Université de Montréal

Système de listes de vérification interactives du niveau de conformité des maquettes avec les recommandations des fabricants de plateformes mobiles

par

Nadir GHEZZAL
Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

Août 2011

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé :

Système de listes de vérification interactives du niveau de conformité des maquettes avec les recommandations des fabricants de plateformes mobiles

présenté par :
Nadir GHEZZAL

a été évalué par un jury composé des personnes suivantes :

Guy Lapalme
président-rapporteur

Jean-Yves Potvin
directeur de recherche

Jean-Marc Robert
co-directeur

Michel Desmarais
membre du jury

Mémoire accepté le : 21 octobre 2011

Résumé

La demande d'applications pour les plateformes mobiles a explosé ces dernières années. Chaque compagnie souhaite maintenant offrir pour ces nouveaux appareils les mêmes services que ceux offerts traditionnellement sur internet. Cependant, ces entreprises n'ont bien souvent que peu ou pas de connaissances concernant le développement et le déploiement de tels services. Généralement, la solution choisie consiste à externaliser ce travail en le sous-traitant à une autre compagnie. Dans ce contexte, il est souvent compliqué de s'assurer d'une part que le sous-traitant respecte le cahier des charges et d'autre part que le travail fourni est conforme aux recommandations émises par les fabricants des plateformes mobiles. Afin de pallier au second problème, nous avons créé un système de listes de vérification interactives pour plateformes mobiles. Ce système permet d'évaluer le niveau de conformité des différents composants de l'interface d'une application développée pour une plateforme mobile avec les recommandations du fabricant de cette plateforme. La solution retenue permet de se concentrer sur certains éléments particuliers de l'interface et pallie ainsi aux limites des listes de vérification classiques qui sont souvent trop longues et peu pratiques. La solution retenue offre de plus la possibilité de comparer facilement les caractéristiques des plateformes mobiles entre elles.

Pour réaliser ce système, nous avons consulté de nombreux documents portant sur l'univers des plateformes mobiles afin de mieux appréhender leurs spécificités. Suite à l'étude de différentes plateformes, deux d'entre elles, soit *iOS* et *BlackBerry*, ont été retenues (il faut noter que la compagnie où s'est effectuée la recherche disposait déjà d'applications pour ces plateformes). Nous avons ensuite analysé plus finement la documentation technique fournie par chacun des fabricants afin d'en extraire les points importants. Afin que les données soient comparables, nous avons créé une nomenclature commune regroupant les composants de l'interface graphique en grandes familles (ex. : barres, saisie d'information, …) en tenant compte également du type d'interaction avec l'appareil (ex. : écran tactile). Nous avons ensuite conçu une solution permettant d'évaluer le niveau de conformité d'une application. L'idée retenue se base sur des listes de vérification permettant de systématiser le processus d'évaluation. Pour pallier aux limites de

ces listes, souvent trop longues, nous permettons à l'utilisateur, via un ensemble de filtres, de se concentrer sur un sous-ensemble de composants de l'interface qu'il souhaite évaluer.

L'implémentation de cette solution a été réalisée en totalité grâce à des technologies libres et de standard ouvert. De cette façon, nous nous sommes assurés de la portabilité de la solution développée et de l'absence de coûts supplémentaires liés à l'achat de licences. Le système utilise les standards web et repose sur une architecture basée sur le système d'exploitation GNU/Linux, le serveur web Apache, la base de données MySQL et le langage de scripts PHP. Ces logiciels ont déjà fait leurs preuves aussi bien pour les entreprises que pour les particuliers. De plus, la communauté très active qui s'est constituée autour de ces logiciels assure un bon support et une grande stabilité. Après avoir fait le choix de l'environnement de développement, la phase d'implémentation s'est ensuite déroulée en plusieurs étapes. Tout d'abord, l'information a été structurée selon la nomenclature mentionnée plus haut afin de créer le schéma de la base de données. Ensuite, toutes les pages ont été codées avec le langage à balises HTML et les scripts PHP pour le côté serveur et avec JavaScript pour le côté client. Enfin, l'information peut être affichée grâce aux feuilles de style en cascade (CSS), une technologie web permettant de séparer le fond de la forme en matière de mise en page.

Nous avons choisi un modèle de développement itératif qui a impliqué les principaux utilisateurs dès les premières étapes de la conception du système. Cette implication s'est poursuivie jusqu'à la fin du projet afin de s'assurer que les fonctionnalités mises en place répondaient aux attentes. L'architecture modulaire qui a été retenue permet également d'adapter facilement le système aux besoins futurs. Afin de mieux comprendre comment on utilise le système, on passe en revue les différentes étapes nécessaires à la réalisation d'une évaluation. Enfin, on a réalisé une étude avec quatre utilisateurs pour évaluer l'utilisabilité du système et recueillir leur niveau de satisfaction.

Mots-clés : Listes de vérification, recommandations ergonomiques, plateformes mobiles, site web interactif.

Abstract

The need for mobile platforms has increased in the last decade. Companies offering traditional internet services now want to move their applications on these new devices. But, most of the time, these companies do not really have the knowledge to create such applications and often ask a third party company to do the job for them. In this context, it is difficult to evaluate if the solution developed follows the recommendations of the device maker. A system based on interactive check lists has thus been created for this purpose. This system enables the evaluation of the conformity level of an application with the recommendations of the device maker, thanks to numerous filters that let the user focus on specific parts of the graphic interface. The idea behind the project was first to address some problematic issues with classical check lists and to enable the comparison of several mobile platforms with regard to specific interface components.

To create this system, a lot of information about mobile platforms has first been collected. After reviewing many mobile platforms, it was decided to focus on *iOS* and *BlackBerry*, since the company where the research was performed had already applications running on them. For each platform, the major recommandations to be satisfied were identified. Also, to be sure that the extracted recommendations could be compared, a common nomenclature has been created, where the recommandations are grouped by interface components (ex: bars, data input,…) and by the type of interaction (ex: touch screen). After these preliminary steps, a solution for evaluating the level of compliance was created. We favored a check list approach because it offers a systematic evaluation process. To avoid lists of excessive length, filters were introduced in the system to allow the user to focus on particular aspects of the interface.

The implementation was totally realized with open source technologies and open standards. This choice was motivated by the portability of the developed system and by the absence of licence fees. The system relies on web standards and runs on an architecture made of the CNU/Linux operating system, Apache web server, MySQL database and the script language PHP. This software has already proven its reliability for enterprises and for home users. Furthermore, the community evolving around this software offers a good

support and ensures a high level of stability. After setting up the development environment, the implementation phase was engaged and took place over a number of phases. The first phase was the creation of the database structure, using the aforementioned nomenclature. The next phase was dedicated to the coding of the different web pages, thanks to the tag language HTML and the PHP scripts on the server side and JavaScript on the client side. Finally, the web page setting was developed using the cascading style sheet (CSS), a web technology that segregates the substance from the style of the web content.

We chose an iterative development model where the end users were involved from the early stages of the project. This approach provides a guarantee that the user requirements are fulfilled and that any new developments will be in accordance with the expectations. Furthermore, the system is such that it can be easily modified to tackle future needs. To be able to understand how we use the systeme, we are reviewing the different steps needed to realise an evaluation. Finally, we have made a study with four users to evaluate the usability of the system and to gather their satisfaction level.

Keywords : Check lists, mobile platforms, ergonomic recommendations, interactive web site.

Table des matières

Liste des tableaux

Liste des figures

Liste des acronymes et abréviations

AMP	Apache, MySQL, PHP
API	Application Programming Interface
CLDC	Connected Limited Device Configuration
CSS	Cascading Style Sheet
DIAL	Device Independent Authoring Language
DIWG	Device Independent Working Group
DoJa	*DoCoMo Java*
HTML	Hypertext Markup Language
JSON	*JavaScript Object Notation*
KISS	Keep It Simple Stupid
KVM	Kilobyte Virtual Machine
MIDP	*Mobile Information Device Profile*
MVC	*Model View Controller*
PHP	originalement Personal Home Page maintenant Hypertext Preprocessor
SDK	Software Development Kit
SGBD	Système de Gestion de Base de Données
SQL	Structured Query Language
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

Remerciements

À Louise Lanoix et Darcy Noonan, mes encadrants chez Air Canada, pour m'avoir permis de réaliser cette recherche au sein de leur équipe, pour les ressources mises à ma disposition et les conseils donnés. Un grand merci aussi à toute l'équipe mobile pour leur accueil et à toutes les personnes d'Air Canada qui ont pris le temps de répondre à mes questions et qui m'ont ainsi aidé dans mon travail.

À Jean-Marc Robert, mon professeur et codirecteur de recherche à l'École Polytechnique de Montréal, pour m'avoir proposé ce sujet de maîtrise, pour l'aide précieuse qu'il m'a procurée à chacune des étapes, pour son soutien et ses conseils et pour m'avoir permis de découvrir et de mieux connaître le domaine de l'ergonomie du logiciel.

À Jean-Yves Potvin, mon directeur de recherche à l'Université de Montréal, pour m'avoir permis de réaliser cette maîtrise, pour son aide déterminante au niveau administratif, pour ses suggestions, ses corrections et pour m'avoir permis d'organiser ma maîtrise en fonction de mes centres d'intérêt.

Introduction

En une trentaine d'années, le nombre de téléphones mobiles dans le monde est passé de quasi nul à environ 4 milliards d'unités. Cette croissance est fulgurante surtout si on la compare au développement des ordinateurs personnels qui sont actuellement moitié moins nombreux. Grâce à la convergence technologique, nous avons maintenant des téléphones qui sont de véritables ordinateurs embarqués permettant de lire différents contenus multimédias, de naviguer sur le NET, de jouer, de prendre des photos ou encore de s'orienter grâce à leur système GPS intégré. Cette multitude de possibilités offertes par les plateformes mobiles est très intéressante et a permis de démocratiser ces appareils qui sont maintenant devenus des objets de consommation courants (Klockar, Carr, Hedman, Johansson, & Bengtsson, 2003).

Cet engouement pour les téléphones mobiles a également favorisé l'émergence d'un nouveau marché, celui des applications mobiles. En effet, avec l'apparition des téléphones intelligents, la demande de logiciels pour ces plateformes a explosé. On assiste à une véritable effervescence car chaque compagnie souhaite maintenant offrir pour ces nouveaux appareils les services qu'elle offrait traditionnellement sur internet. Cependant, ces entreprises n'ont que peu ou pas connaissances en matière de développement et de déploiement de tels services et choisissent souvent d'externaliser ce travail en le sous-traitant à une autre compagnie. Cette approche a toutefois ses limites car, d'une part, les fabricants de plateformes mobiles ont mis en place un ensemble de recommandations afin que les applications puissent fonctionner correctement tout en garantissant une certaine homogénéité et une meilleure expérience utilisateur alors que, d'autre part, les compagnies de sous-traitance qui développent des applications pour les plateformes mobiles s'appliquent avant tout à respecter le cahier des charges fonctionnelles fourni par l'entreprise requérant leurs services. Or, l'entreprise qui fait appel à un sous-traitant pour développer une application n'a souvent qu'une connaissance limitée des recommandations propres à chaque plateforme et des pratiques à observer lors de la création d'applications mobiles. Il devient alors difficile pour l'entreprise d'évaluer correctement l'application. Souvent, l'évaluation ne touchera qu'au respect ou non des différents points du cahier des

charges. L'entreprise est donc contrainte de faire confiance au sous-traitant, pour tout ce qui concerne le niveau de conformité avec les recommandations émises par les fabricants des plateformes mobiles.

À l'heure actuelle, la seule solution possible pour l'entreprise est d'analyser la documentation technique fournie par les fabricants d'une plateforme et de vérifier si le produit fini respecte ou non les différentes recommandations. Cette tâche est difficilement réalisable, en particulier pour des équipes dont les effectifs sont réduits et qui ont généralement d'autres mandats, en plus de la rédaction du cahier des charges. Un autre problème majeur vient de ce que les recommandations émises par les fabricants prennent souvent la forme d'un document texte linéaire. Ces documents, constitués de plusieurs dizaines de pages, ne permettent pas de se concentrer aisément sur un composant précis de l'interface que l'on cherche à évaluer. De plus, chacun des fabricants a sa propre nomenclature et organise ses recommandations à sa manière ce qui ne simplifie pas les choses. Enfin, l'introduction des premiers appareils grand public datant des années '90 seulement, la littérature scientifique sur le sujet est encore assez limitée (Kjeldskov & Graham, 2003). Il n'existe, par exemple, aucun outil générique permettant de mener une évaluation exhaustive du niveau de conformité d'une application par rapport aux recommandations émises par les fabricants de plateformes mobiles.

Une approche s'inspirant des listes de vérification, mais qui introduit un certain nombre de nouveautés, a donc été développée afin de répondre en partie à ces problèmes. Le principe général consiste à générer une liste de recommandations à partir de la documentation fournie par les concepteurs de plateformes mobiles. Une recommandation s'applique ici à un composant particulier de l'interface et comprend un ensemble de points à respecter afin de garantir la conformité de l'application. Le but de cette approche est de fournir un outil aux responsables de l'évaluation au sein de la compagnie partenaire (*Air Canada*) leur permettant de connaître le niveau de conformité du travail reçu, grâce à un score basé sur des données concrètes. Afin d'ajouter plus de flexibilité à ces listes de recommandations, une version numérique a été adoptée. Cette dernière est constituée de deux éléments. D'une part, une interface permettant à l'utilisateur de choisir facilement ce

qu'il souhaite évaluer et, d'autre part, une base de données au sein de laquelle l'ensemble de l'information recueillie est stockée. Cette approche a été retenue car elle permet d'afficher les recommandations de façon dynamique en fonction de critères sélectionnés par l'utilisateur. On remédie ainsi aux deux principales faiblesses des listes de contrôle traditionnelles, soit leur longueur et leur manque de flexibilité.

Objectifs

Le premier objectif de cette recherche est de concevoir, développer et mettre au point un outil permettant d'évaluer le niveau de conformité des maquettes fournies par les sous-traitants avec les recommandations émises par les fabricants de plateformes mobiles. Un objectif secondaire est de disposer d'une base de connaissances permettant de comparer les différentes plateformes mobiles en regard des composants de leur interface.

Toutes les étapes nous ayant permis d'atteindre ces deux objectifs sont décrites dans les chapitres suivants.

Structure du mémoire

Le **chapitre 1** fait d'abord un tour d'horizon des principales plateformes mobiles sur le marché et décrit leurs spécificités. Des observations sont ensuite rapportées suite à des manipulations que nous avons réalisées avec un certain nombre d'appareils. Le chapitre se termine avec des principes généraux qui doivent être observés lorsque l'on souhaite offrir un service sur une plateforme mobile.

Le **chapitre 2** décrit le contexte dans lequel le système a été conçu et en présente ses grandes lignes. La première partie introduit le type de problématique auquel nous devions faire face. Nous décrivons ensuite la solution apportée à cette problématique.

Le **chapitre 3** décrit l'implémentation du prototype basée sur l'architecture LAMP (Linux, Apache, MySQL, PHP). Nous présentons en particulier la structure ou schéma de la base de données. Nous expliquons ensuite comment s'est effectuée la création des pages web en présentant les technologies mises en place et leurs spécificités. Enfin, nous passons

en revue l'interface du système en décrivant ses différents éléments ainsi que leur fonctionnement.

Le **chapitre 4** présente une utilisation typique du système. On y voit les étapes à suivre pour ajouter des recommandations ainsi que le déroulement d'une évaluation du niveau de conformité en prenant comme exemple la barre d'outils et de navigation de la plateforme iOS.

Le **chapitre 5** présente une expérience qui nous a permis d'obtenir une évaluation sommaire du système mis en place. Les résultats obtenus sont présentés et accompagnés d'une discussion.

La **conclusion** résume les avantages ainsi que les limites de la solution proposée. Quelques améliorations possibles sont suggérées afin d'améliorer le système et ainsi offrir une meilleure expérience utilisateur. Enfin, quelques pistes de recherche sont proposées.

CHAPITRE 1 : Deux plateformes mobiles types: iOS et Blackberry

Ce chapitre présente d'abord les caractéristiques physiques propres aux téléphones intelligents retenus ainsi que le type d'interaction privilégié afin de mieux cerner les capacités et les limites de ces appareils. Nous introduisons ensuite le système d'exploitation ainsi que l'interface de programmation de ces plateformes. Enfin, nous passons en revue certains principes de base lorsqu'on souhaite offrir un service sur un appareil mobile. En analysant en détail les deux plateformes, il sera possible de mieux saisir leur fonctionnement et de mettre en contexte les recommandations ergonomiques qui leur sont propres, lorsqu'elles seront évoquées plus loin dans ce travail.

La majeure partie des plateformes disponibles sur le marché peuvent être regroupées en trois catégories :

- **Les plateformes natives** comparables à *iOS* et *BlackBerry*.
- **Les platformes web** reposant sur le réseau internet.
- **Les plateformes hybrides** qui utilisent des technologies web pour générer des applications natives.

Au total, 13 plateformes sur environ une vingtaine ont été analysées. Nous avons fait le choix de ne présenter ici que les plateformes *iOS* d'*Apple* et *BlackBerry* de *RIM* puisque le système développé repose actuellement sur ces deux systèmes. Les autres systèmes étudiés sont présentés à l'Annexe 1.

1.1 Caractéristiques physiques des plateformes

Au deuxième trimestre 2010[1], deux des principaux systèmes d'exploitation disponibles pour les téléphones intelligents sont *iOS* (14 %) et *BlackBerry OS* (18 %). Il est donc important de prendre en considération les caractéristiques des appareils qui reposent sur ces systèmes

1 http://www.gartner.com/it/page.jsp?id=1421013

d'exploitation afin de mieux comprendre les possibilités qui leur sont offertes. Dans ce but, nous avons choisi un téléphone fonctionnant avec chacun des deux systèmes afin de les comparer. Les appareils retenus sont le *Apple iPhone* et le *Blackberry Bold 9000* (cf. Tableau 1).

	 **Apple iPhone**	 **BlackBerry Bold**
Dimensions	61 x 115 x 11.6 mm	66 x 114 x 14 mm
Écran	– 320 x 480 pixels 3.2" – 16M couleurs – Écran tactile multipoints – Accéléromètre auto-rotation	– 480 x 320 pixels 2.6" – 65 536 couleurs
Périphérique de Saisie et de Navigation	– Clavier virtuel – Bouton retour arrière	– Clavier complet (format QWERTY) – Boule de commande (Trackball)
Java	Non compatible	Java MIDP 2.0
OS	iOS (basé sur Mac OS)	BlackBerry OS
Messagerie	SMS, Email	SMS, MMS, Email, IM
Navigateur	HTML (Safari) Acid3 test[2] 100/100	HTML Acid3 test 12/100
Formats Supportés	AAC, AAC Protégés, MP3, MP3 VBR, Audible (formats 1, 2, and 3), Apple Lossless, AIFF, WAV, H.264 video, jusqu'à 1.5 Mbps, 640 x 480 pixels, 30 frames par seconde, m4v, .mp4, et .mov	Word, Excel, PowerPoint, PDF, MP4, WMV, H.263, H.264, MP3, eAAC+, WMA

Tableau 1: Caractéristiques physiques du *iPhone* et du *BlackBerry Bold*

Pour chacun des téléphones, nous avons relevé un certain nombre de caractéristiques qui leur sont propres en se référant à la documentation fournie par les fabricants respectifs. Nous avons également pu manipuler ces appareils afin de mieux

2 Test pour navigateur web mis en place par le groupe Web Standards Project et écrit par Ian Hickson en 2008 (http://acid3.acidtests.org/).

comprendre le type d'interaction offert par chacun. Ces manipulations ont eu lieu à différentes reprises et portaient sur les fonctionnalités de base de chacun des téléphones, comme la navigation au sein de l'appareil ainsi que sur le web, la saisie de texte et l'utilisation de différents programmes natifs. Cette expérience nous a permis de mieux nous imprégner du « look and feel » de chacun des téléphones.

1.1.1 iPhone

L'interaction avec un *iPhone*, marque déposé de la société *Apple*, se fait essentiellement avec l'écran tactile à l'exception d'un bouton de retour arrière au bas du téléphone, d'un bouton de réglage du volume, d'un bouton configurable et du bouton d'allumage (cf. Figure 1). Globalement, toutes les opérations s'effectuent avec un ou deux doigts car l'écran est multipoints. L'outil principal au coeur du *iPhone* est le moteur de recherche interne qui reprend la fonctionnalité de *spotlight* sur *Mac OS X*. L'ensemble des applications du *iPhone* se présentent sous la forme de boutons que l'on peut facilement sélectionner avec le pouce et sont disposées sur un ou plusieurs écrans. Pour la saisie de texte, l'*iPhone* utilise exclusivement un clavier virtuel utilisant la technique *Shift* (Vogel & Baudisch, 2007) qui indique quelle la touche a été appuyée et qui ajoute un un signal sonore pour informer l'utilisateur que la touche a bien été activée. Le téléphone oriente automatiquement l'interface en fonction de la position de l'appareil grâce à un accéléromètre. Il est possible de zoomer à tout moment dans le navigateur en rapprochant le pouce et l'index ou bien en tapant deux fois de suite sur la partie souhaitée de la page web. Pour la sélection des hyperliens, deux possibilités s'offrent à l'utilisateur. La première est classique et dirige l'utilisateur vers le lien sur lequel il a cliqué. La seconde est déclenchée lorsque l'utilisateur appuie sur un lien pendant quelques secondes. Un menu s'ouvre alors, proposant à l'utilisateur soit d'ouvrir le lien, soit

Figure 1: Écran d'accueil verrouillé d'un *iPhone*

d'ouvrir ce dernier dans une nouvelle fenêtre, soit de copier ce dernier ou bien d'annuler l'opération. La technique *Shift* est également utilisée lorsque l'on appuie sur un texte pendant quelques secondes afin de sélectionner des portions de texte et de les copier, couper et coller. Une fonctionnalité intéressante du *iPhone* est l'utilisation d'une boîte de choix pour les listes déroulantes. Cette boîte apparaît automatiquement dans la partie inférieure de l'écran et offre une vue qui permet de sélectionner plus facilement des éléments du menu. Les déplacements horizontaux et verticaux dans *l'iPhone* sont gérés par des glissements de doigts dans le sens du déplacement souhaité. Il faut noter que *l'iPhone* ne supporte pas *Flash* et que les développeurs ne prévoient pas le rendre compatible avec cette application. De ce fait, tous les sites web utilisant cette technologie souffrent d'un problème d'affichage de leur contenu. *Apple* garde une mainmise sur l'appareil en verrouillant certaines fonctionnalités comme le *Bluetooth*, qui n'est utilisable qu'avec des kits mains libres compatibles et ne permet pas de faire de transfert de données vers d'autres périphériques. Elle limite également les technologies de développement permettant de créer de nouvelles applications pour sa plateforme en refusant de distribuer sur son *AppStore* les produits non conformes.

Le système d'exploitation n'est pas vraiment multitâches bien qu'il permette à certaines applications de s'exécuter en tâche de fond comme le lecteur de musique. Mais, en général, il est nécessaire de fermer une application avant de passer à la suivante. Cette limitation sera cependant levée avec les nouvelles versions du système d'exploitation. Le *iPhone* est l'appareil avec le plus grand nombre d'applications disponibles (environ 10 000) ce qui permet à ses utilisateurs de le personnaliser facilement. Le téléphone ne possède pas de GPS (problème réglé avec les modèles plus récents), ce qui le prive des possibilités de géolocalisation. Enfin, l'absence de 3G (problème également réglé avec les modèles plus récents) ne permet pas au téléphone de naviguer à grande vitesse. Il faut donc faire attention à la quantité de données que l'on transfère.

1.1.2 BlackBerry

L'interaction avec le *BlackBerry,* marque déposée de la société *RIM,* repose principalement sur la boule de commande (trackball) (cf. Figure 2) ou du pavé tactile (cf. Figure 3). Cette interface permet de bouger le curseur, de cliquer et de naviguer à travers les différents champs de saisie ou les différents éléments sur lesquels l'utilisateur est susceptible de cliquer (fonctionnement similaire à la touche tabulation sur un clavier normal). L'appareil possède un clavier QWERTY complet. Cette interface de saisie haptique a l'avantage d'être plus commode pour faire de longues saisies. Contrairement au clavier virtuel, elle évite à l'utilisateur expert de regarder constamment où il tape. De plus, compte tenu du grand nombre de touches, il existe de nombreux raccourcis clavier permettant à l'utilisateur d'accéder plus rapidement à certaines fonctionnalités. Le *BlackBerry* est un produit avant tout destiné aux entreprises. En effet, il leur offre de nombreux services afin de contrôler les appareils de leurs employés et de les synchroniser avec les données qu'elles souhaitent partager. Cette caractéristique est à prendre en considération lorsque l'on veut installer des applications sur la plateforme, car il peut exister des limitations de la part des administrateurs du parc de téléphones. Au sein de chaque application, il est possible d'accéder à un menu d'options grâce à la touche menu symbolisée par le logo du

Figure 2: BlackBerry Bold 9000

Figure 3: BlackBerry Curve

BlackBerry. L'inconvénient de ce menu est qu'il ne boucle pas. Il est donc nécessaire de passer par toutes les options afin de trouver celle que l'on cherche. Cela peut être particulièrement fastidieux pour une option se trouvant tout au bas d'un menu comme la fermeture d'une application.

Il existe plusieurs options afin de zoomer sur certaines pages. La première est l'utilisation de la boule de commande pour faire un zoom avant. Il faut alors appuyer sur la touche ALT pour faire un zoom arrière. Il est aussi possible d'utiliser les touches I et O. Cependant, certains modèles ne possèdent pas la possibilité de zoomer sur les pages web qui sont affichées à l'écran. Il faut donc tenir compte de cette particularité dans le développement de l'interface. De plus, l'appareil en mode saisie est fait pour être utilisé à deux mains puisque de nombreux caractères sont accessibles par des combinaisons de touches nécessitant l'appui simultané de la touche ALT et du caractère souhaité. Par défaut, ce sont les caractères qui sont sélectionnés et des manipulations spéciales sont nécessaires pour saisir les chiffres du clavier. Il faut donc prendre garde si l'on souhaite offrir des raccourcis clavier faisant appel à des numéros pour accéder aux différentes rubriques. Lors du choix dans un menu déroulant, l'interface se comporte à la façon d'un ordinateur, c'est-à-dire qu'elle déroule la liste et laisse à l'utilisateur le soin de faire le bon choix. Toutefois, lorsque l'utilisateur clique sur une lettre, il est automatiquement dirigé vers la première entrée commençant par cette lettre. Si l'utilisateur continue à cliquer sur cette lettre, la sélection se porte alors sur l'entrée suivante qui commence avec cette lettre, et ainsi de suite.

Globalement, la famille des *BlackBerry* est assez large, avec 12 catégories constituées de un à neuf téléphones. La plupart des modèles partagent toutefois les mêmes métaphores d 'interaction, à l'exception des 4 catégories qui possèdent un écran tactile. Par exemple, la famille baptisée *Storm* dont les appareils sont complètement tactiles ont la particularité de posséder un écran tactile cliquable (cf. Figure 4). Cette caractéristique oblige à revoir certains aspects de l'interface afin de les adapter aux contraintes du nouvel appareil (ex. : la taille des boutons).

Figure 4: BlackBerry Storm 2

1.2 Systèmes d'exploitation

1.2.1 iOS

Précédemment appelé *iPhone OS*, le système d'exploitation d'*Apple* est disponible sur les téléphones intelligents *iPhone*, les baladeurs multimédias *iPod touch* et la tablette électronique *iPad*. Introduits en 2007 avec la sortie de la première génération des *iPhones*, le système d'exploitation d'*Apple* ainsi que son appareil ont révolutionné l'interaction avec les appareils mobiles et ont contribué à populariser les écrans tactiles multipoints.

Apple a aussi révolutionné le mode de distribution des applications grâce à l'*App Store*. Comme pour le *BlackBerry*, *Apple* est à la fois le développeur du matériel et du logiciel nécessaire pour faire fonctionner la plateforme. L'*iOS* est une solution propriétaire sur laquelle « la firme à la pomme » détient une mainmise totale. De plus, *Apple* privilégie une politique du secret quant aux spécifications de ses appareils. De ce fait, il est très difficile de savoir comment fonctionnent et interagissent les différents composants de la plateforme, bien que la documentation fournie par le constructeur permette quand même d'obtenir certaines informations.

On sait par exemple que le noyau de la plateforme est basé sur une variante du système d'exploitation pour ordinateur *Mac OS X,* au-dessus duquel se retrouve un certain

nombre de couches de services, chacune offrant un certain type de fonctionnalités (cf. Figure 5).

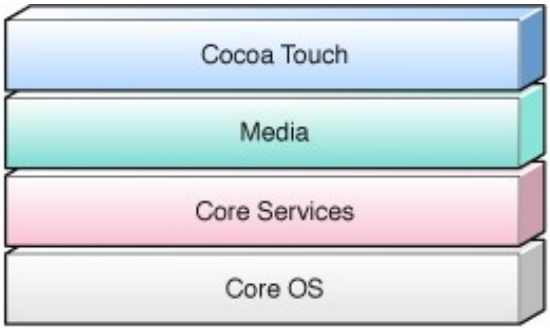


Figure 5: Couches de services iOS [18]

Les couches les plus basses sont *Core OS* et *Core Services* qui contiennent les interfaces fondamentales d'*iOS*, incluant l'accès aux fichiers, aux données de bas niveau, aux services Bonjour (implémentation d'*Apple* de *Zeroconf*, ensemble de protocoles permettant d'automatiser la création d'un réseau IP) et aux interfaces de connexion. Elles sont rédigées majoritairement en langage C et incorporent des technologies comme *Core Foundation*, *CFNetwork*, *SQLite*, ainsi que l'accès aux fils de discussion (thread) *POSIX* et *UNIX*. La couche média est chargée de la gestion des graphiques en 2D et 3D, de l'audio et de la vidéo. Elle repose sur des technologies écrites en C comme *OpenGL ES*, *Quartz*, et *Core Audio*. Elle contient aussi *Core Animation* qui est un moteur d'animation avancé écrit en *Objective-C*. Enfin, la couche *Cocoa Touch* est codée en *Objective-C* et fournit une infrastructure permettant aux différents éléments de l'application de dialoguer entre eux (cf. Figure 6). Elle possède également le *UIKit* qui est une API permettant de gérer les fenêtres, les vues ainsi que les boutons de contrôle des applications.

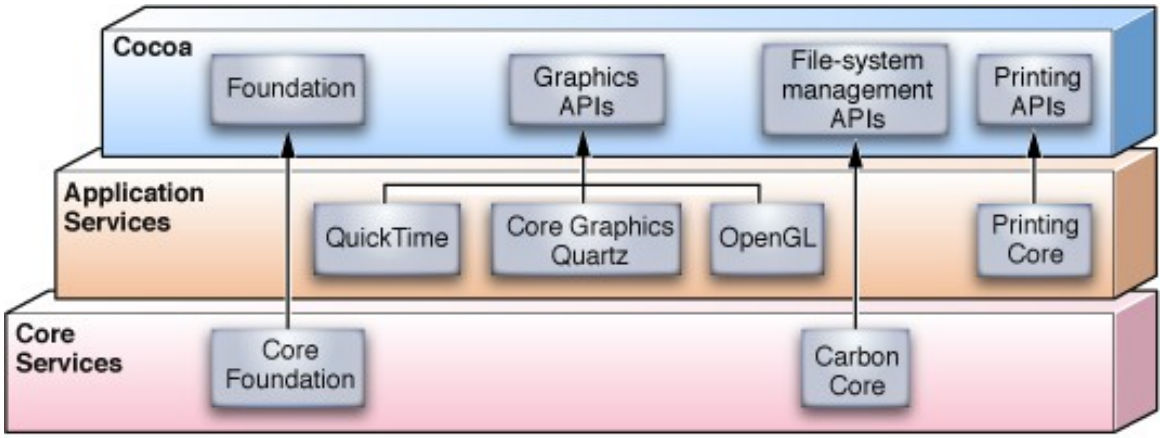


Figure 6: Interaction de Cocoa avec les autres couches de services [18]

L'interface graphique de iOS repose en grande partie sur *Cocoa*, un kit de développement permettant de mettre au point des applications pour le *iPhone*. Cette API repose sur l'approche orientée objet MVC (*Modèle Vue Contrôleur*) et offre des patrons de conception afin d'assister la création de logiciels. *Cocoa* offre également des librairies graphiques permettant d'organiser l'interface graphique et de gérer les interactions avec l'utilisateur. En plus d'offrir une infrastructure assez rigide, *Apple* a beaucoup travaillé en matière de recommandations graphiques pour ses applications. Les documents mis à la disposition des développeurs sont assez complets et permettent de les guider vers les pratiques recommandées. La documentation contient aussi un certain nombre d'exemples clés en main afin d'accomplir des tâches types. Très tôt, la compagnie a mis à la disposition des programmeurs un kit de développement permettant de créer des logiciels pour l'*iOS*. De plus, pour éviter une hétérogénéité des langages de programmation, *Apple* a décidé que son propre langage *Objective-C* serait le seul autorisé pour le développement de logiciels sur sa plateforme. Enfin, comme la seule manière de fournir des applications pour cette plateforme est de passer par l'*App Store*, la compagnie se garde le droit de vérifier que les applications respectent ses standards et d'approuver ou non leur publication. Cette politique contraignante chère à *Apple* lui permet d'avoir un contrôle absolu de sa plateforme et ainsi d'offrir une expérience plus homogène à l'utilisateur.

Au premier septembre 2010, l'*App Store* comptait au moins 250 000 applications et a enregistré plus de 6,5 milliards de téléchargements. La notoriété de la plateforme n'est plus à faire et *Apple* a toujours été un leader en matière de technologie. L'*iPhone* est devenu en l'espace de quelques années une icône et son adoption massive par le grand public lui a permis de se tailler une grande place dans le marché de la téléphonie mobile. La sortie récente de l'*iPad* qui fonctionne avec le même système d'exploitation que l'*iPhone*, offre également de nouvelles opportunités sur le marché des tablettes tactiles. Cependant, les récentes critiques envers la fiabilité de la dernière (quatrième) génération de *iPhones* ainsi que la sensible baisse des parts de marché au profit de la concurrence (14 % en 2010 par rapport à 18 % en 2009) viennent quelque peu ternir le tableau. Malgré le succès de l'*iPhone*, il existe encore un certain nombre de limitations à l'extension de la plateforme. Tout d'abord, celle-ci a été principalement conçue pour le grand public et non pour les

entreprises, ce qui explique l'offre limitée au niveau des interactions avec les infrastructures des compagnies. Le fait que la plateforme soit limitée aux produits *Apple*, la nature propriétaire et fermée de la plateforme ainsi que le prix relativement élevé de ses appareils face à la concurrence sont aussi des facteurs limitants. De plus, le fait que la plateforme de développement ne soit disponible que pour *Mac OS X*, limite grandement les possibilités de développement pour les programmeurs. En effet, ceux-ci doivent se munir d'un Mac pour créer des logiciels pour l'*iOS*. Or, le prix élevé de l'appareil de même que sa faible représentation sur le marché (5 % de Mac par rapport à 91% de PC[3]) ajoutent une difficulté supplémentaire pour les personnes souhaitant développer des applications pour cette plateforme. Enfin, le choix d'*Objective-C* comme langage de programmation unique pour *iOS* réduit le nombre de développeurs potentiels, car ces derniers doivent apprendre un nouveau langage avant de produire des applications pour cette plateforme. Globalement, on peut dire que la politique très restrictive d'*Apple* est responsable de la faible présence de ses produits sur le marché. Mais, d'un autre côté, cette approche permet à l'entreprise de se concentrer sur certaines niches et d'offrir des produits plus aboutis.

1.2.2 BlackBerry OS

Plateforme de prédilection des gens d'affaires depuis son lancement en 1999, *BlackBerry* a su s'imposer dans le monde de l'entreprise et représente la deuxième plateforme la plus importante en nombre d'utilisateurs. La société canadienne *Research In Motion* (*RIM*) est le développeur à la fois du système d'exploitation *Blackberry OS* et des téléphones intelligents *BlackBerry*. La nature fermée et propriétaire de la plateforme *RIM* ne permet pas de connaître son organisation interne, sinon qu'elle repose sur les technologies *Java*.

Récemment, *RIM* a ouvert une plateforme de distribution numérique, baptisée *AppWorld,* pour les applications sur ses téléphones, imitant ainsi le modèle mis en place par *Apple*. Cette stratégie a eu pour effet d'augmenter la visibilité des logiciels conçus pour les *BlackBerry*. Pour les développeurs d'applications, *BlackBerry* offre deux approches, l'une basée sur la plateforme *Java* et l'autre basée sur les technologies web. Cependant, il est

3 http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8

également possible de faire appel à une approche hybride permettant d'utiliser des technologies web tout en ayant accès aux fonctionnalités propres à l'appareil grâce à l'utilisation de *widgets*. Cette approche hybride permet de profiter des avantages du développement d'applications natives et web tout en réduisant le niveau de compétence nécessaire pour créer de nouvelles applications. Elle permet également d'augmenter les capacités d'une application purement web ou de faciliter sa migration. Pour chacune des approches, une documentation importante est disponible. Des exemples de code sont également fournis afin d'illustrer la façon de réaliser certaines opérations.

Les *BlackBerry* étant traditionnellement destinés à une clientèle d'entreprise, leur interface était reconnue comme étant quelque peu spartiate, et se concentrait sur la tâche à accomplir plutôt que sur l'aspect graphique. De plus, le manque de spécifications claires sur l'apparence des logiciels avait conduit à une intégration plus ou moins erratique des différentes applications. Avec l'ouverture récente à un plus large public et la mise en place du *AppWorld*, *RIM* tente peu à peu d'améliorer l'apparence de ses applications. Afin de garantir une certaine homogénéité de l'interface, des guides de style sont maintenant disponibles pour orienter les développeurs sur les bonnes pratiques en matière de design. On y précise, entre autres, les couleurs à utiliser, l'organisation des différents éléments de l'interface, la taille et le type des polices de caractères ainsi que les dimensions et les thèmes des différentes icônes. Une petite section est consacrée à l'accessibilité, mais cette dernière reste assez floue sur les méthodes à mettre en place. Il s'agit plutôt de conseils que de réelles directives. Un guide d'optimisation des contenus web pour le navigateur des *BlackBerry* est également disponible afin de tenir compte de ses limitations et ainsi offrir une bonne expérience utilisateur.

L'environnement de développement offert par *RIM* se compose de plusieurs éléments. Il y a d'abord les outils graphiques constitués d'un studio de thèmes et du *Plazmic Content Developer's Kit* et ensuite des produits d'aide à la création de logiciels. Le studio de thèmes permet de créer facilement des éléments de personnalisation de l'interface du *BlackBerry*. Le *Plazmic Content Developer's Kit,* quant à lui, est plutôt réservé à la création de nouveaux contenus optimisés pour la plateforme, que ceux-ci soient web ou natifs. Les

autres produits sont des SDK permettant de programmer et de tester les logiciels grâce à des simulateurs et des outils de débogage. Des outils dédiés à la création de solutions complètes pour les entreprises qui s'intègrent avec les téléphones et les serveurs de *RIM* sont aussi offerts aux développeurs. Enfin, il est possible d'utiliser des greffons (« plugins ») afin d'intégrer les API *BlackBerry* à des environnements de développement comme *Eclipse* ou *Visual Studio*.

Les téléphones intelligents de la firme canadienne bénéficient de l'appui de nombreux professionnels grâce à leur offre substantielle de solutions dédiées aux entreprises. De plus, la robustesse de ses services de communications et les technologies innovantes comme le *Push*, permettant d'avoir accès directement à ses messages sans avoir à synchroniser l'appareil avec le serveur de messagerie, sont des atouts indéniables. Mais la plateforme doit encore être améliorée si elle veut gagner l'adhésion d'un plus large public. En effet, de nombreux retards en matière du support des standards web sont encore présents dans un certain nombre d'appareils. Bien que ces problèmes aient été corrigés avec la version 6 de son OS, il reste encore un grand nombre de personnes disposant d'appareils plus anciens pour lesquels la migration vers le nouveau système d'exploitation n'est pas possible. *BlackBerry* est également un casse-tête pour les développeurs avec plusieurs modèles de téléphones différents, chacun avec ses particularités comme la taille de l'écran ou le mode d'interaction. L'existence d'une multitude de versions du système d'exploitation, dont la mise à jour est laissée à la discrétion des opérateurs de téléphonie mobile, ajoute une charge de travail importante pour les développeurs qui doivent s'assurer de la compatibilité de leur application sur tous les modèles. Il faut ajouter à cela la relative jeunesse de la plateforme de développement. En effet, en visant un public professionnel assez restreint, seuls *RIM* ou des partenaires choisis étaient chargés du développement des applications pour sa plateforme. Cette situation a conduit à une certaine hétérogénéité au niveau de l'apparence des applications disponibles sur *BlackBerry*, réduisant ainsi la qualité de l'expérience utilisateur.

1.3 Principes de conception propres aux plateformes mobiles

L'ensemble des principes de conception présentés ici sont issus des documents techniques de *RIM* [17] et *Apple* [18] où l'on retrouve des recommandations spécifiques à chaque plateforme. De ces recommandations, nous avons alors extrait les principes qui pouvait être généralisés à toutes les plateformes mobiles et nous les présentons ci-dessous. Pour la partie portant sur la publicité, nous avons utilisé les comptes rendus de l'atelier "Comment rentabiliser son site ou son application mobile : l'exemple des médias" de l'Association Français du Multimédia Mobile (AFMM) [29]. Cette lecture nous a permis de découvrir des idées intéressantes afin d'utiliser plus judicieusement la publicité, ce dont nous faisons part dans la section correspondante.

Les limites du mobile

Il faut toujours garder à l'esprit les contraintes imposées par les plateformes mobiles et ne pas tomber dans le piège consistant à porter des applications conçues pour les ordinateurs sans les adapter. Étant donné la taille de l'écran, il est important de réduire au strict minimum les éléments à présenter à l'utilisateur. Ainsi on ne surchargera pas inutilement l'interface en forçant l'utilisateur à passer par de nombreuses pages-écrans avant d'obtenir ce qu'il désire. Il faut aussi réaliser que la saisie d'informations sur ce type de produit reste fastidieuse malgré les améliorations apportées au fil du temps. Il faut donc éviter autant que possible de demander à l'utilisateur de saisir des informations, surtout s'il est possible de les obtenir autrement. À cet effet, il peut être intéressant d'offrir des raccourcis pour effectuer certaines actions plus rapidement, surtout pour les utilisateurs réguliers d'une application.

Il faut aussi réaliser que les tâches accomplies sur les appareils mobiles sont susceptibles d'être interrompues par des facteurs externes. L'application doit donc permettre à l'utilisateur de suspendre rapidement sa tâche courante et d'y revenir plus tard sans avoir à tout reprendre depuis le début. Enfin, de nouveaux paradigmes d'interaction ont vu le jour avec l'apparition des écrans tactiles, mais qui apportent aussi de nouvelles contraintes,

comme la taille des éléments graphiques à laquelle il faut porter attention afin de faciliter leur manipulation.

Les métaphores

L'utilisation de métaphores permet à l'utilisateur de comprendre rapidement le fonctionnement d'une application en l'associant à un objet ou une fonction que l'on retrouve dans le monde réel. Il faut toutefois savoir les utiliser judicieusement. Entre autres, il faut veiller à respecter les modes d'interaction existants de l'appareil pour ne pas désorienter l'utilisateur. Lors de la mise en place d'une nouvelle fonctionnalité ou d'une nouvelle application, il faut donc découvrir à quelle action de la vie courante elle peut s'apparenter, afin que l'utilisateur puisse réutiliser des notions qu'il maîtrise déjà et ainsi accélérer la prise en main de l'application. Il ne faut toutefois pas abuser des métaphores car on risque alors de produire un effet contraire, par exemple s'il y a une mauvaise adéquation entre l'application et les images mentales de l'utilisateur. Il est donc recommandé d'effectuer des tests utilisateurs pour s'assurer qu'une métaphore est facilement compréhensible et assimilable.

La manipulation directe

La manipulation directe permet d'interagir en temps réel et sans intermédiaire avec les éléments graphiques de l'interface. Ceci est particulièrement important pour les appareils munis d'un écran tactile. En effet, l'implication de l'utilisateur est bien plus grande lorsqu'il manipule directement l'interface. De plus, l'utilisateur comprend mieux le résultat de ses actions s'il interagit sans intermédiaire avec son appareil. Pour que la manipulation directe soit vraiment efficace, il faut que les objets avec lesquels l'utilisateur interagit soient toujours visibles à l'écran. L'effet peut également être renforcé en faisant appel à une rétroaction immédiate. Ainsi, l'utilisateur est en mesure de voir à chaque instant l'impact de ses actions sur le système.

La reconnaissance de l'information

La mémoire de l'être humain ayant une capacité limitée, il est souvent utile de demander à l'utilisateur de reconnaître une information plutôt que de lui demander de la fournir telle quelle. Par exemple, une liste de choix peut être proposée à l'utilisateur afin qu'il sélectionne l'information désirée. Cette approche limite également les probabilités d'erreur.

L'utilisateur est roi

Rien n'est plus frustrant pour un utilisateur que de constater qu'une application prend des décisions à sa place et accomplit des tâches sans qu'il puisse les interrompre. C'est pour cette raison qu'il faut toujours donner le plein contrôle à l'utilisateur en lui permettant, notamment, d'interrompre les actions qu'il a entreprises. De plus, les interactions avec l'appareil doivent demeurer les plus simples et les plus directes possibles. Il est également recommandé de favoriser des formes de contrôle et des comportements qui sont déjà familiers à l'utilisateur.

L'homogénéité

Une application peut avoir été créée pour une plateforme particulière ou pour une gamme d'appareils. Dans le premier cas, il est très important de se conformer aux recommandations émises par les fabricants de l'appareil afin de s'assurer d'une bonne intégration de l'application et ainsi éviter qu'elle ne soit refusée par le comité d'évaluation. Il faut aussi que l'application soit semblable en apparence à celles déjà présentes sur l'appareil. Lorsque la documentation fournie par le fabricant n'est pas suffisamment explicite sur les pratiques à suivre, il est conseillé de se référer à des applications natives afin de reproduire leur style et conserver les mêmes modes d'interaction.

Lorsqu'une application est conçue pour une gamme d'appareils, elle doit tout de même répondre au style des différentes plateformes. Une solution consiste à utiliser une approche vue-contrôleur afin de séparer la présentation de l'interface des actions à effectuer. Il est alors possible d'avoir des applications presque identiques, mais dont le style

des boutons ou celui des champs de texte obéira dans une certaine mesure aux recommandations propres à chaque plateforme.

L'intuitivité

L'utilisateur n'a souvent ni le temps ni la volonté d'étudier le fonctionnement d'une application et encore moins de lire une longue documentation. Il est donc important que l'application soit facile à utiliser et demande peu d'efforts de la part de l'utilisateur. Les fonctions principales de l'application doivent donc être mises en évidence. Cela peut être accompli en réduisant le nombre de boutons de contrôle et en leur donnant des noms faciles à interpréter. Il faut aussi éviter de fournir des textes trop longs dans la documentation. D'ailleurs, il est recommandé d'offrir des alternatives visuelles et interactives telles qu'une courte vidéo ou un tutoriel à la place d'une longue explication. L'exemple fourni par l'application *Stopwatch* (cf. Figure 7) qui fait partie des fonctionnalités d'horloge du *iPhone* illustre parfaitement ces principes. L'intitulé des boutons ainsi que le code couleur rendent les fonctions qui leur sont associées transparentes aux yeux de l'utilisateur et cela, sans avoir à fournir d'information supplémentaire.

Figure 7: Application *Stopwatch* sur *iPhone*

Les tâches primaires

Le développeur doit toujours se demander quelle est la finalité de l'application qu'il crée. Ce questionnement permet d'identifier rapidement les fonctionnalités fondamentales, ou tâches primaires, et de définir la stratégie à mettre en place pour y répondre.

L'application devra ainsi permettre à l'utilisateur d'exécuter facilement et rapidement les tâches primaires sans être dérangé par d'autres éléments de l'interface, comme de la publicité. De même, lorsqu'on décide de modifier la façon d'exécuter une tâche primaire, il est important de procéder à des tests préliminaires afin de s'assurer que l'utilisateur n'est pas désorienté par la nouvelle façon de faire.

La publicité et l'image de marque

Il est essentiel pour une entreprise d'afficher son image de marque afin que les utilisateurs puissent facilement l'identifier. On peut pour cela faire appel à un logo ainsi qu'à des couleurs et une typographie distinctives. Les politiques en matière de communication vont permettre d'obtenir une certaine homogénéité au niveau de l'image de marque que l'on souhaite véhiculer afin de se distinguer de la concurrence. Il faut cependant éviter l'utilisation à outrance du logo ou de faire appel à des designs trop complexes qui peuvent alourdir l'interface et parfois conduire à un rejet de la part de l'utilisateur. Pour les mêmes raisons, il est également important d'éviter les erreurs de transposition. En effet, les politiques mises en place par les compagnies ont souvent été conçues pour un support papier et sont trop souvent transposées telles quelles sur un support numérique. Or, les contraintes de lisibilité ainsi que les métaphores et les codes de couleurs n'étant pas les mêmes sur ces deux supports, on risque de produire un effet contraire à celui désiré. Par exemple, la couleur rouge sur un support papier peut être utilisée pour attirer l'attention de l'utilisateur ou comme effet de style. Mais cette couleur convient rarement sur un support numérique, car elle est généralement utilisée pour signaler un problème.

La publicité est un autre élément crucial pour les entreprises. En effet, comme il s'agit d'une source de revenus potentiellement importante, l'entreprise souhaiterait en afficher le plus souvent possible afin d'augmenter les retombées financières. Là encore, il faut veiller à ne pas tomber dans les extrêmes et déranger l'utilisateur avec un ratio publicité/contenu excessif. Comme on l'a vu plus haut, une des limitations des appareils mobiles est la taille de leur écran. Il faut donc trouver un compromis adéquat entre contenu et publicité. Pour ce faire, deux stratégies peuvent être utilisées. La première consiste à

utiliser de la publicité contextualisée afin que celle-ci soit moins intrusive aux yeux de l'utilisateur. En d'autres termes, il s'agit de fournir des contenus publicitaires qui prennent en compte les actions des utilisateurs comme les recherches effectuées ou la rubrique dans laquelle ils se trouvent et non pas seulement des bannières aléatoires qui s'affichent en bas d'écran. Cette stratégie peut être assez coûteuse toutefois, car elle demande un effort de développement important. La deuxième solution est plus simple et consiste à circonscrire la publicité à quelques pages seulement. Bien entendu, il faut pouvoir attirer les visiteurs sur les pages en question. Un modèle possible est connu sous l'appellation *Freemium*[4], contraction des termes Free et Premium qui signifient gratuit et haut de gamme en anglais. L'idée est ici d'offrir un service gratuit avec de la publicité afin d'attirer les utilisateurs et ensuite leur proposer un service plus complet, souvent sans publicité et moyennant une certaine somme. Ce type de modèle semble approprié pour les plateformes mobiles puisqu'il a été mis en place par Apple pour l'App Store, son service de distribution d'applications (qui a ensuite été imité par d'autres plateformes mobiles). Il faut toutefois que l'entreprise dispose d'une politique assez rigoureuse quant au type de contenu qui peut être distribué.

1.4 Conclusion

Ce chapitre nous a permis de nous familiariser avec les plateformes mobiles *iOS* et *BlackBerry*. Bien que ce chapitre ne soit pas directement lié au système que nous avons développé, il permet néanmoins de mieux saisir les caractéristiques propres à chaque plateforme et ainsi appréhender les recommandations ergonomiques qui seront présentées dans les chapitres suivants.

4 http://www.afmm.fr/img/HP/DossierAFMM_%20Rentabiliser%20son%20site%20ou%20appli%20mobile.pdf

CHAPITRE 2 : Objectifs de recherche

Le but de ce chapitre est de découvrir l'environnement dans lequel le travail a été réalisé, de connaître les objectifs de cette recherche et d'expliquer comment les défis rencontrés ont été relevés. Nous présentons d'abord le contexte de la recherche. On y découvre en particulier les défis auxquelles l'équipe responsable des services mobiles doit faire face. Puis, les grandes ligne du système développé sont présentés, suivi des différentes étapes de réalisation. Enfin, on présente les normes qui ont été suivies lors du recueil des informations afin de faciliter les ajouts futurs.

2.1 Contexte et objectifs

Le travail de recherche a été effectué au sein d'*Air Canada*. Cette compagnie aérienne, fondée dans la seconde moitié des années '30 sous le nom de *Trans-Canada Airlines*, a connu un certain nombre de transformations et de fusions avant de devenir l'entreprise qu'elle est aujourd'hui. Cette compagnie bénéficie d'une longue tradition en matière d'innovation[5] ce qui lui a permis d'être aujourd'hui la principale compagnie aérienne au Canada et la quinzième à l'échelle mondiale. Avec l'évolution des technologies de l'information et des télécommunications, la compagnie a dû adapter son offre afin de proposer des services toujours plus innovants à sa clientèle. De cette volonté sont nés les services mobiles qui permettent aux clients d'accéder à des informations et d'effectuer des tâches en rapport avec leurs réservations depuis leur téléphone intelligent. *Air Canada* a été l'une des premières compagnies aériennes à se lancer dans l'aventure mobile. Évoluant dans un nouveau monde et ayant un manque d'expertise dans le domaine, la compagnie a fait appel à un géant de l'informatique, l'entreprise *IBM*, afin de sous-traiter la mise en place de cette offre de services. Depuis 2005, une équipe de six personnes est affectée aux différents aspects du service mobile chez *Air Canada*. Mais le domaine des appareils mobiles est relativement nouveau et évolue rapidement. En effet, la phase de développement de ces appareils est relativement courte à cause de leur cycle de vie limité, de l'évolution rapide de

5 http://www.aircanada.com/fr/about/media/facts/innovations.html

la technologie et de la présence d'un marché très compétitif (Kallio & Kekalainen, 2004). Il n'est donc pas facile pour l'équipe de se tenir au courant des dernières avancées dans le domaine, de s'assurer que le travail fourni par *IBM* est conforme aux recommandations émises par les fabricants de plateformes mobiles et d'anticiper l'évolution du marché afin de proposer des services mobiles bien adaptés. C'est dans ce contexte, qui est loin d'être un cas unique dans le monde de l'industrie, que ce travail de recherche a été réalisé. Notre but était de répondre à une problématique importante à laquelle l'équipe mobile devait faire face, soit l'évaluation adéquate du travail effectué par *IBM* avant la mise en production. Pour cela, nous avons développé un outil permettant de guider cette évaluation lors de la période de conception, c'est-à-dire lorsque le sous-traitant soumet à l'équipe mobile d'*Air Canada* des maquettes (mock-ups) pour fins de validation. Un objectif secondaire était de fournir à l'équipe mobile un outil lui permettant de comparer les recommandations émises pour différentes plateformes afin d'en extraire des tendances générales et ainsi mieux orienter les choix de l'entreprise lors de nouveaux projets.

2.2 Proposition d'une solution

Un des outils largement utilisés pour vérifier la conformité d'un produit est la liste de vérification. Celle-ci est largement répandue dans le domaine de l'aviation. Il était donc naturel, en travaillant chez *Air Canada*, d'utiliser ce type d'approche afin d'évaluer le niveau de conformité des maquettes fournies par rapport aux recommandations des fabricants de plateformes mobiles. De telles listes permettent de pallier aux capacités limitées de mémorisation de l'humain, tout en permettant de procéder de façon méthodique. La liste de vérification correspond à une liste de points à vérifier, avec des cases à cocher, permettant de passer en revue tous les points de façon systématique et de s'assurer de leur respect d'une façon concise et rapide. Nous avons donc utilisé une liste de vérification développée par Michael Q. Patton dans son livre « How to use qualitative methods in evaluation » (Patton, 1987). Cette liste de vérification, bien que simple d'un point de vue conceptuel, a cependant dû être adaptée afin de pallier à certaines de ses limitations et répondre aux spécificités des plateformes mobiles.

L'idée de base consiste à générer une liste de vérifications, où chacune contient un certain nombre de points à respecter, à partir de la documentation fournie par les fabricants de plateformes mobiles, afin de garantir la conformité d'une application. L'outil va ainsi permettre aux responsables de l'évaluation chez *Air Canada* de connaître le niveau de conformité du travail reçu grâce à un score basé sur des données concrètes. Afin d'ajouter davantage de flexibilité à ces listes de recommandations, nous avons opté pour une version numérique qui est constituée d'une interface facilement manipulable, permettant de sélectionner ce que l'utilisateur souhaite évaluer, ainsi que d'une base de données pour stocker l'ensemble des informations recueillies. L'utilisateur peut ainsi se concentrer sur certains points particuliers choisis de façon dynamique selon des critères bien définis. On remédie ainsi aux deux principales faiblesses des listes de vérification traditionnelles, soit leur longueur et leur manque de flexibilité.

Il est également possible d'affecter un niveau de priorité au différente recommandations et de les organiser en fonction de leur degré d'importance, permettant ainsi aux utilisateurs de s'en tenir à l'essentiel s'ils le désirent. Un autre avantage offert par le format numérique est la possibilité de voir les recommandations de plusieurs plateformes de façon concurrente. Ceci est évidemment beaucoup plus difficile à faire lorsqu'on travaille avec des listes sous format papier. Ainsi, la solution numérique est beaucoup plus facile à maintenir. Les différentes recommandations peuvent être facilement modifiées et il est même possible d'en ajouter de nouvelles ou d'en supprimer. La mise à jour s'en trouve facilitée lorsque de nouvelles versions d'une plateforme apparaissent ou que certaines fonctionnalités deviennent désuètes. Enfin, afin de favoriser un dialogue entre l'équipe des évaluateurs et celle des sous-traitants, nous avons privilégié une évaluation du niveau de conformité d'une maquette échelonnée sur trois niveaux, avec la possibilité d'ajouter des commentaires. On crée ainsi un niveau intermédiaire entre le respect total et le non-respect d’une recommandation qui facilite un tel dialogue. Il est également possible d'exclure des éléments du processus d'évaluation, lorsqu'ils ne sont pas pertinents, lorsqu'un besoin de personnalisation se traduit par un non respect volontaire d’une recommandation ou lorsqu'une vérification additionnelle est nécessaire. Le dernier cas survient lorsqu'un élément ne peut être vérifié directement sur la maquette.

La solution retenue présente encore quelques faiblesses. Tout d'abord, le travail consistant à compiler la documentation et en extraire les recommandations pour inclusion dans la base de données du système représente un travail d'envergure. Ensuite, les ressources allouées à ce projet n'ont pas permis de traduire intégralement la documentation, qui est surtout en anglais. L'interface du système ainsi que les points que l'on retrouve au sein des recommandations sont totalement en français, mais certaines informations supplémentaires sont encore en anglais. Une solution pour le support multilingue, composée de tables contenant l'ensemble des libellés de l'interface dans les langues voulues, a été imaginé mais n'a pu être mise en place à cause des délais.

Un moteur de recherche aurait également pu être ajouté au système afin de permettre le filtrage des données. En effet, la base de données contient des champs réservés pour des mots clés, mais le code qui aurait permis d'associer les mot clés saisis au clavier avec ceux de la base de données n'a pas été réalisé. Enfin, bien que le code source ait été commenté, aucune norme n'a véritablement été suivie pour ce faire ; il en est de même pour l'appellation des variables. Il faut comprendre que le travail réalisé dans le cadre de ce mémoire visait à produire un premier prototype et, de ce fait, certains composants ne sont sans doute pas à la hauteur des standards industriels.

2.3 Architecture de la solution proposée

2.3.1 Recueil des informations

2.3.1.1 *BlackBerry* et *iOS*

On assiste ces dernières années à un consensus entre les développeurs d'interfaces graphiques et les utilisateurs sur l'importance de l'utilisabilité des logiciels (Hartson, Andre, & Williges, 2001 ; Jokela, Iivari, Matero, & Karukka, 2003). Ceci s'est traduit, chez les grandes entreprises de plateformes mobiles, par la mise sur pied de listes de vérification pour s'assurer que le design des interfaces graphiques est conforme aux lignes directrices en matière d'utilisabilité (Scholtz, 2004). Grâce aux différentes recommandations fournies par chacun des fabricants de plateformes, il est possible de nourrir la base de données de notre

système. À cet effet, nous avons d'abord dû identifier ces recommandations pour les deux plateformes considérées, soit *BlackBerry* et *iOS.* Cette étape s'est révélée plutôt fastidieuse, car il a fallu parcourir une documentation assez abondante afin d'en extraire les points essentiels. De plus, la documentation s'adressant principalement à des développeurs, il a fallu adapter les recommandations au contexte de notre entreprise. Nous avons donc procédé à une première lecture de la documentation technique afin d'identifier les aspects ergonomiques qu'il est important de respecter au niveau de l'interface graphique. Ce premier travail a ensuite été soumis pour approbation aux responsables de l'équipe mobile de la compagnie. Nous nous sommes ainsi assurés que les points que nous avions identifiés étaient pertinents et répondaient à leurs attentes. Après cette validation, nous avons illustré chacun des points relevés avec des captures d'écran et avons ajouté un texte explicatif afin d'être sûrs qu'ils soient bien compris lors de l'étape d'évaluation.

2.3.1.2 Image de marque d'Air Canada

En parallèle à cet effort de documentation, nous avons également dû produire des recommandations portant sur l'image de marque de la compagnie. En effet, si l'on veut être en mesure de développer une application pour une entreprise, il est primordial de s'assurer que le logiciel livré reflète bien son image corporative. Nous avons donc consulté les responsables de la mercatique afin de mieux comprendre les chartes graphiques et les normes à suivre en matière d'image de marque de la compagnie. Un des principaux problèmes avec la charte graphique actuelle vient de ce qu'elle a été développée pour le support papier seulement et ne prend donc pas en considération certaines contraintes relatives à l'affichage sur un écran. Il n'y a, par exemple, aucune mention de la taille des différents éléments graphiques selon la résolution utilisée, ni de vérification en ce qui concerne la différence de contraste et de luminosité entre le fond et le texte selon les couleurs utilisées. Ces limites ont été mentionnées lors de nos rencontres et seront très probablement corrigées dans les versions futures de la charte graphique.

2.3.2 Organisation de l'information

2.3.2.1 Différences physique et modes d'interaction

Lors de la cueillette des informations nécessaires au bon fonctionnement du système, il n'a pas été facile d'extraire des recommandations à partir des documents techniques. En effet, la documentation été d'une part relativement longue et d'autre part s'adresser surtout à des développeurs. De plus, les recommandations obtenues étaient assez disparates puisqu'elles reposaient sur des appareils aux spécificités différentes. Nous avons donc développé une typologie permettant de regrouper les recommandations présentant certaines caractéristiques communes (cf. Tableau 2). Le travail rapporté dans le premier chapitre s'est d'ailleurs révélé grandement utile pour arriver à ce résultat.

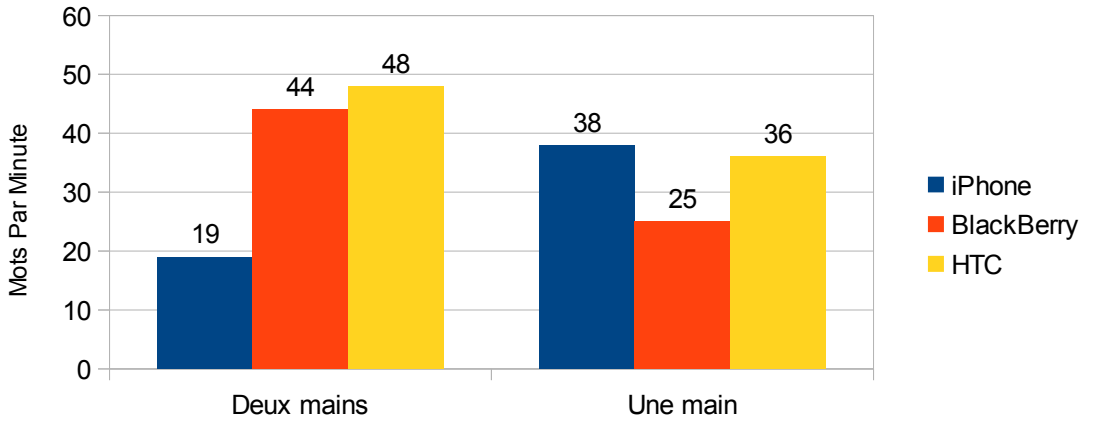
Type d'interaction	Écran tactile	Clavier physique et dispositif de pointage	Hybride
Exemples	 *iPhone*	 *BlackBerry Bold*	 *HTC Dream*
Spécificités	– Écran tactile multi-point – Utilisation de gestes prédéfinis – Métaphore : manipulation directe	– Clavier physique (simplifié ou étendu) – Dispositif de pointage (pavé tactile, boule de contrôle ou pavé directionnel) – Métaphore : clavier et souris	– Ensemble des points des deux autres catégories
Avantages	– Manipulation plus ludique – Plus grande surface d'affichage	– Saisie de données et navigation comparable aux PC – Saisie d'information plus performante – Raccourcis clavier	– Ensemble des points des deux autres catégories
Inconvénients	– Pas de retour haptique lors de la saisie d'informations – Trace de doigt sur l'écran	– Taille d'écran plus petite – Plus encombrant	– Généralement plus encombrant et plus lourd que les autre modèles.

Tableau 2: Différences physiques entre les modèles d'interaction des téléphones intelligents

Comme on peut le constater, les téléphones qu'on retrouve sur le marché ont des modèles d'interaction physique qui diffèrent grandement. On retrouve ainsi plusieurs grandes catégories d'appareils. Il y a tout d'abord ceux qui sont équipés d'un clavier physique simplifié ou étendu ainsi que d'un dispositif de pointage qui peut être un pavé tactile, une boule de contrôle ou un pavé directionnel. Cette catégorie est la plus ancienne et aussi celle qui comporte le plus grand nombre d'appareils. Elle a l'avantage de reprendre les

dispositifs de navigation et de saisie des ordinateurs classiques avec un clavier et une souris.

On retrouve ensuite une nouvelle catégorie qui est en constante évolution, soit les appareils munis d'un écran tactile. Ces derniers ont été rapidement popularisés avec l'introduction de technologies comme le multipoints qui permet d'utiliser plus d'un doigt simultanément sur un même écran. Cette nouvelle catégorie est très séduisante car elle permet de manipuler directement les objets de l'interface avec les doigts. Il reste toutefois quelques faiblesses, en particulier lors de la saisie des données à deux mains si on compare avec un clavier physique, même de taille réduite. En effet, les claviers physiques offrent un retour haptique sur l'état des boutons, pressés ou non, ainsi qu'une position spatiale. Cela permet, par exemple, de s'affranchir du besoin de regarder là où l'on appuie et ainsi d'augmenter la vitesse de frappe (cf. Figure 8).



Source : http://www.wirelessinfo.com

Figure 8: Vitesse de saisie des différents modèle de téléphone

Les concepteurs ont pris en compte ces limitations ainsi que le besoin de concilier productivité et interaction innovante. Ils ont donc créé une catégorie hybride alliant les avantages des deux catégories précédentes. On retrouve ainsi des modèles de téléphone possédant à la fois un écran tactile, un dispositif de pointage et un clavier physique. Ce dernier est d'ailleurs souvent caché derrière l'écran et on peut facilement le faire coulisser au besoin. Introduite par *Android*, le système d'exploitation libre de *Google,* lors du

lancement de son téléphone de développement appelé *HTC Dream*, cette idée a depuis été reprise par *RIM* avec son dernier appareil le *BlackBerry Torch* (cf. Figure 9).

Figure 9: BlackBerry Torch

2.3.2.2 Types de recommandations

Les distinctions qui apparaissent au niveau du mode d'interaction avec l'appareil vont prendre tout leur sens lors de la phase d'extraction des recommandations. En effet, la lecture de la documentation technique relative à chacune des plateformes a permis d'identifier un ensemble de points communs mais aussi des spécificités dans l'organisation des différents éléments de l'interface graphique. À l'issue de l'étude des recommandations émises par les développeurs d'*iOS* et *BlackBerry* sur les différents composantes de l'interface graphique, un certain nombre de points communs sont apparus. De plus, certaines recommandations associées à l'une des plateformes pouvaient facilement être étendues à l'autre plateforme. Au final, nous avons créé une typologie se divisant en neuf types de recommandations touchant à différents composants de l'interface (cf. Tableau 3).

Types de recommandations	Description
Générique	Recommandations s'appliquant à l'ensemble des plateformes.
Exclusif	Recommandations spécifiques à une plateforme et sans équivalent pour les autres plateformes
Navigation	Recommandations à respecter lors de la navigation.
Saisie de données	Recommandations à respecter lors de la saisie d'informations.
Barres	Recommandations à respecter lors de l'utilisation des différentes barres d'outils .
Icônes	Recommandations à respecter lors de l'utilisation des icônes.
Boîte de dialogue	Recommandations à respecter lors de l'affichage d'une boîte de dialogue.
Liste/Tableau	Recommandations à respecter lors de l'utilisation des tableaux ou des listes d'informations.
Sons	Recommandations à respecter lors de l'utilisation de stimuli sonores.

Tableau 3: Types de recommandations

2.3.2.3 Image de marque

Un autre ensemble de recommandations porte sur l'image de marque de la compagnie. En effet, cette image est capitale pour une entreprise, car elle permet à ses clients de l'identifier facilement. Elle se doit donc d'être originale et de se distinguer de la concurrence afin d'éviter toute confusion. Dans le cas d'*Air Canada,* il existe trois grandes divisions au sein de la compagnie, chacune étant spécialisée dans certains types de service. On y retrouve *Air Canada Cargo* qui englobe l'ensemble des services relatifs au transport de marchandises, *Air Canada Vacances* qui est chargée de promouvoir les voyages clés en main et enfin *Air Canada* comme tel qui regroupe l'ensemble des services relatifs au transport de passagers. Pour chacune de ces divisions, une charte graphique spécifique a été conçue. Toutefois, pour *Air Canada Cargo* et *Air Canada Vacances*, il n'a pas été possible de disposer de leur charte graphique car ces divisions ont récemment connu de profonds changements au niveau de leur image de marque. Le développement de la typologie a donc

été réalisée avec la charte graphique fournie par la division principale d'*Air Canada*. Au final, cette dernière se compose de trois volets : les logos, les couleurs et la typographie.

2.4 Règles d'organisation de l'information

Compte tenu de la masse de données que nous avons dû parcourir, il était nécessaire de mettre en place une stratégie d'organisation de l'information afin de pouvoir s'y retrouver par la suite. L'idée étant, d'une part, de garantir la pérennité des informations recueillies en s'assurant de leur adéquation avec les besoins de l'équipe mobile et, d'autre part, d'anticiper les besoins futur en standardisant l'ajout d'informations dans le système.

2.4.1 Recueil des données

Lors de la cueillette des données et de leur arrangement, nous avons appliqué certains principes qui pourront être utilisés à nouveau lors de l'ajout de nouvelles recommandations ou de nouvelles plateformes. Dans les deux cas, la procédure sera simplifiée étant donné que l'on dispose déjà d'une typologie permettant d'organiser l'information. Nous conseillons donc de suivre la démarche suivante :

- Lire la documentation et extraire les informations pertinentes en les illustrant par des images lorsque c'est possible.
- Se référer aux responsables de l'évaluation de l'équipe mobile, afin de s'assurer de la pertinence et de la clarté des recommandations retenues.
- Rédiger les recommandations de manière explicite en mettant l'accent sur les points importants et y associer un texte explicatif au besoin.
- Toujours utiliser l'interface d'ajout du système pour saisir les recommandations afin de bénéficier de la nomenclature déjà mise en place.

Cette méthodologie a été adoptée en s'assurant d'impliquer à chaque étape les membres de l'équipe mobile d'*Air Canada* ainsi que les directeurs de recherche. Ce travail a ainsi permis de standardiser la procédure pour l'ajout de recommandations dans le système et ainsi offrir des recommandations de qualité comparable.

2.4.2 Niveaux de priorité

Comme il n'était pas possible de tester toutes les recommandations que l'on retrouve dans la documentation de base pour s'assurer de leur pertinence, nous présumons que les préceptes mis de l'avant par les fabricants de plateformes proviennent d'études sérieuses de leur part et peuvent donc être acceptées telles quelles. Ceci permet d'ailleurs de standardiser la réalisation de certaines actions sur les plateformes et de fournir une charte graphique distinctive pour l'entreprise.

Par ailleurs, puisque toutes les recommandations ne sont pas de même importance, nous leur associons un niveau de priorité. Pour ce faire, nous avons discuté avec les membres de l'équipe mobile d'*Air Canada* des aspects qu'ils souhaitaient mettre de l'avant. Ensuite, nous avons utilisé les critères et recommandations ergonomiques unifiés d'un professeur à l'École Polytechnique, M. Walter de Abreu Cybis, présentés dans son cours Interfaces humains-ordinateurs (IND6402), afin de prioriser les recommandations des constructeurs en fonction des désirs de l'équipe mobile d'*Air Canada*. Cette unification regroupe les « usability heuristics » (Jakob Nielsen, 1994), les « règles d'or » (Ben Schneiderman, 1994), les « principes de dialogue » (ISO 9241:10, 1996) et certains des « critères ergonomiques » (Scapin & Bastien, 1993).

Par souci de simplicité, nous avons utilisé seulement trois niveaux de priorité, numérotés de 1 à 3, qui représentent respectivement une priorité faible, moyenne ou élevée. Puisque tout schéma de priorisation possède une part de subjectivité, les utilisateurs autorisés d'*Air Canada* peuvent y apporter des modifications s'ils le désirent.

2.5 Conclusion

Cette phase préliminaire a été déterminante dans la réalisation du projet en permettant d'identifier les points communs, mais aussi les spécificités, des différentes plateformes mobiles de même que les caractéristiques d'une image de marque. Ce travail a été réalisé en suivant une méthodologie précise, qui assurera une certaine rigueur lors d'ajouts de recommandations dans le futur. Ce travail a aussi permis d'organiser

l'information afin de répondre aux défis auxquels sont confrontés les membres de l'équipe mobile d'*Air Canada*. Cette organisation de l'information est capitale puisque l'implémentation technique, décrite dans le chapitre suivant, repose sur celle-ci.

CHAPITRE 3 : Implémentation technique

Dans ce chapitre, nous présentons les choix technologiques qui ont été retenus pour l'implémentation du prototype. Nous présentons ensuite le schéma utilisé pour la base de données, puis les technologies et la méthodologie qui ont permis de réaliser l'interface du système avec laquelle l'utilisateur interagit. Nous terminons avec l'interface d'administration du système en présentant l'ensemble des options disponibles, telles que l'ajout d'un nouvel utilisateur, la modification du mot de passe ou encore la modification des tables de la base de données.

3.1 Choix technologiques

Afin d'éviter tout problème de compatibilité, nous avons utilisé des technologies web. Ce choix a été motivé d'une part par la standardisation et la portabilité de ce type de solutions qui permet de s'affranchir d'une plateforme de développement particulière et ainsi assurer un accès facile à l'équipe d'évaluation. Pour fournir ce type de solutions et garder un maximum de contrôle sur les différents éléments du système, nous avons fait appel à des outils basés sur des technologies libres ayant fait leurs preuves. L' infrastructure AMP (*Apache, MySQL, PHP*) a ainsi été retenue. Apache est le serveur web chargé de répondre aux requêtes venant des clients, c'est-à-dire les navigateurs des usagers. *MySQL* est le système de gestion de base de données (SGBD) qui permet de stocker et d'organiser les données. Enfin, le langage de scripts *PHP* permet de générer des pages web dynamiques et de communiquer avec le serveur *MySQL*. Une telle structure permet d'une part une gestion efficace des données et d'autre part un affichage dynamique de l'information souhaitée.

Avant le déploiement de la solution sur une plus grande échelle, un environnement de test a été développé sur un ordinateur portable. Ce dernier repose sur le système d'exploitation *Archlinux*[6], une distribution libre et gratuite basée sur la plateforme GNU/Linux. La particularité de cette distribution est de prôner un modèle KISS (Keep It Simple Stupid), qui favorise la simplicité et évite ainsi d'encombrer le système de

6 www.archlinux.org

fonctionnalités inutiles. Un autre avantage d'*Archlinux* est de fonctionner sur un modèle de distribution tournante (« rolling release ») ce qui permet de bénéficier des derniers logiciels disponibles et des dernières fonctionnalités issues de la communauté, tout en profitant des avancées en matière de correction d'erreurs.

Afin d'enrichir l'infrastructure AMP, nous avons ajouté la suite de logiciels XAMPP[7]. Il s'agit de logiciels libres (*X, Apache, MySQL, Perl, PHP*) offrant une bonne souplesse d'utilisation et qui sont réputés pour leur installation simple et rapide. Cet environnement évite d'entrer dans les détails techniques de la configuration, ce qui peut parfois être fastidieux. Ainsi, le développeur n'a pas à se soucier de la gestion des droits d'accès, étant donné que cette solution est destinée à fonctionner localement et peut être démarrée à la demande.

Toute cette infrastructure a ainsi permis de mettre en place le schéma de la base de données et de générer les pages de façon dynamique en fonction des filtres utilisés par les utilisateurs.

3.2 Schéma de la base de données

La façon de structurer l'information est un élément déterminant dans tout projet informatique. Il faut donc concevoir un bon modèle dès le départ afin de ne pas être confronté à d'éventuels problèmes par la suite. En effet, des millions de dollars sont engloutis chaque année par les entreprises afin de revoir la structure de certaines bases de données dû à des problèmes d'optimisation, voire de mauvaise conception. Il était donc important de faire appel à de bonnes pratiques. Ainsi, avant de passer à l'implémentation physique, nous avons commencé par définir la structure désirée. Pour cela, nous avons identifié les éléments de base ainsi que les relations entre eux, ce que l'on appelle le schéma relationnel. Cette structure est illustrée à la Figure 10 grâce au logiciel libre de création de diagrammes *Dia*[8].

7 www.apachefriends.org/fr/xampp.html

8 http://live.gnome.org/Dia

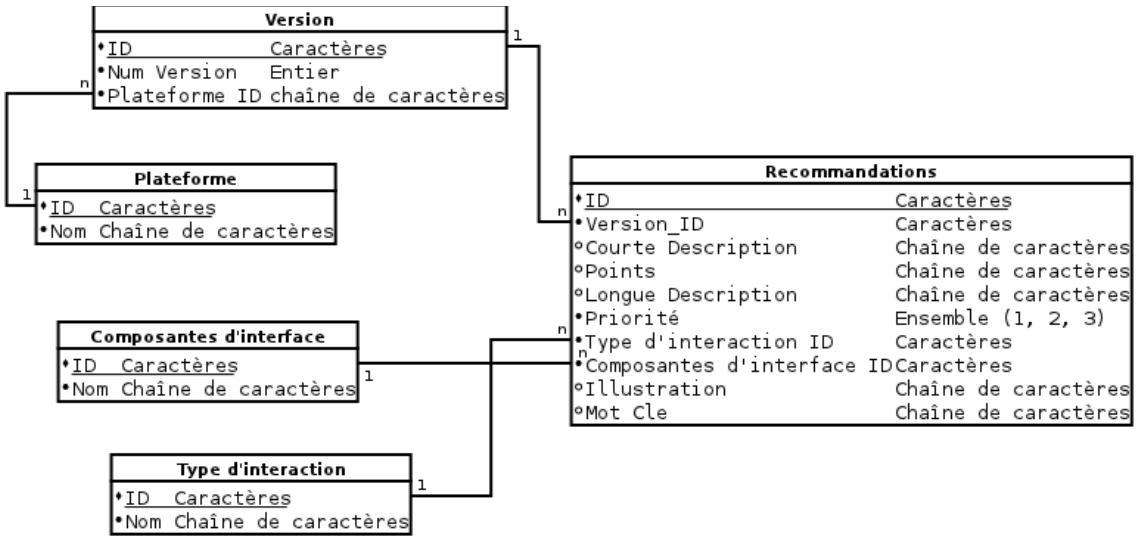


Figure 10: Schéma des tables pour les recommandations

Sur le schéma, on peut voir les différentes tables constituant la base de données. Ces dernières sont symbolisées par deux rectangles superposés, le premier contenant le nom de la table et le second ses attributs. Les flèches représentent les relations entre les différentes tables. Étant donné l'ampleur du projet, une seule base de données a été utilisée, mais cette dernière contient d'une part les recommandations et d'autre part l'image de marque. Bien que les ressources nécessaires pour mener à bien ce projet ne soient pas énormes, nous avons tout de même tenu à définir une structure qui soit efficace. Pour cela, nous avons fait appel à l'expertise de l'un des administrateurs système d'*Air Canada* afin de connaître leurs pratiques en matière de création et de gestion de bases de données.

Après avoir défini le schéma conceptuel de la base de données, nous avons dû le normaliser afin d'éliminer les redondances et faire en sorte que l'organisation générale soit la meilleure possible. Pour cela, nous avons rempli les différentes tables avec certaines des informations qu'elles devaient contenir. Cette opération a permis de mettre en lumière certains problèmes comme l'intégration du numéro de version au sein de la table « plateforme ». En effet, avec la structure retenue, un certain nombre de redondances au niveau des noms de plateformes a été observé, et cela pour chacune des versions de ces dernières. Pour contourner le problème, nous avons opté pour deux tables disjointes, mais

liées par une clé au sein de la table « version » qui fait référence à l'identifiant de la plateforme.

La même approche a également été retenue pour les tables associées à l'image de marque. Afin de pouvoir ajouter ou supprimer des éléments sans pour autant affecter l'ensemble de l'information, nous avons conçu un schéma relationnel à deux niveaux (cf. Figure 11).

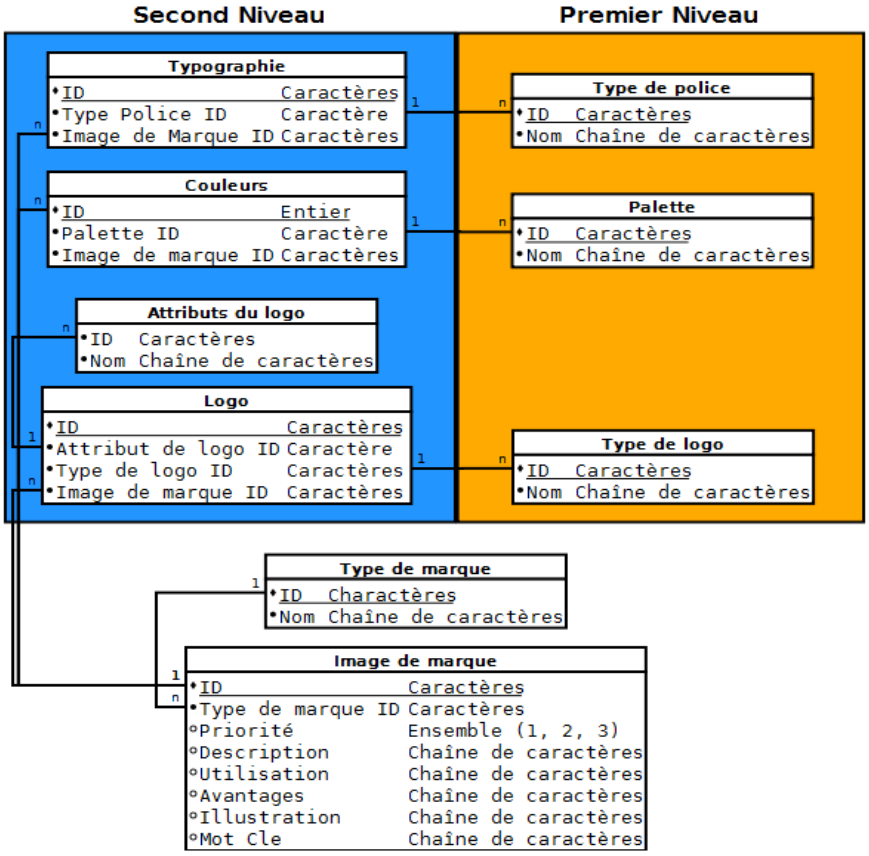


Figure 11: Schéma des tables pour l'image de marque

Dans cette figure, un premier niveau est constitué des tables contenant les informations minimales, comme le type de police, tandis que le second niveau permet de lier les informations du premier niveau avec les recommandations contenues dans la table « image de marque ». Par exemple, dans le cas des polices de caractère, c'est la table « typographie » qui joue ce rôle.

Une base de données est typiquement exploitée à l'aide d'un langage de requêtes respectant la norme SQL (Structured Query Language). Il s'agit d'un langage très puissant

permettant d'effectuer un grand nombre d'opérations sur une base de données. Pour éviter de saisir plusieurs lignes de code afin de créer la structure désirée, nous avons utilisé l'application web *phpMyAdmin*[9] (PMA) qui assure la gestion de bases de données MySQL. Cette application est réalisée en PHP et distribuée sous licence libre. Elle permet d'interagir avec une interface conviviale pour dialoguer avec la base de données plutôt que d'utiliser des lignes de commandes. L'application offre aussi la possibilité de conserver un historique des commandes SQL utilisées, ce qui permet de garder un contrôle sur les transactions effectuées. On peut également sauvegarder le schéma de la base de données dans un fichier texte afin de la faire migrer d'une plateforme à une autre. Enfin, l'application permet de corriger rapidement des erreurs qui peuvent survenir lors du développement de pages web devant accéder à la base de données ou modifier son contenu.

3.3 Interface

Cette section décrit l'interface principale avec laquelle les utilisateurs ont à interagir. Il faut noter que la conception des formulaires d'interaction a été réalisée en partenariat avec les responsables de l'équipe mobile chargés de l'évaluation des travaux fournis par les sous-traitants. L'idée était d'impliquer ces personnes le plus tôt possible dans le processus de développement afin que le prototype réponde au mieux à leurs attentes. Pour cela, nous avons organisé des séances de remue-méninges (« brainstorming ») avec les responsables des plateformes mobiles chez *Air Canada* afin d'identifier l'ensemble des fonctionnalités qui devaient être présentes au sein de l'interface. À partir de là, une approche itérative a été mise en place. D'abord, des maquettes des principaux éléments de l'interface ont été soumises aux responsables pour s'assurer que la présentation générale de l'interface était satisfaisante (cf. Figure 12 et Figure 13).Ces dernières ont peu à voir avec l'interface finale offerte par le système, mais avaient simplement pour but de donner une idée de leur apparence générale à l'équipe mobile et ainsi fournir de support pour les séances de remue-méninges.

9 www.phpmyadmin.net

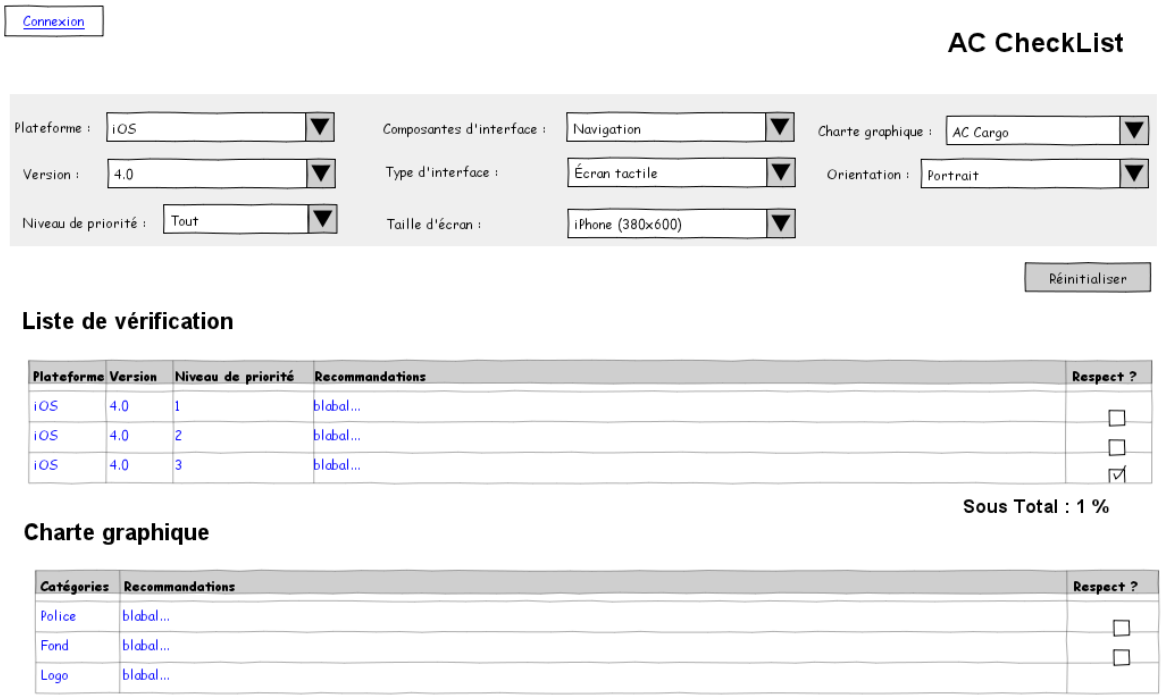


Figure 12: Maquette initiale de l'interface de filtrage des recommandations

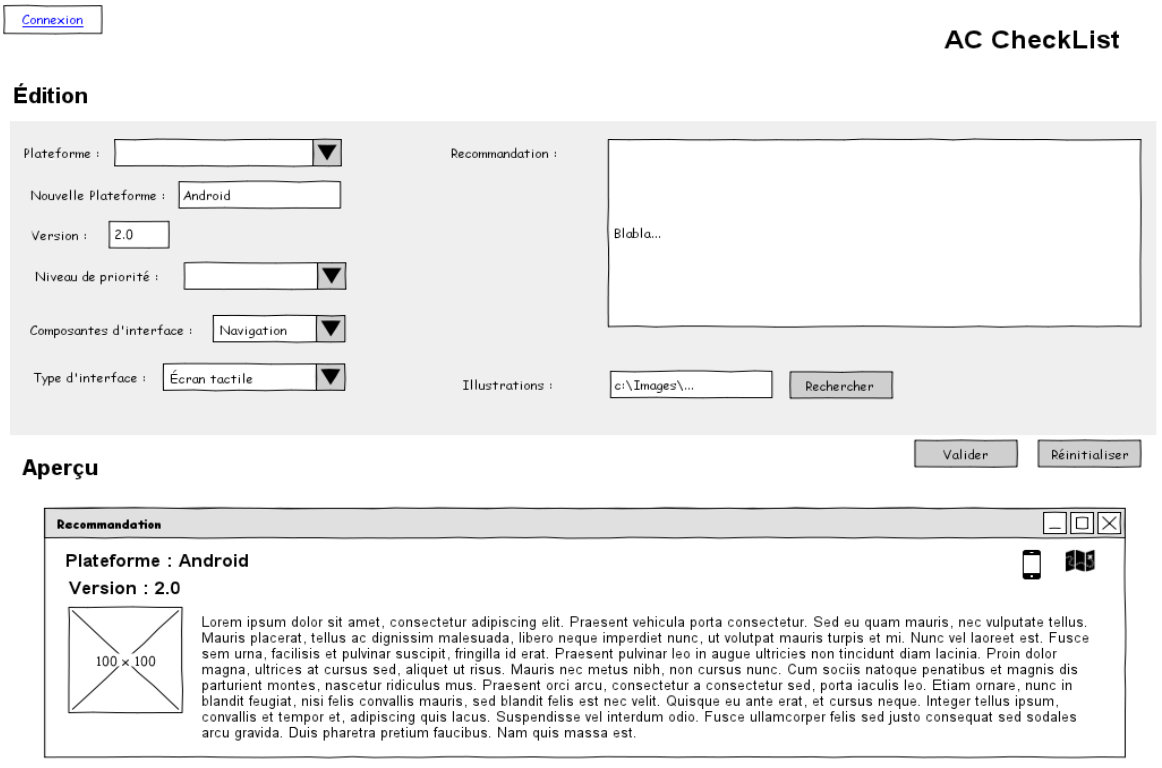


Figure 13: Maquette initiale de l'interface d'ajout et de consultation des recommandations

Pour la réalisation de ce travail, le logiciel de prototypage libre et gratuit *Pencil*[10], qui est un greffon installable sur le navigateur *Firefox*[11], a été utilisé.

Nous avons ensuite expérimenté avec différents modes d'interaction, mais cette fois en créant de véritables pages web avec le langage à balises HTML. Certains problèmes sont alors apparus avec le choix initial des listes de sélection (cf. Figure 14).

Figure 14: Interaction à base de liste de sélection

En effet, bien que l'encombrement spatial soit minimal, les listes de sélection ont le désavantage d'être moins faciles d'utilisation que les cases à cocher. Ainsi, pour sélectionner plusieurs entrées dans une liste de sélection, l'utilisateur doit maintenir la touche Ctrl du clavier enfoncée tandis qu'il effectue la sélection. Il faut donc que l'utilisateur soit familier avec ce type de fonctionnement. Mais surtout, l'utilisateur a besoin d'avoir les deux mains libres. Nous avons donc décidé de privilégier plutôt les cases à cocher (cf. Figure 15).

Figure 15: Interaction à base de cases à cocher

Un autre élément de l'interface a également été modifié. Il s'agit de la liste déroulante qui permet de choisir le type de plateforme ainsi que sa version (cf. Figure 16).

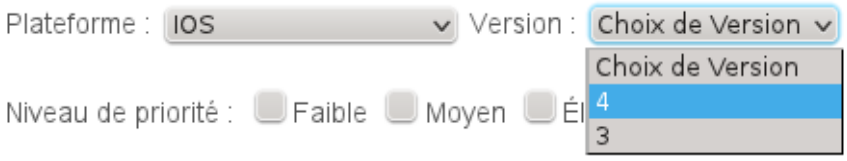


Figure 16: Interaction par listes déroulantes

10 http://pencil.evolus.vn/en-US/Home.aspx

11 www.firefox.com

En effet, l'interface se composait au départ de deux listes afin de permettre à l'usager d'associer une plateforme mobile avec un numéro de version. Cependant, bien que simple d'utilisation, cette approche ne permettait de choisir qu'un seul couple (plateforme, version) à la fois. Or, il est souhaitable que l'interface permette de sélectionner plusieurs plateformes concurremment ainsi que plusieurs versions d'une même plateforme. Comme il peut y avoir un grand nombre de versions pour une plateforme donnée et qu'il fallait éviter de surcharger l'interface avec des redondances ou des listes de choix trop longues, nous avons imaginé une solution basée à la fois sur des cases à cocher et sur l'utilisation d'un tableau afin de réduire l'encombrement spatial (cf. Figure 17).

Figure 17: Interaction par cases à cocher et tableau

En plus du langage à balises HTML, nous avons opté pour l'utilisation des feuillets de style en cascade CSS (Cascading Style Sheet) afin de garantir une plus grande flexibilité et une séparation entre le fond et la forme. Cette solution standard proposée par le W3C (World Wide Web Consortium) permet de s'affranchir des besoins de mise en page lors de la phase de conception.

Au niveau du codage, nous avons affecté des identifiants à la majeure partie des balises utilisées ainsi que des noms de classe aux éléments partageant les mêmes attributs afin de pouvoir modifier leur style plus facilement par la suite. Dans un souci de simplicité et de clarté, nous avons gardé une mise en page relativement simple et épurée en évitant d'introduire un trop grand nombre d'éléments graphiques pour ne pas encombrer l'interface. Nous avons également suivi les directives propres à l'image de marque d'*Air Canada*. En effet, nous avons retenu l'ensemble des couleurs recommandé dans la charte graphique ainsi que la police de caractère *Verdana*.

3.3.1 Interactivité

Afin de disposer de pages web dynamiques, deux types de technologies différentes ont été utilisées, l'une du côté serveur et l'autre du côté client. Il s'agit du langage de scripts PHP, utilisé entre autres pour communiquer avec la base de données et en extraire des informations, et de JavaScript, un langage utilisé pour le navigateur de l'utilisateur afin d'offrir une certaine interactivité avec les éléments affichés à l'écran. Le recours à PHP permet d'offrir une interface qui évolue automatiquement avec les ajouts à la base de données. PHP évite également d'exposer des données sensibles au regard de l'utilisateur, étant donné que tous les traitements sont effectués au niveau du serveur et que l'usager n'a accès qu'au code HTML généré. C'est pour cette raison que PHP est utilisé pour tous les besoins d'identification, de gestion de mot de passe et de connexion à la base de données. PHP permet aussi de transférer les images qui sont ajoutées afin d'illustrer les recommandations. L'emploi de JavaScript est principalement réservé pour afficher ou masquer des éléments de l'interface ainsi que pour le calcul du score obtenu par une maquette en fonction du degré de conformité avec les recommandations (voir plus loin). JavaScript est également utilisé pour afficher en taille réelle les images d'aperçu qui illustrent les différents points de la liste de vérification. Dans ce dernier cas, nous avons aussi fait appel à une application libre distribuée gratuitement sur internet, baptisée *Lightbox2*[12], qui a ensuite été adaptée à nos besoins (cf. Figure 18).

12 http://www.lokeshdhakar.com/projects/lightbox2/

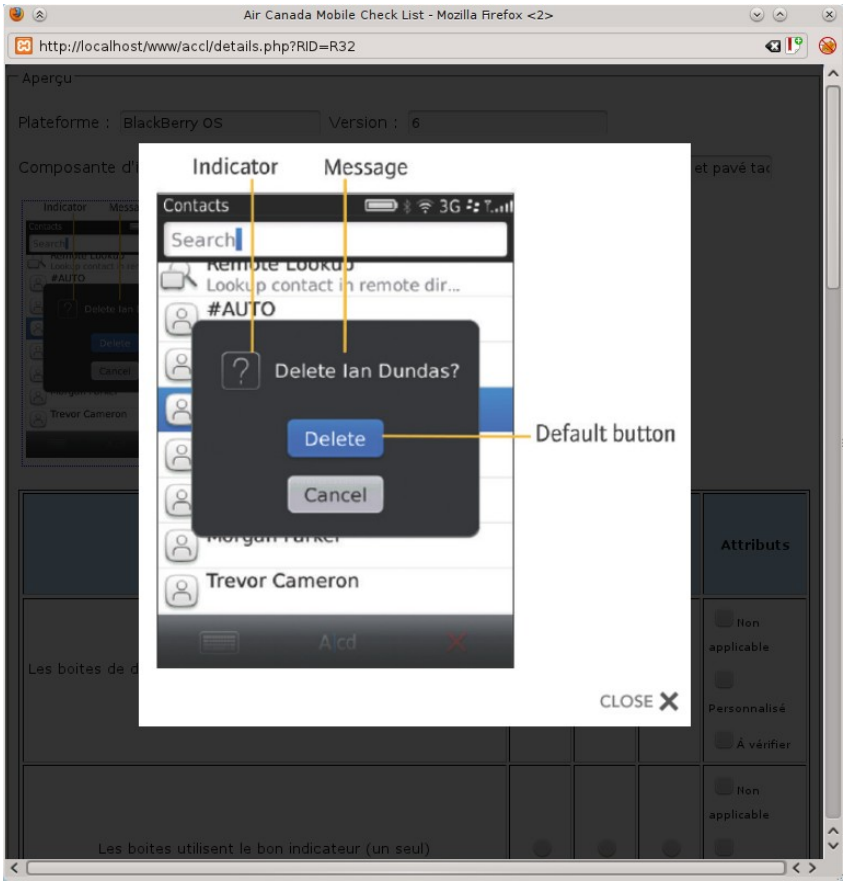


Figure 18: Aperçu d'une image avec Lightbox2

3.3.2 Administration

Pour éviter que « tout un chacun » puisse ajouter de nouvelles recommandations dans la base de données ou modifier celles déjà existantes, il fallait disposer d'une solution de gestion des droits. Or, une telle implémentation peut être relativement longue et complexe, surtout si l'on veut s'assurer qu'il n'y ait pas de failles de sécurité et disposer de fonctions avancées comme la récupération des mots de passe en cas de perte ou encore la gestion de groupes d'utilisateurs. Nous sommes donc partis du principe qu'il n'était pas nécessaire de disposer d'une solution très élaborée pour ce projet et nous nous sommes limités à la connexion avec un identifiant et un mot de passe. Pour obtenir un niveau de sécurité un peu plus élevé, nous avons choisi de ne pas sauvegarder les mots de passe en clair dans la base de données, mais plutôt de les crypter avec une fonction de hachage. Cette méthode prend en entrée une chaîne de caractères et calcule en sortie une empreinte

servant à identifier rapidement la donnée initiale. On peut identifier un utilisateur en calculant son empreinte à chaque fois qu'il fournit son mot de passe et en la comparant avec celle qui est stockée dans la base de données. Ce mécanisme est loin d'être totalement fiable, mais il a l'avantage de fournir un degré satisfaisant de protection tout en étant facile à mettre en place.

Pour éviter à l'utilisateur autorisé de s'identifier à chaque page lorsqu'il effectue des tâches d'administration, un système de session a été mis en place. Ce dernier permet, une fois que l'usager s'est authentifié, de créer un registre au niveau du serveur et d'y stocker certaines informations comme l'identifiant et l'empreinte du mot de passe de l'usager. Ce registre évite ainsi à l'utilisateur légitime de devoir saisir ses identifiants à chaque nouvelle page. De plus, il empêche qu'un utilisateur mal intentionné puisse accéder directement à des pages non autorisées simplement en saisissant le chemin d'accès à ces pages dans la barre d'adresse du navigateur. Dans un tel cas de figure, le système vérifiera au préalable qu'un registre a bien été créé avant d'accorder l'accès. Autrement, l'utilisateur sera automatiquement redirigé vers la page d'authentification afin de s'identifier.

La solution *phpMyAdmin* mentionnée précédemment permet un grand nombre de manipulations dans la base de données. Cependant, bien qu'il s'agisse d'un outil très complet et très utile, le grand nombre de fonctionnalités offertes peut rapidement dérouter le simple utilisateur qui n'est pas familier avec l'application. De plus, les options avancées qui sont offertes entraînent un risque de mauvaise manipulation. Pour simplifier la manipulation des différentes tables de la base de données, nous avons donc opté pour une solution plus simple qui limite le nombre d'options lors de l'ajout, la suppression ou la modification. Pour cela, nous avons fait appel à l'application web libre et gratuite *phpMyEdit*[13]. Cette dernière permet de générer automatiquement du code PHP pour interroger la base de données ainsi que pour afficher et organiser l'information contenue dans les différentes tables de la base de données. Cette approche présente un double avantage. D'une part, elle permet à l'utilisateur de manipuler l'information contenue dans la base de données de manière plus conviviale et, d'autre part, elle permet de gagner du temps

13 www.phpmyedit.org/

en évitant de coder soi-même l'interface pour manipuler les tables ainsi que pour formuler les différentes requêtes permettant d'interroger la base de données. Nous avons tout de même dû adapter une partie du code généré par *phpMyEdit* pour que les pages créées soient compatibles avec notre système.

3.4 Organisation de l'interface

Après avoir présenté les différents éléments qui composent le système de listes de vérification interactives, nous décrivons maintenant l'interface de manipulation de ces listes. Globalement, cette interface offre deux grandes catégories de fonctionnalités. La première catégorie englobe tout ce qui touche à la consultation, c'est-à-dire le filtrage et l'évaluation. La deuxième regroupe les fonctions d'édition et d'administration du système, plus précisément l'ajout et la modification de recommandations, ainsi que la gestion des utilisateurs et des tables de la base de données. La première catégorie est accessible à tous, étant donné qu'elle préserve l'intégrité de l'information contenue dans la base de données. À l'opposé, la gestion des utilisateurs ainsi que l'ajout et la modification des contenus ne doivent pas être accessibles à tous pour des raisons évidentes de sécurité et afin de préserver la cohérence de la base de données.

3.4.1 Consultation

3.4.1.1 Les filtres

L'une des caractéristiques principales du système est la possibilité de filtrer les résultats afin d'obtenir, non pas la liste de toutes les recommandations disponibles dans le système, mais plutôt une sous-liste qui contient seulement les recommandations qui intéressent l'utilisateur (cf. Figure 19). Le haut de l'écran a donc été réservé à ces filtres. Cet espace est divisé en deux parties, l'une pour les recommandations propres aux plateformes mobiles et l'autre pour l'image de marque. Chacune de ces parties comprend un certain nombre d'éléments qui peuvent être sélectionnés afin de filtrer les résultats. Une fois que les sélections désirées ont été faites, il ne reste plus qu'à appuyer sur le bouton « filtrer » pour obtenir une sous-liste qui correspond à la sélection.

3.4.1.1.1 Les plateformes mobiles

La première partie est constituée des filtres permettant de choisir la plateforme mobile, son numéro de version, le niveau de priorité des recommandations, le type de recommandations (voir Tableau 3) ainsi que le type d'interaction. Les choix disponibles sont offerts grâce à des cases à cocher afin que l'usager puisse, d'un simple coup d'œil, voir l'ensemble du contenu et puisse en même temps procéder à la sélection.

3.4.1.1.2 L'image de marque

La deuxième partie correspond à l'image de marque. L'utilisateur peut tout d'abord choisir la marque à laquelle feront référence les autres sélections. Dans notre cas, il n'y a qu'un seul choix (*Air Canada*), car nous n'avons pas encore ajouté les images de marque des autres divisions de la compagnie. On retrouve ici aussi la possibilité de choisir le niveau de priorité, puis un ensemble de choix répartis en trois catégories : le logo, les couleurs et la typographie. Lorsque l'on sélectionne une entrée dans la liste des logos, un autre ensemble de cases à cocher apparaît afin de sélectionner certains attributs associés à ce logo comme on peut le voir dans le cadre rouge (cf. Figure 19).

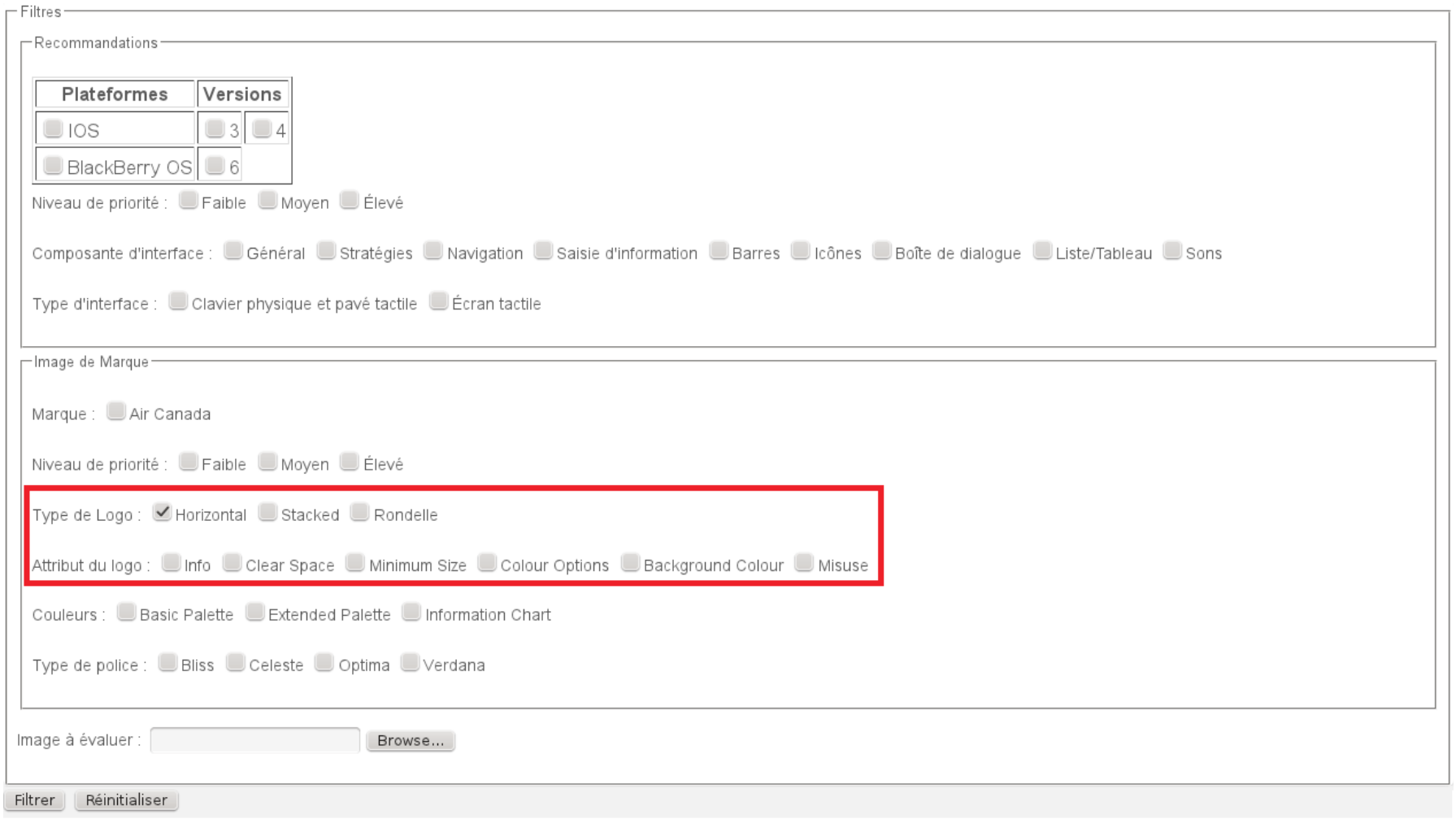


Figure 19: Capture d'écran de l'interface de sélection des filtres

3.4.1.2 Recommandations pour les plateformes mobiles

Les recommandations obtenues sont présentées sous la forme d'un tableau. Pour ne pas surcharger l'interface, l'information est organisée sur deux niveaux. Le premier niveau ne comprend que le nom de la plateforme, le numéro de version, le niveau de priorité de la recommandation, une courte description, le niveau de conformité de la maquette avec la recommandation (tel qu'évalué par l'utilisateur), les critères d'exclusion permettant d'éliminer la recommandation de l'évaluation et un espace réservé pour un commentaire de l'utilisateur (cf. Figure 20).

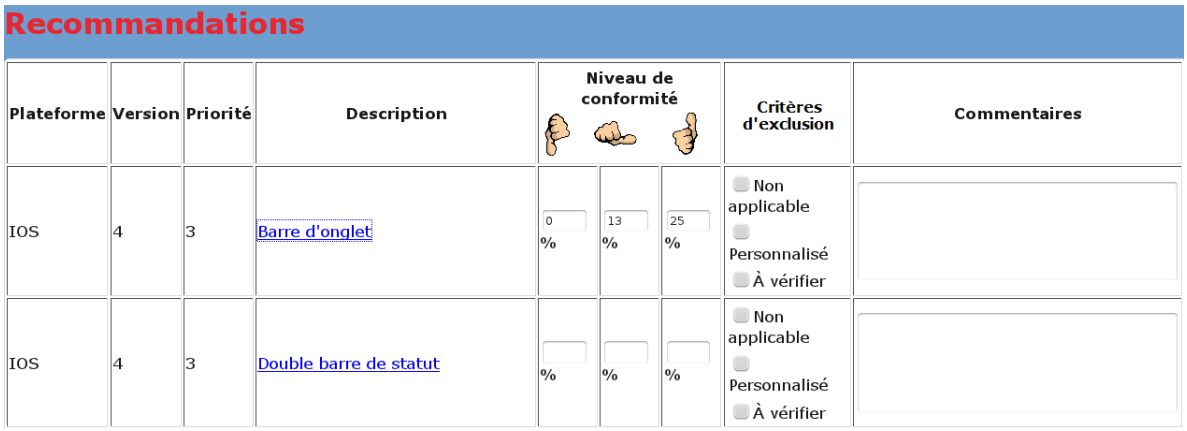


Figure 20: Capture d'écran du premier niveau de l'interface

Tel que mentionné précédemment, un critère d'exclusion peut être activé par l'utilisateur s'il juge que la recommandation ne s'applique pas (« Non applicable »), qu'il choisit délibérément d'y contrevenir (« Personnalisé ») ou qu'il lui est impossible de vérifier si la recommandation est respectée (« À vérifier »).

Le second niveau est accessible lorsque l'on clique sur la description courte d'une recommandation dont le texte constitue un hyperlien. Une nouvelle fenêtre de plus petite dimension composée de trois parties s'ouvre alors (cf. Figure 21).

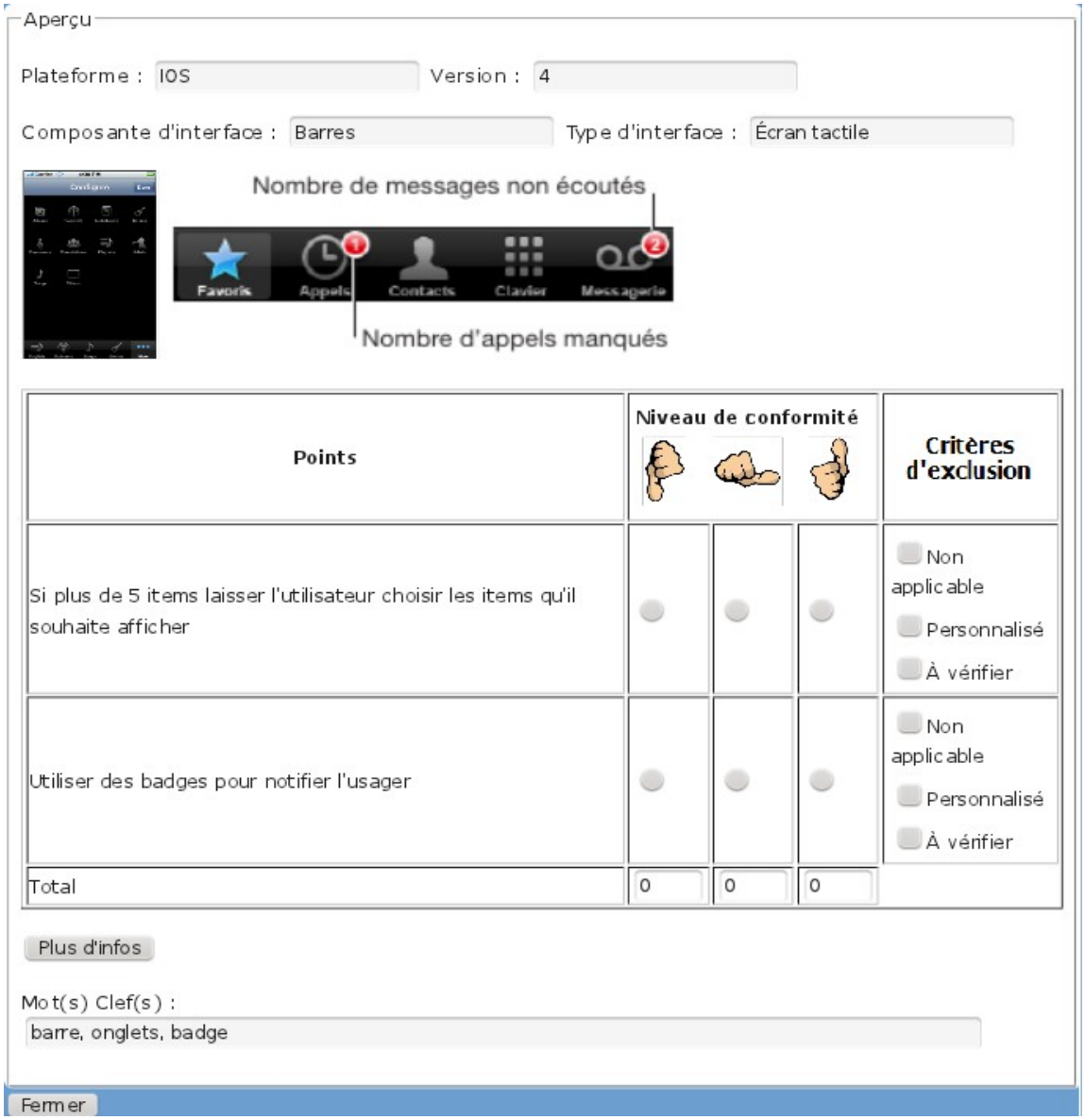


Figure 21: Capture d'écran du deuxième niveau de l'interface

Dans la première partie, soit celle située en haut de la fenêtre, on retrouve à nouveau le nom de la plateforme ainsi que le numéro de version, mais également le composant d'interface correspondant à la recommandation, le type d'interaction et des illustrations. La nouvelle fenêtre venant masquer l'information contenue au premier niveau, il était important de dupliquer le nom de la plateforme ainsi que sa version afin de maintenir un lien sémantique. La seconde partie de la fenêtre se compose d'un tableau présentant l'ensemble des points faisant partie de la recommandation et auxquels il faut se conformer. Des boutons radio

permettent de définir le niveau de conformité de la maquette avec chacun des points. Enfin, on retrouve trois critères d'exclusion, tout comme au premier niveau, si on ne veut pas tenir compte d'un point. La dernière partie, au bas de la fenêtre, contient le bouton "plus d'infos" qui permet d'obtenir un petit texte explicatif sur chacun des points faisant partie de la recommandation ainsi que les mots clés associés à la recommandation.

3.4.1.3 Calcul du score

L'évaluation se fait en deux étapes. La première étape se déroule dans la fenêtre au second niveau qui affiche les différents points associées à une recommandation (cf. Figure 21). L'utilisateur doit alors choisir un niveau de conformité (conforme, partiellement conforme, non conforme) ou un critère d'exclusion. Le nombre de sélections associé à chaque niveau de conformité s'affiche ensuite dans l'entrée « Total ». L'utilisateur n'a plus qu'à fermer la fenêtre pour qu'un score en pourcentage apparaisse au premier niveau pour chaque niveau de conformité de la recommandation correspondante. Le pourcentage pour un niveau de conformité donné est obtenu en additionnant le nombre de points pour lequel le niveau correspondant a été sélectionné divisé par le nombre total de points considérés (cf. Figure 20). En cliquant sur le bouton moyenne, situé dans le bas du tableau des recommandations au premier niveau, un pourcentage moyen pour l'ensemble des recommandations considérées est obtenu (cf. Figure 22).

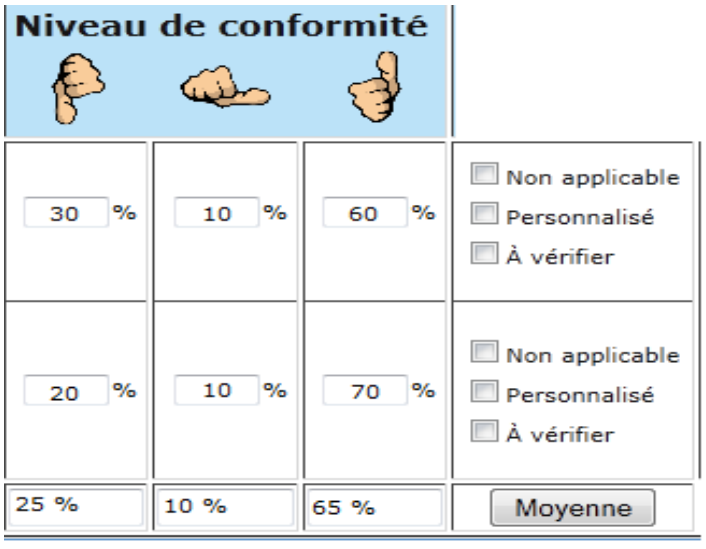


Figure 22: Capture d'écran du calcul du score moyen en fonction du niveau de conformité par recommandation

3.4.1.4 Recommandations pour l'image de marque

Sous le tableau des recommandations, on retrouve une partie réservée à l'image de marque. Cette dernière comprend trois tableaux, chacun destiné à un aspect de la charte graphique à savoir le logo, les couleurs et la typographie. Cette partie a surtout une vocation informative, puisqu'on n'y a pas intégré une évaluation du niveau de conformité. En effet, l'idée était ici de faciliter la consultation de ce type d'information plutôt que de mettre en place une réelle interface d'évaluation. Nous avons pu ainsi consacrer plus de temps à l'évaluation des recommandations pour les plateformes mobiles. Ici aussi, l'interface est organisée sur deux niveaux, le premier n'offrant que le niveau de priorité et la catégorie à laquelle la recommandation appartient. Le deuxième niveau est divisé en trois parties. La première partie contient une explication de l'élément consulté, la deuxième partie explique comment l'élément doit être utilisé et la dernière partie énumère les avantages à respecter la recommandation. Cette organisation a été calquée sur celle mise en place par l'équipe de mercatique d'*Air Canada* afin de faciliter l'intégration d'autres images de marque dans le système.

3.4.2 Édition

Après avoir décrit l'interface de consultation, nous allons maintenant nous intéresser aux fonctionnalités de l'interface permettant à l'utilisateur d'ajouter ou de modifier des informations dans le système. Comme nous l'avons mentionné plus tôt, l'ensemble des options offertes ici n'est destiné qu'aux personnes autorisées, munies d'un identifiant et d'un mot de passe, afin d'éviter que le contenu du système soit modifié inopinément. Afin de respecter la volonté des responsables de l'équipe mobile, il est impossible actuellement de supprimer du contenu afin d'éviter la perte de données, mais également afin de conserver la trace des changements apportés.

3.4.2.1 Ajout

On retrouve ici deux interfaces, l'une pour ajouter des recommandations propres aux plateformes mobiles et l'autre pour celles associées à l'image de marque.

3.4.2.1.1 Les plateformes mobiles

Figure 23: Formulaire d'ajout de recommandations

L'utilisateur peut, au choix, utiliser une plateforme et un numéro de version déjà existants dans le système ou bien ajouter un nouveau numéro de version à une plateforme déjà existante ou encore créer une nouvelle plateforme (cf. Figure 23). Ces options permettent ainsi de faire évoluer le système en fonction des versions futures d'une plateforme existante ou de créer des entrées pour un nouvel appareil et ainsi étendre les possibilités du système. L'utilisateur peut ensuite spécifier le niveau de priorité qu'il souhaite affecter à la nouvelle recommandation, son type et la composante de l'interface à laquelle elle se rapporte. Il faut aussi fournir un titre ou une courte description, puis la liste des points qui la composent. On peut aussi ajouter un texte explicatif ainsi que des images

pour rendre les explications plus claires et les illustrer. Enfin, un ensemble de mots clés peuvent être saisis pour caractériser la recommandation lorsque le moteur de recherche sera implémenté. Afin de rendre l'interface plus conviviale et éviter l'encombrement, nous avons fait en sorte de ne présenter que les champs pertinents lors de la sélection d'une plateforme ou de sa version selon que ces éléments sont nouveaux ou déjà existants dans le système (cf. Figure 24). De plus, une seule ligne apparaît par défaut pour l'ajout d'un point avec un bouton qui permet d'ajouter d'autres lignes au besoin (cf. Figure 25).

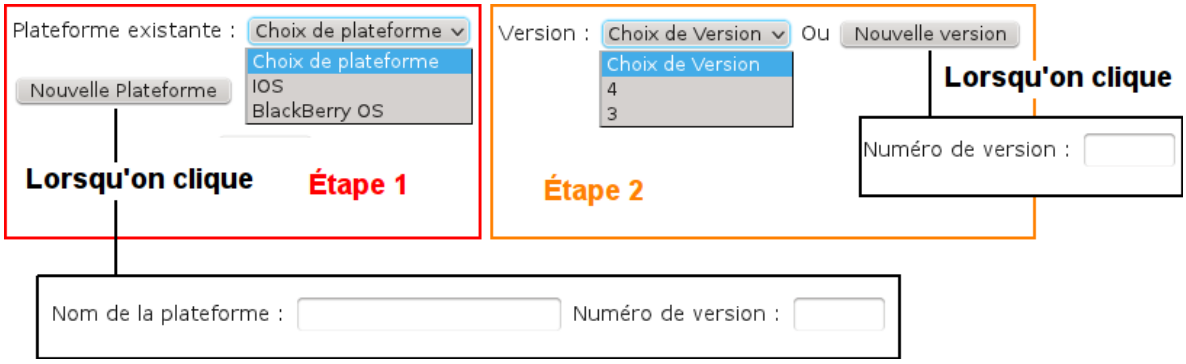


Figure 24: Schéma d'interaction lors de l'ajout d'une nouvelle recommandation

Figure 25: Ajout de point interactif

Pour ajouter des illustrations, nous avons opté pour une solution qui permet d'afficher un aperçu de l'image à ajouter et la possibilité de la supprimer avant de la soumettre (cf. Figure 26).

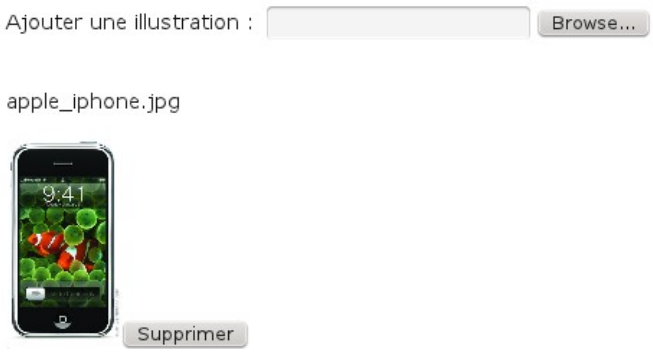


Figure 26: Ajout d'illustrations ergonomiques

Après avoir rempli l'ensemble des champs, l'utilisateur peut alors procéder à la validation afin de nourrir la base de données. Une rétroaction est alors fournie à l'utilisateur pour lui indiquer l'état du processus, en l'occurrence si l'ajout dans la base de données s'est bien déroulé ou s'il y a eu un problème. Le résumé des données saisies s'affiche également afin que l'utilisateur puisse vérifier qu'il n'y a pas d'erreur. Enfin, trois boutons permettent d'ajouter à nouveau une nouvelle entrée, de modifier celle qui vient d'être ajoutée ou de revenir à l'accueil.

3.4.2.1.2 L'image de marque

Après avoir choisi la marque et le niveau de priorité, la procédure d'ajout diffère ici légèrement selon que les informations à fournir concernent le logo, les couleurs ou la typographie (cf. Figure 27).

Figure 27: Formulaire pour l'ajout d'image de marque

L'utilisateur a d'abord la possibilité de choisir l'une des trois catégories pour laquelle il souhaite ajouter une entrée, puis les champs à remplir en fonction de la catégorie choisie lui sont fournis. Les formulaires les plus simples sont ceux pour les couleurs et la typographie puisqu'ils permettent seulement de choisir un type de police ou de palette déjà présent dans le système ou bien d'en ajouter de nouveaux. Pour les logos, c'est à peine plus compliqué puisqu'on retrouve seulement le type de logo et certains attributs. Quelle que soit la catégorie, on retrouve toujours les informations de base relatives à l'image de marque, à savoir une description de l'élément, ses utilisations possibles et ses avantages. Enfin, il est possible de spécifier des mots clés et d'ajouter des images grâce à une interface similaire à celle offerte pour les recommandations pour les plateformes mobiles.

3.4.2.2 Modifications

Conscient que des erreurs peuvent survenir lors de la phase d'ajout, l'utilisateur a la possibilité de modifier des entrées tant pour les recommandations touchant aux plateformes mobiles que pour celles concernant l'image de marque. En outre, cette fonctionnalité permet de faire évoluer la base de données afin qu'elle réponde au mieux aux attentes des utilisateurs. Il est possible, par exemple, de faire migrer un ensemble de recommandations pour une certaine version d'une plateforme vers une autre ou bien de modifier le type d'interaction auxquelles elles réfèrent. Globalement, l'interface est relativement similaire à celle pour l'ajout si ce n'est que le nom de la plateforme ne peut être modifié et que l'information dans les différents champs est déjà remplie. L'utilisateur peut ainsi modifier le contenu des champs, mais également supprimer des illustrations qui ont été ajoutées sur le serveur lors de la création de la recommandation. Afin de rendre les modifications plus faciles, un bouton "modifier" apparaît en bas de la fenêtre d'évaluation lorsque l'utilisateur est authentifié. Ainsi, l'utilisateur peut modifier très simplement la recommandation qu'il est en train de consulter. De plus, grâce au système de session, ce dernier n'a pas besoin de fournir constamment son identifiant et son mot de passe. Enfin, une rétroaction est offerte à l'usager tout comme lors de l'ajout, en l'occurrence une phrase explicative sur l'état de la transaction, un résumé des données saisies et des liens permettant de modifier l'entrée ou de revenir à l'interface d'évaluation.

Nous ne nous étendrons pas davantage sur les modifications à l'image de marque, étant donné que l'approche est semblable à celle adoptée pour les plateformes mobiles.

3.4.3 Administration

Après avoir traité des éléments en lien avec l'ajout de nouvelles entrées ainsi que leur modification, il nous faut maintenant parler de la partie administration du système qui se veut minimale. En effet, cette partie ne participant pas directement au fonctionnement du système d'évaluation, elle est moins importante dans le cadre de ce travail. Nous offrons tout de même certaines fonctionnalités afin de rendre l'interface plus conviviale et faciliter son administration. Nous nous sommes donc restreints à trois fonctions principales, soit l'ajout d'un nouvel utilisateur, la modification du mot de passe et la gestion de la base de données.

3.4.3.1 Ajout d'utilisateurs et modification du mot de passe

Sachant que l'entrée dans la partie administrative du système requiert un identifiant et un mot de passe, nous avons créé un mot de passe par défaut. L'idée est de permettre aux véritables administrateurs de s'y connecter une première fois et ensuite de modifier leur mot de passe pour rendre le système plus sécuritaire. Le système n'ayant pas pour vocation de recueillir un grand nombre d'informations, la procédure d'enregistrement est très simple. Elle ne comporte qu'un champ pour l'identifiant et un autre pour le mot de passe. La modification du mot de passe est tout aussi simple. Là aussi il n'y a que deux champs, l'un destiné à la saisie de l'ancien mot de passe et l'autre pour le nouveau. En demandant l'ancien mot de passe, on s'assure qu'il s'agit d'un utilisateur légitime qui souhaite changer son mot de passe et non pas d'une personne qui utilise la session qu'un autre utilisateur aurait laissée ouverte. La procédure génère simplement une empreinte à partir de l'information fournie par l'utilisateur grâce à une fonction de hachage et la compare avec l'empreinte stockée pour la session courante. Si les deux empreintes correspondent, on remplace alors l'ancien mot de passe par le nouveau et la nouvelle empreinte est associée à la session. Autrement, un message d'erreur s'affiche qui demande à l'utilisateur de saisir à nouveau son ancien mot de passe.

3.4.3.2 Gestion de la base de données

On présente ici à l'utilisateur la liste des différentes tables qui composent la base de données. Ensuite, l'utilisateur peut effectuer un certain nombre d'actions sur les entrées qui composent les tables, comme ajouter une nouvelle entrée, la modifier, la copier ou la supprimer. De plus, pour éviter de surcharger l'interface, on n'affiche qu'une quinzaine d'entrées par page. Il faut noter que les tables de liaison ne sont pas accessibles, car elles sont gérées automatiquement par le système. Ainsi, on n'offre à l'utilisateur que les tables avec des informations facilement compréhensibles. La manipulation de la base de données ne devrait pas normalement se faire avec cette interface, car le système offre déjà des facilités pour ajouter ou modifier son contenu. Mais si jamais l'utilisateur souhaite effectuer des tâches bien particulières, il faut être en mesure de lui offrir cette possibilité. D'autant plus que le système n'est pas à l'abri d'une mauvaise manipulation ou d'une corruption des données dûes à des problèmes de serveur. On recommande néanmoins l'installation en parallèle de l'outil *phpMyAdmin* parce qu'il est bien plus complet et permet de faire des manipulations plus sophistiquées de la base de données aussi bien au niveau de son contenu que de sa structure.

3.5 Conclusion

L'implémentation du système nous a permis de faire appel à différentes technologies du web. Ce fut d'ailleurs une expérience très enrichissante qui nous a permis de constater les limites de chacune de ces technologies, mais également de découvrir comment elles se complètent pour fournir une solution homogène et cohérente. Il faut remarquer que l'on a travaillé exclusivement avec des logiciels libres et gratuits. En effet, du système d'exploitation GNU/Linux, en passant par le serveur web Apache, la base de données MySQL ou encore le navigateur Firefox, tous les produits utilisés sont librement modifiables, gratuitement téléchargeables et ont démontré leurs bénéfices sur le terrain. Cet écosystème de solutions logiciels tend à prouver qu'il est tout à fait possible de mener un projet d'envergure sans budget important ou sans être dépendant d'une quelconque licence comme c'est le cas pour les solutions propriétaires disponibles sur le marché.

CHAPITRE 4 : Un exemple d'utilisation de l'outil d'évaluation de la conformité de la plateforme iOS aux normes du fabricant

Ce chapitre présente les différentes étapes nécessaires pour procéder à une évaluation de conformité d'une maquette avec les normes du fabricant. Nous commençons par présenter certaines spécificités des maquettes. Nous passons ensuite en revue les différentes fonctionnalités nécessaires pour réaliser une évaluation comprenant un filtrage des recommandations, l'évaluation proprement dite et le calcul d'un score. Chacune de ces étapes est illustrée avec un exemple concret portant sur un composant bien précis de l'interface, soit la barre de navigation pour la plateforme iOS.

4.1 Les maquettes

Les maquettes correspondent à des interfaces en cours de développement chez IBM et qui sont soumises à Air Canada pour fins d'évaluation (cf. Figure 28). Au départ, les maquettes prenaient la forme de captures d'écran de l'interface sous différentes résolutions et distribuées sous la forme d'un fichier de présentation *PowerPoint*. Afin de standardiser le format des maquettes, nous avons suggéré que ces dernières soient envoyées à l'échelle 1:1 et sous format PDF afin d'éviter les problèmes d'incompatibilité. Ce sont souvent les maquettes pour la plateforme iOS qui sont générées en premier et elles sont généralement plus soignées que celles du *BlackBerry*. On a donc fait le choix d'évaluer la plateforme iOS pour illustrer le fonctionnement du système de listes de vérification interactives. Pour le moment, le système ne permet pas d'intégrer ces captures d'écran. L'utilisateur doit donc les ouvrir « à part » avec un logiciel approprié avant de démarrer le système.

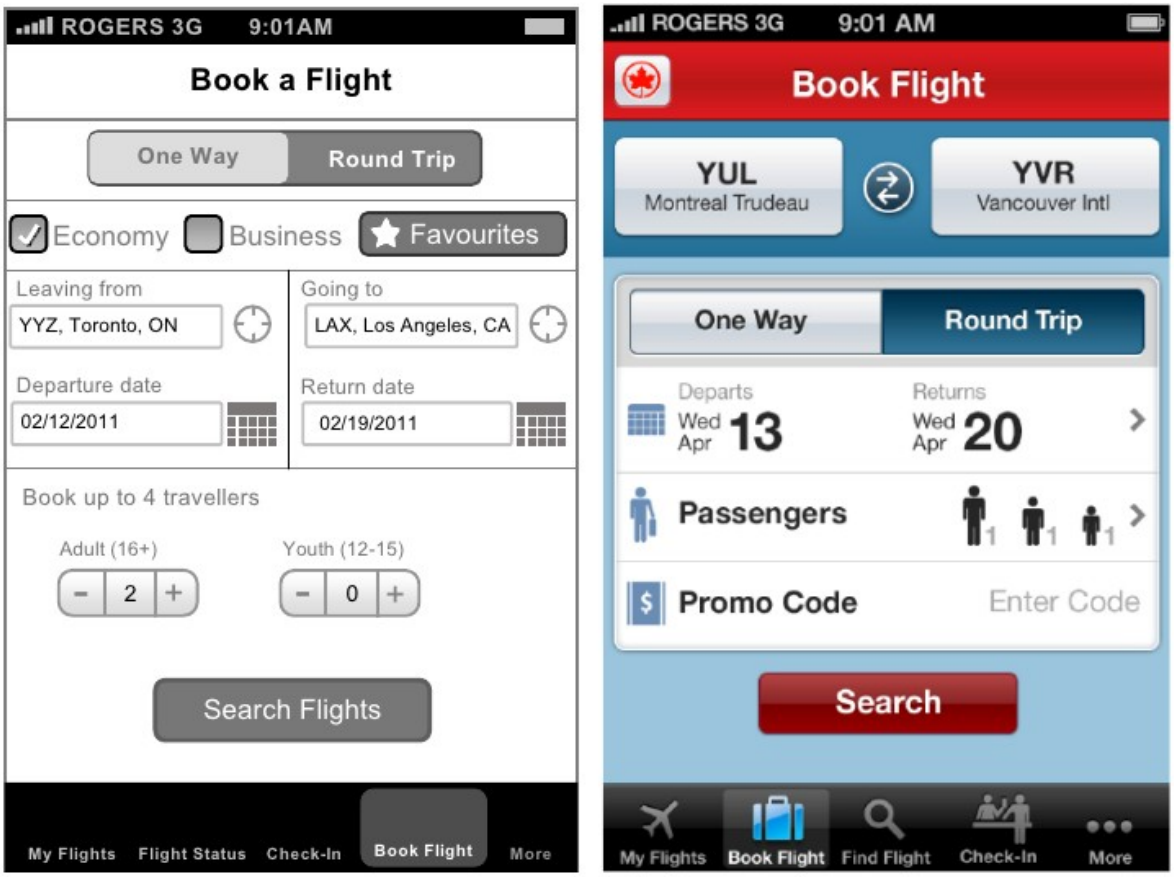


Figure 28: Maquette fournie par *IBM* (à gauche version initiale et à droite le résultat après la 5e phase d'itérations)

4.2 Filtrage des recommandations

4.2.1 Choix de la plateforme et de la version

Si l'utilisateur désire se concentrer sur un aspect particulier de l'interface graphique, il doit filtrer l'ensemble des recommandations disponibles pour ne prendre en considération que celles qu'il souhaite évaluer. Pour ce faire, il doit utiliser les filtres disponibles. Il faut tout d'abord commencer par le choix de la plateforme ainsi que la version (cf. Figure 29). Dans notre exemple, la plateforme iOS version 4 a été choisie.

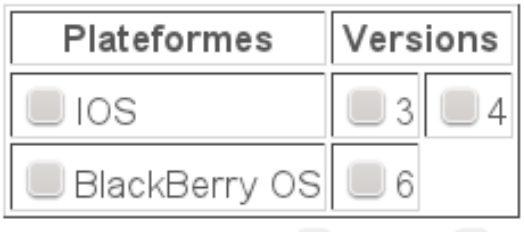


Figure 29: Choix de plateforme et de la version

4.2.2 Choix du niveau de priorité

Contrairement au choix de la plateforme qui est obligatoire, les autres filtres ne le sont pas, ainsi le choix du niveau de priorité des recommandations est facultatif (cf. Figure 30). Lorsque sélectionné, il permet par exemple à l'utilisateur de ne s'attarder qu'aux recommandations les plus importantes. Cela peut être fort utile lorsque le temps presse.

Figure 30: Choix du niveau de priorité

4.2.3 Choix du type de composantes d'interfaces et du type d'interface

Un filtre permet à l'utilisateur de se concentrer sur certains aspects particuliers de l'interface. Il est possible ici de ne considérer qu'un ou plusieurs types de composantes. Enfin, il est possible de choisir un type d'interface. Ainsi, à la Figure 31, l'utilisateur choisit les recommandations associées au composante « Barres » avec « Écran tactile » comme type d'interaction.

Figure 31: Choix des composantes d'interface et du type d'interaction

4.3 Évaluation

4.3.1 Résultat du filtrage

Après avoir activé les différents filtres et appuyé sur le bouton « filtrer », la liste des recommandations résultantes s'affiche (cf. Figure 32). Tel que mentionné au chapitre précédent, les critères d'exclusion permettent de ne pas tenir compte de certaines recommandations.

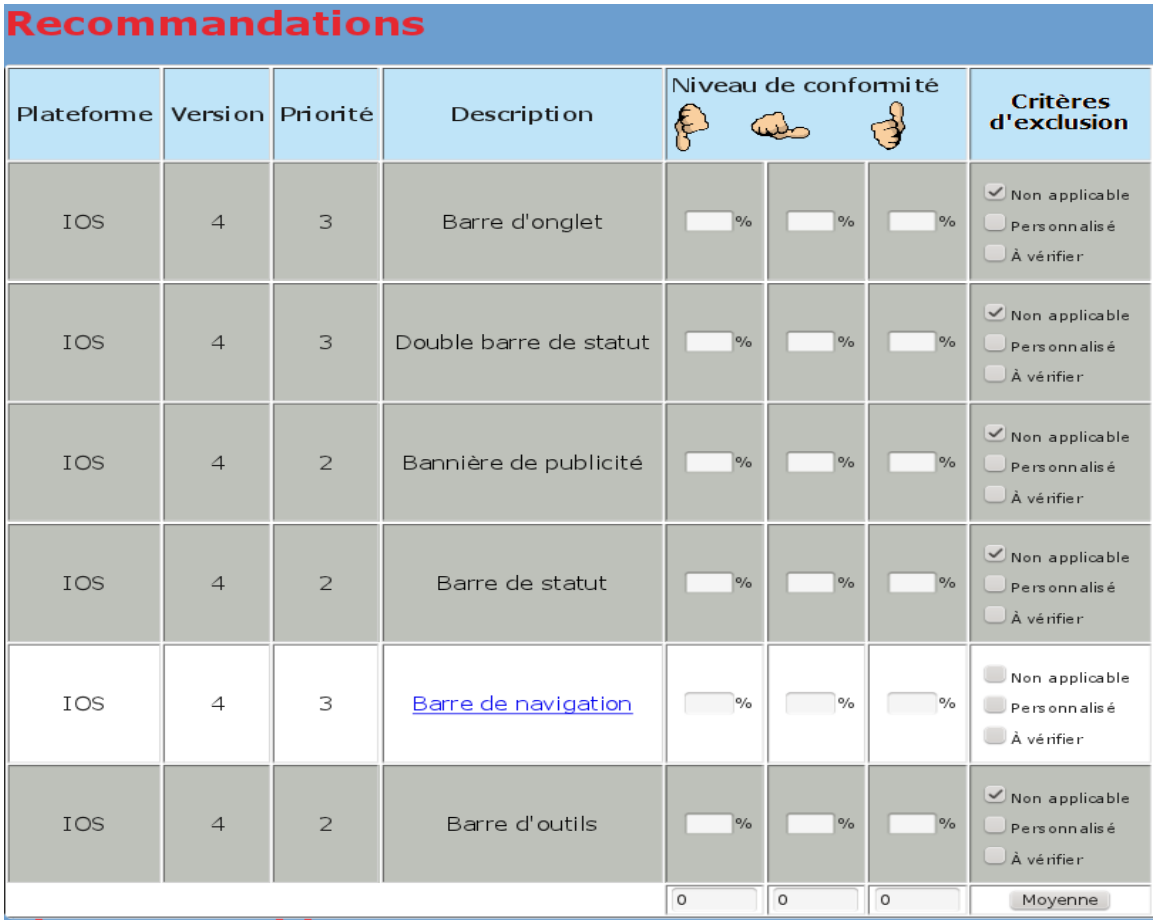


Figure 32: Capture d'écran de la liste des composantes d'interface après filtrage

Dans l'exemple retenu, l'utilisateur ne désire évaluer que la barre de navigation. Toutes les autres recommandations ont donc été désactivées en sélectionnant le critère « Non applicable ».

4.3.2 Liste de points associés à une recommandation

Après avoir choisi un type de composante en cliquant sur son hyperlien, une nouvelle fenêtre contenant la liste des points associés à la recommandation apparaît. L'utilisateur doit ensuite évaluer leur niveau de conformité. Dans le cas de la barre de navigation, voici la liste des points à respecter :

- Située en haut de l'écran juste en dessous de la barre de statut.
- Afficher le titre de la section courante.
- Afficher le bouton de retour à gauche avec le nom de la section précédente si l'on se trouve dans une sous-section.
- Le premier bouton d'action est à droite (un deuxième bouton peut être ajouté à gauche si l'application est non hiérarchique).
- Ne pas utiliser de fil d'Ariane (bread crumb).
- La taille de la barre est relative à l'orientation (éviter le codage en dur).
- En accord avec la barre d'outils et la barre de statut au niveau du style et de la couleur.

Pour chaque point, il faut affecter un niveau de conformité en fonction de ce qui est observé ou sinon ne pas prendre en compte le point dans l'évaluation. Pour aider l'utilisateur à faire son évaluation, un texte avec davantage d'informations apparaît lorsqu'il clique sur le bouton « plus d'infos ». Dans notre exemple, voici le contenu de ce texte :

> Pour la barre de navigation, il faut afficher le titre de la section courante et un bouton de retour avec le nom de la section précédente si l'on se trouve dans une sous-section. On peut éventuellement afficher un ou deux boutons d'action selon que l'on se trouve dans un modèle d'application hiérarchique ou non. Il faut cependant éviter l'utilisation d'un fil d'Ariane à cause des problèmes suivants :
>
> – La taille occupée ne laisse plus de place pour le titre.
>
> – Il n'est pas possible d'indiquer quel élément est sélectionné.
>
> – Plus il y a de liens et plus la zone qui peut être sélectionnée deviens petite.

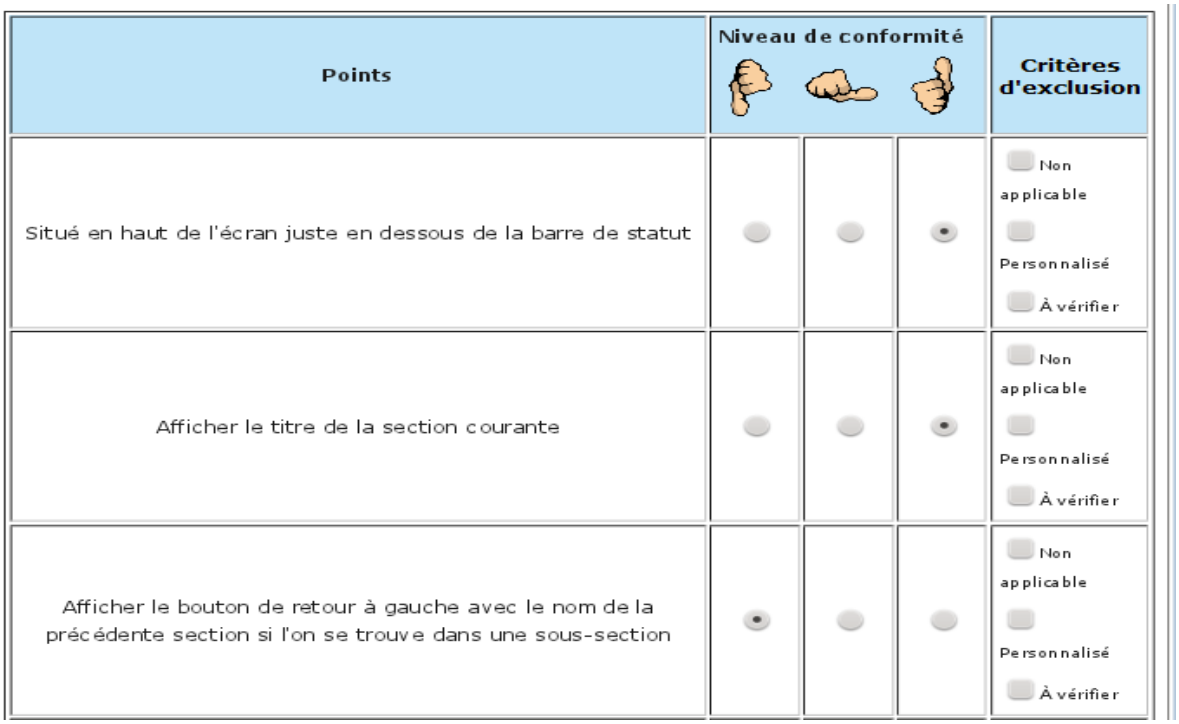
– Il peut devenir difficile de définir le niveau à afficher lorsque l'utilisateur navigue en profondeur dans la hiérarchie de l'interface.

Exceptionnellement, si l'application ne supporte pas la navigation hiérarchique, on peut utiliser, à la place réservée au bouton retour, un bouton d'action offrant une fonction s'appliquant à la section.

Il faut obligatoirement que la barre soit en accord avec les barres d'outils et de statut pour ce qui est de la couleur et du style. Il existe deux couleurs standards pour ces dernières qui sont le bleu, par défaut, ou bien le noir. Il est également possible d'utiliser la transparence, mais dans ce cas il ne faut pas utiliser une barre de statut noire et opaque. Il est cependant possible de le faire avec une barre de statut en gris.

4.3.3 Score

Dans l'exemple retenu, l'utilisateur évalue la maquette initiale (celle de gauche à la Figure 28) ainsi que la dernière maquette soumise (celle de droite à la Figure 28) quant à sa conformité avec les recommandations du fabricant de la plateforme (cf. Figure 33).



Points	Niveau de conformité			Critères d'exclusion
Situé en haut de l'écran juste en dessous de la barre de statut	○	○	●	Non applicable ☐ Personnalisé ☐ À vérifier
Afficher le titre de la section courante	○	○	●	Non applicable ☐ Personnalisé ☐ À vérifier
Afficher le bouton de retour à gauche avec le nom de la précédente section si l'on se trouve dans une sous-section	●	○	○	Non applicable ☐ Personnalisé ☐ À vérifier

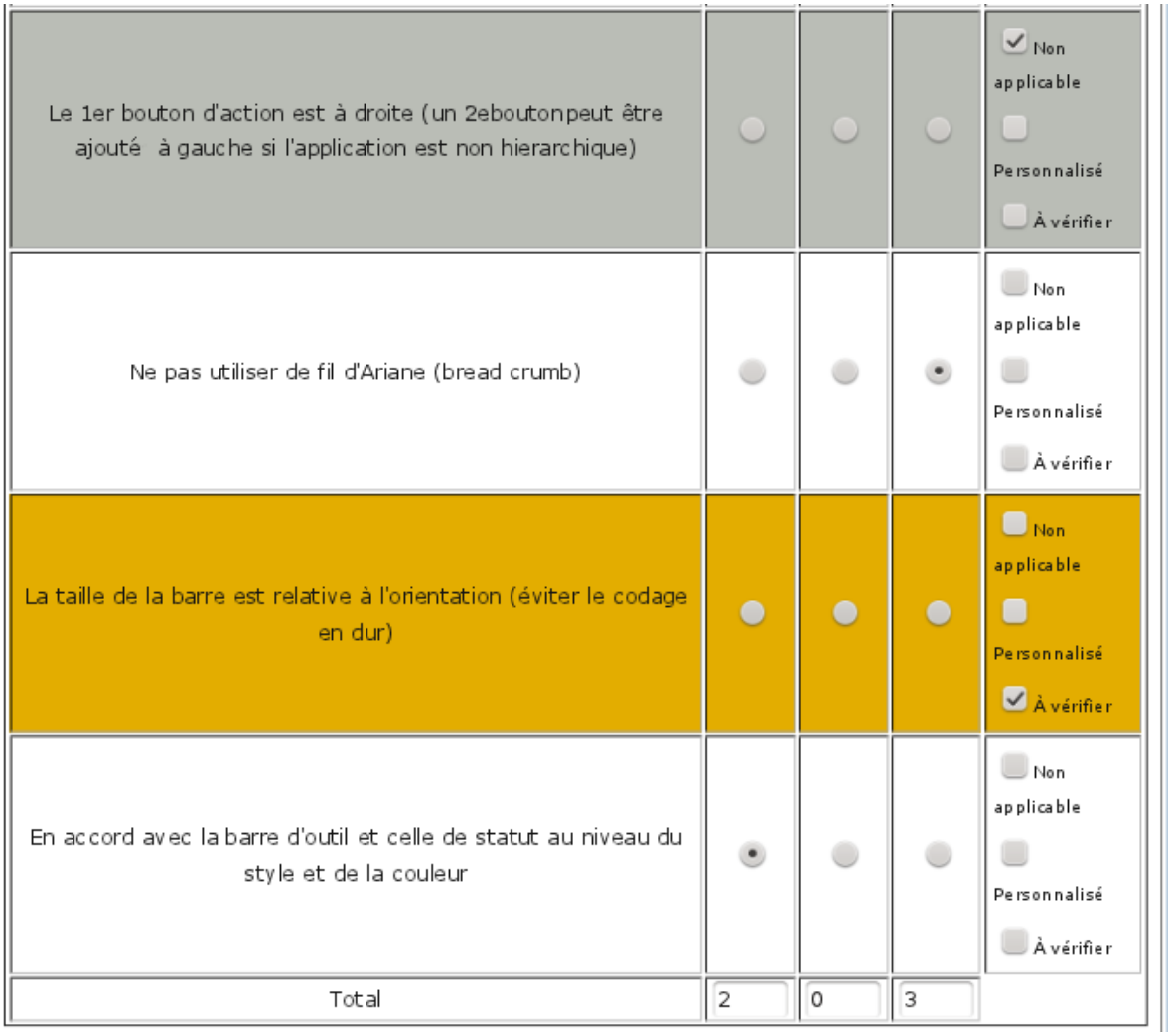


Figure 33: Capture d'écran du détails de l'évaluation de la maquette initiale

Si on considère seulement les points qui présentent une conformité parfaite (pouce vers le haut) on obtient un score de 60 %, soit 3 points conformes sur un total de 5 points. Par ailleurs, 40% des points sont non conformes, soit 2 points non conformes sur un total de 5 points, en l'occurrence l'affichage d'un bouton permettant le retour arrière et le choix des couleurs qui n'est pas en accord avec la couleur de la barre de statut. Dans cette évaluation, on n'a pas tenu compte de la position du premier bouton d'action étant donné qu'il n'y en a pas dans la maquette. Le point traitant de la taille de la barre doit aussi être vérifié ultérieurement auprès de l'équipe de développement. Après cette première évaluation, un certain nombre de raffinements ont été apportés à l'interface. Une nouvelle

évaluation a donc été réalisée avec la dernière maquette fournie afin de pouvoir comparer les scores et voir s'il y a eu amélioration (cf. Figure 34).

Points	Niveau de conformité			Critères d'exclusion
Situé en haut de l'écran juste en dessous de la barre de statut	○	○	◉	☐ Non applicable ☐ Personnalisé ☐ À vérifier
Afficher le titre de la section courante	○	○	◉	☐ Non applicable ☐ Personnalisé ☐ À vérifier
Afficher le bouton de retour à gauche avec le nom de la précédente section si l'on se trouve dans une sous-section	○	○	○	☐ Non applicable ☑ Personnalisé ☐ À vérifier
Le 1er bouton d'action est à droite (un 2e buton peut être ajouter à gauche si l'application est non hierarchique)	○	○	○	☑ Non applicable ☐ Personnalisé ☐ À vérifier

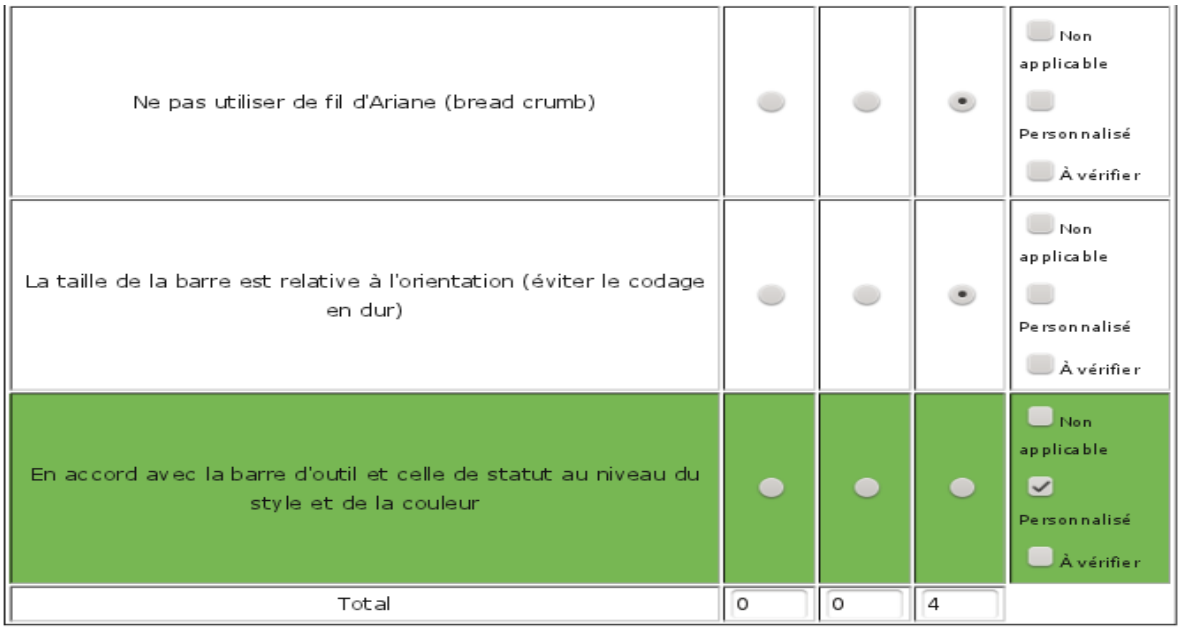


Figure 34: Capture d'écran du détails de l'évaluation de la dernière maquette soumise

Dans cette nouvelle évaluation, *Air Canada* a décidé de ne pas suivre les recommandations d'*Apple* pour le bouton de retour arrière. Elle a fait le choix d'utiliser un bouton de retour arrière personnalisé avec le logo de la compagnie. Il a également été décidé d'utiliser la couleur rouge pour la barre de navigation ce qui est une autre liberté par rapport aux recommandations du fabricant. Après vérification auprès d'*IBM,* il s'avère que la taille de la barre de navigation prend en considération l'orientation et n'est pas codée en dur. Au final, on obtient maintenant un score de 100 % (soit 4 points conformes sur un total de 4 points).

4.4 Conclusion

Ce chapitre a permis d'illustrer l'utilisation de l'outil d'évaluation de la conformité de la plateforme *iOS* aux normes du fabricant. L'implication de l'équipe mobile d'*Air Canada* dans le processus de conception ainsi que leur suggestions concernant différents aspects de l'interface ont permis d'obtenir un prototype qui permet d'évaluer rapidement le niveau de conformité des maquettes soumises par *IBM*. Néanmoins, afin de mieux mesurer l'apport de cet outil, nous avons réalisé quelques tests avec des utilisateurs finaux supplémentaires qui sont présentés au chapitre suivant.

CHAPITRE 5 : Test de l'outil d'évaluation auprès de quelques utilisateurs finaux

Ce chapitre présente d'abord la méthodologie mise en place afin d'évaluer le système de listes de vérification interactives que nous avons développé. Puis, il expose toutes les étapes du protocole expérimental ainsi que les données recueillies durant les évaluations. Enfin, les résultats obtenus sont présentés ainsi qu'une interprétation.

5.1 La méthodologie

Nous avons conçu une méthodologie permettant d'évaluer la qualité du système proposé. Celle-ci s'est faite en deux étapes : d'abord, en ayant recours à des tests d'utilisabilité qui permettent d'étudier la performance des sujets lors de la réalisation de certaines tâches typiques avec le système ; ensuite, en demandant aux usagers de remplir une grille de satisfaction permettant d'évaluer plusieurs aspects de l'interface et de connaître leur avis général sur le système développé.

5.1.1 Les tâches

On a défini sept tâches représentatives de l'utilisation du système développé que les sujets doivent réaliser, divisées en trois groupes. Le premier groupe compte trois tâches qui s'effectuent au sein de l'interface de consultation. Le deuxième groupe en compte deux qui font appel à l'interface d'administration. Enfin, le dernier groupe compte deux tâches qui font appel à la fois à l'interface de consultation et d'administration (cf. Tableau 4). Les tâches ont été conçues de manière à couvrir les différentes interfaces du système.

Groupe	Tâches	Scénario	Description
1	T1	**Évaluation** d'une maquette de la plateforme *iPhone*	Le sujet évalue (a) la barre d'onglets sur une maquette d'*IBM* et (b) le contrôle segmenté sur une autre.
	T2	**Évaluation** d'une maquette de la plateforme *BlackBerry*	Le sujet évalue (a) la barre de titre sur une maquette d'*IBM* et (b) la liste déroulante sur une autre.
	T3	**Recherche d'informations** sur une des catégories de l'image de marque	Le sujet doit trouver (a) l'espace minimal autorisé pour le logo horizontal et (b) la taille minimale autorisée pour la rondelle.
2	T4	Ajout d'un nouvel utilisateur	Dans l'interface d'administration le sujet doit ajouter un nouvel utilisateur.
	T5	Modifier le mot de passe d'un utilisateur	Dans l'interface d'administration le sujet doit modifier le mot de passe d'un utilisateur.
3	T6	Modifier le niveau de priorité d'une recommandation	Le sujet doit s'authentifier puis chercher une recommandation afin de modifier son niveau de priorité.
	T7	Modifier le numéro de version de la plateforme d'une recommandation	Le sujet doit s'authentifier puis changer le numéro de version d'une plateforme.

Tableau 4: Liste et description des tâches

Pour les tâches 1 et 2 des captures d'écran correspondant aux maquettes développées par *IBM* sont fournies et on demande au sujet d'évaluer un élément de l'interface en le lui montrant et en lui indiquant son appellation (cf. Figure 35 et Figure 36). Il faut ensuite que

le sujet recherche l'élément dans le système et l'évalue afin d'obtenir un score. Pour les autres tâches, on explique au sujet ce qu'il doit faire et, une fois que la tâche est bien comprise, on laisse le sujet la réaliser. Comme il ne s'agit pas ici de tâches d'évaluation, il n'y a pas de score associé.

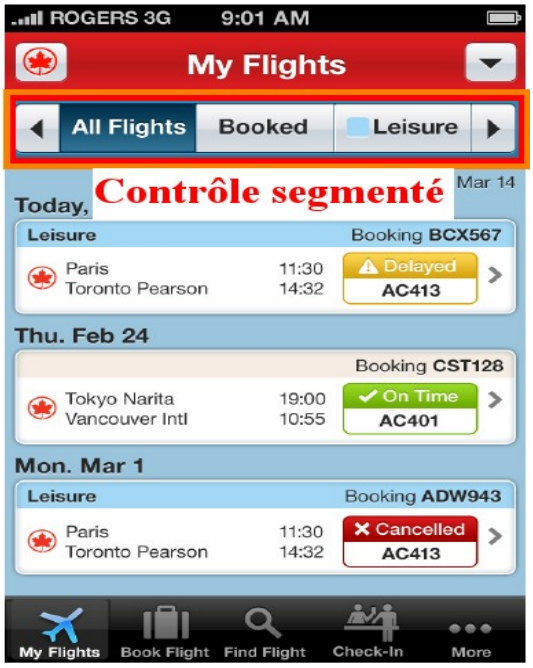


Figure 35: Maquette pour iPhone

Figure 36: Maquette pour BlackBerry

5.1.2 Choix des sujets

Les sujets ont été choisis parmi les six membres de l'équipe mobile d'*Air Canada*. Quatre personnes ont été retenues, soit deux hommes et deux femmes. Parmi les sujets choisis, il y en a deux que l'on peut qualifier d'experts parce qu'ils ont participé au développement du système et deux autres qui peuvent être qualifiés de novices, n'y ayant pas participé directement. Chacun des deux groupes de deux personnes est mixte et comprend une personne experte plus âgée et une personne novice plus jeune.

5.1.3 Tests d'utilisabilité

Chaque tâche est d'abord expliquée au sujet. On s'assure que le sujet a bien compris ce qu'il doit faire avant de démarrer le chronomètre. Afin de faciliter l'évaluation, on demande aux sujets de donner leurs observations à la fin de chaque tâche afin que l'on puisse noter leurs différentes remarques. Un guide a été utilisé afin de retranscrire les différentes remarques du sujet et certaines autres données comme le nombre de clics, la réussite ou non de la tâche et le type de cheminement utilisé (direct ou indirect). On dit qu'un chemin est direct s'il utilise le minimum de temps et de clics et se réalise au sein de l'interface associée à la tâche qui doit être réalisée. Tout chemin qui s'écarte du chemin direct ou qui fait appel à des stratégies différentes est considéré comme étant indirect. Enfin, on note si le sujet a eu besoin d'aide ou non afin de réaliser la tâche. Dans le cas des quatre tâches T1a, T1b, T2a et T2b, où le sujet doit évaluer des captures d'écrans de maquettes fournies par *IBM,* on indique également le score obtenu. Ces scores permettent de voir s'il y a un certain consensus entre les sujets au niveau de l'évaluation.

Avant de faire passer les tests aux sujets, une réunion a été organisée afin d'expliquer le déroulement de l'expérience et le type de données recueillies. On a aussi demandé aux participants de signer une entente de confidentialité dont un modèle peut être consulté à l'Annexe 3. Comme les tests ont été répartis entre le matin et l'après-midi, nous avons pris garde de former des groupes semblables afin d'annuler tout effet associé à l'ordre de passage des tests (cf. Tableau 5). Les mêmes précautions ont également été prises en ce qui concerne l'ordre des tâches à accomplir par les sujets, en les présentant dans un ordre aléatoire. À cet effet, les numéros de chacune des tâches ont été inscrits sur des morceaux de papier de même taille qui ont ensuite été retournés et mélangés. Puis, chaque sujet devait piger au hasard un morceau de papier qui définissait la tâche à accomplir.

Groupes	**Type**	**Genre**
Matin	Expert 1	Femme
	Novice 1	Homme
Après-midi	Expert 2	Homme
	Novice 2	Femme

Tableau 5: Répartition des sujets

5.1.4 Grille de satisfaction

Après les tests d'utilisabilité, nous avons mesuré le niveau de satisfaction des sujets par rapport à certaines caractéristiques du système. Pour cela, nous avons utilisé l'échelle WebPerform (Bressolles & Nantel 2007) qui a été adaptée à nos besoins en retirant les éléments d'évaluation qui ne se prêtaient pas à notre interface. L'échelle permet l'évaluation de sites Web et possède 9 caractéristiques réparties en 3 catégories : A- facilité d'utilisation et ergonomie ; B- qualité et quantité de l'information ; C- esthétique et design. Pour chacune des catégories, le sujet doit répondre à une série de questions à l'aide d'une échelle de Likert constituée de cinq points qui vont de 1 (« pas du tout d'accord ») à 5 (« tout à fait d'accord »). On obtient ainsi un score total sur 45 que l'on transforme en pourcentage afin d'obtenir un taux de satisfaction global pour chaque sujet.

Catégories	**Pas du tout d'accord**	**<====>**			**Tout à fait d'accord**
A - Facilité d'utilisation et ergonomie (score __/20)					
Ce site est facile à utiliser	1	2	3	4	5
Il est facile de chercher de l'information sur ce site	1	2	3	4	5
L'organisation et la mise en page de ce site facilitent la recherche d'informations	1	2	3	4	5
La mise en pages de ce site est claire et simple	1	2	3	4	5
B - Qualité et quantité de l'information (score __/10)					
L'information sur ce site est pertinente	1	2	3	4	5
L'information sur ce site est précise	1	2	3	4	5
C - Design et esthétique (score __/15)					
Ce site est joli	1	2	3	4	5
Ce site fait preuve de créativité	1	2	3	4	5
Ce site est visuellement attirant	1	2	3	4	5

Tableau 6: Grille de satisfaction adaptée de l'échelle Webperform

5.2 Limites de l'expérience

Bien évidemment le nombre de sujets pour cette étude est très limité. Il est clair qu'en ne testant que quatre sujets, il est impossible de procéder à des analyses statistiques poussées permettant de généraliser nos observations et de tirer des conclusions solides.

Cependant, il était difficile de faire mieux puisque l'équipe mobile d'*Air Canada* ne compte que six personnes ! De ce fait, nous avons également dû inclure des sujets ayant participé au développement du système, ce qui n'était évidemment pas souhaitable.

Il faut aussi noter que la solution a été développée pour être déployée sur un serveur distant. Cependant, certaines contraintes techniques liées à la politique de sécurité d'*Air Canada* ne nous ont pas permis de déployer la solution à temps pour la tester. Nous avons donc utilisé un ordinateur portable sur lequel le serveur web a été installé localement. De ce fait, les délais des réponse n'étaient pas représentatifs de ceux que l'on aurait obtenus sur un véritable serveur. Le même ordinateur a été utilisé pour chacun des participants et pour chacun des tests, ce qui ne reflète pas la variété de configurations d'ordinateurs et de fureteurs qui existent au sein d'*Air Canada.* Par contre, ce choix présente aussi l'avantage de standardiser les tests.

Une autre limitation importante de notre étude est l’absence de données de base. En effet, aucune donnée n'a été recueillie par *Air Canada* sur le temps que prend en moyenne l'évaluation des maquettes fournies par *IBM*. D'ailleurs, il n'existe même pas de protocole standard pour évaluer le travail fourni par le sous-traitant. On ne connaît donc pas le nombre d'étapes et encore moins le temps nécessaire pour procéder à l'évaluation d'une maquette. Pour obtenir de telles données, nous aurions dû étudier les pratiques de l'équipe mobile d'*Air Canada* sur un laps de temps beaucoup plus important que la durée impartie à ce mémoire. Il faut aussi dire que le déroulement des différents projets mis en place par l'équipe mobile est de durée très variable ce qui aurait sans doute rendu le recueil de données peu fiable sur une courte période.

5.3 Résultats et discussions

5.3.1 Tests d'utilisabilité

5.3.1.1 Résultats

Globalement, les deux experts on accordé les mêmes scores en ce qui a trait au niveau de conformité des maquettes (cf. Tableau 7).

Sujet	Groupe	Tâches	Temps de réalisation	Nombre de clics	Chemin	Aide	Score		
							☹	😐	☺
Expert 1	1	T1 a	1:11	12	Direct	Non	13%	0%	88%
		T1 b	1:52	8	Direct	Non	75%	0%	25%
		T2 a	2:46	11	Direct	Non	80%	0%	20%
		T2 b	2:30	14	Indirect	Non	0%	0%	100%
		T3 a	0:16	3	Direct	Non	-		
		T3 b	0:24	3	Direct	Non			
	2	T4	0:34	6	Direct	Non			
		T5	0:59	7	Direct	Non			
	3	T6	0:38	13	Direct	Non			
		T7	1:33	13	Direct	Non			
Expert 2	1	T1 a	1:54	15	Direct	Non	13%	0%	88%
		T1 b	0:52	11	Direct	Non	25%	25%	50%
		T2 a	1:37	8	Direct	Non	40%	40%	20%
		T2 b	3:04	18	Indirect	Non	0%	0%	100%
		T3 a	0:17	3	Direct	Non	-		
		T3 b	0:16	3	Direct	Non			
	2	T4	0:28	6	Direct	Non			
		T5	1:10	13	Direct	Oui			
	3	T6	1:44	8	Indirect	Non			
		T7	3:34	16	Indirect	Oui			

Tableau 7: Résultats des tests d'utilisabilité pour les experts

On remarque chez les sujets experts deux stratégies différentes lors de l'évaluation des maquettes. Le premier expert est assez catégorique et n'utilise pas du tout le niveau de conformité partielle (entre conforme et non conforme). À l'inverse, le second expert opte

parfois pour le niveau intermédiaire pour les points qui ne sont que partiellement respectés. Cette différence de comportement s'observe également avec le type de chemin utilisé. Celui-ci est le plus souvent direct pour le premier expert avec un temps de réponse assez court. Il faut dire que le premier expert a une meilleure connaissance des plateformes mobiles, étant donné qu'il a souvent la charge de tester les produits livrés par *IBM*. Il est intéressant de noter que les deux experts on eu du mal à trouver le groupe d'éléments d'interface auquel appartient la liste déroulante qui devait être évaluée pour la tâche T2b. Enfin, on peut également noter que pour les tâches T6 et T7, le second expert a préféré utiliser l'interface de gestion de la base de données présente dans la partie administration du système plutôt que d'utiliser les filtres présents dans l'interface de consultation.

En ce qui concerne les novices, ces derniers ont mis plus de temps à accomplir les tâches et ont également eu besoin d'aide plus souvent que les experts. L'un des novices a eu recours à une stratégie assez originale. Lorsqu'on lui demandait d'évaluer un composant de l'interface, ce dernier cliquait systématiquement sur tous les types de composants, puis se servait de la fonction de recherche du navigateur pour repérer une recommandation adéquate. Ici aussi, l'un des sujets a utilisé l'interface de gestion de la base de données, située dans la partie administration du système, plutôt que l'utilisation des filtres de la partie consultation pour les tâches T6 et T7. Contrairement aux experts, les novices ont souvent fait appel aux critères d'exclusion, lorsqu'ils étaient dans l'incertitude. Enfin, comme dans le premier groupe, on observe l'utilisation d'une stratégie de type binaire pour l'un des sujets (et pas pour l'autre) ainsi qu'un certain consensus au niveau des scores (cf. Tableau 8).

Sujet	Tâches	Temps de réalisation	Nombre de clics	Chemin	Aide	Score		
						☹	😐	☺
Novice 1	T1 a	3:21	16	Indirect	Oui	0%	0%	100%
	T1 b	2:30	16	Indirect	Non	0%	0%	100%
	T2 a	2:58	15	Indirect	Non	67%	0%	33%
	T2 b	2:06	16	Indirect	Non	0%	0%	100%
	T3 a	0:10	3	Direct	Non	-		
	T3 b	0:18	7	Direct	Non			
	T4	0:28	6	Direct	Non			
	T5	0:27	6	Direct	Non			
	T6	0:58	8	Indirect	Non			
	T7	6:12	24	Indirect	Oui			
Novice 2	T1 a	2:10	14	Direct	Non	0%	0%	100%
	T1 b	1:19	10	Direct	Non	0%	0%	100%
	T2 a	3:11	19	Indirect	Oui	60%	20%	20%
	T2 b	2:43	21	Direct	Non	0%	20%	80%
	T3 a	0:27	4	Direct	Non	-		
	T3 b	0:22	4	Direct	Non			
	T4	0:39	6	Direct	Non			
	T5	0:25	6	Direct	Non			
	T6	1:09	12	Direct	Non			
	T7	1:10	14	Direct	Non			

Tableau 8: Résultats des tests d'utilisabilité pour les novices

Il faut remarquer que les tests portaient sur des captures d'écran et non sur un produit fonctionnel, ce qui a causé des hésitations chez certains sujets, entraînant possiblement des perturbations dans les scores. Ce facteur s'est d'ailleurs révélé plus important chez les novices, étant donné que ces derniers ont souvent une moins bonne connaissance du fonctionnement des différentes plateformes mobiles.

5.3.1.2 Discussion

En comparant les temps requis par les experts et les novices, on remarque que les premiers sont généralement plus rapides, sauf pour les tâches T5 et T6 (cf. Figure 37). En fait, cette différence s'explique parce qu'il s'agit de tâches d'administration faisant appel à la création d'un identifiant ainsi que d'un mot de passe. Pour ce type de tâche, les experts ont

tendance à prendre plus de temps afin de ne pas faire d'erreurs et choisissent aussi un mot de passe plus complexe avec alternance de chiffres et de lettres. À l'inverse, les novices semblent être moins conscients des politiques de sécurité et ont tendance à choisir des mots de passe plus courts et constitués seulement de lettres.

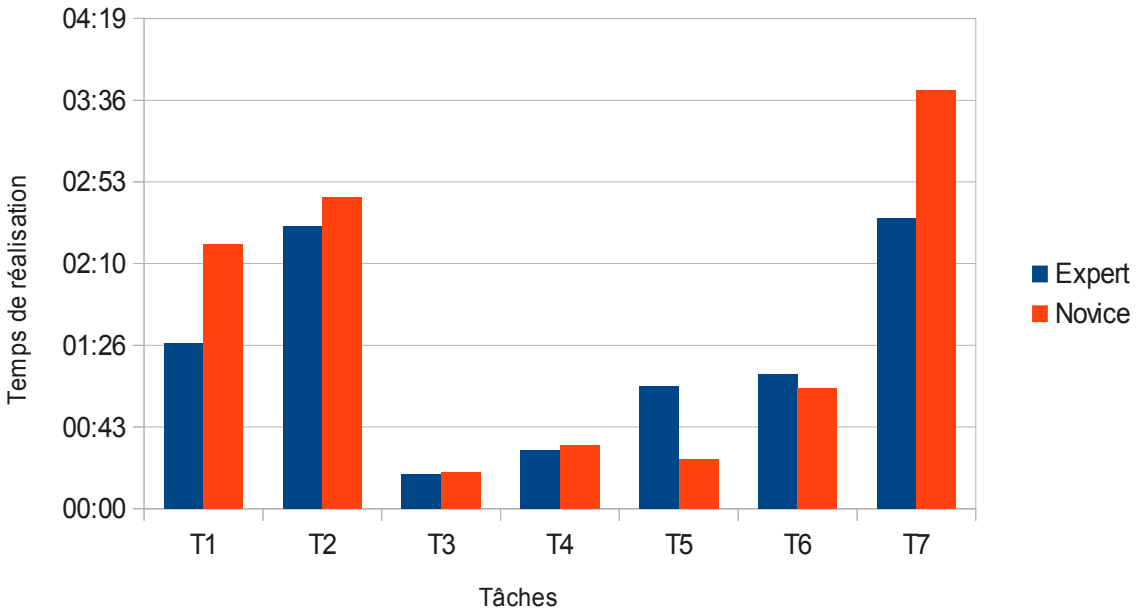


Figure 37: Temps nécessaire pour réaliser les tâches par les experts et novices

Enfin, lorsque l'on regroupe les tâches en fonction du type d'interface utilisé, soit celles de consultation (G1), d'administration (G2) ou les deux (G3), on observe que les tâches d'administration ont été effectuées dans un laps de temps très court (cf. Figure 38). Ceci s'explique en grande partie par la volonté de garder cette partie du système relativement simple et accessible à tous.

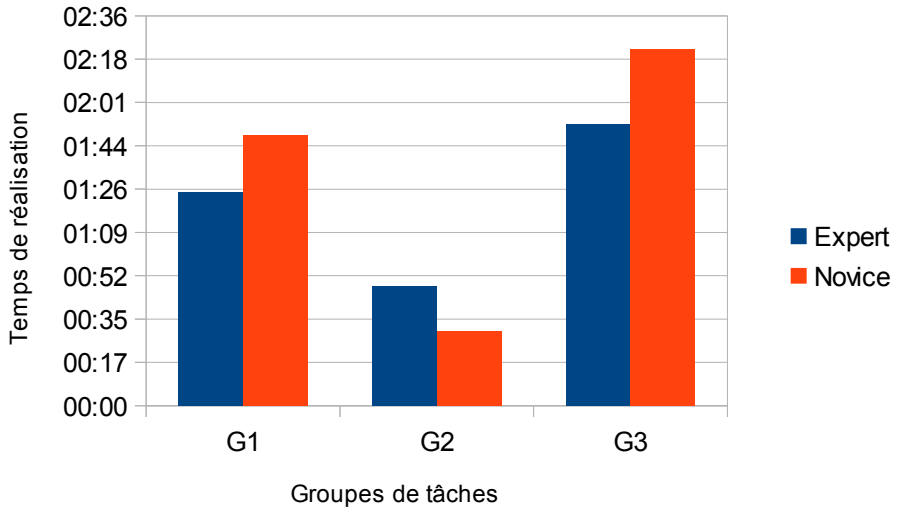


Figure 38: Temps de réalisation par groupes de tâches (selon le type d'interface) pour les experts et novices

5.3.2 Grille de satisfaction

5.3.2.1 Résultats

Globalement les utilisateurs se sont montrés assez satisfaits du système. En effet, on observe un degré de satisfaction globale de 84 %, avec un consensus sur la pertinence et la précision des informations contenues dans le système, ce qui se traduit par un score de 10/10 pour la catégorie B. Les sujets les plus critiques sont les experts, qui ont mis un score de 3/5 pour la facilité d'utilisation parce qu'ils estiment que des améliorations peuvent encore être apportées au système. Il en est de même en ce qui concerne le design et l'esthétique du système qui a obtenu un score légèrement inférieur à la moyenne (cf. Tableau 9).

Sujet	Catégories	Score	Total
Expert 1	A	15/20	82.22 %
	B	10/10	
	C	12/15	
Expert 2	A	15/20	80 %
	B	10/10	
	C	11/15	
Novice 1	A	16/20	83.35 %
	B	10/10	
	C	13/15	
Novice 2	A	17/20	88.89 %
	B	10/10	
	C	13/15	
Total	A	15.75/20	84.44 %
	B	10/10	
	C	12.25/15	

Tableau 9: Résultats obtenus avec la grille de satisfaction

Certaines critiques ont été recueillies lors d'une discussion qui a eu lieu après que les sujets aient rempli la grille de satisfaction. Cette discussion a permis de réaliser que les novices ont plutôt apprécié le design de l'interface du système. Ils ont mentionné que le site faisait preuve de créativité, ce qui s'est traduit par un score de 5/5 pour cette caractéristique. Bien qu'un des novices ait éprouvé certaines difficultés à rechercher de l'information dans le système, ce qui s'est traduit par un score de 3/5, il a tout de même mis 5/5 pour la clarté et la simplicité de la mise en page.

5.3.2.2 Discussion

Les résultats obtenus avec la grille de satisfaction mettent en valeur la qualité ainsi que la quantité des informations contenues dans le système, comme on témoigne les résultats pour la catégorie B (cf. Figure 39). Il faut également observer que le système développé dans le cadre de ce mémoire n'est qu'un prototype, ce qui explique les scores un peu plus bas accordés par les experts qui auraient aimé avoir une version plus aboutie au niveau du design (tel que mentionné dans la discussion qui a suivi). Il faut dire que cet aspect a été un peu négligé lors du développement du système. En effet, nous voulions

surtout privilégier la facilité d'utilisation ainsi que la qualité des informations contenues dans le système plutôt que son esthétique. Nous nous sommes tout de même assurés de respecter les recommandations d'*Air Canada* en matière de couleurs et de polices de caractères.

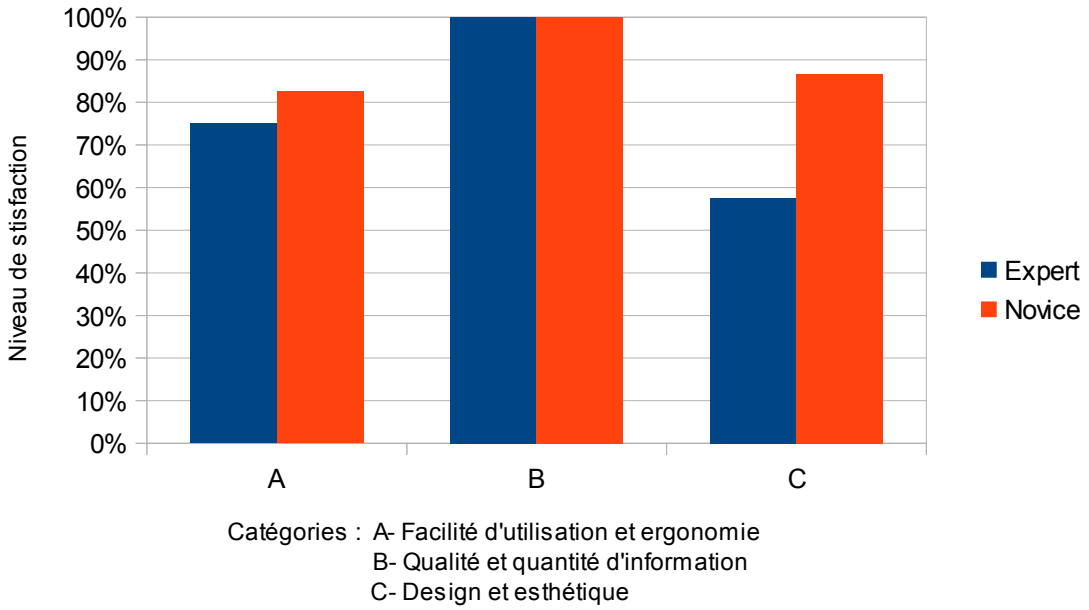


Figure 39: Résultat de la grille de satisfaction par facteur et type d'utilisateur

Les résultats obtenus indiquent que le système répond globalement aux attentes des utilisateurs, bien que des améliorations puissent certainement encore être apportées. Nous présentons justement certaines idées à ce sujet dans la conclusion qui suit.

CONCLUSION

Ce mémoire visait deux objectifs, à savoir :

- Mettre en place un outil permettant d'évaluer le niveau de conformité des maquettes fournies par un sous-traitant avec les recommandations émises par les fabricants de plateformes mobiles.

- Disposer d'une base de données permettant de comparer les plateformes mobiles en regard des différents composants de l'interface.

Pour atteindre ces objectifs, il d'abord fallu comprendre le fonctionnement de l'entreprise afin de lui proposer une solution permettant de répondre à ses besoins. Nous avons donc mis en place un processus itératif avec les responsables de l'équipe mobile pour développer une solution. Au fil des itérations, un système reposant sur des listes de vérification a vu le jour. Cependant, cette solution n'étant pas pleinement satisfaisante, nous avons introduit une part d'interactivité afin de pallier aux limites des approches traditionnelles.

Avantages

En comparant la situation au sein d'*Air Canada* avant et après la mise en place du système de listes de vérification interactives, une amélioration a été notée à plusieurs niveaux. D'abord, le système permet aux employés chargés de l'évaluation des maquettes de mener une évaluation de façon beaucoup plus systématique. De plus, en fournissant un score sur le degré de conformité d'une maquette par rapport aux recommandations émises par les fabricants, des données quantitatives peuvent être recueillies. L'outil accélère aussi le processus d'évaluation puisque les informations fournies par les fabricants des différentes plateformes mobiles, ainsi que les recommandations émises par l'entreprise en matière d'image de marque, sont maintenant centralisées. Les évaluateurs n'ont donc plus à chercher la documentation propre à chacune des plateformes mobiles ou contacter les départements de mercatique des différentes divisions d'*Air Canada* afin de récupérer la charte graphique appropriée. Évidemment, le fait que les listes de vérification sont

produites de façon dynamique en fonction de critères sélectionnés par l'utilisateur contribue aussi au gain d'efficacité. Malgré les efforts fournis, il reste encore quelques points faibles qui sont évoqués dans la suite.

Limites

Il faut d'abord noter que le développement de cet outil a été coûteux en temps. En effet, il a fallu parcourir la documentation fournie par les fabricants afin d'en extraire les points importants et ensuite entrer le tout dans le système. Il faut cependant noter que ces tâches ne doivent heureusement être effectuées qu'une seule fois. De plus, la pertinence des recommandations ergonomiques émises par les fabricants n'a pas été questionnée, faute de temps. Une autre faiblesse provient de la non intégration des images des maquettes fournis par le sous-traitant. Ce problème n'a pas été résolu dans le cadre de ce projet puisqu'aucune solution simple ne semblait adéquate. Enfin, l'impact du système sur le processus d'évaluation des maquettes devrait être évalué plus précisément. Pour cela, il faudrait toutefois une étude qui s'étend sur une période assez longue.

Développements futurs

Certaines améliorations et extensions au système actuel sont évidemment possibles. Un premier développement consisterait à étendre le système de session mis en place afin de conserver un historique des évaluations antérieures. Le système pourrait également gérer différents groupes d'utilisateurs afin de permettre au sous-traitant de soumettre directement de nouvelles maquettes qui pourraient ensuite être évaluées par l'équipe mobile, le tout à l'intérieur d'un même système. Une telle intégration pourrait aussi permettre à l'utilisateur d'identifier une région précise de l'image de la maquette sur laquelle porte son évaluation. Enfin, il serait pertinent de mettre en place une évaluation du système sur une période de trois à six mois afin d'observer les interactions des utilisateurs avec ce dernier. On obtiendrait ainsi des données quantitatives susceptibles de contribuer à l'amélioration globale du système. De telles améliorations pourront éventuellement être mises en place lors d'une prochaine version du système afin qu'il puisse mieux répondre aux attentes.

Au cours de ce travail, nous avons également réalisé tout le potentiel des listes de vérification interactives qui pourraient ainsi être appliquées à d'autres domaines. À cet effet, il serait souhaitable que les utilisateurs puissent définir leur propre typologie ou nomenclature pour les composants de l'interface et créer les formulaires appropriés afin de nourrir la base de données. Il ne resterait alors qu'à offrir une interface permettant de créer des filtres et d'afficher les résultats. Cette dernière étape pourrait d'ailleurs être automatisée étant donné que le système puise l'information dont il a besoin directement dans la base de données.

Bibliographie

1 Bressolles, G., & Nantel, J. (2007). Vers une typologie des sites Web destinés aux consommateurs. Revue française du Marketing, 213 (3), 41-56.

2 Frederick, G., & Rajesh L. (2009). "Chapter 1 - Introduction to Mobile Web Development". Beginning Smartphone Web Development: Building Javascript, CSS, HTML & Ajax-Based Applications for iPhone, Android, Palm Pre, Blackberry, Windows Mobile and Nokia S60. Apress.

3 Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria for evaluating usability evaluation methods. International Journal of Human-Computer Interaction, 13, 373–410.

4 Issott, A. J. (2008). "Chapter 1 - Introduction". Common Design Patterns for Symbian OS: The Foundations of Smartphone Software. John Wiley & Sons.

5 Jokela, T., Iivari, N., Matero, J., & Karukka, M. (2003). The standard of user-centered design and the standard definition of usability: Analyzing ISO 13407 against ISO 9241-11. Proceedings of the Latin American conference on Human-Computer Interaction 53–60. New York: ACM Press.

6 Kallio, T., & Kekalainen, A. (2004). Improving the effectiveness of mobile application design: User-pairs testing by non-professionals. Lectures Notes in Computer Science, 3160, 315–319.

7 Kjeldskov, J., & Graham, C. (2003). A review of mobile HCI research methods. Lecture Notes in Computer Science, 2795, 317–335.

8 Klockar, T., Carr D. A., Hedman, A., Johansson, T., & Bengtsson, F. (2003). Usability of mobile phones. Proceedings of the 19th International Symposium on Human Factors in Telecommucation, 197–204.

9 Patton M. Q., (1987). How to use qualitative methods in evaluation, Sage Publication, Inc, 40-43.

10 Sarkar, B. (2009). LWUIT 1.1 for Java ME Developers: Create Great User Interfaces for Mobile Devices. Chapter 1 - Introduction to LWUIT. Packt Publishing.

11 Scholtz, J. (2004). Usability evaluation, in Berskshire Encyclopedia of Human-Computer Interaction. W.S. Bainbridge (ed.), Berkshire Publishing Group, Great Barrington, MA, U.S.A.

12 Udell, Sterling. (2009). Pro Web Gadgets: Across iPhone, Android, Windows, Mac, iGoogle and More. Part 4 - Mobile Platforms Apress, New York, NY, U.S.A.

13 Vogel, D. & Baudisch, P. (2007). Shift: a technique for operating pen-based interface using touch. Proceeding of the SIGCHI conference on human factors in computing, San Jose, CA, U.S.A., 657-666.

14 http://www.oracle.com/technetwork/java/javame/tech/index.html

15 http://developer.symbian.org/main/documentation/index.php

16 http://developer.android.com/index.html

17 http://docs.blackberry.com/en/developers/?userType=21

18 http://developer.apple.com/iphone

19 http://developer.bada.com/apis/index.do

20 http://developer.limofoundation.org/

21 http://meego.com/developers

22 http://developer.palm.com/

23 http://wiki.openmoko.org

24 http://www.w3.org/Mobile/

25 http://mtld.mobi/

26 http://www.phonegap.com

27 http://www.airplaysdk.com/index.php

28 http://www.metismo.com/bedrock.html

29 http://www.afmm.fr

Annexes

Annexe 1 : L'univers des plateformes mobiles

Traditionnellement, les applications offertes par les appareils mobiles étaient dépendantes des fabricants, qui avaient pour mission de les développer et d'intégrer les applications. Les téléphones portables ne pouvaient pratiquement pas être personnalisés et il n'y avait que très peu d'applications extérieures disponibles. Cette situation a changé peu à peu avec l'apparition des interfaces de programmation (*Application Programming Interface* ou *API)* comme *Java Me*. Ces dernières ajoutent une certaine portabilité aux plateformes mobiles et simplifient le processus de développement pour les programmeurs. Avec l'apparition des téléphones intelligents, cette réalité s'est transformée en une multiplication des plateformes de développement. En effet, il existe actuellement quatre nouvelles API majeures en plus de *Java Me*, soit *iOS*, *BlackBerry OS*, *Symbian Plateform* et *Android*. En parallèle, le web mobile a fait son apparition grâce aux réseaux mobiles de plus en plus étendus et performants. Les technologies initiales étaient le WAP et le *i-mode*. Plus récemment, les possibilités se sont étendues avec l'amélioration des capacités des navigateurs mobiles (meilleur support du HTML, CSS et JavaScript) et le développement de nouveaux standards comme le HTML 5. Enfin, entre les applications embarquées et les applications distantes est apparue une approche hybride essayant d'exploiter les avantages des unes et des autres. Cette multitude de possibilités rend les choix difficiles. En effet, chaque plateforme a ses avantages et ses inconvénients qu'il convient de bien connaître afin de choisir celle qui est la plus appropriée.

Nous allons maintenant tenter de mettre en lumière les méthodes de conception d'interfaces utilisateurs pour plateformes mobiles munies de petits écrans, en présentant les trois types d'approches actuellement disponibles :

- Les applications natives qui sont embarquées sur les téléphones et développées avec une API spécifique.

- Le web mobile où les services sont offerts directement depuis des serveurs distants via Internet et utilisent les standards du web.

- La méthode hybride qui essaie de combiner les avantages des deux méthodes précédentes.

Pour chaque approche, nous allons présenter ses avantages et ses inconvénients en les illustrant par des technologies existantes sur le marché. Le but est d'offrir une vue globale afin d'orienter les chercheurs ou les développeurs vers la méthode de conception la mieux adaptée à leurs besoins.

A. Applications natives

Avec l'apparition ces dernières années de nouveaux compétiteurs comme *Apple* et *Google*, le paysage des plateformes mobiles s'est transformé en proposant une plus grande offre. Les magasins d'applications (*App Store*, *Ovi*, *App Word*, *Android Market Place*, …) représentent de nouveaux moyens de distribution pour les programmeurs et unifient la création de logiciels. Dans la suite, nous présentons un aperçu des technologies existantes.

Java ME

Développé par *Sun* en 1998, *Java Micro Edition* est l'une des plateformes de développement les plus anciennes. Elle se compose d'une *KVM* (*Kilobyte Virtual Machine*), une machine virtuelle capable d'exécuter une application écrite en Java, d'une « configuration » qui est en fait une API permettant d'accéder aux fonctions de base de l'appareil et d'un « profil », une autre *API* offrant l'accès aux fonctions spécifiques de la plateforme. La configuration la plus courante pour les téléphones cellulaires est appelée *CLDC* (*Connected Limited Device Configuration*). En ce qui concerne les profils, deux variantes existent : *MIDP* (*Mobile Information Device*

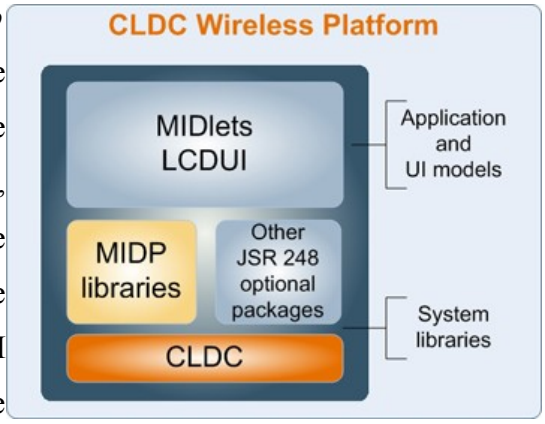


Figure 40: Architecture CLDC [14]

Profile) développé pour les téléphones équipés de *WAP* et *DoJa* (*DoCoMo Java*) réservé à ceux utilisant le i-mode. Le premier est le plus largement répandu, le second est quant à lui limité au Japon et à certains autres pays. Au-dessus de ces composants de base se trouvent les modèles d'interfaces utilisateurs (cf. Figure 40).

Actuellement une API appelée LWUIT (*Light Weight User Interface Toolkit*) est mise à la disposition des développeurs. Elle s'inspire de *Swing*, une autre librairie graphique conçue pour la programmation sur les appareils mobiles grand public. Elle permet de simplifier le travail des programmeurs en leur offrant un certain nombre de fonctionnalités. Par exemple, elle permet de ne plus tenir compte de la taille de l'écran de l'appareil pour lequel on souhaite développer une application, en intégrant tous les éléments d'interface nécessaires (boutons, champs de texte...). Cette API fonctionne selon le modèle MVC (*Model View Controller*) qui permet une séparation entre l'affichage et les fonctionnalités offertes (cf. Figure 41). À titre indicatif, voici une liste non exhaustive des principales capacités de LWUIT :

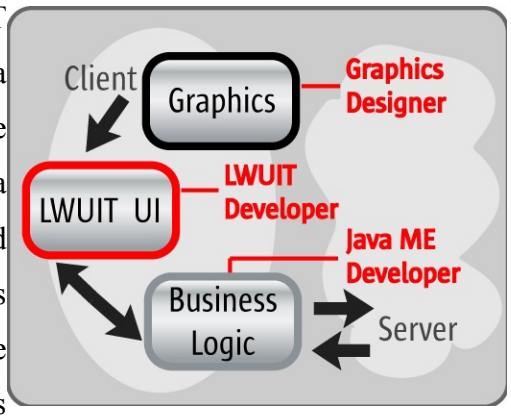


Figure 41: Place de LWUIT dans une architecture client-serveur [14]

- **Contrôle de l'interface utilisateur (IU)** avec des éléments comme les Arbres, les Tableaux, les Boutons, les Listes, etc. inspiré de la librairie Swing.
- **Support du XHTML** permettant le rendu de documents HTML conformes au standard XHTML Mobile Profile 1.0.
- **Gestionnaire de style** très flexible et offrant de puissantes fonctionnalités qui sont particulièrement intéressantes pour des applications s'exécutant sur différentes tailles d'écran.
- **Thèmes et apparence modifiables** grâce au kit de développement qui permet de créer des fichiers, similaires au feuillet de style CSS, pouvant être chargé ou changé lors de l'exécution et permettant de contrôler l'apparence des applications.
- **Polices de caractères** sous format bitmap ainsi que des outils permettant de créer sa propre police.

- **Écran tactile** que tous les éléments de LWUIT supportent. Ainsi, aucun développement supplémentaire n'est nécessaire pour qu'une application créée avec cette librairie fonctionne sur des appareils tactiles.
- **Support de clavier virtuel** permettant d'optimiser la saisie de données sur écran tactile.
- **Animations et transitions** sont inclues par défaut et permettent d'égayer les applications grâce à un large choix d'effets visuels.
- **Intégration de graphiques 3D et vectoriels (SVG)**
- **Des outils** pour créer des thèmes et d'autres ressources de personnalisation.
- **Support bidirectionnel du texte** permettant de gérer les langues se lisant de gauche à droite et de droite à gauche.

Au-dessus de l'interface graphique, on retrouve les algorithmes et toutes les fonctionnalités associées au logiciel. C'est ce qu'on appelle les *MIDlets* dans la terminologie *Java ME* (cf. Figure 40).

Afin de simplifier le développement, *Sun* offre une trousse de développement logiciels baptisée *Java Me SDK* offrant un environnement de développement ainsi qu'un émulateur permettant de tester ses applications comme sur un véritable appareil mobile. Il existe aussi de nombreux outils multiplateformes de développement libres et gratuits comme *Eclipse ME* pour *Eclipse* ou *Mobility* pour *Netbean* permettant de créer assez facilement des logiciels. La nature libre de *Java* ainsi que les nombreux logiciels permettant de développer des produits avec ce langage permettent d'atténuer grandement les obstacles au développement. Il n'est donc pas surprenant de constater que cette technologie se retrouve sur plus de 4,5 milliards d'appareils[14] dont 2,1 milliards sont des téléphones cellulaires et autres appareils portables. La portabilité et l'interaction avec les autres produits de *Sun-Oracle* sont des avantages indéniables pour cette plateforme. Quelques faiblesses viennent toutefois assombrir un peu le tableau. Tout d'abord, cette technologie est exclue des plateformes mobiles d'*Apple* comme l'*iPhone, l'iPod* ou le *iPad*. Ensuite, il est parfois difficile de porter des applications d'un appareil à un autre à cause de l'utilisation par certains fabricants de librairies spécifiques à leurs produits. Il faut également que

14 http://www.java.com/en/about/

l'application et ses données n'excèdent pas 1Mb pour être sûr de pouvoir exécuter le programme sur la plupart des téléphones. Cela est particulièrement vrai pour les vieux téléphones cellulaires possédant une ancienne version du MIDP. La nécessité d'une machine virtuelle pour que le langage *Java* puisse être interprété conduit aussi à certaines limitations au niveau des performances en matière de rapidité d'exécution et de démarrage ainsi que d'utilisation de la mémoire. En effet, les performances d'une machine virtuelle demeurent inférieures à celles d'un logiciel compilé, bien qu'elles soient tout de même supérieures à celles des langages interprétés comme *PHP*, *Ruby* ou *Python*.

Symbian Platform

Ce système d'exploitation est développé depuis 1998 par *Symbian Ldt*, un consortium initialement créé par *Psion, Nokia, Motorola, Ericsson et Panasonic/Matsushita.* En 2008, *Nokia* rachète la totalité des parts de ses partenaires et décide de libérer le code source du produit vers la fin de l'année 2009. Depuis, une organisation sans but lucratif, la *Symbian Fundation*, a été mise en place afin de centraliser les ressources et organiser le développement relatif à la *Symbian Platform.* L'architecture de la plateforme se divise en plusieurs niveaux (cf. Figure 42).

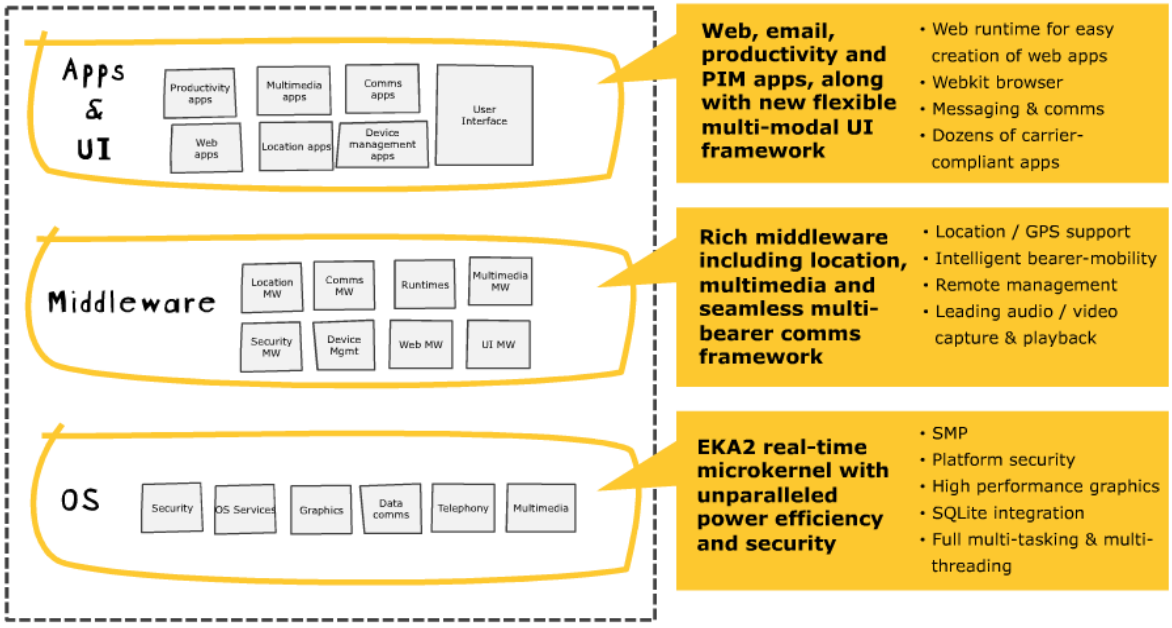


Figure 42: Architecture de la plateforme Symbian [15]

On retrouve d'abord la base qui est constituée d'un micro-noyau qui permet de gérer l'interaction avec le matériel. Ensuite, on retrouve l'environnement de programmation qui, dans le cas de *Symbian,* est très riche. En effet, la plateforme supporte *C, C++, Java ME, Python, Ruby, Flash Lite, Silverlight* ainsi que les technologies web. *Symbian* utilise un navigateur web qui repose sur le moteur de rendu *WebKit.* Il s'agit d'ailleurs de l'une des plus anciennes intégrations de cette technologie sur un appareil mobile. Enfin, *Symbian* possède une interface utilisateur initialement conçue pour être manipulée avec un clavier mais qui a depuis évolué en versions multiples : *S60* développée par *Nokia*, *UIQ* produit de *Sony Ericsson* et *Motorola* ainsi que *MOAP,* création de *NTT DoCoMo*. Avec l'ouverture de *Symbian*, une nouvelle interface baptisée *Orbit* a vu le jour, qui est basée sur la technologie *Qt* et offre une meilleure prise en charge des interfaces tactiles ainsi qu'un nouvel assortiment de transitions et d'animations.

L'organisation de l'interface utilisateur (IU) de *Symbian* (cf. Figure 43) repose sur le kit de composants logiciels (Framework) *Qt*. Originalement développée par la société *Trolltech*, il a été racheté depuis par *Nokia* et offre maintenant des composants d'interface graphique (widgets), l'accès aux données, la gestion des connexions réseaux et des fils d'exécution, l'analyse XML et bien d'autres facilités. L'un des principaux atouts de cette bibliothèque logicielle est sa portabilité, ce qui lui permet de fonctionner sur la majeure partie des systèmes d'exploitation. Au-dessus de *Qt, Symbian* offre un certain nombre de widgets permettant de faciliter le travail des développeurs en leur offrant des outils « clés en main » pour gérer certains aspects de l'interface utilisateur comme les interactions haptiques, l'aide contextuelle ou la reconnaissance vocale. Voici à titre indicatif une liste non exhaustive des possibilités offertes par *Symbian* :

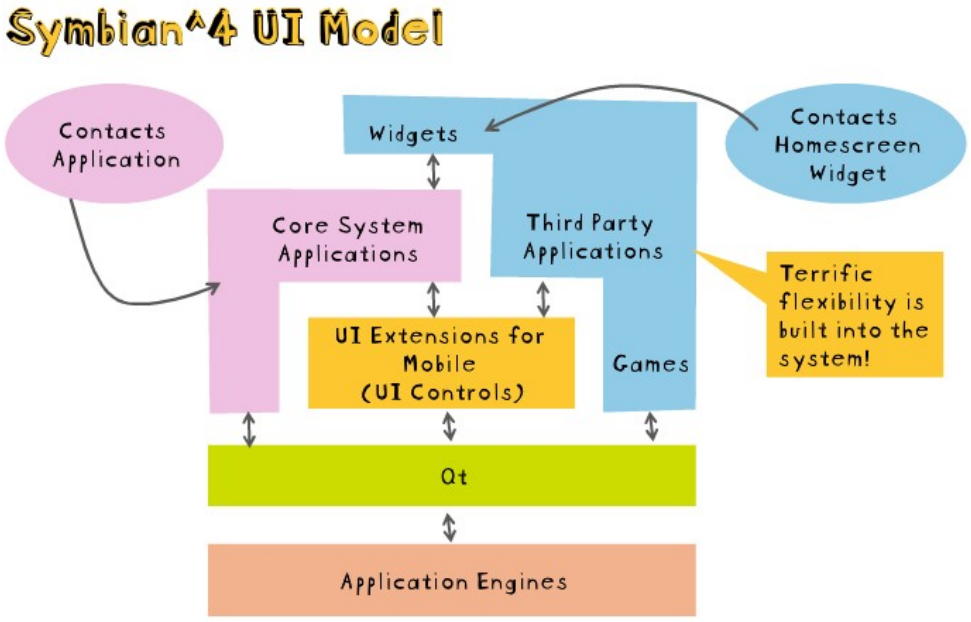


Figure 43: Organisation de l'IU sur Symbian [15]

- L'infrastructure de gestion d'interface utilisateur permet de gérer les fenêtres et l'organisation de l'affichage en offrant des outils de contrôle visuel.
- Des animations et des effets de transition sont disponibles via un ensemble de greffons standards.
- Le système gère les graphiques en 2D, 3D et format vectoriel (SVG).
- Gestion du texte et des polices de caractères.
- Plusieurs méthodes de saisie des données : Processeur de communication, reconnaissance vocale, reconnaissance des mouvements, saisie et contrôle des rétroactions issues des interfaces tactiles, des senseurs, des touches, des pointeurs et autres interfaces haptiques.
- Gestionnaire d'affichage pour le contrôle et la composition de l'affichage.

Il existe de nombreux outils de création de logiciels mis à la disposition des programmeurs. Ils se décomposent en trois types dépendant de ce que le développeur recherche. Il y a tout d'abord la trousse de développement permettant la création de logiciels via *Carbide.c++*, l'outil officiel de développement *C++* pour *Symbian,* ainsi que des scripts (*Build System*) permettant la compilation, l'empaquetage, les tests et le déploiement des programmes créés. Il est possible d'utiliser le SDK pour accéder aux

librairies et en-têtes nécessaires pour développer des applications *C++* pour *Symbian* et bénéficier d'un émulateur afin de tester son programme. La trousse offre également la possibilité d'accéder aux dernières spécifications de l'API public, permettant ainsi aux développeurs d'assurer la compatibilité de leur code sur une large gamme d'appareils *Symbian*. Enfin, une trousse de développement d'applications web permet de porter facilement ses créations internet sur un appareil mobile. Globalement, *Symbian Plateform,* qui possède déjà le plus grand nombre d'appareils fonctionnant avec son système d'exploitation, n'a plus besoin de faire ses preuves. L'API libre *Qt* est présente depuis de nombreuses années, elle est utilisée par des logiciels réputés avec une large communauté de développeurs, comme l'environnement de bureau *KDE* ou le lecteur multimédia *VLC*. Les qualités de *Qt* renforcent ainsi la position de *Symbian*. De plus, avec son code source qui est maintenant libre, la transition vers cette plateforme est facilitée.

Malgré une présence de longue date dans le marché, la récente ouverture du produit suscite un certain nombre de questions. La documentation, par exemple, n'est pas encore bien établie et elle n'est disponible qu'en anglais. Bien que la plateforme y soit assez bien décrite, la cohabitation de différentes versions (déjà existantes ou en cours de développement) rend parfois la lecture difficile. Étant donné que la *Symbian Fundation* débute dans la gestion de projets libres, une certaine inertie au sein de la communauté pourrait être observée. En effet, tout projet d'envergure doit être motivant pour les développeurs. De plus, il faut savoir garder le cap tout en étant ouvert aux suggestions et aux doléances de la communauté. Il faudra sûrement un peu de temps avant qu'une telle dynamique ne se crée.

Android

Nouvel arrivant sur le marché des appareils mobiles, *Android* a su rapidement se tailler une part de marché non négligeable. Ce système d'exploitation basé sur le noyau *Linux* a été originalement développé par *Android, Inc.* puis racheté en juillet 2005 par *Google*. Le 5 novembre 2007, le consortium baptisé *Open Handset Alliance* (*OHA*) voit le jour. Il rassemble des opérateurs comme *NTT DoCoMo*, *Sprint Nextel*, *Telecom Italia* ou

Bouygues Telecom, des équipementiers tels que *Samsung*, *Motorola*, *LG Electronics*, *Intel* ou *Nvidia* et enfin des incontournables de l'Internet dont *eBay*. *Google* décide de libérer l'intégralité du code source d'*Android* le 21 octobre 2008 et appelle la communauté à venir participer au projet en lançant une compétition (*Android Developer Challenge)* avec des prix pour les projets les plus prometteurs.

L'architecture de la plateforme se divise en plusieurs niveaux (cf. Figure 44). On retrouve à la base le noyau *Linux*, qui est chargé de fournir les services de base de l'appareil et de faire le lien entre le matériel et le reste des couches logicielles. On retrouve ensuite l'environnement d'exécution *Android* reposant sur un ensemble de librairies offrant des fonctionnalités similaires à celles offertes par *Java*, ainsi que la machine virtuelle *Dalvik,* optimisée pour répondre aux contraintes de l'informatique mobile. La technologie est compatible avec *Sun-Oracle* ce qui permet aux programmes compilés avec leur machine virtuelle de fonctionner sur *Dalyik* moyennant quelques petites modifications. La plateforme inclut un ensemble de librairies offrant différents services comme la lecture de flux multimédia, le rendu des polices de caractères ou encore la gestion de bases de données. Enfin, on retrouve un kit de composants logiciels qui fournit une base homogène aux différents programmes, leur permettant de réutiliser des services et d'échanger facilement des informations.

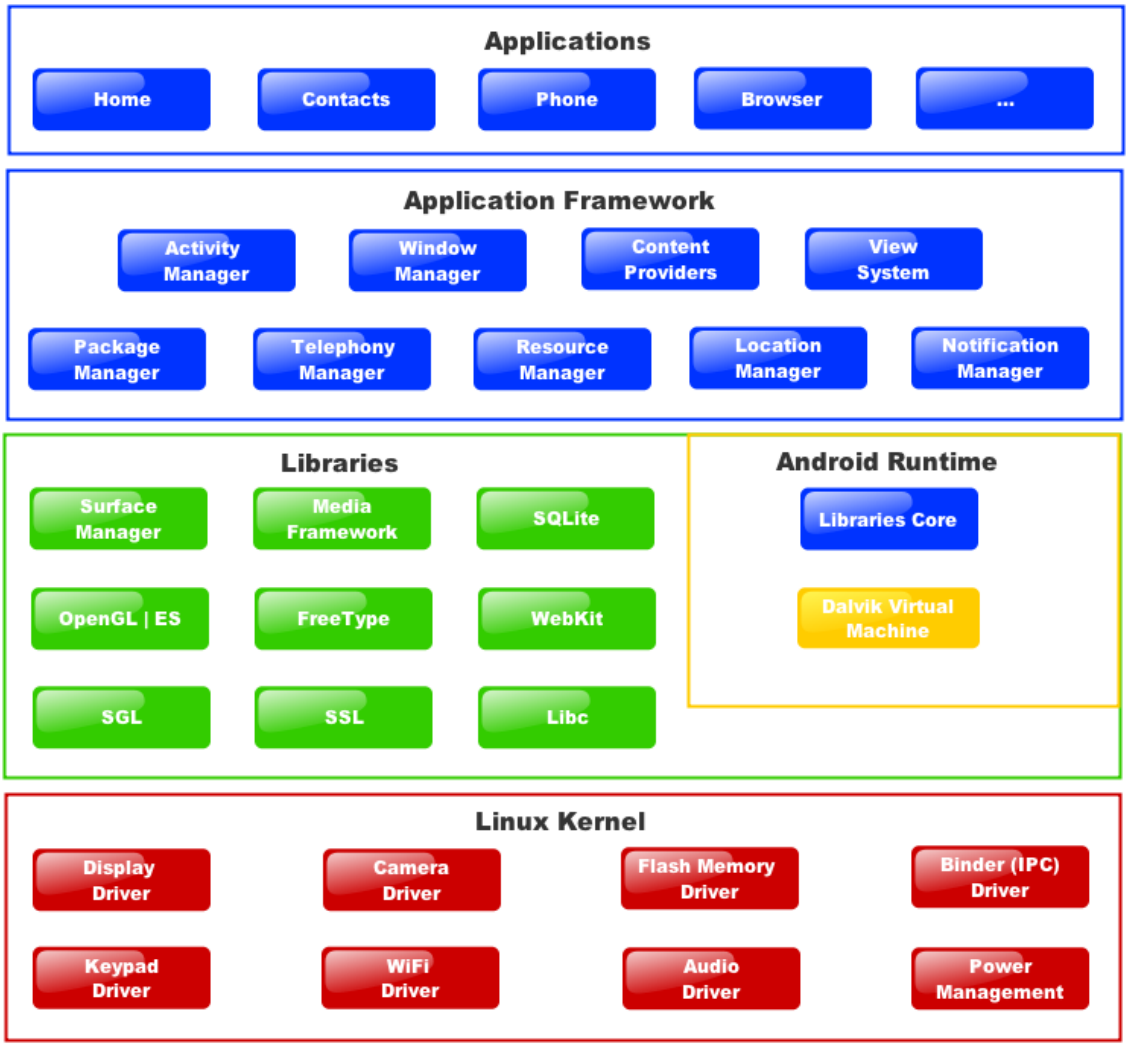


Figure 44: Architecture de la plateforme Android [16]

L'interface utilisateur d'*Android* se divise en plusieurs catégories, chacune répondant à un besoin particulier (cf. Figure 45). On retrouve ainsi les groupes de vues (*GroupView*) et les vues (*View*) qui sont les éléments centraux de l'interface. Une vue est une entité pouvant contenir des widgets, qui sont des éléments d'interaction comme un bouton ou un champ de texte. Le couple vue et widgets produit une mise en page (*layout*) qui définit la façon dont les éléments sont affichés. Enfin, les différents *layouts* peuvent être rassemblés dans un groupe de vues.

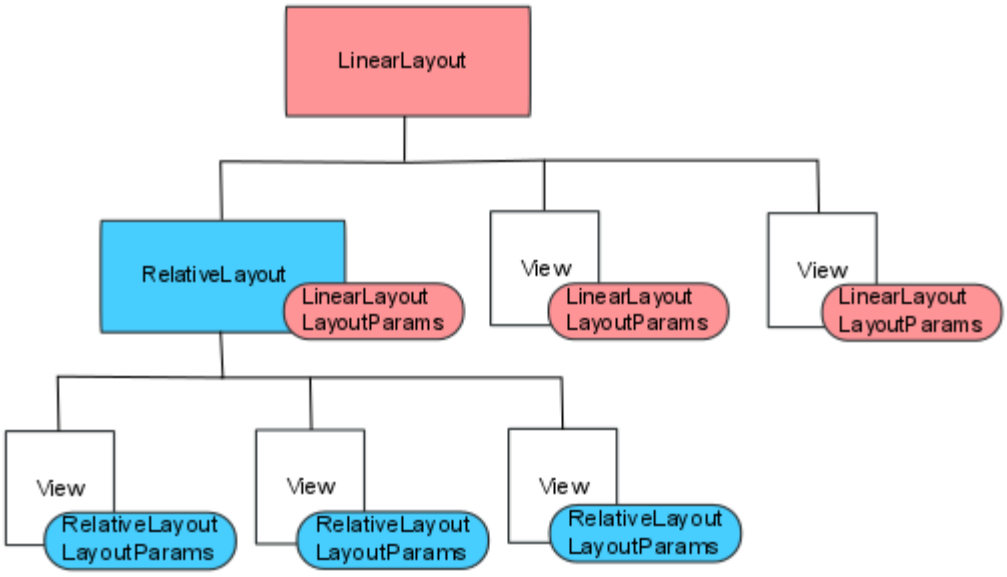


Figure 45: Hiérarchie de l'IU d'Android [16]

Les autres éléments clés de l'interface sont :

- Les événements d'interaction (*UIevents*) permettant de connaître l'état des différents widgets lorsque l'utilisateur interagit avec ceux-ci.

- Les menus qui gèrent les éléments à afficher lorsque l'utilisateur utilise le bouton menu ou lorsqu'il appuie pendant un certain temps sur un élément de l'interface.

- Les adaptateurs, qui sont des éléments permettant d'adapter l'affichage des données extérieures en fonction d'un format donné.

- Les styles et thèmes qui sont des éléments de personnalisation de l'interface. Ils permettent aux développeurs de redéfinir l'apparence des widgets et de les organiser afin de pouvoir les réutiliser plus facilement.

En plus d'offrir une organisation rigoureuse de l'interface graphique, *Android* met à la disposition des développeurs un certain nombre de guides et de recommandations en matière d'interface utilisateur. On y retrouve les pratiques à suivre lors de la création d'icônes, de widgets, de notifications ou de menus. Ces règles permettent aux applications d'avoir un aspect homogène et facilitent leur intégration.

Android met à la disposition des développeurs un SDK qui roule sous *Windows*, *Mac OS* et *Linux*. La trousse de développement contient un certain nombre d'outils, dont un greffon permettant de l'intégrer à la plateforme de développement *Eclipse*. La trousse contient aussi un émulateur afin de faciliter le débogage, de la documentation, des exemples ainsi que les API de *Google* pour faciliter l'intégration de la plateforme avec les services en ligne proposés. Il est certain que le soutien d'un géant comme *Google* confère à la plateforme une certaine prestance et lui a permis de s'imposer. L'entreprise de Mountain View déjà très impliquée dans le domaine des logiciels libres a su rassembler un nombre important de développeurs et commence peu à peu à rattraper son concurrent iOS au niveau du nombre d'applications disponibles. L'*Android Marketplace* possède déjà 70 000 applications et compte plus d'un milliard de téléchargements[15] ce qui prouve le succès de cette plateforme de distribution numérique beaucoup moins contraignante que celle d'*Apple*.

Les récentes accusations de *Sun-Oracle* envers *Google* en matière de violation de brevet concernant la technologie *Java* laissent toutefois planer un doute sur la pérennité de cette plateforme. De plus, l'échec commercial du téléphone *Nexus One* de *Google* est peut-être le signe d'un manque de savoir-faire dans le domaine, ce qui est compréhensible vu le jeune âge de la plateforme. Une des craintes formulées par les détracteurs d'*Android* est la fragmentation de son interface. En effet, il existe à l'heure actuelle pas moins de 5 variantes[16] : *Motoblur de Motorola*, *Sense* de *HTC*, *TouchWiz* de *Samsung* ainsi que *Streak* de *Dell* et *Xperia* de *Sony Ericsson*. Cette hétérogénéité peut confondre l'utilisateur. De plus, le temps nécessaire pour développer une interface peut mener, comme dans le cas de *Dell* et *Sony Ericssson*, à produire des appareils avec une version plus ancienne d'*Android*. Il est donc important qu'une politique soit mise en place à cet effet.

Les autres plateformes

15 http://www.engadget.com/2010/07/15/android-market-now-has-100-000-apps-passes-1-billion-download-m/

16 http://blogs.computerworld.com/16755/android_user_interface

Bien que faiblement présentes sur le marché, d'autres plateformes peuvent tout de même constituer des alternatives dans certains cas de figure et méritent qu'on surveille leur développement. Certaines des plateformes présentées ci-dessous sont d'ailleurs soutenues par de grandes compagnies.

Bada

Ce système d'exploitation mobile, développé par *Samsung*, s'adresse aux téléphones intelligents haut de gamme et de gamme moyenne de la compagnie coréenne. Annoncé le 10 novembre 2009, il propose un catalogue de 920 applications (en date du 8 septembre 2010) et se propose d'offrir plus de 7 000 logiciels d'ici la fin de l'année 2010. Le développement sur cette plateforme est possible grâce au kit de développement fourni par *Samsung* qui fait appel au langage C++ et repose sur une architecture en couches (cf. Figure 46). Pour le moment, un seul téléphone intelligent utilise ce système, soit le *Wave S8500*, un appareil à écran tactile reposant sur l'architecture ARM et le moteur graphique intégré PowerVR SGX 3D capable d'afficher et d'enregistrer des contenus HD à 30 images/seconde. Il est encore trop tôt pour se prononcer sur le sort de cette plateforme. Cependant, certaines critiques apparaissent déjà à propos de la nature fermée de l'API de senseur externe qui rend impossible l'ajout de nouveaux senseurs ou l'ajout de nouvelles fonctionnalités à celles déjà présentes. De plus, les applications *Bada* ne peuvent accéder à la boite de réception SMS/MMS et recevoir des notifications pour ce type de messages. Enfin, il semble que le multitâches ne soit disponible qu'avec les applications natives fournies avec le téléphone.

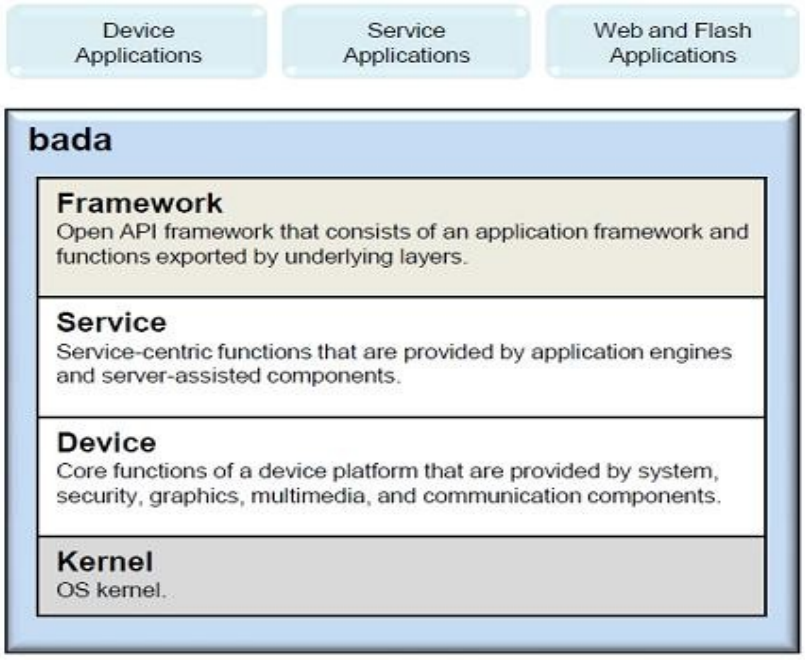


Figure 46: Architecture de la plateforme Bada [19]

LiMo (Linux Mobile)

Cette plateforme a été développée par la *LiMo Foundation*, organisation fondée en janvier 2007 par *Motorola, NEC, NTT DoCoMo, Panasonic Mobile Communications, Samsung Electronics* et *Vodafone*. *Motorola* s'est retirée depuis pour rejoindre *Android*, mais la fondation a recruté d'autres membres et compte actuellement une cinquantaine de participants. La plateforme n'est pas libre, mais repose sur des technologies libres (cf. Figure 47). Son kit de développement permet de créer des logiciels en *Java*, en *C* ou alors en exploitant les technologies web. Son API graphique repose sur la librairie *GTK+*. Une quarantaine d'appareils, répartis entre *Vodafone*, *Motorola*, *NEC, Panasonic* et *Samsung* utilisent *LiMo*. La documentation relative au développement sur cette plateforme est très limitée et il ne semble pas exister de standard pour la distribution des applications.

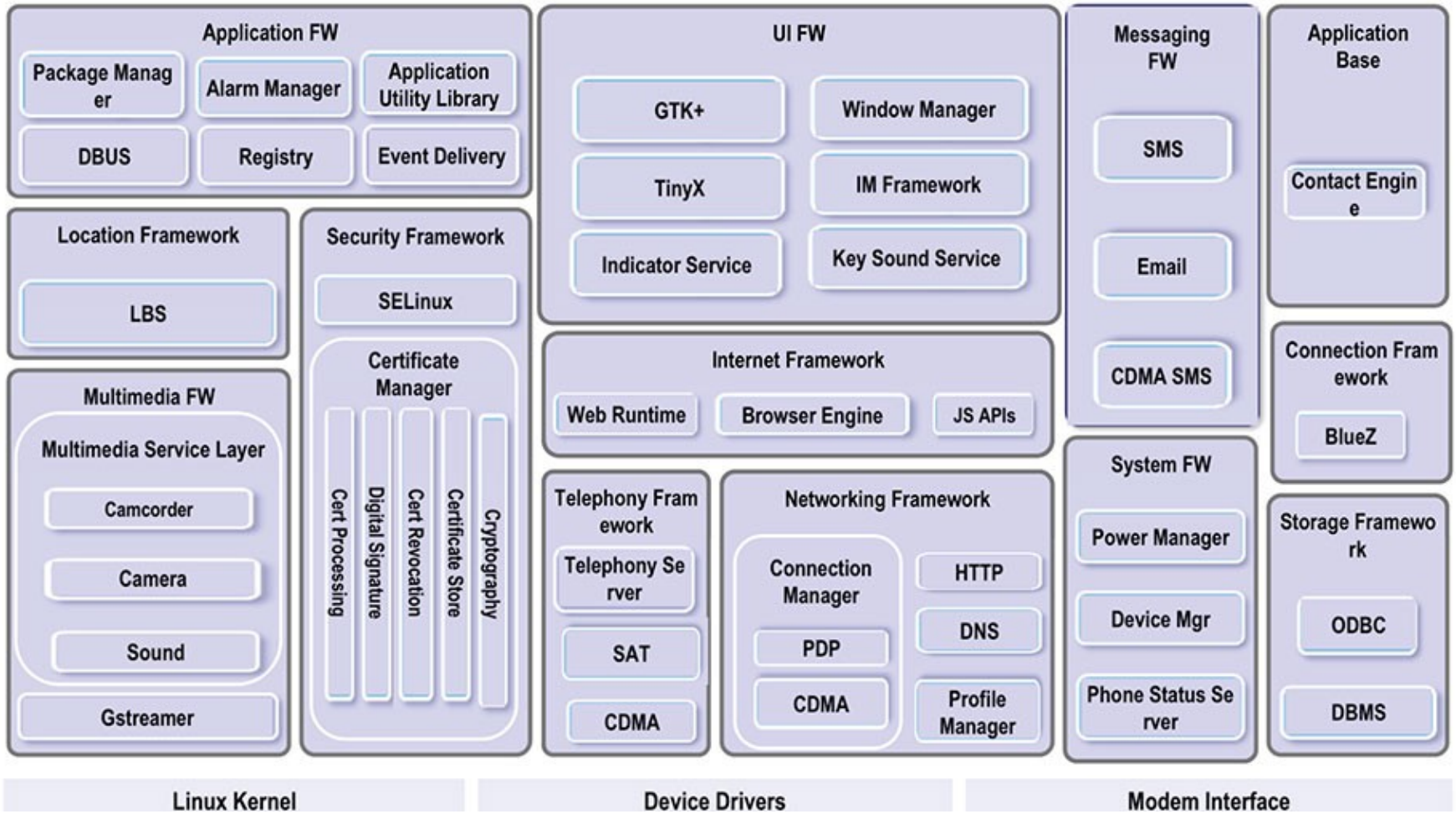


Figure 47: Architecture de la plateforme LiMo [20]

MeeGo

Né de la fusion de deux projets similaires, soit *Moblin* mené par *Intel* et *Maemo* mené par *Nokia*, *MeeGo* est un système d'exploitation libre reposant sur le noyau *Linux* et dont la vocation est d'être compatible avec différentes plateformes mobiles, allant des miniportatifs, aux téléphones intelligents, en passant par les tablettes et les ordinateurs dans les voitures. La plateforme repose sur les librairies graphiques *Qt* et *GTK+/Clutter* et propose deux systèmes de distribution *AppUp* pour *Intel* et *Ovi* pour *Nokia*. L'architecture de *MeeGo* se divise en plusieurs niveaux, chacun apportant un ensemble de services (cf. Figure 48). Un kit de développement multiplateformes est mis à la disposition des programmeurs. Il n'existe pour le moment que deux téléphones intelligents capables de fonctionner avec *MeeGo*, soit le *Nokia N900* et le *Aava Mobile* basé sur la plateforme *Intel Moorestown*.

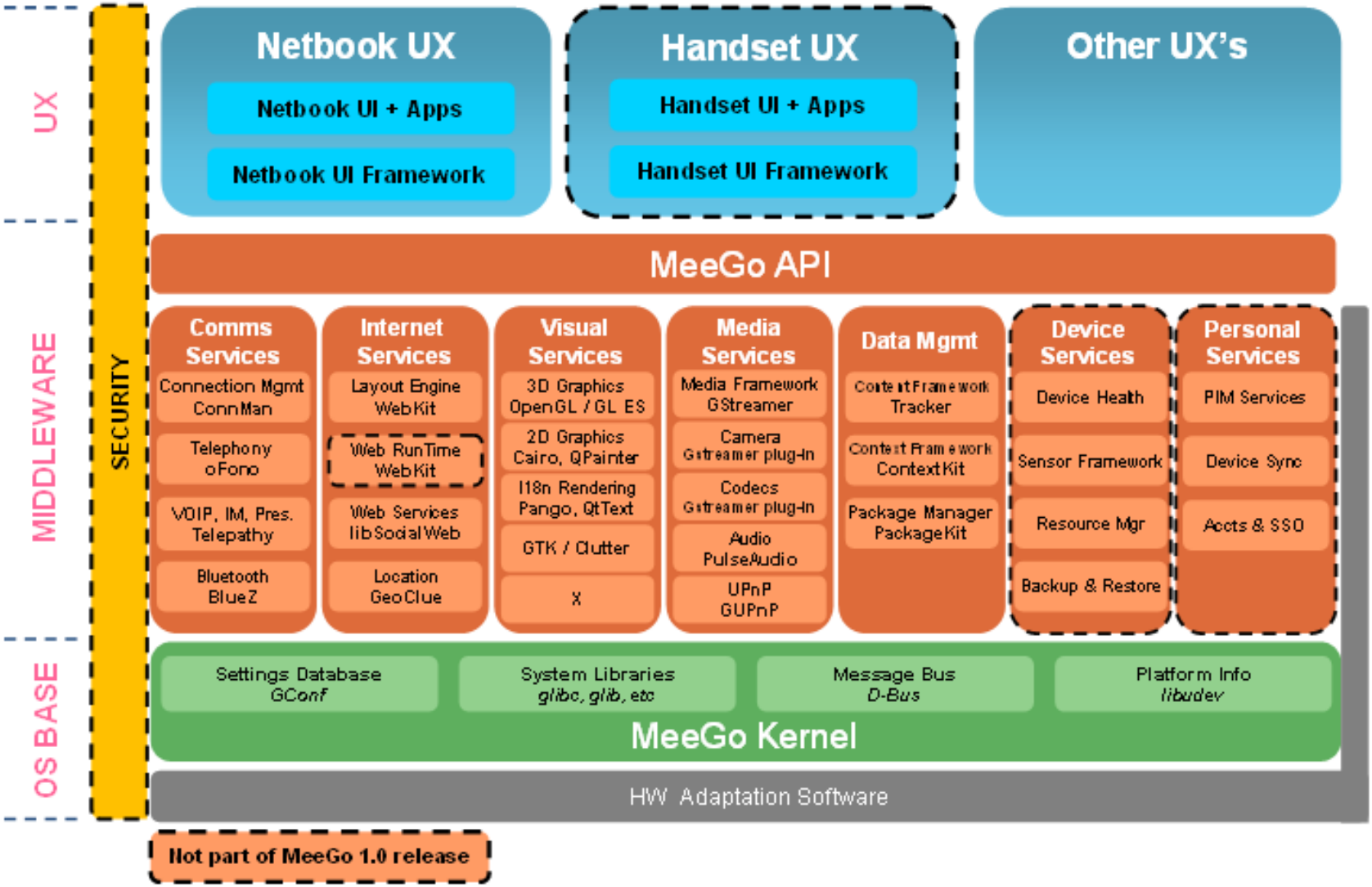


Figure 48: Architecture de la plateforme MeeGo [21]

WebOS

WebOS est un produit de la firme *Palm* principalement connue pour ses assistants électroniques de poche (PDA) et qui a été récemment rachetée par le géant *HP*. Il s'agit d'un système d'exploitation propriétaire reposant sur le noyau *Linux*. Ce système s'appuie sur les technologies web (*HTML 5*, *JavaScript*, *XHTML*, *CSS* et *JSON)* pour le développement et ajoute un certain nombre d'APIs afin d'étendre, par exemple, les capacités de *JavaScript* pour pouvoir accéder aux différents éléments de la plateforme (cf. Figure 49). *Palm* a publié un kit de développement baptisé *Mojo*, grâce auquel il est possible d'accéder directement au mode développeur depuis l'appareil en saisissant un code sur le clavier du téléphone. La plateforme possède son propre système de distribution d'applications appelé *App Catalog*. Il existe deux téléphones actuellement disponibles avec *WebOS*, soit le *Palm Pré* et le *Palm Pixi*.

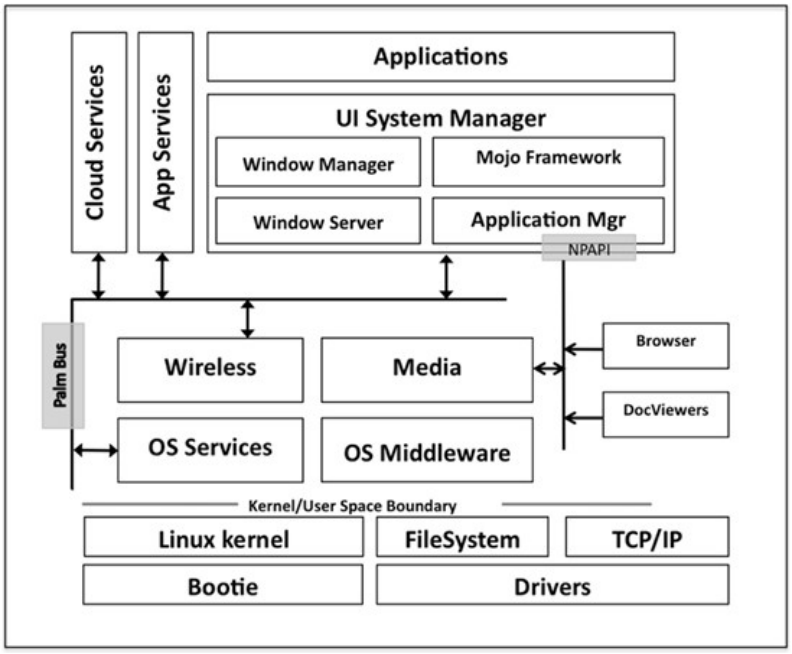


Figure 49: Architecture de la plateforme WebOS [22]

Openmoko

Ce système d'exploitation de la compagnie éponyme permet de faire fonctionner son appareil *Neo Freerunner*. Cette entreprise adopte une approche originale en offrant

l'intégralité de ses programmes sources ainsi que les spécifications matérielles. L'idée est de créer une réelle synergie avec la communauté des développeurs. Ainsi, tous sont invités à se joindre à l'un des projets en cours afin d'y apporter sa contribution au niveau logiciel ou matériel. Pour le moment, la plateforme en est encore au stade expérimental, mais elle offre des perspectives intéressantes (cf. Figure 50). Bien que disponible seulement sur l'appareil développé par la compagnie, la migration du système vers d'autres téléphones a été entrepris (pour l'instant, avec plus au moins de succès). L'environnement de développement est essentiellement basé sur la plateforme *GNU/Linux*. Il est cependant prévu de l'étendre à d'autres plateformes.

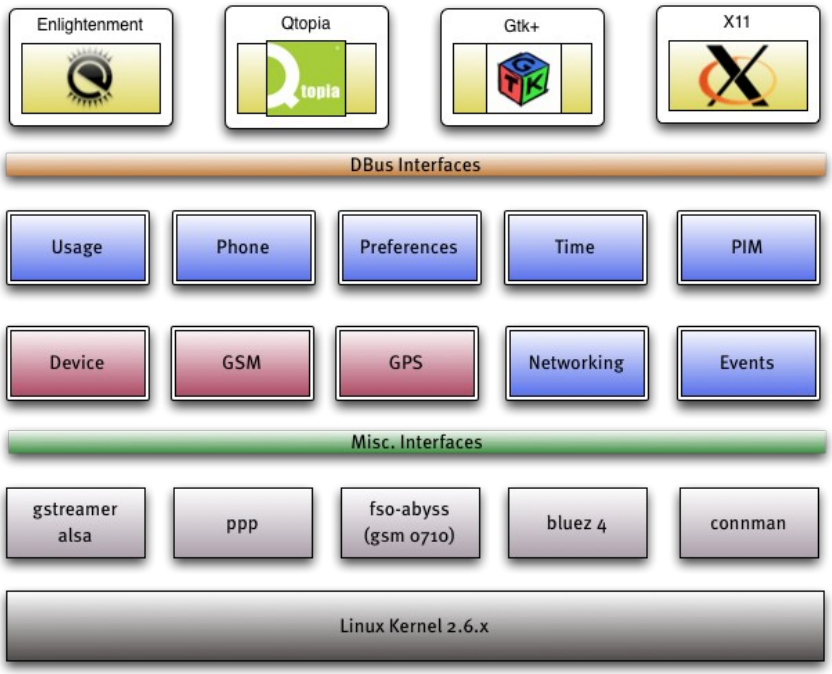


Figure 50: Architecture de la plateforme Openmoko [23]

B. Web mobile

Les technologies web sont de mieux en mieux supportées par les appareils mobiles et permettent d'offrir une expérience de plus en plus riche aux utilisateurs. Cependant, à cause de la taille limitée des écrans et des périphériques d'entrée, il est nécessaire d'adapter le contenu des sites web. Comme les technologies web doivent accommoder une large gamme d'appareils, elles ne tiennent pas forcément compte des spécificités propres à chaque

plateforme. Or, avec un nombre toujours croissant de fonctionnalités disponibles sur les téléphones intelligents (ex. GPS, camera...), il devient primordial de communiquer étroitement avec l'appareil.

Langages à balises

Depuis sa création en 1980, le langage à balises *HTML* s'est imposé en permettant de structurer l'information des pages web. Au fil des années, de nombreuses versions du protocole ont vu le jour, chacune apportant son lot de nouvelles fonctionnalités. *HTML* a été standardisé en 2000 sous la version 4.01 Strict grâce à la norme ISO/IEC 15445:2000. Elle reste aujourd'hui encore la norme la plus utilisée. Les choses sont un peu différentes au niveau des appareils mobiles. Au départ, on a assisté à une émergence de nombreux langages à balises simplifiées dérivant du *HTML* afin de répondre aux contraintes des appareils mobiles, principalement en matière d'optimisation de la bande passante et de la puissance nécessaire au rendu des pages web. C'est ainsi que le *HDML*, devenu ensuite *WML*, et le *cHTML* devenu *iHTML*, ont fait leur apparition principalement dans le marché asiatique. En parallèle, *XHTML* a été créé avec une orientation vers le mobile (cf. Figure 51).

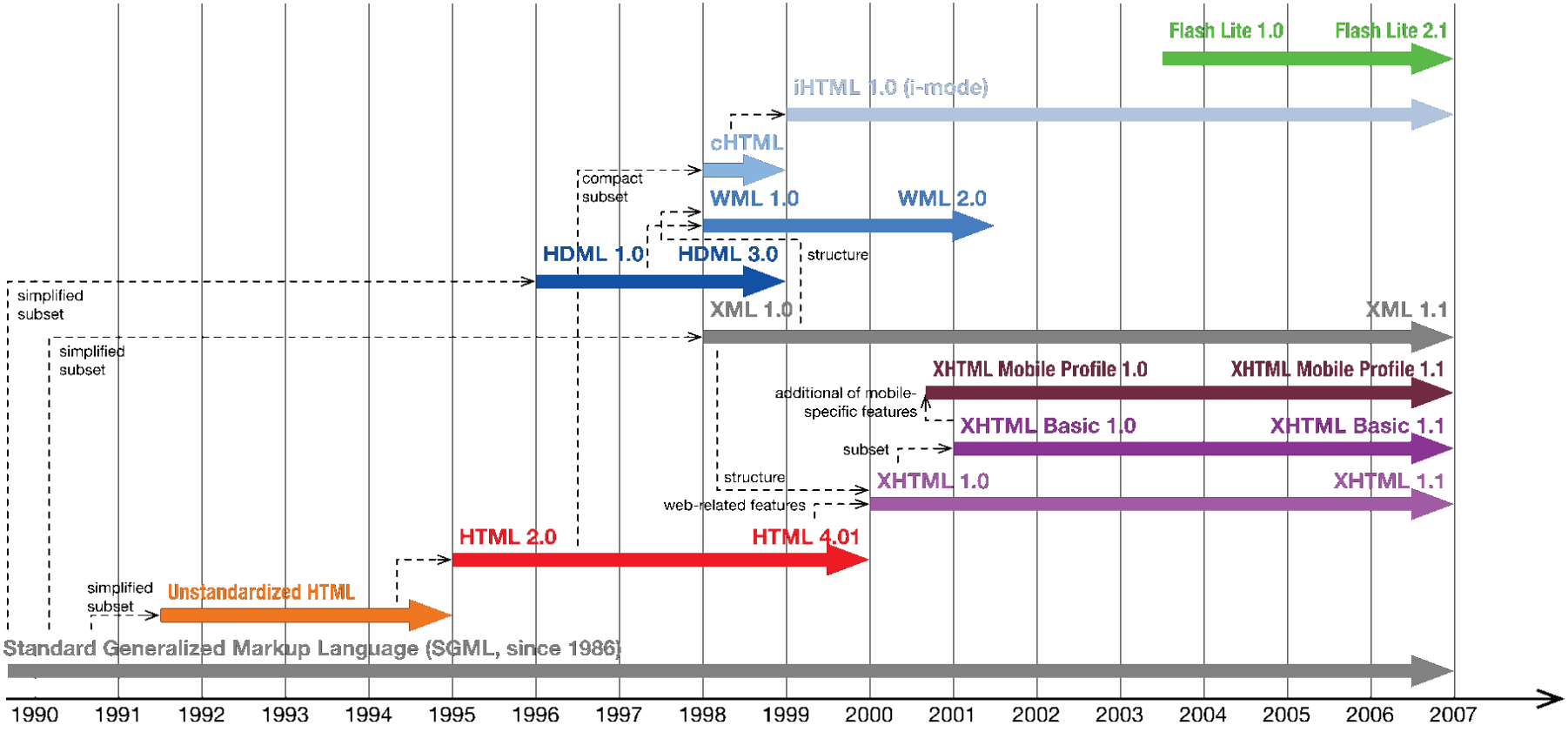


Figure 51: Historique des langages à balises [24]

Face à cette prolifération de langages, le *World Wide Web Consortium* (W3C), organisme chargé du maintien des standards du web, a décidé de développer une nouvelle version du langage *HTML*, soit le *HTML5*. Celui-ci offre de nouvelles possibilités tout en reprenant, en partie, les spécifications du *XHTML*. Il permet, notamment, une intégration simplifiée des contenus multimédias grâce à l'introduction des balises <*audio*> et <*video*>. Une autre innovation est la possibilité de consulter des données hors-ligne et de dialoguer avec certains composants de la plateforme, comme le GPS ou la caméra. Ces nouvelles possibilités permettent de s'affranchir de solutions tierces assez gourmandes en ressources comme *Flash* et assurent une meilleure intégration avec les plateformes. De plus, comme la capacité des téléphones augmente sans cesse, certaines limites à l'utilisation du *HTML* sont maintenant levées. En effet, la plupart des téléphones intelligents disponibles sur le marché sont capables d'afficher ce format sans problème. Par ailleurs, le W3C a lancé une campagne permettant de regrouper les principaux acteurs du domaine (*Mobile Web Initiative*) afin d'offrir des informations sur les pratiques recommandées pour ce genre de plateformes.

Un avantage indéniable des technologies web est leur portabilité. Ainsi, il est facile créer un site unique qui peut être consulté par toute plateforme ayant un navigateur compatible avec le standard. De plus, grâce au feuillet de style (*CSS*), il est facile d'avoir des designs différents s'adaptant à chaque type d'appareil. Cette approche a d'ailleurs été étendue avec le *Device Independent Authoring Language* (DIAL), une nouvelle spécification développée par le groupe *Device Independence Working Group* (DIWG) au sein du W3C. Grâce à DIAL, il est possible de spécifier de manière générique le type d'affichage souhaité en fonction d'un certain nombre de paramètres et d'ajuster plus finement l'interface de son site web, indépendamment du type d'appareil. Enfin, les technologies web sont polyvalentes, libres (format ouvert), standardisées et gratuites, c'est d'ailleurs ce qui a favorisé l'essor d'internet.

L'une des principales contraintes associées à cette technologie est la nécessité de posséder une connexion réseau pour accéder au contenu, ce qui peut être particulièrement limitant dans le domaine de la téléphonie mobile où la couverture réseau fluctue grandement. De plus, les langages à balises ne permettent de structurer l'information que de manière statique et requièrent des langages de scripts comme *JavaScript* pour offrir des contenus dynamiques. Ces langages de scripts, tout comme le *HTML*, doivent être interprétés ce qui signifie que l'appareil de l'utilisateur doit être suffisamment rapide pour que l'expérience utilisateur soit convenable. Ce type de considération est moins important lorsque l'on développe des applications en bureautique, mais devient capital avec des appareils mobiles. Certaines de ces limitations vont disparaître avec *HTML5*, mais encore faudra-t-il attendre que ce

dernier soit standardisé avant que son utilisation ne se généralise. Enfin, comme les langages à balises sont dépendants d'un navigateur, on observe un support plus ou moins poussé des standards.

dotMobi (.mobi)

Il s'agit d'un domaine internet comme *.com* ou *.org* qui est commandité par la *mTLD*, une organisation regroupant de grandes entreprises comme *Google*, *Microsoft*, *Nokia*, *Samsung*, *Ericsson*, *Vodafone*, *T-Mobile*, *Telefónica Móviles*, *Telecom Italia Mobile*, *Orascom Telecom*, *GSM Association*, *Hutchison Whampoa*, *Syniverse Technologies* et *Visa*. Le but de cette initiative est de promouvoir l'internet mobile en lui offrant un nom de domaine facilement identifiable. L'organisme est également impliqué avec le W3C dans le *Mobile Web Initiative* et participe à l'élaboration des bonnes pratiques en matière de développement pour les plateformes mobiles. *dotMobi* propose un outil (*Ready.mobi*) qui permet de tester la compatibilité de son site avec les appareils mobiles. Le domaine permet aussi, grâce à des serveurs spécifiques (proxy), de compresser et de réadapter des pages web afin d'optimiser leur transfert sur une plateforme mobile.

L'idée de *dotMobi* est intéressante car elle offre un domaine clairement identifiable pour tout ce qui touche au web mobile et facilite ainsi le repérage des contenus pour les appareils mobiles. Toutefois, en faisant appel à un nom de domaine, on va à l'encontre de l'une des règles du web, soit l'indépendance de toute plateforme. Certains noms comme *fun.mobi* ou *flowers.mobi* se sont négociés à quelques milliers de dollars, ce qui peut être un frein pour certaines compagnies ou utilisateurs souhaitant offrir des services mobiles. La création d'un nom de domaine spécifique peut aussi conduire à une certaine redondance avec des sites web déjà existants. Enfin, il faut remarquer que le nom de domaine *.mobi* est plus long à taper sur la plupart des téléphones cellulaires que *.com* par exemple.

C. Approches hybrides

Les approches natives et web présentant des avantages et des inconvénients, de nouvelles approches cherchent à tirer profit des avantages de chacune. Globalement, ce type d'approche fait appel à un méta-langage grâce auquel il est possible de spécifier le fonctionnement d'une application. Le code s'adapte ensuite automatiquement aux différentes plateformes supportées. L'idée est intéressante, car elle permet de réduire les coûts de développement multiplateformes tout en offrant un accès plus poussé aux fonctionnalités des appareils.

PhoneGap

PhoneGap est une API reposant sur les technologies web développée par la société *Nitobi*. Le principe de *PhoneGap* est de permettre la création rapide d'applications grâce à des standards ouverts comme *HTML*, *CSS* et *JavaScript*. La transition est ainsi facilitée pour toute personne familière avec ces langages. Grâce à des scripts, le code est traduit de façon à pouvoir être exécuté sur une plateforme particulière. Le kit de développement fourni par *Nitobi* permet également d'enrichir *JavaScript* en offrant un accès privilégié à certains composants du téléphone comme l'accéléromètre ou la caméra. L'approche de *PhoneGap* est assez similaire à celle du *WebOS* de *Palm*, mais étend les possibilités aux autres catégories de téléphones intelligents (*iPhone*, *Android*, *BlackBerry*, *Palm*, *Symbian*). Un des avantages de cette technologie et qu'elle est libre et téléchargeable gratuitement ce qui assure une certaine pérennité des logiciels développés avec cette solution. Toutefois, la documentation n'est pas très bien faite, peut-être dû au fait que *Nitobi* vend des services de formation. Il semble aussi que les programmes créés avec ce kit de développement soient assez longs à exécuter, dépendant du type de téléphone utilisé. Ceci est peut-être dû au code généré qui doit être interprété. Enfin, l'API se voulant multiplateformes, elle ne peut s'exécuter de manière optimale sur chaque type d'appareil ce qui peut donner lieu à différentes expériences utilisateurs pour une même application.

AirPlaySDK/Bedrock

Ces deux projets font appel à un langage de référence, soit C++ pour *AirPlaySDK* et *Java* pour *Bedrock*. Ces langages servent à développer des applications en utilisant les librairies fournies. *Bedrock* se distingue légèrement dans ce domaine, car il offre aussi la possibilité d'inclure, en plus des librairies standard, des librairies spécifiques et optimisées pour le développement d'applications sur plateformes mobiles. *AirPlaySDK* supporte les plateformes *iPhone OS*, *Android*, *Samsung*, *Bada*, *Symbian*, *Windows Mobile*, *BREW*, *Palm/HP*, *WebOS* et *Maemo*. *Bedrock* supporte les mêmes plateformes à l'exception de *Bada*, *WebOS*, *Maemo* et *Symbian*, mais gère en plus *HTML5*, *Nintendo DS*, *Flash*, *Sony PSP* et *Antix*. L'approche est intéressante et permet de s'affranchir de l'apprentissage de nouveaux langages et environnements de développement. Cependant, ces deux solutions demeurent propriétaires ce qui entraîne une dépendance aux compagnies fournissant cet API. Dans le cas d'*AirPlaySDK*, seuls les environnements de développement *Microsoft Visual C++* et *Apple Xcode* sont supportés, ce qui est assez restrictif.

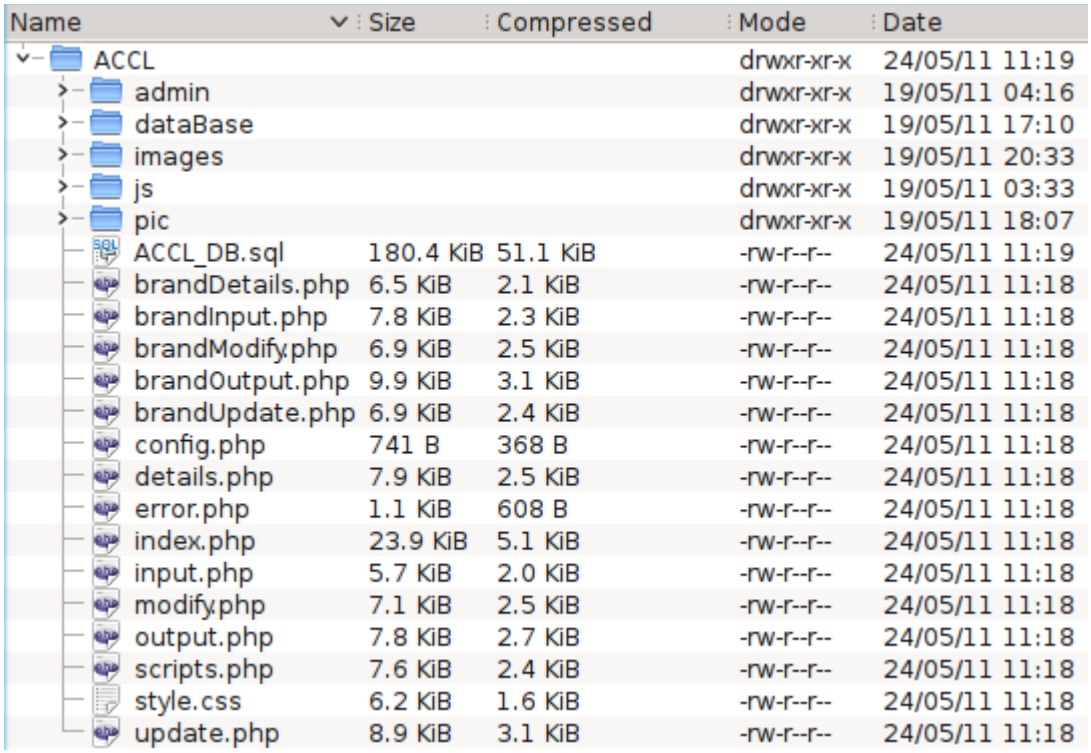
Annexe 2 : Déploiement de la solution

Versions utilisées

Le système a été développé sur la plateforme XAMPP pour *Linux* version 1.7.4. Cette version contient *Apache* version 2,2,17, *MySQL* version 5.5.8, *PHP* version 5.3.5 et *phpMyAdmin* version 3.3.9. La solution a été exclusivement testé sur le navigateur *Firefox* 4.0.1 avec l'encodage UTF-8.

Archive

Avec ce mémoire, nous fournissons également une archive en format zip contenant les différents composants du système de listes de vérification dynamiques. Voici une image de l'arborescence de l'archive :



Name	Size	Compressed	Mode	Date
ACCL			drwxr-xr-x	24/05/11 11:19
admin			drwxr-xr-x	19/05/11 04:16
dataBase			drwxr-xr-x	19/05/11 17:10
images			drwxr-xr-x	19/05/11 20:33
js			drwxr-xr-x	19/05/11 03:33
pic			drwxr-xr-x	19/05/11 18:07
ACCL_DB.sql	180.4 KiB	51.1 KiB	-rw-r--r--	24/05/11 11:19
brandDetails.php	6.5 KiB	2.1 KiB	-rw-r--r--	24/05/11 11:18
brandInput.php	7.8 KiB	2.3 KiB	-rw-r--r--	24/05/11 11:18
brandModify.php	6.9 KiB	2.5 KiB	-rw-r--r--	24/05/11 11:18
brandOutput.php	9.9 KiB	3.1 KiB	-rw-r--r--	24/05/11 11:18
brandUpdate.php	6.9 KiB	2.4 KiB	-rw-r--r--	24/05/11 11:18
config.php	741 B	368 B	-rw-r--r--	24/05/11 11:18
details.php	7.9 KiB	2.5 KiB	-rw-r--r--	24/05/11 11:18
error.php	1.1 KiB	608 B	-rw-r--r--	24/05/11 11:18
index.php	23.9 KiB	5.1 KiB	-rw-r--r--	24/05/11 11:18
input.php	5.7 KiB	2.0 KiB	-rw-r--r--	24/05/11 11:18
modify.php	7.1 KiB	2.5 KiB	-rw-r--r--	24/05/11 11:18
output.php	7.8 KiB	2.7 KiB	-rw-r--r--	24/05/11 11:18
scripts.php	7.6 KiB	2.4 KiB	-rw-r--r--	24/05/11 11:18
style.css	6.2 KiB	1.6 KiB	-rw-r--r--	24/05/11 11:18
update.php	8.9 KiB	3.1 KiB	-rw-r--r--	24/05/11 11:18

- Le répertoire *admin* contient les pages relative à la gestion de la session, la création d'un compte utilisateur ainsi que le changement de mot de passe.

- Le répertoire *dataBase* contient toutes pages permettant de gérer les différentes tables de la base de données.

- Le répertoire *images* contient toutes les illustrations relatives aux recommandations touchant les plateformes mobiles ainsi que l'image de marque.

- Le répertoire *js* contient des scripts JavaScript relatifs au fonctionnement de l'application *Lightbox2* qui permet d'afficher les illustrations en taille réelle de manière dynamique.

- Le répertoire pic contient différentes images servant à l'interface du site web ainsi qu'à *Lightbox2*.

- Le fichier *ACCL_DB.sql* contient le code SQL pour générer la base de données.

- Les fichiers, *input*, *output*, *modify*, *update* et *details,* pour les plateformes mobiles, ainsi que leurs équivalent préfixé de *brand*, pour l'image de marque, constituent les différentes pages permettant respectivement d'ajouter une recommandation, de la mettre dans la base de données, de la modifier, de mettre à jour l'information dans la base de données et enfin d'afficher la recommandation.

- Le fichier *error* permet de gérer les erreurs lorsqu'il y a des problèmes au niveau du transfert d'images sur le serveur.

- Le fichier *scripts* contient un ensemble de fonctions permettant de fournir certaines fonctionnalités au sein de l'interface.

- Le fichier *index* est la page d'accueil ; elle a été nommée ainsi pour permettre son chargement automatique par le serveur web lorsque l'on est dans le répertoire racine.

- Le fichier *style.css* est constitué de l'ensemble des styles s'appliquant au différents éléments de l'interface du site web.

- Le fichier *config* contient les informations nécessaires pour se connecter à la base de données.

Procédure d'installation

Il faut tout d'abord disposer d'un serveur web, d'une base de données et du langage de scripts PHP. À cet effet, deux options sont possibles : soit utiliser un service d'hébergement qui offre toutes ces facilités ou bien installer soit même les logiciels. Dans le second cas, on recommande d'utiliser la solution offerte par XAMPP qui a l'avantage d'être multi-plateformes et téléchargeable gratuitement sur internet. De plus, XAMPP offre un environnement prêt à l'emploi sans configuration complexe. XAMPP peut être téléchargé à l'adresse suivante : www.apachefriends.org/fr/xampp.html

Une fois que l'on dispose d'un serveur fonctionnel, il suffit alors de décompresser l'archive à l'endroit souhaité sur le serveur. Il ne reste plus qu'à générer la base de données et configurer la

connexion. On recommande ici d'utiliser une solution comme *phpMyAdmin* où un logiciel équivalent qui permet d'importer le fichier *ACCL_DB.sql* afin de créer la base de données (en partant du principe que l'on connait son identifiant et son mot de passe et que l'on sait comment se connecter à *phpMyAdmin*). Une fois dans l'interface, il suffit de cliquer sur le lien « Import » en haut de l'écran (cf. Figure 52) afin d'importer un fichier.

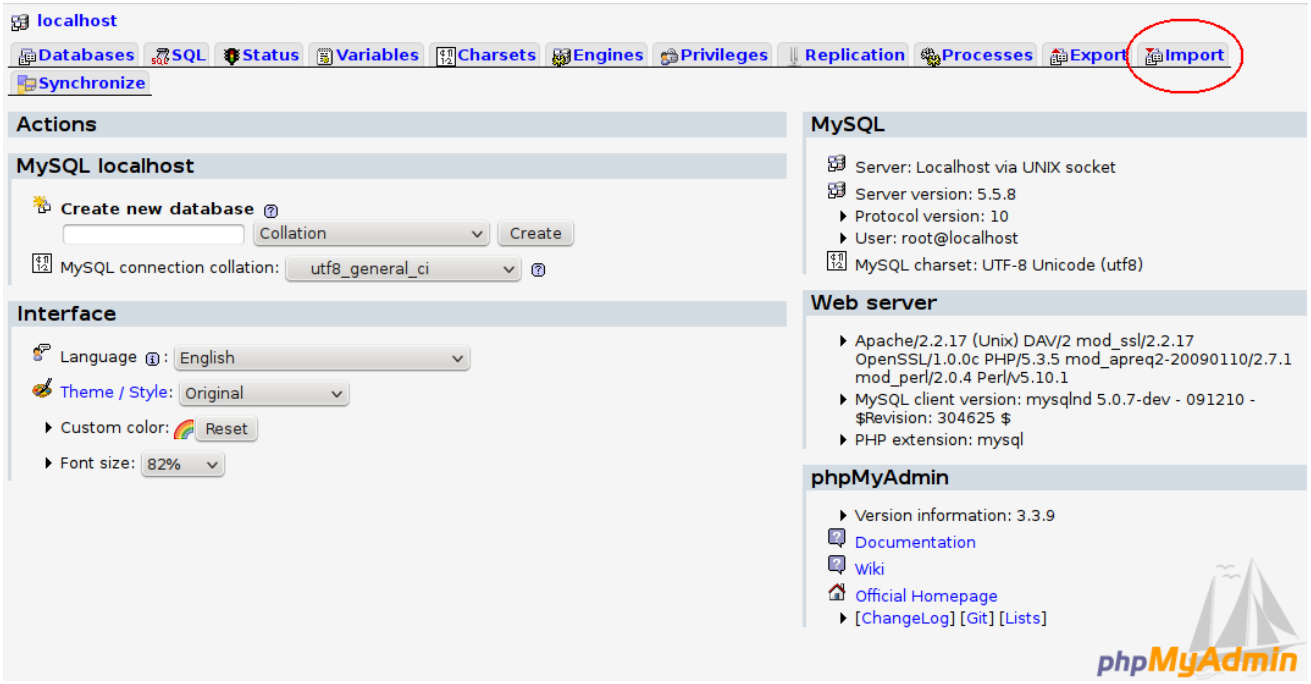


Figure 52: Écran principal de phpMyAdmin

On se retrouve alors avec un nouvel écran où l'on peut sélectionner notre fichier en s'assurant que l'encodage des caractères est bien en UTF-8 (cf. [1] Figure 53). Il faut ensuite s'assurer que le format d'importation est bien SQL (cf. [2] Figure 53) et enfin valider le tout en cliquant sur le bouton « GO » (cf. [3] Figure 53).

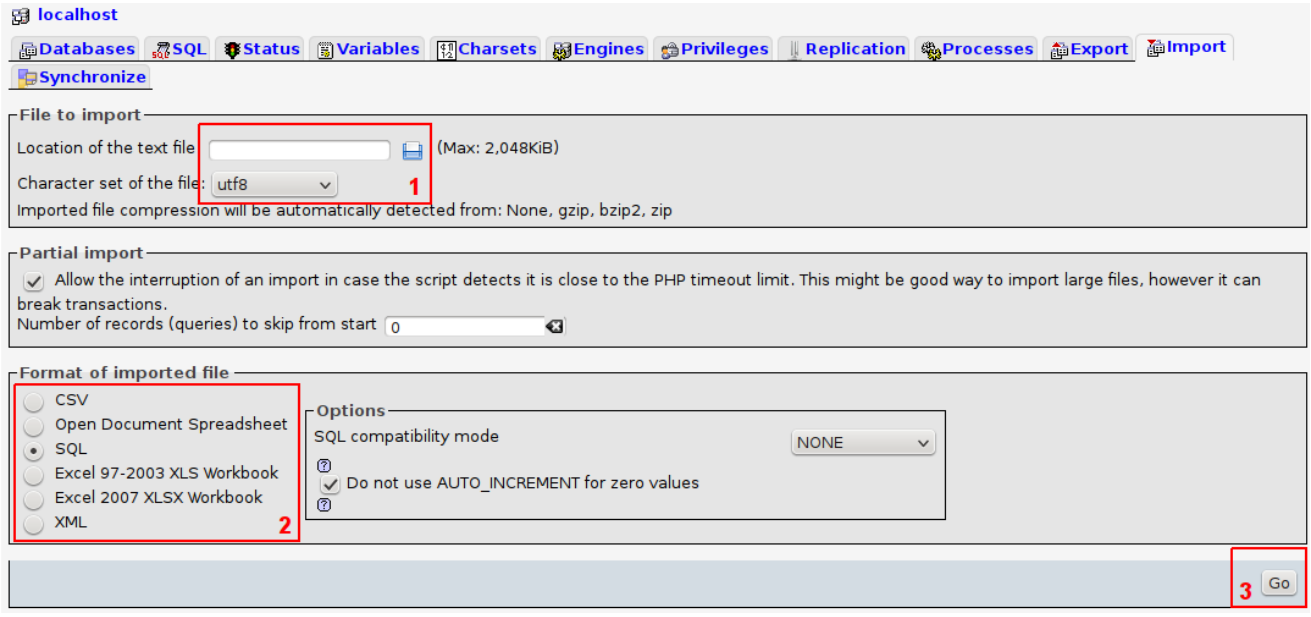


Figure 53: Écran d'importation d'une base de données

Il ne reste plus alors qu'à éditer le fichier *config.php* sur le serveur avec les informations appropriées pour que le système soit fonctionnel. Plus précisément, il faut éditer le texte entre apostrophes (ex : 'root') qui se trouve juste après le mot précédé du signe « $ ». Il faut tout d'abord fournir l'adresse du serveur en changeant la variable *$servername* avec la valeur adéquate. Si l'on ignore cette valeur, il faut laisser « localhost » qui est un alias pointant sur l'adresse IP de la machine sur laquelle s'exécute le serveur. Enfin, il faut changer les variables *$dbusername* et *$dbpassword* avec l'identifiant et le mot de passe que l'on utilise pour se connecter à la base de données. Après avoir sauvegardé le fichier, on peut alors se connecter à l'adresse où l'on a décompressé l'archive afin d'utiliser le système.

Annexe 3 : Entente de confidentialité

Cette étude a pour but d'évaluer un système de listes de vérification interactives. Votre participation nous fournira des informations très utiles à cette fin.

Nous nous engageons à ne divulguer aucune information permettant de vous identifier. De plus, les résultats obtenus seront codés ou traités de façon à ce qu'il ne soit pas possible de vous reconnaître.

En tant que participant, je comprends que :

1. Ma participation est volontaire.
2. J'ai le droit de ne pas répondre à certaines questions.
3. Mon nom et mon identité ne seront pas divulgués.
4. Toutes mes réponses et commentaires seront gardés confidentiels et ne seront utilisés que dans le cadre de cette étude.

J'ai lu ce document et j'ai pu poser toutes les questions voulues afin de clarifier les points obscurs.

Participant :_____ Analyste :_____ Date :_____