Université Grenoble Alpes

05/12/22

Rapport

Voyageur de santé

SAMUEL DAMESSI / SOULEYMEN OUCHANE
L3 MIAGE

Introduction :

Ce projet a pour objectif de voir toutes les notions vues en cours de « Formalisation des Données – Technologies XML ». Dans ce projet, nous devons formaliser un cabinet de santé grâce à la technologie XML. Pour cela, on a appris à utiliser xml ainsi que les technologies xml schema et xslt.

Pour cela, nous avons utilisé le logiciel Netbeans comme environnement de développement.

Nous allons vous présenter les différentes étapes de notre travail.

Cabinet.xml :

Ce document est la base du projet, c'est dans ce fichier que sera stocké toutes les données du cabinet. Il est composé de l'arborescence suivante :

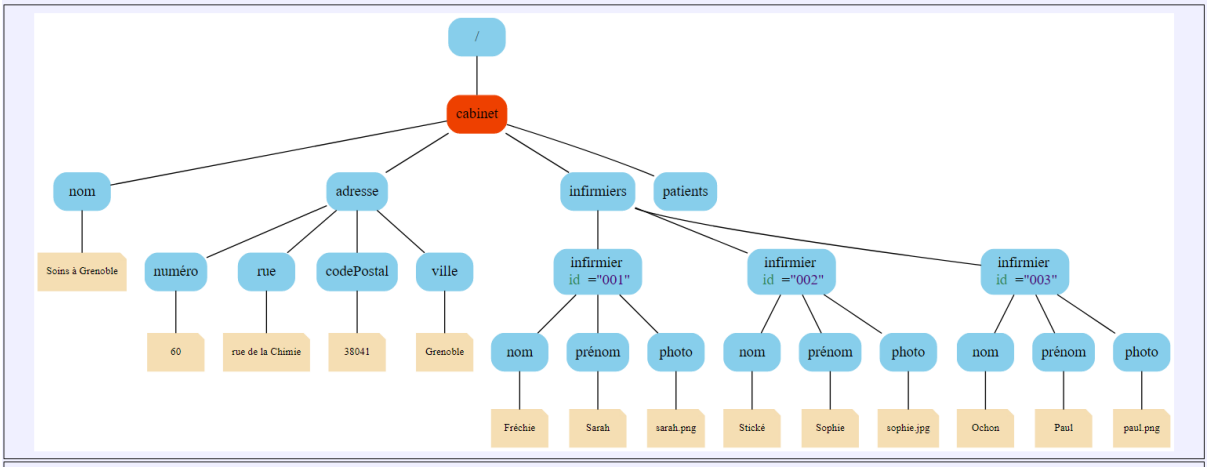
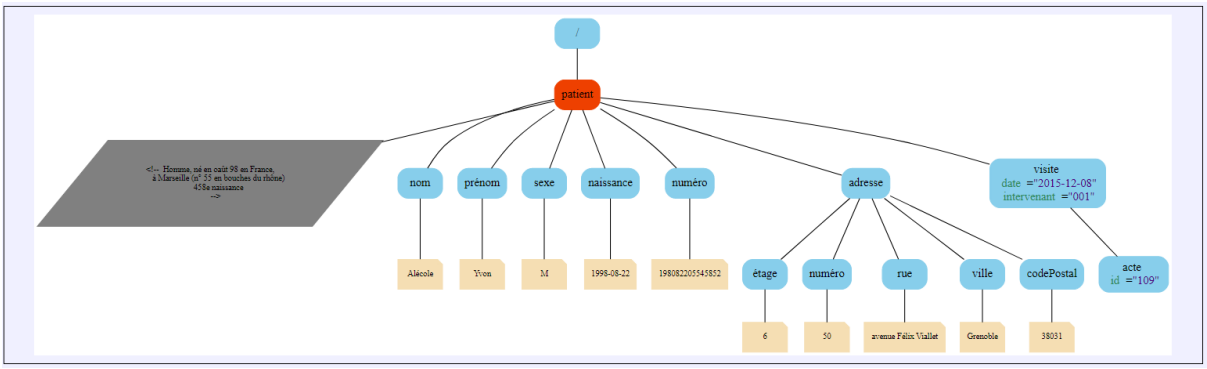


Figure 3.1. *Début du document XML stockant les informations du cabinet.*

La seule modification faite par rapport à cette arborescence est pour l'adresse. Il est demandé plus tard dans le projet d'inverser « codePostal » et « ville » de place afin que l'ordre soit le même que les adresses des patients présentées plus bas.

Dans cet arbre, il manque donc la partie « patients » que nous allons présenter tout de suite. La balise <patients></patients> est composé de plusieurs <patient></patient>. Un patient contient l'arborescence suivante :



Maintenant que nous avons expliqué comment est structuré notre cabinet.xml, nous pouvons montrer comment nous l'avons contraint à des règles et restrictions.

Cabinet.xsd :

Un xml schema permet de créer un schéma d'un fichier xml, c'est-à-dire que nous pouvons donner des règles et des restrictions afin à ce fichier afin que l'utilisateur ne puisse pas faire n'importe quoi.

Nous avons essayé de restreindre le fichier xml un maximum en commençant par mettre des minOccurs et maxOccurs pour chaque balise afin de pouvoir contrôler s'il peut y en avoir plusieurs, si elle est obligatoire ou alors facultative. De plus, nous avons créé de nombreuses restrictions sur des types notamment des string afin de pour voir jouer sur la taille et le pattern de celui-ci. Par exemple, nous avons créé un type NomPropre qui permet d'avoir un mot commençant obligatoirement par une majuscule. Ou alors un CodePostal qui est un string ne contenant que 5 chiffres. Nous avons décidé de prendre des string au lieu de int car nous avons plus de liberté sur la forme de restriction du string plutôt que le int. Voyons par exemple un numéro de sécurité sociale, il est composé de 15 chiffres est chacun de ces chiffres fait parti d'un groupe qui a une signification. Nous allons vous montrer le code de cette restriction car nous avons documenté la signification de chaque chiffre.

```
<!-- Définition du type Symple NumeroSS
        Restriction string :
            - taille : 15
            - que des chiffres
        Pattern :
            - Sexe : 1,2
            - Année de naissance : 2 derniers chiffres de l'année
            - Mois de naissance : de 01 à 12
            - Département de naissance : 2 chiffres ou 2A ou 2B
            - Commune de naissance : 3 chiffres
            - Numéro d'ordre : 3 chiffres
            - Clé de contrôle : 2 chiffres -->
<simpleType name="NumeroSS">
    <restriction base="string">
        <pattern value="(1|2)[0-9]{2}((0[1-9])|(1[0-2]))([0-9]{2}|2A|2B)[0-9]{8}"/>
    </restriction>
</simpleType>
```

Code du type NumeroSS fichier cabinet.xsd

Comme nous pouvons le constater grâce au pattern nous pouvons contrôler plus ou moins le string que l'on doit rentrer. Tous les autres types sont présentés dans le fichier.

Nous avons vu ce qu'était un fichier xml, un fichier xsd, il nous manque plus que le fichier xsl.

Cabinet.xsl :

Un fichier xsl permet de faire une transformation à partir d'un fichier xml. Tous ces types de fichier seront rangés dans le répertoire xslt du projet. Le cabinet.xsl nous permet, à partir du fichier cabinet.xml de créer un fichier html pour chaque infirmière affichant leurs informations ainsi que les visites programmées et leurs patients. Un exemple d'un de ces fichiers est présent dans le répertoire html sous le nom de cabinet.html.

Patient :

Enfin, nous voulons formaliser les données de chaque patient dans un fichier xml pour ensuite pouvoir créer un fichier html permettant d'afficher ces données.

Pour cela, nous commençons par faire une transformation de cabinet.xml pour créer un fichier patient.xml composé des informations de seulement un patient grâce à patient_xml.xsl. Ce fichier xml a un schéma patient.xsd fonctionnant de la même manière que cabinet.xsd. De plus, il reprend son vocabulaire. Ensuite, notre fichier xml est réutilisé par patient_html.xsl afin de créer la page html patient.html du patient concerné.

Dernier point, lorsque nous créons un nouveau fichier xml grâce à une transformation xsl, nous devons rajouter à la main 2 lignes dans la racine du nouveau fichier xml afin qu'il fonctionne avec notre schema xsd. Nous devons ajouter les 2 lignes suivantes (entre ** **) :

```
<patient **xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'**
         xmlns:act="http://www.ujf-grenoble.fr/l3miage/actes"
         xmlns="http://www.ujf-grenoble.fr/l3miage/medical"
         **xs:schemaLocation="http://www.ujf-grenoble.fr/l3miage/medical
../../xsd/patient.xsd"**>
```

Conclusion :

Pour finir, ce projet nous a permis de mettre la main sur les 3 technologies xml étudiés en cours (xml, xml schema, xslt). Nous pouvions accompagner nos pages html de fichier css afin de faire des présentations moins brutes que du html pur. Nous avons seulement créé un fichier style.css permettant de mettre des bordures à nos tableaux et n'avons pas passé plus de temps à soigner la présentation car ce qui était le plus important à nos yeux dans ce projet était la manipulation des différentes technologies xml. De plus, nous avons fourni un grand effort de documentation de nos codes afin qu'ils soient plus lisibles et compréhensibles. Enfin, les tout ce qui est présentation css et la partie web du projet pourra être fait de manière personnelle de notre coté lorsque l'on aura un peu plus de temps.