🔐 Sesiones e Inputs Ocultos en Flask

Persistencia de Datos y Formularios Avanzados

Python Full Stack - Clase 20

¿Qué aprenderemos hoy?

- 🧠 Entender qué es el "estado" en aplicaciones web
- 💾 Aprender sobre sesiones y persistencia de datos
- 🍪 Conocer cómo funcionan las cookies
- 🔧 Configurar sesiones en Flask
- 📝 Guardar y recuperar datos de sesión
- 🔍 Verificar si existe información en sesión
- 🗑️ Eliminar datos de sesión
- 👁️ Usar inputs ocultos en formularios

🧠 El Problema del Estado

¿Qué es el Estado?

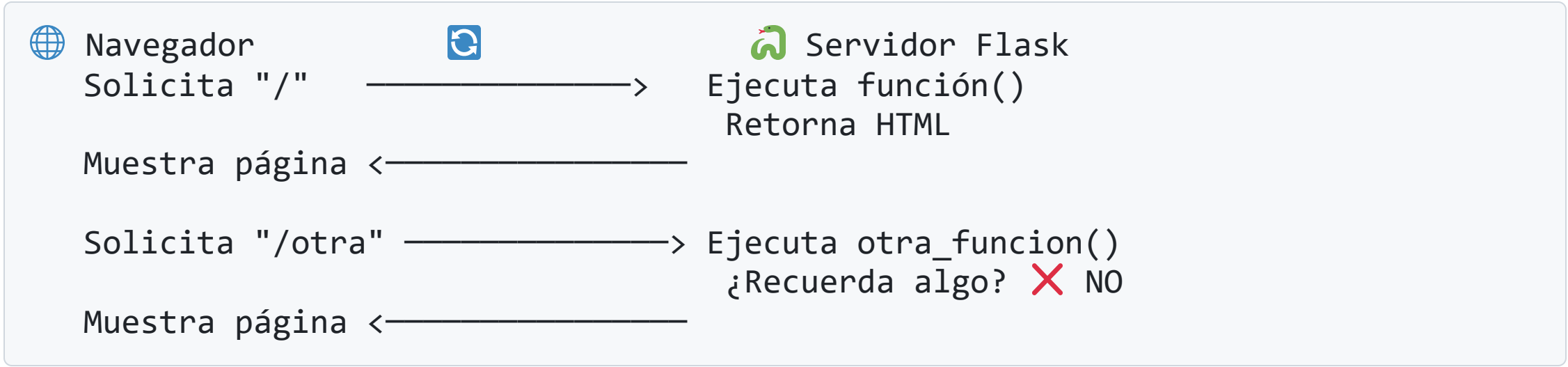
Imagina que estás en una tienda física:

- El vendedor **recuerda** tu nombre
- Sabe qué productos has visto
- Mantiene tu carrito de compras mientras navegas

En una aplicación web, esto es el "estado" 📦

HTTP: Un Protocolo Sin Estado

HTTP es "stateless" (sin estado):



Cada solicitud es independiente 🔄

El Problema Real

¿Recuerdas cuando usamos `request.form`?

```
@app.route('/crear_usuario', methods=['POST'])
def crear_usuario():
    nombre = request.form['nombre']  # ✅ Disponible aquí
    return redirect('/mostrar_usuario')

@app.route('/mostrar_usuario')
def mostrar_usuario():
    # ❌ request.form ya no existe después del redirect
    return render_template("mostrar.html")
```

Después de un redirect, `request.form` desaparece 😱

¿Por Qué Sucede Esto?

Cada función que maneja una ruta es **independiente**:

```
def funcion_a():
    variable = "Hola"
    # variable solo existe aquí

def funcion_b():
    # ❌ No tiene acceso a variable
    print(variable)  # Error!
```

Lo mismo pasa con las rutas de Flask 🔄

La Solución: Persistencia de Datos

Los sitios web usan **almacenamiento persistente**:

- 🛒 Carrito de compras
- 👤 Información de usuario logueado
- 📄 Páginas visitadas
- ⚙️ Preferencias del usuario

Una forma muy común es la SESIÓN 💾

💾 ¿Qué es una Sesión?

Sesión: Una Conversación Persistente

Una **sesión** es como una conversación que el servidor recuerda:

```
👤 Usuario                    Servidor
   "Hola, soy Juan" ──> Guarda: nombre = "Juan"

   "Muéstrame mi perfil" ──> Recuerda: nombre = "Juan"
                               Muestra: "Hola Juan"

   "Agrega al carrito" ──> Recuerda: nombre = "Juan"
                           Guarda: carrito = [...]
```

El servidor "recuerda" información entre solicitudes 🧠

¿Qué Podemos Guardar en Sesión?

- ✅ Si un usuario inició sesión
- ✅ Información del usuario (nombre, email, etc.)
- ✅ Las páginas que ha visitado
- ✅ Carrito de compras
- ✅ Preferencias temporales

¡Cualquier dato que necesites recordar! 📝

Sesión vs Variables Normales

Variables Normales	Sesión
Solo existen en una función	Persisten entre solicitudes
Se pierden al terminar la función	Se mantienen hasta cerrar sesión
No se comparten entre rutas	Accesibles desde cualquier ruta

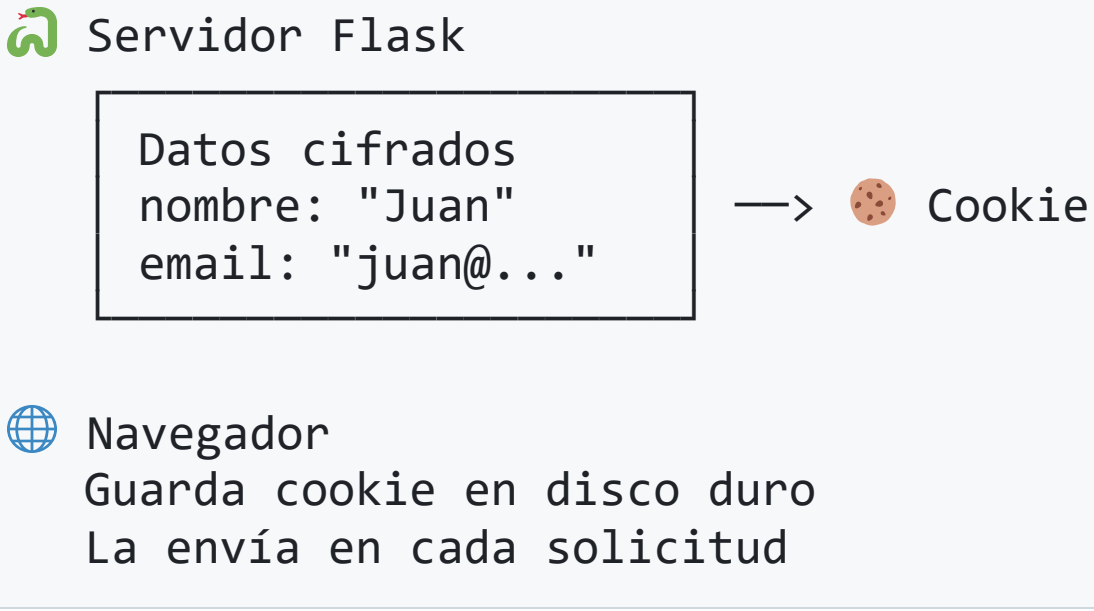
Sesión = Memoria compartida entre todas tus rutas 🧠

 Cookies: El Mecanismo Detrás de las Sesiones

¿Qué son las Cookies?

No hablamos de galletas 🍪

Las **cookies** son pequeños fragmentos de datos que se almacenan en el navegador:



Cómo Funcionan las Cookies

1. **Servidor** envía datos cifrados al navegador
2. **Navegador** guarda la cookie en el disco duro
3. **Navegador** envía la cookie en cada solicitud
4. **Servidor** descifra y recupera los datos

Flask usa cookies para almacenar sesiones 🔐

⚠️ Seguridad de las Cookies

Las cookies están cifradas, pero NO son totalmente seguras:

- ✅ Puedes guardar: nombre, email, preferencias
- ❌ NO guardes: contraseñas, números de tarjeta, datos sensibles

Regla de oro: Solo guarda información que no sea crítica si alguien la ve 🔒

🔧 Configurando Sesiones en Flask

Paso 1: Importar Session

Primero, necesitamos importar `session`:

```
from flask import Flask, render_template, request, redirect, session
#                                                          ^^^^^^^
#                                                          Agregamos session
```

`session` es un diccionario especial que persiste entre solicitudes 📦

Paso 2: Configurar Clave Secreta

Flask necesita una **clave secreta** para cifrar las cookies:

```
app = Flask(__name__)

# Establecemos una clave secreta para cifrar las cookies
app.secret_key = 'Esta es tu clave secreta!'
```

Sin esta clave, las sesiones no funcionarán 🔑

⚠️ Importante: Clave Secreta

En producción, usa una clave segura:

```
import os

app.secret_key = os.environ.get('SECRET_KEY', 'clave-temporal-desarrollo')
```

Por ahora, usaremos una clave simple para aprender 💡

Paso 3: Guardar Datos en Sesión

Guardamos datos como en un diccionario:

```
@app.route('/crear_usuario', methods=['POST'])
def crear_usuario():
    # Guardamos datos en sesión
    session['nombre_usuario'] = request.form['nombre']
    session['email_usuario'] = request.form['email']

    return redirect('/mostrar_usuario')
```

`session` funciona como un diccionario 📝

Paso 4: Leer Datos de Sesión

Podemos leer los datos desde cualquier ruta:

```
@app.route('/mostrar_usuario')
def mostrar_usuario():
    # Leemos datos de sesión
    nombre = session['nombre_usuario']
    email = session['email_usuario']

    print(f"Usuario: {nombre}, Email: {email}")

    return render_template("mostrar.html")
```

Los datos persisten después del redirect ✅

Usando Sesión en Plantillas

También podemos acceder a sesión desde las plantillas:

```
<!-- templates/mostrar.html -->
<h1>Usuario:</h1>
<h3>Nombre: {{ session['nombre_usuario'] }}</h3>
<h3>Email: {{ session['email_usuario'] }}</h3>
```

Jinja2 tiene acceso directo a `session` 🎨

💪 Ejercicio: Contador de Visitas

Crea un proyecto Flask llamado `visitas` con:

1. Una ruta raíz (`/`) que muestre cuántas veces has visitado la página
2. Incrementa el contador cada vez que alguien visita la ruta
3. Guarda el contador en sesión

Pistas:

- Usa `session['visitas']` para guardar el número
- Verifica si existe con `if 'visitas' in session:`
- Si no existe, inicialízalo en 0

⏱️ **Tiempo:** 10 minutos

Verificando si Existe una Sesión

Antes de leer una sesión, debemos verificar si existe:

```
if 'nombre_propiedad' in session:
    # La propiedad existe, podemos usarla
    valor = session['nombre_propiedad']
else:
    # La propiedad NO existe, inicializamos
    session['nombre_propiedad'] = 0
```

Evita errores verificando primero ✅

Ejemplo Completo: Contador de Visitas

```python
@app.route('/')
def index():
    # Verificamos si existe el contador
    if 'visitas' in session:
        # Si existe, lo incrementamos
        session['visitas'] += 1
    else:
        # Si no existe, lo inicializamos en 1
        session['visitas'] = 1

    return f"Has visitado esta página {session['visitas']} veces"
```

¡Ahora el contador persiste entre recargas! 🎉

Eliminando Datos de Sesión

Hay dos formas de eliminar datos:

```
# Eliminar UNA propiedad específica
session.pop('nombre_propiedad')

# Eliminar TODAS las propiedades
session.clear()
```

Útil para cerrar sesión o reiniciar 🗑️

Ejemplo: Destruir Sesión

```python
@app.route('/destruir_sesion')
def destruir_sesion():
    # Eliminamos todas las propiedades de sesión
    session.clear()

    # Redirigimos a la ruta principal
    return redirect('/')
```

Después de esto, la sesión está vacía 🧹

💪 Ejercicio: Contador Mejorado

¡Hora de practicar!

Mejora tu contador de visitas:

1. Agrega un botón que incremente las visitas en +2
2. Agrega un botón para reiniciar el contador a 0
3. Crea una ruta `/destruir_sesion` que elimine la sesión

Pistas:

- Usa formularios con `method="POST"`
- Crea rutas separadas para cada acción
- Usa `session.clear()` para eliminar todo

⏱️ **Tiempo:** 15 minutos

¿Qué son los Inputs Ocultos?

Los **inputs ocultos** son campos de formulario que:

- ✅ Se envían al servidor
- ❌ NO son visibles para el usuario
- 🔍 No aparecen en la pantalla

Útiles para enviar información que el usuario no necesita ver 👁️🗨️

Sintaxis de Input Oculto

```
<input type="hidden" name="nombre_input" value="valor">
```

Atributos importantes:

- `type="hidden"` → Lo hace invisible
- `name` → Nombre del campo (obligatorio)
- `value` → Valor que se enviará

Ejemplo: Dos Formularios Diferentes

```
<!-- Formulario para administradores -->
<form action="/iniciar_sesion" method="POST">
    <input type="text" name="email">
    <input type="password" name="password">
    <input type="hidden" name="tipo_usuario" value="administrador">
    <input type="submit" value="Iniciar Sesión">
</form>

<!-- Formulario para usuarios comunes -->
<form action="/iniciar_sesion" method="POST">
    <input type="text" name="email">
    <input type="password" name="password">
    <input type="hidden" name="tipo_usuario" value="usuario">
    <input type="submit" value="Iniciar Sesión">
</form>
```

Ambos van a la misma ruta, pero con información diferente 🔄

Procesando Inputs Ocultos en el Servidor

```
@app.route('/iniciar_sesion', methods=['POST'])
def iniciar_sesion():
    tipo_usuario = request.form['tipo_usuario']

    if tipo_usuario == 'administrador':
        # Proceso para administrador
        return redirect('/admin')
    elif tipo_usuario == 'usuario':
        # Proceso para usuario común
        return redirect('/dashboard')
```

El servidor puede distinguir qué formulario se envió 🎯

⚠️ Seguridad: Inputs Ocultos NO son Seguros

Aunque no son visibles, los usuarios pueden:

1. Ver el código fuente de la página
2. Inspeccionar el HTML con herramientas de desarrollador
3. Modificar el valor antes de enviar

Ejemplo:

```
<!-- Usuario puede cambiar esto -->
<input type="hidden" name="tipo_usuario" value="administrador">
```

Nunca confíes en inputs ocultos para seguridad 🔒

Cuándo Usar Inputs Ocultos

✅ **Buenos usos:**

- Identificar qué formulario se envió
- Enviar IDs de elementos
- Pasar datos temporales entre páginas
- Identificar acciones (editar vs crear)

❌ **NO uses para:**

- Contraseñas
- Permisos de usuario
- Validaciones de seguridad
- Datos críticos

Ejemplo Práctico: Múltiples Botones

```
<!-- Botón para agregar 2 visitas -->
<form action="/incrementar" method="POST">
    <input type="hidden" name="cantidad" value="2">
    <input type="submit" value="+2 Visitas">
</form>

<!-- Botón para agregar 5 visitas -->
<form action="/incrementar" method="POST">
    <input type="hidden" name="cantidad" value="5">
    <input type="submit" value="+5 Visitas">
</form>
```

Mismo formulario, diferentes valores 🔄

Procesando en el Servidor

```python
@app.route('/incrementar', methods=['POST'])
def incrementar():
    cantidad = int(request.form['cantidad'])

    if 'visitas' in session:
        session['visitas'] += cantidad
    else:
        session['visitas'] = cantidad

    return redirect('/')
```

Una sola ruta maneja ambas acciones 🎯

Renderizando HTML desde Python

A veces necesitas mostrar HTML que viene de Python:

```
@app.route('/')
def index():
    mensaje = "<ul><li>Hola</li></ul>"
    return render_template("index.html", mensaje=mensaje)
```

En la plantilla:

```
{{ mensaje }}  <!-- Muestra el texto literal -->
```

Jinja2 escapa el HTML por seguridad 🔒

Usando `|safe` para Renderizar HTML

Si quieres que Jinja2 renderice el HTML:

```
{{ mensaje|safe }}
```

Ahora el HTML se renderiza correctamente ✅

⚠️ Solo usa `|safe` con contenido confiable 🔒

Resumen de Conceptos Clave

Lo Más Importante

1. **HTTP es sin estado** → Cada solicitud es independiente
2. **Sesiones** → Permiten recordar información entre solicitudes
3. **Cookies** → Mecanismo que Flask usa para sesiones
4. **Clave secreta** → Necesaria para cifrar las cookies
5. **Inputs ocultos** → Envían datos invisibles al usuario
6. **Verificar sesión** → Siempre verifica si existe antes de usar

Conceptos Clave

Concepto	Descripción
Estado	Información que persiste entre solicitudes
Sesión	Diccionario especial que persiste datos
Cookie	Fragmento de datos almacenado en el navegador
Input oculto	Campo de formulario invisible pero funcional
`session.clear()`	Elimina todas las propiedades de sesión
`session.pop()`	Elimina una propiedad específica

Recursos para Seguir Aprendiendo

- 📖 Documentación oficial de Flask - Sessions
- 📖 Documentación de Jinja2
- 🎓 Flask Tutorial - Cookies and Sessions
- 💡 Inputs ocultos en HTML

Próximos Pasos

En la siguiente clase aprenderemos:

- 🔐 Autenticación y autorización
- 👤 Login y registro de usuarios
- 🔓 Protección de rutas
- 💾 Bases de datos con Flask

¡Sigue practicando con sesiones! 💪

Práctica para Casa

Proyecto sugerido: Crea una aplicación de "Lista de Tareas" donde:

1. Los usuarios pueden agregar tareas
2. Las tareas se guardan en sesión
3. Pueden marcar tareas como completadas
4. Pueden eliminar tareas
5. Muestra el total de tareas completadas

¡Usa todo lo que aprendiste hoy! 🎯

Consejos Finales

✅ **Buenas prácticas:**

- Siempre verifica si existe una propiedad antes de usarla
- No guardes información sensible en sesión
- Usa nombres descriptivos para propiedades de sesión
- Limpia la sesión cuando sea necesario

❌ **Errores comunes:**

- Olvidar configurar `app.secret_key`
- Intentar leer sesión sin verificar si existe
- Guardar demasiada información en sesión
- Confiar en inputs ocultos para seguridad

🎉 ¡Felicidades!

Ya sabes trabajar con sesiones e inputs ocultos

¡Has aprendido a hacer que tus aplicaciones recuerden información!

? Preguntas

¿Alguna duda sobre sesiones o inputs ocultos?

🙌 ¡Excelente Trabajo!

¡Nos vemos en la siguiente clase con autenticación y login!

No olvides completar todos los ejercicios 📝