



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

Cahier de spécification système & plan de développement

Projet :	Application mobile d'aide à l'identification d'insectes nuisibles		
Emetteur :	Matthieu ANCERET	Coordonnées : EPU-DI	
Date d'émission :	10 mars 2012		
Validation			
Nom	Date	Valide (O/N)	Commentaires

Historique des modifications

Version	Date	Description de la modification
---------	------	--------------------------------

00 : 13/02/2012 ; Version initiale : synthèse de la 1ère séance de travail

Table des matières

Cahier de spécification système	5
1.1 Introduction	5
1.2 Contexte de la réalisation	5
1.2.1 Contexte	5
1.2.2 Objectifs	5
1.2.3 Hypothèses	5
1.2.4 Bases méthodologiques	6
1.3 Description générale	6
1.3.1 Environnement du projet	6
1.3.2 Caractéristiques des utilisateurs	6
1.3.3 Fonctionnalités et structure générale du système	7
1.3.4 Contraintes de développement, d'exploitation et de maintenance	8
1.4 Description des interfaces externes du logiciel	8
1.4.1 Interfaces matériel/logiciel	8
1.4.2 Interfaces homme/machine	9
1.4.3 Interfaces logiciel/logiciel	15
1.5 Architecture générale du système	17
1.6 Description des fonctionnalités	17
1.6.1 Application PC	17
1.6.2 Application mobile	19
1.7 Conditions de fonctionnement	20
1.7.1 Capacités	20
1.7.2 Contrôlabilité	20
1.7.3 Sécurité	20
 Plan de développement	 22
2.1 Découpage du projet en tâches	22
2.1.1 Rédaction de la modélisation UML	22
2.1.2 Création du fichier XML et de sa DTD	22
2.1.3 Application PC	22
2.1.4 Application mobile	23
2.2 Planning	23

Cahier de spécification système

1.1 Introduction

Ce document est un cahier de spécification fonctionnelle. Il a pour but de définir clairement toutes les demandes du client. Il permettra, lors de la validation finale, de vérifier que le livrable est conforme à ce que le client avait demandé.

Notre projet est un projet collectif à 8 personnes dans le cadre de notre cursus à l'école 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. Notre encadrant de projet est Gilles Venturini. Étant donné qu'il connaît bien le projet, il est aussi un interlocuteur privilégié pour les aspects techniques et opérationnels.

1.2 Contexte de la réalisation

1.2.1 Contexte

L'enjeu du projet est de permettre à l'équipe INNOPHYT d'obtenir un outil performant de reconnaissance d'espèce, et plus précisément, de savoir si une espèce est nuisible ou non. C'est donc un outil d'aide à la décision. Actuellement, l'équipe utilise un fichier Excel qui répertorie les questions à se poser pour arriver à un résultat (sous une forme arborescente). Ce système est quelque peu archaïque et demande un effort important de la part de l'utilisateur. De plus, l'évolution d'un tel système (ajout, modification, suppression d'élément) est compliqué.

Les futurs utilisateurs de l'application peuvent être aussi bien des ingénieurs ou des spécialistes du domaine (biologiste de terrain, membre de l'équipe INNOPHYT...) que des utilisateurs lambda n'ayant pas de connaissances particulières (agriculteur, passionné des insectes...).

1.2.2 Objectifs

Le projet est décomposé en 2 grandes parties :

- Une application mobile qui va permettre à l'utilisateur, directement sur le terrain, de déterminer le type de l'insecte et de décider s'il est nuisible ou non. L'application va poser un certain nombre de questions et déduire des choses à partir des réponses obtenues. Celle-ci proposera à l'utilisateur différents médias (photos, vidéos, animations...) pour l'aider à prendre une décision.
- Une application sur PC qui va permettre de consulter la base de données (l'arbre de décision) et de la faire évoluer (ajout/modification/suppression d'espèces, de questions, de réponses, de médias...). Cette application sera principalement utilisée par l'équipe INNOPHYT ou éventuellement par des experts du domaine. Elle doit permettre de visualiser rapidement les éléments sur lesquels il manque des informations.

L'application PC devra permettre d'exporter la base de données pour la mettre à disposition de l'application mobile. L'application mobile devra fournir un rapport des actions effectuées sur celle-ci (espèces inconnues, questions/réponses floues, résultat...) pour permettre à l'équipe INNOPHYT de corriger ou de faire évoluer la base.

1.2.3 Hypothèses

Nous allons poser un certain nombre d'hypothèses qui permettront de définir le périmètre du projet :

- On considère que la reconnaissance d'image ne fait pas partie du projet. L'utilisateur se base sur une photo de l'insecte ou un "vrai" insecte pour répondre aux questions.
- La plateforme pour l'application mobile est fixée : Android. On considère que l'on ne change pas de plateforme en cours de projet.

- On considère que la tablette ne sera pas forcément connectée à internet, donc pas de synchronisation automatique entre la tablette et l'application PC.
- On considère que l'application PC sera utilisée sur un seul ordinateur simultanément (pas de gestion de plusieurs bases de données et de la cohérence de la base).
- On considère que la base de données sera mise à jour manuellement par l'utilisateur, par simple copie standard par exemple (via support USB...).

1.2.4 Bases méthodologiques

Conformément aux "règles" du projet collectif, nous allons utiliser la méthodologie Agile "Scrum". Le principe de cette méthode est de se focaliser sur une partie limitée et maîtrisable du projet (un "sprint"). A chaque fin de sprint, l'équipe doit être capable de fournir au client un livrable partiel mais fonctionnel. Pour l'application PC, nous allons utiliser le langage C++ avec le framework QT. Nous avons choisi ce langage car nous l'avons déjà utilisé lors de TP-projet à Polytech'Tours. De plus, l'utilisation d'un langage comme le C++ (langage compilé) nous garantis des performances intéressantes pour l'application et une compatibilité avec les principaux systèmes d'exploitation (Windows, Linux et Mac OS).

Pour l'application mobile, sachant que nous avons une plateforme Android, nous allons utiliser le langage Java (c'est le langage officiel pour le développement sous Android).

Étant donné que c'est un travail de groupe, nous allons devoir utiliser des outils de travail collaboratif :

- Pour gérer les tâches entre les différentes personnes, nous allons utiliser le site "Wunderkit". Il permet de créer des listes de tâche et de les affecter à une ou plusieurs personnes. Il est possible de classifier ces tâches selon leur avancement ("à faire", "en cours", "à valider", "terminée"...).
- Pour la gestion des sources, nous allons utiliser un dépôt GIT sur Github.

1.3 Description générale

1.3.1 Environnement du projet

Pour ce projet, nous n'avons aucun existant, mis à part l'arbre de décision que nous a fourni l'équipe INNOPHYT (au format Excel). Nous avons donc carte blanche pour définir le format de la base de données et l'architecture de nos deux applications.

Il existe deux projets qui gravitent autour du notre. Un projet de 5ème année, qui a pour objectif de réaliser une application similaire à la notre sous forme d'application web. Les étudiants concernés par ce projet auront besoin de la structure de notre base de données (fichier XML). Un autre projet, proposé aux 3ème années, a pour but d'étudier un système de modélisation d'insecte en 3D à partir d'un "véritable" insecte.

1.3.2 Caractéristiques des utilisateurs

Dans cette partie, nous allons lister les différents types d'utilisateur de nos deux applications ainsi que leurs caractéristiques :

Utilisateur de l'application PC

Le biologiste (l'équipe INNOPHYT)

- connaissance ou non de l'informatique : niveau bureautique avancé
- expérience de l'application : aucune
- utilisateurs réguliers et/ou occasionnels : régulier
- droits d'accès utilisateurs : complet

Utilisateur de l'application mobile

Le biologiste

- connaissance ou non de l'informatique : niveau bureautique avancé
- expérience de l'application : aucune
- utilisateurs réguliers et/ou occasionnels : occasionnel
- droits d'accès utilisateurs : complet

L'agriculteur

- connaissance ou non de l'informatique : faible
- expérience de l'application : aucune
- utilisateurs réguliers et/ou occasionnels : utilisation occasionnelle voir nulle
- droits d'accès utilisateurs : complet

Le chercheur de terrain (biologiste de terrain)

- connaissance ou non de l'informatique : niveau bureautique avancé
- expérience de l'application : aucune
- utilisateurs réguliers et/ou occasionnels : régulier
- droits d'accès utilisateurs : complet

1.3.3 Fonctionnalités et structure générale du système

Pour exprimer les fonctionnalités de notre système, nous allons utiliser la modélisation UML et plus particulièrement les diagrammes de cas d'utilisation.

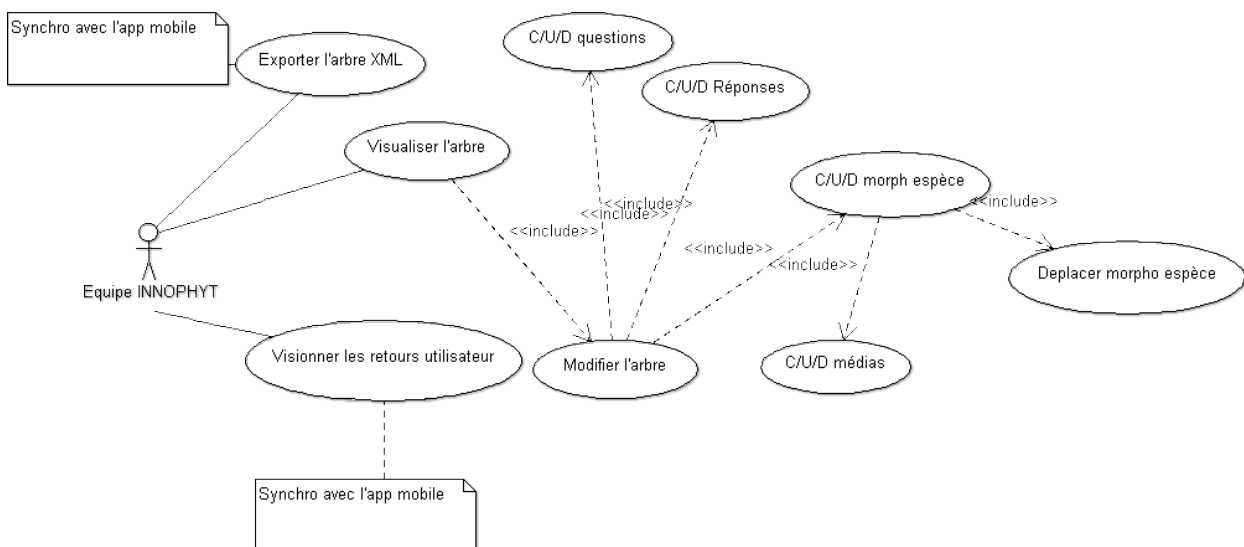


FIGURE 1.1 – Diagramme de cas d'utilisation de l'application PC

Sur le diagramme de cas d'utilisation de l'application PC (1.1), la notation "C/U/D" signifie "Create/Update/Delete" (soit Créer/Modifier/Supprimer). La partie "synchro avec l'application mobile" symbolise un module d'exportation de la base de données dans un fichier (écriture d'un fichier XML). La partie "visionner les retours utilisateurs" ne sera pas forcément traitée par l'application. Elle est malgré tout représentée pour montrer le fait qu'il y a une remontée d'informations de la part de l'application mobile. Sur le diagramme de cas d'utilisation de l'application mobile (1.2), on remarque que l'on peut créer des campagnes, des sessions et des pièges. Une campagne est composée de une ou plusieurs sessions et une session

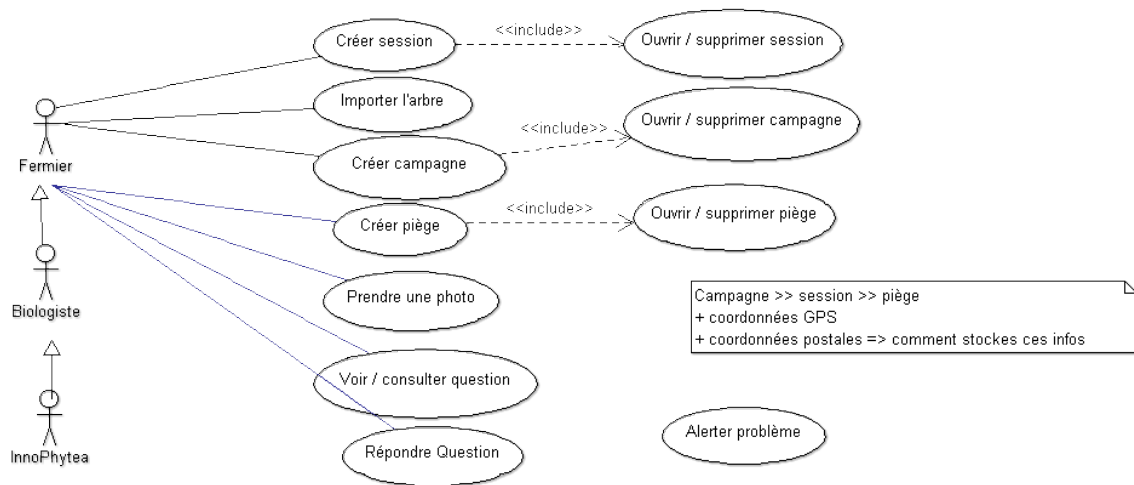


FIGURE 1.2 – Diagramme de cas d'utilisation de l'application mobile

de un ou plusieurs pièges. Une campagne est représentée par un nom, une date de début et éventuellement une date de fin. Un piège est représentée par un nom, une date et des coordonnées GPS.

1.3.4 Contraintes de développement, d'exploitation et de maintenance

Contraintes de développement

- **contraintes matérielles** : étant donné que nous allons travailler avec une tablette (Android), nous allons devoir prendre en compte les contraintes d'autonomie et de performance liées à cet type de périphérique.
- **langages de programmation** : le développement sous Android nous impose le langage Java. En effet, c'est le langage de référence pour cette plateforme. Pour la partie application PC, nous avons décidé de choisir le langage C++ avec le framework QT.
- **logiciels de développement** : Eclipse, qui est l'IDE de référence (et officiel !) pour le développement pour Android. Le temps d'obtenir une véritable tablette, nous allons utiliser les émulateurs fournis avec le SDK Android. Pour la partie PC, nous allons utiliser l'environnement de développement QT Creator.
- **délais de réalisation** : le projet doit être terminé pour fin Mai.

1.4 Description des interfaces externes du logiciel

1.4.1 Interfaces matériel/logiciel

Pour l'application PC, nous développerons une application à destination d'ordinateur sous Windows. Mais notre application sera compatible aussi compatible avec les environnements Linux et Mac OS de part l'utilisation du langage C++ et du framework QT.

Pour l'application mobile, nous allons utiliser une tablette tactile sous Android dont voici les caractéristiques :

- **ASUS Eee Pad Transformer Prime**

- 10,1" (résolution de 1280 x 800)
- Processeur NVidia Tegra 3
- 1 Go de RAM
- 64 Go de mémoire de stockage
- Webcam façade de 1.2Mpx et webcam arrière de 8Mpx (avec flash)
- Wifi + Bluetooth
- GPS
- Android 3.2 Honeycomb (màj vers Android 4.0 Ice Cream Sandwich disponible)
- **Connectique :**
 - 1 micro-HDMI
 - 1 lecteur de carte SD/micro SD
 - 1 entrée microphone
 - 1 sortie audio jack
 - 1 port USB 2.0
- Station d'accueil avec clavier intégré

La liaison entre le PC et la ou les tablette(s) sera, dans un premier temps, fait via un support USB (clé USB, carte SD...). A nous ensuite de réfléchir à une méthode de synchronisation plus souple (Dropbox, serveur...).

1.4.2 Interfaces homme/machine

L'IHM de l'application mobile doit être la plus claire et la plus pratique possible car elle est destinée à des utilisateurs "non-informaticiens".

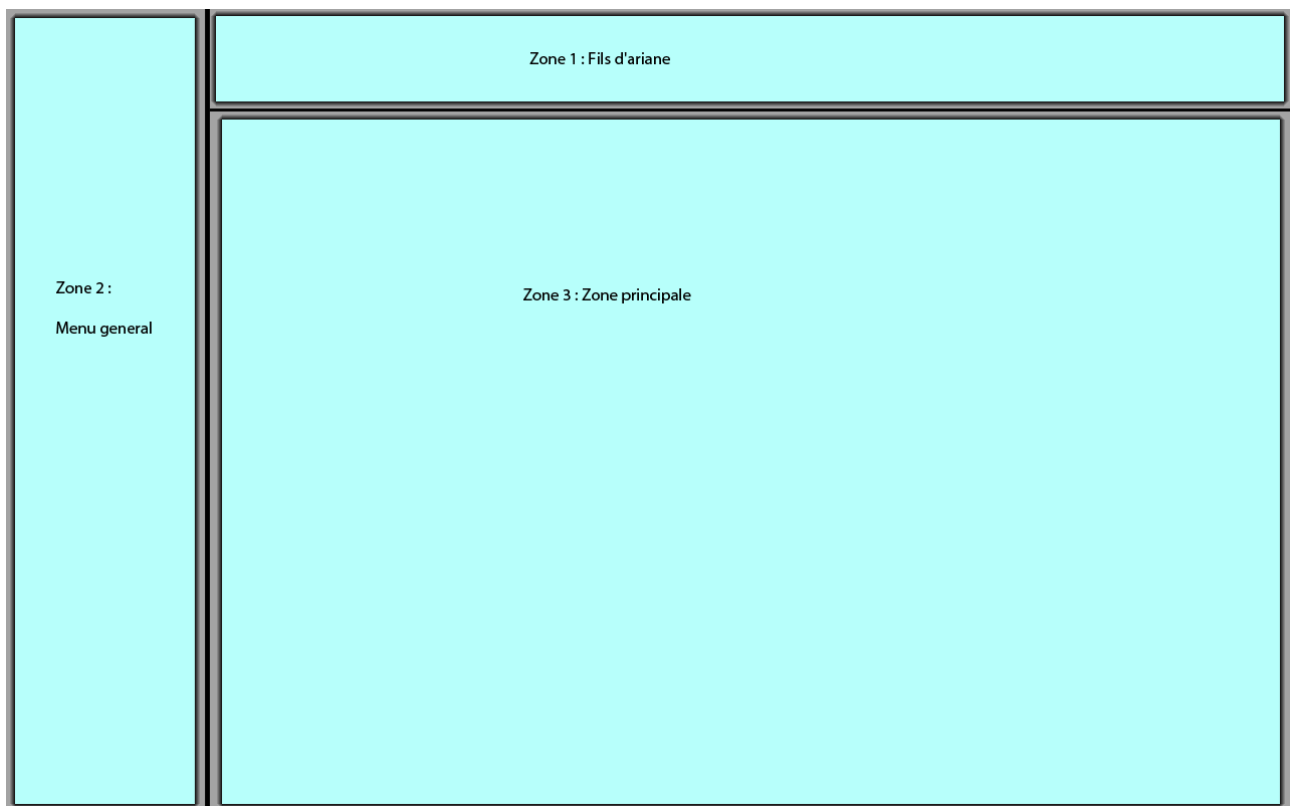


FIGURE 1.3 – Modèle de l'IHM de l'application mobile

Notre interface se décompose en 3 grandes parties :

- En haut, un fil d'ariane, qui permet à l'utilisateur de se repérer lors de son cheminement dans l'application. Cela lui permet à tout moment de savoir où il se situe.
- A gauche, une barre de menu, permettant d'accéder aux principales fonctionnalités de l'application (gestion des campagnes, des pièges, des options...).
- Au centre, la zone principale, permettant d'afficher les informations principales de l'application (les questions, les réponses possibles, l'interface caméra...).

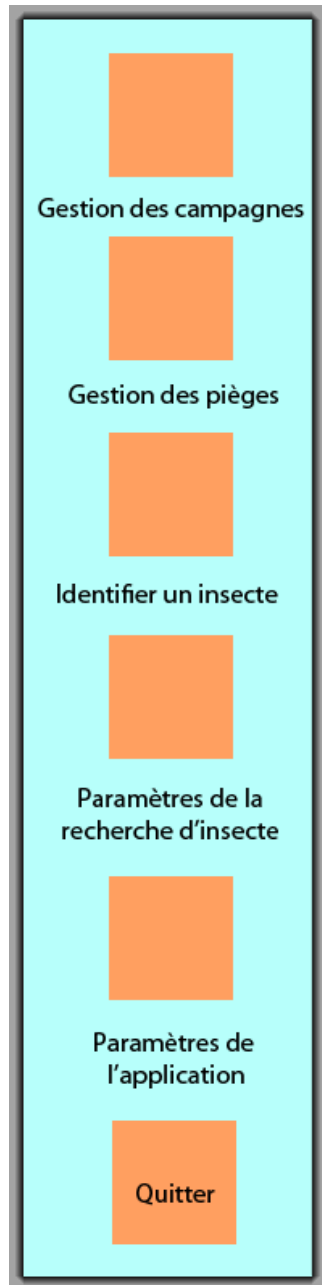


FIGURE 1.4 – Menu latéral de l'application mobile

Campagne :: Zone de pièges
Hiérarchie dans l'arbre de recherche

FIGURE 1.5 – Fil d'ariane de l'application mobile

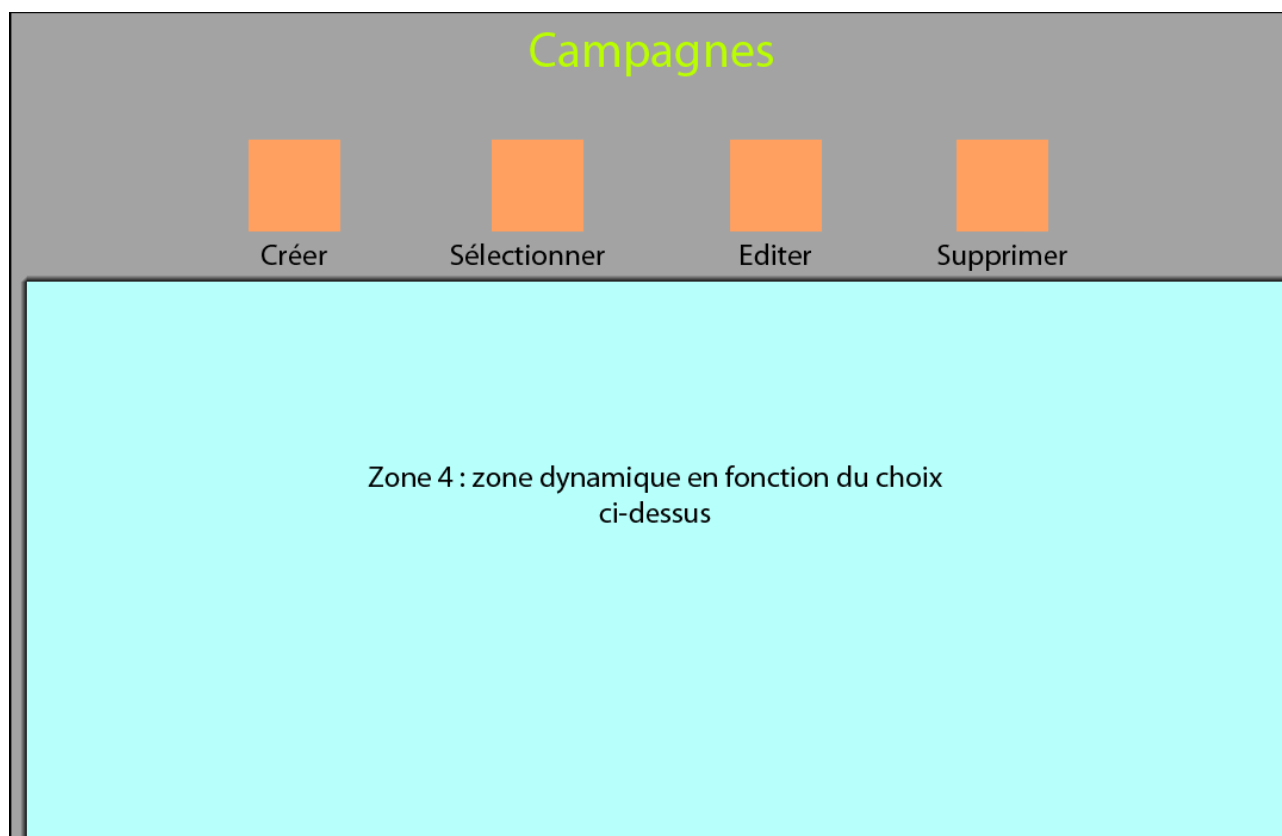
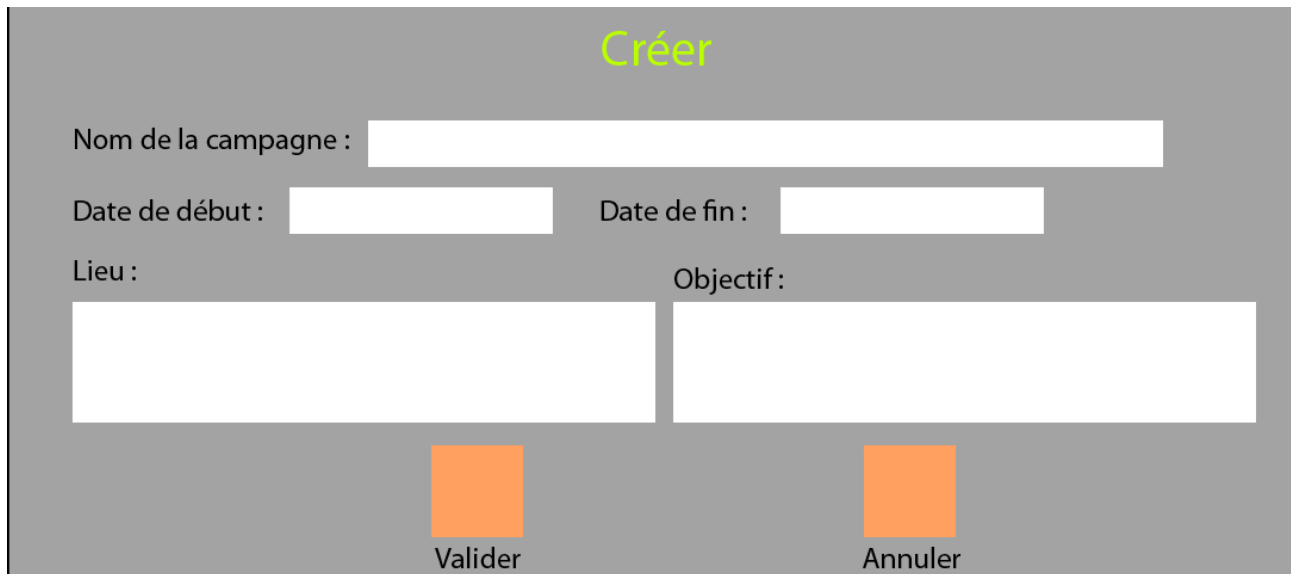


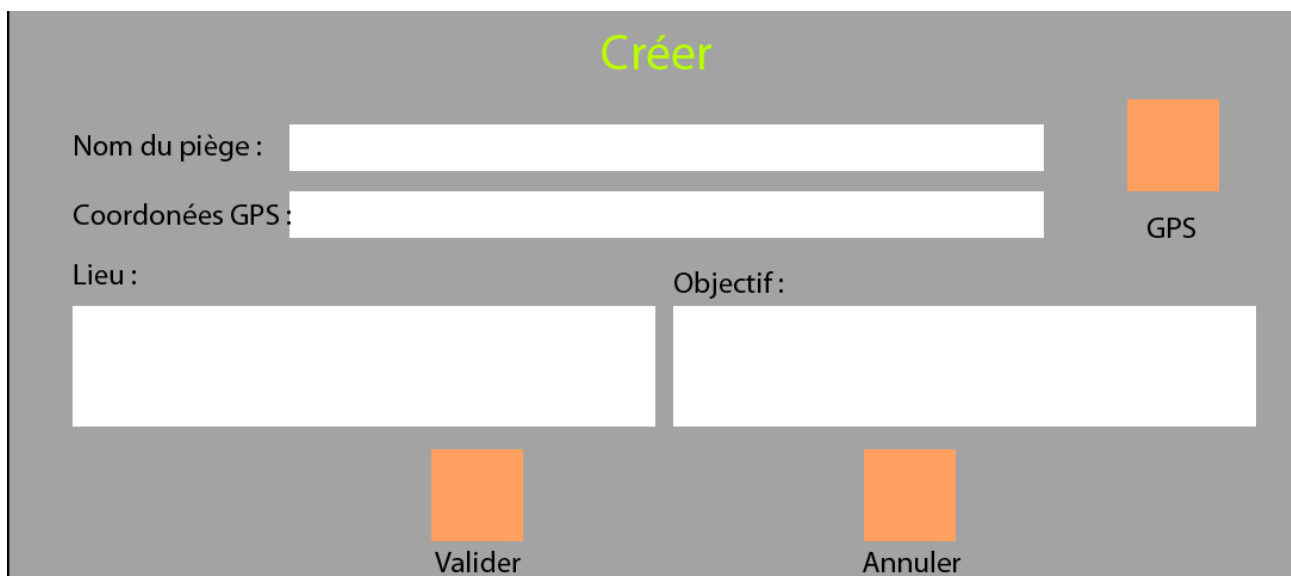
FIGURE 1.6 – Ecran de gestion des campagnes

Cette écran (figure 1.6) permet de créer/continuer/supprimer une campagne. Pour les pièges, l'écran est similaire.



The screenshot shows a grey background with the title 'Créer' in green at the top center. Below the title, there are four input fields: 'Nom de la campagne :', 'Date de début :', 'Date de fin :', and 'Lieu :'. The 'Date de début' and 'Date de fin' fields are grouped together. Below these fields are two orange square buttons labeled 'Valider' and 'Annuler'.

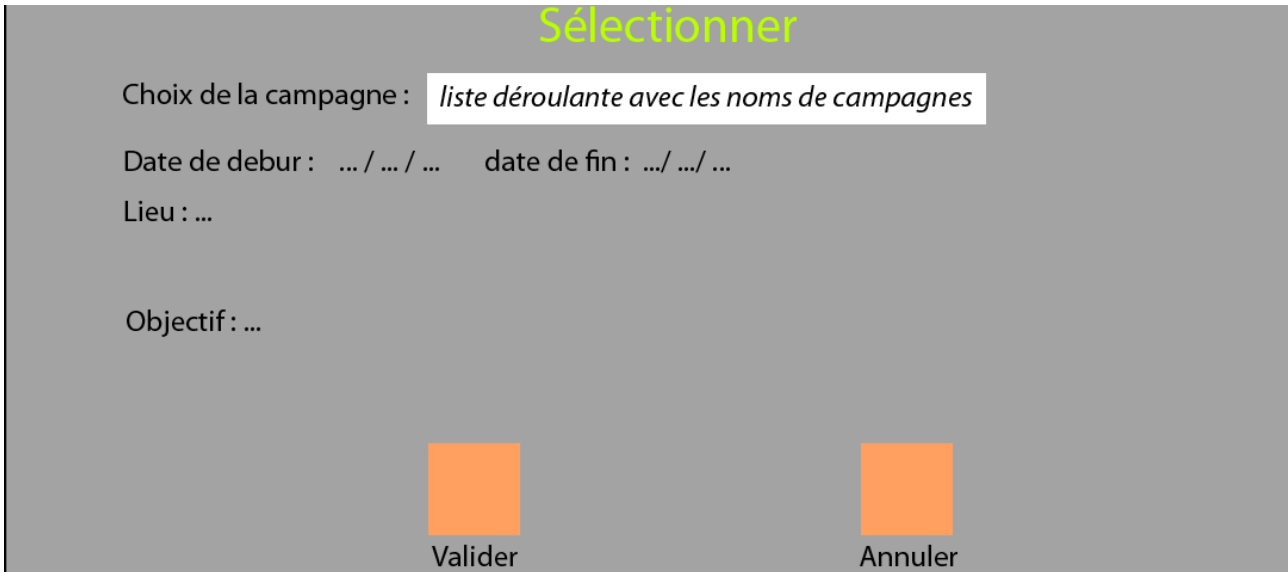
FIGURE 1.7 – Ecran de création d'une nouvelle campagne



The screenshot shows a grey background with the title 'Créer' in green at the top center. Below the title, there are four input fields: 'Nom du piège :', 'Coordonnées GPS :', 'Lieu :', and 'Objectif :'. The 'Coordonnées GPS' field is followed by a small orange square button labeled 'GPS'. Below these fields are two orange square buttons labeled 'Valider' and 'Annuler'.

FIGURE 1.8 – Ecran de création d'un nouveau piège

Cet écran (figure 1.9) est similaire pour l'ouverture d'un piège existant.



Sélectionner

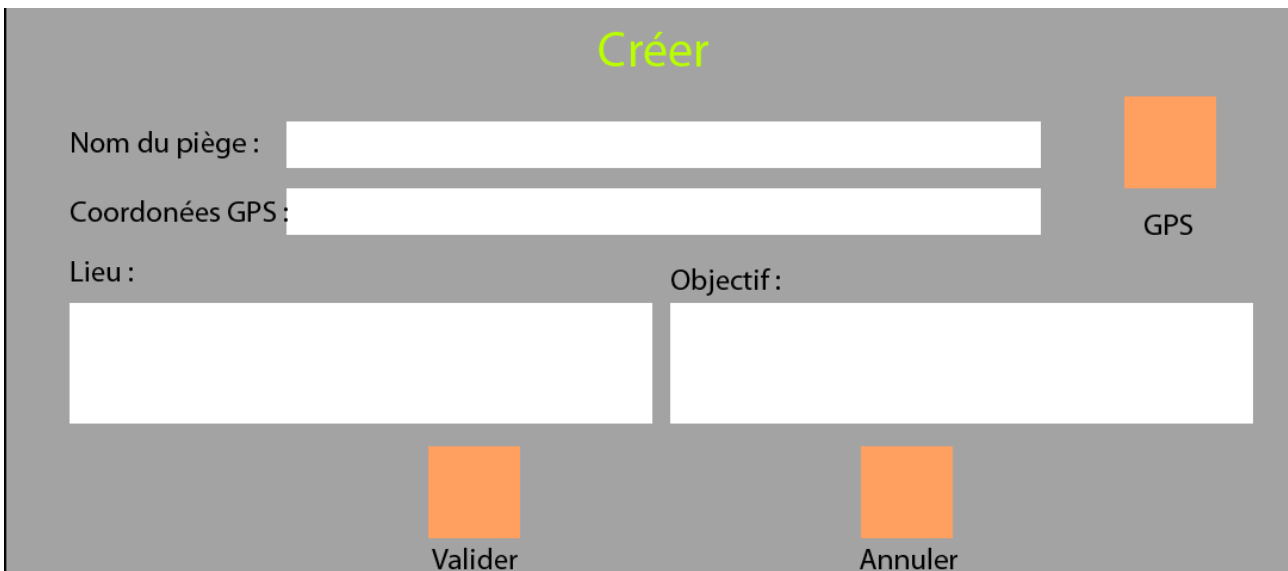
Choix de la campagne :

Date de debut : ... / ... / ... date de fin : ... / ... / ...

Lieu : ...

Objectif : ...

FIGURE 1.9 – Ecran d'ouverture d'une campagne existante



Créer

Nom du piège :

Coordonnées GPS :

Lieu : Objectif :

FIGURE 1.10 – Ecran de création d'un nouveau piège

Cet écran (figure 1.11) propose à l'utilisateur la question à laquelle il doit répondre ainsi que les réponses associées. Si une ressource externe (un média, comme une photo, une vidéo, une animation...) existe pour cette question, elle sera affichée à côté de la question.

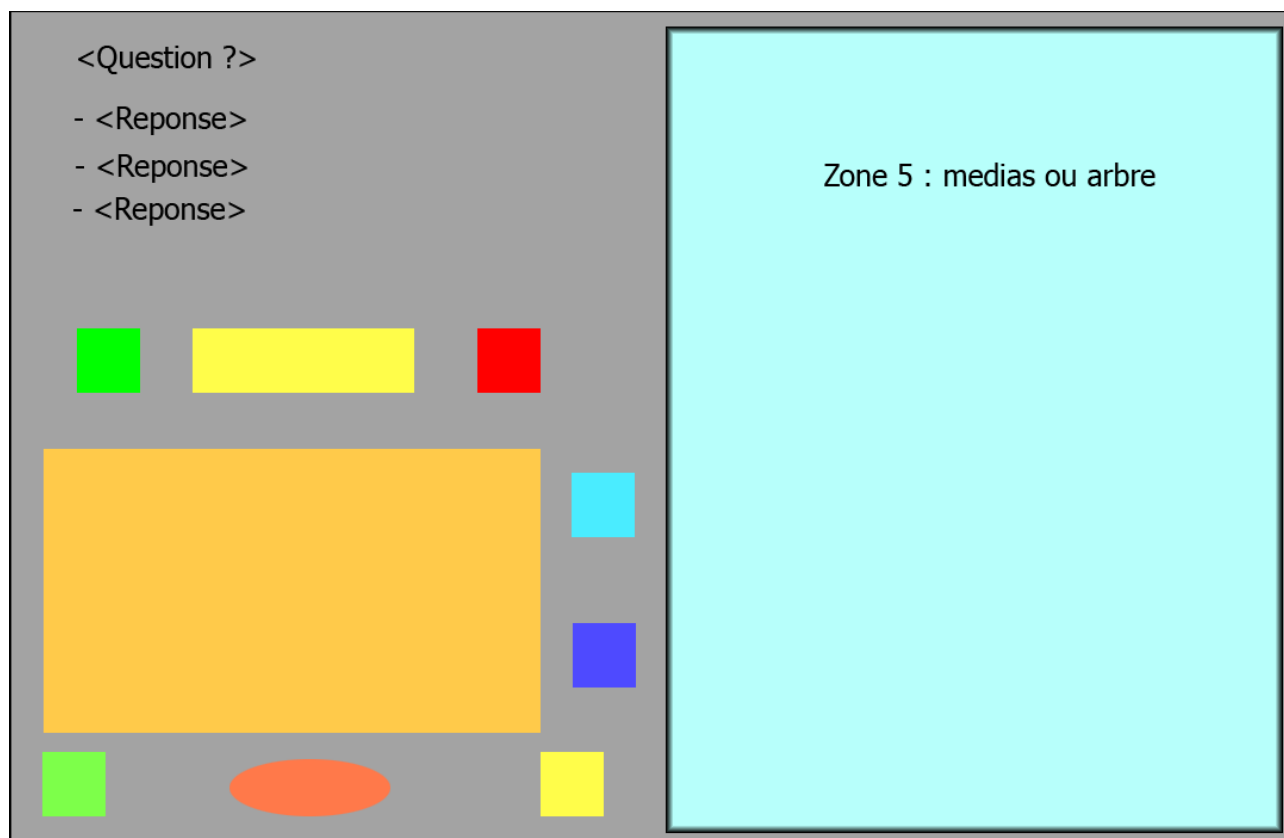


FIGURE 1.11 – Ecran de question/réponse

Cet écran (figure 1.12) permet d'utiliser les fonctionnalités multimédia de la tablette tactile. L'utilisateur peut soit prendre une photo de l'insecte en cours d'étude, soit le filmer et l'afficher en parallèle des questions qui lui sont posées.



FIGURE 1.12 – Ecran du mode caméra/photo

1.4.3 Interfaces logiciel/logiciel

Format du fichier de données

Notre base de données est représentée sous la forme d'un fichier XML. Nous avons privilégié ce mode de représentation, par rapport à une base SQL/SQLite par exemple, en raison du fait que notre base n'est pas amenée à grandir démesurément (dans un futur plus ou moins proche en tous cas). De ce fait, l'utilisation d'une base SQL/SQLITE n'était pas pertinente.

Nous allons voir la structure de notre fichier XML sur un exemple :

```
<?xml version = '1.0' encoding="UTF8"?>
<!DOCTYPE arbre SYSTEM "arbre.dtd">

<arbre>
<branche id="b1" type="accueil" date="jj/mm/yyyy">
  <question id="q1" texte="Que voyez vous ?" visible="true">
    <media>
      <legende>Vous devez nous dire ce que vous avez vue !!!</legende>
    </media>
    <reponse id="r1" texte="6 pattes" visible="true">
      <media>
        
        <legende>c'est toto avec 6 pattes</legende>
      </media>
    </branche id="b2">
```

```

    <!-- ... -->
  </branche>
</reponse>
<reponse id="r2" texte="8 pattes , ailes et antennes absentes">
  <branche id="b3" type="Arachnide">
    <question id="q2" texte="Quelle est la taille de l'arachnide ?">
      <reponse id="r3" texte="Supérieur à 5 mm" >
        <resultat id="res1">
          <nom><!-- Nom de la morpho-espèce trouvée --></nom>
          <type>MEL1</type>
          <regimeAlimentaire>Prédateur</regimeAlimentaire>
          <informations><!-- Infos complémentaires sur la morpho-espèce -->
          <media>
            
          </media>
        </resultat>
      </reponse>
      <reponse id="r4" texte="Inférieur à 5 mm">
        <!-- ... -->
      </reponse>
    </question>
  </branche>
</reponse>
</question>
</branche>
</arbre>

```

A ce fichier XML, nous avons associés une DTD. Une DTD est en quelque sorte le modèle d'un fichier XML. Elle permet de décrire sa structure logique : nom des éléments, contenu, attributs, nombre d'occurrence autorisée... Voici la DTD de notre fichier XML :

```

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

<!ELEMENT arbre (branche)>

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

```



```

<!ELEMENT nom (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT regimeAlimentaire (#PCDATA)>
<!ELEMENT informations (#PCDATA)>

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

```

Format du fichier de résultat

Nous allons maintenant définir le format du fichier de sortie (le fichier de résultat). Ce fichier sera un fichier CSV, il pourra donc être ouvert et modifié facilement avec un tableur standard type Excel ou OpenOffice.

Il sera composé des colonnes suivantes :

- **Date** : date courante de l'observation
- **Campagne** : informations relatives à la campagne (nom, date de début...)
- **Session** : informations relatives à la session
- **Piège** : informations relatives au piège (coordonnées, nom...)
- **1 colonne par morpho-espèce** : cette colonne contiendra une valeur dans la case correspondant au résultat de l'observation.

Format du fichier des retours utilisateurs

Enfin, nous allons définir le format du fichier des retours utilisateurs. Ce fichier permet de centraliser tous les commentaires relevés par les utilisateurs de l'application mobile (questions incomplète, pas réponse correspondante, nouvelle morpho-espèce...).

Il sera composé des colonnes suivantes :

- **Date** : date de la soumission du commentaire
- **Historique questions/réponses** : historique complet des questions et des réponses auxquelles l'utilisateur a répondu
- **Photo insecte** : dans le cas d'une nouvelle morpho-espèce ou si l'utilisateur pense qu'il peut proposer une photo de meilleure qualité que celle proposée
- **Commentaire** : texte libre pour l'utilisateur

1.5 Architecture générale du système

La figure 1.13 présente le diagramme de composant de notre système.

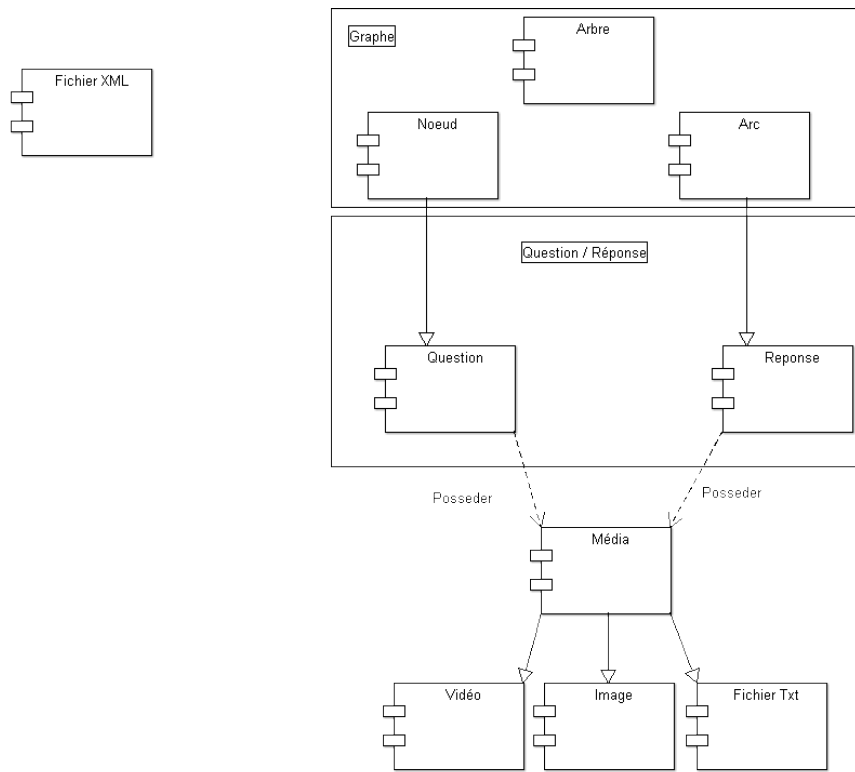


FIGURE 1.13 – Diagramme de composants

1.6 Description des fonctionnalités

1.6.1 Application PC

Visualisation de l'arbre

Cette fonction permet d'obtenir une représentation graphique de notre arbre XML. Nous allons utiliser un mélange entre l'arborescence classique de type "Windows" et l'arborescence utilisée dans l'explorateur de Mac OS (cf. IHM de l'application PC).

Importer un fichier XML

En entrée, nous avons un fichier XML valide. Notre fonction va lire ("parser") le fichier XML puis l'enregistrer en mémoire dans les types de données appropriés.

Exporter l'arbre en fichier XML

Cette fonction permet d'exporter l'arbre courant présent en mémoire. Cette fonction sera appelée régulièrement pour éviter toute perte de données (et notamment lors de la fermeture de l'application). En entrée, nous avons un arbre XML en mémoire. En sortie, nous devons obtenir un fichier XML valide contenant notre arbre.

Ajouter/modifier/supprimer questions/réponses

Cette fonction permet d'ajouter, de supprimer et de modifier une question ou des réponses de l'arbre (soit un élément de l'arbre XML). Cette modification n'est pas réalisée sur le fichier XML "physique" mais est opérée sur la structure en mémoire.

Ajouter/modifier/supprimer morpho-espèce

Même principe que pour la fonction précédente, mais opère sur les éléments de type "resultat" (soit les morphos-espèces).

Ajouter/modifier/supprimer médias

Même principe que pour la fonction précédente, mais opère sur les éléments de type "media". Cette action sera sûrement réalisée via "glisser-déplacer" du fichier dans l'interface de l'application.

Déplacer élément de l'arbre

Cette fonction n'est pas prioritaire. Elle permettra de déplacer un sous-arbre complet à un autre endroit. Il faudra prendre soin de garder la cohérence des données (réponses orphelines, questions sans réponses...).

1.6.2 Application mobile

Créer/modifier/supprimer une campagne

Une campagne est définie au minimum par un nom et une date de début (par défaut, la date de création de la campagne). Il est possible d'ajouter une date de fin, un lieu et un objectif.

Une campagne va être composée de un ou plusieurs pièges.

Il doit être possible de reprendre une campagne déjà commencé.

Créer/modifier/supprimer un piège

Un piège est défini au minimum par un nom. Il est possible d'ajouter un lieu (sous forme de coordonnées GPS) et une observation.

C'est dans le piège que l'on va stocker les différentes observations d'insectes que l'utilisateur va réaliser. De la même façon que pour une campagne, il doit être possible de reprendre un piège déjà créé.

Importer un arbre XML

Cette fonction permet de lire un fichier XML contenant une base de données d'insectes et de questions/réponses. En entrée, nous avons un fichier XML valide suivant notre DTD. En sortie, nous avons un arbre XML en mémoire.

Voir une question

Cette fonction permet d'afficher à l'utilisateur une question de l'arbre et ses réponses associées.

Répondre à une question

Cette fonction permet à l'utilisateur de répondre à une question. Selon la réponse qu'il a donné, cela va l'orienter vers la question correspondante dans l'arbre jusqu'à arriver à un élément terminal (morpho-espèce).

Visualisation de l'insecte

Cette fonction permet à l'utilisateur de choisir le mode de visualisation de l'insecte. Il a le choix entre :

- Mode photo : l'utilisateur prend en photo l'insecte à un instant t. Cette photo est gardée en mémoire et est affichée à l'utilisateur pour lui permettre de répondre aux différentes questions. L'utilisateur a la possibilité à tout moment de reprendre une photo de l'insecte.
- Mode vidéo : l'utilisateur disposera d'une image en temps réel de l'insecte. Cela lui permettra d'observer l'insecte sous différents angles en déplacement l'insecte ou la tablette.

A tout moment, l'utilisateur peut switcher entre ces 2 modes à l'aide d'un interrupteur.

Poster une alerte

Cette fonction permet à l'utilisateur de signaler un quelconque problème en relation avec l'application. Ces problèmes peuvent être de différents types :

- Problème d'indécision : l'utilisateur n'arrive pas à se déterminer sur la réponse à choisir (soit par manque de compétence ou d'expérience, soit parce que la question ou les réponses ne sont pas assez claires ou précises). Ce problème doit être remonté à l'équipe pour qu'il soit pris en compte (modification de la question ou des réponses).
- Problème de morpho-espèce inconnu : dans ce cas, le problème doit être remonté à l'équipe pour quelle puisse modifier l'arbre XML pour prendre en compte cette nouvelle morpho-espèce.
- Problème technique : l'application affiche un message d'erreur ou plante inopinément. Ce bug doit être remonté pour être corrigé par l'équipe technique.

1.7 Conditions de fonctionnement

1.7.1 Capacités

Au niveau de l'application PC :

- nombre max de terminaux : l'application mobile est susceptible d'être installée et utilisée sur plusieurs tablettes, mais de manière indépendante (pas de communication inter-application directe mais seulement via le logiciel PC).
- capacité max de stockage : la tablette dont nous disposons sera équipée de 64 Go de mémoire de stockage. Cela permettra en grande partie de stocker les différents médias associées à l'application.

Au niveau de l'application PC, il n'y aura probablement qu'une seule instance qui fonctionnera de manière indépendante. Donc pour le moment, nous n'aurons pas à gérer les problèmes de cohérence des données entre plusieurs bases de données.

1.7.2 Contrôlabilité

Les actions de l'utilisateur doivent être enregistrées par l'application, d'une part pour faire un feedback à l'équipe INNOPHYT (erreurs, résultats incorrects...) et d'autre part pour assurer un suivi à l'utilisateur de l'application.

1.7.3 Sécurité

Pour le moment, pas de système d'authentification n'est à mettre en place. Les utilisateurs de l'application PC disposent des droits pour tout faire et de même sur l'application mobile.

Plan de développement

2.1 Découpage du projet en tâches

2.1.1 Rédaction de la modélisation UML

- Réalisation des deux diagrammes de cas d'utilisation : un pour l'application mobile et un autre pour l'application PC
- Réalisation du diagramme de classes
- Réalisation du diagramme de composants

2.1.2 Création du fichier XML et de sa DTD

Réflexion sur l'organisation du fichier XML et les informations à y faire apparaître (éléments, données, attributs...).

Suite à cela, rédaction de la DTD associée à ce fichier XML. Enfin, une fois la structure finale définie et la DTD rédigée, écriture du fichier XML complet relatif à l'arbre qui nous a été fourni par l'équipe INNOPHYT (au format Excel).

2.1.3 Application PC

Réalisation de l'IHM de l'application PC

Réflexion au choix d'un mode de visualisation efficace et performant pour les données (arborescence type Windows, explorateur type MacOS, représentation sous forme de carte...).

Développement de la classe de lecture/écriture de fichier XML

Les modules sous-jacents sont donc "importer fichier XML" et "exporter fichier XML".

Développement du module de visualisation de l'arbre

2 types de visualisation : l'arborescence type "Windows" qui reprend les questions et les panneaux type "Mac OS" qui liste les réponses et les médias relatifs à l'élément cliqué dans la 1ère arborescence.

Développement du module de modification de l'arbre XML

Édition, ajout et suppression de noeud (question, réponse, morpho-espèce...).

- Ajouter balise
- Supprimer balise
- Modifier balise

Développement du module de déplacement d'éléments de l'arbre

Un élément seul, un sous-arbre complet... Cette tâche n'est pas prioritaire, elle sera effectuée dans un 2ème temps.

Développement du module d'ajout de média

Ce module doit permettre à l'utilisateur d'ajouter des médias via un drag and drop du fichier contenu dans son ordinateur vers la fenêtre de notre application.



Développement du module de visualisation des médias

Permet d'afficher les différents médias associés à un élément (question/réponse/résultat). Ce module doit être adapté à plusieurs types de médias : photos, vidéos, éventuellement animation... Il faut prendre soin de gérer les différents formats possibles pour chaque type de média.

Développement du module de recherche des informations manquantes

Ce module permet à l'utilisateur de savoir quels sont les éléments où il manque de l'information.

2.1.4 Application mobile

Prise en main de l'environnement de développement Android

Réalisation de l'IHM de l'application mobile

Réflexion à la réalisation d'une interface claire et accessible à tout type d'utilisateur (même ceux n'ayant pas l'habitude d'utiliser l'outil informatique). Il faut penser que l'application est amenée à être utilisée dans un cadre parfois peu idéal (en extérieur, luminosité forte, gants de travail...).

Développement des IHM en Java

Développement de la logique de l'application

- Gestion des campagnes (création/modification/suppression)
- Gestion des sessions (création/modification/suppression)
- Gestion des pièges (création/modification/suppression)
- Affichage et parcours des questions/réponses

Développement du module de log

Permet d'obtenir le fichier des retours utilisateurs comme défini dans "Interfaces logiciel/logiciel".

2.2 Planning

Étant donné que nous utilisons la méthodologie Agile "Scrum", nous n'avons pas à faire de diagramme de Gantt pour le moment. Il sera réalisé au fur et à mesure du projet.

- **Sprint 0** : nous avons fait l'inventaire des différentes "User-stories" du projet. Pour chacune d'elle, nous avons détaillés les tâches à réaliser. Nous avons ensuite reparti ces tâches entre les différents membres de l'équipe.
- **Sprint 1** : nous avons précisé, avec l'aide de notre encadrant, le format du fichier XML contenant notre base de données. Nous avons aussi validé les différentes IHM de nos applications PC et mobile. Ce sprint 1 a aussi été l'occasion de prendre en main les environnements de développement et les technologies associées avec Android et QT.

Le prochain sprint sonnera le début d'une première phase de développement. Nous avons divisé l'équipe en 2 sous-équipe : l'une d'elle travaillera sur la partie mobile et l'autre sur la partie PC. A la fin de ce sprint, nous devrions pouvoir fournir 2 applications fonctionnelles.