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 May 30, 2012

Contents

List of Figures

List of Tables

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.

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 associées à l'application mobile, c'est-à-dire les questions, les réponses et les médias.

Analyse complémentaires

Nous allons présenter ici les modélisations et diagrammes UML réalisés à posteriori du cahier de spécifications.

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.

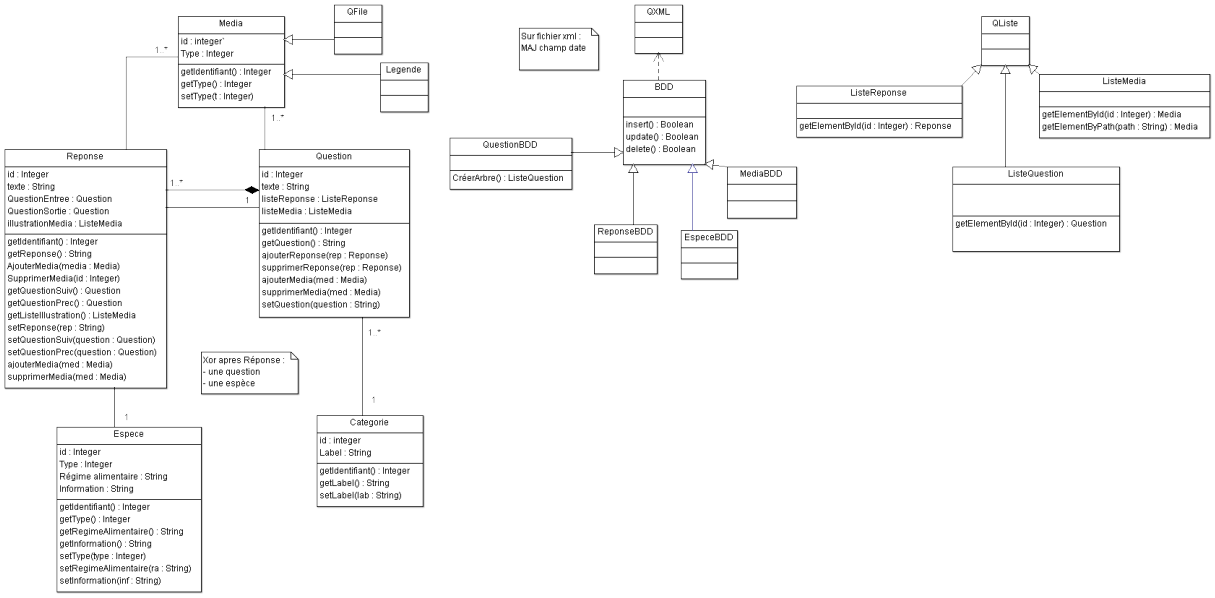

Figure 3.1: Diagramme de classes de l'application PC

3.2 Fichier XML final + 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.

Nous allons donc exposer la version finale du fichier 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="Première 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="Réponse 1">
<media>
<img id="img1" src="images/insecte2.jpg" />
<img id="img1" src="images/insecte3.jpg" />
<legende>Ceci est le texte de la réponse 1</legende>
</media>
<branche id="b2">
<question id="q3" texte="Autre question" visible="false">
...
</question>
</branche>
</reponse>
<reponse id="r2" texte="Réponse 2">
<media>
<img id="img1" src="images/insecte2.jpg" />
<legende>Ceci est le texte de la réponse 2</legende>
</media>
<branche id="b3" type="Arachnide">
<question id="q2" texte="Deuxième question ?" visible="true">
<reponse id="r3" texte="Réponse 3">
<resultat id="res1">
<nom>Insecte 1</nom>
<type>MEL1</type>
<regimeAlimentaire>Prédateur</regimeAlimentaire>
<informations>Informations insecte 1</informations>
<media>
<img id="img2" src="images/insecte.jpg" />
</media>
</resultat>
</reponse>
<reponse id="r4" texte="Réponse 4">
<branche id="b2">
<question id="q6" texte="Encore une question ?" visible="both">
...
</question>
</branche>
```

```
</reponse>
</question>
</branche>
</reponse>
</question>
</branche>
</arbre>
```

Nous avons bien sur corrigé la DTD associée à ce fichier XML. Cette DTD est le modèle du fichier XML et permet donc de le valider.

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

<!ELEMENT arbre (branche)>
<!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)>
```

Nous avons aussi, suite à plusieurs discussions avec le client, modifié le format de sortie des résultats.

Application PC

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.

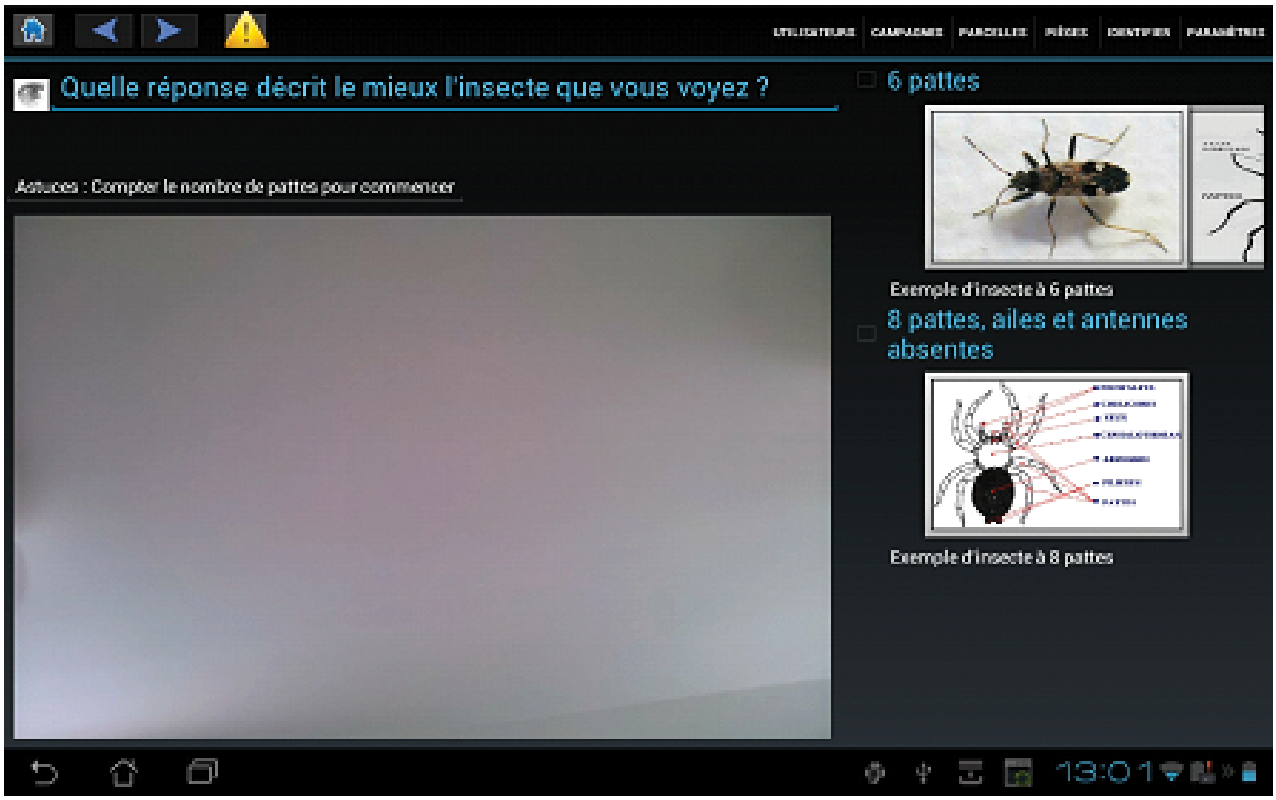

Figure 5.1: Interface d'identification d'un insecte - application mobile

5.1 Fonctionalitées

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

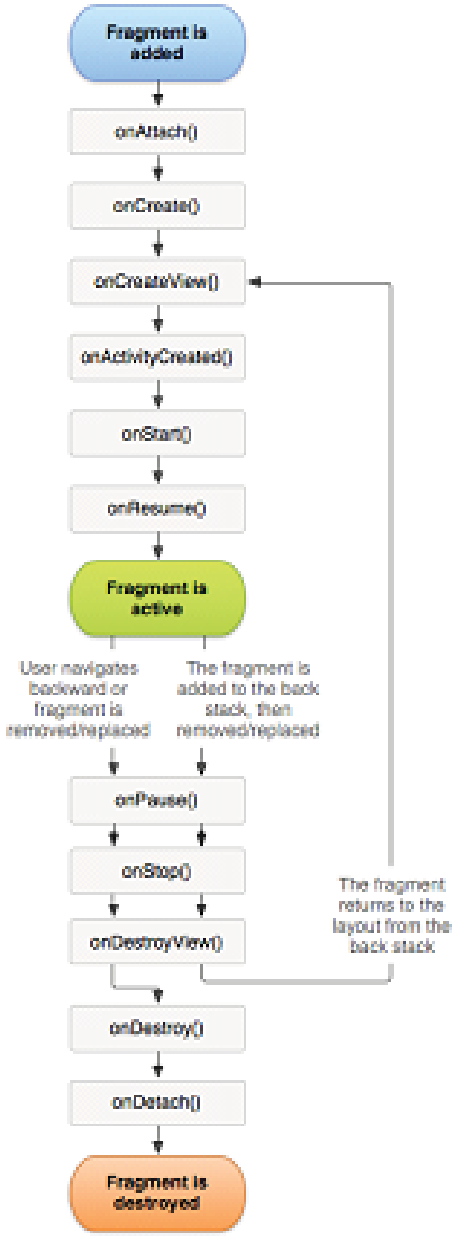

Figure 5.2: Schéma explicatif du fonctionnement des fragments

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.

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 Compétences acquises

Parler des compétences acquises grâce à ce projet.

6.2 Calendrier prévisionnel et calendrier réel

6.3 Erreurs commises et problèmes rencontrés

+ parler des solutions mises en place

Futur du projet

Ce qu'il reste à faire
- Les priorités
- ...

Conclusion

Documentation Doxygen
- Documentation utilisateur (manuel d'utilisation)
- Quelques éléments techniques bien précis

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

Département Informatique
4ieme année
2011 - 2012

Rapport projet collectif

Résumé : Description en français

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