Implementación de XSLT en una Aplicación Web para Visualización de Estaciones de Carga EV

Tópicos de Ciencias de la Computación III

Andre Pacheco | Walter Rivera | Sergio Pezo | Gustavo Delgado | Gabriel Barrientos

Universidad Nacional de Ingeniería

30 de junio de 2025

Contenido

Contenido

¿De qué trata esta presentación?

En esta presentación explicaremos cómo se desarrolló una aplicación web que muestra estaciones de carga para autos eléctricos, usando una tecnología llamada XSLT. Iremos paso a paso, desde lo más básico hasta cómo se integra todo en el proyecto.

¿Qué es XSLT?

- XSLT (eXtensible Stylesheet Language Transformations) es un lenguaje que sirve para transformar datos en formato XML a otros formatos, como HTML.
- Es muy útil cuando queremos mostrar datos estructurados en una página web.
- Permite separar los datos de la presentación visual.
- En nuestra aplicación, lo usamos para convertir datos de estaciones de carga en una interfaz web completa y dinámica.

Contenido

¿Qué problema resuelve nuestro proyecto?

Como problema de ejemplo el profesor del curso asignó un archivo xml que incluye la lista de las direcciones de estaciones de carga de vehículos eléctricos. El objetivo es crear una aplicación web que permita a los usuarios ver y filtrar estas estaciones de manera fácil y rápida usando XSLT para transformar los datos XML en una página web interactiva. Los objetivos específicos son:

- Acceso a datos en tiempo real de estaciones en Connecticut
- Interfaz web interactiva para filtrar estaciones
- Transformación de datos XML a un formato visual atractivo usando XSLT

¿Qué hace la aplicación?

- Extrae datos XML de una API pública de estaciones de carga en Connecticut
- Aplica transformaciones XSLT para generar HTML dinámico
- Permite filtrar estaciones por nombre, dirección, ciudad, tipo de cargador, etc.
- Ofrece una búsqueda en tiempo real con AJAX
- Implementa ordenamiento de resultados por columnas
- Presenta la información con una interfaz moderna y receptiva

Vista de la aplicación

La aplicación muestra datos de estaciones con filtros avanzados y estilizado usando XSLT + CSS

Contenido

¿Qué veremos en esta sección?

Explicaremos los conceptos básicos de XSLT y cómo se usa para transformar datos XML en páginas web.

¿Cómo funciona XSLT?

- Recibe datos en formato XML (en nuestro caso, de la API de estaciones de carga)
- Usa una hoja de estilo (archivo `estilos.xslt`) para definir cómo se verán esos datos
- El resultado es una página HTML lista para mostrar al usuario
- En nuestra aplicación, Flask maneja la obtención y transformación del XML

```
# Como se procesa en app.py (version conceptual)
xml_data = obtener_xml_data()  # Obtiene XML de la API
xslt_doc = etree.parse('estilos.xslt')
transform = etree.XSLT(xslt_doc)
html_resultado = transform(xml_data)  # XML transformado en
    HTML
```

Estructura básica de nuestro archivo XSLT

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org
       /1999/XSL/Transform">
3    <xsl:output method="html" indent="yes"/>
4    <xsl:template match="/">
5      <html>
6        <head>
7          <!-- Metadatos, estilos CSS y titulo -->
8        </head>
9        <body>
10         <!-- Estructura HTML generada dinamicamente -->
11       </body>
12     </html>
13   </xsl:template>
14 </xsl:stylesheet>
```

Características clave de nuestro XSLT

Nuestro archivo `estilos.xslt` incluye:

- CSS embebido para estilizar completamente la aplicación
- JavaScript para manejo de interacciones del usuario
- Transformación de datos XML a elementos HTML
- Plantillas para componentes de interfaz reutilizables
- Lógica condicional para manejar casos especiales

```
<!-- Ejemplo de estilizado en estilos.xslt -->
<style>
  .station-name {
    font-weight: 600;
    color: #4c51bf;
    display: flex;
    align-items: center;
    gap: 8px;
  }
</style>
```

Contenido

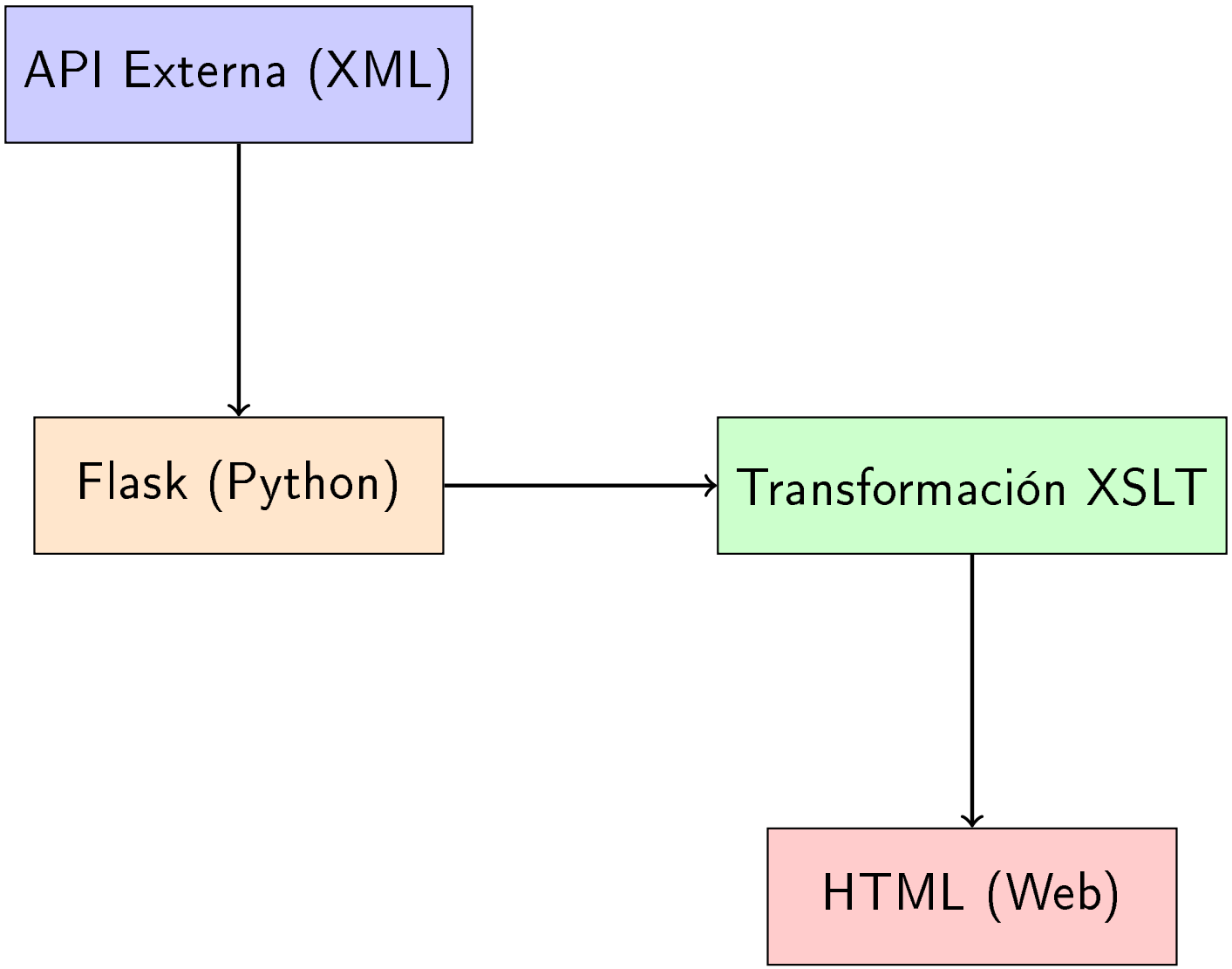
¿Qué veremos en esta sección?

Mostraremos cómo se conectan las diferentes partes del proyecto: la fuente de datos, el backend, la transformación XSLT y el frontend.

Componentes principales

- **API Externa**: Proporciona los datos en XML.
- **Flask (Python)**: Recibe los datos y aplica la transformación XSLT.
- **XSLT**: Convierte los datos XML en HTML.
- **Navegador**: Muestra la página web al usuario.

Diagrama de arquitectura



Datos no formateados

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<response>
  ▼<row>
    ▼<row _id="row-9abt-u2cg.ibnh" _uuid="00000000-0000-0000-47BC-84BB080A3B73" _position="0"
     _address="https://data.ct.gov/resource/jfhb-ebu6/row-9abt-u2cg.ibnh">
       <station_name>BMW OF DARIEN</station_name>
       <street_address>138-142 Ledge Rd</street_address>
       <city>Darien</city>
       <access_days_time>24 hours daily</access_days_time>
       <ev_level1_evse_num>NONE</ev_level1_evse_num>
       <ev_level2_evse_num>2</ev_level2_evse_num>
       <ev_dc_fast_count>NONE</ev_dc_fast_count>
       <ev_other_info>NONE</ev_other_info>
       <geocoded_column>POINT (-73.4764687 41.072882)</geocoded_column>
     </row>
    ▼<row _id="row-tg6x~7upx_ceuz" _uuid="00000000-0000-0000-6386-C23B23427E39" _position="0"
     _address="https://data.ct.gov/resource/jfhb-ebu6/row-tg6x~7upx_ceuz">
       <station_name>Dunkin' - Tesla Supercharger</station_name>
       <street_address>893 E Main St</street_address>
       <city>Meriden</city>
       <access_days_time>24 hours daily; for Tesla use only</access_days_time>
       <ev_level1_evse_num>NONE</ev_level1_evse_num>
       <ev_level2_evse_num>NONE</ev_level2_evse_num>
       <ev_dc_fast_count>8</ev_dc_fast_count>
       <ev_other_info>NONE</ev_other_info>
       <geocoded_column>POINT (-72.773473 41.527367)</geocoded_column>
     </row>
```

Contenido

¿Qué veremos en esta sección?

Veremos ejemplos concretos de cómo XSLT transforma los datos y cómo se integra con el backend en Python.

Ejemplo: Extracción de datos con XSLT

```
<td class="station-name">
  <xsl:value-of select="station_name"/>
</td>
```

Esto toma el nombre de la estación desde el XML y lo muestra en la tabla.

Ejemplo: Iteración sobre elementos

```
<xsl:for-each select="//row/row">
  <tr>
    <!-- Celdas de la tabla -->
  </tr>
</xsl:for-each>
```

Esto permite mostrar muchas estaciones, una por cada fila del XML.

Ejemplo: Lógica condicional

```
<xsl:choose>
  <xsl:when test="ev_level1_evse_num = 'NONE'">
    <span>Ninguno</span>
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="ev_level1_evse_num"/>
  </xsl:otherwise>
</xsl:choose>
```

Esto muestra "Ninguno"si no hay cargador de nivel 1, o el número si existe.

Contenido

¿Qué veremos en esta sección?

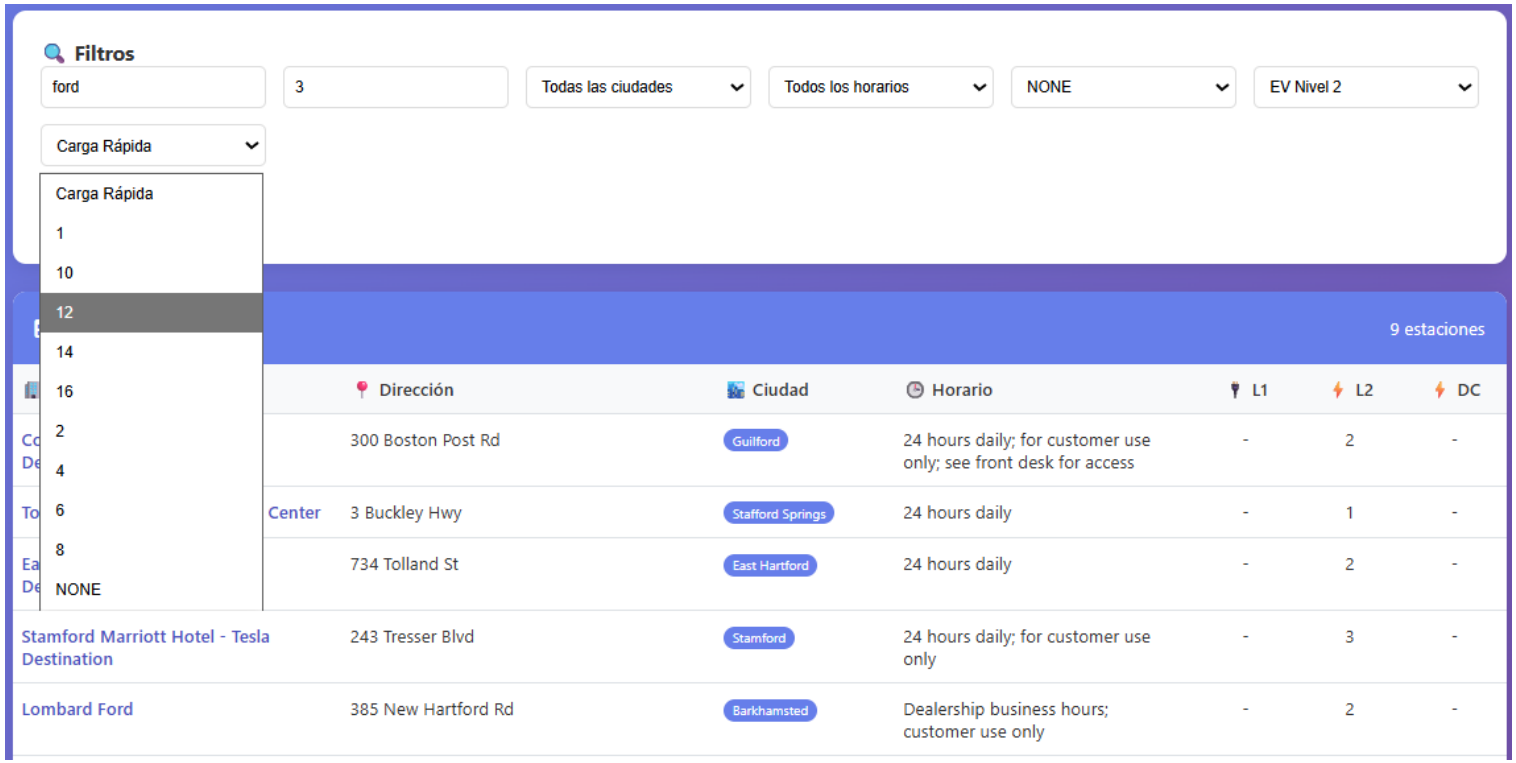
Explicaremos cómo el backend en Python (Flask) obtiene los datos y aplica la transformación XSLT antes de mostrar la página al usuario.

Código clave en Flask

```
@app.route('/')
def home():
    xml_data = obtener_xml_data()
    xslt_path = os.path.join(os.path.dirname(__file__), 'static', 'estilos.xslt')
    xslt_doc = etree.parse(xslt_path)
    transform = etree.XSLT(xslt_doc)
    resultado = transform(xml_data)
    return Response(str(resultado), mimetype='text/html')

```

Vista de los filtros en la aplicación



Contenido

Ventajas de usar XSLT

- Permite separar los datos de la presentación.
- Facilita el mantenimiento y la actualización de la interfaz.
- Es eficiente para transformar grandes volúmenes de datos XML.
- Permite generar HTML, CSS y JavaScript en un solo paso.

Desafíos y limitaciones

- La sintaxis puede ser difícil para quienes no la conocen.
- Depurar errores en XSLT puede ser complicado.
- No es tan interactivo como frameworks modernos, pero se puede complementar con JavaScript.

Contenido

Ideas para el futuro

- Implementar caché para acelerar la carga.
- Añadir paginación para manejar más datos.
- Permitir exportar los datos filtrados.
- Mejorar la accesibilidad y el diseño visual.
- Integrar mapas interactivos.

Contenido

Conclusiones

- XSLT es una herramienta poderosa para transformar y mostrar datos XML.
- Su integración con Flask permite crear aplicaciones web modernas y eficientes.
- La separación entre datos y presentación facilita el mantenimiento.
- Aunque tiene retos, sigue siendo útil en muchos contextos.

¿Preguntas?

¡Gracias por su atención!