**Paso 1: Instalar Flask**

Primero, necesitas instalar Flask en tu entorno de desarrollo. Puedes hacerlo ejecutando el siguiente comando en tu terminal o símbolo del sistema:

pip install Flask

**Paso 2: Configurar la Estructura del Proyecto**

Crea un directorio para tu proyecto y dentro de él, crea un archivo llamado app.py que será el punto de entrada de tu aplicación Flask. También puedes crear un directorio llamado templates para almacenar las plantillas HTML y un directorio llamado static para almacenar archivos estáticos como CSS, JavaScript e imágenes.

arduino

mi\_proyecto/

|- app.py

|- templates/

|- static/

**Paso 3: Escribir el Código de la Aplicación Flask**

Abre el archivo app.py que acabas de crear y escribe el siguiente código para crear una aplicación web básica con Flask:

python

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Paso 4: Crear una Plantilla HTML**

Dentro del directorio templates, crea un archivo llamado index.html. Este será tu archivo HTML principal que Flask renderizará cuando se acceda a la ruta raíz ("/") de tu aplicación. Puedes escribir HTML y utilizar etiquetas de Jinja2 para agregar contenido dinámico si es necesario.

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My Flask Website</title>

</head>

<body>

<h1>Bienvenido a mi sitio web</h1>

<p>Este es un sitio web básico creado con Flask.</p>

</body>

</html>

**Paso 5: Ejecutar la Aplicación Flask**

En tu terminal o símbolo del sistema, navega hasta el directorio donde creaste tu proyecto (mi\_proyecto) y ejecuta el archivo app.py con Python:

python app.py

Esto iniciará el servidor de desarrollo de Flask y podrás acceder a tu sitio web visitando la dirección http://localhost:5000 en tu navegador web.

¡Y eso es todo! Ahora has creado una página web básica en Python utilizando Flask. Puedes expandir tu aplicación añadiendo más rutas, plantillas HTML, estilos CSS y funcionalidades dinámicas según sea necesario.

me ayudas con ejemplos para Puedes expandir tu aplicación añadiendo más rutas, plantillas HTML, estilos CSS y funcionalidades dinámicas

ChatGPT

¡Por supuesto! Aquí tienes algunos ejemplos para expandir tu aplicación Flask añadiendo más rutas, plantillas HTML, estilos CSS y funcionalidades dinámicas:

**Añadir Más Rutas**

Puedes agregar más rutas a tu aplicación Flask para manejar diferentes URLs y mostrar diferentes páginas. Por ejemplo:

python

@app.route('/about')

def about():

return render\_template('about.html')

@app.route('/contact')

def contact():

return render\_template('contact.html')

Estas nuevas rutas responderán a las URL /about y /contact respectivamente. Puedes crear las plantillas HTML correspondientes (about.html y contact.html) en el directorio templates.

**Plantillas HTML Adicionales**

Crea nuevas plantillas HTML en el directorio templates para las nuevas rutas que has añadido. Por ejemplo, about.html y contact.html:

html

<!-- about.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>About Us</title>

</head>

<body>

<h1>About Us</h1>

<p>This is the about page of our website.</p>

</body>

</html>

html

<!-- contact.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Contact Us</title>

</head>

<body>

<h1>Contact Us</h1>

<p>This is the contact page of our website.</p>

</body>

</html>

**Estilos CSS**

Crea un archivo CSS en el directorio static para agregar estilos a tus plantillas HTML. Por ejemplo, styles.css:

css

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h1 {

color: #333;

}

p {

color: #666;

}

Luego, incluye este archivo CSS en tus plantillas HTML utilizando la etiqueta <link> en el <head> de cada plantilla.

**Funcionalidades Dinámicas**

Puedes añadir funcionalidades dinámicas a tus plantillas HTML utilizando Jinja2. Por ejemplo, pasando variables desde Python a las plantillas:

python

@app.route('/user/<username>')

def user\_profile(username):

return render\_template('profile.html', username=username)

Y luego, en la plantilla profile.html, puedes acceder a la variable username:

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>User Profile</title>

</head>

<body>

<h1>Welcome, {{ username }}</h1>

<p>This is your profile page.</p>

</body>

</html>