École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
4ieme année
2011 - 2012

Rapport projet collectif

Application d'aide à l'identification d'insectes nuisibles

Encadrants
Gilles VENTURINI
gilles.venturini@univ-tours.fr

Université François-Rabelais, Tours
Client
Ingrid ARNAULT
ingrid.arnault@univ-tours.fr
Damien MUNIER
munier.damien@aliceadsl.fr
Alexandre DEPOILLY
alexandre.depoilly@etu.univ-tours.fr

Équipe CETU INNOPHYT

Étudiants
Matthieu ANCERET
matthieu.anceret@etu.univ-tours.fr
Jérome HEISSLER
jerome.heissler@etu.univ-tours.fr
Julien TERUEL
julien.teruel@etu.univ-tours.fr
Martin DEMEULEMEESTER
martin.demeulemeester@etu.univ-tours.fr
Mickael PURET
mickael.puret@etu.univ-tours.fr
Simon FAUSSIER
simon.faussier@etu.univ-tours.fr
Zheng ZHANG
zheng.zhang@etu.univ-tours.fr
Zhengyi LIU
zhengyi.liu@etu.univ-tours.fr

DI4 2011 - 2012

Version du June 1, 2012

Contents

List of Figures

Introduction

Notre projet est un projet collectif à 8 personnes dans le cadre de notre cursus à Polytech'Tours. Le client de notre projet est l'équipe INNOPHYT, et plus particulièrement la responsable de cette équipe, Ingrid Arnault. Cette équipe fait partie de l'Université François Rabelais de Tours et elle est consacrée aux activités de valorisation et de recherche dans le domaine de la lutte anti-parasitaire durable. Gilles Venturini, professeur au sein de l'école Polytech'Tours, est notre encadrant de projet. De part sa bonne connaissance du projet, il est aussi un interlocuteur privilégié pour les aspectes techniques et opérationnels. Ce rapport est un guide technique, qui pourra servir à un développeur désireux de reprendre et de comprendre les applications existantes, mais aussi un guide utilisateur, qui permettra à l'usager de trouver les réponses à ses questions à propos du fonctionnement des logiciels.

Présentation du projet

L'objectif du projet est de réaliser deux applications : une application PC et une application mobile. L'application mobile permet, à partir du visuel d'un insecte, de déterminer grâce à une succession de question si celui-ci est nuisible ou non. L'application PC permet quand a elle de consulter et de modifier la base de données associée à l'application mobile, c'est-à-dire les questions, les réponses et les médias. <!– à compléter... –>

Analyse complémentaires

Nous allons présenter ici les modélisations et diagrammes UML réalisés à posteriori du cahier de spécifications. En effet, lors de la rédaction du cahier de spécification, nous n'avions pas encore modélisé la partie "intelligence" des applications.

3.1 Modélisation UML

Avant de développer l'application PC, nous avons pris soin de modéliser une structure fiable et efficace à même de réaliser toutes les fonctionnalités demandées et de pouvoir évoluer de façon simple.

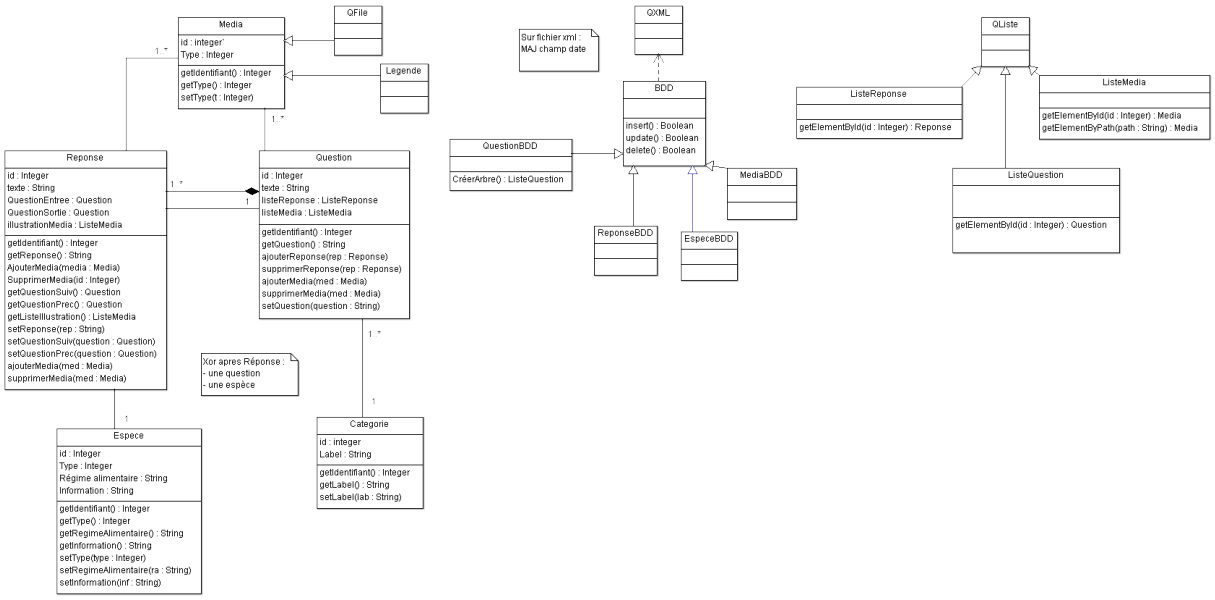

Figure 3.1: Diagramme de classes de l'application PC

3.2 Le fichier XML et sa DTD

Notre base de données est représentée sous la forme d'un fichier XML. Ce format a été privilégié car il est relativement simple à manipuler et assez performant pour de petites bases, comparativement à une base SQL/SQLite par exemple. N'ayant de toutes manières pas besoin de relation, d'intégrité et d'indexation, la choix du XML s'est imposé de lui-même.

En raison de nouveaux éléments apparus tout au long du projet (après la rédaction du cahier de spécification), la structure du fichier XML de données a évoluée. Par exemple, nous avons inséré un nouvel attribut "visible" sur les questions et les réponses suite à la demande du client de voir apparaitre ce critère dans l'interface (dans le but d'aider l'utilisateur de l'application). De plus, au fur et à mesure de l'avancement du projet, nous nous sommes rendus compte de quelques erreurs ou incohérence dans le

fichier XML. Nous avons donc du corriger ces problèmes, ce qui a engendré des modifications au niveau du code C++ de traitement du fichier XML.

Le fichier XML final est représenté sur la figure 3.2.

```xml
<?xml version = '1.0' encoding="UTF-8"?>
<!DOCTYPE arbre SYSTEM "arbre.dtd">

<arbre>
    <branche id="b1" type="accueil" date="01/06/2012">
        <question id="q1" texte="Premiere question ?" visible="true">
            <media>
                <legende>Astuces : regarder attentivement l'insecte</legende>
                <img src="images/insecte.jpg" />
                <audio id="aud1" src="images/test_audio.wav" />
            </media>
            <reponse id="r1" texte="Reponse 1">
                <media>
                    <img id="img1" src="images/insecte2.jpg" />
                    <img id="img1" src="images/insecte3.jpg" />
                    <legende>Ceci est le texte de la reponse 1</legende>
                </media>
                <branche id="b2">
                    <question id="q3" texte="Autre question" visible="false">
                        ...
                    </question>
                </branche>
            </reponse>
            <reponse id="r2" texte="Reponse 2">
                <media>
                    <img id="img1" src="images/insecte2.jpg" />
                    <legende>Ceci est le texte de la reponse 2</legende>
                </media>
                <branche id="b3" type="Arachnide">
                    <question id="q2" texte="Deuxieme question ?" visible="true">
                        <reponse id="r3" texte="Reponse 3">
                            <resultat id="res1">
                                <nom>Insecte 1</nom>
                                <type>MEL1</type>
                                <regimeAlimentaire>Predateur</regimeAlimentaire>
                                <informations>Informations insecte 1</informations>
                                <media>
                                    <img id="img2" src="images/insecte.jpg" />
                                </media>
                            </resultat>
                        </reponse>
                        <reponse id="r4" texte="Reponse 4">
                            <branche id="b2">
                                <question id="q6" texte="Encore une question ?" visible="both">
                                    ...
                                </question>
                            </branche>
                        </reponse>
                    </question>
                </branche>
            </reponse>
        </question>
    </branche>
</arbre>
```

Figure 3.2: Version finale du fichier XML

Nous avons bien sur corrigé la DTD associée à ce fichier XML. Cette DTD est le modèle, la grammaire, du fichier XML et permet donc de le valider. Elle indique les propriétés de chaque balise ainsi que

l'arborescence de notre fichier :

```
<!ATTLIST branche date CDATA #IMPLIED>

<!ELEMENT branche (question)>
<!ATTLIST branche id ID #REQUIRED>
<!ATTLIST branche type CDATA #IMPLIED>

<!ELEMENT question (reponse+, media*)>
<!ATTLIST question id ID #REQUIRED>
<!ATTLIST question texte NMTOKENS #REQUIRED>
<!ATTLIST question visible NMTOKENS #IMPLIED>

<!ELEMENT reponse ((branche|resultat), media*)>
<!ATTLIST reponse id ID #REQUIRED>
<!ATTLIST reponse texte NMTOKENS #REQUIRED>
<!ATTLIST reponse visible NMTOKENS #IMPLIED>

<!ELEMENT resultat (nom, type, regimeAlimentaire, informations, media*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT regimeAlimentaire (#PCDATA)>
<!ELEMENT informations (#PCDATA)>

<!ELEMENT media (img+, video+, audio+, legende+)>
<!ELEMENT img EMPTY>
<!ATTLIST img id ID #REQUIRED>
<!ATTLIST img src ENTITY #REQUIRED>
<!ELEMENT video EMPTY>
<!ATTLIST video id ID #REQUIRED>
<!ATTLIST video src ENTITY #REQUIRED>
<!ELEMENT audio EMPTY>
<!ATTLIST audio id ID #REQUIRED>
<!ATTLIST audio src ENTITY #REQUIRED>
<!ELEMENT legende (#PCDATA)>
```

Figure 3.3: DTD de notre fichier XML

Nous allons passer en revue les différentes balises utilisées dans cette DTD :

Balise <!ELEMENT>

La balise de type ELEMENT nous indique que nous aurons une nouvelle balise dans notre fichier XML. C'est ce que l'on appel un élément.

- **(question) :** cet opérateur nous indique que élément contiendra forcément un unique élément question
- **(reponse+) :** cet opérateur nous indique que l'élément contiendra au moins un élément reponse

- **(branche|resultat) :** cet opérateur nous indique que l'élément contiendra un élément branche ou d'un élément résultat
- **(nom, type) :** cet opérateur nous indique que l'élément contiendra un élément nom et d'un élément type
- **(media*) :** cet opérateur nous indique que l'élément contiendra 0 ou plusieurs éléments media
- **(#PCDATA) :** cet opérateur indique que l'élément contiendra une donnée, généralement un texte en relation avec le nom utilisé par l'élément
- **ANY :** cet opérateur indique que tout type d'élément est autorisé

Balise <!ATTLIST>

La balise ATTLIST nous indique que l'élément qui lui est relié aura des attributs. Les attributs sont des données indiquées à l'intérieur des balises dans le fichier XML. Elles donnent généralement des détails sur l'élément qui ne seront pas considérés comme des données.

- **#IMPLIED :** indique que l'attribut est facultatif pour l'élément
- **#REQUIRED :** indique que l'attribut est obligatoire pour l'élément
- **CDATA :** indique que l'attribut sera une chaine de caractères
- **NMTOKENS :** indique que l'attribut sera une chaine de caractères réduite (seul une partie des caractères sont autorisés)
- **ENTITY :** indique que l'attribut sera une entité externe qui ne sera pas analysée

3.3 Résultats et statistiques

Nous avons aussi, suite à plusieurs discussions avec le client, modifié le format de sortie des résultats. <!– mettre le nouveau format des données + explications –>

Application PC

4.1 Interface et utilisation

- Pseudo mode d'emploi / guide utilisateur de la partie PC - Explication de l'interface - Explication des boutons

Nous devions afficher une grande quantité de données à l'écran (questions, réponses, résultats et médias), sous une forme pas forcément évidente (arborescence). Nous avons donc conçu une interface sous forme de 4 bandeaux verticaux contenant chacun un type d'information. Cette disposition permet d'avoir accès à toutes les informations nécessaires en même temps à l'écran.

La figure 4.1 permet de voir l'interface principale de l'application au démarrage (les questions ne sont pas déroulées). On remarque les 4 bandeaux, qui, de gauche à droite, servent à : gérer les questions, gérer les médias associés à la question courante, gérer les réponses et les médias des réponses de la question courante et enfin d'afficher les images et les résultats.

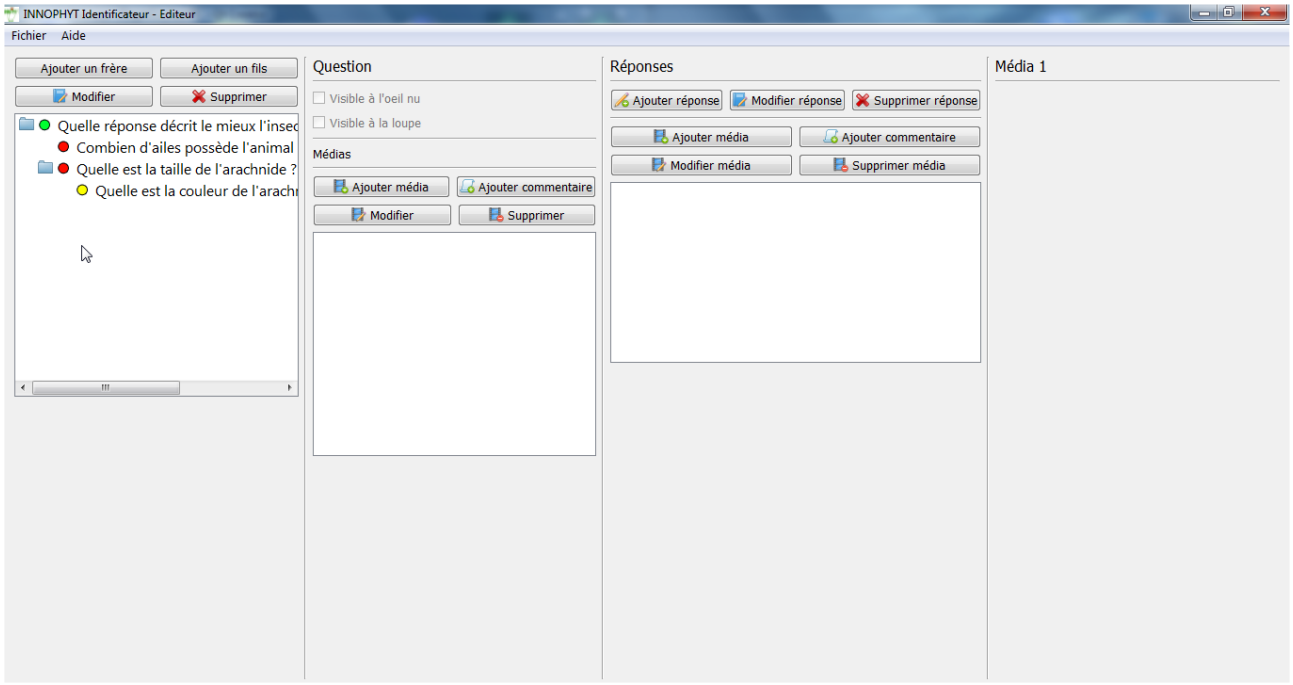

Figure 4.1: Interface principale de l'application au démarrage

Sur la figure 4.2, on peut voir que les bandeaux sont remplis. En effet, l'utilisateur ayant sélectionné une question, les médias et les réponses associés à cette question vont venir remplir les bandeaux correspondants. De plus, on voit qu'une image apparait dans la bandeau le plus à droite. C'est le résultat du clic sur le media "insecte.jpg" dans le bandeau des médias de la question. <!– Descriptif des boutons de l'interface –>

Lorsque l'utilisateur veut créer/modifier une question, il aura affaire à la fenêtre présente sur la figure 4.3. Cette fenêtre lui permet d'écrire le texte de la question et de choisir si cette question est visible à l'oeil nu, à la loupe ou les deux via les checkbox. En validant la fenêtre, il crée/modifie la question sélectionnée.

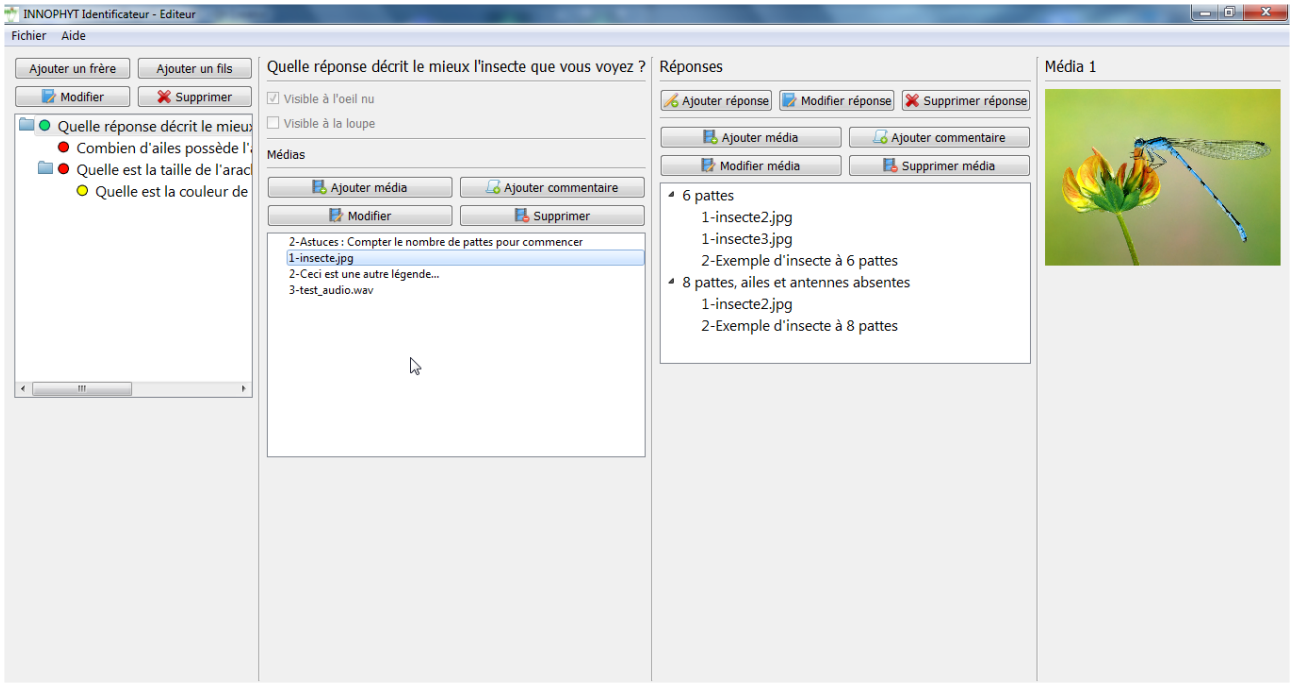

Figure 4.2: Interface principale après avoir sélectionné une question

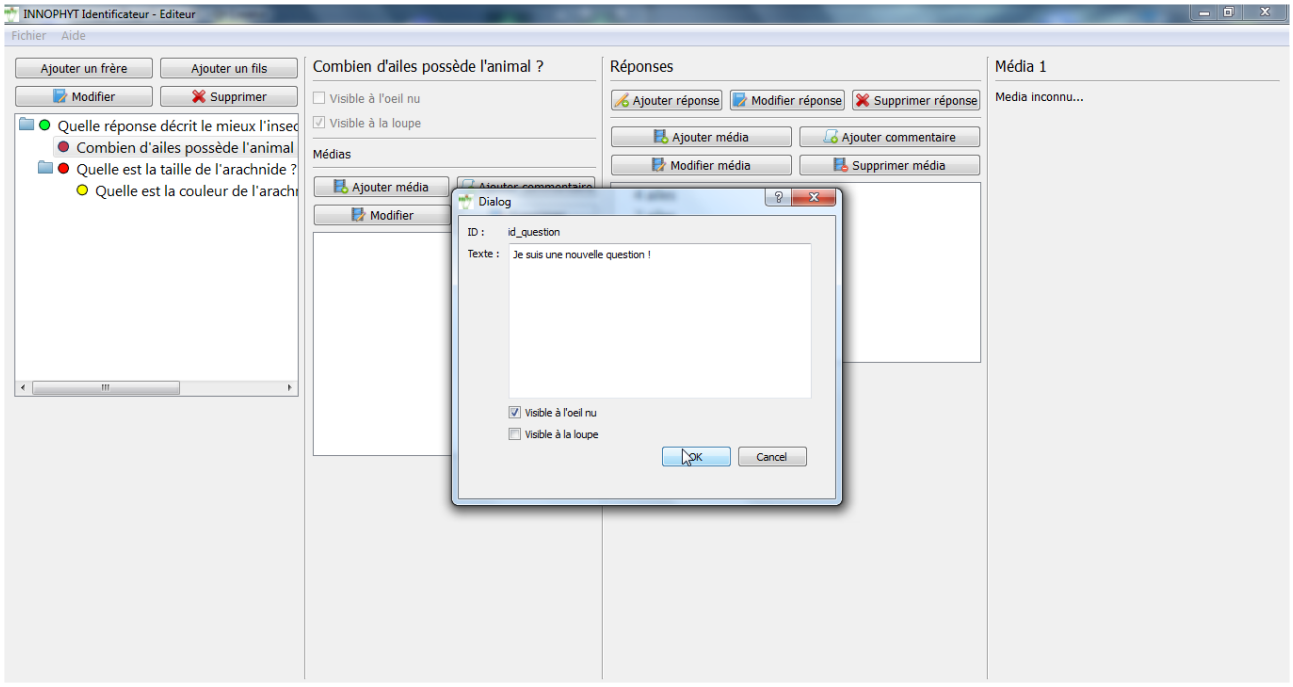

Figure 4.3: Boite de dialogue lorsque l'on veut créer/modifier une question

4.2 Module de lecture/écriture XML

Partie de Martin et Jérome

4.3 Mécanisme d'affichage des arborescences

Expliquer comment fonctionne les TreeView : mécanisme d'index, de coordonnées...

4.4 Visualisation des médias

Les questions, réponses et résultats de notre arborescence peuvent être accompagnés de médias pour aider l'utilisateur à se déterminer. Les 3 principaux formats de médias retenus sont les images, les sons et les vidéos. Nous avons donc développer 3 méthodes pour afficher ces médias.

4.4.1 Visualisation des images

Le média le plus utilisé et donc le plus important sera l'image. Pour intégrer une image dans une application Qt, il nous faut posséder son chemin (le "path"). Ce chemin est stocké dans les objets "Media" (attribut "path"). A partir de ce chemin, on peut créer un objet QImage. Pour vérifier que le chemin est correct et qu'il existe bien un fichier image à cet emplacement, il faut s'assurer que l'objet QImage obtenu n'est pas NULL. Ensuite, on peut éventuellement redimensionner notre image à la bonne taille avant de la convertir en QPixmap. Il suffit désormais d'utiliser cet objet QPixmap dans un QLabel pour voir s'afficher notre image dans l'interface de l'application PC.

```
QImage * myImg = new QImage("images/" + txtCurIdx);

if(myImg->isNull() != true)
{
    QImage myScaledImg = myImg->scaled(QSize(250, 250), Qt::KeepAspectRatio);

    QPixmap * img = new QPixmap();
    img->convertFromImage(myScaledImg, Qt::AutoColor);

    ui->labelImage1->setPixmap(*img);
}
```

Figure 4.4: Code permettant d'afficher une image

4.4.2 Lecteur audio

L'objectif était de pouvoir lire plusieurs types de médias et donc, parmi eux, se trouvaient les fichiers audio. Nous avons donc choisi de créer un lecteur intégré à l'application pour ne pas avoir à dépendre du système d'exploitation sur lequel on l'installera. Le lecteur est très simple. Il est constitué de 6 éléments :

- Un bouton PLAY,
- Un bouton PAUSE,
- Un bouton STOP,
- Une barre de défilement du temps,
- L'affichage numérique du temps actuel du média,
- Une barre de volume.

Après quelques recherches sur les librairies Qt permettant de gérer mes fichiers audio, nous avons choisi d'utiliser Phonon, une librairie qui permettait à la fois de gérer les fichiers audio et vidéo.

Pour programmer notre lecteur audio nous utiliseront différents objets de la librairie. Tout d'abord le "MediaObject" qui permet de recueillir toutes les informations du média entré en paramètre. Il sera donc l'origine de notre lecteur. Nous aurons ensuite les objets "SeekSlider" et "VolumeSlider" permettant de gérer, pour la première, le déroulement du temps grâce à une barre avec un curseur en mouvement et, pour la seconde, gérer le volume grâce à une barre similaire.

Après avoir relié ces trois objets entre eux dans le constructeur de notre classe "mainwindow", il nous suffit de connecter grâce à la méthode "connect()" l'action "clicked" des boutons aux actions "play()", "pause()" et "stop()" qui respectivement mettent le média en lecture, pause ou l'arrête en le réinitialisant au début. Il est aussi nécessaire de connecter le signal de changement de média, "currentSourceChanged(Phonon::MediaSource)" où "MediaSource" est le média entré en paramètre dans le "MediaObject", avec les slot "changerSourceVolume()", qui permet de modifier le volume du média en fonction de la barre de volume, ainsi que "changerSourceAvancement()" permettant de modifier la source de la barre d'avancement.

Enfin, il faut utiliser le signal ń tick() ż qui permet d'envoyer un signal à un temps régulier pour pouvoir actualiser le temps du média sur la barre ainsi que sur l'affichage LCD qui nous permet de visualiser le temps de la vidéo. Nous avons laissé le temps d'actualisation à sa valeur par défaut qui est d'une seconde soit 1000 millisecondes car le paramètre d'entrée de cette fonction doit être donné dans cette unité de temps. Grâce à ce signal envoyé toutes les secondes, nous pourrons effectuer le calcul du temps qui s'est écoulé dans le média grâce à une soustraction, effectuée dans la fonction "changerTemps()" entre le temps total, donné par la fonction "totalTime()" du MediaObject, et la durée restante, donnée par la fonction "remainingTime()" de ce même objet. Il ne nous reste plus qu'à faire un calcul pour passer cette durée en heures, minutes et secondes tout en rentrant ces données dans un objet QTime. On affiche alors le résultat dans le "LCD Number".

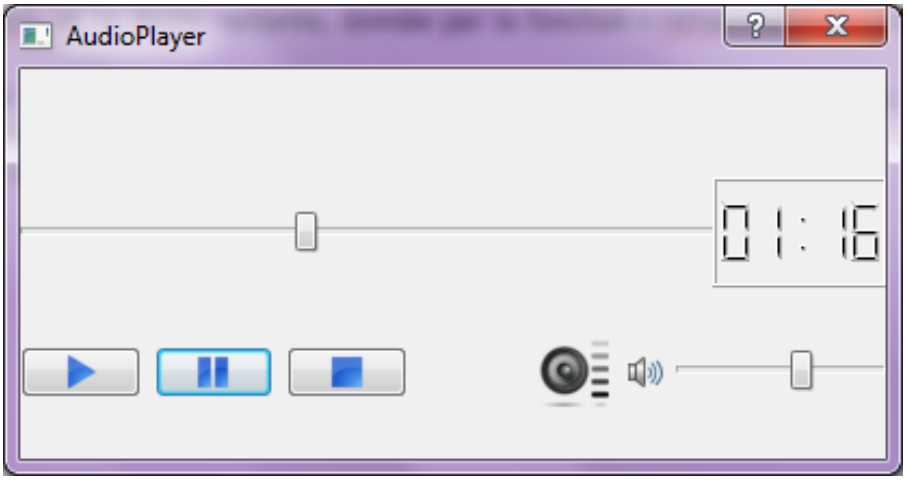

Figure 4.5: Capture d'écran du lecteur audio

4.4.3 Lecteur vidéo

Tout comme le lecteur audio, le lecteur vidéo devait comporter des boutons ainsi que des barres pour pouvoir gérer le média en lecture. Dans le cadre ce lecteur, nous avons simplement ajouté la gestion de la vidéo grâce à un écran pour visualiser les images.

Le lecteur vidéo gère les actions sur les médias de la même façon que le lecteur audio. Cependant, il faut lui ajouter la gestion des images. Pour cela, nous utilisons un objet "Phonon::VideoWidget". Celui-ci permet de créer une zone de visualisation de la vidéo. Il faudra ensuite relier notre MediaObject à

notre VideoWidget ainsi qu'à notre sortie audio (AudioOutput) grâce à la fonction "Phonon::createPath()" prenant en paramètre le MediaObject ainsi que la sortie. Il faudra donc utiliser cette fonction deux fois dans notre cas, une fois pour le son et une fois pour l'image.

Il est ensuite possible de paramétrer son lecteur vidéo avec différents paramètres pour modifier l'aspect de la zone l'affichage comme ajouter un filtre sur l'image. Dans notre cas, nous avons choisis de garder le filtre normal de toute notre fenêtre.

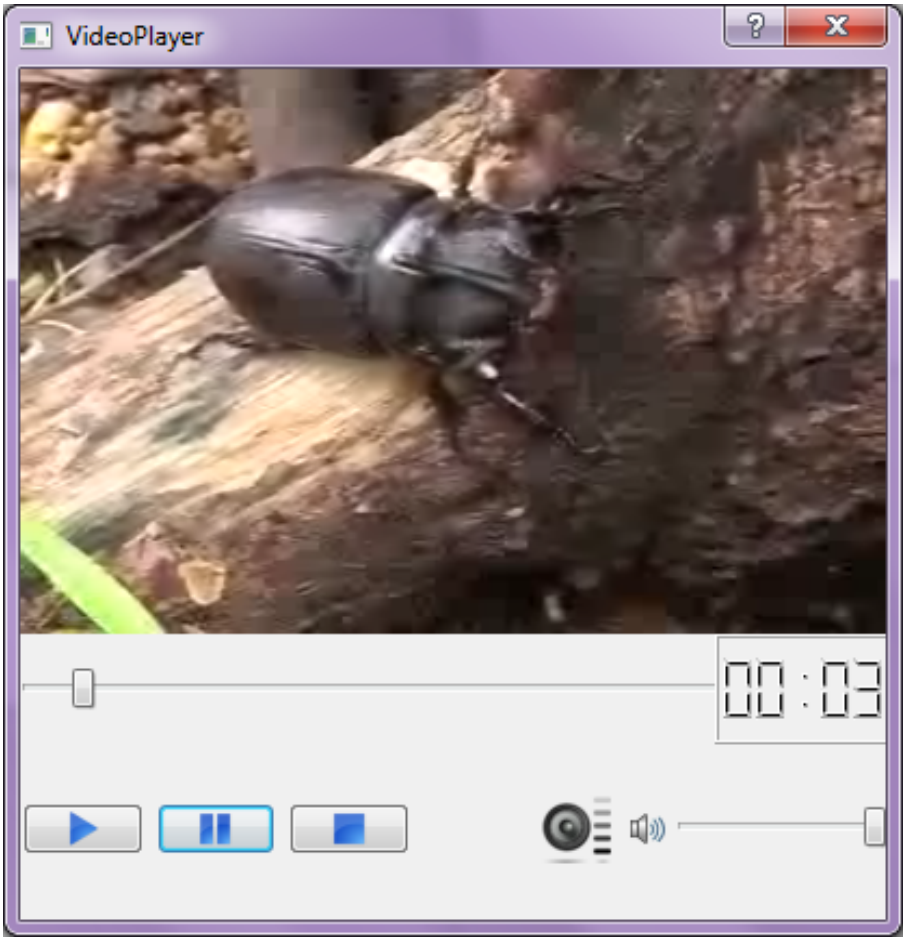

Figure 4.6: Capture d'écran du lecteur vidéo

4.5 Algorithme de coloration des questions

L'objectif de cette fonction était de modifier la petite bulle de couleur permettant de savoir s'il y a un nombre d'informations suffisantes ou non. Notre code couleur fut le suivant :

- Rouge : aucune information,
- Jaune : une information ou média,
- Vert : au moins deux informations ou médias.

L'algorithme de ce programme est directement inséré dans la fonction "peuplerListeQuestionsXML()" qui permet de ressortir toute la liste des questions. Dès qu'une question est récupérée du fichier XML, nous regardons le nombre de médias et informations insérés dans sa "listeMedia". L'objet "listeMedia" étant une liste chainée, il nous suffit de récupérer sa taille pour déterminer son nombre d'éléments. Ensuite nous

effectuons des tests suivant le nombre d'éléments détectés puis de modifier la couleur du "QStandardItem" correspondant à la question courante.

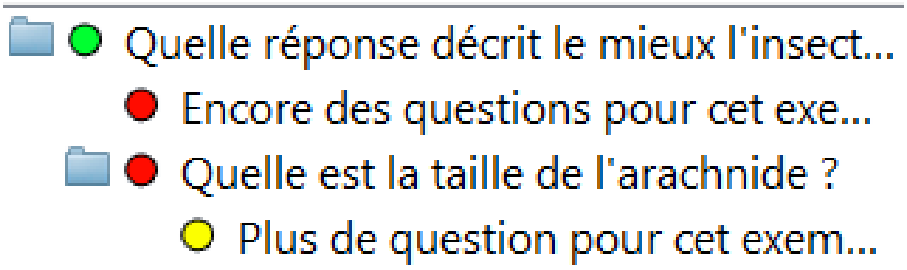

Figure 4.7: Coloration des questions selon la quantité d'informations présentes

Application mobile

L'application tablette doit permettre d'exploiter le fichier XML conçu via le logiciel bureau. Un utilisateur doit pouvoir, via une série de questions, associer l'insecte qu'il observe sous la caméra avec une morpho espèce de notre base de données. Pour cela il dispose d'aide textuelle, ou visuelle lui permettant de repérer des points clés sur les insectes permettant leurs identifications.L'application tablette est destinée à des personnes non scientifiques. Ceci a été très important pour nous puisqu'il a fallu concevoir une interface graphique adapté simple, sans terme technique particulier et tout en gardant en tête que la personne devait pouvoir se retrouver facilement sans aucune connaissance pré requises.
Les croquis de base de l'interface d'identification ont constamment évolué au cours du développement de l'application. En effet, nos manipulations fréquentes nous ont permis de nous rendre compte de certaines choses non pratiques notamment en temps de réponse, comme par exemple l'acquisition de la caméra. Les remarques de l'équipe Innophyt ainsi que de Mr Venturini, nous ont aussi poussées à constamment modifier notre maquette.

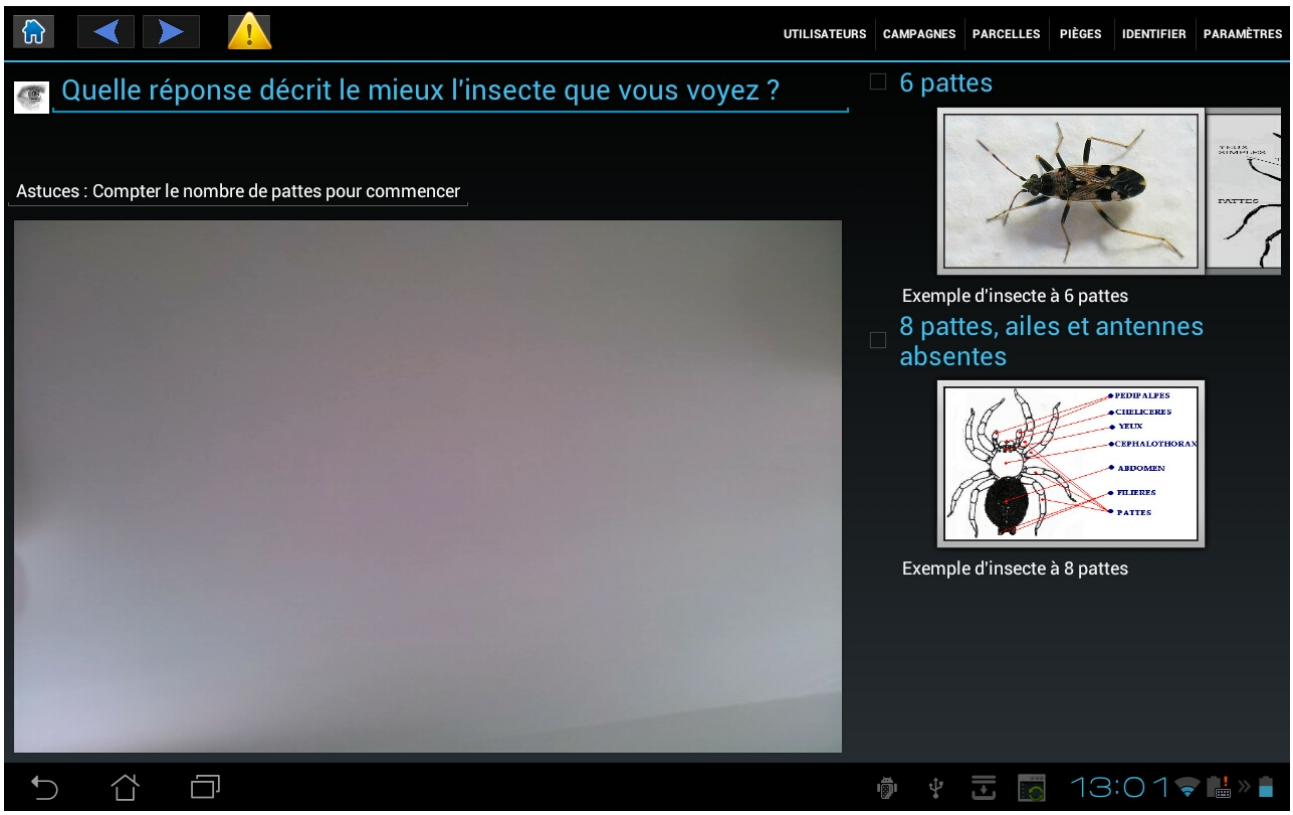

Figure 5.1: Interface d'identification d'un insecte - application mobile

5.1 Fonctionnalités

La partie identification de la tablette possède un certain nombre de fonctionnalités requises que nous allons détailler.

L'utilisateur voit directement dans une zone assez grande le flux vidéo en direct de la caméra arrière de l'appareil. Par une simple pression sur cette zone celui-ci sauvegarde l'image, pour pouvoir la comparer à d'autres photos par la suite. Juste au-dessus de cette zone, ce trouve la question de base de l'arbre XML avec une aide visuelle et textuelle au besoin. Sur la droite de l'application se situe la zone des réponses. Chaque réponse peut contenir une galerie de média, pouvant être scroller de droite à gauche et comparer à l'image sauvegardée via un long clique sur une des images de la galerie. Si l'espace occupé verticalement par les réponses est trop grand la zone sera rendu scrollable et pourra contenir autant de réponses que l'on souhaite.Le changement de questions et donc l'avancer dans l'arbre d'identification s'effectue en cochant une des réponses. Dans le cas où la question suivante de conviendrai pas à l'utilisateur, un historique permet de revenir en arrière sur ses choix.Lors de la mise en veille ou en arrière-plan de l'application celle-ci est sauvegarder dans son état et sera restaurer avec l'historique dans l'état dans lequel elle était. On peut quitter et reprendre l'identification à son aise.Lorsque nous sommes dans la partie identification un bouton d'alerte triangulaire orange apparait dans la barre d'action, permettant à tout moment de signaler que nous sommes bloqués dans le processus d'identification. Cela génère un rapport contenant des informations utiles comme un screen de l'écran utilisateur ou ses commentaires, pour analyse ultérieur.
Ceci est une présentation sommaire des fonctionnalités offertes, dans la prochaine partie nous allons rentrer plus en détails dans la partie technique.

5.2 La zone d'affichage

L'application possède deux zones importantes, l'action barre contenant un menu fixe, et le reste de l'écran en dessous qui change en fonction du besoin que l'on a. Cette dernière zone est gérée via des fragments. Un fragment est un bout d'interface graphique qui a son propre cycle de vie et qui peut être réutilisés et dupliquer en plusieurs objet différents. Ceci est plus complexe à gérer qu'un simple changement d'interface, mais est plus optimisé, stable, propre et très fortement recommandé par la documentation Android pour les applications sérieuses.

Voici quelques conseils et considérations sur l'utilisation des fragments :

1. Lors du onAttach conserver dans votre fragment une référence sur l'activity. Cela est essentiel notamment pour retrouver le fragentManager de l'activity dans notre QuestionFragment.
2. On ne peut accéder aux éléments définie dans un fichier de layout XML personnalisée que lors du onActivityCreated pas avant, difficilement après.
3. La veille via le bouton retour de l'application détruit le fragment contrairement à celle du menu home qui la met en pause et au bouton d'alimentation qui la met en position stop.
4. Un système de log complet à était mis en place il est très utile pour savoir ce qu'il se passe, penser à l'utiliser et le compléter au besoin.
5. Lors de l'ajout de plusieurs fragment à la suite via le code, il faut bien penser qu'un fragment doit obligatoirement être contenue dans un layout.

Au moment où l'on accède à l'interface d'identification nous chargeons dans l'espace principale le Fragment QuestionFragment. Celui-ci est responsable de beaucoup de chose :- Il récupère l'accès à la camera - Lit la question dont il a besoin dans le XML- Gère l'historique des questions parcourue

5.3 La caméra

La gestion de la caméra est assez délicate. En effet, si l'on passe outre l'intent de base fournie par Android, il faut tout redévelopper à la main. Ce que nous avons fait, il faut garder à l'esprit que la zone

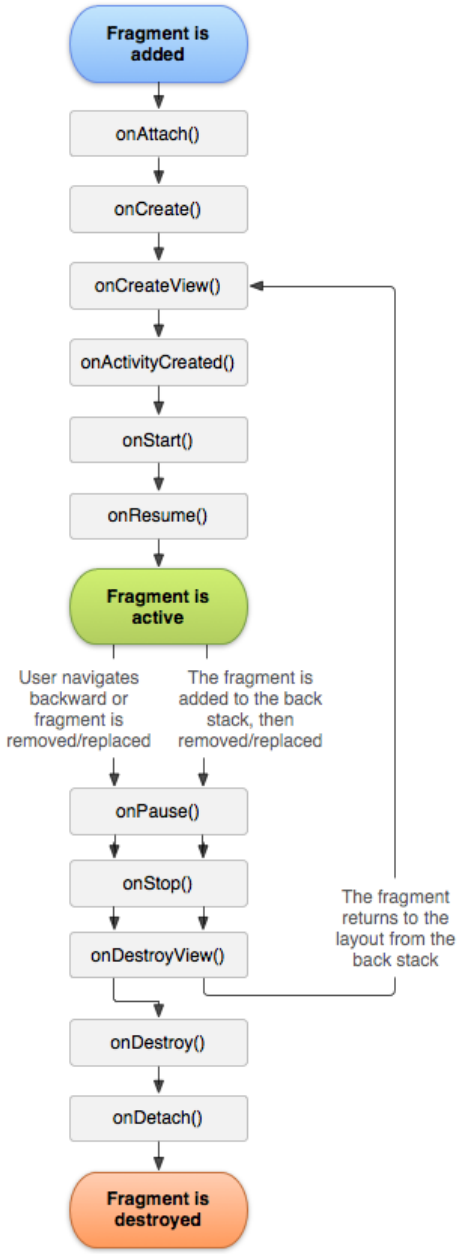

Figure 5.2: Schéma explicatif du fonctionnement des fragments

d'affichage quand elle est mise en pause ou détruite doit rendre l'accès à la caméra sinon, elle n'arrivera pas à la récupérer lors de sa sortie du mode pause. Et plus important aucunes autres applications, dont l'appareil photo de la tablette, ne pourront plus utiliser la caméra. **En cas de plantage de l'application sur l'acquisition de la caméra lors de test, killer l'application puis attendre un petit moment et essayer de lancer l'appareil photo de la tablette si celle-ci y arrive, reprendre les tests, sinon il faut redémarrer la tablette avant de continuer**.L'exemple fourni dans la documentation Android ne nous suffisait pas et faisait fréquemment planter l'application. Pour éviter cela nous avons ajouté un booléen couplé avec l'acquisition de la caméra pour ne pas pouvoir la rendre ou la capturer deux fois au système, ce qui causait ces crashs. Une fois ce problème d'acquisition résolue nous avons pu nous consacrer à la libération de la caméra. Quel que soit la mise en veille ou action tuant notre fragment ils passeront tous par l'étape en onPause du cycle de vie du fragment, c'est pourquoi la libération de la caméra s'effectue

à cet endroit. Même raisonnement pour le retour de mise en veille et la création du fragment ils passent tous par l'étape onResume c'est donc là que nous faisons notre acquisition de la caméra.

```
public Camera getCameraInstance() //Retourne null si la camera n'a pas pu etre acquise
{
    if( !aquis )   //Si la camera n'est pas deja acquise
    {
        Camera c = null;

        try
        {
            c = Camera.open();  //Obtention de la camera de base

            Camera.Parameters params = c.getParameters(); //Activation des paramètres de base de la caméra
            params.setWhiteBalance( Camera.Parameters.WHITE_BALANCE_AUTO );
            params.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);
            params.setAntibanding( Camera.Parameters.ANTIBANDING_AUTO );
            c.setParameters(params);

            aquis = true;
            mCamera.startPreview();
        }
        catch (Exception e)
        {
           Log.d("Camera", "la camera n'est pas accessible ou n'existe pas");
        }
        return c;
    }
    else
    {
        Log.d("Camera", "getCameraInstance camera non relaché" );

        return mCamera;
    }
}
```

Figure 5.3: Code permettant d'éviter les problèmes liés à la caméra

La caméra récupérer est maintenant récupérée et affichée, seulement l'image n'était pas bonne du tout, et cela est normale puisque nous gérons la caméra d'un point de vue bas niveau il faut lui spécifier dans ses paramètres tous les modules que nous voulons activer. Cela s'effectue au niveau de la fonction d'acquisition de la caméra.Nous avons activé trois modules :

1. La balance automatique des blancs
2. La mise au point automatique
3. Et l'antibanding qui essaye d'atténuer l'effet de bande horizontal, comme lorsque qu'on filme un écran de télé avec un caméscope.

Maintenant reste à expliquer un point qui fût particulièrement problématique. Lors de la mise en veille, via la touche power, de la tablette et de sa remise en route nous retrouvions l'application dans l'exact même état qu'avant à l'exception de la zone vidéo qui était figée sur la dernière image, vue avant la mise en veille.Nous avons cherchons longtemps, en partant notamment sur des fausses pistes comme une mauvaise acquisition ou libération de la caméra, finalement cela venait du bout de code donner par Android pour commencer. En effet, nous devons définir un nouveau type de zone pour pouvoir afficher le flux de la caméra sur cette zone. Ce bout de code nous est fournie par la documentation Android et nous l'avons considérer trop longtemps comme acquis sans y regarder de plus près. Et lorsque nous l'avons fait nous nous sommes rendu compte qu'il manquait tout simplement une fonction de rafraîchissement de cette zone. Pourquoi il y a besoin de rafraîchir la zone uniquement lors de la mise en veille de la tablette et non lors de la mise en veille de l'application cela reste pour le moment un mystère.

```
public void update( Camera cam )
{
    mCamera = cam;

    try
    {
        mCamera.setPreviewDisplay(mHolder);
        mCamera.startPreview();
    }
    catch (Exception e)
    {
        Log.d("tag", "Error starting camera preview: " + e.getMessage());
    }
}
```

Figure 5.4: Mise à jour de l'objet caméra et rafraichissement de la zone de visualisation de la classe CameraPreview

5.4 Les bitmaps

Toutes les images affichées dynamiquement dans l'interface graphique sont des bitmaps. Cela soulève plusieurs problèmes. La taille des images chargées provoque quasi instantanément un dépassement mémoire, provoquant un segfault lors de l'affichage de l'image suivante. Ce problème nous a demandé beaucoup de travail, car il est plus ou moins rapide à survenir. Au départ le chargement d'une seule image provoquait l'erreur, puis au bout de cinq ou plus. Nous avons mis en place deux procédures pour éviter ce problème. Nous libérons dès que possible les objets de type bitmap pour éviter de surcharger le système. Cela se fait via l'appel à la fonction recycle des objets de type bitmap. Conséquence direct de cette procédure nous avons dû rajouter dans la classe ImageAdapter un ń destructeur ż, ce qui normalement ne se fait pas en java.

```
public void finalize()
{
    for( int i = 0; i < bitmaps.length; i++ )
    {
        bitmaps[i].recycle();
    }
}
```

Figure 5.5: Destructeur de la classe ImageAdapter

Nous avons aussi dû demander au système de réduire la taille des images enregistrées en mémoire d'un facteur de 1/8 (valeur recommandé par plusieurs utilisateurs ayant rencontrés le même problème et confirmer par nos soins) diminuant ainsi grandement les chances de dépassement mémoire.

5.5 L'historique

La gestion de l'historique des questions n'a pas posé de problème mais nécessite d'être expliqué.Au démarrage du fragment, le parsage du fichier XML est lancé, il nous renvoie la première question que nous stockons dans la classe sous le nom de currentQuestion. Une fois cette question récupérer nous appelons la fonction changeUI qui adapte l'interface graphique du fragment à la currentQuestion. Le but du jeu est donc de changer en fonction de la position dans l'historique la currentQuestion et de rappeler changeUI. Pour cela nous avons trois éléments :

1. Un vecteur contenant toutes les questions vues par l'utilisateur listQuestion
2. Un vecteur contenant toutes les réponses choisies par l'utilisateur listReponseChoisi
3. Un int navigation qui indique la position dans le vecteur listQuestion de l'historique

Nous pouvons en déduire que lorsque navigation = listQuestion.size()-1 nous sommes à la fn du vecteur listQuestion est donc nous sommes sur la dernière question vue pas l'utilisateur, une question à laquelle il n'a pas encore répondue.Nous devons donc activer la flèche de retour en arrière tant que navigation != listQuestion.size()-1 et navigation >= 0. Si navigation atteint 0 c'est que nous sommes au début de l'historique il faut désactiver la flèche de retour dans l'historique.Même principe pour la flèche suivant de l'action barre, si navigation >= 0 et navigation != listQuestion.size()-1 alors il faut activer la flèche sinon la désactiver0.
Pour obtenir la question suivant de la question actuelle il suffit d'utiliser la classe ReacherQR responsable du parsing du fichier XML.Dans le cas où la question n'est pas une question finale, et mène donc à une autre question cela ce résume à :

```
currentQuestion = reacher.findQuestionById( reponse.getIdQuestionSuivante() );
listQuestion.add( currentQuestion );
```

Figure 5.6: Code pour passer à la question suivante

Il faut bien penser à gérer le cas où l'utilisateur revient en arrière dans l'historique et fait un choix différent. Dans ce cas il faut supprimer toute la suite de l'historique à partir du point changé.
Maintenant il faut pouvoir retrouver et cocher la checkbox de l'ancienne réponse que l'utilisateur a choisie, parmi toutes les réponses que possède la question. Pour cela nous conservons toutes les réponses à la currentQuestion dans un vecteur listReponseFragment. Cela nous permet de le parcourir et par comparaison de retrouver et de cocher la checkbox de l'ancienne réponse coché.

5.6 JDom

cf. rapport de Zhengyi LIU

Aspect gestion de projet

Pour coller au mieux aux attentes du client, nous avons essayé d'organiser le plus souvent possible des réunions de travail avec le client. Ces réunions avaient pour but de discuter des choix d'ergonomie des applications et de présenter notre travail. Cela a permis à l'équipe INNOPHYT de suivre notre travail au plus près, et de relever les erreurs ou les incohérences au plus tôt, afin que nous puissions les corriger au plus vite.

Pour les aspects plus techniques, nous organisions des bilans réguliers avec notre encadrant de projet, Gilles Venturini. Notamment, pour valider les étapes essentielles du projet : le format des données, le contenu du fichier XML, certains points de l'interface, certaines fonctionnalités critiques...

Enfin, au sein même de l'équipe, nous avons essayé d'organiser des points le plus régulièrement possible. Ces points permettaient de partager les difficultés rencontrées, de connaitre l'avancement des membres, de savoir ce qu'il restait à faire... Nous faisions un point par semaine, de préférence en début, qui durait en moyenne 10 à 15 minutes.

6.1 Témoignage de l'équipe

Ayant voulu exprimer au mieux l'ambiance de l'équipe ainsi que le ressenti de chacun à propos du projet, nous avons décidé de publier un florilège des témoignages de chaque membre de l'équipe. Ceci est plus complet qu'un bilan synthétique écrit par une seule personne tout en restant assez informel et quelque peu personnel. En complément de ces 8 témoignages, vous trouverez aussi les fiches d'auto-évaluation qui ont été remplis personnellement par chacun (fiche manuscrite qui seront remises lors de la soutenance).

> Dans un premier temps, je trouve que ce projet a été correctement mené. Pour notre premier projet avec une équipe de huit personnes, c'est vraiment correct et digne d'un projet de futur ingénieur. On se rend compte des problèmes qui nous arriveront dans notre future vie professionnelle. On remarque que les personnes peuvent s'investir plus ou moins, que les retards sont fréquent... C'est donc une très bonne expérience. Cependant, je suis un peu déçu du travail que j'ai pu fournir et de ce que j'ai pu apprendre. En effet, la majorité du projet, j'ai dû travailler avec Jérôme. C'est quelqu'un d'intelligent et de bon. Malgré cela, je pensais pouvoir en apprendre de lui mais malheureusement cela ne s'est pas passé ainsi. De ce fait, j'ai du me débrouiller avec ce qu'il faisait quand il le voulait, et je ne pouvais pas vraiment avancer. Il aurait fallut tout reprendre mais c'était trop tard pour tout recommencer. Avec le recul, il aurait fallut que je sois seul sur une partie, et avancer par moi-même. Je retiens ainsi de cette petite histoire pas mal de choses. En effet, cela nous permet de voir qu'il est souvent difficile de s'adapter à la personne avec qui l'on travaille. Il faut prendre des décisions, rapidement, et elles doivent être bonnes. Pour conclure, je retiendrais une bonne expérience tout de même étant donné que ce projet m'aura permis de voir comment se passe un projet avec un nombre de personne assez élevé. Je tiens à féliciter Matthieu qui je trouve a bien géré la situation, ce qui n'est pas évident. Nous avons au final une application fonctionnelle et vraiment agréable.

Martin DEMEULEMEESTER — *statut*.

> blabla...

Julien TERUEL — *statut*.

blabla...

Mickael PURET — *statut*.

Ce projet a été très constructif pour moi. Il m'a permit de découvrir de nouvelles technologies comme la gestion de différents types de médias sous Qt grâce à la librairie Phonon. Il m'a aussi fait de réutiliser des connaissances déjà acquises auparavant comme la création de fichiers XML et d'une DTD. Ce projet nous a mis dans des conditions réelles de projet, avec un client, des objectifs précis, un cahier des charges ainsi qu'une équipe de développement contenant un chef de projet. Celui-ci s'est très bien débrouillé en créant différentes tâches en fonction du cahier des charges et en les répartissant en équipes de deux personnes. Grâce à ces conditions, nous avons aussi pu réaliser les diverses difficultés d'un projet d'équipe comme les délais à respecter ainsi que d'apprendre de nouvelles technologies indispensables au projet.

Simon FAUSSIER — *statut*.

blabla...

Jerome HEISSLER — *statut*.

blabla...

Zhengyi LIU — *statut*.

Je pense que notre groupe est très bon. Tout les membres ont leurs caractéristiques et peuvent faire les tâches qui correspondent à leurs compétences. Sur les séances de projet, le taux de présence est proche de 99%. C'est très bon d'avoir fait le projet ensemble. Pour le chef de projet, il a proposé un bon planning pour notre groupe. Il a bien organisé l'avancement du projet et il a bien communiqué avec l'encadrant, le client et les membres de groupe. Je pense qu'il est un bon chef. Pour ma part, ce projet m'a permis d'apprendre beaucoup de choses : structure d'un ficher XML, gestion du XML en Java (lecture/écriture) et développement sous Android. Je suis très content d'avoir fait un projet dans ce groupe. Je regrette de ne pas avoir communiqué suffisamment avec mes camarades français, à cause de mon français et de mon caractère timide. C'est un défaut que je dois modifier pour la suite des mes études en France.

Zheng ZHANG — *statut*.

Ce projet m'a permis d'acquérir une première expérience de "chef de projet". J'ai été confronté aux difficultés de gérer une équipe sur une équipe assez longue (environ 3 mois) et de gérer les demandes parfois fluctuantes du client. C'était un projet très intéressant. Il m'a permis d'allier du développement d'application "desktop" en C++ et du développement d'application mobile en Java, domaine et langage qui me passionne. Je suis juste un peu déçu ne pas avoir pu consacrer plus de temps sur la partie mobile, en cause, les retards et les soucis rencontrés sur la partie PC qui m'ont contraints à travailler en particulier sur cette partie là. Au niveau du groupe, nous avions une équipe très intéressante, composée de personnes toutes douées dans un domaine différent. Je suis très content du fort taux de présence lors des séances de projet. Il témoigne de l'intérêt de l'équipe à s'investir dans le projet. ...

Matthieu ANCERET — *chef de projet*.

6.2 Calendrier prévisionnel et calendrier réel

<!– expliquer les différences existantes entre prévisionnel et réel éventuellement, produire un diagramme avec les tâches critiques, les tâches en retard... –>

6.3 Erreurs commises et problèmes rencontrés

+ parler des solutions mises en place Cette partie est en relation avec la partie calendrier. En effet, les retards du calendrier réel sont dus aux erreurs qui seront reportés ici.

N°	Nom de la tâche	Durée	Début	Fin
0	**Calendrier**	**90 jours**	**Lun 06/02/12**	**Ven 08/06/12**
1	1 Découverte du projet	4 jours	Lun 06/02/12	Jeu 09/02/12
2	**2 Phase d'analyse**	**18 jours**	**Jeu 09/02/12**	**Lun 05/03/12**
3	2.1 Réalisation des prototypes des IHM	7 jours	Dim 19/02/12	Dim 26/02/12
4	2.2 Réalisation du fichier de données XML	7 jours	Dim 19/02/12	Dim 26/02/12
5	2.3 Création de la DTD associée au fichier XML	7 jours	Dim 19/02/12	Dim 26/02/12
6	2.4 Diagramme de cas d'utilisation	6 jours	Lun 27/02/12	Lun 05/03/12
7	2.5 Diagramme de composants	5 jours	Lun 27/02/12	Ven 02/03/12
8	2.6 Rédaction du cahier de spécification	6 jours	Lun 27/02/12	Dim 04/03/12
9	2.7 Remise du cahier de spécification	1 jour	Lun 05/03/12	Lun 05/03/12
10	**3 Modélisation UML**	**11 jours**	**Lun 05/03/12**	**Lun 19/03/12**
11	3.1 Diagramme de classes - application mobile	6 jours	Lun 05/03/12	Lun 12/03/12
12	3.2 Diagramme de classes - application PC	6 jours	Lun 05/03/12	Lun 12/03/12
13	3.3 Modélisation structure application PC	6 jours	Lun 12/03/12	Lun 19/03/12
14	**4 Phase de développement**	**39 jours**	**Lun 12/03/12**	**Jeu 03/05/12**
15	4.1 Installation et configuration des environnements de développements	6 jours	Lun 12/03/12	Lun 19/03/12
16	**4.2 Application PC**	**39 jours**	**Lun 12/03/12**	**Jeu 03/05/12**
17	4.2.1 Implémentation des IHM	6 jours	Lun 12/03/12	Lun 19/03/12
18	4.2.2 Module de lecture fichier XML	6 jours	Mer 14/03/12	Mer 21/03/12
19	4.2.3 Module d'écriture fichier XML	6 jours	Jeu 22/03/12	Jeu 29/03/12
20	4.2.4 Visualisation de l'arbre	6 jours	Jeu 22/03/12	Jeu 29/03/12
21	4.2.5 Ajouter/modifier/supprimer des éléments via l'interface	21 jours	Ven 30/03/12	Ven 27/04/12
22	**4.2.6 Module de visualisation des médias**	**12 jours**	**Lun 12/03/12**	**Mar 27/03/12**
23	4.2.6.1 Module photo	3 jours	Lun 12/03/12	Mer 14/03/12
24	4.2.6.2 Module audio	4 jours	Jeu 15/03/12	Mar 20/03/12
25	4.2.6.3 Module vidéo	5 jours	Mer 21/03/12	Mar 27/03/12
26	4.2.7 Module de recherche des informations manquantes	5 jours	Ven 27/04/12	Jeu 03/05/12
27	**4.3 Application mobile**	**31 jours**	**Lun 12/03/12**	**Lun 23/04/12**
28	4.3.1 Prise en main de l'environnement de développement Android	6 jours	Lun 12/03/12	Lun 19/03/12
29	4.3.2 Implémentation des IHM	11 jours	Lun 19/03/12	Lun 02/04/12
30	4.3.3 Module de recherche rapide	6 jours	Lun 19/03/12	Lun 26/03/12
31	4.3.4 Gestion du GPS	3 jours	Lun 19/03/12	Mer 21/03/12
32	4.3.5 Gestion du DatePicker	3 jours	Mer 21/03/12	Ven 23/03/12
33	4.3.6 Gestion des campagnes/sessions/pièges	11 jours	Lun 19/03/12	Lun 02/04/12
34	4.3.7 Navigation entre questions et réponses	11 jours	Lun 02/04/12	Lun 16/04/12
35	4.3.8 Développement de la base de données	11 jours	Lun 02/04/12	Lun 16/04/12
36	4.3.9 Module de log et d'écriture des résultats	6 jours	Lun 16/04/12	Lun 23/04/12
37	4.3.10 Album et identification rapide	11 jours	Lun 26/03/12	Lun 09/04/12
38	5 Phase de test	6 jours	Ven 04/05/12	Ven 11/05/12
39	6 Rédaction du compte-rendu	14 jours	Lun 14/05/12	Jeu 31/05/12
40	7 Remise du compte rendu	1 jour	Ven 01/06/12	Ven 01/06/12
41	8 Soutenance	5 jours	Lun 04/06/12	Ven 08/06/12

Figure 6.1: Calendrier prévisionnel par tâches

N°	Nom de la tâche	Durée	Début	Fin
0	**Calendrier**	**90 jours**	**Lun 06/02/12**	**Ven 08/06/12**
1	1 Découverte du projet	6 jours	Lun 06/02/12	Lun 13/02/12
2	**2 Phase d'analyse**	**16 jours**	**Lun 13/02/12**	**Lun 05/03/12**
3	2.1 Réalisation des prototypes des IHM	7 jours	Dim 19/02/12	Dim 26/02/12
4	2.2 Réalisation du fichier de données XML	6 jours	Jeu 16/02/12	Jeu 23/02/12
5	2.3 Création de la DTD associée au fichier XML	6 jours	Jeu 16/02/12	Jeu 23/02/12
6	2.4 Diagramme de cas d'utilisation	7 jours	Dim 19/02/12	Dim 26/02/12
7	2.5 Diagramme de composants	7 jours	Dim 19/02/12	Dim 26/02/12
8	2.6 Rédaction du cahier de spécification	6 jours	Lun 27/02/12	Dim 04/03/12
9	2.7 Remise du cahier de spécification	1 jour	Lun 05/03/12	Lun 05/03/12
10	**3 Modélisation UML**	**23 jours**	**Lun 05/03/12**	**Mer 04/04/12**
11	3.1 Diagramme de classes - application mobile	6 jours	Lun 05/03/12	Lun 12/03/12
12	3.2 Diagramme de classe - application PC	6 jours	Mar 13/03/12	Mar 20/03/12
13	3.3 Modélisation structure de l'application PC	11 jours	Mer 21/03/12	Mer 04/04/12
14	4 Correction du fichier XML et de sa DTD	6 jours	Mer 28/03/12	Mer 04/04/12
15	**5 Phase de développement**	**56 jours**	**Lun 19/03/12**	**Dim 03/06/12**
16	5.1 Installation et configuration des environnements de développement	5 jours	Lun 19/03/12	Ven 23/03/12
17	**5.2 Application PC**	**53 jours**	**Lun 19/03/12**	**Mer 30/05/12**
18	5.2.1 Implémentation des IHM	11 jours	Lun 19/03/12	Lun 02/04/12
19	5.2.2 Module de lecture fichier XML	26 jours	Lun 02/04/12	Lun 07/05/12
20	5.2.3 Module d'écriture fichier XML	11 jours	Lun 07/05/12	Lun 21/05/12
21	5.2.4 Visualisation de l'arbre	6 jours	Lun 07/05/12	Lun 14/05/12
22	5.2.5 Ajouter/modifier/supprimer des éléments via l'interface	11 jours	Lun 14/05/12	Lun 28/05/12
23	**5.2.6 Module de visualisation des médias**	**50 jours**	**Jeu 08/03/12**	**Mer 16/05/12**
24	5.2.6.1 Module photo	6 jours	Jeu 08/03/12	Jeu 15/03/12
25	5.2.6.2 Module audio	16 jours	Jeu 15/03/12	Jeu 05/04/12
26	5.2.6.3 Module vidéo	30 jours	Jeu 05/04/12	Mer 16/05/12
27	5.2.7 Module de recherche des informations manquantes	6 jours	Mer 16/05/12	Mer 23/05/12
28	**5.3 Application mobile**	**58 jours**	**Lun 12/03/12**	**Mer 30/05/12**
29	5.3.1 Prise en main de l'environnement de développement Android	6 jours	Lun 12/03/12	Lun 19/03/12
30	5.3.2 Implémentation IHM	17 jours	Lun 26/03/12	Mar 17/04/12
31	5.3.3 Module de recherche rapide	31 jours	Jeu 08/03/12	Jeu 19/04/12
32	5.3.4 Gestion du GPS	2 jours	Mer 11/04/12	Jeu 12/04/12
33	5.3.5 Gestion du DatePicker	3 jours	Ven 13/04/12	Mar 17/04/12
34	5.3.6 Gestion des campagnes/sessions/pièges	11 jours	Lun 19/03/12	Lun 02/04/12
35	5.3.7 Navigation entre questions et réponses	11 jours	Lun 02/04/12	Lun 16/04/12
36	5.3.8 Développement de la base de données	16 jours	Mar 08/05/12	Mar 29/05/12
37	5.3.9 Module de log et d'écriture des résultats	3 jours	Lun 21/05/12	Mer 23/05/12
38	5.3.10 Album et interface d'identification	13 jours	Lun 14/05/12	Mer 30/05/12
39	6 Rédaction du compte-rendu	11 jours	Lun 21/05/12	Dim 03/06/12
40	7 Remise du compte rendu	1 jour	Dim 03/06/12	Dim 03/06/12
41	8 Soutenance	5 jours	Lun 04/06/12	Ven 08/06/12

Figure 6.2: Calendrier réel par tâches

Futur du projet

Globalement, le projet est utilisable en l'état actuel. Il reste malgré tout plusieurs éléments à terminer ou à compléter, que ce soit par manque de temps ou parce que ces fonctionnalités ne faisaient pas partis du périmètre du projet. En voici une liste détaillée :

- Avec le recul, nous avons remarqué quelques points perfectibles dans l'interface de l'application PC. Ces points nuisent à l'ergonomie de l'application. Notamment, comment symboliser le lien entre une réponse et la question suivante. Nous avons choisi de séparer l'affichage des questions de celui des réponses. Nous avons donc symbolisé la question suivante en affichant son nom lors du clic sur une réponse. Ce nom permettant de se repérer dans l'arborescence des questions. Représenter clairement et lisiblement une arborescence XML complexe dans une interface est assez compliqué. Nous avons choisi une méthode, peut-être pas la meilleure, car il en existe d'autre.

- Actuellement, nous stockons tous les fichiers des médias dans un seul et unique dossier. Il pourrait être intéressant ce trier ces médias dans une arborescence de dossiers (par exemple, reprenant l'arborescence de l'arbre XML ; ou encore par type de médias : un dossier images, un dossier son, un dossier vidéo...). Ceci pourrait apporter plus de souplesse dans le traitement et l'analyse de ces médias.

- Au niveau de l'application mobile, il serait intéressant de pouvoir visualiser les résultats et les statistiques directement sur la tablette, via une interface pratique et adaptée. Actuellement, la seule façon de consulter les résultats est de produire un fichier tableur à partir de tablette, de récupérer ce fichier et de l'ouvrir sur un PC avec un logiciel adapté (Excel, OpenOffice...). Pouvoir les consulter directement sur la tablette permettrait d'éviter cette étape d'exportation et de pouvoir gagner du temps dans le cas où l'utilisateur veut simplement vérifier ou consulter certains résultats intermédiaires en cours d'une campagne d'acquisition.

- Toujours au niveau de l'application mobile, il n'est actuellement pas possible de changer le fichier de données lorsque l'application fonctionne. Il faut arrêter l'application, modifier le fichier XML et son dossier de médias associé et redémarrer l'application. Il serait intéresser de pouvoir changer la base de données (fichier XML + médias) "à la volée", à partir d'un menu de paramètre par exemple. Cela permettrait d'avoir, directement embarquée sur la tablette, plusieurs bases de données (insectes, oiseaux, plantes...) pour étendre les possibilités de l'application.

- Pour rendre l'application tablette presque entièrement indépendante, il faudrait mettre en place un système de mise à jour automatique des bases de données embarquées. Actuellement, une fois la base modifiée via l'application PC, il faut copier manuellement les fichiers de cette nouvelle base sur la tablette. Une idée pourrait être d'installer un serveur sur lequel serait poussée les mises à jour réalisées par l'application PC et où l'application mobile irait régulièrement chercher les nouvelles informations. Ce mode de fonctionnement permettrait aussi à plusieurs tablettes de récupérer rapidement la dernière version de la base. La seule contrainte étant d'avoir un accès à internet à partir de la tablette (de plus en plus courant). Ce système permettrait aussi d'envoyer sur le serveur (et donc de centraliser) les résultats et les statistiques de chaque tablette.

Conclusion

Faire un bilan global sur le projet et une petite ouverture sur son avenir...

Documentation Doxygen

Un lien vers la doc Doxygen ou directement la doc complète au format PDF ?

Documentation utilisateur (manuel d'utilisation)

Cette partie sera surement intégré tout au long du rapport en lui-même

Quelques éléments techniques bien précis

Je ne pense pas que cette partie soit utile, on en parle déjà au fur et mesure dans le rapport

Application d'aide à l'identification d'insectes nuisibles

Département Informatique
4ieme année
2011 - 2012

Rapport projet collectif

Résumé : Blabla de présentation du sujet (cf. intro + présentation). Ce rapport est la synthèse de tout le projet, des phases de modélisation aux phases de tests en passant par les phases de développement et les parties gestion de projet. Ce rapport est un guide technique, qui pourra servir à un développeur désireux de reprendre et de comprendre les applications existantes, mais aussi un guide utilisateur, qui permettra à l'usager de trouver les réponses à ses questions à propos du fonctionnement des logiciels.

Mots clefs : Mots clés français

Abstract: Description en anglais

Keywords: Mots clés en anglais

Encadrants
Gilles VENTURINI
gilles.venturini@univ-tours.fr

Université François-Rabelais, Tours
Client
Ingrid ARNAULT
ingrid.arnault@univ-tours.fr
Damien MUNIER
munier.damien@aliceadsl.fr
Alexandre DEPOILLY
alexandre.depoilly@etu.univ-tours.fr

Équipe CETU INNOPHYT

Étudiants
Matthieu ANCERET
matthieu.anceret@etu.univ-tours.fr
Jérome HEISSLER
jerome.heissler@etu.univ-tours.fr
Julien TERUEL
julien.teruel@etu.univ-tours.fr
Martin DEMEULEMEESTER
martin.demeulemeester@etu.univ-tours.fr
Mickael PURET
mickael.puret@etu.univ-tours.fr
Simon FAUSSIER
simon.faussier@etu.univ-tours.fr
Zheng ZHANG
zheng.zhang@etu.univ-tours.fr
Zhengyi LIU
zhengyi.liu@etu.univ-tours.fr

DI4 2011 - 2012