El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

# CICLO III:

## Desarrollo de software

MisionTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

# Sesión 11:

# Desarrollo Software

## Comunicación entre Front-end y Back-end

MisiónTIC2022

# Objetivos de la sesión

## Al finalizar esta sesión estarás en capacidad de:

1. Manejo de formularios.
2. Intercambio de datos entre el servidor y el cliente.
3. El concepto de API REST para desacoplar el front del back de una aplicación WEB.

# Desarrollo Web
# Front y Back:  Arquitectura Desacoplada

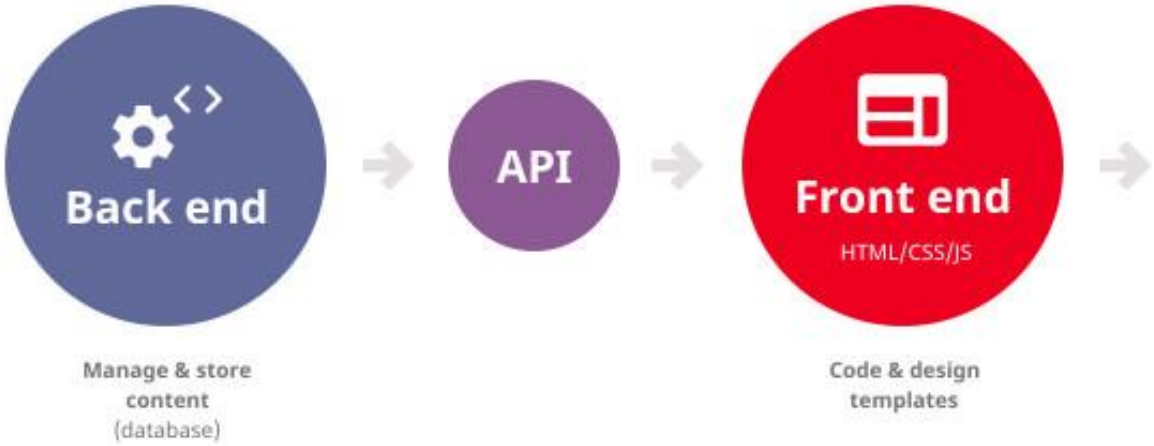
- La arquitectura desacoplada separa, o desacopla, el back-end y el front-end de un sitio web en dos aplicaciones diferentes: la primera para la creación y almacenamiento de contenido, y la segunda (pueden ser varias), es la responsable de procesar esos datos y mostrarlos al usuario por medio de alguna interfaz.
- En un CMS (Sistema de gestión de contenidos) desacoplado, tanto el backend como el frontend se separan. Una vez que el contenido se crea y edita en el back-end, el front-end utiliza los servicios web y API (Interfaz de programación de aplicaciones) flexibles y rápidos para proporcionar el contenido base a cualquier diseño de front-end utilizando cualquier dispositivo o canal.

# Desarrollo Web
# Front y Back:  Arquitectura Desacoplada

Una plataforma CMS desacoplada se compone de:

- Base de Datos: en esta se almacenan contenidos y recursos digitales (back-end).
- Back-end de gestión de contenido: aquí se crea contenido (back-end).
- API que permita conectar el back-end de gestión de contenido con el front-end.
- Finalmente, front-end para realizar publicación de contenido predeterminado.

# Desarrollo Web
# Front y Back:  Arquitectura Desacoplada



# Desarrollo Web
# Front y Back:  Arquitectura Desacoplada

**Beneficios**:

- **Gestión de la experiencia:** En la arquitectura tradicional, se necesita evaluar y estudiar cómo realizar cambios en la apariencia, diseño o mensaje que se desea transmitir por medio de la experiencia del usuario, a través del sitio web, ya que factores como la falta de recursos (monetario o humano) y tiempo influyen directamente en ellos. Sin embargo, con la arquitectura desacoplada, se minimizan estas limitaciones, debido a que los cambios se realizarán solo en el front-end.

- **Flexibilidad:** El re-diseño del sitio web se vuelve complejo con respecto al tiempo y recursos en una arquitectura tradicional; sin embargo, cuando el sitio web está construido con una arquitectura desacoplada, existe una mayor flexibilidad para actualizar la tecnología del back-end o para actualizar el diseño y crear nuevas experiencias de forma independiente la una de la otra.

# Desarrollo Web
# Front y Back:  Arquitectura Desacoplada

**Beneficios**:

- **Eficiencia en la publicación de contenidos:** Una arquitectura desacoplada utiliza los servicios web del núcleo para suministrar datos para el front-end, de esta forma envía el contenido a otros lugares de una manera más fácil. Siendo esto de gran importancia, ya que una vez que se publica contenido en un sitio, el mismo, puede estar disponible para su uso en muchos lugares diferentes, inclusive en aplicaciones móviles, dispositivos de IOT, etc.
- **Inteligencia de Negocios:** Se mantiene intacta, es decir, no se verá afectada por los cambios que se realicen en el sitio web. Esto se debe a que toda esta inteligencia está desarrollada en el back-end de la aplicación, lo cual permite realizar muchos cambios y pruebas sin afectar en lo más mínimo la lógica de negocios.

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

**Para recordar**:

- **API** (Application Programming Interface o Interfaz de Programación de Aplicaciones), es una colección de reglas, códigos y especificaciones que las aplicaciones deben seguir para que otro software las use y se comuniquen entre ellos.

- **JSON**, formato de texto sencillo para el intercambio de datos. Subconjunto de la notación literal de objetos de JavaScript.

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

**Por tener en cuenta**:

- **REST** (Representational State Transfer o Transferencia de Estado Representacional) se refiere a un estilo arquitectónico y enfoque de comunicaciones utilizado en el desarrollo de servicios web.
- REST es cualquier interfaz entre sistemas que utilicen HTTP para el transporte de datos usando los métodos de este protocolo para comunicarse (GET, POST, PUT, PATCH y DELETE).
- REST Acepta múltiples formatos de datos: XML, JSON, datos binarios y texto plano.

Mision TIC 2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

## Características de REST

- Protocolo cliente/servidor sin estado.
- Las operaciones más importantes entre los datos de cualquier aplicación REST y el protocolo HTTP son: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- Los objetos en REST siempre se manipulan a partir de la URL.
- Interfaz uniforme.
- Sistema de capas.
- Uso de hipermedios.

MisionTIC2022

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

## Ventajas del desarrollo con REST

- **Separación entre el cliente y el servidor:** REST separa la interfaz de usuario del servidor y el almacenamiento de datos.
- **Visibilidad, fiabilidad y escalabilidad:** La separación entre cliente y servidor contribuye a que cualquier equipo de desarrollo puede escalar el producto sin tantos problemas.
- **La API REST siempre es independiente del tipo de plataformas o lenguajes:** Siempre se adapta a la sintaxis o plataformas con las que se estén desarrollando, lo que implica una gran libertad a la hora de cambiar o probar nuevos entornos dentro de la aplicación.

# Desarrollo Web
# Front y Back: Flask – APIs RESTful

**Pasos para crear una API REST sencilla**:

1. Crear dos archivos: app.py (será el backend o REST API) y articulos.py (Datos), por lo general al trabajar con un REST API se utiliza un BD, en este caso se hará la simulación de esta con los datos en el archivo producto.
2. En el archivo articulos.py, escribir lo siguiente:

```
articulos=[
        {“nombre”:”mouse“, “precio:”25000“, “cantidad”:”5“},
        {“nombre”:”teclado“, “precio:”45000“, “cantidad”:”3“},
        {“nombre”:”usb“, “precio:”35000“, “cantidad”:”7“},
]
```

De esta forma ya están listos los datos de prueba.

# Desarrollo Web
# Front y Back: Flask – APIs RESTful

3. Crear la aplicación del servidor, para esto escribir en archivo app.py lo siguiente:

```
from flask import Flask, jsonify

app = Flask(__name__)

from productos import productos

@app.route('/api')
def api():
    return jsonify({"message":"Hola Mundo!"})

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

→ jsonify convierte un objeto a un json típico del navegador

Ejecutar en el navegador para observar el resultado.

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

4. Ahora, agregar ruta articulos:

```
@app.route('/articulos')
def getArticulos():
    return jsonify({"articulos":articulos, "message":"Listado de artículos"})
```

Ejecutar en el navegador para observar el resultado.

# Desarrollo Web
# Front y Back:  Flask – APIs RESTful

5. Hasta el momento se ha mostrado todo el listado de productos, a continuación se presentará agregará otra ruta que permita mostrar un producto especificando el nombre:

```
@app.route('/articulos/<string:nom_articulo>')
def getArticulo(nom_articulo):
   encontrado = [articulo for articulo in articulos if     articulo['nombre']==
nom_articulo]
    return jsonify({"articulo": encontrado[0]})
```

Ejecutar en el navegador para observar  el resultado.

# Desarrollo Web
# Front y Back: Flask – APIs RESTful

6. Ahora, que pasaría si el artículo no se encuentra, se procede a realizar la validación agregando el un if antes de retornar el resultado, quedando el código de la siguiente forma:

```
@app.route('/articulos/<string:nom_articulo>')
def getArticulo(nom_articulo):
    encontrado = [articulo for articulo in articulos if
articulo['nombre']== nom_articulo]
    if (len(encontrado) > 0):
        return jsonify({"articulo": encontrado[0]})
    return jsonify({"message":"Articulo no encontrado"})
```

En este if, se verifica si la longitud del objeto encontrado es mayor que 0

Ejecutar en el navegador para observar el resultado.

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

# Ejercicios de práctica

Mision TIC 2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

# ¡GRACIAS

## POR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!


Mision TIC 2022