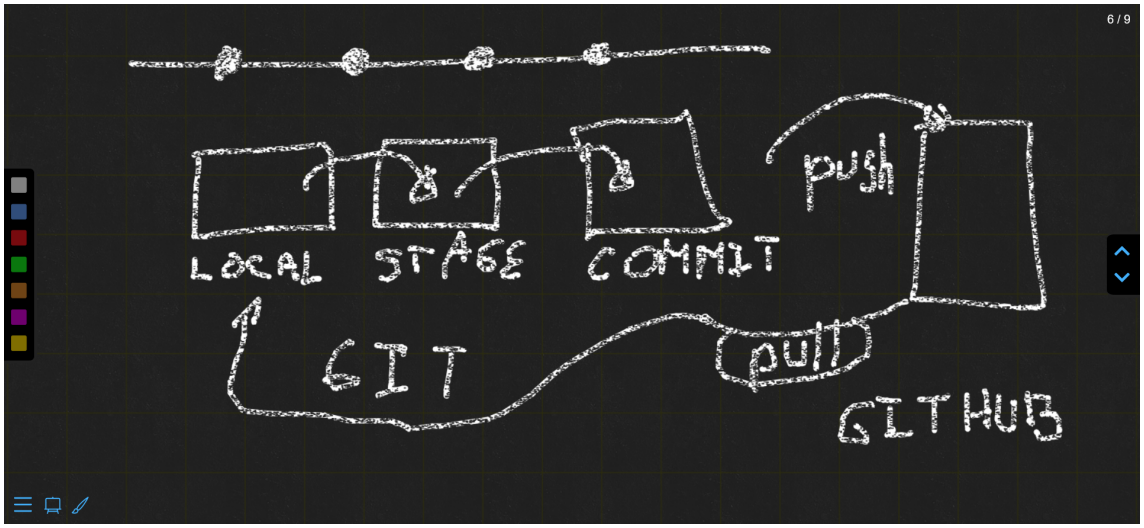
GitHub es una plataforma de alojamiento de repositorios Git en la nube.

Crea una cuenta en GitHub si no tienes una.

Para configurar un nuevo repositorio en GitHub, sigue las instrucciones en la página web.

2.3 Uso Básico de Comandos de Git.

Flujo básico de git.



En la imágen anterior se describe el proceso básico para pasar de **Local** a **Stage**, de **Stage** a **Commit** y de **Commit** a **Github** y/o **Cloud**.

Clonar un repositorio existente desde GitHub a tu máquina local:

```
git clone url_repositorio
```

Crear una nueva rama para trabajar en una funcionalidad específica:

```
git checkout -b nombre_rama
```

Hacer commits para guardar los cambios realizados:

```
git add archivo_modificado.py
git commit -m "Mensaje del commit"
```

Fusionar Ramas y Resolución de Conflictos

Cambiar a la rama principal:

```
git checkout main
```

Fusionar una rama con la rama principal:

```
git merge nombre_rama
```

Resolver conflictos que puedan surgir durante la fusión.

2.4 Colaboración en un Proyecto Utilizando GitHub

Para colaborar en un proyecto en GitHub, realiza lo siguiente:

Haz un Fork del repositorio original en tu cuenta de GitHub.

Clona tu Fork a tu máquina local.

Crea una nueva rama para realizar tus cambios.

Hace commits en tu rama.

Envía un Pull Request al repositorio original para que los colaboradores revisen tus cambios y los fusionen.

3 Ejercicio práctico:

3.1 Configurar un Repositorio y Realizar Cambios

1. Crea un nuevo repositorio en GitHub.
2. Clona el repositorio a tu máquina local con el comando git clone url_repositorio.
3. Crea una nueva rama con el comando git checkout -b nombre_rama.
4. Realiza cambios en tus archivos y haz commits con git add y git commit.
5. Cambia a la rama principal con git checkout main.
6. Fusiona tu rama con la rama principal con git merge nombre_rama.
7. Envía tus cambios al repositorio en GitHub con git push origin main.

¡Excelente! Ahora has aprendido los conceptos básicos de Git y GitHub, así como cómo configurar un repositorio y colaborar en un proyecto utilizando esta plataforma. En los próximos módulos, abordaremos el desarrollo web con Django.

3.2 Resolución del Ejercicio Práctico

Configurar un Repositorio y Realizar Cambios en el Proyecto Blog

En este ejercicio, configuraremos un repositorio Git para el proyecto del blog que vamos a dearrollar en los módulos 1 al 4. Luego, haremos algunos cambios en el proyecto y realizaremos commits para registrar esos cambios en el historial de versiones.

Paso 1: Configurar el repositorio en GitHub

Abre tu cuenta de GitHub y haz clic en el botón “New” para crear un nuevo repositorio.

Asigna un nombre al repositorio y configura la visibilidad como desees.

Opcionalmente, puedes agregar una descripción y una licencia.

Haz clic en “Create repository” para crear el repositorio en GitHub.

Paso 2: Clonar el repositorio en tu máquina local

Copia la URL del repositorio que acabas de crear en GitHub (se verá como https://github.com/tu_usuario/nombre_repositorio.git).

Abre una terminal o línea de comandos en la carpeta donde deseas clonar el repositorio.

Utiliza el siguiente comando para clonar el repositorio en tu máquina local:

```
git clone url_repositorio
```

Reemplaza “url_repositorio” con la URL que copiaste en el paso 1.

Paso 3: Realizar cambios en el proyecto del blog

Abre el proyecto del blog en tu editor de código o IDE favorito.

Realiza algunos cambios en los archivos de tu proyecto, como agregar nuevas funcionalidades, modificar plantillas o corregir errores.

Paso 4: Hacer commits para registrar los cambios

Después de hacer cambios en el proyecto, utiliza los siguientes comandos para hacer commit y registrar esos cambios en el historial de versiones:

```
git add .
git commit -m "Mensaje descriptivo del commit"
```

El comando git add . agrega todos los cambios realizados en los archivos del proyecto al área de preparación, y el comando git commit -m "Mensaje" crea un nuevo commit con un mensaje descriptivo para los cambios realizados.

Paso 5: Enviar los cambios al repositorio en GitHub

Después de hacer commit de los cambios en tu repositorio local, utiliza el siguiente comando para enviar los cambios al repositorio en GitHub:

```
git push origin main`
```

Reemplaza "main" con el nombre de la rama principal de tu proyecto si utilizas otro nombre diferente.

¡Felicitaciones! Ahora has configurado un repositorio Git para tu proyecto y has realizado cambios en el proyecto, registrando esos cambios mediante commits.

Los cambios ahora están disponibles en el repositorio en GitHub. Puedes repetir estos pasos cada vez que desees realizar cambios en el proyecto y mantener un historial de versiones de tu proyecto en GitHub.

Part II

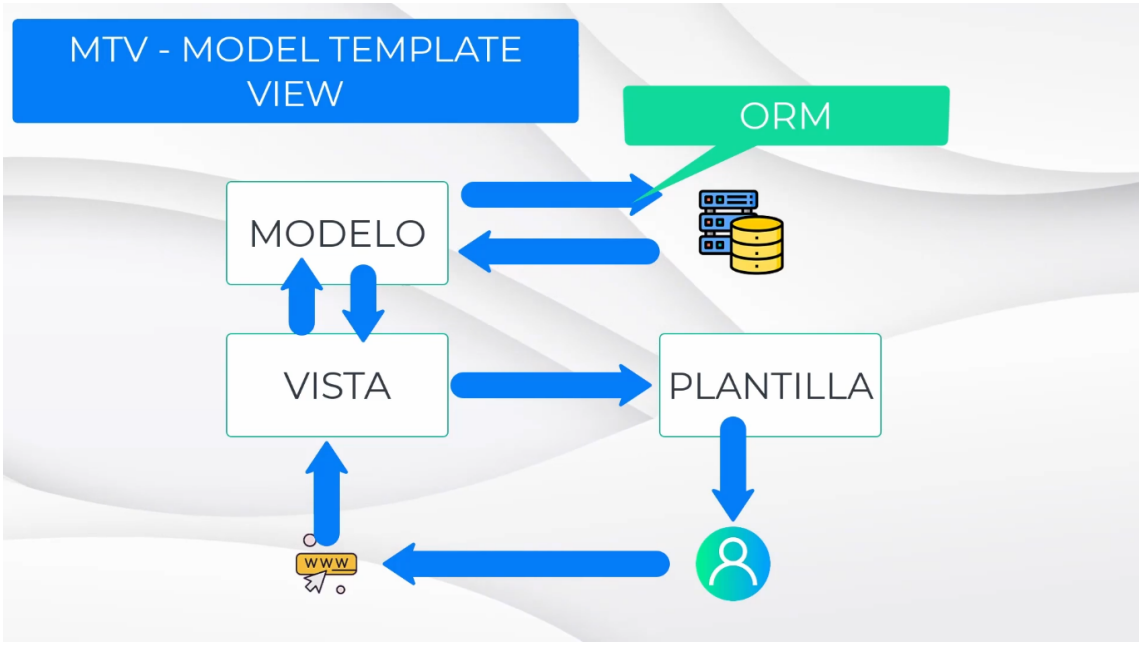
Unidad 1: Introducción a Django

4 Módulo 1: Introducción a Django.

4.1 ¿Qué es Django y por qué utilizarlo?

- Django es un framework web de alto nivel basado en Python.
- Facilita el desarrollo rápido de aplicaciones web robustas y seguras.
- Ventajas de Django: MVC (Modelo-Vista-Controlador), administrador de base de datos, seguridad integrada y comunidad activa.

4.2 MVC vs MTV.



4.3 Instalación de Django 4.2.3 y Configuración del Entorno de Desarrollo

Para instalar Django 4.2.3, se recomienda utilizar un entorno virtual (por ejemplo, con virtualenv o conda).

Comandos para instalar Django y crear un entorno virtual:

```
# Instalación de virtualenv

pip install virtualenv

# Creación del entorno virtual

python -m venv env

# Activación del Entorno Virtual

cd env/Scripts
activate

# Salir del directorio Scripts
cd ../../

# Instalación de Django usando pip
pip install django==4.2.3
```

4.4 Creación de un Proyecto en Django.

Para crear un nuevo proyecto Django, utilizamos el comando

```
django-admin startproject nombre_proyecto .
```

4.4.1 Estructura de directorios generada:

```
nombre_proyecto/
    manage.py
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

4.4.2 Estructura de Directorios de un Proyecto Django.

La estructura de directorios de un proyecto Django se organiza de la siguiente manera:

```
nombre_proyecto/
   manage.py
```

```
nombre_proyecto/
     __init__.py
     settings.py
     urls.py
     asgi.py
     wsgi.py

otras_aplicaciones/
      ...
```

Directorio y/o Archivo	Descripción
nomb re_proyecto/	Es el directorio raíz del proyecto. Contiene el archivo “manage.py”, que es una herramienta para administrar el proyecto y ejecutar comandos de Django.
• *nombre_proyec to/nomb re_proyecto/**	Es el directorio de la configuración del proyecto. Contiene varios archivos esenciales:
init.py	Este archivo indica a Python que el directorio es un paquete y permite la importación de módulos dentro de él.
• *settings.py**	Aquí se encuentran todas las configuraciones del proyecto, como bases de datos, aplicaciones instaladas, rutas de plantillas, configuraciones de seguridad, etc.
urls.py	Contiene las configuraciones de las URLs del proyecto, es decir, cómo se manejan las solicitudes y se mapean a las vistas.
asgi.py y **wsgi.py**	Son archivos de configuración para el servidor ASGI (Asynchronous Server Gateway Interface) y WSGI (Web Server Gateway Interface), respectivamente. Estos archivos son utilizados por servidores web para servir la aplicación Django.
otras_a plicaciones/	Es un directorio opcional donde puedes organizar las aplicaciones adicionales que desarrolles para el proyecto. Cada aplicación puede tener su propia estructura de directorios.

4.5 Creación de una Aplicación en Django

Una aplicación es un componente reutilizable de un proyecto Django. Comando para crear una nueva aplicación:

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

Estructura de directorios de una aplicación:

```
nombre_app/
    migrations/
        ...
        __init__.py

        admin.py
        apps.py
        models.py
        tests.py
        views.py
```

D irectorio y/o Archivo	Descripción
n ombre_app	Es el directorio raíz de la aplicación. Contiene los archivos esenciales y directorios para el funcionamiento de la aplicación.
mi grations/	Es un directorio generado automáticamente por Django cuando se realizan cambios en los modelos de la aplicación. Contiene archivos de migración que representan los cambios en la base de datos.
init.py	Este archivo indica a Python que el directorio es un paquete y permite la importación de módulos dentro de él.
admin.py	En este archivo se pueden registrar los modelos de la aplicación para que aparezcan en el panel de administración de Django.
apps.py	Es el archivo donde se define la configuración de la aplicación, como su nombre y configuraciones adicionales.
models.py	Aquí se definen los modelos de la aplicación utilizando la clase "Model" de Django. Los modelos representan las tablas en la base de datos y definen los campos y relaciones de la aplicación.
tests.py	Es el archivo donde se pueden escribir pruebas unitarias y de integración para la aplicación.
views.py	Contiene las vistas de la aplicación, que son funciones o clases que manejan las solicitudes y generan las respuestas. Las vistas determinan qué se muestra en las páginas web de la aplicación.

5 Ejemplo Práctico:

5.1 Creación de un Proyecto de E-commerce en Django

En esta actividad práctica, crearás un nuevo proyecto de e-commerce en Django desde cero. Asegúrate de tener Django instalado en tu entorno de desarrollo antes de comenzar.

Paso 1: Crear un Proyecto de Django

Abre una terminal o línea de comandos en la ubicación donde desees crear tu proyecto de e-commerce.

Ejecuta el siguiente comando para crear un nuevo proyecto Django llamado "mi_ecommerce":

```
django-admin startproject mi_ecommerce
```

Verás que se ha creado un nuevo directorio llamado "mi_ecommerce" que contiene la estructura inicial del proyecto.

Paso 2: Crear una Aplicación para el E-commerce

Cambia al directorio recién creado "mi_ecommerce":

```
cd mi_ecommerce
```

Ahora, crea una nueva aplicación llamada "ecommerce" utilizando el siguiente comando:

```
python manage.py startapp ecommerce
```

Se creará un nuevo directorio "ecommerce" dentro de tu proyecto, que contendrá todos los archivos necesarios para la aplicación.

Paso 3: Configurar el Proyecto y la Aplicación

Abre el archivo "settings.py" ubicado en el directorio "mi_ecommerce/mi_ecommerce".

Asegúrate de agregar la aplicación "ecommerce" en la lista de "INSTALLED_APPS" para que Django la reconozca:

```
INSTALLED_APPS = [
    # Otras aplicaciones...
    'ecommerce',
]
```

Paso 4: Ejecutar el Servidor de Desarrollo

Ahora, ejecuta el servidor de desarrollo para ver el proyecto en acción:

```
python manage.py runserver
```

Abre tu navegador y visita la dirección "http://localhost:8000/". Deberías ver la página de bienvenida de Django.

Si has llegado a este punto, ¡Felicidades! Has creado con éxito un nuevo proyecto de e-commerce en Django y una aplicación llamada "ecommerce". Ahora puedes comenzar a desarrollar las funcionalidades del e-commerce y diseñar las diapositivas para cada uno de los módulos del curso.

¡Buena suerte!

6 Actividad Práctica.

6.1 Creación de Un Blog

En esta actividad práctica, crearás un nuevo proyecto de Blog en Django desde cero. Asegúrate de tener Django instalado en tu entorno de desarrollo antes de comenzar.

- ☐ Iniciar un Repositorio
- ☐ Crear el entorno virtual
- ☐ Activación del entorno virtual
- ☐ Instalación de Django
- ☐ Crear el archivo requirements.txt
- ☐ Agregar el archivo .gitignore
- ☐ Crear un Proyecto con Django
- ☐ Crear la App dentro del Proyecto
- ☐ Agregar la App al Archivo settings.py del Proyecto
- ☐ Correr el Servidor de Pruebas

6.2 Resolución de la Actividad Práctica:

Para crear el proyecto de Blog en Django y completar las actividades prácticas, sigue los siguientes pasos:

Paso 1: Iniciar un Repositorio

Inicia un nuevo repositorio en tu sistema de control de versiones (por ejemplo, en GitHub) para gestionar el código de tu proyecto.

Paso 2: Crear el Entorno Virtual

Crea un nuevo entorno virtual para aislar las dependencias del proyecto y evitar conflictos con otras aplicaciones Python instaladas en tu sistema.

```
# Ejecutar en la terminal o consola
python -m venv mi_entorno_virtual
```

Paso 3: Activación del Entorno Virtual

Activa el entorno virtual antes de instalar Django o trabajar en el proyecto.

```
# En Windows
mi_entorno_virtual\Scripts\activate
```

```
# En macOS o Linux
source mi_entorno_virtual/bin/activate
```

Paso 4: Instalación de Django

Dentro del entorno virtual, instala Django utilizando pip.

```
# Ejecutar en la terminal o consola
pip install django
```

Paso 5: Crear el Archivo requirements.txt

Crea un archivo "requirements.txt" en la raíz del proyecto para almacenar todas las dependencias de Python utilizadas en el proyecto.

```
# requirements.txt
Django==4.2.3
```

Paso 6: Agregar el Archivo .gitignore

Crea un archivo ".gitignore" en la raíz del proyecto para evitar que los archivos y directorios innecesarios se incluyan en el repositorio.

```
# .gitignore
mi_entorno_virtual/
__pycache__/
*.pyc
db.sqlite3
```

Paso 7: Crear un Proyecto con Django

Crea un nuevo proyecto de Django llamado "mi_blog".

```
# Ejecutar en la terminal o consola
django-admin startproject mi_blog
```

Paso 8: Crear la App dentro del Proyecto

Crea una nueva aplicación llamada "blog" dentro del proyecto "mi_blog".

```
# Ejecutar en la terminal o consola
cd mi_blog
python manage.py startapp blog
```

Paso 9: Agregar la App al Archivo settings.py del Proyecto

Abre el archivo "settings.py" en la carpeta "mi_blog" y agrega la aplicación "blog" a la lista de aplicaciones instaladas.

```
# settings.py

INSTALLED_APPS = [
    ...
    'blog',
]
```

Paso 10: Correr el Servidor de Pruebas

Para verificar que todo está configurado correctamente, inicia el servidor de pruebas de Django.

```
# Ejecutar en la terminal o consola
python manage.py runserver
```

Ahora podrás acceder a la página de inicio de Django en tu navegador web en la dirección "http://127.0.0.1:8000/".

Con estos pasos, has creado un nuevo proyecto de Blog en Django, configurado un entorno virtual, instalado Django y creado una nueva aplicación dentro del proyecto. Ahora estás listo para comenzar a desarrollar tu blog utilizando Django.