



El futuro digital
es de todos

MinTIC



CICLO III: Desarrollo de software

Mision
TIC2022





El futuro digital
es de todos

MinTIC

UN UNIVERSIDAD
DEL NORTE

Vigilada Mineducación

Sesión 9: Desarrollo Software

Comunicación entre Front-end y Back-end

Mision
TIC2022



Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Intercambio de datos entre el servidor y el cliente.
2. Manejo de formularios.



Desarrollo Web Front y Back: Flask - Rutas

- Las rutas constituyen uno de los conceptos más importantes en las aplicaciones que se realizan para las diferentes páginas.
- En Flask, existe la posibilidad de definir rutas amigables de manera muy sencilla, sin tener que utilizar las expresiones regulares.
- El objeto Flask app suministra un decorador router que es capaz de filtrar la función que se va ejecutar analizando la petición HTTP, principalmente por la ruta y el método que se hace la petición.



Desarrollo Web Front y Back: Flask - Rutas

```
@app.route("/")
```

```
def inicio():
```

```
    return "Página principal"
```



Asignando a la ruta root del sistema una función que muestra o lleva a «Página principal»

```
@app.route("/productos/")
```

```
def productos():
```

```
    return "Listado de productos"
```



Asignando a la ruta específica productos una función que muestra o lleva a «Listado de productos»

```
@app.route("/contactanos")
```

```
def contactanos():
```

```
    return "Página contactanos..."
```



Asignando a la ruta específica contactanos una función que muestra o lleva a «Página contactános»



Desarrollo Web

Front y Back: Flask - Rutas

Barra final.

- En los ejemplos anteriores, la diferencia entre las dos últimas URLs o rutas específicas (productos y contactanos), es que en el caso de productos, al estar definida con el slash («/») al final, hace que cuando se acceda a dicha url sin el slash, automáticamente Flask redirecciona a «/productos/», con la barra incluida.
- En el caso de la URL «/contactanos», si se accedes a «/nosotros/» se generará un error 404.



Desarrollo Web Front y Back: Flask - Rutas

Tipos de datos.

- **string**: para cualquier tipo de string, sin barras («/»).
- **int**: Acepta números enteros.
- **float**: Acepta números con decimales.
- **path**: Igual a string, pero también acepta barras («/»).
- **any**: Intenta encontrar alguno de los items dados.
- **uuid**: Acepta strings UUID.



Desarrollo Web Front y Back: Flask - Rutas

Rutas dinámicas (con parámetro variable o ruta variable)

```
@app.route('/productos/<int:id>')  
def mostrar_producto(id):  
    return "Mostrar el artículo con id:{}".format(id)
```

Como se observa, en este caso se le envía el parámetro por la URL y se muestra qué es lo que llega con un return.



Desarrollo Web Front y Back: Flask - Rutas

Ejemplo de ruta dinámica:

```
@app.route('/hola/')
@app.route('/hola/<string:nombre>')
@app.route('/hola/<string:nombre>/<int:edad>')
def hola(nombre=None, edad=None):
    if nombre and edad:
        return "Hola, {0} tienes {1} años.".format(nombre, edad)
    elif nombre:
        return "Hola, {0}".format(nombre)
    else:
        return "Hola mundo"
```



Desarrollo Web

Front y Back: Flask - Métodos HTTP

Para acceder a las distintas URLs se pueden utilizar varios métodos en la petición HTTP. Entre los métodos que normalmente se pueden utilizar en un navegador web están los métodos GET y POST.

- **GET:** Se realiza una petición para obtener un recurso del servidor web. Es el método más utilizado.
- **POST:** Aunque con el método GET se puede enviar información al servidor (por medio de parámetros escritos en la URL), se usa el método POST para enviar información a una determinada URL. Normalmente se utilizan los formularios HTML para enviar información al servidor por medio del método POST.



Desarrollo Web

Front y Back: Flask - Métodos HTTP

Por defecto las rutas indicadas en la funciones route sólo son accesibles utilizando el método GET. Si una URL recibe información por medio del método POST y no se desea que se acceda a ella con un método GET, se definirá de la siguiente forma:

```
@app.route('/productos/new', methods=["POST"])  
def productos_new():  
    return 'URL recibe información de un formulario con el método POST'
```



Desarrollo Web

Front y Back: Flask - Métodos HTTP

Es posible acceder a una URL con los dos métodos, de tal forma que se realice una serie de instrucciones cuando se acceda con GET y otra cuando se acceda con POST. Ejemplo:

```
@app.route('/login', methods=['GET', 'POST'])  
def login():  
    if request.method == 'POST':  
        return 'Entra con POST'  
    else:  
        return 'Entra con GET'
```



Desarrollo Web

Front y Back: Flask – Rendering y Template

Renderizar y utilizar plantillas en Flask permite presentar un cuerpo HTML que se encuentre en un archivo externo al del script Python.

En Flask, para mostrar un encabezado sencillo se hace lo siguiente :

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hola_mundo():
    return "<h1>Hola Mundo</h1>"
```



Desarrollo Web

Front y Back: Flask – Rendering y Template

Al utilizar las etiquetas HTML, como las mostradas en slide anterior, se puede observar lo engorroso que se vuelve el código. Es por esta razón que flask permite utilizar plantillas y renderizar.

Para renderizar plantillas, se utilizará el método `render_template()` el cual recibe como argumento el nombre de la plantilla. Por Ejemplo:

Para una mejor experiencia, lo primero será ordenar los archivos. Crear una carpeta `templates` para guardar todas las plantillas que se deseen utilizar utilizar. En este caso se creará un archivo HTML (`index.html`).

```
/app_flask_templates.py  
/templates  
  /index.html
```



Desarrollo Web

Front y Back: Flask – Rendering y Template

Crear una plantilla muy sencilla:

```
<!DOCTYPE html>
<html>
<head>
  <title>Usando Plantillas en Flask</title>
</head>
<body>

  <strong>
    Hola Mundo!!!
  </strong>

</body>
</html>
```



Desarrollo Web

Front y Back: Flask – Rendering y Template

Ahora a escribir el código en el archivo `app_flask_templates.py`, para poder renderizar esta plantilla:

```
# Importar "render_template"
from flask import Flask, render_template

# Crear instancia de Flask
app = Flask(__name__)

# Definir el route
@app.route('/')
def render():
    # Retornar la plantilla "index.html"
    return render_template("index.html")
```




Desarrollo Web

Backend Flask - Rendering y Template

Para enviar datos a través de la URL de la página, las palabras claves, o parámetros , se debe colocar un segundo argumento al método `render_template()`. Ejemplo:

```
# Importar "render_template"
from flask import Flask, render_template

# Crear la instancia de Flask
app = Flask(__name__)

# Definir el route
@app.route('/')
# Un segundo route con el nombre del parámetro
@app.route('/<nombre>')
def render(nombre=None): # Inicializa "nombre"
    # Retornar plantilla "index.html" y pasar parámetro a
    # el método render_template
    return render_template("index.html", nombre=nombre)
```



Desarrollo Web

Front y Back: Flask – Rendering y Template

Cambiar el archivo HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Usando Plantillas en Flask</title>
</head>
<body>

  <strong>
    Hola {{nombre}}
  </strong>

</body>
</html>
```

Se agrega {{ }} y dentro de estos, el nombre del parámetro (en este caso nombre) especificado en el código anterior.



El futuro digital
es de todos

MinTIC



Vigilada Mineducación

Ejercicios de práctica





Seguimiento Habilidades Digitales en Programación

* De modo general, ¿Cuál es grado de satisfacción con los siguientes aspectos?

	Nada Satisfecho	Un poco satisfecho	Neutra	Muy satisfecho	Totalmente satisfecho
Sesiones técnicas sincrónicas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sesiones técnicas asincrónicas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sesiones de inglés	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Apoyo recibido	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Material de apoyo: diapositivas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Material de apoyo: ejercicios prácticos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Completa la siguiente encuesta para darnos
retroalimentación sobre esta semana ▼▼▼

<https://www.questionpro.com/t/ALw8TZIxOJ>



El futuro digital
es de todos

MinTIC

UN UNIVERSIDAD
DEL NORTE

Vigilada Mineducación

¡GRACIAS
POR SER PARTE DE
ESTA EXPERIENCIA
DE APRENDIZAJE!



Misión
TIC 2022