



HASSIBA BENBOUALI UNIVERSITY

ASSIGNMENT REPORT

**Semi-Structured Data:
XML, XSD, DTD, XPath and XQuery**

Authors:

Abdelhalim Esselami
Abdelkadir Cheklal

Supervisor:

Boudabous Sofiane

May 11, 2023

Contents

0.1	Introduction	3
0.2	XML Data	3
0.3	DTD	3
0.4	XML Schema	5
0.5	XSLT and CSS	7
0.6	XPath and XQuery	8
0.7	XML to HTML Transformation	9
0.8	Conclusion	10

List of Figures

1	Screenshot of the XML file	3
2	Screenshot of the DTD file	4
3	Screenshot of the XSD file	5
4	Screenshot of the XSD file	6
5	The Schema	6
6	XSLT file	7
7	CSS file	8
8	XPath Output	8
9	XPath Output	9

0.1 Introduction

In this report, we will discuss the assignment on "Données Semi-Structurées" and the steps involved in completing it. The assignment involved working with XML data, creating a DTD, developing an XML schema, implementing XSLT and CSS for styling, executing XPath and XQuery queries, and transforming XML to HTML. We will provide an overview of each step and present the solutions implemented.

0.2 XML Data

We started by collecting data related to a tourist trip in the Hautes-Alpes region. The data included information about the guide, the locations, and the itineraries. We organized this data into an XML file named `trip_hautes_alpes.xml`.

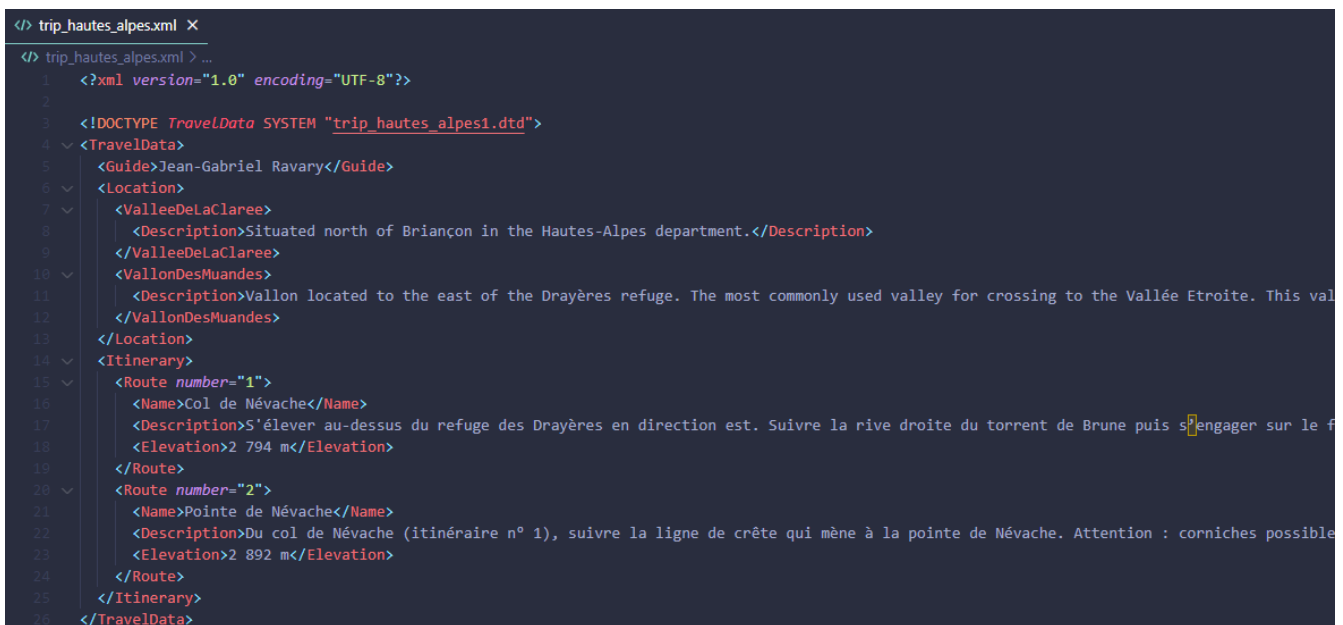


Figure 1: Screenshot of the XML file

0.3 DTD

To define the structure and constraints of the XML data, we created a Document Type Definition (DTD) file named `trip_hautes_alpes.dtd`. The DTD specified the element structure, allowed content, and attribute declarations. The DTD (Document Type Definition) file defines the structure and constraints of the XML document for the trip to Hautes-Alpes. It specifies the elements, attributes, and their relationships allowed in the document.

The DTD consists of the following declarations:

- **TravelData:** The root element representing the travel data.
- **Guide:** Element for specifying the guide's name.

- **Location:** Element for describing the various locations.
- **ValleeDeLaClaree:** Element for providing details about Vallee de la Claree.
- **VallonDesMuandes:** Element for providing details about Vallon des Muandes.
- **Itinerary:** Element for defining the travel itineraries.
- **Route:** Element representing an individual travel route.
- **Name:** Element for specifying the name of a route.
- **Description:** Element for describing the details of a route.
- **Elevation:** Element for specifying the elevation of a route.
- **Route number:** Attribute for assigning a number to a route.

The DTD enforces the structure and content rules for the XML document, ensuring it conforms to the defined schema.

This DTD file is used to validate and ensure the integrity of the XML data for the trip to Hautes-Alpes.

```

trip_hautes_alpes1.dtd X
trip_hautes_alpes1.dtd > TravelData
1  <!ELEMENT TravelData (Guide,Location,Itinerary)>
2  <!ELEMENT Guide (#PCDATA)>
3  <!ELEMENT Location (ValleeDeLaClaree,VallonDesMuandes)>
4  <!ELEMENT ValleeDeLaClaree (Description)>
5  <!ELEMENT Description (#PCDATA)>
6  <!ELEMENT VallonDesMuandes (Description)>
7  <!ELEMENT Itinerary (Route+)>
8  <!ELEMENT Route (Name,Description,Elevation)>
9  <!ATTLIST Route number NMTOKEN #REQUIRED>
10 <!ELEMENT Name (#PCDATA)>
11 <!ELEMENT Elevation (#PCDATA)>

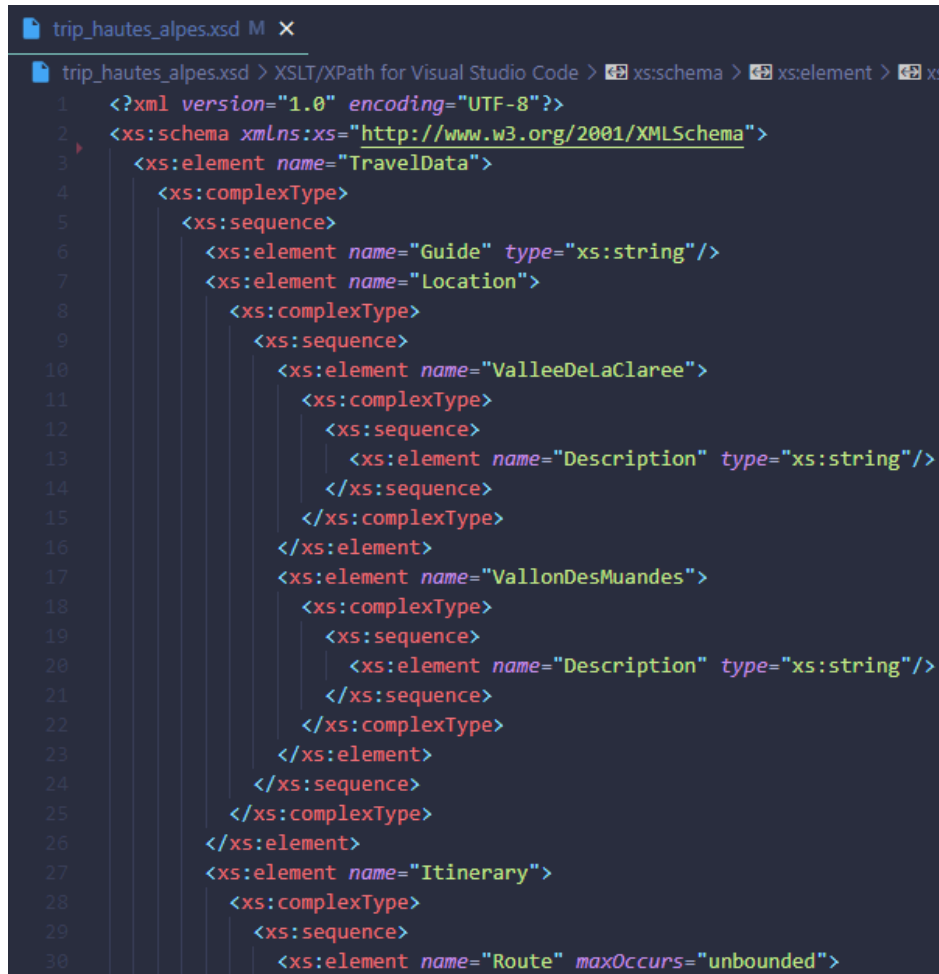
```

Figure 2: Screenshot of the DTD file

0.4 XML Schema

Next, we developed an XML Schema Definition (XSD) file named `trip_hautes_alpes.xsd`. The XML schema provided a more robust and flexible way to define the structure and constraints of the XML data compared to DTD.

Here are some screenshots:

The image shows a screenshot of a code editor window titled 'trip_hautes_alpes.xsd'. The editor displays an XML Schema Definition (XSD) file. The schema is defined with the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="TravelData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Guide" type="xs:string"/>
        <xs:element name="Location">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ValleeDeLaClaree">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Description" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="VallonDesMuandes">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Description" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Itinerary">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Route" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The code is color-coded, with tags in blue, attributes in green, and text in white. The editor also shows a breadcrumb trail at the top: 'trip_hautes_alpes.xsd > XSLT/XPath for Visual Studio Code > xs:schema > xs:element > xs:complexType > xs:sequence'.

Figure 3: Screenshot of the XSD file

```

30      <xs:element name="Route" maxOccurs="unbounded">
31        <xs:complexType>
32          <xs:sequence>
33            <xs:element name="Name" type="xs:string"/>
34            <xs:element name="Description" type="xs:string"/>
35            <xs:element name="Elevation" type="xs:string"/>
36          </xs:sequence>
37          <xs:attribute name="number" type="xs:string" use="required"/>
38        </xs:complexType>
39      </xs:element>
40    </xs:sequence>
41  </xs:complexType>
42</xs:element>
43</xs:sequence>
44</xs:complexType>
45</xs:element>
46</xs:schema>

```

Figure 4: Screenshot of the XSD file

```

XML DOCUMENT
├── </> schema (attributes: 1, children: 1)
│   ├── @ xs = "http://www.w3.org/2001/XMLSchema"
│   └── </> element (attributes: 1, children: 1)
│       ├── @ name = "TravelData"
│       └── </> complexType (children: 1)
│           ├── </> sequence (children: 3)
│           │   ├── </> element (attributes: 2)
│           │   │   ├── @ name = "Guide"
│           │   │   └── @ type = "xs:string"
│           │   └── </> element (attributes: 1, children: 1)
│           │       ├── @ name = "Location"
│           │       └── </> complexType (children: 1)
│           │           ├── </> sequence (children: 2)
│           │           │   ├── </> element (attributes: 1, children: 1)
│           │           │   │   ├── @ name = "ValleeDeLaClaree"
│           │           │   │   └── </> complexType (children: 1)
│           │           │   │   ├── </> sequence (children: 1)
│           │           │   │   └── </> element (attributes: 2)
│           │           │   │       ├── @ name = "Description"

```

(a) First Screenshot

```

@ type = "xs:string"
├── </> element (attributes: 1, children: 1)
│   ├── @ name = "VallonDesMuandes"
│   └── </> complexType (children: 1)
│       ├── </> sequence (children: 1)
│       │   ├── </> element (attributes: 2)
│       │   │   ├── @ name = "Description"
│       │   │   └── @ type = "xs:string"
│       └── </> element (attributes: 1, children: 1)
│           ├── @ name = "Itinerary"
│           └── </> complexType (children: 1)
│               ├── </> sequence (children: 1)
│               │   ├── </> element (attributes: 2, children: 1)
│               │   │   ├── @ name = "Route"
│               │   │   ├── @ maxOccurs = "unbounded"
│               │   └── </> complexType (children: 2)
│               │       ├── </> sequence (children: 3)
│               │       │   ├── </> element (attributes: 2)
│               │       │   │   ├── @ name = "Name"

```

(b) Second Screenshot

```

@ type = "xs:string"
├── </> element (attributes: 2)
│   ├── @ name = "Description"
│   └── @ type = "xs:string"
├── </> element (attributes: 2)
│   ├── @ name = "Elevation"
│   └── @ type = "xs:string"
└── </> attribute (attributes: 3)
    ├── @ name = "number"
    ├── @ type = "xs:string"
    └── @ use = "required"

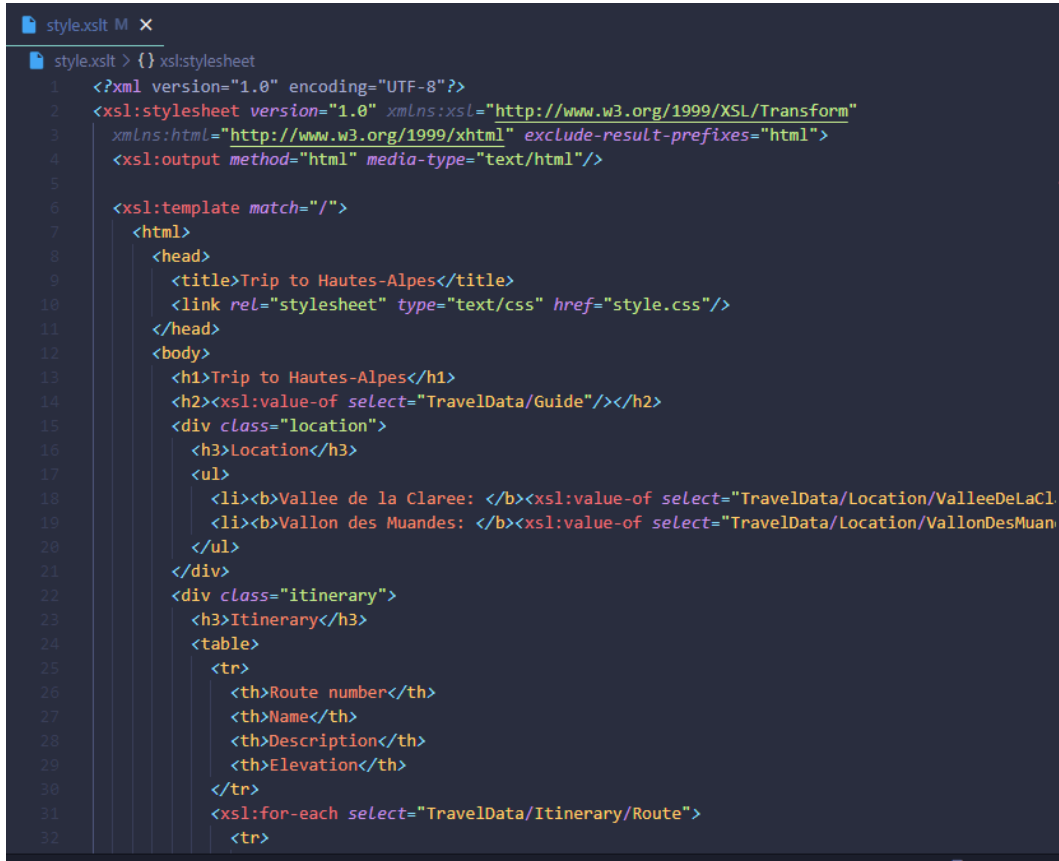
```

(c) Third Screenshot

Figure 5: The Schema

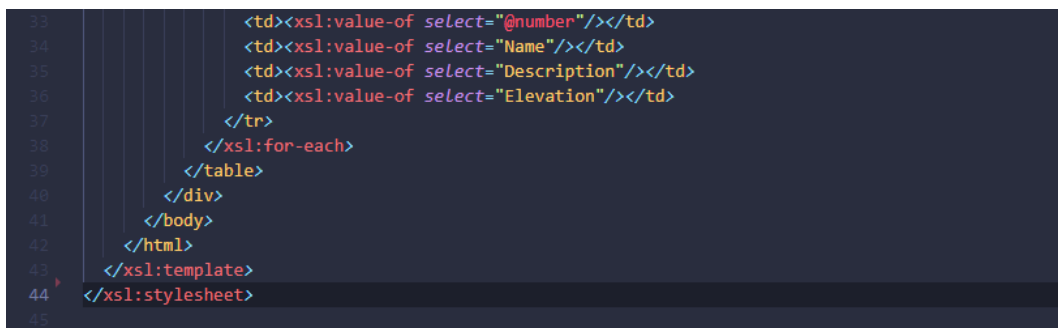
0.5 XSLT and CSS

To transform the XML data into HTML and apply styling, we created an XSLT file named `transform.xsl`. The XSLT file defined templates and rules to transform the XML elements into HTML elements. We also developed a CSS file named `style.css` to apply styles and layout to the HTML output.



```
style.xslt M x
style.xslt > {} xslstylesheet
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3   xmlns:html="http://www.w3.org/1999/xhtml" exclude-result-prefixes="html">
4   <xsl:output method="html" media-type="text/html"/>
5
6   <xsl:template match="/">
7     <html>
8       <head>
9         <title>Trip to Hautes-Alpes</title>
10        <link rel="stylesheet" type="text/css" href="style.css"/>
11      </head>
12      <body>
13        <h1>Trip to Hautes-Alpes</h1>
14        <h2><xsl:value-of select="TravelData/Guide"/></h2>
15        <div class="location">
16          <h3>Location</h3>
17          <ul>
18            <li><b>Vallee de la Claree:</b><xsl:value-of select="TravelData/Location/ValleeDeLaCl
19            <li><b>Vallon des Muandes:</b><xsl:value-of select="TravelData/Location/VallonDesMuan
20          </ul>
21        </div>
22        <div class="itinerary">
23          <h3>Itinerary</h3>
24          <table>
25            <tr>
26              <th>Route number</th>
27              <th>Name</th>
28              <th>Description</th>
29              <th>Elevation</th>
30            </tr>
31            <xsl:for-each select="TravelData/Itinerary/Route">
32              <tr>
```

(a) First Screenshot



```
33      <td><xsl:value-of select="@number"/></td>
34      <td><xsl:value-of select="Name"/></td>
35      <td><xsl:value-of select="Description"/></td>
36      <td><xsl:value-of select="Elevation"/></td>
37    </tr>
38  </xsl:for-each>
39 </table>
40 </div>
41 </body>
42 </html>
43 </xsl:template>
44 </xsl:stylesheet>
45
```

(b) Second Screenshot

Figure 6: XSLT file


```

1  body {
2    font-family: 'Helvetica Neue', sans-serif;
3    font-size: 16px;
4    line-height: 1.5;
5    margin: 0;
6    padding: 0;
7  }
8
9  h1, h2, h3, h4, h5, h6 {
10   font-family: 'Roboto', sans-serif;
11   font-weight: bold;
12   margin-top: 0;
13   margin-bottom: 1rem;
14 }
15
16 h1 {
17   font-size: 2rem;
18 }
19
20 h2 {
21   font-size: 1.5rem;
22 }
23
24 h3 {
25   font-size: 1.25rem;
26 }
27
28 h4 {
29   font-size: 1.125rem;
30 }
31

```

(a) First Screenshot

```

32 h5 {
33   font-size: 1rem;
34 }
35
36 h6 {
37   font-size: 0.875rem;
38 }
39
40 p {
41   margin-top: 0;
42   margin-bottom: 1rem;
43 }
44
45 ul {
46   margin-top: 0;
47   margin-bottom: 1rem;
48   list-style-type: disc;
49 }
50
51 table {
52   border-collapse: collapse;
53   width: 100%;
54 }
55
56 th, td {
57   padding: 0.75rem;
58   border: 1px solid #ccc;
59 }
60
61 th {
62   background-color: #f2f2f2;
63 }
64

```

(b) Second Screenshot

```

65 /* Specific styles for the travel data */
66 .location {
67   margin-top: 2rem;
68 }
69
70 .location ul {
71   margin-top: 0;
72   margin-bottom: 0;
73 }
74
75 .location b {
76   font-weight: bold;
77 }
78
79 .itinerary {
80   margin-top: 2rem;
81 }
82
83 .itinerary table {
84   margin-top: 1rem;
85   margin-bottom: 0;
86 }
87
88 .itinerary th {
89   text-align: left;
90 }
91
92 .itinerary td {
93   text-align: center;
94 }
95

```

(c) Second Screenshot

Figure 7: CSS file

0.6 XPath and XQuery

Using XPath and XQuery, we performed queries on the XML data. With XPath, we retrieved all the itineraries using the expression `/TravelData/Itinerary/Route`. With XQuery, we obtained the names of the tour guides using the query `for $guide in /TravelData/Guide return $guide`.

```

XPath Query: /TravelData/Itinerary/Route

[Line 15] Route:
Col de Névache
S'élever au-dessus du refuge des Drayères en direction est. Suivre la rive droite du torrent de Brune puis
s'engager sur le flanc droit du ravin des Muandes que l'on quitte vers 2500 m pour rejoindre le col situé au
nord. Descente possible sur Valmeinier. Départ assez raide.
2 794 m

[Line 20] Route:
Pointe de Névache
Du col de Névache (itinéraire n° 1), suivre la ligne de crête qui mène à la pointe de Névache. Attention :
corniches possibles. Crampons utiles au printemps.

```

Figure 8: XPath Output

0.7 XML to HTML Transformation

Finally, we executed the XSLT transformation to convert the XML data to HTML using the Saxon XSLT processor. The output HTML file named `output.html` displayed the trip information with the applied styles defined in the CSS file.

The command used :

```
xsltproc -o output.html style.xslt trip_hautes_alpes.xml
```

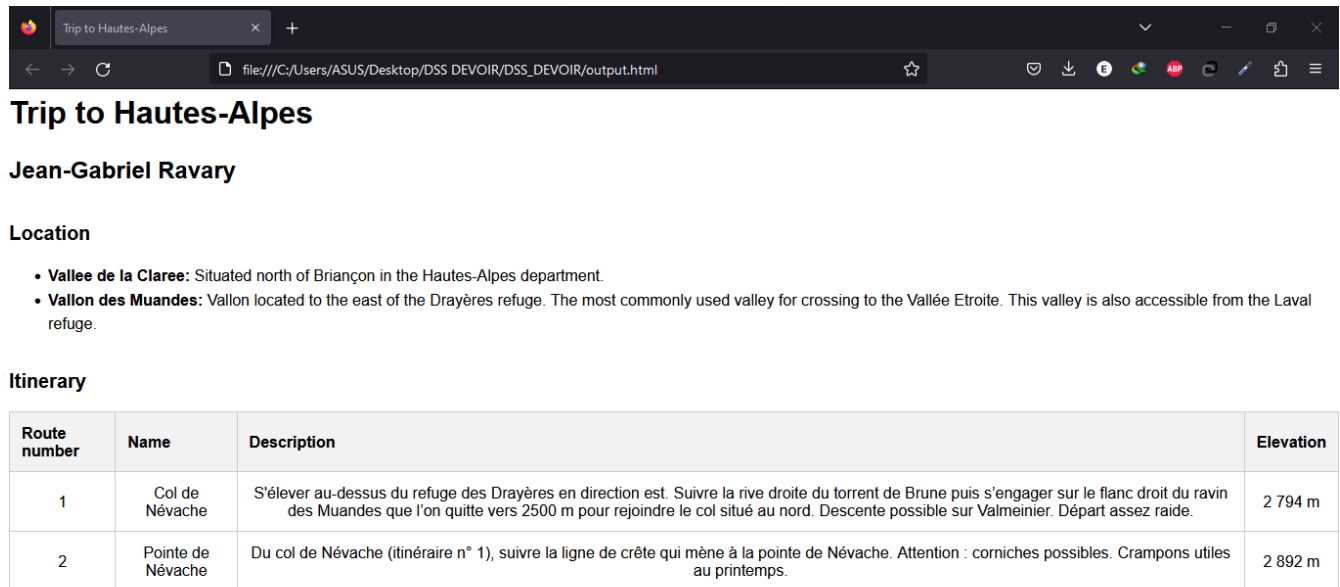


Figure 9: XPath Output

0.8 Conclusion

In conclusion, the assignment on "Données Semi-Structurées" involved working with XML data, creating a DTD, developing an XML schema, implementing XSLT and CSS for styling, executing XPath and XQuery queries, and transforming XML to HTML. By following the provided instructions and utilizing the appropriate tools, we successfully completed the assignment and obtained the desired results. XML (eXtensible Markup Language) has proven to be a valuable tool for structuring and organizing data in a hierarchical and self-descriptive format. Throughout this assignment, we explored the use of XML for representing travel data related to a trip in the Hautes-Alpes region. XML's flexibility allows us to define custom elements and attributes tailored to our specific needs, such as defining the guide, locations, and itineraries.

By adhering to a defined DTD and XML schema, we ensure the integrity and consistency of the data. The DTD serves as a blueprint for the structure of the XML document, while the XML schema provides a more robust validation mechanism with data types, constraints, and more.

Additionally, we leveraged XPath and XQuery to query and extract specific information from the XML data. XPath provided a concise and powerful syntax for navigating the XML hierarchy and selecting elements based on criteria. XQuery allowed us to perform more complex queries, such as retrieving the names of the tourist guides.

Overall, XML offers a standardized and versatile approach to storing and exchanging data, making it a widely adopted technology in various domains. Its ability to represent both structured and semi-structured data makes it suitable for a wide range of applications, including data integration, document storage, and web services.

In conclusion, our exploration of XML and its associated technologies has provided valuable insights into the benefits and capabilities of this markup language. As technology continues to evolve, XML remains a fundamental tool for managing and exchanging data in a structured and interoperable manner.