

Ecole supérieur de technologie Meknès

Université Moulay Ismail

Filière : Génie Informatique (GI)

Rapport de mini projet XML
Mise en œuvre d'une
Plateforme de
Location de voiture

Amchayd Marian

Aggagui Yassine

Section : A

Groupe : 1

2016-2017

Dans un premier temps nous tenons à exprimer notre profonde gratitude au seigneur Dieu pour les grâces reçues tout au long de notre vie. Au terme de ce travail, nous tenons à remercier vivement et nous exprimons toute notre gratitude à Madame SAMIA NASIRI pour leur aide, pour leur soutien et leur disponibilité, ainsi que pour leurs précieux conseils.

Nous tenons aussi à exprimer toute nos gratitude et notre respect, à tout le corps professoral du département « Informatique », ainsi que tout le corps administratif de l'école, pour leur enseignement et leur soutien.

Nous vifs remerciements vont également à nos parents, pour leurs disponibilités, leurs aides, leurs compréhensions, leurs patiences et leurs soutiens permanents. Enfin nous voudrais faire part de nos gratitude à tous ceux qui ont participés de près ou de loin au bon déroulement de ce travail.

Table de figure :

| | |
|---|----|
| Figure 1: XML Clients..... | 6 |
| Figure 2 : DTD Clients | 6 |
| Figure 3 : Schéma XML Clients – élément racine | 7 |
| Figure 4 : Schema XML Clients- SimpleType | 7 |
| Figure 5 : Schéma XML Clients- Simple Type | 8 |
| Figure 6 : élément Client..... | 8 |
| Figure 7 : XML Véhicule..... | 9 |
| Figure 8 : DTD véhicule | 9 |
| Figure 9 : Schéma XML Véhicule | 10 |
| Figure 10 : Schéma XML Véhicule - simple Type..... | 10 |
| Figure 11 : XML Loues | 11 |
| Figure 12 : DTD Loues | 11 |
| Figure 13 : Schéma XML Loues..... | 12 |
| Figure 14 : simple Type Loue | 12 |
| Figure 15 : requête 1 affiche le CIN -Nom-Prénom de client..... | 13 |
| Figure 16 : résultat de requête 1 | 13 |
| Figure 17 : requête 2 affiche le nombre de clients | 13 |
| Figure 18 : résultat de requête 2 | 14 |
| Figure 19 : requête 3 affiche tous les informations de véhicule | 14 |
| Figure 20 : résultat de requête 3 | 14 |
| Figure 21 : requête 4 affiche le membre de voiture | 15 |
| Figure 22 : affiche résultat de requête 4 | 15 |
| Figure 23 : requête 5 affiche les informations de client par ordre croissant..... | 15 |
| Figure 24 : résultat de requête 5 | 16 |
| Figure 25 : requête 6 affiche un client par son nom | 16 |
| Figure 26 : résultat de requête 6 | 16 |
| Figure 27 : requête 7 affiche le client qui appartient à deux tables client et loue | 17 |
| Figure 28 : requête 8 affiche les informations de véhicule par son immatriculation | 17 |
| Figure 29 : logo EditiX | 18 |
| Figure 30 : Logo BaseX | 18 |

Sommaire

| | |
|---|----|
| Les types de modélisation : | 5 |
| Partie 1 : Clients (XML-DTD-Schéma XML)..... | 6 |
| 1. Document XML de Clients : | 6 |
| 2. DTD de Clients : | 6 |
| 3. Schéma XML de Clients : | 7 |
| Partie 2 : Véhicule (XML-DTD-Schéma XML)..... | 9 |
| 1. Document XML de Véhicule:..... | 9 |
| 2. DTD de Véhicule:..... | 9 |
| 3. Schéma XML de Véhicule: | 10 |
| Partie 3 : Loue (XML-DTD-Schéma XML)..... | 11 |
| 1. Document XML de Loues: | 11 |
| 2. DTD de Loues : | 11 |
| 3. Schéma XML de Loues :..... | 12 |
| Partie 4 : Xquery..... | 13 |
| Partie 5 : Les outils utilisent : | 18 |
| Conclusion | 19 |

Introduction

On souhaite de réaliser un mini projet qui se déroule sur **La location de voiture** courte durée (LCD) ou location de véhicule, qui est un service proposé par une entreprise offrant aux clients la location des automobiles pour de courtes périodes pouvant aller de quelques heures à quelques semaines. Les sociétés de location sont souvent constituées de nombreuses agences locales permettant notamment aux clients de retourner leur véhicule à un endroit différent de celui de la prise en charge.

Les types de modélisation :

➤ XML :

XML ou eXtensible Markup Language est un langage informatique de balisage générique qui permet de décrire des documents structures hiérarchiquement, utilisant des balises.

➤ DTD :

La document type définition (DTD), ou définition de type de document, est un document permettant de décrire un modèle de document XML. Le modèle est décrit comme une grammaire de classe de documents¹ : grammaire parce qu'il décrit la position des termes les uns par rapport aux autres, classe parce qu'il forme une généralisation d'un domaine particulier, et document parce qu'on peut former avec un texte complet.

➤ XML Schéma

XML Schéma publié comme recommandation par le W3C en mai 2001 est un langage de description de format de document XML permettant de définir la structure et le type de contenu d'un document XML. Cette définition permet notamment de vérifier la validité de ce document.

➤ Xquery :

XQuery est un langage de requête informatique permettant non seulement d'extraire des informations d'un document XML, ou d'une collection de documents XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML.

Partie 1 : Clients (XML-DTD-Schéma XML)

1. Document XML de Clients :

On souhaite écrire un document **clients.xml** en utilisant le formalisme XML, le client doit contenir la liste des informations suivantes :

{ CIN - Nom – Prénom – Adresse – Ville – CodePostal - date_permis_conduire }

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- New XML document created with EditiX XML Editor (http://www.editix.com) at Mon Nov 14 14:35:01 WET 2016 -->
<!DOCTYPE clients SYSTEM "dtdclient.dtd">
<clients xsi:noNamespaceSchemaLocation="shemaclient.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <client CIN="A152368">
    <nom>AGGAGUI</nom>
    <prenom>Yassine</prenom>
    <adresse>adresse1</adresse>
    <ville>Kenitra</ville>
    <codePostal>11111</codePostal>
    <date_permis_conduire>08/05/2016</date_permis_conduire>
  </client>
  <client CIN="Z126598">
    <nom>AMCHAYD</nom>
    <prenom>Maryame</prenom>
    <adresse>adresse2</adresse>
    <ville>Meknes</ville>
    <codePostal>22222</codePostal>
    <date_permis_conduire>01/12/2016</date_permis_conduire>
  </client>
</clients>
```

Figure 1: XML Clients

2. DTD de Clients :

Se qui concerne le DTD de client **DTDClient.dtd** on a intégrer les valeurs par défaut au balise de document XML sauf l'attribut CIN qui est obligatoire (en mode required) et identifiant de chaque client (ID).

```
<!ELEMENT clients (client+)>
<!ELEMENT client (nom,prenom,adresse,codePostal,ville,date_permis_conduire)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT adresse (#PCDATA)>
<!ELEMENT codePostal (#PCDATA)>
<!ELEMENT ville (#PCDATA)>
<!ELEMENT date_permis_conduire (#PCDATA)>
<!ATTLIST client CIN ID #REQUIRED>
```

Figure 2 : DTD Clients

3. Schéma XML de Clients :

Pour le schéma XML **schemaClient.xsd**, on a créer un élément racine clients qui a une référence sur un élément client de même ce dernière a plusieurs références sur plusieurs type.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- New XSD document created with EditiX XML Editor (http://www.editix.com) at Mon Nov 14 16

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="clients">
    <xs:complexType>
      <xs:sequence >
        <xs:element ref="client" maxOccurs="unbounded"/></xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Figure 3 : Schéma XML Clients – élément racine

```
<xs:simpleType name="CINType">
  <xs:restriction base="xs:ID">
    <xs:pattern value="[A-Z][1-9]+"

```

Figure 4 : Schema XML Clients- SimpleType

Pour chaque simple Type on a intégré un pattern pour définir la structure et la forme de l'information a insérait .par exemple le CIN doit commencer par une lettre majuscule suivie d'un ensemble de chiffre de même pour les autres types.

```
<xs:element name="nom" type="nom"/>
<xs:element name="prenom" type="prenom"/>
<xs:element name="ville" type="ville"/>
<xs:element name="adresse" type="xs:string"/>
<xs:element name="date_permis_conduire" type="xs:string"/>
<xs:element name="codePostal" type="codePostaleType"/>
```

Figure 5 : Schéma XML Clients- Simple Type

Pour ce genre de type on met le nom et le type de chaque balise.

```
<xs:element name="client">
  <xs:complexType>
    <xs:sequence>

      <xs:element ref="nom" />

      <xs:element ref="prenom" />

      <xs:element ref="adresse" />

      <xs:element ref="ville" />

      <xs:element ref="codePostal" />

      <xs:element ref="date_permis_conduire" />

    </xs:sequence>

    <xs:attribute name="CIN" type="CINType"/>

  </xs:complexType>
</xs:element>
</xs:schema>
```

Figure 6 : élément Client

Dans cette capture on a l'élément client contient autre élément qui on référencies au types précédents.

Partie 2 : Véhicule (XML-DTD-Schéma XML)

1. Document XML de Véhicule:

Dans ce document **vehicule.xml**, on a les informations suivantes :

{IdVehicul – marque – immatriculation – couleur – nombrePlace}

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- New XML document created with EditiX XML Editor (http://www.editix.com) at Mon Nov 14 14:46:27 WET 2016 -->

<!DOCTYPE vehiculs SYSTEM "dtdvehicul.dtd">
<vehiculs>
  <vehicul idVehicul="v1">
    <marque>dacia</marque>
    <immatriculation>59876-a-25647</immatriculation>
    <couleur>blanc</couleur>
    <nombrePlace>5</nombrePlace>
  </vehicul>
  <vehicul idVehicul="v2">
    <marque>peugeot</marque>
    <immatriculation>56497-a-85694</immatriculation>
    <couleur>noir</couleur>
    <nombrePlace>6</nombrePlace>
  </vehicul>
</vehiculs>
```

Figure 7 : XML Véhicule

2. DTD de Véhicule:

De même pour **DTDVehicule.dtd** on a intégré les valeurs par défaut au balise de document XML sauf l'attribut idVehicul qui est obligatoire et identifiant de chaque véhicule.

```
<!ELEMENT vehiculs (vehicul+)>
<!ELEMENT vehicul (marque,immatriculation,couleur,nombrePlace)>
<!ELEMENT marque (#PCDATA)>
<!ELEMENT immatriculation (#PCDATA)>
<!ELEMENT couleur (#PCDATA)>
<!ELEMENT nombrePlace (#PCDATA)>
<!ATTLIST vehicul idVehicul ID #REQUIRED>
```

Figure 8 : DTD véhicule

3. Schéma XML de Véhicule:

Pour le schéma XML schemaVehicul.xsd, on a créer un élément racine véhicules qui a une référence sur un élément véhicule de même ce dernière a plusieurs références sur plusieurs type.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="vehiculs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="vehicul" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="vehicul">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="marque"/>
        <xs:element ref="immatriculation"/>
        <xs:element ref="couleur"/>
        <xs:element ref="nombrePlace"/>
      </xs:sequence>
      <xs:attribute name="idVehicul" type="xs:ID" use="required"/>
    </xs:complexType>
  </xs:element>
```

Figure 9 : Schéma XML Véhicule

```
<xs:simpleType name="nombrePlace">
  <xs:restriction base="xs:decimal">
    <xs:maxInclusive value="7"></xs:maxInclusive>
    <xs:minInclusive value="5"></xs:minInclusive>
  </xs:restriction>
</xs:simpleType>
<xs:element name="nombrePlace" type="nombrePlace"/>
<xs:element name="couleur" type="xs:string"/>
<xs:element name="immatriculation" type="xs:string"/>
<xs:element name="marque" type="xs:string"/>
</xs:schema>
```

Figure 10 : Schéma XML Véhicule - simple Type

Pour ce simple Type on a intégré une restriction de base décimale pour définir la valeur minimum et maximum de nombres de places.

Partie 3 : Loue (XML-DTD-Schéma XML)

1. Document XML de Loues:

Dans ce document **Loue.xml**, on a les informations suivantes :

{Date_debut – date_fin – montantTotal}

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- New XML document created with EditiX XML Editor (http://www.editix.com) at Mon Nov 14 14:41:49 WET 2016 -->
<!DOCTYPE loues SYSTEM "dtdloue.dtd">
<loues>
  <loue idloue="l1" idVehicul="v1" CIN="A152368">
    <date_debut>23/08/2014</date_debut>
    <date_fin>25/08/2016</date_fin>
    <montantTotal>2500DH</montantTotal>
  </loue>
</loues>
```

Figure 11 : XML Loues

2. DTD de Loues :

De même pour DTDLoue.dtd on a intégrer les valeurs par défaut aux balises de document XML mais cette fois on a ajoute trois identifiants « idloue » identifier la balise <loue>, « CIN » identifier la balise <loue> de même est un référence sur le document Client.xml et aussi « idVehicul » identifier aussi loue références sur le document Vehicule.xml.

```
<!ELEMENT loues (loue+) >
<!ELEMENT loue (date_debut,date_fin,montantTotal) >
<!ELEMENT date_debut (#PCDATA)>
<!ELEMENT date_fin (#PCDATA)>
<!ELEMENT montantTotal (#PCDATA)>
<!ATTLIST loue idloue ID #REQUIRED >
<!ATTLIST loue CIN IDREF #REQUIRED >
<!ATTLIST loue idVehicul IDREF #REQUIRED>
```

Figure 12 : DTD Loues

3. Schéma XML de Loues :

Pour le schéma XML schemaLoue.xsd, on a créer un élément racine Loues qui a une référence sur un élément Loue de même ce dernière a plusieurs références sur plusieurs type.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="loues">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="loue" maxOccurs="unbounded"/></xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="loue">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="date_debut"/>
        <xs:element ref="date_fin"/>
        <xs:element ref="montantTotal"/>
      </xs:sequence>
      <xs:attribute name="CIN" type="xs:IDREF" use="required"/>
      <xs:attribute name="idVehicul" type="xs:IDREF" use="required"/>
      <xs:attribute name="idloue" type="xs:ID" use="required"/>
    </xs:complexType>
  </xs:element>
```

Figure 13 : Schéma XML Loues

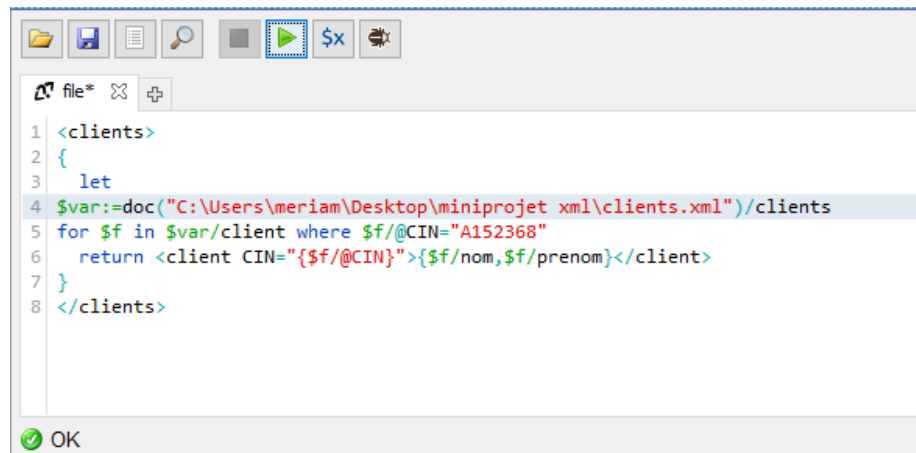
```
<xs:simpleType name="montantTotal">
  <xs:restriction base="xs:string">
    <xs:pattern value="[1-9]+"></xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:element name="montantTotal" type="montantTotal"/>
<xs:element name="date_debut" type="xs:string"/>
<xs:element name="date_fin" type="xs:string"/>
</xs:schema>
```

Figure 14 : simple Type Loue

De même Pour ce schéma on a un simple Type intégré un pattern pour définir la structure et la forme de montant Total.

Partie 4 : Xquery



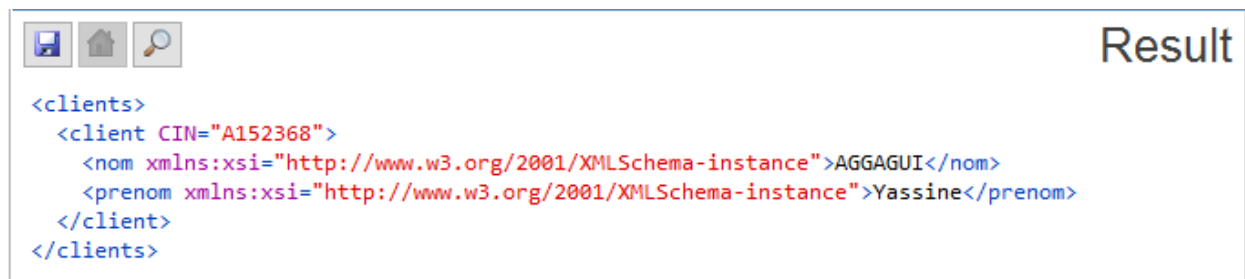
```

1 <clients>
2 {
3   let
4   $var:=doc("C:\Users\meriam\Desktop\miniprojet xml\clients.xml")/clients
5   for $f in $var/client where $f/@CIN="A152368"
6   return <client CIN="{ $f/@CIN}">{ $f/nom,$f/prenom}</client>
7 }
8 </clients>

```

OK

Figure 15 : requête 1 affiche le CIN -Nom-Prénom de client



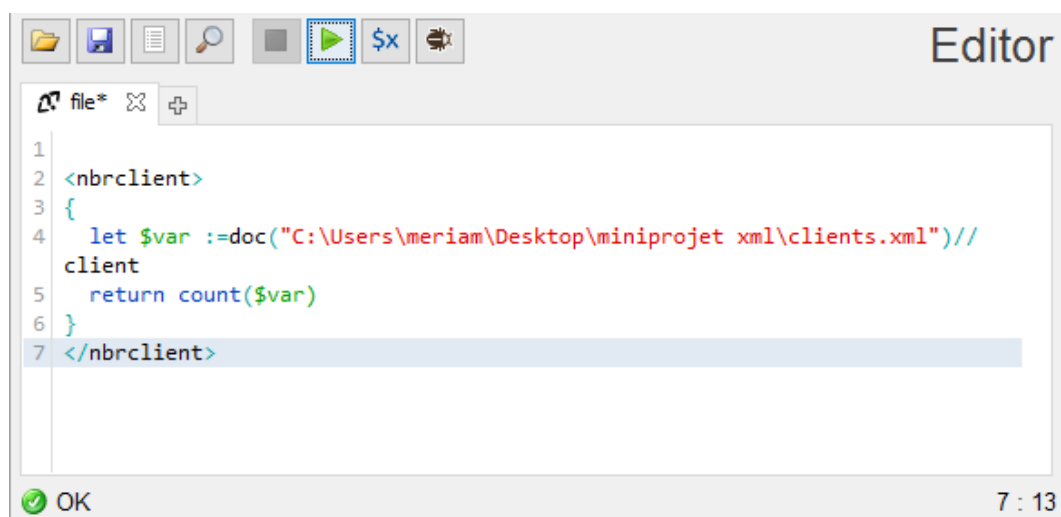
Result

```

<clients>
  <client CIN="A152368">
    <nom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">AGGAGUI</nom>
    <prenom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Yassine</prenom>
  </client>
</clients>

```

Figure 16 : résultat de requête 1



```

1
2 <nbrclient>
3 {
4   let $var :=doc("C:\Users\meriam\Desktop\miniprojet xml\clients.xml")//
  client
5   return count($var)
6 }
7 </nbrclient>

```

OK

7 : 13

Figure 17 : requête 2 affiche le nombre de clients

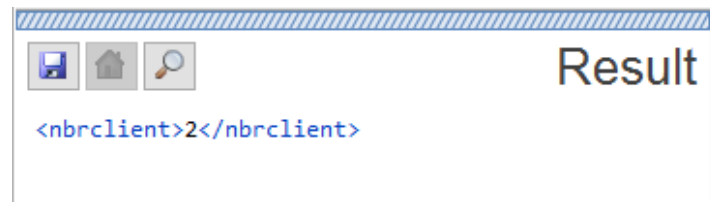


Figure 18 : résultat de requête 2

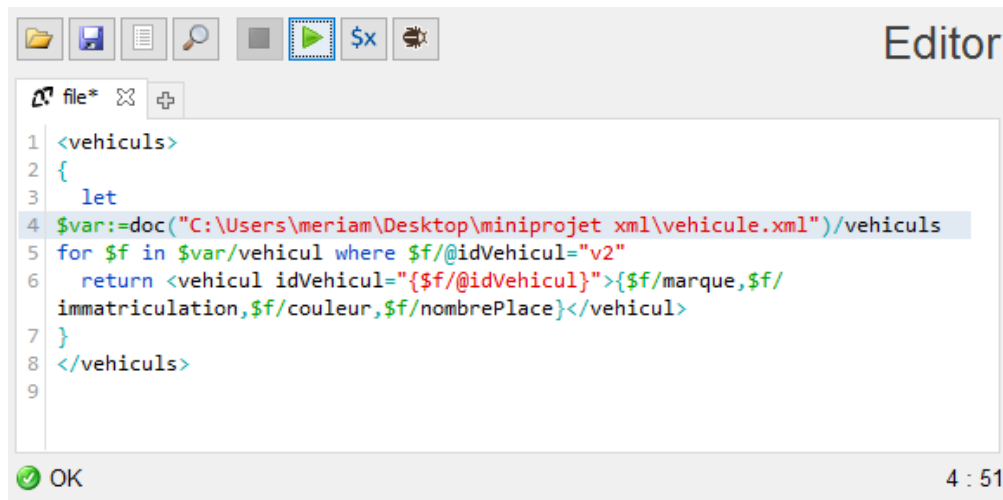


Figure 19 : requeté 3 affiche tous les informations de vehicule

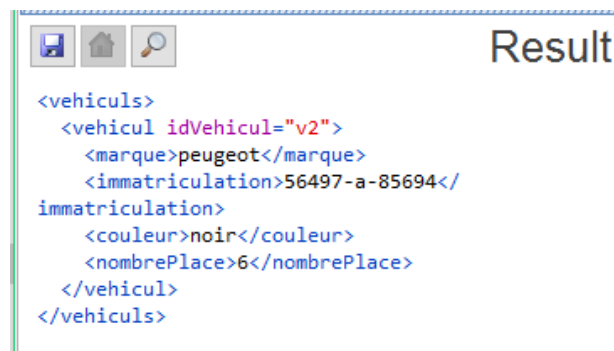
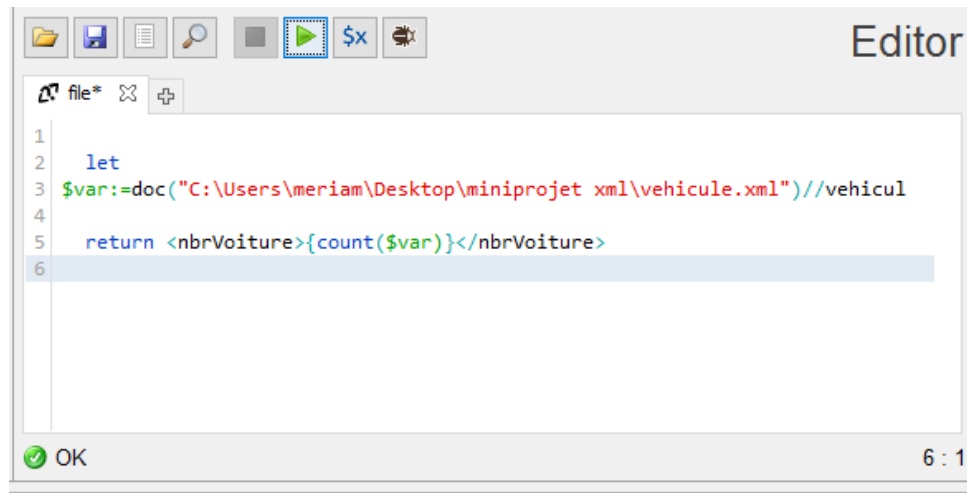


Figure 20 : résultat de requête 3



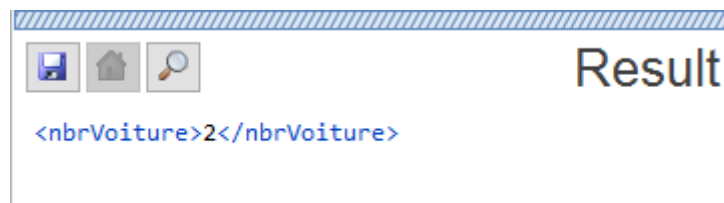
```

1
2   let
3   $var:=doc("C:\Users\meriam\Desktop\miniprojet xml\vehicule.xml")//vehicul
4
5   return <nbrVoiture>{count($var)}</nbrVoiture>
6

```

OK 6 : 1

Figure 21 : requête 4 affiche le membre de voiture



```

<nbrVoiture>2</nbrVoiture>

```

Figure 22 : affiche résultat de requête 4



```

1 <clients>{
2   let
3   $var:=doc("C:\Users\meriam\Desktop\miniprojet xml\clients.xml")//client
4
5   for $f in $var order by $f/ID ascending
6   return <client ID="{ $f/ID}">{$f/nom,$f/prenom,$f/adresse,$f/ville,$f/codePostal,$f/date_permis_conduire}</client>
7 }</clients>
8
9

```

OK

Figure 23 : requête 5 affiche les informations de client par ordre croissant



Figure 24 : résultat de requête 5

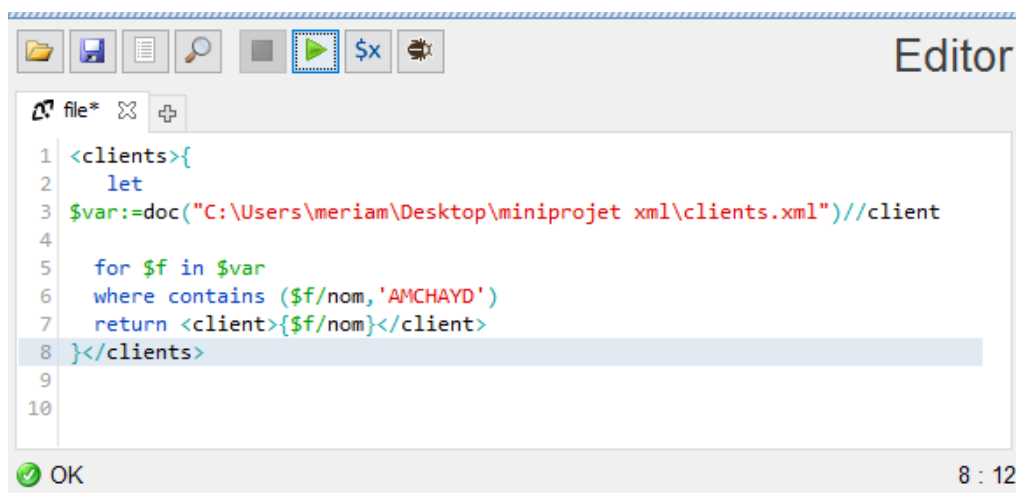


Figure 25 : requête 6 afficher un client par son nom

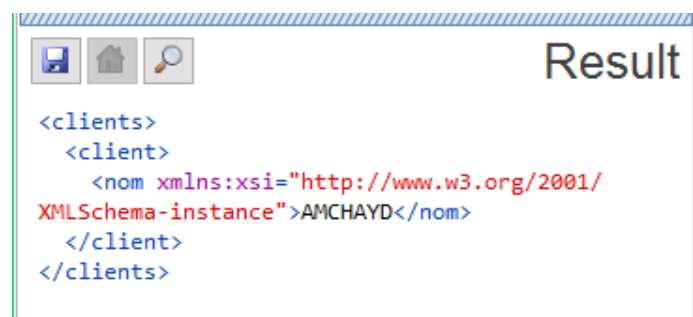


Figure 26 : résultat de requête 6


```

1 <clients>{
2   let
3   $f:=doc("C:\Users\meriam\Desktop\miniprojet xml\clients.xml")//client,
4   $d:=doc("C:\Users\meriam\Desktop\miniprojet xml\loue.xml")//loue
5   where
6     $d/loue/@idRef=$f/ID
7
8   return ($f/nom,$f/prenom)
9 }</clients>
10
11

```

OK 9 : 12

Figure 27 : requête 7 afficher le client qui appartient a deux table client et loue

```

1   let
2   $var:=doc("C:\Users\meriam\Desktop\miniprojet xml\vehicule.xml")//
vehicule
3   for $d in $var
4   where $d/immatriculation='59876-a-25647'
5   return
6   <vehicule>{$d/marque,$d/couleur,$d/nombrePlace}</vehicule>
7
8

```

OK 8 : 1

Figure 28 : requête 8 afficher les informations de véhicule par son immatriculation

Partie 5 : Les outils utilisent :

➤ EditiX :



Figure 29 : logo EditiX

EditiX XML Editor est un éditeur XML propriétaire multiplateforme (Windows, Linux, Mac OS X). Il propose une panoplie d'assistants pour différents types de document (DTD, XSLT, XSL-FO, Relax NG...). Un débogueur XSLT est disponible pour la version 1.0 et 2.0. Un concepteur graphique de schéma XSD est également présent depuis la version 2008.

➤ BaseX



Figure 30 : Logo BaseX

BaseX est un système de gestion de base de données XML native et légère, développé en tant que projet communautaire sur GitHub¹. BaseX est spécialisé dans le stockage, le requêtage et la visualisation de larges documents et collections de documents XML.

Conclusion :

La location de voiture répond en premier lieu à un besoin temporaire pour celles et ceux qui ne possèdent pas leur propre véhicule, qui ont besoin d'un véhicule de remplacement à la suite de l'immobilisation de leur automobile.

Notre mini projet va organiser les informations des clients ont réserve un véhicule d'un façon structurer.