

M1 MIAGE

Projet XML

Dossier d'analyse

Erik HELSENS

Année universitaire 2008-2009

Table des matières

1. Rappel du sujet.....	3
2. La démarche d'analyse.....	4
3. Les structures utilisées (DTD).....	6
4. Feuilles de style (XSL).....	9
5. Utilisation du programme.....	11
5.1 Lancement.....	11
5.2 Le répertoire.....	12
5.3 L'agenda.....	15
Conclusion.....	18

1. Rappel du sujet

Projet XML - Multimédia

Il s'agit de réaliser un agenda électronique avec carnet d'adresses.

Cet agenda contient une interface permettant l'entrée de nouvelles adresses, de nouveaux RDV, cours.., la modification, la suppression des données existantes. Ces données seront rangées sous forme XML de manière à pouvoir les interroger, les modifier par programme, etc.

Les fonctionnalités précieuses dans cet agenda sont : la recherche par rubrique, multi-critères, etc., l'affichage de l'agenda, par semaine, par jour, etc.

Organisation

Ce projet comprend deux parties :

- Une partie XML :
 - o Contenant l'information et les transformations : XSL
- Une partie Multimédia
 - o Interface Flex ou Flash permettant l'affichage, l'animation, l'interaction.

2. La démarche d'analyse

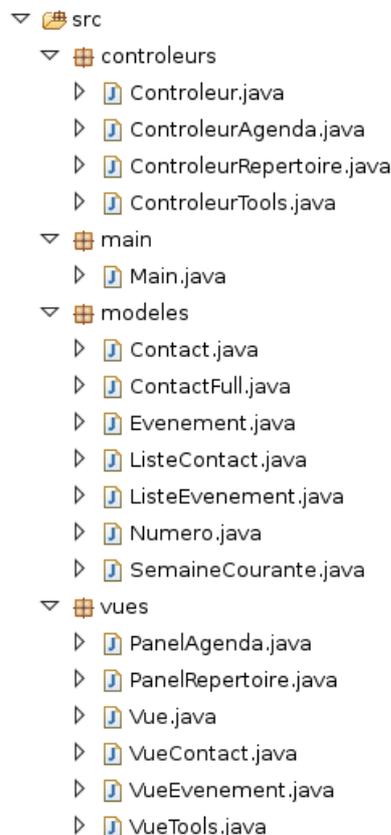
On peut tout d'abord distinguer deux fonctionnalités bien distinctes d'après le sujet qui sont l'agenda et le répertoire. On y associe un fichier xml pour chacune (« agenda.xml » et « repertoire.xml »).

Dans un premier temps je me suis consacré à la réalisation du répertoire et répété les mêmes étapes pour l'agenda.

La première étape fut de réaliser les structures (DTD), puis j'ai créé des exemples en xml afin de les valider et enfin j'ai fait les feuilles de style XSL (voir ces trois éléments en détail dans les paragraphes suivants).

Mon choix pour l'interface graphique s'est porté sur l'utilisation de Java avec eXist. L'application repose sur un modèle MVC (Modèle Vue Contrôleur) où l'aspect graphique, structure de données et fonctionnalités de contrôle sont séparés. Ceci permet une meilleure lisibilité du programme et les mises à jour sont possibles plus facilement du fait de la séparation claire du code.

Voici l'ensemble des classes utilisées:



Le package « controleurs » contient une classe « Controleur » principale, deux autres plus spécifiques pour chaque composante du programme (l'agenda et le répertoire) ainsi qu'une classe utilitaire « ControleurTools ».

Le package « main » contient la classe principale c'est à dire celle qui s'exécute en premier lors du lancement du programme.

Le package « modeles » contient les différentes classes associées aux structures de données du contact, de l'événement, des listes comportant ceux ci, du type « numero » contenu dans le champs téléphone d'un contact et une classe « SemaineCourante » permettant la gestion du calendrier de l'agenda.

Le package « vues » contient les classes d'affichage de l'agenda, du répertoire, de la vue regroupant ces derniers, de la vue d'ajout (et d'édition) d'un contact, celle d'un événement et enfin une classe utilitaire « VueTools ».

On voit clairement ici la séparation des deux fonctionnalités principales du programme qui sont l'agenda et le répertoire. Ils sont également graphiquement séparés en deux onglets distincts. Des captures d'écran sont disponibles dans la partie « utilisation du programme » de ce rapport.

Les requêtes sur eXist sont effectuées avec le langage Xquery. Les ajouts, édition et suppression d'éléments sont effectuées avec des méthodes DOM fournies par Java.

3. Les structures utilisées (DTD)

Le répertoire contient un nœud racine <repertoire> qui contient des nœuds <contact>.

Un contact possède un id en attribut qui permet de l'identifier et des fils <nom>, <prenom>, <rue>, <codepostal>, <ville>, <pays>, <telephone> et <email>. Le champ <telephone> contient des fils <numero> qui eux mêmes contiennent en attribut le type de numéro (parmi les valeurs 'mobile', 'fixe', 'travail', 'autre') ainsi que la valeur de ce numéro.

Les champs id, nom et prénom sont obligatoires, les autres sont optionnels. Un contact peut avoir jusque trois numéros et deux emails. Ces limites ont été choisies pour adhérer avec l'interface graphique.

L'agenda contient un nœud racine <agenda> qui a pour fils des nœuds <evenement>.

Un événement possède également un id en attribut qui permet de l'identifier, des fils <idcontact> (pour l'identifiant du contact auquel l'événement est associé), <date>, <heure>, <duree>, <objet> et <lieu>. Seuls les champs de durée et de lieu sont optionnels, les autres sont obligatoires. Tous les champs d'un événement sont uniques (il ne peut y en avoir plusieurs d'un même type).

J'ai donc choisi d'associer un contact pour chaque événement ce qui permet de filtrer les événements en fonction d'une personne dans l'interface.

Voici les DTD créées:

```
<!ELEMENT repertoire ( (contact)*) >
<!ELEMENT contact (nom,prenom,rue?,codepostal?,ville?,pays?,telephone?,email?,email?)>
  <!ATTLIST contact id CDATA #REQUIRED>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT rue (#PCDATA)>
  <!ELEMENT codepostal (#PCDATA)>
  <!ELEMENT ville (#PCDATA)>
  <!ELEMENT pays (#PCDATA)>
  <!ELEMENT telephone ( numero?,numero?, numero? )>
    <!ELEMENT numero (#PCDATA) >
    <!ATTLIST numero type (mobile | fixe | travail | autre ) "autre">
  <!ELEMENT email (#PCDATA)>
```

Fig 3.1: repertoire.dtd

```
<!ELEMENT agenda ( evenement* )>
<!ELEMENT evenement (idcontact, date, heure, duree?, objet, lieu?) >
  <!ATTLIST evenement id CDATA #REQUIRED>
  <!ELEMENT idcontact (#PCDATA)>
  <!ELEMENT date (#PCDATA)>
  <!ELEMENT heure (#PCDATA)>
  <!ELEMENT duree (#PCDATA)>
  <!ELEMENT objet (#PCDATA)>
  <!ELEMENT lieu (#PCDATA)>
```

Fig 3.2: agenda.dtd

Après la réalisation de ces structures, j'ai créé deux fichiers xml (agenda et repertoire) pour tester mes DTD et les valider.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="repertoire.xsl"?>
<!DOCTYPE repertoire SYSTEM "repertoire.dtd">
<repertoire>
  <contact id="7">
    <nom>Helsens</nom>
    <prenom>Erik</prenom>
    <rue>47 rue Jacquinot</rue>
    <codepostal>54000</codepostal>
    <ville>Nancy</ville>
    <pays>France</pays>
    <telephone>
      <numero type="mobile">0607646109</numero>
      <numero type="fixe">0951862712</numero>
      <numero type="autre">0329451751</numero>
    </telephone>
    <email>erik.helsens@gmail.com</email>
    <email>Erik.Helsens@etudiant.univ-nancy2.fr</email>
  </contact>
</repertoire>
```

Fig 3.3: Exemple de répertoire contenant un contact

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="agenda.xsl"?>
<!DOCTYPE agenda SYSTEM "agenda.dtd">
<agenda>
  <evenement id="1">
    <idcontact>7</idcontact>
    <date>30/04/2009</date>
    <heure>20:00</heure>
    <duree>inconnue...</duree>
    <objet>Projet XML</objet>
    <lieu>At Home</lieu>
  </evenement>
</agenda>
```

Fig 3.4: Exemple d'agenda contenant un événement

4. Feuilles de style (XSL)

Le répertoire se présente avec un sommaire qui contient les noms des contacts. Lorsque l'on clique sur l'un d'eux, on est redirigé, au moyen d'ancres html, vers les informations lui correspondant.

L'agenda est affiché de façon linéaire sous forme de liste d'événements où chacun d'eux est détaillé.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html" />

  <xsl:template match="/">
    <html>
      <body>
        <h1>Agenda</h1>
        <xsl:apply-templates select="/agenda/evenement"></xsl:apply-templates>

        </body>
      </html>
    </xsl:template>

    <xsl:template match="evenement">
      <ul>
        <li>
          <b>contact: </b> <xsl:value-of select="idcontact" /><br/>
          <b>date: </b> <xsl:value-of select="date" /><br/>
          <b>heure: </b><xsl:value-of select="heure" /><br/>
          <xsl:apply-templates select="./duree"></xsl:apply-templates>
          <b>objet: </b><xsl:value-of select="objet" /><br/>
          <xsl:apply-templates select="./lieu"></xsl:apply-templates>
        </li>
      </ul>
    </xsl:template>

    <xsl:template match="duree">
      <b>durée: </b><xsl:value-of select="." /><br/>
    </xsl:template>

    <xsl:template match="lieu">
      <b>lieu: </b><xsl:value-of select="." /><br/>
    </xsl:template>
  </xsl:stylesheet>
```

Fig 4.1: agenda.xsl

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html" />

  <xsl:template match="/">
    <html>
      <body>
        <h1>Répertoire téléphonique</h1>
        <xsl:apply-templates select="/repertoire/contact" mode="sommaire">
        </xsl:apply-templates>
        <br/>
        <xsl:apply-templates select="/repertoire/contact" mode="details">
        </xsl:apply-templates>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="contact" mode="sommaire">
    <a href="{concat('#contact', position())}">
      <xsl:value-of select="nom"/><xsl:text> </xsl:text>
      <xsl:value-of select="prenom"/>
    </a><br/>
  </xsl:template>

  <xsl:template match="contact" mode="details">
    <a name="{concat('contact', position())}"></a>
    <ul>
      <li><b>
        <xsl:value-of select="nom"/><xsl:text> </xsl:text>
        <xsl:value-of select="prenom"/>
      </b></li>
      <xsl:value-of select="rue"/><br/>
      <xsl:value-of select="codepostal"/><xsl:text> </xsl:text>
      <xsl:value-of select="ville"/><br/>
      <xsl:value-of select="pays"/><br/>
      <xsl:apply-templates select="./telephone"></xsl:apply-templates><br/>
    </ul>
  </xsl:template>

  <xsl:template match="telephone">
    <table>
      <xsl:apply-templates select="./numero"></xsl:apply-templates>
      <xsl:apply-templates select="../email"></xsl:apply-templates>
    </table>
  </xsl:template>

  <xsl:template match="numero">
    <tr>
      <td><b><xsl:value-of select="@type"/><xsl:text>: </xsl:text></b></td>
      <td><xsl:value-of select="."/></td>
    </tr>
  </xsl:template>

  <xsl:template match="email">
    <tr>
      <td><b><xsl:text>mail: </xsl:text></b></td>
      <td><xsl:value-of select="."/></td>
    </tr>
  </xsl:template>
</xsl:stylesheet>

```

Fig 4.2: repertoire.xsl

5. Utilisation du programme

5.1 Lancement

Le programme est fourni avec un exécutable .jar qui est un format d'archive propre à Java. On peut le lancer, après avoir démarré au préalable la base de données eXist, directement en double cliquant dessus. Dans ce cas, la collection doit être nommée 'M1MIAGE_HELSENS_ProjetXML' et doit contenir les deux fichiers repertoire.xml et agenda.xml fournis sinon un message d'erreur est affiché:

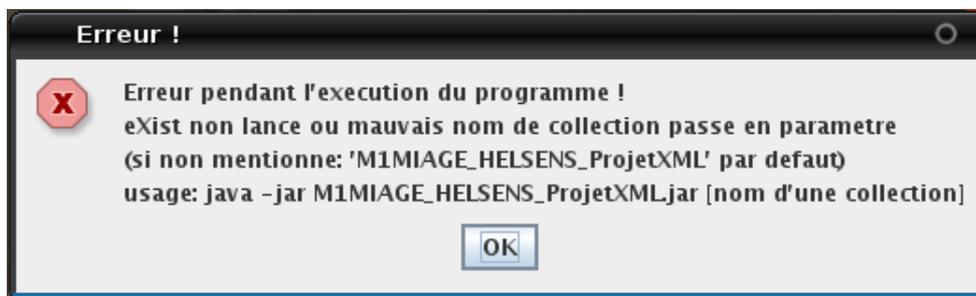


Fig 5.1: Message d'erreur

L'autre solution est de lancer le programme à partir d'un terminal (sous Unix) ou d'une invite de commande MSDOS (sous Windows) avec la commande suivante:

```
java -jar M1MIAGE_HELSENS_ProjetXML.jar [nom d'une collection]
```

Le nom de la collection est optionnel.

NB: sous Unix la commande `./M1MIAGE_HELSENS_ProjetXML.jar` (optionnellement suivi du nom de la collection) marche également.

Les sources étant fournies (ainsi que les fichiers de préférences), on peut également importer le programme sous Eclipse, qui est un environnement de développement pour Java, et l'exécuter directement depuis ce logiciel.

Lorsque le programme est lancé correctement, on obtient une fenêtre principale contenant deux onglets: un pour le répertoire premièrement affiché et un pour l'agenda.

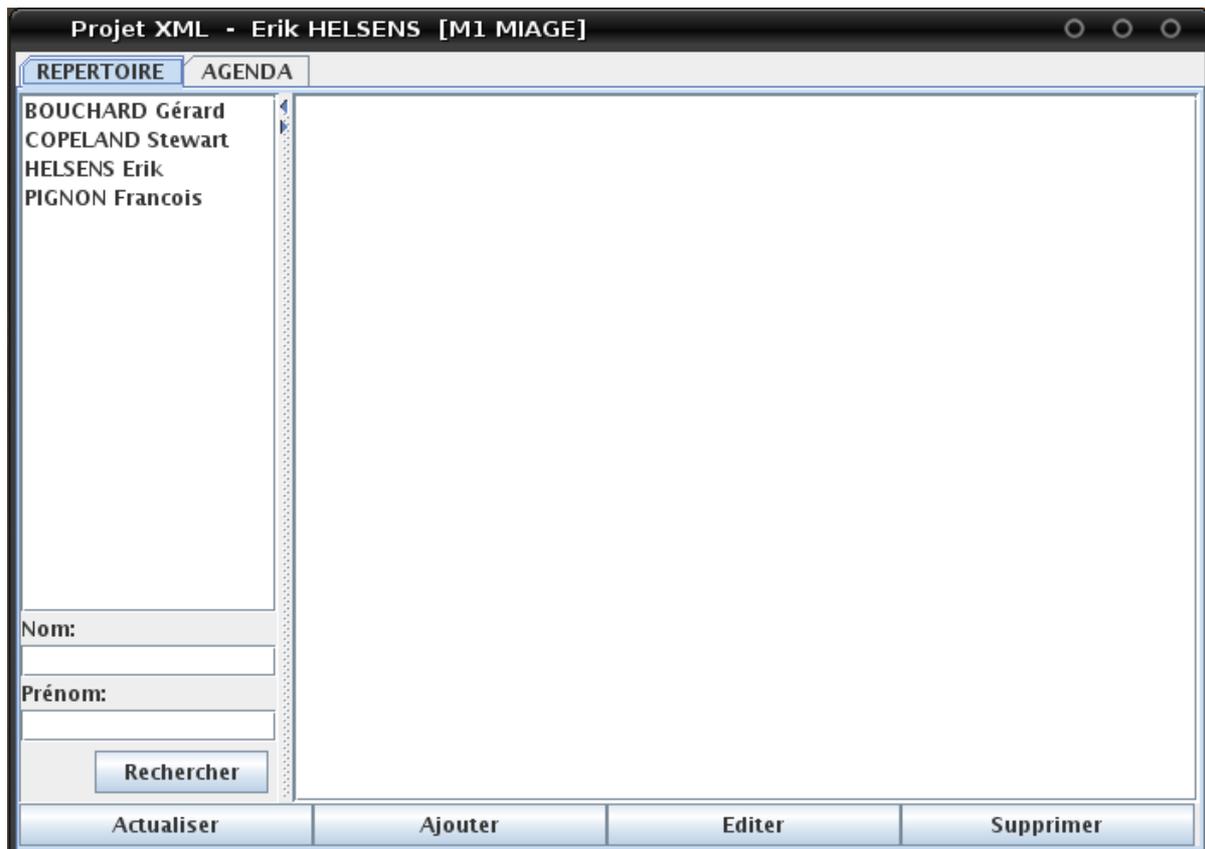


Fig 5.2: Démarrage

5.2 Le répertoire

La vue du répertoire contient à gauche la liste des contacts avec en dessous des champs de recherche sur le nom et le prénom d'un contact et à droite un champ d'affichage des informations du contact. En dessous de ces deux parties, on peut voir des boutons associées aux fonctions essentielles du programme qui sont l'actualisation de l'affichage, l'ajout, l'édition et la suppression de contact.

Pour voir les informations associées à un contact, il suffit de cliquer sur son nom dans la liste. On peut filtrer ces contacts en entrant les premières lettres du nom et/ou du prénom dans les champs prévus à cet effet (en dessous de la liste de contacts) puis en cliquant sur rechercher. Cela marche également avec des lettres contenues au milieu du nom, par exemple 'CHAR' pour 'BOUCHARD'. Pour retrouver la liste complète des contacts après une recherche, il suffit de cliquer sur le bouton actualiser.

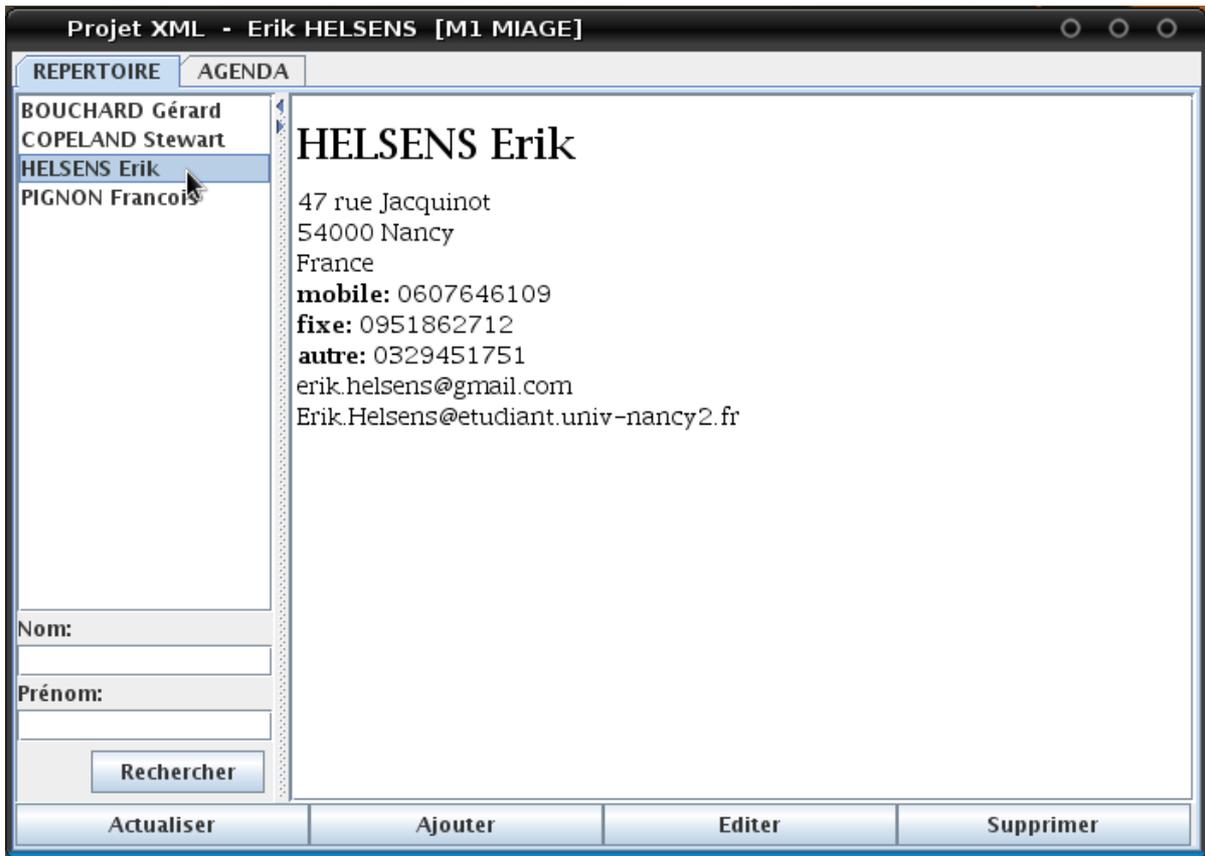


Fig 5.3: Sélection d'un contact

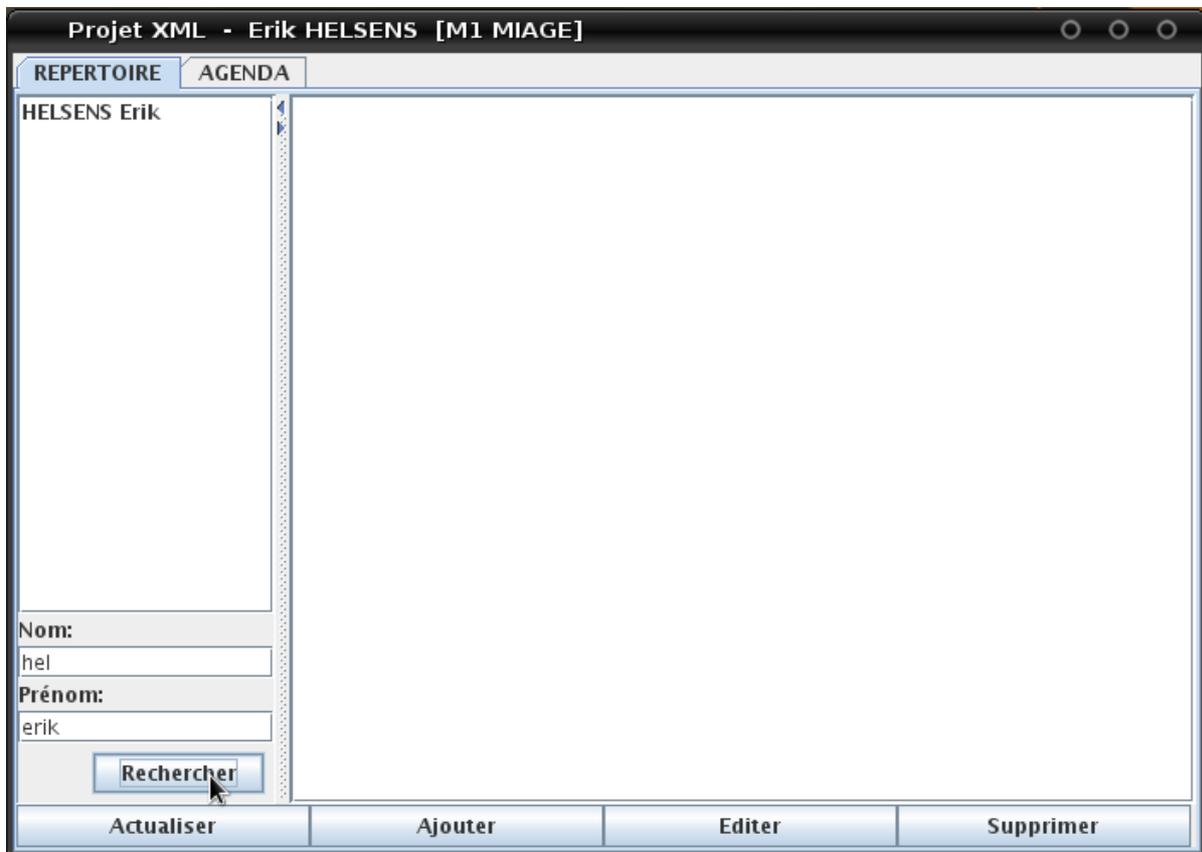
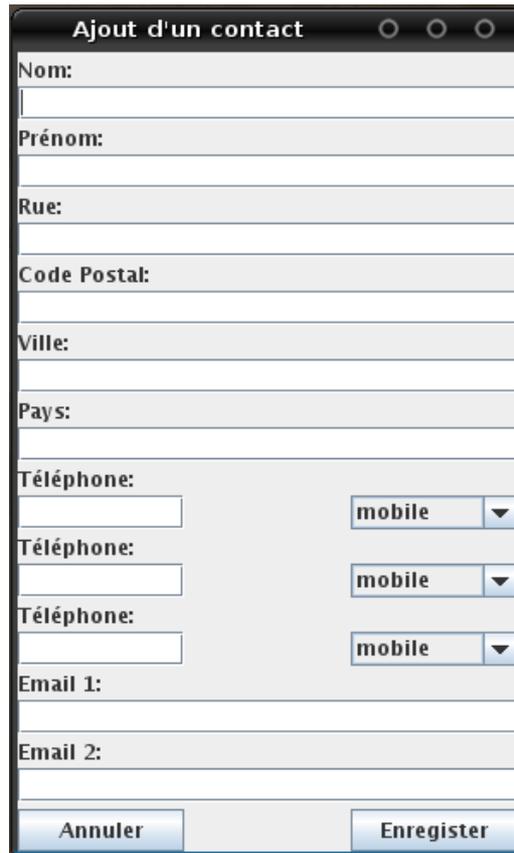


Fig 5.4: Recherche sur un contact

Pour ajouter un contact, il faut cliquer sur le bouton ajouter, on obtient la vue suivante:



The image shows a dialog box titled "Ajout d'un contact". It contains the following fields and controls:

- Nom: [text input]
- Prénom: [text input]
- Rue: [text input]
- Code Postal: [text input]
- Ville: [text input]
- Pays: [text input]
- Téléphone: [text input] with a dropdown menu set to "mobile"
- Téléphone: [text input] with a dropdown menu set to "mobile"
- Téléphone: [text input] with a dropdown menu set to "mobile"
- Email 1: [text input]
- Email 2: [text input]
- Buttons: "Annuler" and "Enregistrer"

Fig 5.5: Ajout d'un contact

Il suffit de remplir les champs texte, de sélectionner les catégories de numéro de téléphone, puis de cliquer sur le bouton Enregistrer pour valider.

L'édition de contact se fait en sélectionnant le contact désiré puis en cliquant sur le bouton Éditer, on obtient alors la même vue que celle d'ajout.

La suppression se fait de la même manière en sélectionnant le contact puis en cliquant sur le bouton supprimer. Une confirmation est alors demandée. On remarque que si un contact est associé à des entrées dans l'agenda, il est impossible de le supprimer car il y aurait des incohérences dans la base de données avec des événements affectés à des contacts n'existant pas. Un message d'erreur est affiché si le cas se présente.

5.3 L'agenda

On accède à la partie agenda en cliquant sur l'onglet portant son nom.

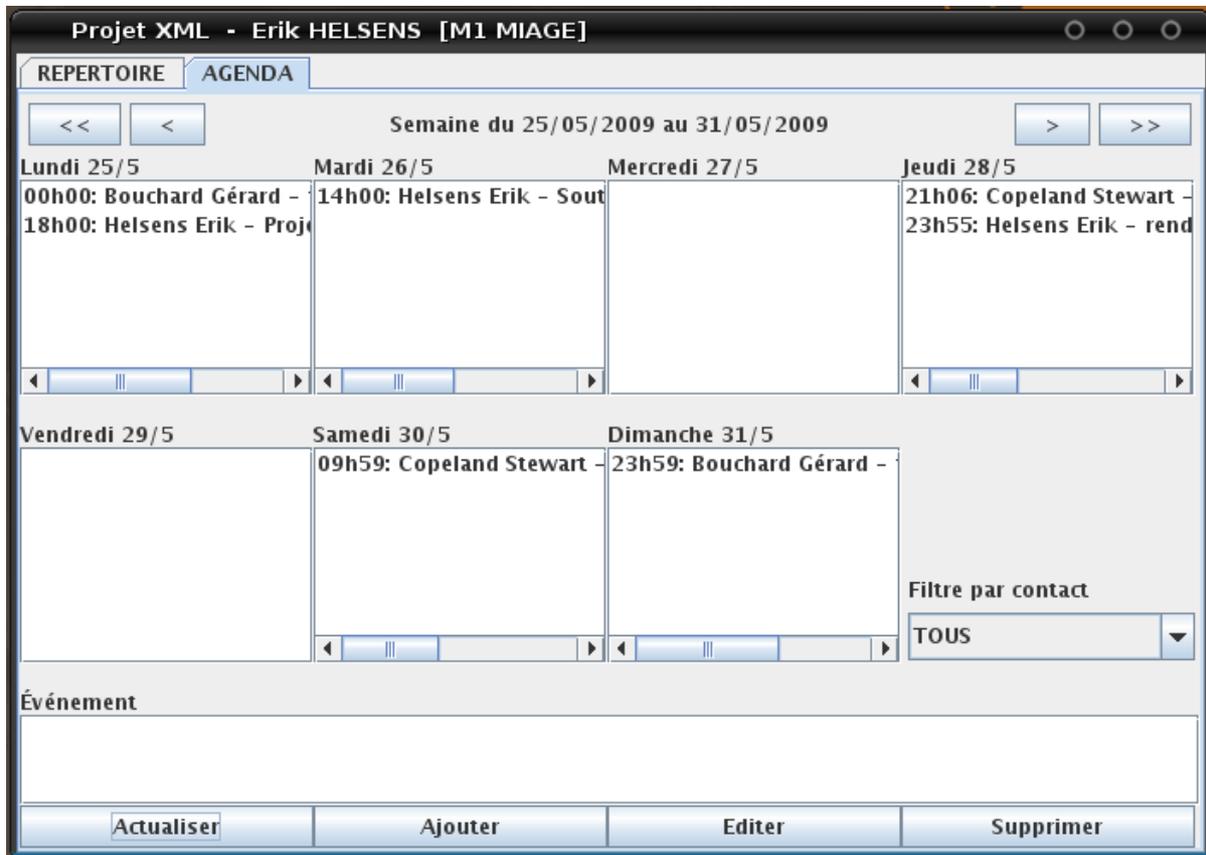


Fig 5.6: Vue de l'agenda

L'agenda est hebdomadaire et s'affiche sur la semaine en cours. La date du jour est affichée en bleu (si elle est dans la semaine courante affichée). On peut naviguer de semaine en semaine ou de mois en mois avec respectivement les boutons < > et << >>. Par défaut, les rendez vous de la semaine pour tous les contacts sont affichés. On peut les filtrer pour un contact particulier en sélectionnant ce dernier dans la liste déroulante en bas à droite de l'écran.

Pour obtenir toutes les informations d'un événement, on peut déplacer l'ascenseur horizontalement de la liste correspondante ou on peut cliquer dessus et ses informations s'afficheront dans le champs « Événement ».

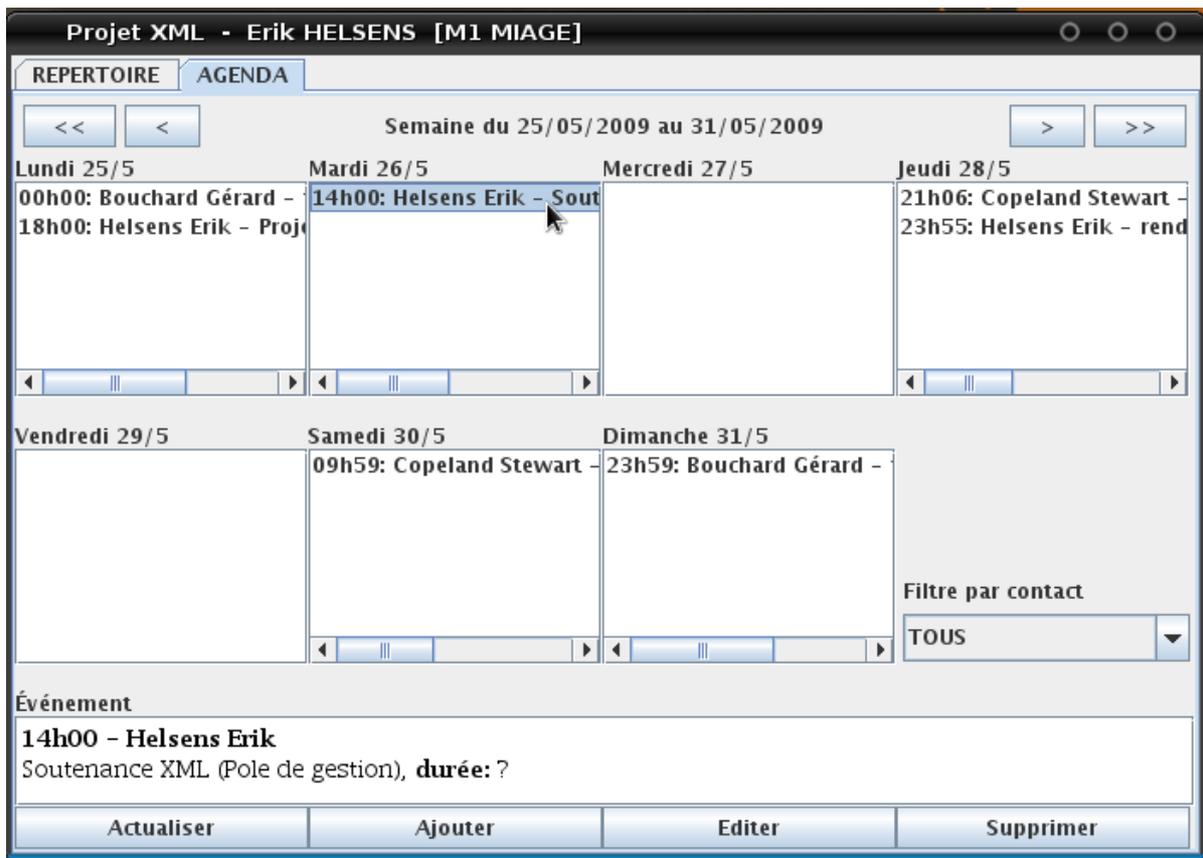


Fig 5.7: Sélection d'un événement

On remarque que les boutons situés en bas de l'écran sont les mêmes que dans la vue du répertoire dans un but de cohérence et de simplicité de l'application. Tout comme leurs homologues, ils correspondent aux fonctions d'actualisation, d'ajout, d'édition et de suppression d'un événement.

L'ajout et l'édition affiche la même vue contenant les champs associés à un événement. La suppression demande également une confirmation de la part de l'utilisateur.

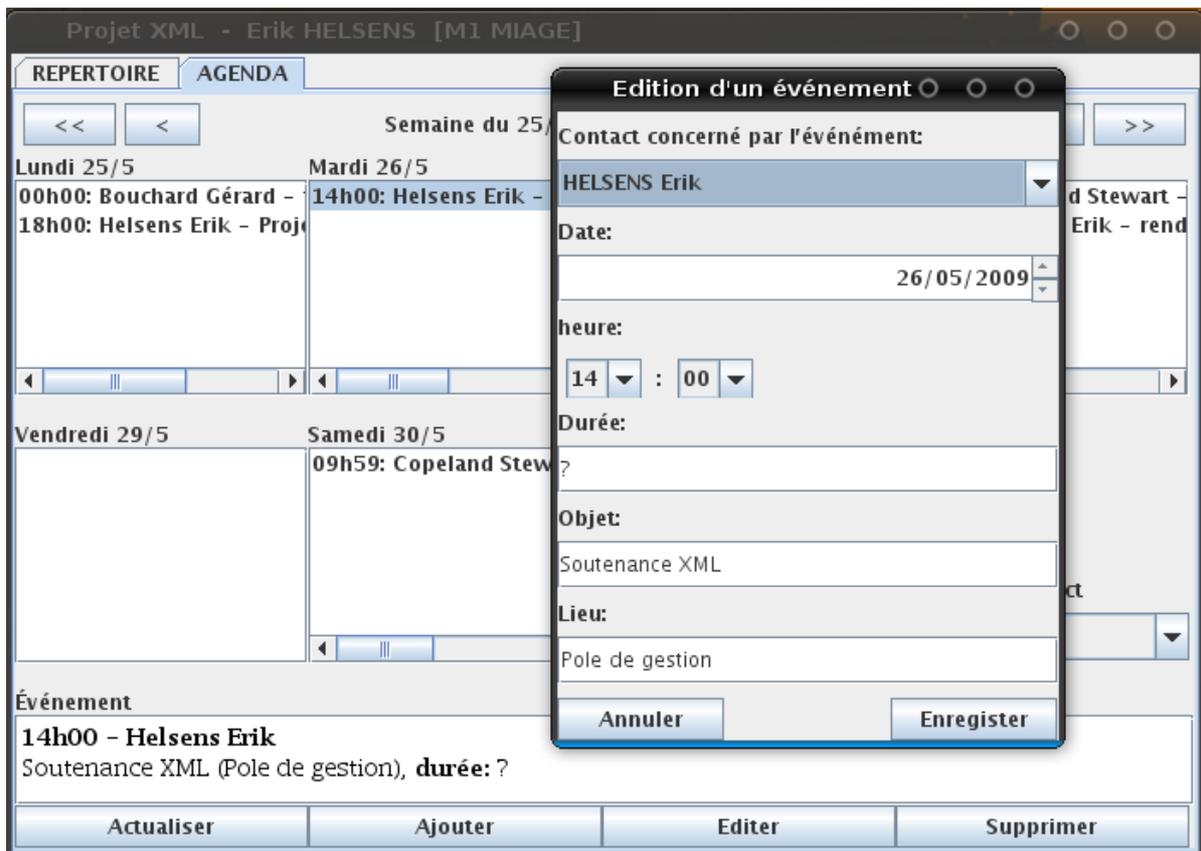


Fig 5.8: Édition d'un événement

Conclusion

Ce projet m'a permis, outre d'appliquer les connaissances vues en cours, de réaliser une véritable mini application Java interagissant avec une base de données basés sur des fichiers xml, chose que je n'avais jamais faite auparavant. On regrettera le manque de documentation pour eXist et son interaction avec les différents langages de programmation compatibles.

L'utilisation de XML permet une importation et une exportation aisée des données. Le langage Java permet le développement d'applications graphiques agréables et contient toutes les méthodes pour manipuler du XML et interagir avec eXist.