



HAL
open science

Conception et réalisation d'un thème iPhone pour le Genero Web Client

Geoffrey Kretz

► **To cite this version:**

Geoffrey Kretz. Conception et réalisation d'un thème iPhone pour le Genero Web Client. Informatique mobile. 2011. dumas-01233441

HAL Id: dumas-01233441

<https://dumas.ccsd.cnrs.fr/dumas-01233441>

Submitted on 25 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**CONSERVATOIRE NATIONAL DES ARTS ET METIERS
CENTRE RÉGIONAL ASSOCIE DE STRASBOURG
MÉMOIRE**

**présenté en vue d'obtenir
le DIPLÔME D'INGÉNIEUR CNAM**

**SPÉCIALITÉ : INFORMATIQUE
OPTION : Informatique Système d'information**

**par
Geoffrey KRETZ**

Conception et réalisation d'un thème iPhone pour le Genero Web Client

Soutenu le 05 décembre 2011

JURY

PRÉSIDENT : Mr I. WATTIAU

**MEMBRES : Mr O. IMBERT
Mr H. KRAESS
Mr B. GILLIGMANN
Mr C. KLEINPETER**

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

En premier lieu, je voudrais remercier mes filles, Lou-Andréa et Sidonie, la première m'a donné l'envie de me lancer dans le cursus du C.N.A.M. et ensemble elles m'ont donné le courage et la force de le finir.

Je voudrais également remercier ma compagne Yasmina pour sa présence et son soutien sans faille pendant ces longues années.

Je désire aussi adresser toute ma gratitude responsable pédagogique de la filière informatique du C.N.A.M. Alsace, Cédric Kleinpeter, pour sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je voudrais remercier mon tuteur dans la société Four J's et directeur de projets, Olivier Imbert, et le directeur général du centre de Strasbourg, Bruno Sittler, de m'avoir accordé leur confiance et permis de mener ce mémoire à bon terme.

Enfin, j'aimerais exprimer ma reconnaissance envers les amis et collègues qui m'ont apporté leur support moral et intellectuel tout au long de ma démarche. Un grand merci en particulier à Christophe Dehlinger pour ses relectures et conseils concernant la rédaction.

Liste des abréviations

4 GL : 4 th Generation Language, langage de quatrième génération

AJAX : Asynchronous Javascript and XML

API : Application Programming Interface

AUI : Abstract User Interface

CGI : Common Gateway Interface, une technologie utilisable par un serveur web

cHTML : condensed HTML

CSF : Client Side Framework

CSS : Cascading Style Sheet

DVM : Dynamic Virtual Machine, le moteur d'exécution de code Genero

EDI : Environnement de Développement Intégré, programme regroupant un ensemble d'outils pour le développement de logiciel

Four J's : Four J's Development Tools

GAS : Genero Application Server

GCS : Genero Clients and Services

GDBEM : Genero db Entreprise Manager

GDCAX : Genero Desktop Client Active X

GDC : Genero Desktop Client

GEP : Genero Evaluation Program

GPL : General Public License

GRW : Genero Report Writer

GWC : Genero Web Client

GWS : Genero Web Services

HTML : HyperText Markup Language

HTTP : HyperText Transfert Protocol

IIS : Internet Information Services, le logiciel de serveur web de la plateforme Windows NT

ISAPI : Internet Server Application Programming Interface, une technologie utilisable par un serveur web IIS

J2EE : Java 2 Entreprise Edition

LGPL : Lesser General Public License

ODI : Open Database Interface

OS : Operating System

PNG : Portable Network Graphics

QA : Quality Assurance

RDSQL : Relational Database Structured Query Language

RIA : Rich Internet Application

SBRE : Snippet-Based Rendering Engine

SDK : Software Development Kit

SGBD : Système de Gestion de Bases de Données

SI : Système d'Information

SQL : Structured Query Language

URI : Uniform Resource Identifier

URL : Uniform Resource Locator

VoIP : Voice over Internet Protocol

WAP : Wireless Application Protocol

WML : Wireless Markup Language

WiX : Windows Installer XML

XAML : Extensible Application Markup Language

XML : eXtensible Markup Language, un langage de balisage informatique générique

XSL : eXtensible Stylesheet Language

XSLT : XSL Transformations

xHTML : eXtensible HyperText Markup Language

Table des matières

Introduction	6
I. Le lieu d'accueil	8
I.1. Qui sommes-nous ?	8
I.2. Que faisons-nous ?	9
I.3. Une architecture ouverte	10
I.4. Une logique métier indépendante du SGBD	10
I.5. Codez une fois, déployez n'importe où	10
I.6. Architectures Orientées Services	11
I.7. À propos de nos produits	12
I.7.1. Genero 4GL	12
I.7.2. Genero Clients and Services	12
I.7.3. Genero Studio	13
I.7.4. Genero db	13
I.7.5. Genero Report Writer	13
I.7.6. Genero Evaluation Program	13
I.8. Exemple d'utilisation	13
I.9. Données commerciales et financières	14
I.9.1. Chiffre d'affaires	14
I.9.2. Marchés	15
I.9.3. Canaux de distribution	16
I.9.4. Modèle économique	16
I.9.5. Où est située la société	17
I.10. Ressources humaines	18
I.10.1. Place du candidat dans l'organigramme	18

I.11.« Énergie » informatique disponible au moment du lancement du projet	19
I.12.Langages utilisés	19
II. Le contexte du projet	20
II.1.Service dans lequel se déroule le projet	20
II.2.La révolution mobile	20
II.3.L'iPhone	24
II.3.1.L'iPhone en chiffre	24
II.3.2.Données techniques	27
II.4.L'App Store	30
II.4.1.L'App Store en chiffre	30
II.4.2.Caractéristiques	31
II.4.3.Conditions d'utilisations	32
II.4.4.Distribution des applications	32
II.4.5.Développement	34
II.5.Les applications mobiles	35
II.5.1.Simplicité et facilité d'usage	35
II.5.2.Focalisation	39
II.5.3.Cohérence	40
II.5.4.Mise à disposition aux utilisateurs	40
II.6.Genero	40
II.6.1.Un fonctionnement en mode connecté	40
II.6.2.L'arbre AUI	40
II.6.3.Vue d'ensemble du Genero Application Server	42
II.6.4.Vue d'ensemble du Genero Desktop Client	45
II.6.5.Vue d'ensemble du Genero Web Client	47
III.Le projet	49

III.1.Introduction du projet	49
III.1.1.Comparatif	49
III.1.2.Résumé	52
III.1.3.Choix	53
III.2.Les objectifs du projet	53
III.3.Les contraintes du projet	53
III.3.1.Contraintes de temps	53
III.3.2.Contraintes de moyen	53
III.3.3.Contraintes financières	53
IV.Déroulement du projet	54
IV.1.Organisation générale du projet	54
IV.2.Méthode de conduite de projet	54
IV.3.Responsabilité du candidat	54
IV.4.Équipes prévues, encadrement assuré par le candidat	54
IV.5.Options déjà connues entre lesquelles se feront certainement les choix ultérieurs	55
IV.6.Tests prévus	55
IV.7.Modalité de validation du projet	55
V. État de l'Art	56
V.1.Fonctionnement du moteur de rendu du GAS	56
V.1.1.Le SBRE	56
V.1.2.Comment une application est rendue par le GWC	58
V.1.3.Modèle	59
V.1.4.Cascading Style Sheet	59
V.1.5.JavaScript	60
V.1.6.Langage de macro	60

V.2.Le thème AJAX	61
V.3.Apple Web Application Guidelines	62
V.3.1.Création d'une icône pour les applications web	62
V.3.2.Une mise en page adaptée à l'iPhone	63
V.3.3.L'approche par les listes	66
V.3.4.Utilisation des éléments natifs	69
V.3.5.Création d'éléments de formulaires personnalisés	71
V.3.6.L'attention portée au texte	73
V.4.Cadriciels Web	73
V.4.1.iUi	73
V.4.2.jQTouch	76
V.4.3.WebApp.net	76
V.4.4.iWebKit	77
V.4.5.Sencha Touch	77
V.4.6.Cappuccino et SproutCore	77
V.4.7.PastryKit	78
V.4.8.Les autres cadriciels	78
V.5.Conclusion	78
VI.Recherche de solutions	79
VI.1.Évolution du thème AJAX	79
VI.1.1.Fichier .per	79
VI.1.2.Le rendu GDC	84
VI.1.3.Le rendu GWC	84
VI.1.4.Problématique	85
VI.2.Mise en page des applications	85
VI.2.1.Limitation de la grille	86

VI.3.Réduire la complexité	86
VI.3.1.Genero Mobile Profile	86
VI.3.2.La question des tables	87
VI.3.3.Le mode d’affichage	90
VI.4.Réalisation	90
VI.4.1.Grille	90
VI.4.2.Unsupported	93
VI.4.3.Géolocalisation	95
VII.Risques	97
VII.1.Évaluation des risques	97
VII.1.1.Liste des risques	97
VII.1.2.Classification des risques	99
VII.1.3.Plan d’action	99
VIII.Portabilité	102
IX.Devenir du projet	103
Conclusion	104
X. Bibliographie	107
X.1.Livres	107
X.2.Publications	107
XI. Liste de figures	108
XII.Liste des tableaux	111

Introduction

L'informatique a connu une nouvelle révolution lors de la décennie passée. En effet, après les changements apportés dans la façon d'appréhender l'informatique, mais aussi dans la manière de percevoir le monde qui nous entoure avec la démocratisation de l'Internet à la fin des années 1990, le début du XXIe siècle a vu l'arrivée de terminaux permettant de poursuivre l'expérience de l'Internet de façon mobile et nomade. Cette révolution a définitivement pris son envol en 2007 lorsque Steve Jobs a présenté l'iPhone, le smartphone¹ d'Apple, basé sur une technologie tactile multipoints et embarquant le système d'exploitation iOS et le navigateur web Safari mobile. L'iPhone a été certainement le premier téléphone dont l'usage premier n'était pas de téléphoner, mais d'accéder à l'Internet. L'iPhone a connu un succès commercial et un engouement spectaculaire, aussi bien au niveau des consommateurs que des concepteurs de logiciels et il a obligé la concurrence à s'adapter et à proposer elle aussi des smartphones offrant des fonctionnalités similaires, en particulier Google et son système d'exploitation mobile Android.

La société pour laquelle je travaille commercialise Genero, un environnement de développement et de déploiement permettant de créer des applications puissantes et fiables dans tous les domaines professionnels. Une des forces de Genero est de pouvoir exécuter une même application dans des environnements hétérogènes sans avoir besoin d'en modifier les sources. Néanmoins, la société ne proposait au moment du choix du sujet de mémoire aucune solution spécifique à l'iPhone et adaptée à des écrans de plus petite taille que celle des écrans d'ordinateur et plus particulièrement à des écrans tactiles.

Je suis de très longue date un amateur de la marque à la pomme, mais aussi des nouvelles technologies en général et de l'Internet en particulier. J'ai toujours attaché une importance particulière aux interfaces et à l'expérience utilisateur. En juin 2009, j'ai donc saisi l'opportunité de concilier ces centres d'intérêt avec la réalisation du mémoire d'ingénieur CNAM en proposant à ma société de concevoir et de participer au développement d'une solution logicielle permettant aux applications Genero de s'exécuter sur des iPhone, offrant aux utilisateurs une interface et une expérience en adéquation avec ces terminaux.

¹ Bien que le mot smartphone soit un anglicisme, et qu'il pourrait être remplacé par téléphone intelligent, il sera utilisé dans le reste du document pour éviter la lourdeur du terme francisé.

Après une présentation du lieu d'accueil, de son activité et de ses produits, je développerai de manière approfondie le contexte du projet en présentant tour à tour l'iPhone et l'App Store, plateforme de téléchargement d'application distribuée par Apple sur les appareils mobiles fonctionnant sous iOS, puis sur ce qui fait l'ADN des applications mobiles et je finirai cette partie en décrivant les produits Genero dans lequel ce projet est directement impliqué. Je parlerai ensuite du projet à proprement parler, des choix qui l'ont motivé, de ses objectifs et de ses contraintes. Dans la partie suivante, j'expliquerai le déroulement du projet. Ensuite, j'essayerai de couvrir l'état de l'art aussi bien au niveau des produits existants dans la société et présentant des points communs avec ce projet, que de manières externes en étudiant et analysant les recommandations faites par Apple en matière de développement d'applications mobiles ainsi que les cadres web mobiles existants. Dans une nouvelle partie, je présenterai les différentes recherches que j'ai réalisées pour trouver les solutions les plus pertinentes et je donnerai quelques exemples sur la manière dont a été réalisé le projet. Enfin, je finirai en parlant tour à tour de la manière dont j'ai abordé la gestion des risques, de la portabilité du projet et du devenir du projet.

I. Le lieu d'accueil

Fondée en 1995 et dirigée par Jean-Georges Schwartz depuis 1998, la société Four J's (Four J's Development Tools) conçoit, développe et commercialise Genero, un environnement de développement et de déploiement permettant de créer des applications puissantes et fiables dans tous les domaines professionnels.

Basé en Europe et présent dans le monde entier, Four J's a des clients dans des domaines aussi variés que la banque, l'énergie, les médias, les télécommunications, la fabrication industrielle et les gouvernements.

I.1. Qui sommes-nous ?

Four J's a été fondé par quatre développeurs prénommés « Jean » au début des années quatre-vingt-dix et tire son succès du respect de deux valeurs de base : l'accent mis sur la création du meilleur environnement de développement pour les applications professionnelles et l'écoute des besoins du client. C'est en respectant ces deux valeurs que Four J's a fidélisé sa clientèle et a connu croissance et rentabilité depuis sa création.

Four J's est né de la frustration d'un homme par rapport à l'industrie du logiciel. Cet homme est Jean-Georges Schwartz, fondateur et PDG de Four J's. Il a développé et vendu son propre logiciel aux entreprises dans les années 1980 et a pu se faire la main quant aux problèmes et aux tribulations de la communauté de développeurs.

Il a constaté que les applications, malgré les investissements lourds réalisés pour leur développement, n'avaient pas une durée de vie supérieure à cinq ans. À chaque changement de paradigme technologique majeur, qu'il s'agisse de l'évolution des mini-ordinateurs propriétaires vers des systèmes ouverts, ou des architectures orientées serveur vers les architectures client-serveur puis vers l'Internet, les outils avec lesquels il a travaillé présentaient le même défaut de conception inhérent : la désuétude.

Pour les grandes entreprises avec des budgets conséquents, il s'agissait d'un problème mineur. Néanmoins, pour les petites et moyennes entreprises, la refonte des applications avec de nouveaux outils destinés à « réinventer la roue » était un problème majeur. Les ressources auraient été mieux investies dans l'amélioration de l'application et l'accroissement de son avantage concurrentiel.

Après que son entreprise fut abandonnée à son sort une fois de trop avec une application qui n'était plus fonctionnelle sur les nouvelles architectures matérielles, il a cherché en vain un outil qui ne ferait que prolonger la durée de vie de son application ainsi que sa logique métier. Dès lors, il ne lui a pas fallu longtemps pour comprendre que sa mission était semblable à la quête du Saint Graal et il a donc décidé de créer de lui-même un outil qui allait répondre à ses propres besoins et probablement à ceux de sociétés comme la sienne, Four J's était né.

En 1995, la première incarnation de son idée de base a vu le jour et a été baptisée « Universal Compiler »². Elle était capable de créer une application qui s'exécute sur des systèmes d'exploitation différents sans recompilation. À l'époque, les applications étaient basées sur des écrans ASCII conçus pour les terminaux asynchrones.

L'arrivée de Microsoft Windows et de l'Internet ont rendu obsolète les terminaux asynchrones à la fin des années 1990. Est alors arrivé le besoin de déployer et exécuter les applications sur des interfaces utilisateurs graphiques dans des environnements hétérogènes. Four J's a alors ajouté à « Universal Compiler » une couche de présentation abstraite permettant aux applications de s'exécuter indifféremment sur des machines sous Windows, dans une machine virtuelle Java et sur des navigateurs web (clients légers). Cette version améliorée de « Universal Compiler » est rebaptisée en « Business Development Suite »³ qui était à ce moment un produit unique.

À l'automne 2003, une nouvelle évolution majeure a vu le jour. Appelée « Genero »⁴, elle améliore nettement l'existant en basant sa couche de présentation abstraite sur le standard XML (eXtensible Markup Language).

I.2. Que faisons-nous ?

Dans un monde d'instabilité géopolitique et économique, les budgets informatiques sont généralement les premières victimes des financiers. Pourtant le rôle des services informatiques est de réaliser des économies d'échelle avec la rationalisation des processus opérationnels.

Genero permet de faire plus avec moins et donne la possibilité aux équipes de développement d'innover avec moins de ressources. En effet, Genero permet le développement

² Compilateur Universel

³ Suite de Développement pour Entreprises

⁴ Dans la suite du document, «Genero» sera simplement écrit Genero. Ce terme regroupe l'ensemble des solutions proposées par l'entreprise.

d'applications professionnelles avec moins de code. En réduisant la longueur du code jusqu'à cinq fois, cela laisse plus de temps pour la maintenance évolutive. Moins de code signifie également moins de maintenance corrective et des délais de mise sur le marché plus courts pour les nouvelles applications. Une plus grande innovation signifie une logique métier qui va de pair avec les tendances du marché, elle permet d'accroître l'avantage concurrentiel des entreprises utilisant Genero.

I.3. Une architecture ouverte

Four J's propose une infrastructure (une machine virtuelle dynamique, une couche de présentation abstraite, un langage de programmation, un environnement de développement complet et une base de données) pour le développement et le déploiement d'applications de hautes performances pouvant être utilisées dans des missions critiques.

De surcroît, l'infrastructure de déploiement offerte permet de mettre en production rapidement et efficacement des applications professionnelles. Cette infrastructure prend en charge plusieurs bases de données, systèmes d'exploitation et interfaces graphiques avec un minimum de modification lors de la migration vers un autre SGBD (Système de Gestion de Bases de Données), un autre système d'exploitation ou une autre interface graphique.

I.4. Une logique métier indépendante du SGBD

Faciliter l'accès aux SGBD est au cœur même de Genero, du fait de l'orientation professionnelle des applications réalisées. Les développeurs peuvent implémenter la logique métier avec un langage de programmation simple, facile à apprendre et qui intègre des instructions SQL (Structured Query Language) comme élément du langage. Genero « Open Database Interface »⁵ fonctionne avec Genero db, IBM DB2, Informix, Microsoft SQL Server, MySQL, Oracle, PostgreSQL et Sybase par le biais de pilotes compilés nativement sur chaque plate-forme. Genero permet d'écrire facilement des applications exploitant un environnement hétérogène de SGBDs.

I.5. Codez une fois, déployez n'importe où

Beaucoup de langages prétendent être plateforme-indépendant⁶, avec divers degrés de vérité. Avec Genero, l'indépendance à la plate-forme est bien réelle. Par exemple, les applications peuvent s'exécuter du côté serveur sur Windows, Mac OS X, Linux et les principaux UNIX du marché, sans

⁵ Accès ouvert vers les SGBD

⁶ Indépendant de la plate-forme, du système d'exploitation

besoin de recompilation des binaires. Les applications sont accessibles du côté client sur toutes les plateformes du marché grâce aux clients graphiques.

L'infrastructure de déploiement client/serveur est elle aussi très polyvalente et fonctionne aussi bien avec le serveur web Microsoft IIS (Internet Information Services) sur Windows NT qu'avec les serveurs web Apache/Lighttpd sur Linux/UNIX et avec Apache sur Mac OS X.

De même, il est possible de réduire les cycles de tests, du fait qu'une application donnée fonctionne exactement de la même façon sur AIX, HP-UX, Linux, Solaris, Windows ou Mac OS X.

I.6. Architectures Orientées Services

Les produits de Four J's interopèrent avec des logiciels écrits dans une variété de langages et s'exécutant sur une large gamme de plates-formes en utilisant des protocoles standards tels que les services web. Avec Genero, il ne sera jamais nécessaire de changer les bases de données, les applications ou les sites web existants.

Fidèle à la vision de son fondateur, Four J's s'efforce d'isoler la logique métier de la présentation et des couches de données. Au fil des ans, les clients de Four J's ont modernisé leur matériel et leurs systèmes d'exploitation sans avoir à réécrire leur logique métier.

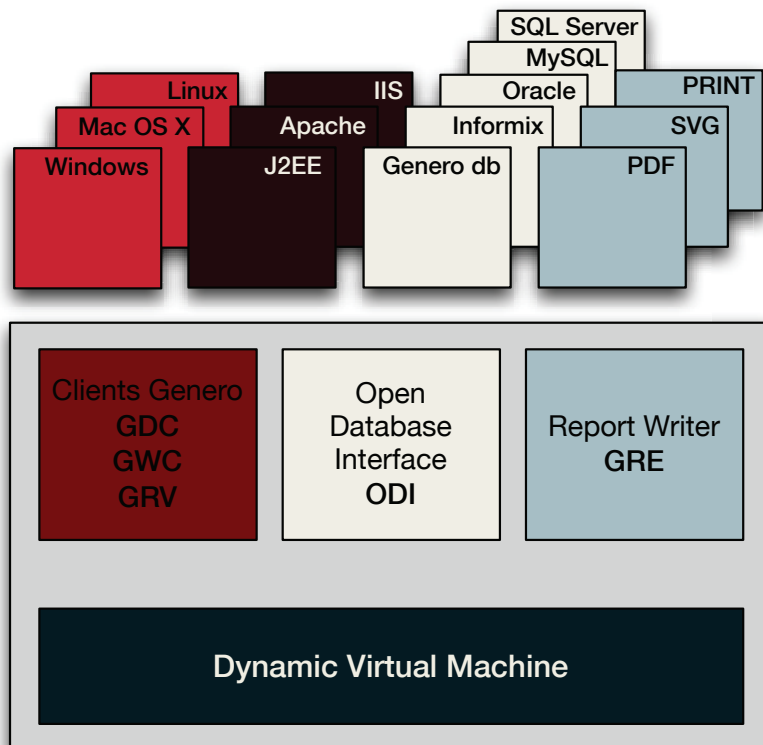


Figure 1 : Les couches logiques de l'écosystème Genero

I.7. À propos de nos produits

I.7.1. Genero 4GL

Genero 4GL (4th Generation Language) est un langage de programmation conçu pour le développement d'applications. Il est facile à apprendre, lisible et orienté vers les applications professionnelles. C'est un langage de 4e génération, ce qui signifie que contrairement au C, au Pascal et aux autres langages de troisième génération qui ne sont pas spécialisés, il est conçu pour communiquer avec plusieurs types de bases de données avec les mêmes sources grâce à la couche ODI (Open Database Interface). Il est basé sur le RDSQL (Relational Database Structured Query Language), une extension du SQL. Il est par exemple possible de définir le type d'une variable en fonction d'un champ d'une table.

Un exemple simple de programme Genero 4GL, le classique Hello World (HelloWorld.4gl) :

```
MAIN
    DISPLAY « Hello World »
END MAIN
```

Figure 2 : Exemple simple de programme 4GL

I.7.2. Genero Clients and Services

Du fait de l'utilisation d'XML pour décrire la couche de représentation de Genero, il est possible d'afficher la logique métier sur des clients graphiques hétérogènes sans recompilation. Genero est ainsi entièrement graphique et aux clients Windows, Linux et Mac OSX, Four J's ajoute également une version web sous forme de CGI (Common Gateway Interface) ou filtre ISAPI (Internet Server Application Programming Interface), permettant de générer dynamiquement du HTML (HyperText Markup Language), à partir du Genero 4 GL original grâce à un moteur de rendu dynamique. La plupart des bases de données du marché sont supportées et des extensions services web permettent d'ouvrir Genero au monde extérieur. Enfin, un serveur d'application interfaçable avec la majorité des serveurs web du marché (Apache, IIS, J2EE [Java 2 Entreprise Edition]) facilite la communication client/serveur.

I.7.3. Genero Studio

Genero Studio est un environnement de développement intégré regroupant l'ensemble des outils nécessaires pour le développement d'applications Genero.

Genero Studio regroupe entre autres un éditeur de texte, un éditeur d'interfaces graphiques, un compilateur, des outils automatiques de fabrication, un débogueur et donne un accès direct aux schémas de bases de données. Genero Studio fournit donc les outils nécessaires pour développer la logique métier, pour générer facilement des écrans à partir de schémas de bases de données et répondre rapidement aux besoins d'un marché en constante évolution.

I.7.4. Genero db

Genero db est une base de données relationnelle proposant une architecture unique sans verrou offrant d'excellentes performances et pleinement intégrée dans l'écosystème Genero.

Genero db est compatible avec les normes SQL-92 et les extensions Oracle PL/SQL, Microsoft Transact-SQL et SPL Informix.

Au système de gestion de base de données s'ajoute un outil graphique d'administration permettant de gérer les bases de données de façon professionnelle appelé Genero db Entreprise Manager.

I.7.5. Genero Report Writer

Genero Report Writer est un outil qui permet de préparer des rapports et des graphiques très facilement, et cela avec une large quantité de données. Un éditeur visuel est également proposé.

I.7.6. Genero Evaluation Program

Genero Evaluation Program est une offre marketing regroupant l'ensemble des produits proposés par Four J's ainsi que des programmes de démonstration et des tutoriels. Disponible gratuitement et livrée avec une licence valable 90 jours, elle aide à faire la promotion de Genero.

I.8. Exemple d'utilisation

Un exemple d'utilisation de l'écosystème Genero peut être Peace Software, le leader mondial en développement de système de gestion de clients. Peace, son produit phare, est installé comme utilitaire principal sur 35 marchés de l'énergie hautement régulés et compétitifs et est utilisé pour la facturation et la gestion des relations clients pour l'électricité, le gaz et l'eau. Il dessert des

millions de clients sur différents marchés et gère des flux financiers de plusieurs milliards de dollars.

Récemment, Peace Software a remplacé le SI (Système d'Informations) hétérogène et hautement personnalisé d'Xcel Energy, héritage d'un rachat et d'une consolidation d'entreprise, par un SI unique fortement évolutif. Le nouveau système basé sur Genero sert 3,3 millions de clients en électricité et 1,8 million de clients en gaz répartis dans 10 États aux États-Unis. Le système génère 250 000 factures chaque jour ouvré, supporte 2,200 taux et traite 20 millions de dollars de paiements journaliers.

I.9. Données commerciales et financières

I.9.1. Chiffre d'affaires

Le chiffre d'affaires annuel de la société était de 18 millions de dollars en 2008, réparti géographiquement comme indiqué sur la figure 3.

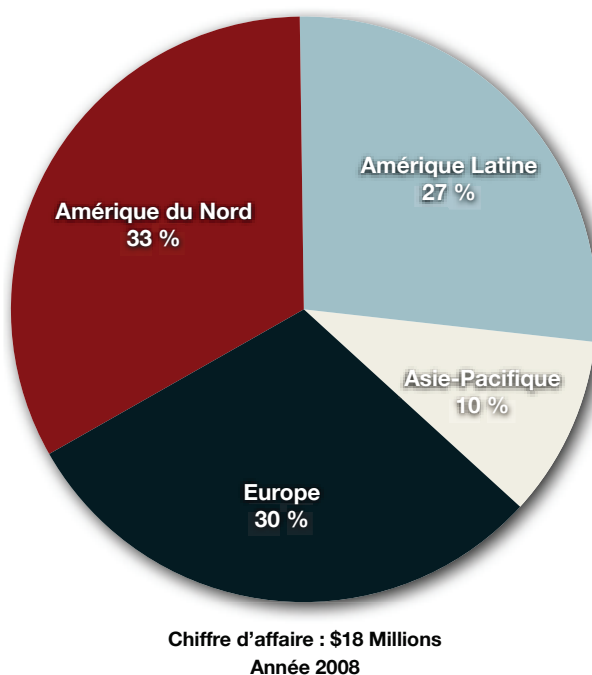


Figure 3 : Répartition du CA par continent pour l'année 2008

On constate que le chiffre d'affaires est réparti de façon assez homogène entre les différents continents. Il faut noter que Four J's a des clients en Afrique du Sud, mais que ceux-ci sont gérés par Four J's Royaume-Uni et qu'ils sont donc intégrés dans le chiffre d'affaires européen.

I.9.2. Marchés

Une des forces de Four J's est sa diversité, qu'elle soit géographique, qu'elle concerne le secteur industriel ou encore la taille de l'entreprise cliente. Four J's est présent sur quatre continents. Les clients de la société sont dans des secteurs aussi variés que : la banque, les assurances, l'énergie, la grande distribution, les télécommunications, les médias, l'industrie, le conditionnement, la défense, la santé ou les finances. Enfin, la taille des clients va de la PME/PMI locale aux multinationales du Fortune 500⁷.

La figure 4 représente la part de chaque secteur parmi les sociétés utilisant les produits de Four J's.

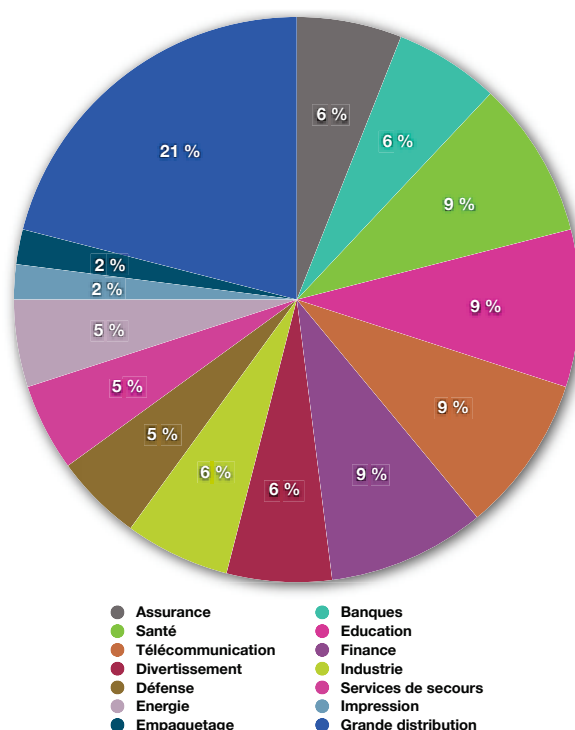


Figure 4 : Répartition des revenus par secteur d'activité pour l'année 2008

Four J's compte de prestigieuses sociétés utilisant ses produits. Des plus notables nous pouvons citer Sears et Wall-Mart, deux géants de la grande distribution, Q8 Petroleum et BP pour l'énergie, Sprint et AT&T dans les télécommunications, l'US Navy et le Ministère de la Défense espagnol au niveau des agences gouvernementales, Adidas, Reebok et HP pour les fabricants et la Société Générale dans le domaine bancaire.

⁷ Le Fortune 500 correspond aux 500 plus grosses entreprises mondiales.

I.9.3. Canaux de distribution

Four J's distribue ses produits de deux façons. La première représente environ 75 % des ventes et correspond aux sociétés de développement logiciel et aux sociétés de services qui proposent des solutions basées sur Genero aux clients finaux. La seconde, comptant pour environ 25 % des ventes, est la vente directe par Four J's de licences aux clients finaux qui développent alors leurs logiciels en interne.

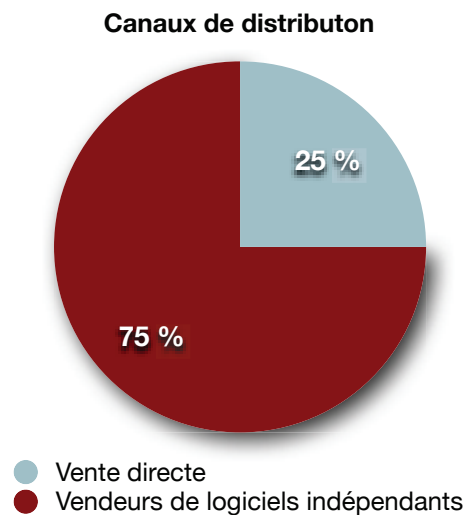


Figure 5 : Canaux de distribution des produits Four J's

I.9.4. Modèle économique

Le modèle économique est entièrement basé sur la vente de licence et de contrat de maintenance, et ce, de façon quasi égale comme nous pouvons le voir sur la figure 6.

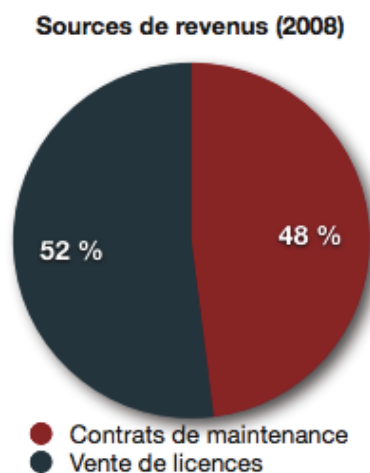


Figure 6 : Sources de revenus (2008)

Les licences sont facturées par utilisateur, par nombre de processeurs, par entreprise ou par abonnement. Les sociétés de services proposant des produits basés sur Genero sont une bonne source de revenus grâce aux licences qu’elles vendent aux clients finaux.

Les contrats de maintenance incluent l’assistance technique, les mises à jour et le support de nouvelles plateformes et systèmes d’exploitation. Un tiers des revenus générés par les contrats de maintenance proviennent de la première année de maintenance pour les nouveaux clients et les deux tiers restants viennent de contrats de maintenance à long terme.

1.9.5. Où est située la société

Le quartier général nord-américain est situé à Irving au Texas, et le quartier général européen est à Shannon, en Irlande. Four J’s possède 14 centres à travers l’Europe, les États-Unis, la zone APAC, l’Inde et l’Amérique latine.

Le centre de Strasbourg est le principal centre de recherche et développement de la société. Les trois autres centres de développement se trouvent en Inde, aux États-Unis et en Allemagne. Le centre de Strasbourg sert également de centre de support principal.

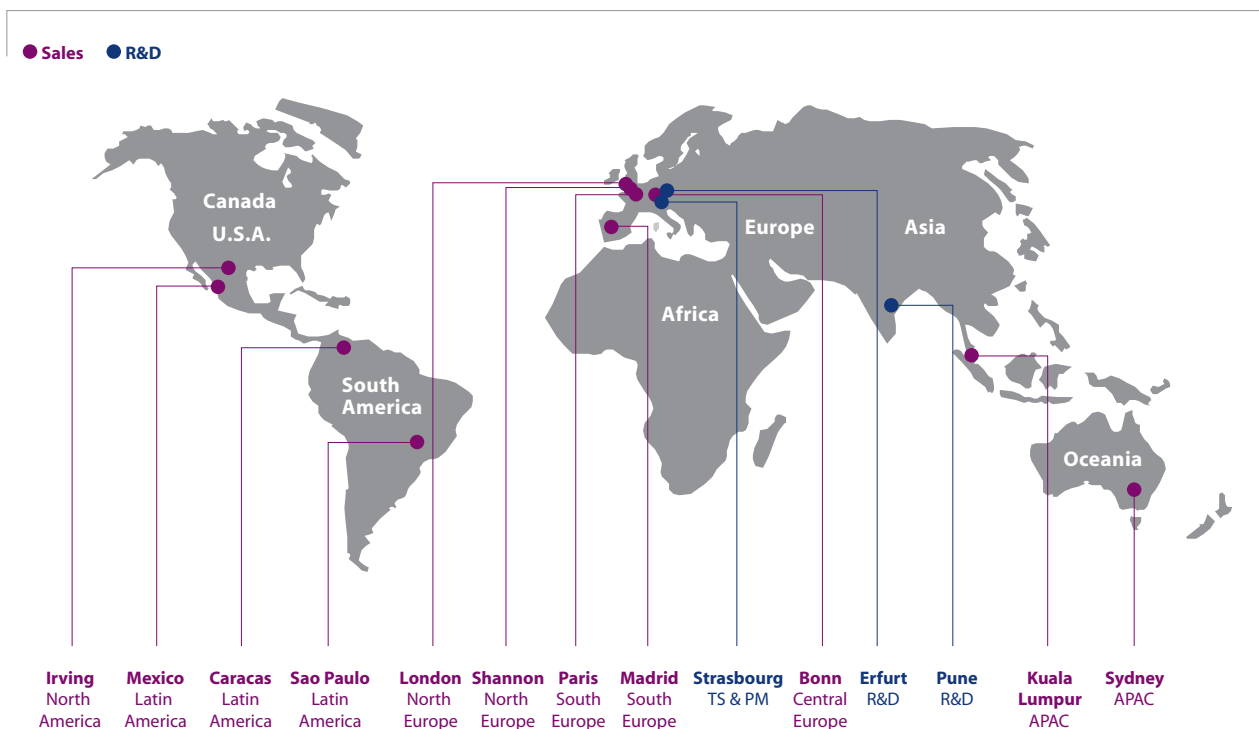


Figure 7 : Implantation de la société à travers le monde

I.10.Ressources humaines

En 2008, Four J's comptait 132 employés dans le monde, réparti comme suis entre les 4 pôles recherche et développement, vente et mercatique, assistance technique et financier et administratif.

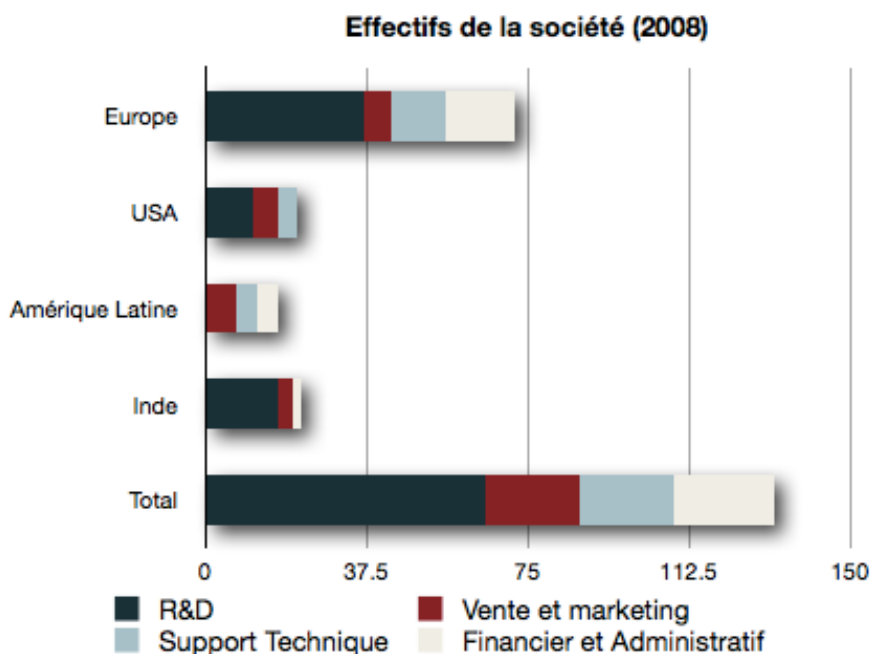


Figure 8 : Ressources humaines (2008)

I.10.1.Place du candidat dans l'organigramme

Je fais actuellement partie de la branche QA (Quality Assurance ou Assurance Qualité) pour Genero 4GL. Je suis en particulier chargé de tout ce qui touche aux tests du langage Genero avec les différentes bases de données du marché, et plus spécialement avec Genero db. Je rédige des plans de tests, m'occupe de leur implémentation. Je supervise également l'installation et la configuration des bases de données utilisées lors des tests.

Les types de tests effectués sont des tests fonctionnels, vérifiant le fonctionnement de nouvelles fonctionnalités ainsi que des tests de non-régression, vérifiant que le comportement attendu ne change pas entre les différentes versions du produit.

J'ai de surcroit participé ou dirigé plusieurs projets transversaux par exemple le GEP (Genero Evaluation Program), un produit regroupant l'ensemble de l'offre de la société Four J's permettant d'évaluer très simplement cette offre. J'ai enfin été responsable de la partie packaging

(procédures d'installation des produits) et réalisé la réécriture complète des scripts d'installations pour les plateformes Linux/UNIX.

Par souci de confidentialité, il m'est impossible de fournir l'organigramme de la société.

I.11.« Énergie » informatique disponible au moment du lancement du projet

L'activité de la société est le développement logiciel, l'infrastructure matérielle et logicielle dans le centre de recherche et développement de Strasbourg est conséquente.

Il y a des serveurs et machines faisant tourner la majorité des systèmes d'exploitation du marché ainsi que la majorité des SGBD. Chaque développeur et chef de projet a un vaste choix de logiciel à sa disposition.

I.12.Langages utilisés

De nombreux langages sont utilisés dans la société, mais les principaux sont :

- langages web : HTML, CSS (Cascading Style Sheet), JavaScript, XSL (eXtensible Stylesheet Language),
- langages de développement : QT, Genero 4GL, C, C++, C#, Java,
- langages de script : Python, Shell-Script,
- autres langages : XML, WiX (Windows Installer WML).

Le langage de programmation en lui-même est développé en C. Le C++ et Java sont utilisés pour les services web ainsi que pour le serveur d'application.

Les clients graphiques de bureau ainsi que Genero Studio, l'EDI (Environnement de Développement Intégré) et GDBEM (Genero db Entreprise Manager), l'outil d'administration de Genero db, sont développés en QT. Quant aux langages web, ils sont utilisés pour toute la partie client web.

Genero est utilisé en interne pour quelques outils comme la gestion des licences. Les langages de script sont utilisés pour toutes les parties portage, scripts d'installation et pour les validations et tests des produits de la société.

II. Le contexte du projet

II.1. Service dans lequel se déroule le projet

Le projet va se dérouler dans l'équipe GCS (Genero Clients and Services) qui est responsable de la partie concernant les clients graphiques ainsi que les services web et le serveur d'application.

L'équipe GCS a en charge le développement des produits Genero suivants :

- GDC (Genero Desktop Client) : un client graphique multiplateforme,
- GDCAX (Genero Desktop Client Active X) : un client graphique exploitant la technologie Active X de Microsoft,
- GWC (Genero Web Client) : un client graphique web,
- GAS (Genero Application Server) : un serveur d'application,
- GWS (Genero Web Services) : la gestion des services web.

Le projet sera supervisé par Olivier Imbert, directeur de projet de l'équipe GCS. Au moment du lancement du projet, je ne fais pas partie de l'équipe GCS.

II.2. La révolution mobile

Depuis quelques années, nous sommes entrés dans un nouveau cycle informatique majeur, le cinquième : l'Internet mobile.



Figure 9 : Les cycles informatiques de 1960 à nos jours

Les débuts de l'Internet mobile ont été poussifs, les premiers terminaux étant loin de ceux qui existent actuellement. Les écrans étaient petits, affichaient peu de couleurs et disposaient d'une faible résolution. De plus, l'accès au réseau était lent et les protocoles utilisés alors, le WAP (Wireless Application Protocol) et l'i-Mode, n'ont jamais décollé du fait d'une complexité élevée et

d'un potentiel faible. Enfin, les langages utilisés pour le développement d'applications mobiles⁸ étaient pauvres et ne permettaient pas de faire grand-chose.

Ces premiers pas de l'Internet mobile, durant la première moitié de la décennie, furent un échec commercial. Il a fallu une conjonction d'éléments pour transformer ces débuts mitigés en véritable révolution.

Le premier est l'émergence, à partir de 2003, du « Web 2.0 ». L'expression « Web 2.0 » désigne certaines des technologies et des usages du World Wide Web qui ont suivi la forme initiale du web, en particulier les interfaces permettant aux internautes ayant peu de connaissances techniques de s'approprier les nouvelles fonctionnalités du web. Ainsi, les internautes ont pu interagir (partager, échanger, etc.) de façon simple, à la fois avec le contenu et la structure des pages, mais aussi entre eux, créant ainsi notamment le Web social.⁹

À partir de là, l'usage de l'Internet s'est largement démocratisé, le nombre d'utilisateurs connaissant une croissance quasi exponentielle.

Néanmoins, pour que cette révolution passe aux terminaux mobiles, il fallait un appareil portable réellement adapté à cet usage.

Jusqu'à 2007, les principaux acteurs du marché des smartphones étaient Nokia et son OS (Operating System) mobile Symbian, Microsoft avec Windows Mobile et Research In Motion et son BlackBerry OS. Mais bien qu'assez capables, ces appareils n'étaient utilisés qu'au niveau professionnel, et pour des tâches simples comme la gestion des courriers électroniques, des calendriers et des tâches. La croissance de la vente de smartphones était bonne, mais n'avait pas encore atteint le point critique.

C'est en janvier 2007 que l'appareil qui allait changer la donne a été présenté au public. Steve Jobs, PDG d'Apple, a introduit le produit de la sorte « Your life in your pocket »¹⁰. Il s'agit de l'iPhone d'Apple, qui est sorti aux États-Unis le 29 juin 2007. Ce smartphone, protégé par Apple à l'aide de plus de 200 brevets, a tout de suite rencontré l'engouement du public. En effet, il n'a fallu à Apple que 3 mois pour vendre 1 million de terminaux aux États-Unis, alors même que les ventes n'étaient pas subventionnées par l'opérateur (AT&T) et donc que le coût de l'appareil était élevé (599 \$ pour le modèle 8 Go et 699 \$ pour le modèle 16 Go).

⁸ le cHTML (condensed HyperText Markup Language) et WML (Wireless Markup Language)

⁹ http://fr.wikipedia.org/wiki/Web_2.0

¹⁰ «Votre vie dans votre poche»

À ce moment, Apple ne proposait pas d'outil de développement. De fait, Steve Jobs avait mis en avant Safari mobile, le navigateur intégré à l'iPhone, comme point d'accès vers des sites web adaptés à l'usage mobile.

En septembre 2007, Apple a également sorti l'iPod touch, qui est pour résumer un iPhone sans la capacité de téléphoner, mais intégrant le WiFi et le même navigateur web que l'iPhone : Safari mobile.

Ces deux appareils, présentant une interface révolutionnaire entièrement tactile et un OS spécifiquement développé dans ce sens, ont été le deuxième facteur déterminant de la révolution mobile.

Il a fallu attendre presque un an et la sortie de la deuxième version de l'iPhone (iPhone 3G) le 11 juin 2008 pour qu'Apple fournisse aux développeurs la première mouture non bêta d'un SDK (Software Development Kit) leur permettant de développer des applications natives pour l'iPhone : xCode.

Par là même, Apple annonçait ce qui allait être le troisième élément de cette révolution mobile, l'App Store. C'est une plateforme de téléchargement d'applications offrant un endroit centralisé aux utilisateurs pour leurs achats et téléchargements d'applications.

Le 10 juillet 2008, l'App Store était lancé et 500 applications étaient initialement disponibles. Le succès fut immédiat, avec 10 millions d'applications téléchargées les 3 premiers jours. En décembre 2008, ce sont 10 000 applications qui sont disponibles et le nombre d'applications téléchargées s'élève à 300 millions. En juin 2009, 50 000 applications sont disponibles et la barre d'un milliard d'applications téléchargées est franchie.

C'est en comparant l'adoption et le développement de l'Internet mobile à celui de l'Internet que l'on se rend compte de la tendance qui émerge, la convergence. De plus, l'Internet mobile sera bien plus grand que beaucoup ne le pensent, du fait de 4 tendances qui convergent : la 3G qui offre une vitesse de connexion élevée aux terminaux mobiles, les réseaux sociaux qui font utiliser l'Internet au plus grand nombre, la VoIP (Voice over Internet Protocol) et des terminaux mobiles de plus en plus puissants et capables.

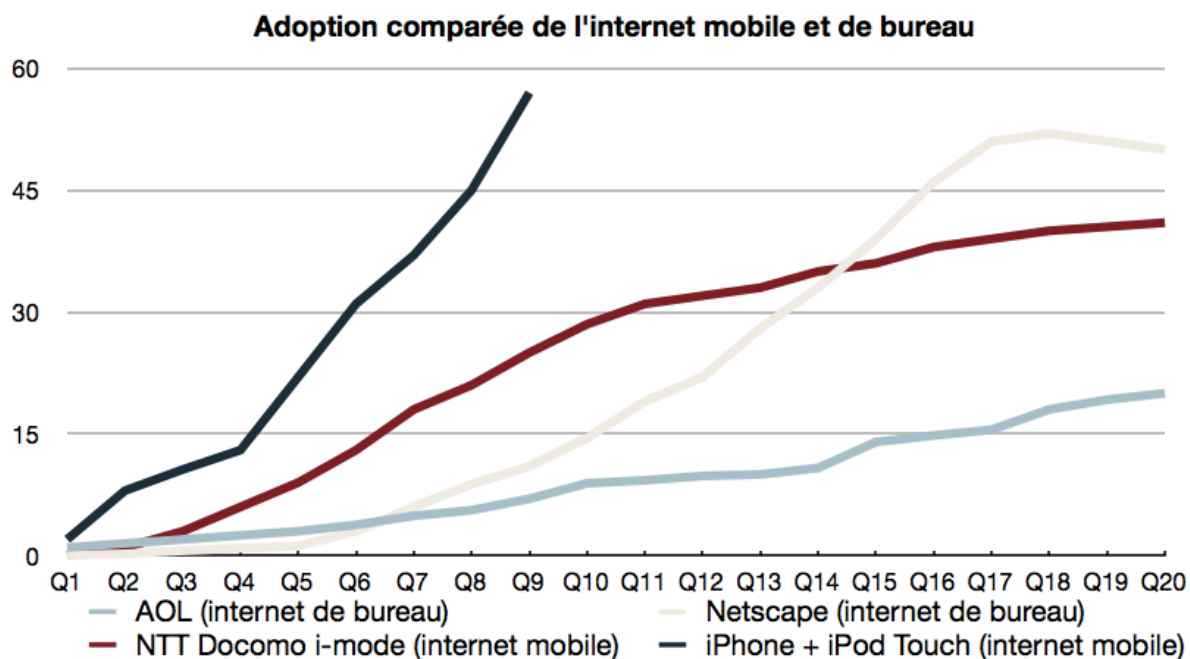


Figure 10 : Les rythmes d'adoption comparés de différentes technologies de l'Internet de bureau et mobile¹¹

De surcroît, depuis le lancement de l'iPhone puis de l'App Store, les autres acteurs de l'Internet, de l'informatique et des télécommunications ont suivi le mouvement. Les téléphones mobiles tactiles sont de plus en plus répandus, les OS mobiles similaires à iPhone OS se développent (je pense particulièrement à Android de Google, à WebOS de Palm et à Windows Phone 7 de Microsoft) et les dépôts d'applications similaires à l'App Store se multiplient (Nokia a lancé son OviStore, Google l'Android Marketplace, Microsoft le Windows Marketplace et Research In Motion l'AppWorld).

2009 est une année importante pour l'Internet mobile, avec une évolution d'un marché de matériel, vers celui des logiciels et des services.¹²

Pour finir, j'aimerais noter que c'est tout le secteur de l'Internet mobile et des smartphones qui bénéficie de cette révolution mobile, comme l'indiquent les chiffres de la croissance des ventes de smartphones depuis 2005, et plus particulièrement entre 2008 et 2009, et cela malgré le contexte de crise économique mondiale.

¹¹ <http://www.businessinsider.com/mary-meecker-mobile-internet?op=1>

¹² Morgan Stanley Internet Trends

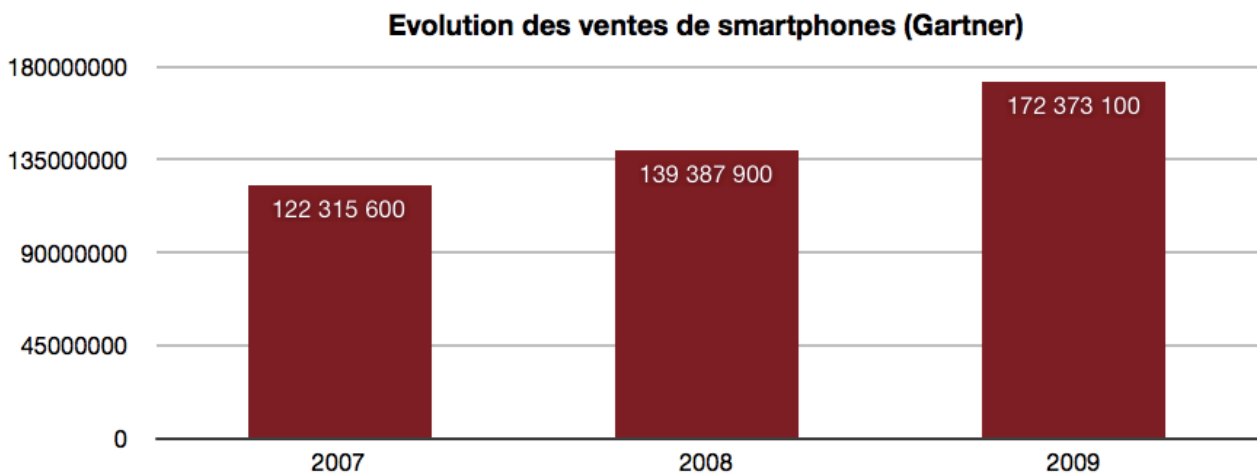


Figure 11 : Évolution des ventes de smartphones au niveau mondial (sources Gartner¹³)

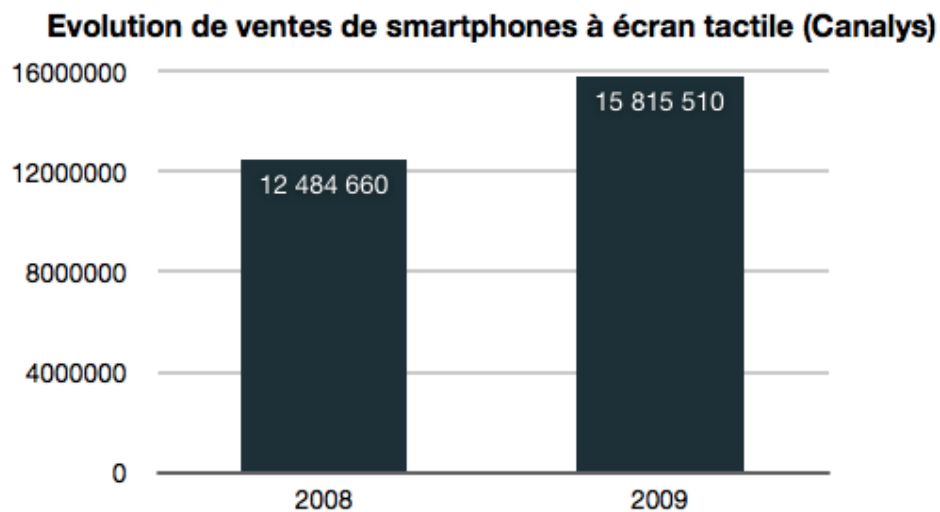


Figure 12 : Évolution des ventes de smartphones à écran tactile au niveau mondial (source Canalys¹⁴)

II.3.L'iPhone

Je vais maintenant présenter plus en détail l'iPhone, élément clef de la révolution mobile.

II.3.1.L'iPhone en chiffre

Dans cette partie, je vais détailler les chiffres les plus significatifs relatifs à l'iPhone.

¹³ <http://www.gartner.com>

¹⁴ <http://www.canalys.com/>

II.3.1.1. Évolution des ventes de smartphones

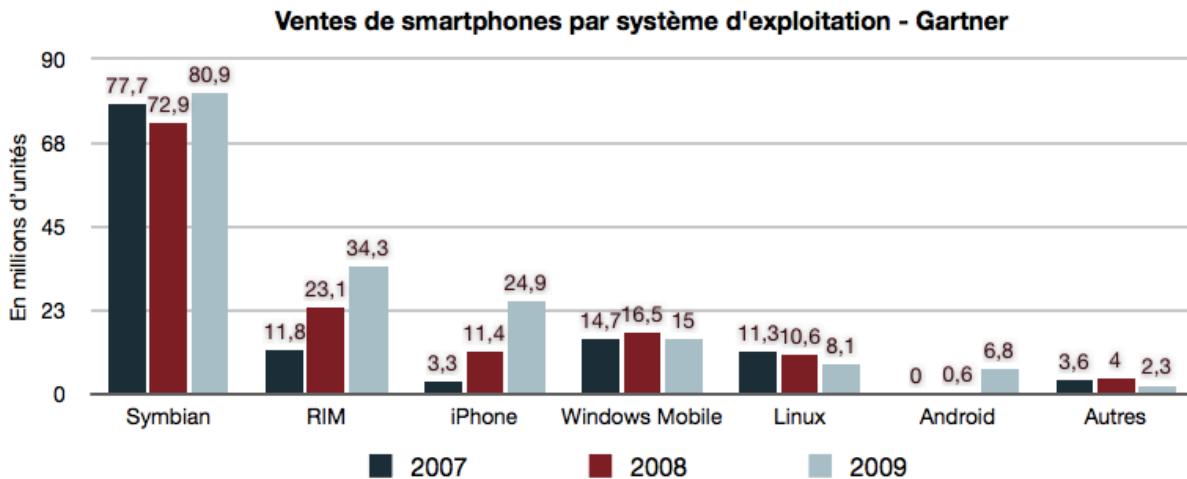


Figure 13 : Ventes mondiales de smartphones par système d'exploitation (sources Gartner)

D'après les chiffres de Gartner¹⁵, concernant les ventes mondiales de smartphones, on peut voir qu'entre 2007 et 2009, les ventes d'iPhone ont connu une croissance quasi exponentielle, passant de 3 millions en 2007 à presque 25 millions en 2009. L'iPhone est un des smartphones qui connaît la plus belle croissance.

De surcroît, au vu de l'évolution de ce marché sur les dernières années (figures 11 et 12), les prévisions de croissance, malgré le contexte de crise économique, sont bonnes.

II.3.1.2. Répartition des smartphones connectés à Internet

Afin d'analyser la répartition des smartphones connectés à Internet, deux graphiques sont à notre disposition : celui de StatCounter (figure 14) et celui de NetApplication (figure 15). Dans les deux cas, l'iPhone OS et Safari mobile arrivent assez loin devant en terme d'usage.

L'iPhone est largement en tête d'usage mobile pour accéder à l'Internet. On peut en conclure que les gens qui possèdent un iPhone sont plus enclins à se connecter à Internet que les gens qui possèdent un autre smartphone. Ceci est en particulier dû à la facilité d'utilisation du navigateur Safari mobile.

¹⁵ <http://www.gartner.com>

Parts de marché des navigateurs mobile - Q4 2008 à Q4 2009 (StatCounter)

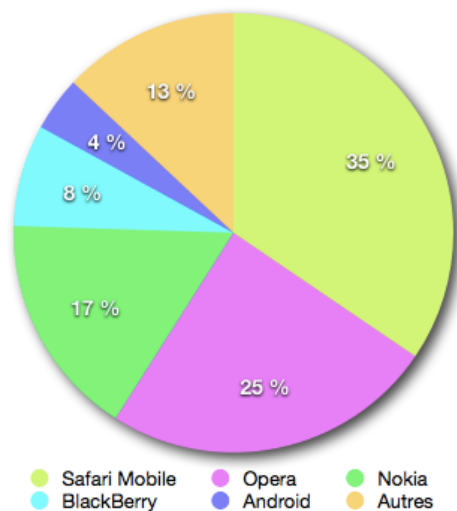


Figure 14 : Parts de marché des navigateurs mobile — Q4 2008 à Q4 2009 (sources StatCounter¹⁶)

Parts de marché des navigateurs mobile - 2009 (NetApplications)

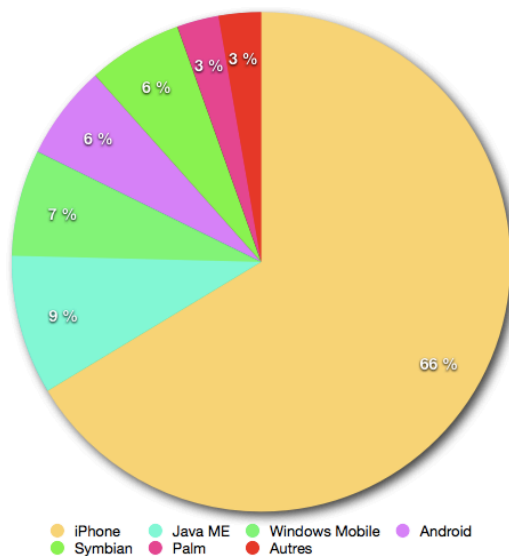


Figure 15 : Parts de marché des navigateurs mobile — 2009 (NetApplications¹⁷)

II.3.1.3. Attractivité

D'après Flurry, une société analysant les applications mobiles, que se soit en terme d'attractivité pour les développeurs ou pour les consommateurs, l'iPhone est loin devant.

Ainsi bien aussi bien les développeurs que les clients plébiscitent l'iPhone et iOS.

¹⁶ <http://statcounter.com/>

¹⁷ <http://www.netapplications.com/>

Attractivité pour les développeurs (Flurry)

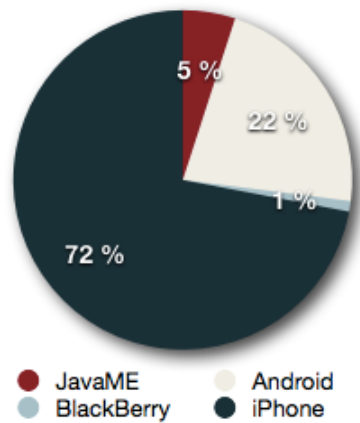


Figure 16 : Attractivité pour les développeurs (source Flurry¹⁸)

Attractivité pour les clients (Flurry)

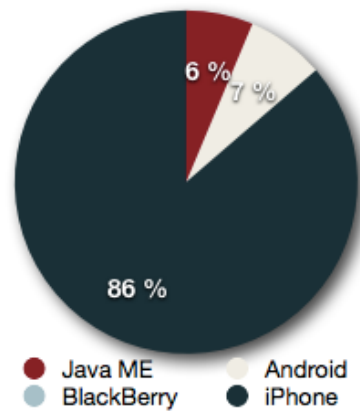


Figure 17 : Attractivité pour les clients (source Flurry¹⁹)

II.3.2. Données techniques

Je vais présenter les informations techniques qui nous intéressent, c'est-à-dire celle concernant l'interface tactile, et celles concernant le navigateur web utilisé par iPhone OS.

II.3.2.1. Un OS tactile multipoint

Apple avec son iPhone a pris le parti de se passer de clavier physique pour offrir un écran entièrement tactile disposant d'un clavier virtuel.

Pour cela, Apple a développé un système d'exploitation basé sur Mac OS X, le système d'exploitation de ses ordinateurs, et l'a appelé iPhone OS²⁰. iPhone OS partage les fondations de

¹⁸ <http://www.tipb.com/2009/04/22/iphone-undisputed-king-smartphone-app-mountain/>

¹⁹ <http://www.tipb.com/2009/04/22/iphone-undisputed-king-smartphone-app-mountain/>

²⁰ Le nom a changé le 7 juillet 2010 pour iOS.

Mac OS X, c'est-à-dire le noyau hybride XNU basé sur le micronoyau Mach et les services UNIX et Cocoa. iPhone OS comporte quatre couches d'abstraction, similaires à celles de Mac OS X : une couche « Core OS », une couche « Core Service », une couche « Media » et une couche « Cocoa »²¹.

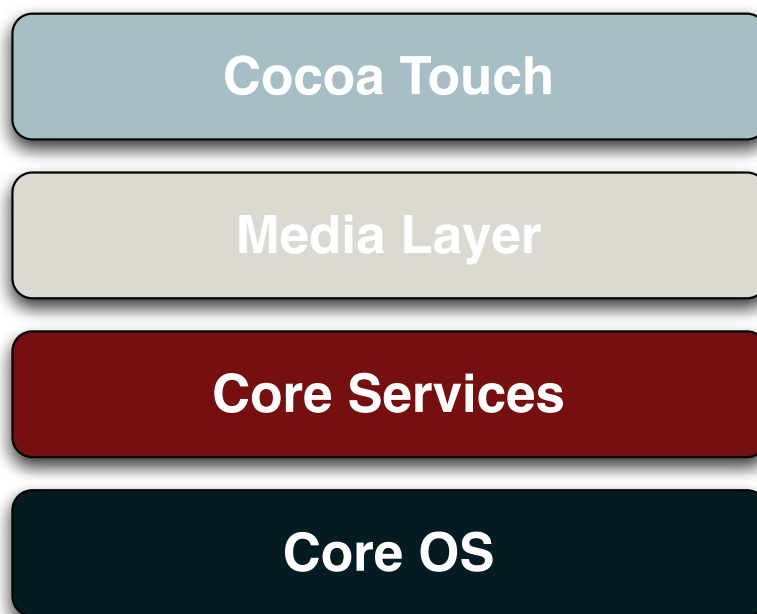


Figure 18 : Couches d'abstraction du système d'exploitation pour mobile iOS

L'interface du système d'exploitation iPhone OS est basée sur le concept de la manipulation par contact tactile de l'écran. Elle se révèle particulièrement fluide et réactive, incluant notamment la technologie multipoint, permettant de reconnaître des gestes à plusieurs doigts simultanés.

Le tableau I introduit les principales interactions tactiles disponibles.

Nom	Action	Geste
Le 'tap'	Contact simple rapide de l'index sur l'écran. Il s'agit du geste de base des interfaces tactiles (tablettes, smartphones) et équivaut au clic simple avec une souris (clic gauche). Il permet généralement de sélectionner un élément à l'écran ou d'activer un contrôle (validation de formulaire, bouton radio, case à cocher, etc.).	
Le 'double tap'	N'est pas nécessairement au tap ce que le double clic est au clic. Si ce mouvement peut servir à ouvrir un élément sélectionné sur certains environnements, il est souvent utilisé par les éditeurs de services pour effectuer un zoom sur une image.	
Le 'press'	Contact prolongé d'un doigt sur l'écran. Il peut notamment être utilisé pour afficher des informations contextuelles (menu, etc.) concernant un élément. Il permet également la sélection d'un élément affiché à l'écran en vue d'un déplacement de cet élément.	

²¹ http://fr.wikipedia.org/wiki/IOS_%28Apple%29


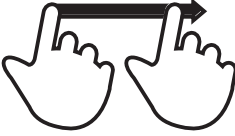
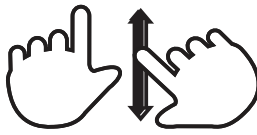
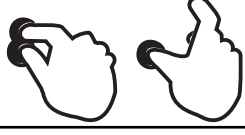

Nom	Action	Geste
Le 'flick' ('chiquenaude' en français)	Correspond à un balayage rapide du doigt sur l'écran. Ce geste permet de faire défiler le contenu d'un écran de manière accélérée. Sur les interfaces tactiles, cette action de scroll peut être utilisée aussi bien verticalement qu'horizontalement.	
Le 'drag' ('glissement' en français)	Consiste à maintenir et à faire glisser un doigt sur l'écran. Comme le flick, ce geste est utilisé pour effectuer un défilement du contenu affiché à l'écran, mais de manière plus lente et plus précise, dans le cadre de la lecture d'un livre électronique par exemple. Il peut également être utilisé pour des actions de glisser-déplacer. Ce geste peut être effectué avec plusieurs doigts.	
Le 'press and drag' ('presser et déplacer')	Consiste en un contact prolongé avec un doigt et un déplacement simultané (horizontal ou vertical) avec un autre doigt. Ce mouvement peut être effectué avec une seule main (le pouce sert à presser et l'index à déplacer) ou avec deux mains sur des tablettes (l'index de la main gauche sert à presser et celui de la main droite à déplacer). Il peut notamment permettre de faire défiler une sous-zone d'une zone déjà scrollable.	
Le 'spread' ('écartement' en français)	Consiste à maintenir appuyés et à éloigner le pouce et l'index l'un de l'autre (sur une tablette, il peut s'agir des deux index). Ce geste est essentiellement utilisé pour effectuer un zoom sur un élément. Il s'agit d'un geste typique de l'environnement iOS.	
Le 'pinch' ('pincement' en français)	Correspond à un rapprochement du pouce et de l'index sur un élément de l'écran. Ce geste est essentiellement utilisé pour effectuer un zoom arrière sur un élément qui peut être agrandi ou rétréci. Il s'agit de l'inverse du spread.	

Tableau I : Les gestes multipoints de base

II.3.2.2.Safari mobile

Safari mobile est un navigateur Web développé par Apple, dans une version adaptée à iPhone OS, et son interface multipoint. La partie supérieure de l'écran y est composée d'une barre d'adresse, dans laquelle il est possible d'entrer directement une adresse par le clavier virtuel, et d'une barre de recherche.

Safari mobile est comme son grand frère Safari basé sur WebKit, un moteur de rendu open source et développé par Apple à partir d'un fork de KHTML, le moteur de rendu utilisé par le projet KDE. WebKit est également utilisé par Google dans son navigateur web Google Chrome, et dans le navigateur web utilisé par Android, son système d'exploitation pour smartphone.

Safari mobile exploite à merveille l'interface multipoint de l'iPhone. Ce navigateur web est reconnu pour sa simplicité d'utilisation, et pour son ergonomie. En effet, l'utilisateur y explore des pages web en glissant simplement son doigt sur l'écran, et peut zoomer ou dézoomer sur le contenu en écartant ou pinçant deux doigts. De plus, le mode paysage est également géré pour cette application, par une simple rotation de l'appareil. On notera également la présence d'un zoom

intelligent. Un double tapotement sur du contenu adaptera automatiquement le zoom à la largeur du bloc de texte, de l'image, ou de n'importe quel contenu.

En outre, il dispose des fonctions basiques d'un navigateur : aller à la page précédente ou suivante, arrêter le chargement d'une page, rafraîchir celle-ci, un gestionnaire de marque-pages, et d'historique. Il est également possible d'ajouter ses pages web favorites à l'écran d'accueil de l'appareil, sous forme d'une icône identique à celle des applications. Enfin, il est possible d'ouvrir plusieurs onglets à la fois, et de naviguer entre eux par l'icône située dans le coin inférieur droit de Safari²².

En revanche, Safari ne permet pas le téléchargement de fichiers (doc, PDF, MP3, etc.) dans la mémoire de l'appareil, hormis les fichiers image. Il ne permet pas non plus la visualisation de contenu Flash.

II.4.L'App Store

L'App Store est une plateforme de téléchargement d'applications mobiles lancée le 11 juillet 2008 et distribuée par Apple sur les appareils fonctionnant sous iPhone OS.

Ce logiciel s'intègre dans le service iTunes Store et permet de télécharger des applications tierces.

L'App Store connaît une croissance impressionnante. En effet, lors de la première semaine de décembre 2008, soit 6 mois après son lancement, Apple annonce que l'App Store propose 10 000 applications et revendique plus de 300 millions de téléchargements.

II.4.1.L'App Store en chiffre

Durant la première année de fonctionnement du service, entre juillet 2008 et 2009, un milliard cinq cents millions d'applications avaient été téléchargées depuis le portail App Store, parmi un catalogue de plus de 65 000 applications validées par Apple. Ces chiffres furent publiés au 11 juillet 2009 par Apple, à l'occasion du premier anniversaire du service. Le milliard d'applications avait lui été dépassé en avril 2009, pour 50 000 applications disponibles en juin.

Au 9 septembre 2009, la firme a annoncé un cumul d'un milliard huit cents millions de téléchargements depuis l'ouverture du service, parmi un catalogue proposant plus de 75 000 applications. Mais moins de trois semaines plus tard, l'entreprise annonce de nouveaux chiffres : parmi 85 000 applications, les deux milliards de téléchargements sont dépassés le 28 septembre

²² http://fr.wikipedia.org/wiki/IOS_%28Apple%29

2009. Deux nouveaux caps sont franchis par la suite avec l'annonce le 4 novembre de la disponibilité de 100 000 applications, et le cap des 3 milliards de téléchargements franchi le 5 janvier 2010. Enfin, en juillet 2011, c'est la barre des 15 milliards d'applications téléchargées qui a été franchie.

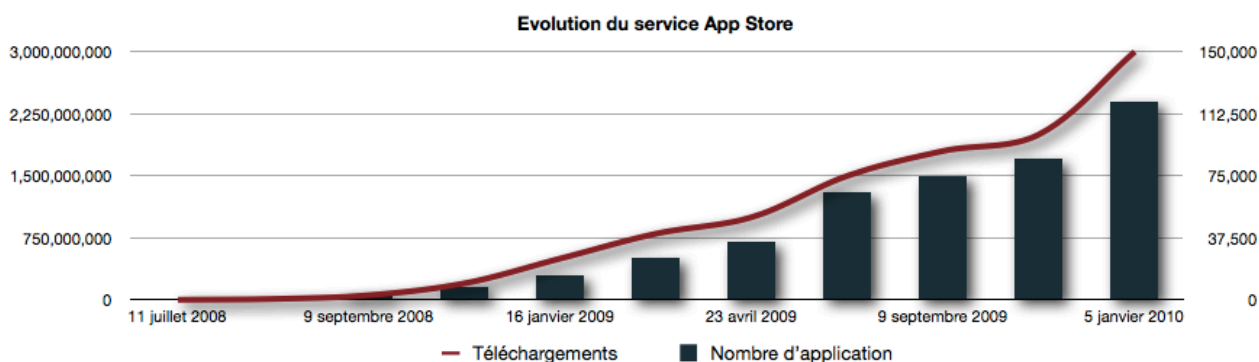


Figure 19 : Évolution du service App Store²³

II.4.2. Caractéristiques

Je vais présenter ici les caractéristiques principales de l'App Store.

II.4.2.1. Marché unique

L'App Store offre un endroit unique aux développeurs pour soumettre leurs applications. Le processus de soumission est transparent et le marché est unique. L'App Store est le seul endroit où télécharger des applications.

Les applications sont certifiées du fait qu'elles sont développées avec les outils fournis par Apple.

Ces mêmes outils permettent une façon simple d'identifier le mobile ainsi que la région du globe d'où sont téléchargées les applications.

Enfin, l'App Store fournit aux développeurs un moyen de fixer le prix de vente des applications.

II.4.2.2. Centralisation des paiements

Le mécanisme pour traiter le paiement d'une application est transparent. Apple offre également un moyen aux développeurs de suivre l'évolution des ventes de leurs applications et les revenus dégagés.

²³ http://en.wikipedia.org/wiki/App_Store

II.4.2.3.Distribution globale

Quel que soit l'opérateur téléphonique de l'utilisateur, le canal de distribution est le même. Il s'agit de l'App Store. Ensuite, bien que les achats et téléchargements se font directement dans l'App Store, Apple fournit des API (Application Programming Interface) pour permettre à des sites web de proposer les applications de façon indépendante.

II.4.2.4.Mise à disposition

Les applications sont licenciés à l'utilisateur réalisant l'achat. L'utilisateur pourra ensuite installer les applications achetées sur 5 appareils.

L'installation des applications se fait directement dans le système de fichiers de l'appareil.

Enfin, la mise à jour des applications est facilitée et se fait directement à travers l'App Store, l'utilisateur étant notifié des mises à jour.

II.4.2.5.Recherche directe sur le mobile

Apple fournit une application pour iPhone permettant de rechercher et d'installer directement des applications depuis son mobile. L'application gère un système de notation des applications par les utilisateurs et permet à ces derniers de laisser des commentaires.

Les applications y sont rangées par catégories et certaines applications sont mises en avant par Apple dans une partie « applications préférées ». Enfin, sont également mises en avant les applications les plus vendues et téléchargées de chaque catégorie.

II.4.3.Conditions d'utilisations

Une fois le SDK pour l'iPhone OS gratuitement téléchargé, il est possible de développer des applications et de les tester sur un Mac à l'aide d'un émulateur. Cependant, pour les mettre à la disposition des utilisateurs d'iPhone ou d'iPod touch, il est nécessaire de devenir officiellement développeur pour iPhone, en participant au programme de développement iPhone. Ceci peut se faire par le site des développeurs d'Apple.

II.4.4.Distribution des applications

II.4.4.1.Mode de distribution

Apple met à disposition des développeurs 2 types de licences pour distribuer des applications, que ce soit par le biais de l'App Store ou directement au sein d'une entreprise.

Le tableau II synthétise les contenus des 2 licences.

iPhone Developer Program	iPhone Developer Entreprise Program
Prix : 99 \$ par an.	Prix : 299 \$ par an.
Permet de soumettre autant d'applications que l'on veut sur l'App Store.	Réservé aux entreprises de plus de 500 salariés.
Apple se réserve le droit de refuser l'application et prend 30 % des bénéfices liés aux ventes du programme.	Possibilité de distribution en mode 'in house' (développement spécifique destiné à l'usage exclusif de l'entreprise).
Possibilité de distribution en mode 'ad-hoc' sur 100 iPhone au maximum sans soumettre l'application à Apple.	Les application 'in house' ne nécessitent pas de validation de la part d'Apple.

Tableau II : Les programmes développeurs Apple

On voit qu'il est possible dans les deux cas de ne pas publier en passant par l'App Store.

C'est le mode de distribution « Ad Hoc » dans le cas d'une licence basique ou « in House » avec le programme entreprise. Dans ces contextes, Apple fournit simplement au développeur une clé de chiffrement à associer au programme et le développeur peut déployer son application.

Notons qu'avec l'arrivée de l'iOS 4, il va être possible de proposer des applications « in House » téléchargeables sans passer par iTunes, ce qui n'est pas le cas aujourd'hui. Un collaborateur pourra ainsi télécharger directement une application sur l'intranet de son entreprise sans avoir à brancher son iPhone à son ordinateur.

II.4.4.2. Règles de validation

Apple considère que les applications sont différentes des livres ou des chansons, qu'Apple ne régule pas. Si un développeur veut critiquer une religion, Apple lui conseille d'écrire un livre. Si un développeur veut décrire l'acte sexuel, Apple lui conseille d'écrire un livre ou une chanson, ou de créer une application médicale. Apple a décidé de ne pas permettre certains types de contenus dans l'App Store.

Apple dans son contrat de licence qui l'associe aux développeurs, donnent quelques directives dont voici les plus intéressantes :

- les applications qui reproduisent des applications déjà disponibles sur l'App Store pourront être rejetées, particulièrement s'il y en a déjà beaucoup,
- les applications qui sont conçues pour tromper et manipuler ou offrir de fausses fonctionnalités sans être pour autant clairement indiquées comme telles seront rejetées,

- toute application qui est diffamatoire, offensante, de mauvais esprit, ou susceptible de mettre en danger l'individu ou le groupe visé sera rejetée,
- les applications offrant l'accès à du contenu généré par les utilisateurs qui est fréquemment pornographique (par exemple les applications « Chat Roulette ») seront rejetées,
- les développeurs qui « spamment » l'App Store avec de nombreuses versions d'applications similaires seront bannis du programme développeur iOS,
- les développeurs qui tentent de manipuler ou bien de tricher avec les avis des utilisateurs ou le classement dans l'App Store avec de fausses critiques des critiques payées, ou toute autre méthode inappropriée, seront bannis du programme développeur iOS,
- les développeurs qui tentent de collecter, associer, récolter, ou exploiter de quelque manière que ce soit l'identification des joueurs sur le Game Center, ou toutes autres informations à leur sujet seront bannis du programme développeur iOS.

II.4.5.Développement

Les applications sont développées avec le SDK d'Apple, mis à disposition gratuitement aux développeurs. L'outil principal se nomme xCode et fonctionne uniquement sur les machines Apple tournant sous Mac OS X.

Le langage de programmation est l'Objective-C, un langage orienté objet réflexif utilisé quasi exclusivement dans le monde Mac. C'est une extension du C ANSI, comme le C++, mais qui se distingue de ce dernier par sa distribution dynamique des messages, son typage faible, son typage dynamique et son chargement dynamique. Contrairement au C++, il ne permet pas l'héritage multiple²⁴.

Je vais expliquer point par point cette définition.

L'Objective-C est un langage orienté objet qui intègre une grande partie des notions de la programmation orientée objet. C'est un langage réflexif, cela signifie qu'il a une capacité à se modifier à l'exécution. Par exemple, le type d'une variable peut changer à l'exécution en fonction des actions de l'utilisateur.

L'Objective-C est une extension du C. Cela veut dire que l'Objective-C se construit autour des mécanismes du C ; il hérite donc de sa syntaxe. Mais l'Objective-C est une stricte surcouche du

²⁴ <http://fr.wikipedia.org/wiki/Objective-C>

C. Cela veut dire que n'importe quel code écrit en C est compilable par un compilateur Objective-C. Il n'y a aucune incompatibilité entre le C et l'Objective-C, contrairement au C++.

L'Objective-C utilise un système de messages : c'est la clé de voûte de l'Objective-C. C'est une partie du langage qui est héritée de Smalltalk. Les messages servent principalement à la programmation orientée objet.

L'Objective-C possède un typage dynamique et faible : le type d'une variable est défini à l'exécution, et il peut changer en cours de route.

L'Objective-C possède un système de chargement dynamique. Cela veut dire que l'Objective-C s'exécute dans un moteur d'exécution, à la manière de Java. Il y a donc une machine virtuelle qui se charge de distribuer les messages, de créer les objets et les classes, d'évaluer le type des variables, etc. De plus, contrairement à Java, elle est préinstallée et reste donc totalement invisible à l'utilisateur.

II.5. Les applications mobiles

Les écrans des smartphones sont petits, c'est pourquoi lorsque l'on développe une application mobile, il faut se concentrer sur des applications destinées à gérer une ou deux tâches, et pas plus. Une application mobile devrait proposer une solution efficace à un problème précis.

Néanmoins, une application mobile incorpore beaucoup des caractéristiques des applications traditionnelles et en suit les mêmes principes de conception et de développement.

Pour développer des applications mobiles offrant une grande expérience utilisateur, il faut avoir à l'esprit les éléments suivant : la simplicité, la focalisation sur les fonctionnalités essentielles de l'application et l'affichage clair des informations à l'écran.

Par exemple, une façon d'arriver à la simplicité est d'éviter d'encombrer l'interface utilisateur de trop d'éléments visuels qui réduisent l'attention de l'utilisateur. Dans une application mobile, cela pourra prendre les formes suivantes : une réduction du nombre de contrôles, des textes moins longs affichés, la suppression des éléments purement décoratifs et un espacement suffisant entre les contrôles.

II.5.1. Simplicité et facilité d'usage

La simplicité et la facilité d'utilisation sont des principes fondamentaux pour créer tous types de logiciels, mais pour les applications mobiles, ces principes deviennent critiques. Les utilisateurs doivent comprendre facilement comment utiliser l'application.

Au fur et à mesure de la conception de l'application et de son interface utilisateur, il faut suivre ces lignes directrices : rendre évidente l'utilisation, éviter l'encombrement, les espaces vides inutilisés et les fonds visuellement intrusifs, réduire le nombre de données à saisir par l'utilisateur, afficher les informations essentielles de façon succincte, proposer des contrôles et des liens adaptés à l'usage tactile et éviter les interactions homme-machine superflues.

II.5.1.1. Rendre évident

Dans une application mobile, la fonction principale de l'application doit être immédiatement apparente. Il est possible d'arriver à cela en minimisant le nombre de contrôles et en les nommant clairement de façon à ce que l'utilisateur comprenne exactement ce qu'il fait [APPL].



Figure 20 : Capture de l'application Chronomètre d'Apple

II.5.1.2. Éviter l'encombrement visuel

Vu sur un petit écran, l'effet négatif de l'encombrement visuel est amplifié, rendant des applications qui pourraient être acceptables sur un écran de PC très difficiles à utiliser sur un smartphone.

Dans une application mobile, il est important d'éviter de surcharger l'interface avec une profusion d'images et d'éléments. L'espace étant réduit sur l'écran des smartphones, il ne faut afficher que les éléments qui fournissent des informations essentielles et n'avoir que des fonctionnalités liées à un contexte précis. Il faut, de plus, éviter d'afficher les éléments et les images qui sont purement décoratifs.

Il est également important d'éviter de laisser trop d'espaces vides autour du contenu de l'application. En effet, s'il y a trop d'espace entre les éléments, l'utilisateur devra faire défiler la page pour accéder à chaque information, et ne pourra pas avoir une vue globale autrement qu'en faisant un zoom arrière. Il ne faut utiliser les espaces vides que pour faciliter la saisie entre chaque élément, pas plus [APPL].

II.5.1.3. Réduire la saisie

Saisir des informations prend du temps et de l'attention, que l'utilisateur touche les contrôles ou utilise le clavier virtuel. Si l'application demande beaucoup de saisies utilisateurs, que ce soit en une fois ou en plusieurs, cela va ralentir l'utilisateur et limiter l'expérience qu'il aura avec l'application.

Bien évidemment, certaines informations utilisateurs sont requises pour les besoins de l'application, mais il faut équilibrer cela avec ce qu'offre l'application en retour. En d'autres termes, il faut chercher à fournir autant d'informations et de fonctionnalités que possible pour chaque saisie de l'utilisateur.

Par exemple, une application de bureau aidant l'utilisateur à acheter un ticket de concert demanderait d'abord de l'utilisateur les entrées suivantes : nom de l'artiste, date du concert et lieu du concert avant d'afficher une liste de concerts répondant à ces critères. Puis l'application offrirait la possibilité à l'utilisateur d'acheter les tickets qu'il a sélectionnés. Néanmoins, dans une application mobile qui proposerait la même fonction, il serait préférable d'éviter à l'utilisateur de saisir autant d'informations avant qu'il n'arrive au but principal de l'application qui est d'acheter des tickets de concert. Ainsi, l'application mobile pourrait afficher dès le lancement une courte liste des concerts les plus populaires prévus ces prochaines semaines. De cette façon, la plupart des utilisateurs auraient déjà une partie de l'information recherchée avant même d'avoir saisi quoi que ce soit comme données.

Lorsque des entrées utilisateurs sont requises, il faut autant que possible privilégier l'utilisation d'une table ou d'un menu déroulant plutôt que d'un champ de saisie de texte, car il est plus facile de choisir un élément d'une liste que de saisir des mots sur le clavier virtuel [APPL].

II.5.1.4. Afficher les informations essentielles

Quand les textes affichés sur l'interface utilisateur sont courts et directs, les utilisateurs peuvent rapidement et facilement les comprendre. C'est pourquoi il faut identifier les informations

les plus importantes, les exprimer de façon concise, et les afficher de manière proéminente pour que les utilisateurs n'aient pas à lire trop de mots avant de trouver ce qu'ils cherchent ou de comprendre ce qu'ils doivent faire [MIND].

Pour réussir dans cette tâche, il faut penser comme un éditeur de journal papier, et chercher à apporter l'information d'une façon condensée, comme pour les titres des articles. Il faut donner aux contrôles les labels courts ou utiliser des symboles bien compris de tous de façon à ce que les utilisateurs comprennent leurs usages du premier coup.

II.5.1.5. Être adapté à l'usage tactile

Si la disposition des liens et des contrôles est trop rapprochée, les utilisateurs perdront du temps et de l'attention à regarder lorsqu'ils touchent l'écran et auront de fortes chances de se tromper involontairement d'élément [APPL].

On peut voir dans la figure 21 une capture de l'application calculatrice d'Apple. Les contrôles y sont suffisamment grands et suffisamment espacés pour un usage tactile optimal.



Figure 21 : Capture de l'application Calculatrice d'Apple

Il faudrait de surcroît respecter les consignes données dans le Apple Human Interface Guidelines, un guide proposé par Apple regroupant un ensemble de meilleures pratiques autour du sujet de la création d'applications.

II.5.1.6. Éviter les interactions superflues

L'interactivité est souvent utilisée pour éveiller l'intérêt des utilisateurs et leur imagination. Par exemple, certaines applications web présentent des séries élaborées d'écrans à travers lesquelles les utilisateurs doivent naviguer avant d'arriver au contenu principal de l'application. Bien que ce type de pages puissent être acceptables pour une application de bureau, quand c'est utilisé pour une application mobile, cela va surtout ralentir l'accès des utilisateurs à leur contenu. Dans un design d'application mobile, il faut être sûr d'utiliser l'interactivité pour rapprocher les utilisateurs de leurs buts et éviter l'interactivité n'ayant aucun but fonctionnel [MIND].

De plus, l'interactivité non fonctionnelle rajoute une couche non nécessaire d'indirections entre les utilisateurs et le contenu auquel ils veulent accéder. Ainsi, il faudrait éviter l'interactivité qui pourrait être remplacée par des éléments simples et directs.

Lorsque l'on crée une application mobile, le contenu est par définition interactif. Toutefois, il faut examiner en détail le design de l'application et éliminer les éléments interactifs qui ne fournissent pas une fonctionnalité essentielle et chercher à les remplacer par des contrôles simples et clairement libellés [APPL].

II.5.2. Focalisation

Une application de bureau ou une application mobile qui établit et maintient la focalisation sur sa fonctionnalité première est plus agréable et plaisante à utiliser. Particulièrement critique pour les applications mobiles, la focalisation renforce la perception que l'utilisateur a de l'application et de son contenu. Lors du design d'applications mobiles, il faut rester concentré sur cette fonctionnalité première et faire en sorte que chaque fonctionnalité de l'application et chaque élément de l'interface graphique la soutiennent.

Dans une application mobile, une bonne façon de réussir à se focaliser sur la fonctionnalité première est de déterminer ce qui est le plus important dans chaque contexte. Lorsqu'il faut décider de ce qui est affiché sur chaque écran de l'application, il faut se demander : est-ce que cette information ou fonctionnalité est critique et utile aux utilisateurs à cet endroit ? Si la réponse est non, il faut alors décider si cette information ou fonctionnalité est critique dans un contexte différent ou si, finalement, ce n'est pas si important que cela. Par exemple, une application mobile qui aide les utilisateurs à stocker des informations relatives à la consommation de leurs voitures perd en focalisation si elle stocke et affiche également l'historique des maintenances et réparations de leurs voitures [APPL].

II.5.3.Cohérence

Dans le développement d'applications mobiles, il a deux types de cohérences à respecter. La première est celle entre les écrans d'une même application. Elle permet à l'utilisateur de retrouver ses marques d'un écran à l'autre. La seconde est la cohérence d'un même écran, au niveau du choix des couleurs aussi bien que de la taille des différents contrôles. [APPL]

II.5.4.Mise à disposition aux utilisateurs

Il y a deux façons de proposer des applications aux utilisateurs de smartphones : soit en développant une application native, par exemple avec xCode pour l'iPhone et en la mettant à disposition des utilisateurs via l'App Store, soit en créant une application web, c'est-à-dire un site web adapté aux smartphones, en respectant les mêmes principes de design que pour une application mobile auquel cas l'utilisateur y accèdera avec le navigateur web.

Néanmoins, il faut savoir qu'il est possible d'associer les deux méthodes et de proposer une application native consistant en un navigateur web basique affichant uniquement le web application.

II.6.Genero

II.6.1.Un fonctionnement en mode connecté

En raison de son architecture, présentée en préambule, le fonctionnement de Genero nécessite un mode connecté. En effet, le client et le serveur communiquent au fur et à mesure des interactions avec les utilisateurs.

II.6.2.L'arbre AUI

Le code est indépendant de la présentation. Sur la figure 22, nous pouvons voir un exemple de code ouvrant une fenêtre et affichant des données. Aucun élément de présentation n'est inclus, car les éléments de l'interface graphique sont séparés de la logique applicative.

```
FUNCTION wizard()  
  ... defines  
  OPEN WINDOW formwizard WITH FORM  
  "formwizard"  
  ... init arrays  
  DIALOG ATTRIBUTES(UNBUFFERED)  
  INPUT BY NAME currtable  
  DISPLAY ARRAY afields TO a.*  
  DISPLAY ARRAY cfields TO c.*
```

Figure 22 : Exemple de code affichant des données

➤ AUI is a logical UI map

```

- <Form name="giftcert" text="Gift Certificates" width="78" height="4">
- <VBox>
- <Group text="Certificate Details">
- <Grid width="60" height="6">
  <Label text="Recipient:" width="10" posX="8" posY="0" />
  <Edit posY="0" posX="29" width="30" fieldId="0" colName="issued_to"
  sqlTabName="gift_cert" sqlType="CHAR(30)" shift="up" />
  <Label text="Sender:" width="7" posX="8" posY="1" />
  <Edit posY="1" posX="29" width="30" fieldId="1" colName="sender"
  sqlTabName="gift_cert" sqlType="CHAR(30)" shift="up" />
  <Label text="Certificate Amount:" width="19" posX="8" posY="2" />
  <Edit posY="2" posX="29" width="10" fieldId="4" colName="cert_amt"
  sqlTabName="gift_cert" sqlType="DECIMAL(8,2)"
  include="25.00|50.00|75.00|100.00" comment="Valid amounts include
  25, 50, 75, or 100 dollars" />
  <Label text="Merchant:" width="9" posX="8" posY="3" />
  <ComboBox posY="3" posX="29" width="5" fieldId="5"
  colName="merchant_id" sqlTabName="gift_cert" sqlType="CHAR(3)"
  required="1" notNull="1" />
  <Edit posY="3" posX="36" width="23" fieldId="6" colName="merchant_name"
  sqlTabName="merchant" sqlType="CHAR(20)" noEntry="1" />
  <Label text="Message:" width="8" posX="8" posY="4" />
  <TextEdit posY="4" posX="29" width="30" height="2" fieldId="2"
  colName="message_text" sqlTabName="gift_cert" sqlType="CHAR
  (200)" />
</Grid>
</Group>
+ <Grid width="78" height="4">
</VBox>
+ <RecordView tabName="merchant">
+ <RecordView tabName="gift_cert">
+ <DefaultActions>
</Form>

```

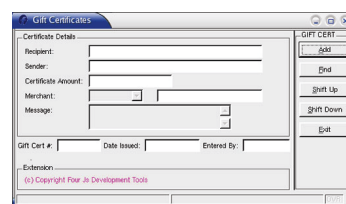
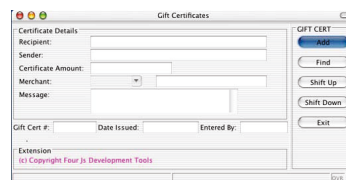
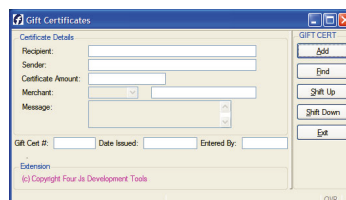


Figure 24 : L'arbre AUI permet d'afficher des applications natives sur chaque OS

II.6.3. Vue d'ensemble du Genero Application Server

Le GAS est un moteur qui délivre des applications Genero. Il crée les liens entre les différents clients graphiques (GDC, GDCAX, GWC) et les DVM (Dynamic Virtual Machine) qui exécutent les applications. Le GAS gère également une réserve de DVM pour les applications utilisant des services web.

Le GAS est interfacé à un serveur web qui gère les requêtes de l'Internet. La communication entre le serveur web et le GAS est gérée par un connecteur qui gère les requêtes entre eux.

II.6.3.1. Architecture

La figure 25 présente l'architecture du GAS, qui est basée sur :

- un moteur évolutif, hautement disponible, et construit sur un noyau de composant réutilisable,
- des connecteurs : permettant de se connecter à des serveurs web (IIS, J2EE),
- des interfaces de liaison entre le moteur et les connecteurs : qui informe le moteur des demandes et réponses HTTP.

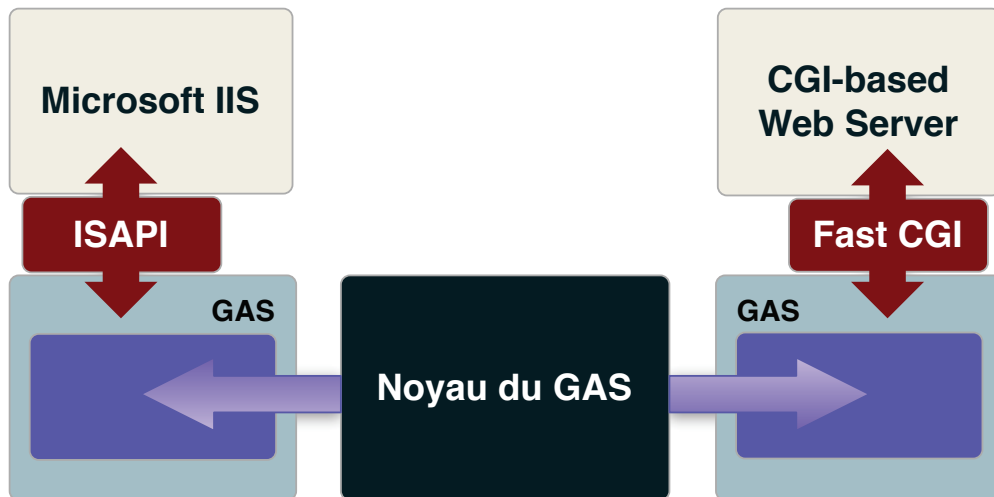


Figure 25 : L'architecture du GAS

Le noyau du GAS correspond au code commun pour tous les types de serveurs web. A cela vient se rajouter du code spécifique à chaque serveur web, qui va permettre au GAS de se connecter à ces derniers.

II.6.3.1.1. Fonctionnement et rôle

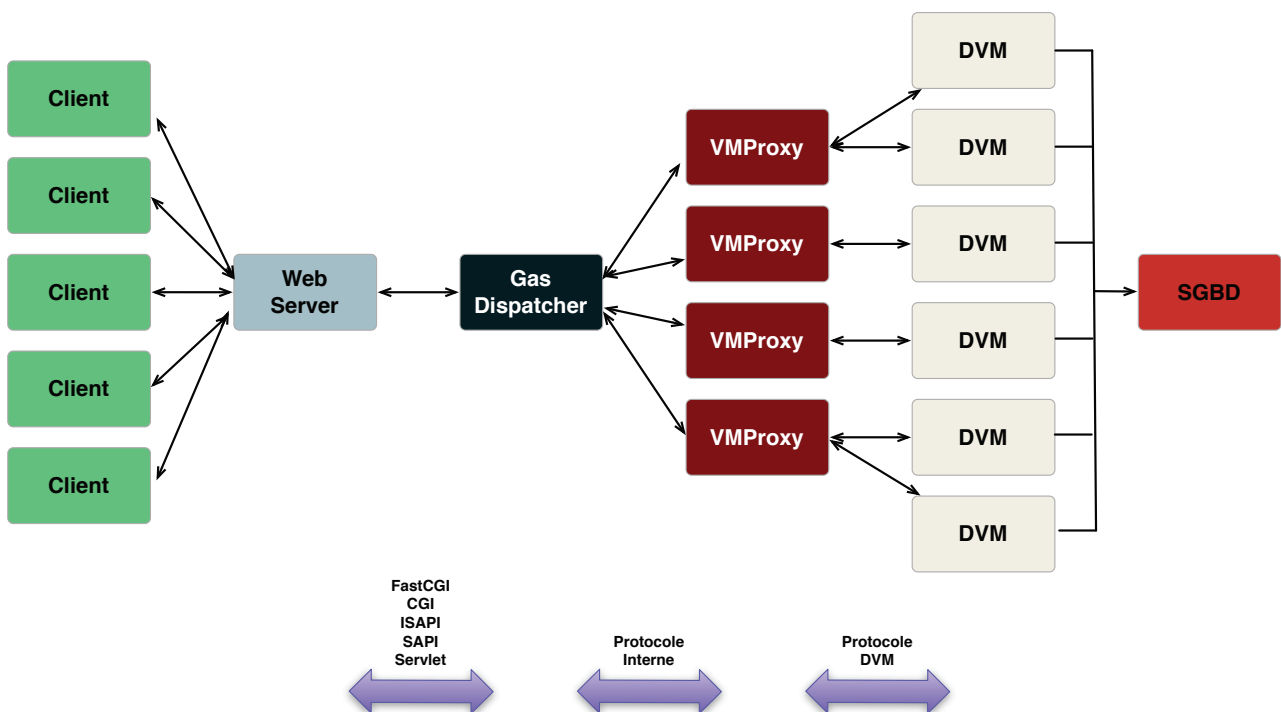


Figure 26 : Le fonctionnement du GAS

L'architecture présentée sur la figure 26 a deux rôles :

- servir de répartiteur (GAS Dispatcher),
- servir de proxy pour la DVM (VMProxy).

Le répartiteur a les rôles suivants :

- gérer les requêtes statiques sur les fichiers,
- transférer les requêtes HTTP au VMProxy correspondant,
- gérer une table de sessions ouvertes persistante et partagée qui distribue les requêtes aux VMProxies adéquats,
- démarrer de nouveau VMProxies lorsqu'une nouvelle session utilisateur est instanciée.

Le rôle du VMProxy quant à lui est de démarrer une DVM et de gérer les DVM fils. Il réalise des actions différentes en fonction du service demandé :

- GWCProxy : Le VMProxy héberge le SBRE (Snippet-Based Rendering Engine) qui va transformer l'arbre AUI en pages web et traduit les requêtes HTTP du CSF²⁵ (Client Side Framework) en protocole DVM,
- GDCProxy : Le VMProxy traduit le protocole HTTP depuis et en protocole DVM,
- GWS : Le VMProxy gère le pool de DVM GWS et transmet les requêtes HTTP de services web de et vers les DVM.

Cette architecture offre deux avantages principaux : la fiabilité et les performances.

La fiabilité, car grâce à la fonctionnalité de restauration du répartiteur, il est possible de redémarrer à chaud sans perdre les sessions utilisateurs actives. De plus, avec l'isolation de chaque utilisateur par rapport aux autres, le fait qu'une session utilisateur échoue ne perturbe pas les autres.

Et les performances du fait que cette architecture est basée sur les fils (threads) et la gestion des entrées-sorties offrant de très bonnes performances, une faible fragmentation de la mémoire, une faible consommation des ressources et un très bon temps de réponse.

II.6.3.2.Le GAS Standalone

Avec le support du protocole HTTP, le GAS offre une connexion directe pour accéder aux applications sans serveur web. Cette fonctionnalité est proposée uniquement pour le cycle de

²⁵ Le CSF est un framework javascript interne servant entre autre pour les échanges entre le GWC et la DVM.

développement, permettant de retirer le serveur web de l'environnement de développement. Pour le déploiement et la mise en production, un serveur web est obligatoire.

Le GAS simplifie la phase de déploiement en s'occupant de la connexion avec les applications. Du côté client, aucune installation de logiciel ni aucune configuration ne sont nécessaires. En effet, un navigateur web est tout ce qui est nécessaire du côté client pour accéder aux programmes.

II.6.3.3. Front Ends et extensions

Le GAS peut proposer les applications via les Front Ends et extensions suivantes :

- le GDC permet d'exécuter les applications à travers le GAS et de les exécuter localement via le GDC. Le logiciel GDC doit être installé du côté client et est compatible avec tous les systèmes d'exploitation du marché (Microsoft Windows, Mac OS X, Linux et les principaux UNIX),
- le GWC permet de délivrer des applications Genero dans un navigateur web, sans besoin d'installation de logiciel sur la machine cliente,
- les GWS permettent d'implémenter des services web. Les services web sont un moyen de communication standard entre des applications sur l'Internet et l'intranet. Un Service Web peut être un serveur qui expose et propose des services, ou un client qui consomme ces services.

II.6.4. Vue d'ensemble du Genero Desktop Client

Le GDC est le premier outil de la suite Genero permettant l'affichage des écrans. Il est basé sur QT²⁶, un langage multiplateforme racheté par Nokia et toujours développé.

Nous pouvons voir sur la figure 27 que tous les contrôles requis par des applications de gestion sont supportés.

²⁶ <http://qt.nokia.com/>

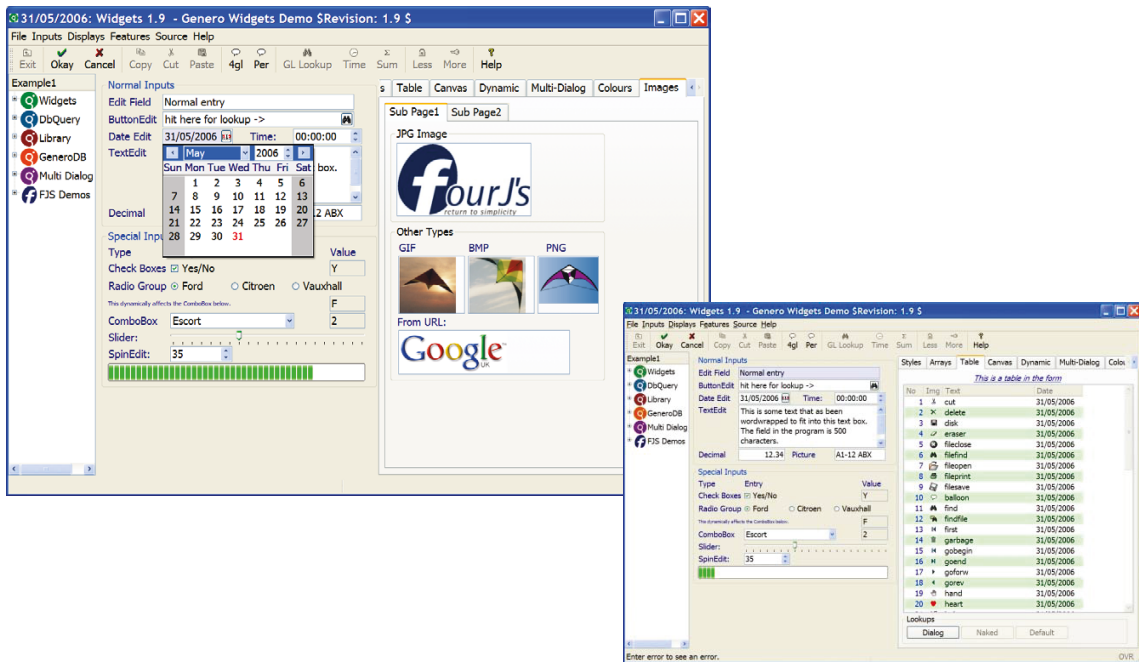


Figure 27 : Exemple d'applications affichées avec le GDC

L'aspect natif de chaque système d'exploitation est bien respecté comme nous pouvons le constater sur la figure 28.

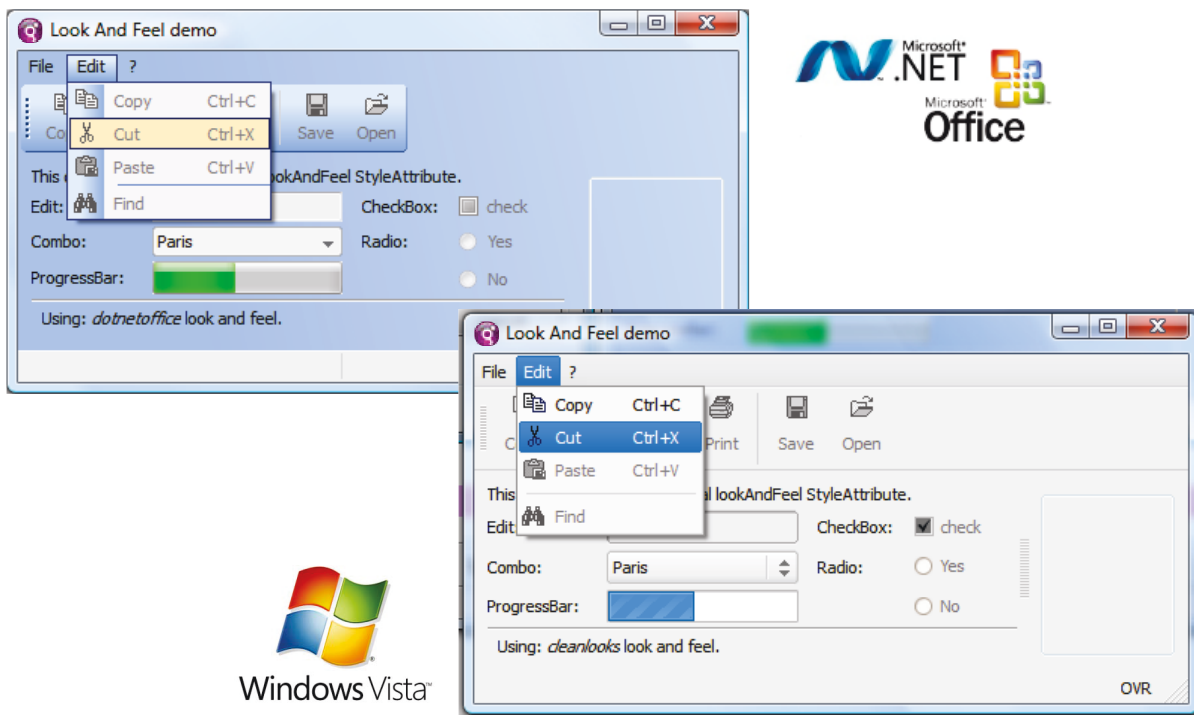


Figure 28 : Applications Genero rendues de façon native

Le GDC fonctionne sur toutes les plates-formes (Microsoft Windows, Mac OS X, Linux et les principaux UNIX du marché) et consiste en une application à installer sur les postes clients.

Cette application reçoit les informations de la DVM via un port ouvert sur le réseau, et utilise ces informations pour afficher les fenêtres de l'application et pour échanger les données avec la partie logique métier de l'application.

II.6.5. Vue d'ensemble du Genero Web Client

Avec le GWC, il est possible de délivrer de vraies applications web à partir d'applications développées avec Genero 4GL. Avoir le code source de l'application écrit en Genero 4GL signifie que le GWC est suffisamment flexible pour construire une application web simple à une application web professionnelle et très complète avec seulement quelques limitations.

Le GWC amène les applications Genero 4GL à l'Internet, et permet également aux applications d'être intégrées directement à des sites web existants.

Les applications sont délivrées en utilisant des technologies supportées par les navigateurs web et ainsi le GWC permettra d'utiliser les applications sur n'importe quelle machine ou systèmes dotés d'un navigateur web.

II.6.5.1. Pourquoi proposer une application en tant qu'application web ?

Le déploiement des applications web est bien plus simple et moins coûteux que le déploiement d'application de bureau. Le client nécessite uniquement un navigateur web, et aucune autre application n'a besoin d'être installée sur le client.

Le GWC utilise des thèmes, composé de gabarits et de jeux de snippets²⁷, pour créer des pages web dynamiques. Différents jeux de snippets sont fournis pour permettre la création de pages web dynamiques dans le langage approprié pour le navigateur web utilisé par le client.

Le thème AJAX (Asynchronous JavaScript and XML) fournit des gabarits et un jeu de snippets basé sur les langages web communs : xHTML (eXtensible HyperText Markup Language), XSLT (XSL Transformation) et JavaScript alors que le thème Silverlight utilise XAML, C# et les composants spécifiques de Silverlight.

II.6.5.2. Les thèmes GWC pour les navigateurs web

Le GAS exploite un moteur de rendu basé sur des snippets. Quand le GAS affiche une application en utilisant le GWC, il détecte le type de navigateur web. En fonction du navigateur

²⁷ Le terme snippet désigne un fragment de code

web, il identifie et définit le thème à appliquer. Le thème spécifie le jeu de snippets à utiliser et le GAS affiche les applications en utilisant le langage approprié au navigateur web destinataire.

Il y a actuellement deux thèmes.

Le thème AJAX (GWC pour AJAX) est basé sur un cadriciel en JavaScript. Il affichera les applications avec une expérience utilisateur très proche de ce que propose le GDC, mais dans un environnement Web 2.0. Le thème AJAX est compatible avec les principaux navigateurs web du marché.

Le thème GWC pour Silverlight est basé sur la technologie Microsoft Silverlight et sur XAML (Extensible Application Markup Language) plutôt que sur le HTML. C'est un front-end Genero offrant la possibilité d'être personnalisé, comme GWC pour AJAX. Il offre la possibilité de créer des RIA (Rich Internet Application).

III. Le projet

III.1.Introduction du projet

C'est fort des constats faits dans la partie précédente que j'ai proposé à ma société un projet de développement d'une application mobile permettant d'exploiter les applications Genero sur les terminaux mobiles sous iOS (iPhone et iPod touch).

J'ai présenté trois approches possibles pour arriver à cette fin. Un client graphique Genero sous forme d'application native, une application web et enfin une application hybride qui utilise au sein d'une application native le moteur de Safari mobile.

J'ai soumis à mon entreprise un comparatif de ces trois solutions afin qu'elle puisse choisir laquelle s'intégrait le mieux dans la stratégie globale de la société.

III.1.1.Comparatif

III.1.1.1.Langages

L'application native est à développer en Objective-C, le langage utilisé pour le développement d'application pour iOS. L'application web utilise le HTML, le CSS et le JavaScript. Enfin, l'application hybride utilise tous ces langages à la fois : l'Objective-C pour l'application à proprement parler et les langages web pour l'affichage des pages.

III.1.1.2.Développement/Coûts

Je ne maîtrise parfaitement ni Objective-C (que je commence néanmoins à connaître par autoformation), ni HTML/JavaScript (dont je ne connais que les bases).

Par contre, en interne, la connaissance des technologies web est assez largement répandue, alors que la connaissance des outils de développement Apple ne l'est pas. De plus, pour la solution application web, il y a déjà une base de travail avec les différents thèmes du GAS.

Enfin, le SDK de l'iPhone ne tourne que sur Mac OS X, ce qui nécessite l'achat d'une machine Apple. Le coût pour une entreprise pour mettre une (ou plusieurs) application sur l'App Store est de 99 \$ par an.

III.1.1.3.Mise en production/Réactivité

L'application web a un avantage ici, car même si Apple fournit un outil de déploiement d'application gratuit (iPhone Configuration Utility) permettant aux entreprises de distribuer une application dans l'entreprise, la mise en production d'une application web se fait plus rapidement, car elle ne passe pas par le processus de validation d'Apple pour mettre une application sur l'App Store.

De surcroît, à chaque mise à jour d'une application native, il faut attendre la validation d'Apple avant qu'elle ne soit disponible pour les utilisateurs.

III.1.1.4.Accessibilité sur l'iPhone

Équivalentes, les trois approches permettent d'afficher une icône sur le bureau.

III.1.1.5.Accessibilité

Puisque la DVM ne fonctionne qu'en mode connecté, application web, application native et application hybride ont des contraintes similaires à ce niveau. Il faudra obligatoirement avoir son terminal connecté au réseau pour fonctionner.

III.1.1.6.Vitesse

Une application native sera plus rapide qu'une application web ou hybride. En effet, l'Objective-C est le langage natif de l'iPhone et il est exécuté plus rapidement.

III.1.1.7.Effet marketing

L'application native est largement en tête. Une application web n'aura pas d'effet « Waouh », alors qu'une application native permettra de communiquer plus largement sur ce point. De surcroît, il y a souvent dans les magazines professionnels (ou non) des articles avec les applications Salesforce (2 applications), Oracle (5 applications natives déjà) ou encore SAP qui sont présentés.

Il peut y avoir beaucoup à gagner en publicité indirecte avec une application native (bien plus qu'avec une application web).

Une application hybride a les mêmes avantages qu'une application native à ce niveau.

III.1.1.8.Utilisation des fonctionnalités matérielles de l'iPhone

Tous les composants du téléphone peuvent être utilisés à travers une application native, ce n'est pas le cas d'une application web. Dans le cadre d'évolutions futures (utilisation du GPS, de l'appareil photo numérique, de l'accéléromètre, etc.), ce sera limitatif d'avoir choisi l'application web. On peut imaginer de nombreuses fonctionnalités à partir de ces composants matériels. Un exemple serait l'utilisation de l'appareil photo numérique comme scanner de code-barre.

Concernant l'application hybride, celle-ci permet d'exploiter les composants du téléphone presque de la même façon qu'une application native en associant des fonctions JavaScript à des méthodes Objective-C.

III.1.1.9.Intégration avec les applications de l'iPhone

Une application native permet d'accéder, entres autres, aux contacts, agendas et photos de l'iPhone, alors qu'une application web ne le permet pas. De plus, il est possible de stocker des données sur l'iPhone par le biais d'une application native bien plus facilement qu'avec une application web.

L'application hybride permet d'accéder aux mêmes fonctionnalités qu'une application native.

III.1.1.10.Facilité d'utilisation

Application web, native et hybride ont la même facilité d'utilisation et peuvent avoir une ergonomie similaire.

III.1.1.11.Portée et réutilisabilité

Avantage à l'application web, car elle permet d'atteindre potentiellement tous les smartphones ayant un navigateur web basé sur WebKit, et éventuellement les autres navigateurs mobiles.

Dans le cas d'une application native, le travail de design pourra être capitalisé dans le cas ou l'on voudrait développer une application native pour un autre OS mobile (Android, WebOS, BlackBerry OS, Windows Phone 7...), mais pas le code.

Pour l'application hybride, toute la partie web pourra être exploitée sur les autres OS mobiles.

III.1.1.12.Préférence personnelle

Ayant déjà commencé à apprendre Objective-C et xCode, je préférerais la piste de l'application native. L'application hybride est une approche qui me plait également, car elle combine l'ensemble des technologies et me permettrait de mieux comprendre le fonctionnement du moteur de rendu du GAS.

III.1.2.Résumé

Application native	Application web
+ Outils de 'buzz' et couverture par la presse + Accès à toutes les fonctions de l'iPhone + Plus d'effets graphiques + Stockage de données sur le smartphone + Vitesse	+ Développement plus rapide et moins cher + Compatibilité avec tous les smartphones + Plus de compétences en interne + Base existante (mode PDA du GAS)
- Développement plus coûteux - Non-portabilité du code	- Effet marketing moindre - Pas d'accès aux composants matériels ni aux ressources de l'iPhone

Tableau III : Comparatif entre applications natives et applications web

Quant à l'application hybride, elle cumule les avantages des applications natives et des applications web et surtout elle minimise les désavantages de chacune des autres solutions :

- distribution via l'AppleStore,
- temps de développement HTML vs Objective-C,
- code de l'application à 80 % en HTML et donc facilement portable sur Android et autres smartphones ou sous forme d'application web,
- code source HTML protégé au sein de l'application,
- maintenance à distance de l'application sans revalidation par Apple,
- pas d'obligation de maintenir plusieurs versions fonctionnelles,
- interactions possibles entre les pages web encapsulées et les différents composants et fonctionnalités de l'iPhone.

III.1.3.Choix

Au vu des éléments fournis à mon supérieur et à ma société, le choix a été fait de partir sur l'application hybride.

Le choix technologique associé à chaque solution a été déterminant dans ce choix. En effet, l'utilisation des langages web pour 80 % du projet permet de mettre à profit les ressources de la société. De plus, l'application hybride est la solution la plus versatile des trois.

III.2.Les objectifs du projet

Le choix a donc été fait de partir sur le développement d'une application mobile hybride permettant d'exploiter des applications Genero sur les terminaux mobiles sous iOS.

Ce projet se découpe en deux étapes majeures :

- la création d'un thème pour le GAS reproduisant l'aspect et le comportement des applications natives iPhone,
- la création d'une application iPhone pour gérer et exécuter les applications Genero à partir de l'iPhone.

La priorité a été donnée à la première étape, la seconde n'a pas encore été planifiée. C'est pourquoi la suite du mémoire ne portera que sur la création d'un thème pour le GAS.

III.3.Les contraintes du projet

III.3.1.Contraintes de temps

La sortie du thème iPhone pour le GAS a été annoncée dans la feuille de route de la société pour juin 2010. Donc la première partie du projet devait impérativement être achevée à ce moment-là.

Il n'y a pas de contrainte de temps pour la seconde partie concernant l'application iPhone à proprement parler.

III.3.2.Contraintes de moyen

Outre moi-même, les ressources humaines affectées à ce projet en terme de développeurs seront restreintes.

III.3.3.Contraintes financières

Il n'y a pas de contrainte financière particulière.

IV. Déroulement du projet

IV.1. Organisation générale du projet

Pour ce projet, la société m'a accordé une grande latitude et une grande autonomie. En effet, l'organisation dans notre société fait que les ingénieurs sont assez libres dans la gestion du temps et leurs méthodes de travail. Ceci est essentiellement dû à notre activité exclusivement informatique, il y a donc seulement un minimum de contraintes administratives, et les contraintes hiérarchiques sont souples.

La seule contrainte est de faire un point hebdomadaire avec mon supérieur hiérarchique, Olivier Imbert, directeur de projet GCS.

IV.2. Méthode de conduite de projet

Il n'y a pas de méthode de conduite de projet générique dans notre entreprise, j'ai donc été libre de choisir la méthode que je souhaitais.

La méthode de conduite de projet choisie se rapproche des méthodes agiles. C'est donc avant tout une méthode itérative sur la base d'un affinement du besoin mis en œuvre dans des fonctionnalités en cours de réalisation et même déjà réalisées.

C'est également une méthode incrémentale, la production des fonctionnalités s'effectue en plusieurs étapes.

IV.3. Responsabilité du candidat

Sous la direction d'Olivier Imbert, j'ai la responsabilité du management du projet (analyse fonctionnelle et technique, gestion des ressources). Je participe également au développement et surtout je suis responsable de la partie interface graphique et expérience utilisateur.

Enfin, je superviserai les phases de validation et de test.

IV.4. Équipes prévues, encadrement assuré par le candidat

L'équipe est transversale, il n'y a pas de hiérarchie directe.

Les différents acteurs prévus sont :

- un développeur web « junior »,
- deux développeurs web « senior »,

- un designer web,
- un ergonomiste,
- une équipe qualité,
- une personne pour la documentation et le support.

Les ressources ne sont pas disponibles tout le temps et ne se consacrent pas à 100 % à ce projet.

IV.5.Options déjà connues entre lesquelles se feront certainement les choix ultérieurs

Pour la première partie du projet, il y a un support souhaité d'autres navigateurs mobiles, principalement ceux basés sur WebKit (Android et BlackBerry OS 6.0).

Il faudrait également prendre en compte de l'arrivée prochaine des tablettes tactiles.

Enfin, il ne faut pas utiliser de sources commerciales ou externe (comme jQuery), mais plutôt adapter des éléments existants et faire de la rétro-ingénierie. Il y a, en effet, un existant important dans l'entreprise, qui permet de se passer de sources commerciales ou externes.

IV.6.Tests prévus

Des sessions de tests manuels sont prévues. Ceux-ci seront réalisés par l'équipe QA du GAS située en Inde.

Pour les tests automatiques, il faudra étudier leur faisabilité avant de prendre une décision.

IV.7.Modalité de validation du projet

La première partie du projet sera validée lorsque le thème iPhone sera intégré aux sources du GAS et livré à nos clients.

La deuxième partie du projet sera validée lorsque l'application iPhone sera disponible aux utilisateurs sur l'App Store.

V. État de l'Art

J'ai découpé les recherches concernant l'état des connaissances existantes concernant le projet en trois parties.

J'étudierai le fonctionnement du moteur de rendu du GAS, puis les contraintes ergonomiques liées à la taille de l'écran et aux contrôles tactiles et je finirai avec un aperçu des cadres web existants.

V.1.Fonctionnement du moteur de rendu du GAS

Puisque le thème iPhone fonctionnera d'une façon similaire aux thèmes déjà existants du GAS, il est utile de s'intéresser au fonctionnement de l'existant.

Le GWC permet aux développeurs de construire des applications en utilisant Genero BDL et de les délivrer en tant qu'applications web. Pour délivrer une application Genero comme une application web, le GWC doit générer le rendu comme une application XHTML qui sera exploitable par un navigateur web.

Le GWC utilise des modèles pour le rendu. Il y a quatre composants principaux pour le rendu GWC : le HTML généré (le cœur de l'application), le CSS (l'aspect de l'application), du JavaScript (les fonctionnements et comportements des éléments de l'interface utilisateur) et un langage de macro propre au GWC.

V.1.1.Le SBRE

Le SBRE permet aux développeurs et aux architectes d'applications de personnaliser entièrement l'aspect et le rendu d'une application Genero dans un navigateur web à l'aide de jeux de snippets prédéfinis. Avec ces snippets, il est possible d'adapter les écrans à tous les navigateurs web, du plus simple des assistants numériques personnels aux plus puissants des navigateurs web contemporains. De plus, un langage balisé spécifique à chaque sortie peut être ajouté pour connecter des appareils externes (comme un lecteur de code-barre) à des applications Genero.

Par défaut, le GWC utilise le SBRE. Le moteur de rendu utilisé est sélectionné en fonction de la signature du navigateur web ou peut être spécifiquement défini dans l'URL (Uniform Resource Locator) de l'application.

Le SBRE repose sur le modèle objet du GWC. Lors de la composition des pages web, le modèle objet du GWC permet d'avoir accès à toutes les données disponibles d'une application. Les sources de données incluent l'arbre AUI de l'application, les données du document (contient les informations sur le rendu du document courant comme l'URL où envoyer le formulaire, les erreurs de rendu, etc.) et les données du serveur (donnent accès à des informations techniques sur le serveur, comme le numéro de version du serveur d'application).

Le modèle objet du GWC expose ses données à travers des propriétés. Ces propriétés sont accessibles à l'aide d'une notation de type chemin à l'intérieur d'expressions du GWC.

Le GWC lit l'arbre AUI de l'application, qui correspond à la description abstraite de l'interface utilisateur courante de l'application. Certains des objets du modèle objet du GWC pointent vers des entités bien définies dans l'arbre AUI (comme un objet fenêtre, un objet formulaire, un champ d'édition, etc.). D'autres objets du modèle objet du GWC correspondent à une entité utilisée par d'autres objets de l'arbre AUI (comme le composant `GridLayout`, recrée à chaque fois qu'un groupe `Grid`, `ScrollGrid` ou `Group` apparaît dans l'arbre AUI). Les objets dans ce modèle objet du GWC sont couverts par des composants graphiques du GWC.

Le rendu des composants graphiques du GWC est généré à l'aide du jeu de snippets du modèle. Les snippets sont utilisés dynamiquement lors de l'exécution de l'application. Un snippet du modèle est décodé en fonction du contexte du composant graphique associé. Le modèle principal (`main`) contrôle le rendu général de la page de l'application.

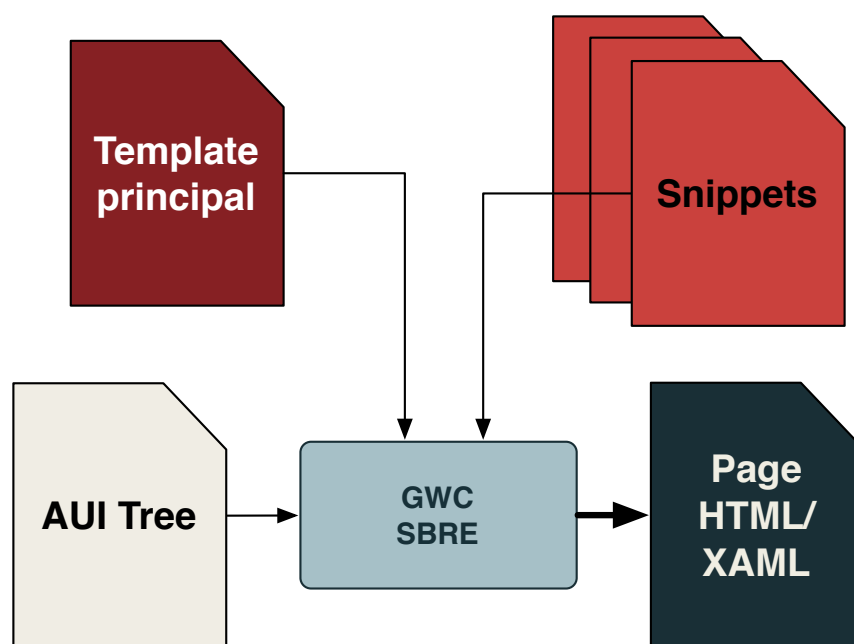


Figure 29 : Le fonctionnement du SBRE

Le modèle et les fichiers correspondants aux snippets fournis avec le SBRE sont écrits soit en xHTML, soit en XSL.

xHTML est plus ou moins une adaptation du HTML pour faire respecter à ce langage le formatage clair du XML. Avec un snippet xHTML, il est possible d'utiliser le langage de modèle du GWC à l'intérieur de chaque fichier décrivant un snippet.

V.1.2. Comment une application est rendue par le GWC

Le schéma suivant montre une vue d'ensemble des étapes nécessaires au GWC pour délivrer une application, en débutant avec l'utilisateur qui saisit une URL et en finissant par l'application affichée avec l'user agent²⁸ correct.

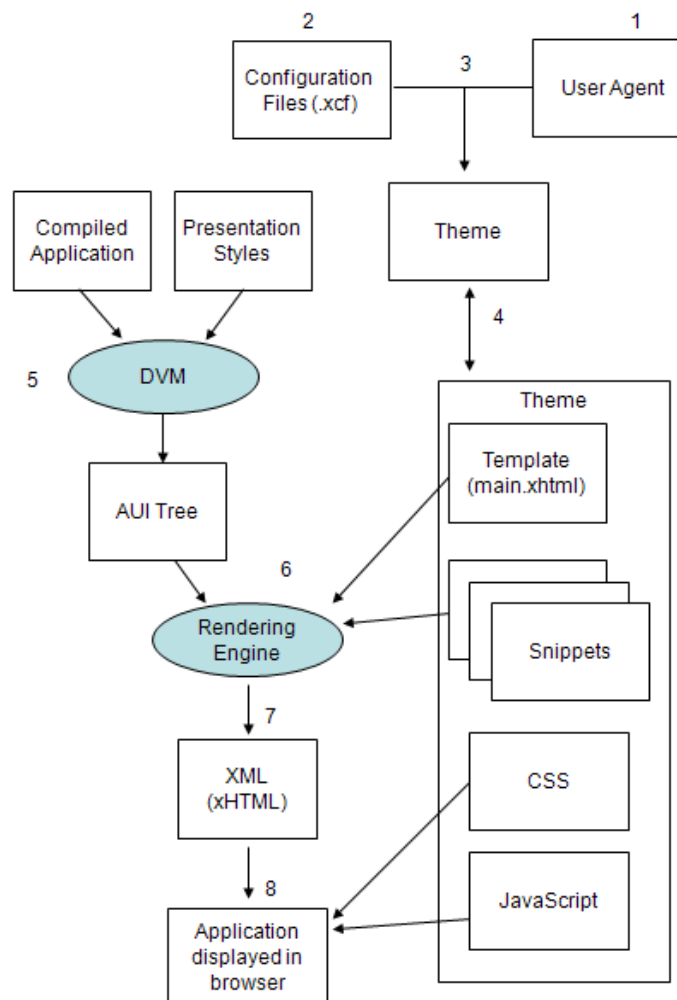


Figure 30 : Comment une application est rendue par le GWC

²⁸ Un user agent est une application cliente utilisée avec un protocole réseau particulier ; l'expression est plus généralement employée comme référence pour celles qui accèdent au World Wide Web. Les User Agents du Web vont de la gamme des navigateurs jusqu'aux robots d'indexation, en passant par les lecteurs d'écran ou les navigateurs braille pour les personnes ayant une incapacité.

L'utilisateur final lance une application en saisissant l'URI (Uniform Resource Identifier) approprié dans l'user agent (1).

La requête pour l'application est dirigée vers un GAS. Le fichier de configuration du GAS contient les informations sur le thème à utiliser pour l'application demandée (2).

À moins d'être spécifiquement déclaré dans l'URI ou dans la configuration de l'application, le thème approprié est choisi en fonction du type de l'user agent (navigateur web, application de bureau, assistant numérique personnel, etc.). Le thème à utiliser en fonction de la signature du navigateur est stocké dans un fichier (adua.xrd) (3).

Dans le fichier de configuration de GAS, un thème est constitué d'un modèle et d'un jeu de snippets. Le fichier de modèle fournit les références aux fichiers JavaScript et CSS nécessaires (4).

Le GAS, pendant ce temps, a démarré une DVM pour servir l'application. La DVM crée l'arbre AUI qui est envoyé au GAS (5).

Le SBRE utilise l'arbre AUI fourni par la DVM ainsi que le modèle et les snippets pour créer un document XML/xHTML qui est fourni à l'user agent (6, 7).

Les méthodes JavaScript et styles CSS sont appliqués au fichier xHTML par l'user agent avant que l'application ne soit affichée à l'utilisateur (8).

V.1.3.Modèle

Un modèle est un fichier xHTML qui affiche l'application dans un navigateur web en utilisant un front-end Genero. Un modèle définit comment et où l'application est affichée à l'intérieur d'une page HTML. Le GWC a un modèle par défaut affichant la fenêtre courante de l'application.

V.1.4.Cascading Style Sheet

Le CSS est un langage informatique qui sert à décrire la présentation des documents HTML, XML et xHTML. Les standards définissant CSS sont publiés par le W3C (World Wide Web Consortium). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000²⁹.

²⁹ http://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade

Le GWC utilise les CSS pour placer et façonner les widgets.

Par exemple, le style CSS de la figure 31, tiré du fichier gwcomponents.css (du thème AJAX), décrit comment sont affichés les menus.

```
.gMenu {  
  float: left;  
  clear: both;  
  width: 95 %;  
  padding: 0px 16px 2px 0px ;  
  margin: 0px 2px 10px 2px ;  
  -moz-border-radius-bottomright: 16px ;  
  -moz-border-radius-topright: 16px ;  
  background-color: #F4F4F4;  
}
```

Figure 31 : Exemple de style CSS pour le menu avec le thème AJAX

N'importe quel champ de saisie à l'intérieur d'un élément de la classe gMenu n'a pas de contour.

Grâce au concept de snippet, les développeurs ne sont pas limités au CSS par défaut. Ils peuvent ajouter n'importe quelle classe CSS à leur code HTML généré.

V.1.5.JavaScript

Le GWC peut interagir avec le navigateur web grâce au cadriciel JavaScript développé en interne et appelé CSF. Le GWC fournit au navigateur une page XHTML décrivant l'état courant d'une application Genero. Le JavaScript peut être utilisé du côté du navigateur pour n'importe quel besoin, comme par exemple afficher la fenêtre du calendrier, ou un élément de saisie, une image, une vérification de format, etc.

Le GWC peut également interagir avec le navigateur en utilisant le cadriciel JavaScript. Dans le cas, le GWC fournit au cadriciel JavaScript les modifications du fichier XHTML de façon incrémentale (en fonction du modèle et du jeu de snippets) à partir des informations fournies par la DVM. Le cadriciel JavaScript permet de reproduire le comportement d'une application Genero exécutée de façon native.

V.1.6.Langage de macro

Le langage de macro développé par Four J's pour le GWC permet de créer des pages web plus dynamiques et d'étendre les capacités communes du HTML. Le langage de macro peut s'utiliser avec les outils de design web, n'influe pas sur la disposition et l'affichage de la page et est très versatile et puissant.

Le langage de macro est utilisé dans les modèles et les snippets. Il est interprété par le moteur de rendu du GWC qui génère du code HTML à partir de macros. Il est possible d'exécuter des instructions qui vont de simples tests conditionnels à des boucles à l'intérieur de lignes de tables.

Le langage de macro donne accès à trois types d'objet : des instructions, des expressions et des chemins :

- une instruction est un attribut prédéfini ajouté à une balise HTML. Il décrit comment la balise HTML est interprétée,
- une expression correspond à la valeur d'une instruction. Cela peut être une chaîne, une opération ou une fonction de conversion,
- un chemin est utilisé pour accéder à un élément de l'application courante. L'élément en question peut aller de la fenêtre tout entière de l'application à la valeur d'un champ d'une table.

Par exemple dans le snippet représentant une barre d'outil (`toolbar`), il est possible d'utiliser le chemin suivant `/application/ui/toolbar/items` qui contient la liste de tous les éléments d'une barre d'outil.

V.2.Le thème AJAX

Le thème AJAX est le thème par défaut utilisé lorsque l'on exécute une application à travers le GWC. Il génère l'application en utilisant des technologies supportées par le navigateur web, des standards comme XHTML, XSLT, CSS et JavaScript.

Cela permet de délivrer des applications sur n'importe quels appareils ou ordinateurs équipés d'un navigateur web et supportant les technologies AJAX. Le thème AJAX est conçu pour des navigateurs web modernes comme Firefox, Opera, Chrome, Internet Explorer et Safari.

Bien que le thème AJAX permette aux applications Genero d'avoir une expérience utilisateur proche de celle d'une application de bureau, une application destinée à être utilisée via l'Internet (ou l'intranet) ne sera pas développée de la même façon qu'une application destinée à une utilisation en local via le GDC. Il faut prendre en compte les contraintes des navigateurs pour le rendu, la charge du réseau et la bande-passante, ainsi que la sécurité. Le mieux est de suivre les règles d'ergonomie du web.

V.3.Apple Web Application Guidelines

Apple fournit un document appelé « iPhone Human Interface Guidelines for Web Applications »³⁰ dans lequel sont fournis des éléments intéressants pour la création d'applications web destinée à l'iPhone.

J'ai également lu « iPhone and iPod Touch Programming - Building Applications for Mobile Safari »

Je vais en résumer les parties les plus pertinentes pour cette partie du projet.

V.3.1.Création d'une icône pour les applications web

À partir de Safari mobile, il est possible d'ajouter une icône vers une application mobile sur le springboard³¹. Un traitement graphique est automatiquement appliqué à l'icône pour qu'elle ait le style iPhone.

Une icône attractive donne envie à l'utilisateur d'utiliser l'application. L'icône devrait également être originale, pour se différencier d'autres icônes présentes sur le springboard et être repérée facilement par l'utilisateur.

Un style est appliqué automatiquement par iPhone OS à l'icône pour qu'elle ait la même apparence que les icônes des applications fournies par Apple.



Figure 32 : Une icône simple avant transformation en Web Clip



Figure 33 : Une icône affiche comme un Web Clip sur l'écran d'accueil

Pour que l'icône profite des améliorations graphiques apportées par le système, il faut respecter les points suivants :

- l'icône doit avoir une taille de 57 x 57 pixels, avec des angles à 90°,

³⁰ Ce document n'est plus disponible sur le site d'Apple.

³¹ Le springboard correspond à l'écran de l'iPhone sur lequel sont affichés les icônes permettant de lancer les applications, c'est l'équivalent du bureau d'un micro-ordinateur.

- pas de brillance sur l'image,
- la balise <head> de la page HTML doit comporter le lien vers l'icône :

```
<link rel="apple-touch-icon" gwc:attributes="href ressourceuri('webapp-  
icon.png', 'SetAjax') » />
```

V.3.2. Une mise en page adaptée à l'iPhone

Lorsque l'on crée l'interface utilisateur d'une application web dédiée à l'iPhone, il est utile de savoir quelle taille d'affichage est disponible pour le contenu. Par défaut, Safari Mobile affiche une barre de statuts, un champ texte pour l'URL de la page web et une barre des boutons.

La barre de statuts peut afficher le niveau de charge de la batterie, les informations sur les connexions réseau et l'heure. Au champ texte pour l'URL s'ajoute de chaque côté un bouton, le premier permet de gérer les favoris, l'autre de rafraîchir la page. Enfin, la barre des boutons s'affiche en bas de la page et contient les actions comme précédent, suivant ou afficher les pages chargées.

Safari Mobile affiche le contenu entre le bord inférieur du champ texte de l'URL et le bord supérieur de la barre des boutons comme nous pouvons le voir sur la figure 34.

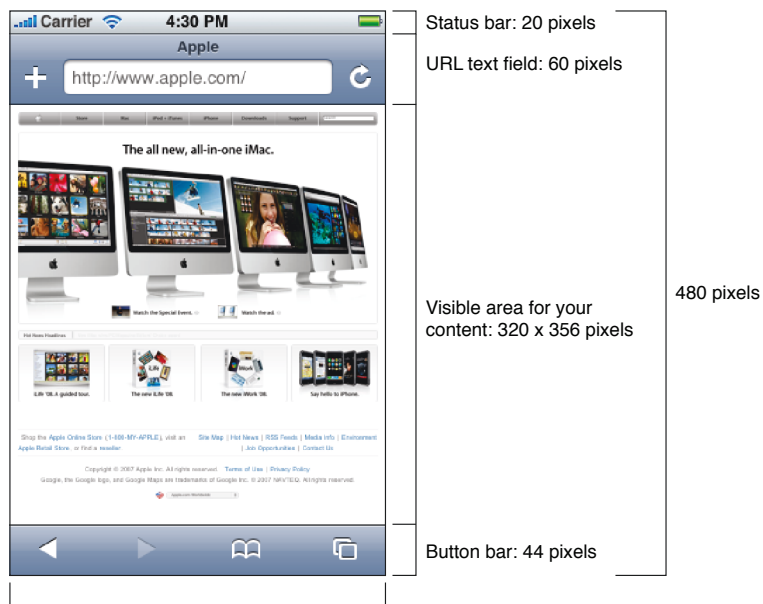


Figure 34 : Mesures en pixels en mode portrait

Lorsque l'iPhone est tenu horizontalement, l'affichage bascule en mode paysage et la taille d'affichage change telle que représentée ci-dessous sur la figure 36.



Figure 35 : Mesures en pixels en mode paysage

Il faut noter que la barre de statuts ne peut jamais être masquée.

Pour Safari mobile, il existe la propriété `viewport`, qui peut être défini dans le fichier HTML et qui correspond une surface rectangulaire qui déterminera comment la page sera affichée sur l'écran.

Si la propriété `viewport` n'est pas définie, Safari Mobile va automatiquement adapter l'affichage de la page de façon à ce que la page entière soit visible. De plus, Safari Mobile va fixer la valeur de la largeur du `viewport` à 980 pixels.



Figure 36 : Différence d'affichage d'une page web en spécifiant le `viewport` pour la largeur

Il est souhaitable de spécifier la propriété `viewport` si la largeur de la page web est très différente de 980 pixels, et plus spécialement si elle est moins large. En effet, si la largeur de la page est bien plus petite, l'adaptation de la page à l'écran de l'iPhone fera que le contenu sera

affiché trop petit, et sera donc difficile à lire pour l'utilisateur. Si la largeur est plus grande, l'utilisateur devra zoomer pour afficher tout le contenu de la page à l'écran.

Sur la figure 36, nous pouvons voir la même page affichée avant que son viewport soit défini et après qu'il ait été défini à 590 pixels.

Ceci est encore plus important pour les applications web que pour les pages web, car il faut que l'interface utilisateur remplisse l'espace de façon optimale, sans requérir de l'utilisateur qu'il ait à zoomer. Cela renforcera la perception de l'application web comme d'une vraie application native.

La figure 37 présente la même application web sans la propriété viewport définie, et avec l'attribut viewport correctement défini.

La propriété viewport possède plusieurs attributs, dont device-width et device-height qui permettent de spécifier la hauteur et la largeur de la zone d'affichage.

Apple recommande également de désactiver le zoom pour une application web destinée à l'iPhone. De cette façon, Safari Mobile ne zoomera pas automatiquement sur le contenu lorsque l'utilisateur fera un « double tap » ou un « pinch open ».



Figure 37 : Différence d'affichage d'une application web en spécifiant le viewport pour la largeur

V.3.3.L'approche par les listes

Lorsque l'on conçoit des applications web, il est fréquent de devoir afficher des informations similaires, ou partageant un même contexte. Apple préconise d'utiliser l'approche par liste.

En effet, la liste est une manière naturelle de présenter de l'information organisée par date, popularité, nom, localisation ou n'importe quel autre critère de tri. De surcroît, les listes sont conformes aux principes de simplicité, de facilité d'usage et de cohérence vus au chapitre II.5. Elles représentent une façon particulièrement efficace pour une application web destinée à l'iPhone d'afficher de l'information.

Les listes encouragent une mise en page claire et simple. Grâce à leur similitude avec un menu, les utilisateurs savent déjà comment elles fonctionnent.

Nous allons présenter dans cette partie deux possibilités d'affichages pour les listes, et donner les métriques et informations sur les styles recommandés par Apple.

V.3.3.1.Liste simple

Le premier style de mise en page est la liste simple, qui affiche chaque élément dans des lignes de taille identique. Cette mise en page est particulièrement adaptée aux applications qui doivent afficher un grand nombre d'éléments, comme des lieux, des noms ou des objets, les rendant facilement accessibles à l'utilisateur.

Lorsque l'utilisateur choisit un élément d'une liste simple, l'application web peut répondre :

- en affichant les détails de l'élément choisi, comme les informations de contact d'une personne, ou les informations sur les films à l'affiche d'un cinéma,
- en exécutant une action correspondant à l'élément sélectionné. Par exemple si la liste simple affiche une liste d'objets en vente, la sélection d'un objet pourrait afficher un bon de commande,
- en affichant une autre liste, plus restreinte, d'éléments qui permet à l'utilisateur de naviguer dans une hiérarchie d'éléments. Par exemple une application web qui permet de naviguer dans la liste des ouvrages d'une bibliothèque, avec comme première liste une liste de genres littéraires, puis une liste d'auteurs, et enfin la liste des ouvrages de chaque auteur.

Pour afficher une liste simple, il faut utiliser toute la largeur de l'écran pour laisser le plus de place possible au contenu. Chaque ligne ou cellule a 44 pixels de haut et 320 pixels de large.

La police recommandée est l'Helvetica, de corps 20 pixels, en noir. Il faudrait afficher le texte en gras par défaut, pour avoir une lisibilité maximale, mais il est possible d'utiliser la police en normal pour les informations les moins importantes (par exemple pour une liste de noms complets, utiliser le gras pour les noms et le normal pour les prénoms). Le texte devrait commencer à 10 pixels du côté gauche, et être à 14 pixels du bord inférieur de la ligne.

La couleur de fond est le blanc, et la couleur des lignes de séparation est la suivante : R=217, G=217, B=217.

S'il faut fournir des boutons, il est recommandé d'utiliser des boutons rectangulaires aux coins arrondis ayant 29 pixels de haut et un corner radius de 5 pixels. La police à utiliser pour le texte des boutons est l'Helvetica, en corps 12 pixels. Le bouton doit être positionné à 10 pixels du bord droit de l'écran, et être verticalement centré dans la ligne.

La figure 38 présente les métriques dans le détail, sous la forme d'une liste simple.

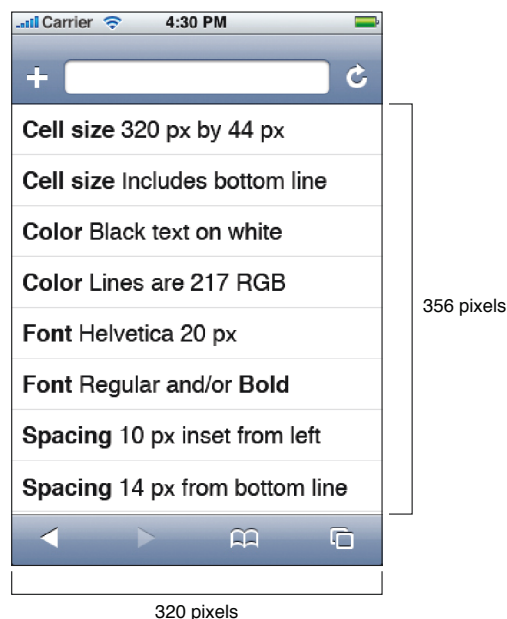


Figure 38 : Mise en page d'une table en liste simple

V.3.3.2. Liste groupée

Le second style de mise en page pour une liste est la liste groupée. Il affiche les informations dans des cadres séparés aux bords arrondis qui ont une couleur visuellement différente du fond de l'application web.

Ce type de mise en page est pratique pour afficher un petit nombre d'informations relatives à un seul sujet. Par exemple, pour afficher les informations de contact d'une personne, nous pourrions afficher les différents numéros de téléphone dans un cadre, les adresses électroniques dans un autre cadre et l'adresse dans un troisième cadre.

Cette mise en page est particulièrement adaptée pour afficher les informations de dernier niveau d'une liste simple. Dans notre exemple de l'application web permettant de parcourir les livres présents dans une bibliothèque, la liste groupée est parfaite pour afficher le dernier niveau d'information, c'est-à-dire le détail d'un livre : un petit résumé, la date de publication, l'éditeur, le code IBAN et la disponibilité par exemple.

Pour afficher les éléments de façon groupée, il est conseillé d'utiliser un cadre rectangulaire à bord arrondi (corner radius de 8 pixels), ayant une largeur de 300 pixels et avec un fond blanc. Les contours du cadre, ainsi que les lignes de séparation à l'intérieur d'un groupe auront la couleur : R=217, G=217, B=217.

Du fait qu'il y a un peu moins de place pour afficher le contenu par rapport à une liste simple, Apple recommande d'utiliser la police Helvetica en corps 17 pixels. Comme pour la liste simple, la police devrait être de couleur noire, et en gras par défaut pour une meilleure lisibilité.

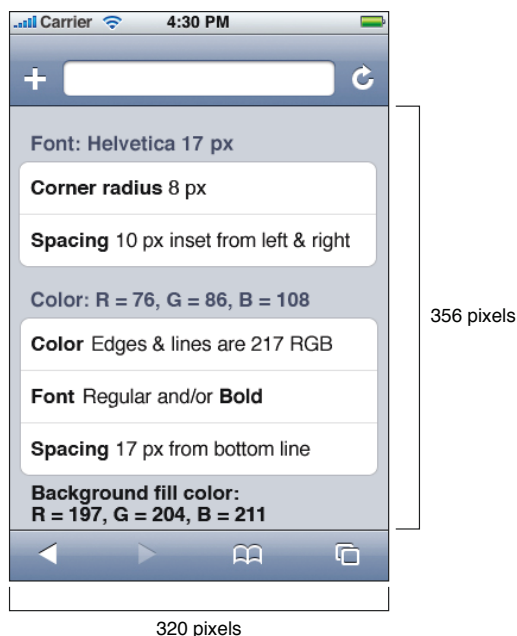


Figure 39 : Mise en page d'une table en liste groupée

À l'intérieur d'un cadre, le texte devrait être affiché à 10 pixels du bord gauche et à 14 pixels du bas. De plus, le cadre devrait être positionné à 10 pixels des bords gauche et droit de l'écran, et à 17 pixels du haut de la barre des boutons.

La figure 39 résume toutes ces informations et présente l'aspect d'une liste groupée.

V.3.4.Utilisation des éléments natifs

Dans certaines circonstances, Safari Mobile fournit à l'utilisateur certaines facilités, comme un clavier virtuel, une barre de navigation et un assistant pour les formulaires. Le fait que ces éléments soient natifs signifie qu'il n'y a pas de nécessité de les coder, mais cela signifie également qu'ils doivent être pris en compte lorsque l'on crée la mise en page.

Un formulaire HTML (HTML `<form>`) est une section d'un document HTML qui contient des éléments particuliers avec lesquels l'utilisateur peut interagir, par exemple des boutons, des cases à cocher et des champs de saisie de texte. Les actions des utilisateurs se résument en général à saisir du texte, choisir des éléments dans des menus ou cocher des cases à cocher avant de soumettre le formulaire au serveur web pour traitement.

Dans cette partie, nous allons décrire comment travailler avec les objets de l'interface utilisateur que Safari Mobile affiche dans certains cas.

V.3.4.1.Les claviers virtuels

Lorsqu'un utilisateur touche un champ de saisie (HTML `<input>`) sur le formulaire d'une page web, Safari Mobile affiche automatiquement un clavier virtuel et l'assistant à la place de la barre des boutons. L'assistant contient des boutons pour naviguer dans le formulaire ainsi qu'un bouton pour faire disparaître le clavier.

Il y a deux façons pour gérer l'affichage de formulaires. La première est de permettre à Safari Mobile de zoomer automatiquement et de centrer visuellement chaque contrôle au moment où l'utilisateur y opère sa saisie ou son action. L'autre façon est d'avoir une mise en page qui occupe tout l'espace de l'écran lorsque le clavier virtuel est affiché.

La première méthode s'applique surtout pour les pages web, et non pour les applications web qui cherchent à ressembler à des applications iPhone natives. La seconde méthode est recommandée dans le cas d'application web, et sous-entend que la propriété `viewport` a été spécifiée.

Dans ce cas, le zoom automatique est désactivé, et lorsque l'on s'occupe de la mise en page du contenu de l'application web, il faut avoir à l'esprit la place que prend le clavier virtuel à l'écran.

Lorsque le clavier virtuel et l'assistant de formulaire sont affichés, les métriques correspondent à la figure 40.

À noter que dans le mode paysage, la hauteur du clavier virtuel est de 162 pixels, et celle de l'assistant de formulaire est de 32 pixels. Il faut donc penser à tenir compte de la différence des dimensions du clavier entre le mode paysage et le mode portrait.

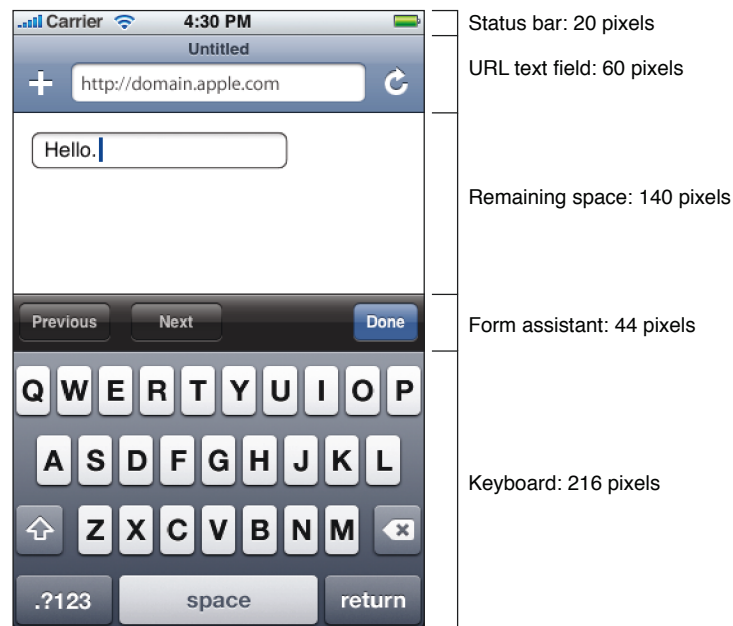


Figure 40 : Mesures de l'écran lorsque l'écran est affiché en mode portrait

V.3.4.2. Le menu déroulant

Lorsqu'un utilisateur touche et active un menu déroulant dans une application web sur iPhone, Safari Mobile affiche un élément graphique différent de ce qui se fait pour les menus déroulants dans un navigateur web de bureau ou dans une application de bureau classique.

En effet, vu la taille réduite de l'écran, il aurait été difficile de choisir un élément d'un menu déroulant classique sur un iPhone. C'est pourquoi Apple affiche les menus déroulants avec un style original de liste qui apparaît en bas de l'écran comme nous pouvons le voir sur l'image ci-dessous.

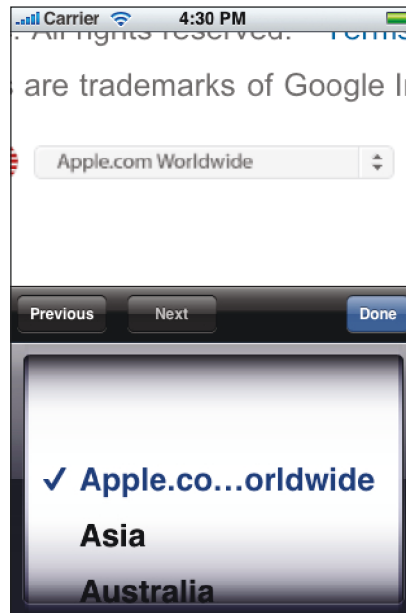


Figure 41 : Affichage par Safari mobile d'un menu déroulant

V.3.5. Création d'éléments de formulaires personnalisés

Le moteur de rendu WebKit, sur lequel est basé Safari mobile, permet la personnalisation des éléments graphiques grâce au support du CSS. Il est ainsi possible de créer des cases à cocher, champs de saisie de texte et autres contrôles personnalisés.

Bien qu'il soit possible de personnaliser les éléments graphiques, il faut se concentrer sur la création de contrôles qui sont faciles à utiliser, attractifs et cohérents. Il y a quelques points en particulier qu'il faut garder à l'esprit :

- créer des contrôles qui ont au minimum 29 pixels de hauteur avec une zone « cliquable » de 44 pixels de haut : l'utilisateur touche les contrôles avec ses doigts, donc des contrôles trop petits seraient frustrants et difficiles à utiliser. Suivre cette recommandation sur la taille des contrôles garantit que l'utilisateur ne va pas toucher par mégarde un contrôle adjacent,
- créer des contrôles visuellement cohérents entre eux : des instances différentes du même type de contrôles ne devraient pas trop différer en terme de rendu visuel et de taille à moins que cette différence ait une signification dans le contexte de l'application,
- coordonner l'aspect graphique des contrôles avec le reste de l'application pour renforcer son impact visuel : une interface agréable encourage les utilisateurs à utiliser l'application web.

V.3.5.1. Les contrôles natifs

De même que Safari pour ordinateurs de bureau, Safari Mobile fournit un style par défaut pour les contrôles de formulaire des applications web. Le style est spécifique à l'iPhone. La figure 42 présente l'apparence des contrôles natifs.

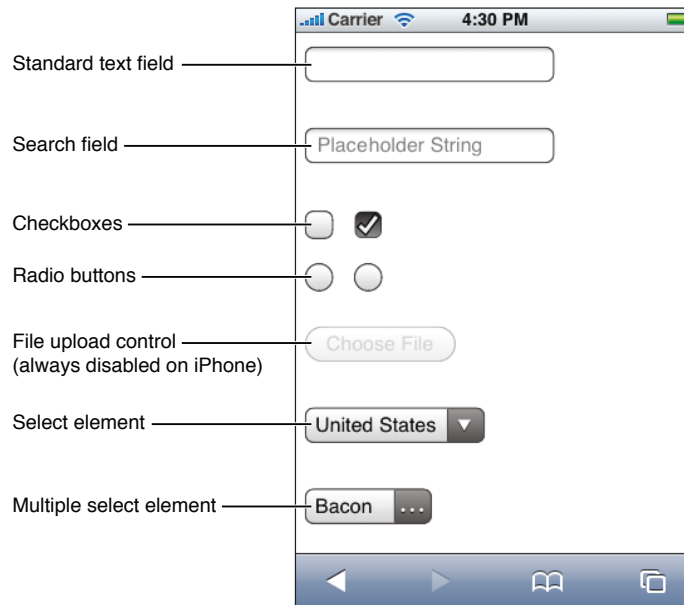


Figure 42 : Contrôles affichés de façon native par Safari mobile

Les cinq premiers éléments présentés sur la figure 42 sont obtenus en spécifiant différentes valeurs pour l'attribut type de la balise HTML `<input>`. La table ci-dessous présente les valeurs utilisées pour chaque type.

Description	Valeur de l'attribut type
Champs de saisie de texte standard	text (Notons que text est la valeur par défaut de l'attribut type)
Champs de recherche	search
Case à cocher	checkbox
Radio button	radio
Upload de fichier Ce contrôle est toujours désactivé, car l'upload de fichier n'est pas supporté sur iPhone	file

Tableau IV : Correspondance entre balise HTML et contrôles natifs

Les deux contrôles de sélection présentés sur la figure 42 sont tous les deux générés avec la balise HTML `<select>`. Pour afficher une liste à sélection simple, il faut utiliser la balise HTML `<select>` associée à autant de tags HTML option nécessaires. Pour une liste à sélection multiple, il

faut rajouter l'attribut multiple à la balise HTML select. Lorsque l'utilisateur touche un de ces contrôles, un menu déroulant s'affiche comme décrit dans la partie précédente.

Notons pour finir que les contrôles natifs sont transparents de façon à ce qu'ils s'intègrent au mieux dans n'importe quelle application web quelque soit le fond choisi pour cette application.

V.3.6.L'attention portée au texte

Apple recommande de suivre les principes de design de base lorsqu'il s'agit du texte et d'éviter les polices, couleurs et tailles de polices difficiles à lire. Il est également conseillé d'afficher les textes en mode colonnes afin de faciliter la lecture des informations.

Il y a quelques points qu'il est préférable de suivre pour s'assurer de la lisibilité du texte :

- utiliser une police ayant une taille de 17 à 22 pixels,
- utiliser le gras pour mettre en avant les informations importantes, pour décrire des éléments d'une liste ou pour montrer une hiérarchie dans l'information,
- faire des labels aussi courts que possible, commencer le mot par une majuscule et ne pas le finir par de doubles points,
- aligner le texte à gauche, en particulier dans une liste,
- éviter d'utiliser le style souligné pour afficher des liens, car cela fait apparaître le texte comme surchargé.

V.4.Cadriels Web

Il existe plusieurs bibliothèques web basées sur HTML, CSS et JavaScript aidant à la réalisation d'applications web optimisées pour iPhone. Ces cadriels pourront nous servir de base de travail, c'est pourquoi je vais présenter les plus intéressants dans le détail et listerai les autres.

Leurs fonctionnements étant similaire, je ne présenterai le détail de mise en place et d'utilisation que du premier présenté : iUi.

V.4.1.iUi

iUI³² est une bibliothèque open source compacte (10 Ko pour la version compressée du code JavaScript) qui permet de donner à une page HTML affichée le style iPhone, à l'aide de divers fichiers CSS livrés avec celle-ci. Le code JavaScript permet quant à lui d'enchaîner le défilement

³² <http://code.google.com/p/iui/>

des pages HTML entre elles, la plupart du temps sans faire appel au serveur (sauf pour les appels AJAX).

iUi permet de créer des menus de navigations identiques à ceux des applications iPhone natives. De plus, la plupart des éléments graphiques tels que les boutons, champs de saisie de texte et autres sont présents.

iUI utilise certaines combinaisons de balises HTML pour reproduire des éléments de l'interface graphique de l'iPhone. Cet outil utilise principalement les lignes (balise) pour réaliser la mise en page. Un fichier CSS et un fichier JavaScript sont à associer aux pages HTML pour exploiter les fonctionnalités d'iUi.

Cette bibliothèque est recommandée pour naviguer dans des informations hiérarchiques, les fonctionnalités sont néanmoins limitées. Les performances sont correctes, sans plus.

La documentation est très succincte, mais cette bibliothèque est la référence à l'heure actuelle.

V.4.1.1.Mise en place

La mise en place de ce cadriciel se fait simplement dans l'entête (balise <head>) du fichier HTML comment nous pouvons le voir sur la figure 43. Le viewport est positionné, une icône spécifiée et les fichiers CSS et JavaScript de iUI sont chargés.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<title>iUI Demo</title>
<meta name="viewport" content="width=device-width;
initial-scale=1.0; maximum-scale=1.0; user-
scalable=0;"/>
<link rel="apple-touch-icon" href="iui-logo-touch-
icon.png" />
<meta name="apple-touch-fullscreen" content="YES" />

<style type="text/css" media="screen">
  @import "iui.css";
</style>

<script type="application/x-javascript"
src="iui.js"></script>

</head>
```

Figure 43 : Code HTML pour la mise en place du cadriciel web iUi

V.4.1.2.Utilisation

Le cadriciel iUI repose essentiellement sur l'utilisation de liste (balises et) pour l'affichage des données. Des classes particulières sont utilisées et permettent un affichage proche de celui d'une application iPhone native.

```
<body>
  <div class="toolbar">
    <h1 id="pageTitle"></h1>
    <a id="backButton" class="button" href="#"></a>
    <a class="button" href="#searchForm">Search</a>
  </div>

  <ul id="home" title="Music" selected="true">
    <li><a href="#artists">Artists</a></li>
    <li><a href="#settings">Settings</a></li>
    <li><a href="stats.php">Stats</a></li>
    <li><a href="http://code.google.com/p/iui/"
target="_self">About</a></li>
    <li>Nothing</li>
  </ul>
  <ul id="artists" title="Artists">
    <li class="group">B</li>
    <li><a href="#TheBeatles">The Beatles</a></li>
    <li><a href="#BelleSebastian">Belle & Sebastian</a></li>
    <li class="group">C</li>
    <li><a href="#CrowdedHouse">Crowded House</a></li>
    <li class="group">J</li>
    <li><a href="#JennyLewis">Jenny Lewis</a></li>
    <li><a href="#JohnMayer">John Mayer</a></li>
    <li class="group">Z</li>
    <li><a href="#Zero7">Zero 7</a></li>
  </ul>
```

Figure 44 : Exemple de code HTML d'utilisation du cadriciel web iUi

V.4.1.3.Résultats

Nous pouvons voir sur la figure 45 une application web réalisée avec iUi. La première représente une liste permettant d'afficher un menu. La seconde affiche des propriétés avec quelques contrôles classiques comme une case à cocher et des champs de saisie de texte.



Figure 45 : Exemple d'application développée avec le cadriciel web iUi

V.4.2.jQTouch

jQTouch³³ est un plug-in JQuery open source permettant de développer des applications web mobiles avec un « look and feel » natif en utilisation HTML, CSS et JavaScript.

jQTouch gère de façon native les animations WebKit ainsi que la navigation. jQTouch gère également un mode non connecté. Il est possible de personnaliser l'apparence des applications à l'aide de thèmes. Certains gestes multipoints comme le « swipe » sont gérés.

jQTouch utilise la bibliothèque JQuery, une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et qui a pour but de simplifier des commandes communes de JavaScript.

jQTouch est plus rapide que iUi, et gère également la géolocalisation. Les transitions sont plus fluides et mieux gérées, mais jQTouch est plus complexe que iUi et nécessite obligatoirement l'utilisation de JQuery.

V.4.3.WebApp.net

WebApp.net³⁴ est un cadriciel open source très léger destiné aux iPhone, iPod touch et navigateurs mobiles basés sur WebKit. Ce cadriciel est lui aussi basé sur le CSS et le JavaScript.

³³ <http://jqtouch.com/>

³⁴ <http://webapp-net.com/>

Il est utilisé pour quelques applications web mobiles commerciales comme celle des transports publics de Genève³⁵.

La documentation est très complète et ce cadriciel est réellement simple à utiliser. De plus, il offre un certain support des fichiers multimédia et des fonctionnalités de débogage.

WebApp.net offre un mode plein écran ainsi que le support des PNG (Portable Network Graphics) animés.

V.4.4.iWebKit

iWebKit³⁶ est un ensemble de fichiers HTML, CSS et JavaScript permettant de créer des applications web ayant le « look and feel » de l'iPhone. Il est sous licence LGPL (Lesser General Public License). iWebKit est rapide, simple et facile à faire évoluer.

iWebKit permet de créer des menus de navigation identiques à ceux des applications iPhone natives. Il permet également de créer des onglets. Il supporte plus d'éléments graphiques qu'iUi et l'aspect des applications obtenues est plus fini.

iWebKit est facilement extensible, simple et minimaliste. Il est lui aussi plus rapide qu'iUi.

La documentation est complète et ce cadriciel est mature.

V.4.5.Sencha Touch

Sencha Touch³⁷ est un cadriciel HTML5 qui permet de créer des applications web mobiles avec un « look and feel » natif pour iOS, mais aussi pour Android. Ce cadriciel est disponible sous deux licences : une licence GPL (General Public License) open source, et une licence commerciale.

Sencha Touch est assez complet et gère de nombreux contrôles tactiles. Il est de surcroît très rapide et la documentation est complète. Il est enfin possible d'avoir un support payant.

V.4.6.Cappuccino et SproutCore

Cappuccino³⁸ et SproutCore³⁹ sont des cadriciels plus évolués. Ils sont open source tous les deux.

³⁵ <http://iphone.tpg.ch>

³⁶ <http://snippetspace.com/projects/iwebkit/>

³⁷ <http://www.sencha.com/products/touch/>

³⁸ <http://cappuccino.org/>

³⁹ <http://www.sproutcore.com/>

Cappuccinno est inspiré de Cocoa et de Mac OS X et basé sur l'Objective-J (un langage inspiré de l'Objective-C qui se compile en JavaScript) et SproutCore est basé sur le HTML 5.

Ils peuvent être utilisés pour créer des applications web mobiles à destination de l'iPhone, mais ils en sont encore au commencement de leur développement. Il pourra être intéressant de les suivre.

V.4.7.PastryKit

Apple a développé un cadriciel JavaScript appelé PastryKit qu'elle a mis en application pour l'aide en ligne de l'iPhone. C'est sans doute la première fois qu'une application web ressemble à ce point à une application native avec un niveau de performances assez correct.

Comme pour une application native iPhone, ce cadriciel permet de créer une liste d'éléments qui fonctionne de manière identique au niveau du défilement, de remplacer la barre d'URL par une barre d'information qui reste au sommet de l'écran même lorsque l'on fait défiler la liste, de sauvegarder des données pour une consultation déconnectée.

Néanmoins, ce cadriciel n'a pas été mis à disposition des développeurs pour le moment et n'est absolument pas documenté. Il est de ce fait difficilement exploitable tel quel.

V.4.8.Les autres cadriciels

Il existe encore d'autres cadriciels et bibliothèques tels que Safire, PhoneGap, LiquidGear, XUI, Rhodes, jPint, Magic Framework qu'il est inutile de détailler ici, car ils sont moins aboutis que ceux présentés dans ce document.

V.5.Conclusion

Grâce aux recherches réalisées dans ce chapitre, j'ai pu me rendre compte que la plupart des outils nécessaires à la réalisation de ce projet existent déjà, aussi bien en interne avec le SBRE et le thème AJAX qu'en externe avec les nombreux projets visant à reproduire l'aspect des applications iPhone natives.

Enfin, Apple fournit des informations claires et détaillées pour la réalisation d'applications web mobiles.

VI. Recherche de solutions

Une fois l'étude de l'existant réalisée, je suis passé à l'étape de la recherche de solution. Je vais décrire dans ce chapitre les étapes et recherches qui m'ont amené à la solution choisie.

VI.1.Évolution du thème AJAX

Dans un premier temps, j'ai commencé par regarder s'il n'était pas possible d'adapter directement le thème AJAX en changeant simplement l'apparence des différents éléments graphiques.

Pour cela, j'ai essayé d'exploiter les différents cadres existants vus au chapitre précédent. J'ai extrait les fichiers CSS et je les ai adaptés pour les appliquer aux éléments graphiques du thème AJAX.

Néanmoins, je me suis rapidement retrouvé confronté à un problème de taille : la façon dont la page est affichée avec le thème AJAX.

En effet, le thème AJAX a été fait pour reproduire au mieux l'interface obtenue avec le GDC. Le GDC lui-même a comme contrainte de respecter le plus précisément possible la définition de la fenêtre comme le développeur de l'application l'a construite dans le fichier de description .per.

Pour plus de clarté, je vais présenter un fichier .per, décrire comment le GDC utilise ce fichier pour son réaliser l'interface graphique, puis expliquer comment le thème AJAX reproduit cette même interface.

VI.1.1.Fichier .per

Un fichier .per définit l'interface d'une application. Ce fichier spécifie la disposition et la présentation des éléments, ainsi que certains comportements des éléments graphiques.

Un fichier .per est composé de plusieurs sections, apparaissant dans l'ordre suivant :

- la section `SCHEMA` : identifie le schéma de base de données sur lequel la fenêtre est basée, les champs de la fenêtre peuvent automatiquement être associés à des champs de la base de données,
- la section `ACTION DEFAULTS` : définit et centralise les actions locales pour les éléments de la fenêtre,
- la section `TOPMENU` : définit un menu déroulant avec des options qui sont liées à des actions,

- la section `TOOLBAR` : définit une barre d'outils avec des boutons liés à des actions,
- la section `LAYOUT` : définit l'emplacement des éléments graphiques de la fenêtre,
- la section `TABLES` : liste toutes les tables ou vues de base de données référencées dans le fichier `.per`,
- la section `ATTRIBUTES` : définit les propriétés des éléments graphiques utilisés dans la fenêtre,
- la section `INSTRUCTIONS` : permet de définir certains tableaux de données, affichés ou non sur la fenêtre.

Chaque section débute par son mot clef. Les sections `LAYOUT` et `ATTRIBUTES` sont obligatoires, les autres sont optionnelles.

La section `LAYOUT` est celle qui nous intéresse tout particulièrement et je vais détailler.

VI.1.1.1. La section `LAYOUT`

La présentation de la fenêtre se compose d'un arbre de conteneurs. Différents conteneurs sont disponibles, chacun ayant un rôle spécifique. Certains conteneurs peuvent avoir des conteneurs fils, alors que d'autres conteneurs peuvent définir des zones de l'écran. Ces derniers définissent des régions formatées de l'écran contenant des labels de texte statique, des éléments graphiques et des éléments de mise en page.

Les éléments graphiques de la fenêtre, par exemple un champ de saisie de texte, un menu déroulant ou un groupe de boutons, sont encapsulés dans ces conteneurs.

```

LAYOUT
  VBOX
    GRID
      grid-area
    END
    GROUP
      HBOX
        GRID
          grid-area
        END
        TABLE
          table-area
        END
      END
    END
  END
END
  
```

Figure 46 : Exemple de mise en page d'application (`.per`)

Il faut terminer chaque bloc avec le mot clef `END`.

La figure 46 présente un exemple de section LAYOUT avec différents conteneurs. Un premier conteneur VBOX contenant un conteneur GRID et un conteneur GROUP, lui-même ayant un conteneur fils HBOX dans lequel se trouvent deux autres conteneurs, un GRID et un TABLE.

Les différents conteneurs disponibles sont les suivants :

- VBOX : présente les éléments encapsulés verticalement, sans aucune décoration,
- HBOX : présente les éléments encapsulés horizontalement, sans aucune décoration,
- GROUP : présente les éléments dans un cadre ayant un titre,
- FOLDER : présente les éléments PAGE sous forme d'onglets,
- PAGE : définit le contenu d'un onglet d'un conteneur FOLDER,
- GRID : présente des éléments encapsulés uniques avec des labels et des champs positionnés. La plupart des composants graphiques se positionnent dans des GRID,
- SCROLLGRID : présente des éléments encapsulés multiples avec des labels et des champs positionnés,
- TABLE : contient des données présentées sous la forme d'un tableau avec des colonnes et des lignes,
- TREE : contient des données présentées sous la forme d'une arborescence.

Le principe est de respecter une grille pour la disposition des éléments.

VI.1.1.2.Exemple 1

Je vais présenter un premier exemple très simple afin d'expliquer le concept.

```
GRID
{
  First Name [fname ]
  Last Name  [lname ]
}
END
```

Figure 47 : Code représentant une application simple

La figure 47 présente le code d'un fichier .per représentant une grille avec 2 labels : « First Name » et « Last Name » et deux champs textes associés : « fname » et « lname ». Ces données peuvent être très facilement représentées sur une grille comme présentées sur la figure 48.

```

G R I D
{
                                1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 F i r s t   N a m e   [ f n a m e   ]
1 L a s t   N a m e   [ l n a m e   ]

}
E N D

```

Figure 48 : Représentation graphique d'une application sur une grille

Dans une interface texte ou avec une interface graphique à taille de caractères fixes, c'est très facile de respecter la grille. Néanmoins Genero a introduit les polices proportionnelles, les objets sont créés et ajoutés à la grille avec une position de départ (un attribut X et un attribut Y) et un nombre de cellules utilisées (width et height). La figure 49 ci-après présente cela de façon détaillée.

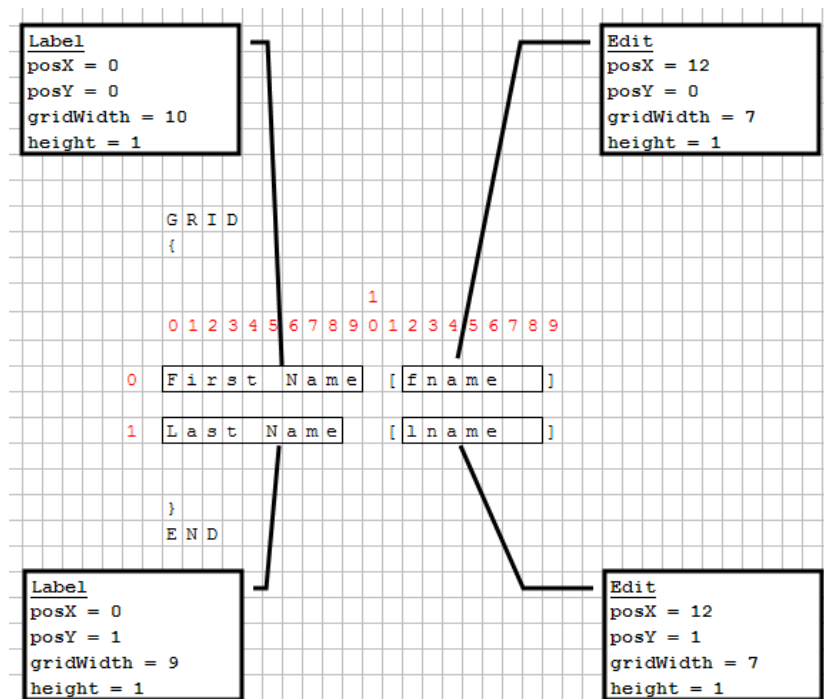


Figure 49 : Calcul des positions des éléments sur la grille

VI.1.1.3.Exemple 2

Je vais présenter un second exemple un peu plus complexe qui me servira à présenter le rendu exact d'un fichier .per.

```

LAYOUT ( TEXT="My Todo List" )

FOLDER

PAGE Main (TEXT="Tasks")
TABLE ( DOUBLECLICK=modify )
{
  Task                Due date  State
[m_name              |m_duedate |m_status]
[m_name              |m_duedate |m_status]
[m_name              |m_duedate |m_status]
[m_name              |m_duedate |m_status]
}
END --TABLE
END --PAGE Main

PAGE about (TEXT="About")
GRID
{
[m_title             ]
[m_logo              ]
[                    ]
[                    ]
[                    ]
}
END --GRID
END --PAGE about

END --FOLDER
END --LAYOUT

ATTRIBUTES
  EDIT m_name=formonly.m_name, SCROLL;
  PHANTOM formonly.m_description;
  PHANTOM formonly.m_priority;
  CHECKBOX m_status=formonly.m_status, VALUECHECKED="1", VALUEUNCHECKED="0";
  DATEEDIT m_duedate=formonly.m_duedate;
  LABEL m_title : m_title, TEXT="Demo for iPhone theme";
  IMAGE m_logo : m_logo, IMAGE="card-lolo.png";
END

```

Figure 50 : Exemple de fichier .per

La figure 50 présente le code d'un fichier .per représentant une application basique de gestion de tâches. On peut voir que cette application se présente sous forme de page (FOLDER) avec deux onglets (PAGE).

Le premier onglet, identifié par le mot clef « Main », correspond à celui des tâches. Il contient une simple table (TABLE) avec trois colonnes. Une pour le nom de la tâche (« Task »), une autre pour la date à laquelle la tâche doit être accomplie (« Due date ») et une dernière pour le statut de la tâche (« State »). En regardant la section des attributs (ATTRIBUTES), on remarque que le nom est un champ d'édition de texte (EDIT), la date est un champ d'édition de la date (DATEEDIT) et le statut est une case à cocher (CHECKBOX) qui prend deux valeurs différentes en fonction de son état (0 ou 1).

Le second, identifié par le mot clef « About » correspond à une page d'informations sur l'application. Il contient un titre et un logo. Dans la section des attributs, on constate que le titre est un simple label (LABEL) valant « Demo for iPhone theme » et que le logo est une image (IMAGE) correspondant au fichier « card-lolo.png ».

VI.1.2. Le rendu GDC

La figure 51 correspond à une capture d'écran de l'application de gestion de tâches introduite précédemment exécutée à travers le GDC.

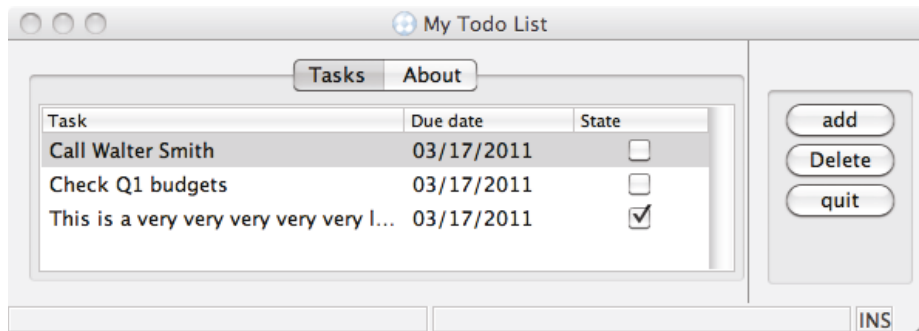


Figure 51 : Rendu de l'exemple sous Mac OS X avec le GDC

On remarque que tous les éléments décrits dans le .per sont présents. Les deux onglets, avec le contenu du principal affiché, la taille de la table, des champs, et de l'application correspond à la taille définie dans le .per (où un espace correspond à une taille fixe dans l'application).

Les 3 boutons dans l'espace de droite ne sont pas définis dans le .per mais sont programmatically créés dans les fichiers sources de l'application.

VI.1.3. Le rendu GWC

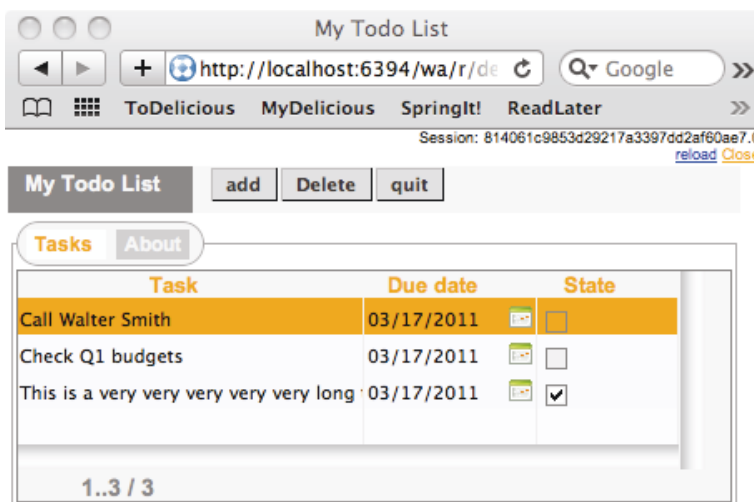


Figure 52 : Rendu de l'exemple dans un navigateur avec le thème AJAX

La figure 52 correspond à une capture d'écran de l'application exécutée dans un navigateur web en passant par le GAS et en utilisant le thème AJAX.

À quelques détails près, comme l'emplacement des boutons « add », « Delete » et « quit », et l'aspect de l'application qui n'est pas native au système d'exploitation côté client, le rendu est identique à celui du GDC.

Le principe fondamental associant une taille fixe à l'écran à chaque espace dans le .per est bien respecté.

VI.1.4.Problématique

Pour reproduire au mieux l'aspect du rendu du GDC, le GWC doit créer une page web permettant de maîtriser au mieux la mise en page. Le GWC devant fonctionner avec de nombreux navigateurs, et surtout avec des versions relativement anciennes (comme Internet Explorer 6), la seule façon possible de le faire est de découper entièrement la page en une table, avec chaque élément du .per correspondant à une cellule de cette table. Les espaces vides sont eux aussi représentés par une cellule de cette table.

Or aucun des cadriciels web permettant de reproduire l'interface graphique de l'iPhone n'utilise de table, car cela alourdit la mise en page et n'est pas nécessaire dans ces cadriciels.

La première chose à faire sera donc de revoir comment les écrans des applications seront mis en page en cherchant un moyen différent de celui du thème AJAX.

VI.2.Mise en page des applications

J'ai donc constaté lors de mes recherches précédentes qu'il ne serait pas possible d'utiliser directement les cadriciels web existants sur le thème AJAX du fait de la façon dont les écrans des applications étaient mis en page. Il fallait donc concevoir une façon nouvelle de générer cette mise en page.

Une seconde problématique m'est apparue lors de ces recherches. Dans le thème AJAX et avec le GDC, les écrans des applications pouvaient avoir des tailles différentes, allant du très petit pour les écrans simples à des tailles bien plus grandes pour des écrans plus complexes.

Or l'écran de l'iPhone a une taille et une résolution réduites. De plus, les directives d'Apple conseillent d'afficher à l'écran tous les éléments de l'application et de faire en sorte que ces éléments aient une certaine taille. En effet, un doigt est bien moins précis que le pointeur d'une souris, et des éléments trop petits ou trop rapprochés rendraient l'application inutilisable.

La solution qui m'est apparue être la plus évidente fut d'utiliser une mise en page de largeur fixe et correspondant à la largeur de l'iPhone, c'est-à-dire 320 pixels.

VI.2.1.Limitation de la grille

Cette limitation de la largeur de l'application fut décidée, mais la contrainte par rapport au .per demeurait. C'est-à-dire que les éléments du .per devait toujours respecter la règle de correspondance : un caractère égal à une taille fixe sur l'écran.

Cette taille fixe sur l'écran a été fixée à 12 pixels de large, correspondant à la largeur maximale d'un caractère, pour 44 pixels de haut, correspondant à la norme proposée par Apple dans ses directives.

Je me suis tout d'abord intéressé à la largeur, car celle-ci est totalement fixe, alors qu'il est toujours possible de naviguer dans l'écran verticalement.

Sachant qu'Apple recommande une marge de 10 pixels à gauche de l'écran, j'ai décidé d'appliquer la même marge à droite pour que les informations soient centrées. De ce fait, il reste 300 pixels ($320 - 10 * 2$) pour les caractères, soit 25 caractères ($300/12$). Au-delà de cette limite, il ne serait tout simplement pas possible d'afficher l'application sur l'écran de l'iPhone.

Cette décision de limiter en largeur la taille des applications dut être approuvée par le directeur de projet, car l'objectif initial de pouvoir afficher toutes les applications Genero existantes ne pouvait pas être rempli de façon satisfaisante.

VI.3.Réduire la complexité

VI.3.1.Genero Mobile Profile

À partir du moment où le constat qu'il ne serait pas possible d'afficher toutes les applications Genero existantes a été fait, j'ai cherché quels étaient les éléments essentiels qu'il faudrait garder pour réaliser des applications Genero pour iPhone intéressants et exploitables.

Un profil décrivant quels éléments de LAYOUT était supporté, et dans quel ordre, a été réalisé. Ce profil a été nommé « Genero Mobile Profile » et est présenté sur la figure 53.

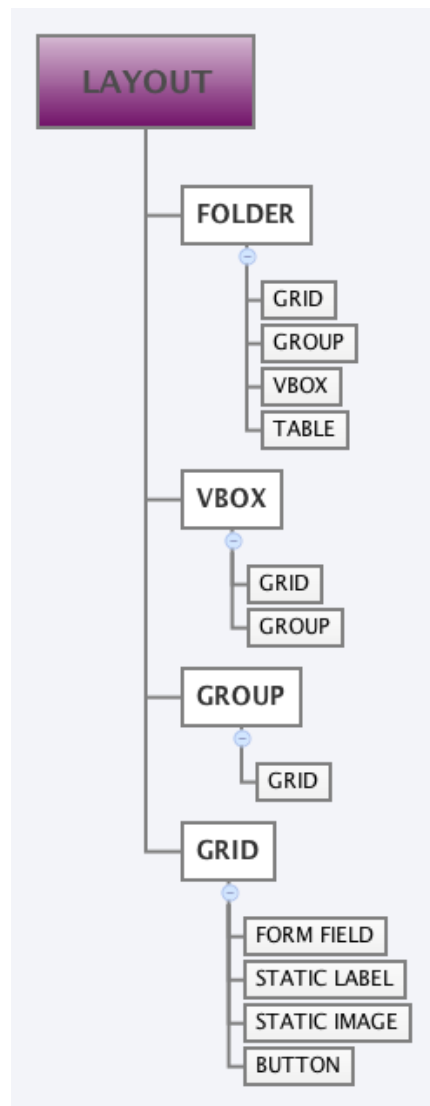


Figure 53 : Mises en page supportées par le thème iPhone

Globalement, l'idée retenue était de limiter les possibilités offertes aux développeurs utilisant Genero pour créer leurs écrans, ceci afin de leur permettre de réaliser des applications mobiles respectant les meilleures pratiques du développement d'applications mobiles.

Finalement, ce principe de n'offrir qu'un jeu réduit de possibilités rejoint le concept d'applications mobiles à but précis et concentré sur une ou deux tâches.

VI.3.2.La question des tables

VI.3.2.1.Problème

Rejoignant la problématique précédente est celle de la gestion des tables. J'ai cherché en vain un moyen satisfaisant d'afficher des tables avec plus d'une dizaine de colonnes sur une taille d'écran réduit.

The screenshot shows a web browser window with the URL <http://localhost:6363/wa/r/demo/bigtable>. The browser's address bar and tabs are visible at the top. Below the browser interface, a table is displayed with 18 columns labeled a1 through a18. The table has multiple rows, with the first row highlighted in yellow. The bottom status bar of the browser indicates '1.6 / 10'.

Figure 54 : Exemple d'une table avec beaucoup de colonnes

Pour répondre à ce problème, j'ai regardé en détail comment les tables étaient affichées par les applications iOS existantes ainsi que la façon dont elles étaient traitées par les différents cadres web vus précédemment.

VI.3.2.2.Représentation

Tous les cadres web que j'ai analysés utilisent le principe de liste pour représenter les données sur iPhone sous forme de table. J'ai donc décidé de garder ce principe.

Néanmoins, lorsqu'il y a plusieurs colonnes dans la table, les cadres web étudiés ont deux façons d'afficher les données. La première façon est présentée avec la figure 55 (exemple du cadre iWebkit) et correspond à afficher une colonne de donnée par ligne.

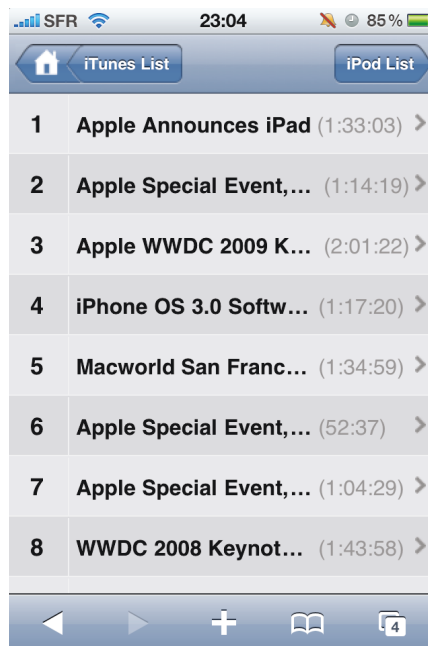
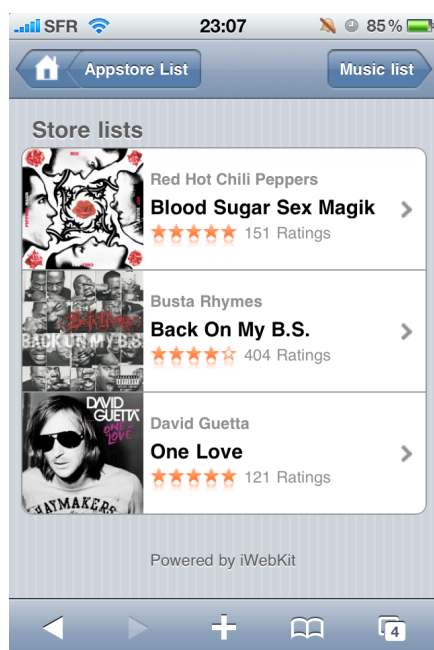


Figure 55 : Affichage sur l'iPhone de données d'un tableau en ligne

L'autre manière, présentée sur la figure 56 (exemple du cadriciel iWebkit), prend le parti pris d'afficher une colonne par ligne, et de regrouper les colonnes d'une même ligne dans une ligne plus grande, séparée visuellement de la ligne de données suivante.



gure 56 : Affichage sur l'iPhone de données d'un tableau en grou

J'ai choisi la deuxième solution, car elle permet d'un côté d'afficher les titres des colonnes de la table avant la valeur du champ, et de plus elle me permet d'afficher des données plus longues.

VI.3.2.3.Limitation

J'ai constaté que les tables des applications mobiles ne contenaient finalement que peu de colonnes. C'est pourquoi, une fois encore, j'ai choisi de limiter les fonctionnalités et de restreindre à 3 le nombre de colonnes que l'utilisateur allait pouvoir utiliser.

Néanmoins, il s'agira de laisser la possibilité à l'utilisateur de lever cette limitation s'il le souhaite.

VI.3.2.4.Exemple

La figure 57 montre une capture d'écran de l'exemple présenté précédemment, l'application simple de gestion des tâches, tel qu'elle est affichée par le thème iPhone.

Nous voyons bien le titre de chaque colonne avant sa valeur, et ceci répété pour chaque ligne du tableau.



Figure 57 : Rendu de l'exemple sur l'iPhone avec le thème iPhone

VI.3.3. Le mode d'affichage

L'iPhone offre deux modes d'affichages : le mode portrait et le mode paysage. Il aurait été techniquement possible de supporter les deux modes d'affichages. Néanmoins, par souci de simplicité et de temps lors du développement, il a été décidé de ne supporter que le mode portrait.

VI.4. Réalisation

Je vais présenter dans cette partie quelques exemples de réalisations : le traitement de la grille de mise en page des applications, la gestion des éléments non supportés et enfin le support de la géolocalisation.

VI.4.1. Grille

Trois snippets sont utilisés par le SBRE pour générer la mise en page en plus du modèle de base (main.xhtml).

C'est tout d'abord le snippet Form (form.xhtml, figure 58) qui est utilisé par le SBRE. À la ligne 28, nous voyons qu'un test est exécuté sur le type d'éléments conformément aux limitations présentées sur la figure 53. Si l'élément est supporté, le snippet correspondant est appelé (ligne 31), sinon l'écran n'est pas supporté (ligne 30).

```

1 |<?xml version="1.0"?>
2 |<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
3 |transitional.dtd">
4 |<!--
5 |FOURJS_START_COPYRIGHT(D,2010)
6 |Property of Four Js*
7 |(c) Copyright Four Js 2010, 2011. All Rights Reserved.
8 |* Trademark of Four Js Development Tools Europe Ltd
9 |in the United States and elsewhere
10 |
11 |This file can be modified by licensees according to the
12 |product manual.
13 |FOURJS_END_COPYRIGHT
14 |-->
15 |<html xmlns:gwc="http://www.4js.com/GWC">
16 |  <!-- the head element is ignored -->
17 |  <head>
18 |    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
19 |    <title>Form snippet</title>
20 |  </head>
21 |  <body>
22 |    <!-- the template snippet is the content of the gwc:snippet-root element -->
23 |    <gwc:snippet-root>
24 |      <!-- Form unsupported feature detection-->
25 |      <div gwc:condition="topmenu"
26 |        class="notsupported">TOP MENU NOT SUPPORTED</div>
27 |      <div gwc:condition="toolbar"
28 |        class="notsupported">TOOLBAR NOT SUPPORTED</div>
29 |      <!-- Form rendering : the form main component is a VBox -->
30 |      <div gwc:condition="(item/type != 'VBox') &amp;&amp; (item/type != 'Folder') &amp;&amp; (item/
31 |        type != 'Group') &amp;&amp; (item/type != 'Grid')"
32 |        class="notsupported">ELEMENT {item/type} NOT SUPPORTED</div>
33 |      <div gwc:replace="item"/>
34 |    </gwc:snippet-root>
35 |  </body>
36 |</html>

```

Figure 58 : Snippet Form.xhtml

Le cas qui nous intéresse est celui de la grille (Grid) et la macro `gwc:replace` indique au SBRE de remplacer la balise HTML courante par le snippet correspondant : Grid.xhtml.

```

1 |<?xml version="1.0"?>
2 |<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
3 |transitional.dtd">
4 |<!--
5 |FOURJS_START_COPYRIGHT(D,2010)
6 |Property of Four Js*
7 |(c) Copyright Four Js 2010, 2011. All Rights Reserved.
8 |* Trademark of Four Js Development Tools Europe Ltd
9 |in the United States and elsewhere
10 |
11 |This file can be modified by licensees according to the
12 |product manual.
13 |FOURJS_END_COPYRIGHT
14 |-->
15 |<html xmlns:gwc="http://www.4js.com/GWC">
16 |  <!-- the head element is ignored -->
17 |  <head>
18 |    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
19 |    <title>Grid snippet</title>
20 |  </head>
21 |  <body>
22 |    <!-- the template snippet is the content of the gwc:snippet-root element -->
23 |    <gwc:snippet-root>
24 |      <div gwc:condition="hidden!=1" gwc:replace="layout" />
25 |    </gwc:snippet-root>
26 |  </body>
27 |</html>

```

Figure 59 : Snippet Grid.xhtml

La figure 59 présente le snippet Grid.xhtml qui est en fait un snippet très simple qui ne fait qu'appeler un autre snippet (Layout) dans le cas où le snippet est visible. Cela se passe à la ligne 23 et le test de visibilité est fait à l'aide d'une fonction du SBRE : `gwc:condition="hidden!=1` ».

Le snippet le plus compliqué est celui de la mise en page à proprement parler de la grille : GridLayout.xhtml. Son code principal est montré sur la figure 60.

```

21 <body>
22 <gwc:snippet-root>
23 <div gwc:condition="width > 25" class="notsupported">GRID TOO WIDE ({width})<br/>max is 25</div>
24 <div
25 class="iPhoneGridLayout"
26 gwc:condition="width <= 25"
27 style="height:{height*44}px;"
28 >
29 <gwc:omit
30 gwc:omit-tag="true"
31 gwc:repeat="l tableLines"
32 >
33 <gwc:omit
34 gwc:omit-tag="true"
35 gwc:repeat="c l/cells"
36 >
37 <div
38 class="iPhoneGridLayoutCell"
39 style="top:{c/y*44}px;left:{c/x*12}px;width:{c/width*12}px;height:{c/height*44}px;"
40 >
41 <span gwc:condition="c/isEmpty">&#160;</span>
42 <span
43 gwc:condition="
44 !c/isEmpty & &
45 (c/item/type!='FormField') & &
46 (c/item/type!='StaticLabel') & &
47 (c/item/type!='StaticImage') & &
48 (c/item/type!='Button')
49 "
50 class="notsupported">{c/item/type} NOT SUPPORTED IN GRID</span>
51 <span
52 gwc:condition="
53 !c/isEmpty & &
54 ((c/item/type=='FormField') ||
55 (c/item/type=='StaticLabel') ||
56 (c/item/type=='StaticImage') ||
57 (c/item/type=='Button'))
58 "
59 gwc:replace="c/item"/>
60 </div>
61 </gwc:omit>
62 </gwc:omit>
63 </div>
64 </gwc:snippet-root>
65 </body>

```

Figure 60 : Snippet GridLayout.xhtml

Nous voyons déjà à la ligne 23 que si la grille est trop large, la mise en page n'est pas supportée. Lignes 24 à 28, une balise div globale est générée, elle a la classe `iPhoneGridLayout` et une hauteur correspondant au nombre d'éléments qu'elle englobe multiplié par 44 pixels, valeur conseillée par Apple pour la hauteur d'un élément dans une table [APPL].

Ensuite lignes 29 à 32 et lignes 33 à 36 consistent en des itérations imbriquées qui sont faites sur le nombre de lignes et le nombre de cellules dans chaque ligne. La macro `gwc:omit-tag="true"` indique au SBRE que les balises ne sont pas à afficher dans le document HTML final.

Dans les lignes 37 à 40, une balise div est créée pour chaque cellule, elle a la classe `iPhoneGridLayoutCell` et ligne 39 le positionnement de la cellule est calculé.

Le reste du snippet affiche les éléments supportés de l'application, et dans le cas où l'élément n'est pas supporté, affiche la page non supportée.

Les balises `<div> iPhoneGridLayout` et `iPhoneGridLayoutCell` ont des éléments de style propres présents dans le fichier CSS et montrés sur la figure 61.

Ces éléments n'ont aucune marge intérieure et extérieure, car nous voulons qu'ils prennent toute la place disponible sur l'écran. En effet, la taille de l'écran d'un smartphone est limitée et il ne faut pas perdre d'espace.

La balise globale `iPhoneGridLayout` a un positionnement relatif par rapport à son élément parent alors que la balise `iPhoneGridLayoutCell` a un positionnement absolu du fait que son positionnement est calculé directement dans le snippet (ligne 39 de la figure 60).

Enfin, chaque `iPhoneGridLayoutCell` a une hauteur de 44 pixels en accord avec les recommandations d'Apple [APPL].

```
349  /* ===== */
350  /* Grid */
351  /* ===== */
352
353  .iPhoneGridLayout {
354      /* border: 1px solid green; /* for debug */
355      display: block;
356      position: relative;
357      padding: 0;
358      margin: 0;
359  }
360
361  .iPhoneGridLayoutCell {
362      /* border: 1px solid red; /* for debug */
363      position: absolute;
364      display: block;
365      padding: 0;
366      margin: 0;
367      line-height: 44px;
368  }
```

Figure 61 : Feuille de style de la grille

VI.4.2.Unsupported

Comme nous l'avons vu dans la partie VI.3 de ce document, il a fallu réduire, dans le thème iPhone, la complexité et donc le nombre d'éléments et de combinaisons supportées par rapport au GDC ou au thème AJAX du GWC.

Nous avons pour cela ajouté un snippet : *Unsupported.xhtml* présenté sur la figure 62. Une balise particulière de la classe `notsupported` est créée dès qu'un élément non supporté est détecté par le SBRE (ligne 23). Le mot-clef `{type}` est une fonction du SBRE qui remplace le mot-clef par le type de l'élément non supporté.

```

1 <?xml version="1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <!--
4 FOURJS_START_COPYRIGHT(D,2010)
5 Property of Four Js*
6 (c) Copyright Four Js 2010, 2011. All Rights Reserved.
7 * Trademark of Four Js Development Tools Europe Ltd
8 in the United States and elsewhere
9
10 This file can be modified by licensees according to the
11 product manual.
12 FOURJS_END_COPYRIGHT
13 -->
14 <html xmlns:gwc="http://www.4js.com/GWC">
15 <!-- the head element is ignored -->
16 <head>
17 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
18 <title>Unsupported widget snippet</title>
19 </head>
20 <body>
21 <!-- the template snippet is the content of the gwc:snippet-root element -->
22 <gwc:snippet-root>
23 <div class="notsupported">ELEMENT {type} NOT SUPPORTED</div>
24 </gwc:snippet-root>
25 </body>
26 </html>
27

```

Figure 62 : Snippet Unsupported.xhtml

La classe `notsupported` a un style particulier présenté sur la figure 63. Un écran rouge est affiché, avec le texte de ce qui n'est pas supporté écrit en blanc. Ligne 23, l'attribut `z-index` est mis avec une valeur de 10 000. Cela nous assure que cette balise `notsupported` sera affichée par-dessus toutes les autres.

```

13 .notsupported {
14     color: white;
15     font-weight: bold;
16     background-color: red;
17     display: block;
18     position: fixed;
19     left: 0;
20     top: 0;
21     width: 100%;
22     height: 1000px;
23     z-index: 10000;
24 }

```

Figure 63 : Feuille de style pour les écrans non supportés

Sur la figure 64, nous pouvons voir un exemple d'écran non supporté.



Figure 64 : Capture d'un écran non supporté

VI.4.3. Géolocalisation

Le langage Genero 4GL permet d'appeler certaines fonctions exécutées par les moteurs de rendu avec une méthode appelée `frontCall`. Ainsi, il a par exemple été possible d'ajouter des fonctions exploitant les capacités de géolocalisation de l'iPhone.

Ces fonctions sont présentées sur la figure 65 et sont assez simples. Il faut d'abord appeler une fonction de départ `startGeolocation` qui va vérifier que la géolocalisation est supportée et active. Ensuite, il est possible d'obtenir dans un programme Genero 4GL la longitude et la latitude avec les fonctions `getLongitude` et `getLatitude` et ainsi il est possible de créer des applications mobiles exploitant la géolocalisation.

Et sur la figure 66 se trouve une capture d'écran de l'activation de la géolocalisation dans une application Genero sur iPhone. Le mécanisme utilisé permet d'afficher le message natif d'iOS.

```

804 var geolocationIsReady=false;
805 var geolocationCurrentLatitude=0;
806 var geolocationCurrentLongitude=0;
807
808 gwc.frontCallModuleList.iphone = {
809     startGeolocation: function() {
810         if(navigator.geolocation)
811         {
812             navigator.geolocation.getCurrentPosition(function(p) {
813                 geolocationIsReady=true;
814                 geolocationCurrentLatitude=p.coords.latitude;
815                 geolocationCurrentLongitude=p.coords.longitude;
816             });
817             return 0;
818         }
819         else
820         {
821             geolocationIsReady=false;
822             geolocationCurrentLatitude=-1;
823             geolocationCurrentLongitude=-1;
824             return -1;
825         }
826     },
827     getLatitude: function() {
828         if(geolocationIsReady) {
829             return geolocationCurrentLatitude;
830         } else {
831             return "unavailable";
832         }
833     },
834     getLongitude: function() {
835         if(geolocationIsReady) {
836             return geolocationCurrentLongitude;
837         } else {
838             return "unavailable";
839         }
840     }
841 }
842 }
843 }

```

Figure 65 : Java Script pour le support de la géolocalisation



Figure 66 : Capture d'écran de l'activation de la géolocalisation

VII.Risques

VII.1.Évaluation des risques

Différentes méthodes existent pour gérer les risques : il est tout d'abord possible d'utiliser le diagramme d'Ishikawa et l'arbre des causes de manière prospective, ainsi que le réseau PERT. Je vais plutôt utiliser un graphique, inspiré de celui de Mintzberg qui définit la priorité des tâches en fonction de leur urgence et de leur importance. Cela donne une cartographie des risques en fonction de leur criticité et de leur probabilité.

La première étape consiste à lister tous les risques identifiés inhérents au projet. Ensuite, ces risques seront classés dans un tableau en fonction de leur gravité et probabilité d'apparition afin de prioriser les actions en matière de gestion des risques. Puis un plan d'action sera défini en fonction de ces résultats : des actions préventives de façon à limiter l'apparition d'aléas, mais également des actions curatives pour le cas où le risque deviendrait réalité.

VII.1.1.Liste des risques

J'ai classé les risques par catégories que je vais détailler.

VII.1.1.1.Risques stratégiques

Le thème iPhone est stratégique pour Four J's, car il positionne la société sur le marché en pleine expansion des téléphones intelligents. C'est le premier pas de la société vers l'informatique mobile moderne. De plus, l'image de la société auprès des clients et des utilisateurs peut être affectée par l'utilisation du thème iPhone.

Il y a un risque stratégique assez important, car l'image de la société peut directement être touchée par ce projet.

VII.1.1.2.Risques financiers

Le risque financier pour l'entreprise est moindre que le risque en terme d'image. Un retard ou un produit initial non fidèle aux attentes aurait des répercussions plus grandes au niveau de l'image de la société qu'au niveau financier.

Le risque financier est donc faible.

VII.1.1.3.Risques lors du développement

VII.1.1.3.1.Risques technologiques

Bien que l'essor des téléphones intelligents soit récent, les technologies utilisées pour développer le thème iPhone sont des technologies éprouvées et standardisées. Elles sont à la fois utilisées en interne depuis de nombreuses années, et sont très utilisées en externe et donc bénéficient à la fois d'une bonne documentation et de la disponibilité de nombreuses ressources sur l'Internet.

Le risque technologique est faible et une chance minimale de se produire.

VII.1.1.3.2.Risques de dépassement de délais

La première version livrable devait être disponible en juin 2010, c'est pourquoi il était primordial que le développement, la phase de test ainsi que la documentation fussent prêts à cette date.

Du fait de la justesse des délais, il y a ici un risque fort.

VII.1.1.3.3.Risques de dépassement des coûts

La société a assez de ressources humaines pour pallier un éventuel retard de développement, et la société possède toute l'infrastructure matérielle et logicielle pour l'activité de développement logiciel.

Le risque financier d'une explosion des coûts de développement est donc minimal.

VII.1.1.3.4.Risques au niveau des ressources

Le développement sera confié à un développeur débutant dans les technologies requises pour mener à bien ce projet, et ne maîtrisant pas le fonctionnement du GWC et du GAS.

Le risque est ici réel et devra être pris en compte.

Concernant les autres ressources, celles-ci ne présentent pas de risque particulier, car leurs tâches dans ce projet seront situées dans leurs domaines de compétences et dont les processus sont éprouvés en interne.

VII.1.1.3.5. Risques managériaux

L'équipe pour ce projet est une équipe transversale. Il peut donc y avoir un risque dans le cas où la structure du projet ne serait pas clairement définie ou dans le cas où les objectifs du projet ne seraient pas connus de tous.

De surcroît, j'ai déjà par le passé managé des projets transversaux et je bénéficie donc d'une petite expérience dans ce domaine.

Néanmoins, le risque est important, mais sa probabilité est très faible.

VII.1.1.3.6. Risques organisationnels

L'organisation autour de ce projet est une organisation classique au sein de l'entreprise, les différentes équipes concernées ayant l'habitude de travailler ensemble.

Le risque ici est minime.

VII.1.1.3.7. Risques au niveau de la qualité

Il est important que la qualité du thème pour iPhone soit bonne, mais la société dispose d'une équipe QA spécialement dédiée à la qualification et au contrôle des produits.

VII.2. Classification des risques

La figure 67 présente la classification des risques faite selon leur probabilité d'apparition et selon leur gravité. Cela permet de voir quels sont les risques à traiter (sur fond rouge), ceux à suivre (sur fond orange) et ceux qui sont négligeables (sur fond jaune).

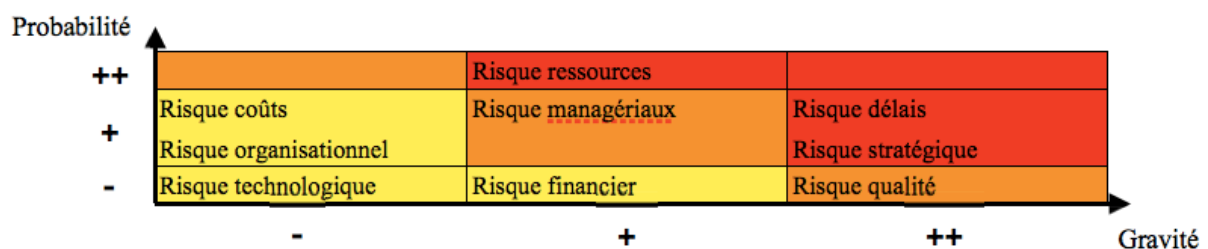


Figure 67 : Classification des risques

VII.3. Plan d'action

Je vais présenter les différentes actions mises en œuvre pour minimiser les risques autour de ce projet. Seuls les risques à traiter et ceux à suivre font l'objet d'une action préventive.

VII.3.1.1.Risques à traiter

VII.3.1.1.1.Risque au niveau des ressources

Le développeur principal étant débutant dans les domaines relatifs à ce projet, il a été décidé de l'encadrer par deux développeurs confirmés.

Le premier sera là pour une formation autour du fonctionnement du GAS et du fonctionnement du thème AJAX dont le thème iPhone est très proche. Il est de plus disponible pour répondre à toutes les questions du développeur principal.

Le second interviendra pour aider le développeur principal lors du développement, pour implémenter les parties les plus complexes du projet.

VII.3.1.1.2.Risques stratégiques

Les clients de Four J's ont exprimé, en passant par nos centres de support, des besoins en matière de développement mobiles en particulier autour de l'iPhone.

De ce fait, une grosse partie du projet concerne l'étude de l'État de l'Art du développement d'applications mobiles et plus particulièrement d'applications mobiles iPhone, aussi bien au niveau purement graphique qu'au niveau interfaces utilisateurs.

Le thème iPhone se rapprochera autant que possible d'une application native iPhone afin de répondre au mieux aux besoins des clients et ainsi renforcer l'image de la société en tant que porteuse d'innovation.

VII.3.1.1.3.Risques de dépassement des délais

Le délai avant la sortie du thème iPhone (pour juin 2010) est court et il était tout simplement impossible de dépasser les délais.

Il n'était, de plus, pas question de réduire la qualité pour respecter la date butoir.

Pour éliminer le risque de dépassement des délais, j'ai décidé de faire un point régulier de l'avancement du projet avec le directeur de projet en veillant à ce qu'aucun retard ne soit pris. Le cas échéant, il a été convenu de rendre le développeur senior disponible à 100 % pour accompagner le développeur principal du projet.

VII.3.1.2.Risques à suivre

VII.3.1.2.1.Risques managériaux

Le projet, ses acteurs ainsi que ses enjeux ont été clairement présentés à l'ensemble de l'équipe GCS. Le projet a de plus l'entier soutien de la société.

En cas de problèmes au niveau managérial, le directeur de projet pourra intervenir pour un recadrage.

VII.3.1.2.2.Risques au niveau de la qualité

Pour éviter tout risque au niveau de la qualité du thème iPhone, il a été prévu de nombreuses sessions de tests manuels de la part de l'équipe qualité située en Inde.

Les tests seront menés parallèlement au développement de manière à ce que les problèmes remontés puissent être traités rapidement. Ainsi, à la fin du développement, le thème iPhone ne devrait pas avoir de bogue grave.

VIII.Portabilité

Bien que le thème iPhone ait été initialement développé pour le téléphone intelligent d'Apple, le fait que Safari mobile soit basé sur le moteur de rendu de WebKit assure une portabilité certaine car de nombreux systèmes d'exploitations mobiles ont construit leur navigateur autour de WebKit.

En effet, le navigateur du système d'exploitation mobile Android est lui aussi basé sur WebKit et ainsi le thème iPhone est portable au mieux directement sur cette plateforme mobile, et au pire avec quelques ajustements mineurs. Il a d'ailleurs été réalisé quelques tests qui ont été très concluants.

Nokia aussi a basé son navigateur mobile pour Symbian S60 sur WebKit, ainsi que Blackberry depuis la version 6 de son système d'exploitation. Ces plateformes mobiles devraient elles aussi être supportées par le thème iPhone. Aucun test n'a par contre encore été réalisé.

Un dernier système d'exploitation mobile, moins connu et présent que les précédents, mais très prometteur, et nommé WebOS utilise lui aussi WebKit. WebOS a été développé par Palm qui a été racheté par HP en 2010.

En se basant sur les chiffres des parts de marché des téléphones intelligents du quatrième trimestre 2010 de Gartner⁴⁰, ce sont 92 % des systèmes d'exploitation mobiles qui sont potentiellement supportés par le thème iPhone.

Le principal absent est Windows Mobile, et il le restera, car Windows Phone 7, son successeur, n'utilise pas WebKit comme moteur pour son navigateur, mais se base sur une version allégée d'Internet Explorer 7.

⁴⁰ http://en.wikipedia.org/wiki/Mobile_operating_system

IX. Devenir du projet

Ces dernières années ont marqué l'émergence d'un nouveau type d'appareil mobile, les tablettes tactiles avec en particulier l'iPad d'Apple, des tablettes sous Android comme la Galaxy Tab de Samsung et la Xoom de Motorola ainsi qu'une tablette de BlackBerry, le PlayBook.

Le support de ces appareils, bénéficiant d'un écran plus grand que celui des smartphones et permettant ainsi de faire disparaître de nombreuses limitations liées à la taille de l'écran tout en gardant l'approche tactile des interactions homme-machines, est prévu.

À l'heure actuelle, un nouveau thème est développé dont le but est, entre autres, de permettre d'exploiter au mieux ces appareils. Ce thème exploite à la fois le thème AJAX, le thème iPhone et les possibilités offertes par HTML5.

Conclusion

Par rapport au projet en lui-même, je retiens plusieurs choses, notamment la complexité de l'environnement et de l'écosystème Genero et l'impression que le projet s'est déroulé en deux temps.

Ce qui m'a le plus marqué lors de ce projet est la complexité de l'environnement et les dépendances fortes présentes entre les différentes briques qui composent l'écosystème Genero. Bien que travaillant à Four J's depuis 2003, je n'avais été amené à travailler que sur des projets périphériques qui ne m'avaient pas permis de prendre conscience de cette complexité.

En effet, il y a les spécificités inhérentes au langage Genero 4GL et les multiples possibilités qu'il offre aux utilisateurs d'une part, et le souhait de ces mêmes utilisateurs qui s'attendent à ce qu'une application Genero fonctionne de la même façon sur un écran d'ordinateur que sur un smartphone d'autre part.

Une troisième couche de complexité a été rajoutée à ce projet, celle des contraintes nouvelles liées aux smartphones, qui avec leur taille d'écran réduite et l'absence de clavier physique nous ont forcés à inventer une nouvelle façon d'utiliser des applications Genero.

Du fait de cette complexité, une majeure partie de la charge de travail a été l'analyse de l'existant et la recherche des solutions possibles. De nombreux essais ont été réalisés, afin d'étudier la viabilité de chacune, la plupart se soldant par des échecs. Mais chacun d'eux a été important, car ils ont permis en définitive de faire les choix nécessaires afin de converger vers la solution choisie.

D'un point de vue macroscopique, on peut considérer que ce projet s'est déroulé en deux temps. Le premier fut celui de l'analyse et de la recherche de solutions pendant lequel de nombreuses pistes ont été explorées. Je n'avais finalement pas l'impression d'avancer réellement dans la réalisation du projet. Le second fut celui de la réalisation proprement dite qui a été fortement simplifiée par tout le travail réalisé en amont.

D'un point de vue plus personnel, ma plus grande frustration a été de devoir couper le projet initial en deux projets indépendants. Bien que c'eût été irréalisable dans les délais impartis, j'aurais souhaité réaliser l'intégralité du projet, et ce d'autant plus que c'est avant tout l'envie de développer une application iPhone qui m'a incité à proposer ce projet à mes supérieurs.

Je retiens également plusieurs autres points de ce projet : la nécessité de très bien connaître tous les composants d'un projet, la gestion du stress pour finir un projet dans les délais, la confrontation à une démarche projet sur un projet de développement, et enfin, la nécessité de remise en cause constante et l'importance de la prise de recul.

Mener ce projet a nécessité l'apprentissage d'une somme conséquente de choses. Je me suis vite rendu compte que je ne connaissais qu'une partie minime du langage Genero 4GL et que cette connaissance était indispensable pour la conduite de ce projet, pour faire les bons choix et trouver les solutions adéquates aux problèmes qui se sont présentés.

J'ai eu la chance d'avoir des personnes disponibles pour m'expliquer les points délicats et pour me guider lors de mes recherches. Sans cela, ce manque en terme de connaissance aurait été fatal au projet.

Vu la part importante qu'a prise l'analyse et la recherche de solution dans le déroulement du projet, la phase de réalisation a été particulièrement brève. La date butoir étant connue et non modifiable, les dernières semaines ont été particulièrement stressantes. J'ai dû apprendre à gérer ce stress et à le transformer en énergie positive afin d'achever la réalisation dans les temps. Ce qui m'est apparu essentiel dans cette phase, c'est de prendre les tâches les unes après les autres, et de faire des points d'avancement réguliers. Encore une fois, le temps passé sur la préparation de la phase de réalisation s'est réellement avéré précieux, car il a permis de donner dans le détail les étapes de la phase de développement.

Un point très intéressant de ce projet a été, pour moi, la confrontation à une démarche projet complète. De la phase de décision, où ce projet a été amorcé, jusqu'à la livraison avec l'ajout du thème iPhone dans le GWC qui a été livré au client. Cela m'a permis de réaliser l'importance de toutes les phases. En effet, l'analyse, la recherche de solution, la documentation et les tests sont tout aussi importants que la réalisation à proprement parler. Cela m'a donné la chance de confronter les connaissances théoriques, apprises à l'IUT et lors du cursus du CNAM à la pratique en entreprise.

Enfin, j'ai constaté que deux éléments déterminants dans la conduite d'un projet sont la remise en cause des solutions trouvées et la prise de hauteur par rapport au projet. La remise en cause garantit que chaque piste est correctement analysée, et évite par exemple de se rendre compte tardivement d'un problème éventuellement bloquant. La prise de hauteur permet d'analyser de manière plus objective la situation, et force à mettre de côté ses a priori et ses partis pris.

Le moment le plus difficile de ce projet a été pour moi vers le début du projet. Je m'étais plongé dans le projet avec trop de naïveté et je n'avais pas correctement analysé le problème. Je m'étais donc attaqué à la recherche de solution trop tôt. De ce fait, il est arrivé un moment où j'ai eu l'impression que je n'y arriverais jamais. À ce moment-là, j'ai été aidé par Olivier Imbert, le directeur de projet GCS et mon tuteur dans l'entreprise, qui m'a aidé à prendre du recul par rapport à ce que j'avais fait, et qui m'a fortement aiguillé sur les éléments à analyser avant de recommencer à chercher des solutions.

Pour finir, j'aimerais dire que j'ai vraiment beaucoup appris, ma manière d'appréhender un projet de développement a changé à tout jamais. Je tire également une certaine fierté à la conclusion favorable de ce projet, savoir que quelque chose que l'on a réalisé est réellement utilisé par des clients est particulièrement valorisant.

Bibliographie

Livres

CLARK, J. (2010). *Tapworthy: Designing Great iPhone Apps*. publication place: O'Reilly Media. 310p.

COLBORNE, G. (2011). *Simple and Usable: Web, Mobile, and Interaction Design*. Berkeley, CA: New Riders Press. 196p.

GINSBURG, S. (2010). *Designing the iPhone User Experience: A User-Centered Approach to Sketching and Prototyping iPhone Apps*. Addison-Wesley Professional. 294p.

[MIND] JOHNSON, J. (2010). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Amsterdam: Morgan Kaufmann. 186p.

KOCHAN, SG. (2009). *Programming in Objective-C 2.0*. 2nd ed. Indianapolis, Ind.: Addison-Wesley Professional. 600p.

STARK, J. (2010). *Building iPhone Apps with HTML, CSS, and JavaScript*. Sebastopol, CA: O'Reilly Media. 192p.

WAGNER, R. (2008). *Professional iPhone and iPod Touch Programming: Building Applications For Mobile Safari*. Indianapolis, Ind.: Wrox. 284p.

ZDZIARSKI, J. (2009). *iPhone SDK Application Development*. Sebastopol, CA: O'Reilly Media. 366p.

Publications

[APPL] APPLE (2009). *iPhone Human Interface Guidelines for Web Application : User Experience*. 52p.

APPLE (2010). *Safari Web Content Guide : User Experience*. 104p.

SCHUMACHER, J. (2009). *Advanced iPhone Web Development*. <http://blog.joshschumacher.com/wp-content/uploads/2009/05/advanced-iphone-web-development.pdf>. 27p.

Liste de figures

Figure 1 : Les couches logiques de l'écosystème Genero

Figure 2 : Exemple simple de programme 4GL

Figure 3 : Répartition du CA par continent pour l'année 2008

Figure 4 : Répartition des revenus par secteur d'activité pour l'année 2008

Figure 5 : Canaux de distribution des produits Four J's

Figure 6 : Sources de revenus (2008)

Figure 7 : Implantation de la société à travers le monde

Figure 8 : Ressources humaines (2008)

Figure 9 : Les cycles informatiques de 1960 à nos jours

Figure 10 : Les rythmes d'adoption comparés de différentes technologies de l'Internet de bureau et mobile

Figure 11 : Évolution des ventes de smartphones au niveau mondial (sources Gartner)

Figure 12 : Évolution des ventes de smartphones à écran tactile au niveau mondial (source Canalys)

Figure 13 : Ventes mondiales de smartphones par système d'exploitation (sources Gartner)

Figure 14 : Parts de marché des navigateurs mobile — Q4 2008 à Q4 2009 (sources StatCounter)

Figure 15 : Parts de marché des navigateurs mobile — 2009 (NetApplications)

Figure 16 : Attractivité pour les développeurs (source Flurry)

Figure 17 : Attractivité pour les clients (source Flurry)

Figure 18 : Couches d'abstraction du système d'exploitation pour mobile iOS

Figure 19 : Évolution du service App Store

Figure 20 : Capture de l'application Chronomètre d'Apple

Figure 21 : Capture de l'application Calculatrice d'Apple

Figure 22 : Exemple de code affichant des données

Figure 23 : Arbre AUI présenté de façon visuelle

Figure 24 : L'arbre AUI permet d'afficher des applications natives sur chaque OS

Figure 25 : L'architecture du GAS

Figure 26 : Le fonctionnement du GAS

Figure 27 : Exemple d'applications affichées avec le GDC

Figure 28 : Applications Genero rendues de façon native

Figure 29 : Le fonctionnement du SBRE

Figure 30 : Comment une application est rendue par le GWC

Figure 31 : Exemple de style CSS pour le menu avec le thème AJAX

Figure 32 : Une icône simple avant transformation en Web Clip

Figure 33 : Une icône affiche comme un Web Clip sur l'écran d'accueil

Figure 34 : Mesures en pixels en mode portrait

Figure 35 : Mesures en pixels en mode paysage

Figure 36 : Différence d'affichage d'une page web en spécifiant le `viewport` pour la largeur

Figure 37 : Différence d'affichage d'une application web en spécifiant le `viewport` pour la largeur

Figure 38 : Mise en page d'une table en liste simple

Figure 39 : Mise en page d'une table en liste groupée

Figure 40 : Mesures de l'écran lorsque l'écran est affiché en mode portrait

Figure 41 : Affichage par Safari mobile d'un menu déroulant

Figure 42 : Contrôles affichés de façon native par Safari mobile

Figure 43 : Code HTML pour la mise en place du cadriciel web iUi

Figure 44 : Exemple de code HTML d'utilisation du cadriciel web iUi

Figure 45 : Exemple d'application développée avec le cadriciel web iUi

Figure 46 : Exemple de mise en page d'application (.per)

Figure 47 : Code représentant une application simple

Figure 48 : Représentation graphique d'une application sur une grille

Figure 49 : Calcul des positions des éléments sur la grille

Figure 50 : Exemple de fichier .per

Figure 51 : Rendu de l'exemple sous Mac OS X avec le GDC

Figure 52 : Rendu de l'exemple dans un navigateur avec le thème AJAX

Figure 53 : Mises en page supportées par le thème iPhone

Figure 54 : Exemple d'une table avec beaucoup de colonnes

Figure 55 : Affichage sur l'iPhone de données d'un tableau en ligne

Figure 56 : Affichage sur l'iPhone de données d'un tableau en groupe

Figure 57 : Rendu de l'exemple sur l'iPhone avec le thème iPhone

Figure 58 : Snippet Form.xhtml

Figure 59 : Snippet Grid.xhtml

Figure 60 : Snippet GridLayout.xhtml

Figure 61 : Feuille de style de la grille

Figure 62 : Snippet Unsupported.xhtml

Figure 63 : Feuille de style pour les écrans non supportés

Figure 64 : Capture d'un écran non supporté

Figure 65 : Java Script pour le support de la géolocalisation

Figure 66 : Capture d'écran de l'activation de la géolocalisation

Figure 67 : Classification des risques

Liste des tableaux

Tableau I : Les gestes multipoints de base

Tableau II : Les programmes développeurs Apple

Tableau III : Comparatif entre applications natives et applications web

Tableau IV : Correspondance entre balise HTML et contrôles natifs

Conception et réalisation d'un thème iPhone pour le Genero Web Client

Mémoire d'Ingénieur C.N.A.M., Strasbourg 2011

RESUME

L'essor de l'Internet mobile au début du XXI^e siècle, puis son envol en 2007 à l'arrivée sur le marché de l'iPhone d'Apple ont motivé la société Four J's, qui commercialise un environnement de développement et de déploiement d'applications professionnelles, à inclure dans sa suite logicielle une solution adaptée à cet usage et en particulier à ce smartphone.

La conception et le développement d'un thème iPhone pour le Genero Web Client ont nécessité d'appréhender adéquatement toutes les problématiques des applications mobiles. Aussi bien les nouvelles contraintes liées à la taille des écrans et à l'usage nomade de ces appareils que les possibilités innovantes offertes par une interface tactile multipoints.

En effet, comment permettre au mieux aux utilisateurs de Genero la création d'applications qui fonctionnent aussi bien sur un ordinateur que sur un smartphone, et de telle sorte que l'expérience utilisateur corresponde à l'appareil utilisé ?

Mots clés : Genero, iPhone, Internet, GWC, ergonomie, smartphone, Safari mobile, développement web.

SUMMARY

The boom of mobile internet since Apple first entered the market in 2007 has motivated Four J's, a vendor who designs and sells a development and deployment solution for business applications, to augment its software suite with a solution aimed at the new mobile phone medium, with a special focus on Apple's iPhone.

The design and development of a theme for the iPhone Genero Web Client has required to consider issues specific to mobile applications, most notably the smallness of the screen, as well as the new possibilities offered by the multi-touch user interface.

How can we best enable Genero users to create applications that work equally well on a desktop as on a smartphone, in such a way that the user experience is pleasant and natural on every device ?

Key words : Genero, iPhone, Internet, GWC, ergonomics, smartphone, Safari mobile, web development.