Foro construido con Django Framework

Antes de comenzar a crear un proyecto con Django

Comenzaremos creando el proyecto, para crearlo necesitaremos la versión más nueva de python, podemos descargarla del sitio web oficial https://www.python.org. Ictualmente estoy usando **python version 3.11**.

Crearemos una carpeta de nuestro proyecto.

Windows

```
> md my_directory
```

Linux

```
$ mkdir my_directory
```

y accede a la carpeta

Linux, Windows, Mac

```
cd my_directory
```

GIT y GITHUB

En este caso usaremos git o github para poder visualizar mejor nuestras versiones y tener un orden en nuestro proyecto, este curso no tratará a profundidad todo lo relacionado con git, sin embargo puedes documentarte con el libro oficial que se encuentra en la página oficial git página o descargando directamente su pdf.

Inicializamos nuestro entorno git

Abriremos nuestra consola y escribiremos

```
$ git init
```

crearemos un archivo llamado `README.md` y un archivo llamado `.gitignore`

Windows

```
> fsutil file createnew .\README.md 1
> fsutil file createnew .\.gitignore 1
```

Linux or Mac

```
$ touch README.md
$ touch .gitignore
```

¿Qué es esto de README y `.gitignore`

README.md

El README es un archivo que se encuentra en la raíz de un proyecto de software y proporciona información sobre el proyecto a los usuarios y colaboradores. El README normalmente incluye información sobre cómo instalar y usar el software, así como detalles sobre cómo contribuir al proyecto. También puede incluir información sobre la licencia del software, el equipo de desarrollo y cualquier otra información relevante para los usuarios y colaboradores del proyecto. Es común que el README se escriba en un formato de texto sin formato, como Markdown, o en formato HTML.

.gitignore

El archivo .gitignore es un archivo utilizado en proyectos de software que utilizan Git como sistema de control de versiones. El archivo .gitignore le dice a Git qué archivos o carpetas ignorar al rastrear el proyecto. Esto es útil para evitar que ciertos tipos de archivos, como binarios o archivos temporales, se incluyan en el repositorio de Git.

Por ejemplo, si tiene un proyecto de software que incluye una carpeta de archivos binarios, puede agregar la ruta de la carpeta a su archivo .gitignore para evitar que Git rastree los cambios en esos archivos. De esta manera, puede asegurarse de que solo se rastrean los cambios en el código fuente del proyecto, y no en los archivos binarios que se pueden regenerar fácilmente.

El archivo .gitignore se usa a menudo en proyectos de software de código abierto, ya que permite a los colaboradores evitar agregar accidentalmente archivos que no deberían incluirse en el repositorio de Git.

Ahora crearemos un repositorio en tu cuenta de github. Una vez creado tu repositorio, volvemos a la carpeta donde alojaremos nuestro proyecto y escribimos lo siguiente línea por línea:

Encontrará estas instrucciones una vez que cree el nuevo repositorio

¿Qué hacer con README y .gitignore?

Abriremos README con nuestro editor favorito y agregaremos lo siguiente

Incluye el símbolo # esto representa un h1 (título principal)

```
# Título del proyecto
```

O si quieres sentirte hacker puedes usar en tu terminal escribe lo siguiente:

Esto funciona tanto en windows como en linux

```
echo "# Title from project" >> README.md
```

Ahora abrimos nuestro archivo .gitignore

Agrega los elementos que quieras ignorar en este caso el env de virtualización de python que veremos más adelante en este curso.

```
env_name/
```

Primero confirme con el repositorio en github

crearemos nuestro primer commit(compromiso) y lo enviaremos a nuestro repositorio con el siguiente script de git.

```
git add README.md
git commit -m "name from commit"
git branch -M main
git remote add origin https://github.com/**NAME_USER**/**NAME_PROJECT**.git
git push -u origin main
```

Verifique su repositorio creado, vuelva a cargar la página y verá los cambios en un repositorio.

¿Qué es todo esto que estamos viendo desde git? te lo resumo.

git init

```
git init
```

`git init` es un comando de Git que se usa para inicializar un nuevo repositorio de Git. Cuando ejecuta `git init` en una carpeta, Git crea un nuevo repositorio en esa carpeta y configura todo para que pueda comenzar a rastrear los cambios en los archivos del proyecto.

El comando `git init` es especialmente útil cuando comienza a trabajar en un nuevo proyecto de software y desea usar Git para realizar un seguimiento de los cambios en el código fuente. Al ejecutar `git init`, puede crear un repositorio local en su máquina y comenzar a confirmar sus cambios sin tener que conectarse a un servidor remoto.

Por ejemplo, para inicializar un nuevo repositorio Git en una carpeta llamada my-project, puede usar el comando `git init` de la siguiente manera:

```
cd my_project
git init
```

Una vez que inicialice el repositorio, puede comenzar a confirmar sus cambios usando comandos como `git add` y `git commit`. También puede agregar un repositorio remoto usando el comando `git remote` add para que pueda enviar sus cambios al repositorio remoto y extraer los cambios del repositorio remoto a su repositorio local.

git add

```
git add
```

`git add` es un comando de Git que se usa para agregar archivos al área de ensayo de Git. El área de preparación es un área temporal donde se almacenan los cambios que se incluirán en la próxima confirmación.

Cuando realiza cambios en los archivos de su proyecto, Git no los rastrea automáticamente. En su lugar, debe usar el comando `git add` para decirle a Git qué archivos deben incluirse en la próxima confirmación. Luego puede usar el comando `git commit` para crear la confirmación con los cambios agregados.

Por ejemplo, si modificó el archivo `file_name_extension` y desea incluir los cambios en la próxima confirmación, puede usar el comando git add `file_name_extension` para agregar el archivo al área de ensayo. Luego puede usar el comando `git commit` para confirmar los cambios en file_name_extension.

Es importante tener en cuenta que el comando `git add` solo agrega los cambios a la zona de ensayo. No crea la confirmación ni realiza ningún otro cambio en el repositorio. Para crear el compromiso, debe usar el comando `git commit`.

Por ejemplo:

```
git add file_name_extension
```

or

```
git add .
```

el punto (.) representa que se seleccionan todos los archivos que tienen un cambio

git commit

```
git commit
```

En el sistema de control de versiones de Git, una confirmación es una operación que se utiliza para guardar los cambios realizados en el código fuente del proyecto en el historial del repositorio. Cada confirmación tiene un mensaje asociado que describe los cambios realizados en la confirmación. Las confirmaciones se utilizan para realizar un seguimiento de los cambios realizados en el proyecto a lo largo del tiempo y para revertir fácilmente cambios específicos si es necesario.

To commit a software project using Git, you must first add the files you want to commit using the `git add` command. Then, you must use the `git commit` command to create the commit.

```
git add example.py
git commit -m "I added a function to calculate the sum of two numbers"
```

En este ejemplo, el comando git add agrega el archivo `file_example` al área de preparación y luego el comando `git commit` crea una confirmación que incluye los cambios realizados en `file_example` y asigna a la confirmación un mensaje que describe los cambios realizados. .

Es importante tener en cuenta que una confirmación solo incluye los cambios realizados en los archivos que se agregaron al área de ensayo mediante `git add`. Si realiza cambios en otros archivos y no los agrega al área de preparación, esos cambios no se incluirán en la confirmación..

git branch -M main

```
git branch -M main
```

El comando `git branch -M main` es una forma de cambiar el nombre de una rama en un repositorio de Git. La opción `-M` indica que se debe cambiar el nombre de la rama actual, en lugar de crear una nueva rama.

Por ejemplo, si tiene una rama llamada función en su repositorio y desea cambiarle el nombre a principal, puede usar el comando `git branch -M main`. Esto cambiará el nombre de la rama de características a principal y todas las confirmaciones en esa rama seguirán siendo parte de la principal.

Es importante tener en cuenta que el comando `git branch -M main` solo cambia el nombre de la rama. No cambia la rama seleccionada actualmente ni realiza ningún otro cambio en el repositorio. Si desea cambiar a la rama principal después de cambiarle el nombre, debe usar el comando principal de git checkout.

Se recomienda usar este comando con cuidado, ya que puede causar confusión o problemas si otros usuarios del repositorio esperan que la rama principal se llame de manera diferente.

git remote add origin

```
git remote add origin https://www.github.com/...
```

El comando `git remote add origin` se usa para agregar un repositorio remoto a un proyecto Git local. Un repositorio remoto es una copia del repositorio que se encuentra en otro lugar, como en un servidor o en la nube. Al agregar un repositorio remoto, puede enviar sus cambios locales al repositorio remoto y también puede extraer cambios del repositorio remoto a su repositorio local.

El comando `git remote add` toma dos argumentos: el nombre del repositorio remoto y la dirección del repositorio. Por ejemplo, si desea agregar un repositorio remoto llamado origen ubicado en https://example.com/repo.git, puede usar el comando:

```
git remote add origin https://example.com/repo.git`
```

Una vez que agrega un repositorio remoto, puede usar comandos como `git push` y `git pull` para sincronizar sus cambios con el repositorio remoto. Por ejemplo, para enviar sus cambios locales al repositorio de origen remoto, puede usar el comando git push origin master, donde maestro es el nombre de la rama en el repositorio remoto donde desea enviar sus cambios.

Nuestro árbol de carpetas ahora debería parecerse a sí mismo:

Folder tree 1

├── Forum
│ └── README.md
│ └── .gitignore

Que es Django?

Django es un marco de desarrollo web de código abierto escrito en Python. Fue diseñado para ayudar a los desarrolladores a crear aplicaciones web de forma rápida y sencilla. Django ofrece una amplia variedad de funciones y herramientas que facilitan el desarrollo de aplicaciones web, como un sólido sistema de administración de bases de datos, un marco de autenticación y autorización seguro y un motor de plantillas para crear interfaces de usuario. Además, Django tiene una gran comunidad de desarrolladores que constantemente aportan nuevas características y mejoras. Es utilizado por muchas empresas y organizaciones de todo el mundo para crear aplicaciones web de alta calidad.

Empecemos

Primero crearemos una virtualización para ellos necesitaremos instalar virtualenv, hay varios paquetes diferentes que nos permiten virtualizar nuestros proyectos python. en este caso usaremos virtualenv.

Instalaremos virtualenv:

Windows

```
pip install virtualenv` or `python -m pip install virtualenv
```

Linux

```
sudo pip install virtualenv` or `sudo python -m pip install virtualenv
```

Alternativas a virtualenv

Tenemos otros paquetes como virtualenv estos son los más populares

- pyenv
- Pipenv
- Poetry
- Autoenv
- rez
- Pew
- p
- virtualenvwrapper
- venv

Crearemos una virtualizacion con virtualenv

Crearemos una nueva virtualización con el siguiente comando de terminal:

```
python -m virtualenv name_virtualization
```

or

```
virtualenv -m name_virtualization
```

windows

```
name_virtualization\Script\activate
```

linux or mac

```
source name_virtualization/bin/activate
```

Instalando Django

Para comenzar con Django, primero debe asegurarse de tener Python instalado en su sistema. Luego puede instalar Django usando el administrador de paquetes `pip` de Python. Abra una consola y escriba:

Windows

```
pip install django
```

or

```
python -m pip install django
```

Linux

```
sudo pip install django
```

or

```
sudo python -m pip install django
```

Creando tu proyecto

Un proyecto Django se divide en varias partes Antes de comenzar tenemos que entender que un **proyecto** y una **aplicación**.

Que es un proyecto en Django?

Un proyecto Django es un conjunto de código utilizado para desarrollar una aplicación o conjunto de aplicaciones. Un proyecto Django consiste en una o más aplicaciones, así como un conjunto de archivos de configuración y código de soporte que controlan la configuración del proyecto y cómo las aplicaciones se relacionan entre sí.

- Aplicaciones: Cada aplicación es un conjunto de código que realiza una tarea específica dentro del proyecto. Un proyecto Django puede tener una o más aplicaciones.
- Configuración: un conjunto de archivos de configuración que controlan la configuración del proyecto, incluidos elementos como la base de datos que se utilizará, las opciones de autenticación y la configuración del servidor web.
- Código de soporte: archivos adicionales que se usan para controlar la funcionalidad del proyecto, como vistas generales que se usan en varias aplicaciones o archivos de plantilla que se usan para presentar la interfaz de usuario.

Conclusión

Un proyecto Django es el conjunto completo de código utilizado para desarrollar una aplicación o conjunto de aplicaciones. Se puede pensar en un proyecto Django como la "caja grande" que contiene todo el código y la configuración necesarios para que una aplicación funcione..

Que es un App en Django?

Una aplicación de Django es un conjunto de código que realiza una tarea específica dentro de un proyecto de Django. Un proyecto de Django puede tener una o más aplicaciones, y cada aplicación se puede reutilizar en diferentes proyectos de Django.

Una aplicación de Django generalmente contiene al menos dos partes: un conjunto de modelos que representan los datos que se manipulan en la aplicación y un conjunto de vistas que controlan cómo se muestran esos datos al usuario. También puede incluir plantillas (que controlan la presentación de la interfaz de usuario) y formularios (que facilitan la entrada de datos por parte del usuario).

Por ejemplo, una aplicación de Django podría ser un blog, un sistema de gestión de tareas o una tienda en línea. Cada aplicación maneja una parte específica de la funcionalidad del proyecto y se puede reutilizar en diferentes proyectos de Django.

Sigamos con la creación de nuestro proyecto

`django-admin startproject nombre_proyecto` es un comando de Django que se usa para crear un nuevo proyecto de Django. Este comando crea un directorio básico y una estructura de archivos para el proyecto, que incluye:

- Un directorio padre con el nombre del proyecto, que contiene todos los archivos y recursos del proyecto.
- Un archivo `manage.py`, que es un script que le permite administrar el proyecto usando comandos de línea de comandos.
- Un archivo `settings.py`, que contiene la configuración del proyecto, como el idioma, el formato de fecha y hora y la configuración de la base de datos.
- Un archivo `urls.py`, que define las URL del proyecto y las asocia a las vistas correspondientes.
- Un archivo `asgi.py`, que es un archivo de entrada para el servidor ASGI (Asynchronous Server Gateway Interface).
- Un archivo `wsgi.py`, que es un archivo de entrada para el servidor WSGI (Web Server Gateway Interface).

Es importante tener en cuenta que esta es solo una estructura básica y puede agregar otros archivos y directorios según sea necesario. Por ejemplo, puede crear aplicaciones adicionales para el proyecto usando el comando `django-admin startapp`, o agregar archivos static/ o templates/ para almacenar archivos estáticos o plantillas HTML, respectivamente.

Una vez que se instala Django, puede crear un nuevo proyecto ejecutando el siguiente comando en la consola:

```
django-admin startproject app
```

Este comando creará una nueva carpeta con el nombre de su proyecto y dentro generará varios archivos y carpetas necesarios para iniciar un proyecto de Django.

Para iniciar el servidor de desarrollo de Django y ver su proyecto en acción, vaya a la carpeta de su proyecto y ejecute el siguiente comando:

```
python manage.py runserver
```

Con esto, se iniciará el servidor de desarrollo de Django y tu proyecto estará disponible en la dirección `http://127.0.0.1:8000/`. Ahora puede comenzar a desarrollar su aplicación Django siguiendo la documentación y los tutoriales disponibles en línea.

Forum with django

Si has llegado hasta aqui, Felicitaciones! ahora continuaremos nuestro proyecto que por nombre tenndra ***THEME FORUM ANON***.

Este será el proyecto con el que aprenderemos mucho más sobre django y podremos publicarlo, desplegarlo y practicar.

Motor de plantillas

En este proyecto usaremos el motor de plantillas de Django se llama Django Template Engine. Es un sistema de plantillas que le permite definir la estructura de la interfaz de usuario de su aplicación de manera separada del lógico de la aplicación. Las plantillas se pueden reutilizar y se pueden combinar fácilmente con datos dinámicos para crear vistas para su aplicación.

Que es un sitio web dinamico?

Un sitio web dinámico es un tipo de sitio web que genera contenido dinámicamente en función de diversos factores como la entrada del usuario,la base de datos. Se crea utilizando lenguajes de programación del lado del servidor en este caso con Python y proporciona una experiencia más interactiva y personalizada para los usuarios.

Estilos

Necesitaremos usar una libreria llama `django-tailwind` para darle estilos con el famosos frameworks de css llamado tailwindcss. De esta foram daremos un estilo propio y modernos neustros sitios web con esta poderosa herramienta.

Antes de emepzar recuerda tener la misma estructura de carpetas mostradas en la pagina [titulopagina](titulo pagina) y activar nuestra virtualizacion ya creada.

- Instalaremos el siguiente paquete para darle estilos a nuestro proyecto con tailwindcss.

```
python -m pip install django-tailwind
```

- En nuestra carpeta App de nuestro proyecto abriremos el fichero `settings.py` y agregamos en la variable global `INSTALLED_APPS` nuestra aplicacion `tailwind`.

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

- Crearemos una aplicacion que por defecto tiene como nombre `theme`. Esto creara la compatibilidad tailwindcss con django. Escribimos en nuestra terminal:

```
python manage.py tailwind init
```

- Agreguamos la aplicacion `theme` recién creada a `INSTALLED_APPS` en `settings.py`:

```
INSTALLED_APPS = [
...,
'tailwind',
```

```
'theme'
]
```

- Registre la aplicación `theme` generada agregando la siguiente línea al archivo `settings.py` ve hacia el final del archivo y escribe lo siguiente:

```
TAILWIND_APP_NAME = 'theme'
```

- Asegúrese de que la lista `INTERNAL_IPS` esté presente en el archivo `settings.py` y contenga la dirección IP `127.0.0.1`. La variable global `INTERNAL_IPS` no existe tienes que crearla, puedes ponerla al final de archivo.

```
# Inter IPS communication tailwind
INTERNAL_IPS = [
    "127.0.0.1",
]
```

- Instalaremos las dependencias de tailwindcss. Escribimos:

```
python manage.py tailwind install
```

- Agregaremos la ruta de nuestro manejador de paquetes `npm` en la variable global `NPM_BIN_PATH` creada por nosotros y agregada el archivo `settings.py`:

  windows

```
NPM_BIN_PATH = "C:\\Program Files\\nodejs\\npm.cmd"
```

  linux

```
NPM_BIN_PATH = '/usr/local/bin/npm'
```

- Registramos una nueva ruta en el fichero `urls.py`.
  - Importamos El objeto `TemplateView`

```
from django.views.generic import TemplateView
```

  - Agregamos nuestra ruta

```
urlpatterns = [
    path('', TemplateView.as_view(template_name="base.html")),     #
new
    # deault rut admin
    path('admin/', admin.site.urls),
]
```

Nuestra App **theme** tiene un archivo llamado `base.html` en la ruta `app/theme/templates/base.html` sera la base de todas nuestras templates.

- Para ejecutar tailwindcss y django tendremos que repetir el proceso mostrado a continuacion en dos terminales distintas ambas activadas virtaulmente python:
  - Terminal uno

    ```
    (env) $ python manage.py runserver
    ```

  - Terminal dos

    ```
    (env) $ python manage.py tailwind start
    ```

Con esto realizado ya podras ver el puesto asignado por defecto en **Django** abre tu navegador pega el puerto y mira tu primera pagina web creada con django y **tailwindcss**.

Creditos a https://github.com/timonweb/django-tailwind

Usando PostgreSQL

Instalaremos Postgresql para Windows, macOS o alguna Linux el instalador oficial esta disponible en su sitio oficial de descargas https://www.enterprisedb.com/downloads/postgres-postgresql-downloads lo descargaremos e instalaremos segun las intrucciones dadas por la propia documentacion.

Por defecto el instalador incluira otros programas adicionales uno de ellos que nos interesa **PgAdmin4**. Segun su sistema operativo pueden buscarlo y abrirlo. Recuerden tener en cuenta la ruta en donde se encuentra instalado y por supuesto en el instalador te habra pedido **user and password** no podemos olvidarnos de las credenciales ya que tendremos que usarla mas adelante.

Una vez finalizada la instalacion abriremos **PgAdmin4**: seguiremos los pasos correspondientes para ejecutar **PostgreSQL**.

`Server>PostgreSQL(Doble Click)>Agregar Credenciales en ventana emergente`

Con esto realizado podremos ver un panel administrativo del modelo de ejemplo de `PostgreSQL`.

Conectar Django a PostgresSQL

Si has notado un error de color destacable rojo cuando ejecutamos **Django**, notaremos que especifica sobre migracion y sin ella no podremos sacarle provecho o no funcionara correctamente django, podemos realizar la migracion y por defecto django usa **sqlite3**, sin embargo no podemos usar este gestor de bases de datos para proyectos mas grandes y por supuesto pensando en la escalabilidad del proyecto o proyectos que queramos llevar a cabo en un futuro, unicamente esta pensado para proyectos de poca escalabilidad o proyectos sencillos.

En nuestro Proyecto Forum usaremos postgreSQL como te has dado cuenta anteriormente. En esta ocacion estaremos conectando **PostgreSQL** con **Django**.

- Una vez descargado **PostgeSQL**. Abriremos pgAdmin accederemos con nuestras credenciales que nos pidieron en el momneot en el que instalamos postgress.
- Desplegamos la el elemento de lista llamado **Server** >**opcionPostgresSQL**>**DataBases** (*Click derecho*) **Create**>**Database...**.
- Asigamos un nombre a nuestra base de datos. En este Caso el nombre sera **Forum_Django_DB**.
- Dejamos todo por defecto ysalvamos nuestra base de datos.

Una vez creada la base de datos en **PostgresSQL** nos dirigimos a nuestro proyecto de Django y realizaremos los siguiente.

- Recodermos que no tenemos el modulo correspondiente para conectarnos directamente a **PostgresSQL**, para ello instalaremos psycopg2:

  ```
  python -m pip install psycopg2
  ```

- Accedemos a `settings.py` cambiaremos los siguientes valores:

  El motor que estaremos usando sera `postgresql_psycopg2`

  ```
  DATABASES = {
  'default': {
      'ENGINE': 'django.db.backends.postgresql_psycopg2',
      'NAME': 'Forum_Django_DB',
      'USER' : 'ADMIN',
      'PASSWORD' : 'admin',
      'HOST' : 'localhost',
      'PORT': '5432',
  }
  }
  ```

Creando las apps de nuestro proyecto

Crearemos las siguientes aplicaciones:

- login
- question
- answer
- register

migracion anadir el tailwind y algunos plugins tambien explicar los que se agregaron por defecto configurar la template base hacer la prueba del admin agregar el administrador