Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de Bachelor 2012

Filière Informatique de gestion

Application mobile multiplateforme pour l'événement viticole *Vinéa*

Etudiant : Léonard Stalder

Professeur : Yann Bocchi

Préface

Aujourd'hui, le développement mobile est en plein essor. De nombreuses nouvelles tendances se profilent et il n'est pas toujours facile de choisir la technologie la plus appropriée. La conception d'applications mobiles est un domaine qui m'intéresse beaucoup et qui n'a été que peu étudié lors de mon cursus académique. Ce travail me donnait l'occasion d'approfondir mes connaissances dans ce domaine et de réaliser une application qui allait être mise en production.

Dans mon rapport, l'état de l'art des nouvelles tendances en matière de développement mobile, répertorie de manière exhaustive les dernières nouveautés en la matière.

Grâce à ces technologies, j'ai pu réaliser une application mobile multiplateforme pour l'événement viticole *Vinéa*. Je vous invite à la découvrir à travers ce rapport. Liste des vins, carte interactive, fiches de dégustation ou parcours de découverte tout y est afin de permettre aux visiteurs de *Vinéa* de profiter pleinement de la manifestation.

Si vous prévoyez de vous rendre à *Vinéa* du 31 août au 1er septembre, téléchargez l'application mobile sur l'AppStore ou sur Google Play.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé pour ce travail de Bachelor et particulièrement :

M. Yann Bocchi, professeur responsable, qui m'a soutenu et conseillé tout au long de ce travail.

M. Pierre Kenzelmann, responsable de projet chez Icare, qui m'a suivi au sein de l'institut Icare.

M. François Chabbey, ingénieur en développement chez Icare, qui m'a aidé pour la partie d'intégration sur iOS ainsi que pour la mise en production de notre application.

M. David Werlen, ingénieur en développement chez Icare, qui m'a conseillé dans mes choix technologiques.

Je remercie également tout le personnel de l'institut Icare pour les conseils qu'ils m'ont donnés.

Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué ce travail de Bachelor seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

M. Yann Bocchi

M. Pierre Kenzelmann

Léonard Stalder

Glossaire

Android

Il s'agit d'un système d'exploitation open source pour Smartphones développé par Google.

App Store

Plateforme de téléchargement d'applications du système d'exploitation mobile iOS.

Application mobile multiplateforme

Application développée à l'aide d'un seul langage pour ensuite être déployée sur plusieurs systèmes d'exploitation différents.

CSS3

Il s'agit de la dernière révision de CSS. Le CSS (Cascading Style Sheets : feuilles de style en cascade) est utilisé afin de mettre en forme des pages HTML.

Framework

Il s'agit d'un ensemble de bibliothèques qui servent à créer les grandes lignes d'une partie d'un logiciel.

Google Play

Google Play est une plateforme de téléchargement d'applications du système d'exploitation mobile Android.

HTML5

Il s'agit de la dernière révision d'HTML. HTML est un langage de balises utilisé pour représenter des pages web.

iOS

iOS est un système d'exploitation mobile développé par Apple pour ses appareils mobiles.

JavaScript

Le JavaScript est un langage de programmation de scripts principalement utilisé dans des projets web.

Jquery

Il s'agit d'un Framework JavaScript qui permet de simplifier la manipulation de pages HTML.

JqMobi

Il s'agit d'un Framework JavaScript qui permet de simplifier la manipulation de pages HTML. Celui-ci est optimisé pour les moteurs de rendu Webkit.

Librairie de manipulation du DOM

Ensemble de composants JavaScript qui permettent de manipuler le contenu de pages HTML

Moteur de rendu

Il s'agit d'un composant essentiel d'un navigateur internet. Celui-ci est en charge de l'affichage des éléments d'une page web. Plusieurs moteurs de rendu existent et les plus connus sont Gecko, Presto et Webkit.

Plateforme de téléchargement d'applications

Il s'agit d'une boutique en ligne sur laquelle peuvent être téléchargés des applications mobiles. Chaque système d'exploitation a sa propre plateforme de téléchargement.

Projection cartographique

Il s'agit d'un ensemble de techniques utilisées pour représenter la surface de la terre sur une carte.

Webservice

Un webservice est une application web qui permet l'échange de données avec d'autres applications via internet.

Résumé

La partie d'**introduction** situe le contexte dans lequel le projet a été réalisé. Un descriptif du projet ainsi que les objectifs à réaliser y sont également présentés.

La partie d'**analyse de l'existant est du marché** regroupe une enquête de terrain auprès de potentiels utilisateurs de l'application *Vinéa* ainsi qu'une analyse des applications similaires.

La partie de **proposition d'application** décrit les fonctionnalités que l'institut Icare et moi-même avons décidé de développer.

La partie de l'**état de l'art** répertorie de manière exhaustive les dernières nouveautés en matière de développement mobile ainsi que les choix que nous avons effectués pour le développement de l'application mobile.

La partie de **description de l'application** recense toutes les fonctionnalités qui ont été développées. Ce chapitre traite également du fonctionnement de notre application, des problèmes rencontrés lors du développement et de la mise en production de l'application.

La partie de **gestion de projet** présente les différentes méthodes et outils qui ont été utilisés pour gérer le suivi du projet.

La partir de **conclusion** donne un bilan personnel sur le travail effectué.

Table des matières

1. Introduction

1.1. Contexte du projet

Le travail de Bachelor, qui finalise trois années passées à la HES-SO Valais, a pour but de mettre en pratique les connaissances acquises et de les appliquer à un projet informatique réel. Parmi un vaste choix de projets proposés par différents mandataires, j'ai choisi d'effectuer un travail sur le développement mobile pour le compte de l'institut Icare.

A la pointe des sciences de l'information, cet institut de recherche élabore des solutions informatiques qui répondent aux besoins du marché à partir de technologies testées, stables et efficientes. Mon travail, réalisé au sein de cet institut, se focalise sur la création d'une application mobile multiplateforme[1] à l'aide de nouvelles technologies.

1.2. Description du projet

Le but de ce travail est de créer une application mobile multiplateforme pour l'événement viticole *Vinéa*. Cet outil performant permettra aux visiteurs de profiter pleinement de la manifestation. Afin de réaliser le projet et de définir les fonctionnalités à créer, une analyse de marché ainsi qu'une analyse de l'existant ont été effectuées. Dans un deuxième temps, j'ai recherché et testé des technologies mobiles existantes et sélectionné les plus adaptées au projet mobile à réaliser. Sur la base des résultats obtenus j'ai réalisé l'application mobile pour les systèmes d'exploitation Android[2] et iOS[3]. Ce document finalise le travail effectué et rend compte de la démarche utilisée.

1.3. Acteurs

Léonard Stalder	Etudiant
Yann Bocchi	Professeur responsable
Pierre Kenzelmann	Responsable du projet chez Icare

1.4. Etapes à réaliser

Le cahier des charges[4] a été approuvé par les parties prenantes.

[1] Glossaire : Application mobile multiplateforme
[2] Glossaire : Android
[3] Glossaire : iOS
[4] Annexe 1 : Cahier des charges

1.5. Remarques

Le test des technologies mobiles actuelles et la rédaction du travail de recherche ont nécessité un grand investissement en temps. Jean-Baptiste Lathion, étudiant de troisième année à la HES-SO Valais, devait également rédiger pour l'institut Icare un état de l'art sur le développement mobile multiplateforme. Au terme de notre phase de recherche, l'institut Icare nous a conseillé de partager nos résultats afin d'opter pour les choix les plus performants. En accord avec les responsables de nos projets, Jean-Baptiste Lathion et moi-même avons réalisé notre application avec les mêmes technologies.

La seconde partie, à savoir le développement en lui-même, était un travail conséquent. Vu que mon application répondait à une demande du marché, elle devait absolument être stable et disponible à temps sur les différentes plateformes de téléchargement d'applications[5]. La mise en production de notre application a été réalisée.

[5] Glossaire : Plateforme de téléchargement d'applications

2. Analyse de l'existant et du marché

En 2009 Icare a développé l'application *Vinéa* pour le système iOS uniquement. Afin de développer une application performante et adaptée aux besoins du marché une analyse des forces et des faiblesses de cette application ainsi qu'une enquête de terrain ont été réalisées.

2.1. Analyse de l'application existante

Actuellement, l'application mobile *Vinéa*, développée en 2009, est exclusivement disponible sur iOS et offre les fonctionnalités suivantes :

- une carte de l'événement qui permet de localiser l'utilisateur parmi les stands
- une liste des encaveurs présents avec des informations les concernant
- une liste des vins avec fiches explicatives
- un système d'évaluation permettant d'enregistrer les vins favoris des utilisateurs et de les consulter sur une plateforme web

Dans la version actuelle, aucun aspect social (Intégration de Facebook) ni d'interactivité entre les utilisateurs et *Vinéa* n'ont été pris en compte.

Il en résulte qu'un utilisateur ne peut:

- pas télécharger l'application sur un autre système qu'iOS
- que consulter ses évaluations via la plateforme web ou l'application elle-même (aucun autre partage n'est possible)
- pas géo-localiser ses connaissances sur la carte
- que se géo-localiser sur un schéma des lieux et non sur une carte précise
- pas avoir accès à des actions et offres exclusives
- pas choisir des « parcours » de dégustation (ex : je veux déguster tous les Cornalin)

2.2. Analyse du marché

2.2.1. Objectifs de l'analyse

Afin de définir les fonctionnalités à développer dans la nouvelle application deux études ont été réalisées. La première repose sur une enquête effectuée auprès de visiteurs de *Vinéa* et la seconde sur une étude des fonctionnalités proposées par des applications de manifestations similaires.

2.2.2. Enquête de terrain

L'enquête de terrain auprès de différents visiteurs de *Vinéa* a permis de réaliser que les fonctionnalités de l'application existante n'étaient plus toutes d'actualité. L'enquête s'est déroulée en deux phases :

1) Conversation ouverte avec des personnes spécialisées dans les domaines du vin et de l'informatique afin de trouver des idées de fonctionnalités qui inciteraient les visiteurs à télécharger l'application.
2) Sondage auprès de onze visiteurs de *Vinéa* (tout âge et statut social confondus) afin d'évaluer le succès de chaque proposition de fonctionnalité.

Sur les onze personnes sondées, toutes ont prévu d'aller à *Vinéa* cette année. Seules deux personnes avaient déjà utilisé l'ancienne application iOS.

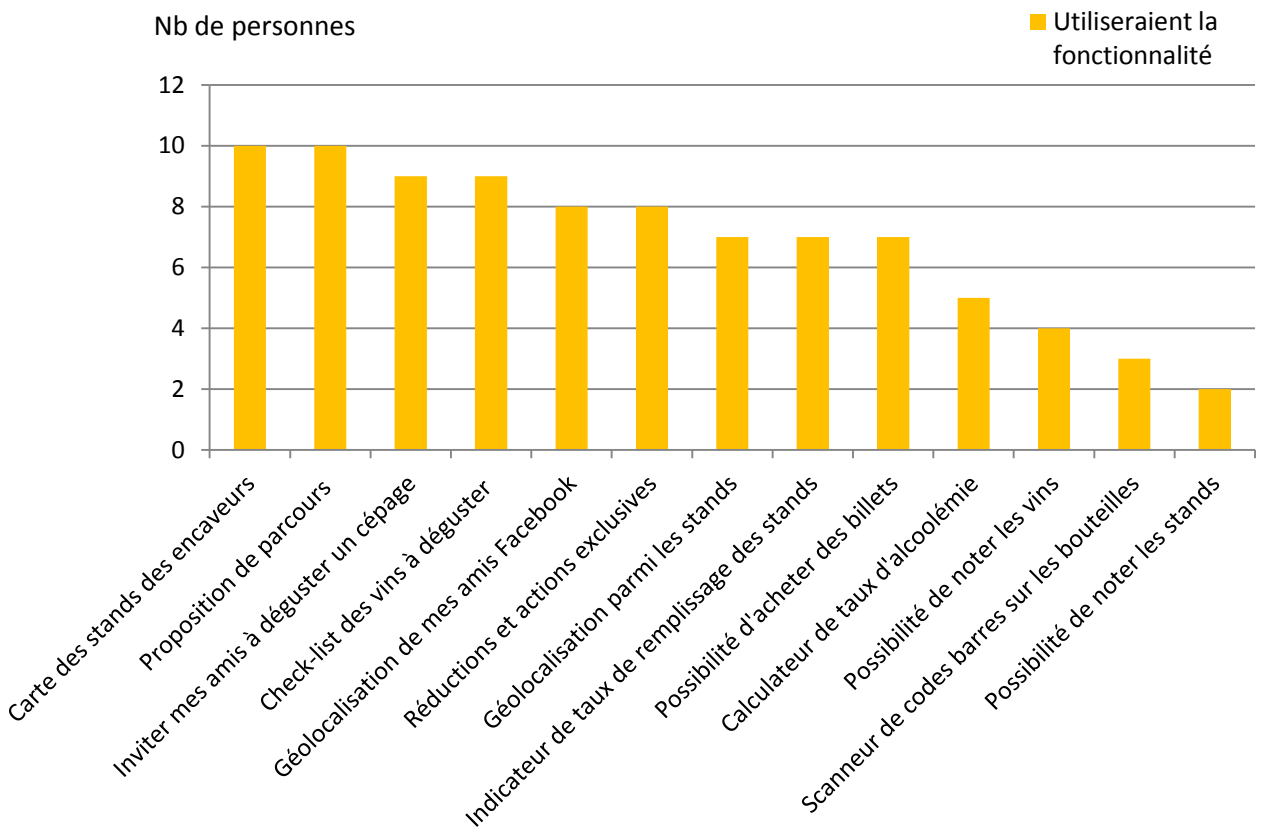


Tableau 1 - Résultats de l'enquête de terrain auprès des visiteurs de Vinéa

Cette enquête révèle que certaines fonctionnalités très demandées comme la proposition de parcours ou l'invitation à déguster un cépage avec ses amis n'existaient pas sur l'application actuelle. D'autres, telle la consultation des « coups de cœur » sur une plateforme web, y figurent mais n'intéressaient plus les utilisateurs.

2.2.3. Analyse de l'existant

Cette analyse a pour but de lister les principales applications similaires à celle de *Vinéa*. Cette sélection a aidé à élaborer des fonctionnalités qui soient adaptées à la réalité du marché.

Les critères de sélection étaient d'avoir au moins deux fonctionnalités similaires à l'application *Vinéa*. De plus il devait s'agir d'une application liée à une manifestation ou au domaine du vin. Nous avons sélectionné les quatre applications les plus populaires qui répondaient à ces critères. Elles ont été classées en deux catégories :

Foires, festivals et manifestions avec application Buzz

Application T'es où ? (Foire du Valais)

Fonctionnalités similaires : Plan de situation, Géolocalisation personnelle et des amis Facebook.

Forces : Convivial et simple à utiliser.

Faiblesses : Le temps de chargement et de synchronisation est élevé (les données sont à chaque fois téléchargées depuis un serveur distant). Lors d'une recherche sur le plan, aucun indicateur graphique ne nous permet de situer le stand recherché, ni d'afficher le détail du stand par un clic sur l'endroit indiqué sur la carte.

Paléo Buddy (Paléo Festival Nyon)

Fonctionnalités similaires : Plan de situation, possibilité de se donner rendez-vous en positionnant des points sur le plan de situation et en invitant des amis.

Forces : Beaucoup d'informations en direct.

Faiblesses : peu intuitif et la géolocalisation de l'utilisateur sur le plan de situation n'est pas possible.

Manifestations viticoles

My Vinexpo (Vinexpo)

Fonctionnalités similaires : Plan de situation, liste des exposants (avec possibilité de les géo localiser sur la carte), liste des vins, possibilité d'ajouter des vins à sa liste de favoris.

Forces : Très complet au niveau des informations à disposition, liste des encaveurs et produits très complète avec beaucoup de détails à disposition.

Faiblesses : application peu intuitive, beaucoup d'informations souvent mal structurées.

LIWF2012 (London International Wine Fair 2012)

Fonctionnalités similaires : Plan de situation, liste des exposants (avec possibilité de les géo localiser sur la carte).

Forces : Navigation sur la carte très conviviale, possibilité de localiser les stands.

Faiblesses : application peu intuitive, navigation difficile.

2.2.4. Synthèse

L'enquête de terrain nous a permis de constater qu'il y avait un vif intérêt pour les fonctionnalités sociales et un réel besoin de pouvoir interagir avec ses amis Facebook.

L'analyse des applications des autres manifestations a principalement fait ressortir le manque de convivialité dans l'utilisation. Cependant, la plupart des applications offrent bien plus de renseignements que l'application actuelle de *Vinéa*. Cela nous encourage à intégrer plus d'informations générales concernant la manifestation.

3. Proposition d'application

3.1. Côté client

En tenant compte du résultat de l'enquête menée auprès des utilisateurs, ainsi qu'en accord avec l'institut Icare, nous avons élaboré la proposition d'application suivante:

Figure 1 - Proposition d'application

« **Carte** » : permet à l'utilisateur de visualiser le plan de situation de *Vinéa* et de se géo-localiser.

« **Inviter mes amis à déguster un cépage** » : affiche les amis Facebook sur la carte de *Vinéa*.

« **Vins/encaveurs** » : présente les différents encaveurs et leurs vins et permet d'en ajouter à ses favoris.

« **Sentiers des vins** » : les visiteurs de *Vinéa*, à travers des parcours œnologiques, peuvent se laisser guider et en apprendre plus sur certains vins.

« **Favoris** » : permet de gérer les vins favoris.

« **Plus…** » : regroupe différentes informations générales sur la manifestation.

Des scénarios d'utilisation [6] qui ont été présentés à *Vinéa* ont été réalisés.

3.2. Côté « back-end »

Actuellement, chaque mise à jour de l'application doit être redéployée sur l'App Store[7]. Même s'il ne s'agit que d'un accent. De plus, il n'existe aucun moyen de gérer les stands de l'application par après.

Nous proposons de stocker toutes les données sur un serveur distant et ensuite d'y accéder depuis l'application mobile. Ainsi, toutes les données de l'application pourront être mises à jour à distance et synchronisées automatiquement.

[6] Annexe 2 : Scénarios d'utilisation
[7] Glossaire : App Store

4. Etat de l'art

4.1. Objectifs de l'analyse

Aujourd'hui, les méthodes de développement d'applications mobiles sont de plus en plus nombreuses. Depuis la sortie des premiers langages de programmation natifs mobiles en 2004, de multiples tendances se profilent.

L'objectif de cet état de l'art est de dresser un bref aperçu de toutes les pratiques existantes, de cibler, puis d'analyser celle qui correspond le mieux aux besoins de l'institut de recherche Icare et au projet d'application mobile à réaliser. Cette recherche est effectuée dans un cadre fixé par l'institut Icare notamment en ce qui concerne les critères technologiques.

4.2. Problématique

Les critères de base, imposés par Icare, en matière de choix des technologies sont relativement restrictifs :

- L'application native doit être exécutable sur les systèmes d'exploitation Android et iOS.
- Elle doit pouvoir être téléchargée depuis l'App Store et Google Play[8].
- Toutes les données doivent être externalisées sur un serveur distant afin de réduire le temps de mise à jour d'une application mobile.
- Les données de l'application doivent pouvoir être mises à jour sans faire passer l'application par les différents processus de validation de l'App Store.

Pour respecter ces critères, nous avons envisagé un développement natif. Toutefois, ce type de programmation nécessite la maîtrise de plusieurs langages et rend le développement compliqué : une application iOS est développée à l'aide du langage de programmation Objective C et une application Android grâce à Java. C'est pourquoi nous avons envisagé une autre alternative.

[8] Glossaire : Google Play

4.3. Solution

La démocratisation massive des Smartphones a démultiplié les différentes manières de développer des applications mobiles. L'arrivée du HTML5[9] et CSS3[10], modifie radicalement la conception des applications mobiles. Actuellement, des Frameworks[11] permettent de développer une application dans un seul langage pour ensuite le déployer sur les différents systèmes d'exploitation mobiles. Il devient par exemple possible de développer des applications natives en combinant les langages natifs avec des langages web ou de générer des applications natives à partir du langage JavaScript[12], entre autres. Un seul développement permet de diffuser une application sur plusieurs systèmes d'exploitation mobiles. Cet ensemble de pratiques, appelé le développement « crossplatform », est, en respectant les critères d'Icare, exactement ce dont nous avons besoin.

Etant donné que la pratique est actuellement encore assez nouvelle, il n'existe aucune tendance prédominante en matière de développement « crossplatform ». Plusieurs Frameworks ont vu le jour et chacun est présenté comme étant le meilleur, mais aucun ne sort vraiment du lot. Nous avons analysé et testé une multitude de ces Frameworks. Ceci nous a permis de les classer en quatre approches. Chacune a ses avantages mais aussi ses inconvénients par rapport à notre problématique.

4.3.1. Développer des applications natives générées

Cette pratique consiste à développer une application en un langage indépendant de la plateforme mobile (en général Ruby ou JavaScript). Un Framework nous permet d'accéder aux fonctionnalités natives du téléphone. Une fois le développement terminé, un traducteur de code source génère, à partir du Ruby ou du JavaScript, le code natif pour chaque plateforme.

Les avantages

- Possibilité d'accéder aux fonctions du téléphone : GPS, appareil photo, contacts…
- Déploiement sur les plateformes de téléchargements
- Performance élevée (aucune couche supplémentaire)

[9] Glossaire : HTML5
[10] Glossaire : CSS3
[11] Glossaire : Framework
[12] Glossaire : JavaScript

L'inconvénient

- Notre projet est dépendant de l'outil utilisé et par conséquent ne peut qu'être difficilement réutilisé au sein d'un autre environnement de développement

Les principaux outils

MoSync, XMLVM, Eqela

4.3.2. Développer des applications natives interprétées

Cette pratique consiste à développer une application mobile dans un langage spécifique à l'aide d'un environnement de travail donné. Nous sommes donc dépendants des limitations de l'outil. Des méthodes nous permettent ensuite d'accéder aux fonctionnalités du téléphone. Certains éditeurs (ex. Titanium Appcelerator) offrent même la possibilité d'accéder à des éléments graphiques natifs du téléphone. Ainsi, le « look and feel » de l'application est adapté en fonction de la plateforme utilisée. Le développement terminé, un interpréteur multiplateforme (couche intermédiaire entre le natif et le code développé) peut interpréter le code, le transcrire en natif et l'exécuter sur le périphérique mobile.

Les avantages

- Possibilité d'accéder aux fonctions du téléphone : GPS, appareil photo, contacts…
- Déploiement sur les plateformes de téléchargement d'applications
- Look and feel natifs

L'inconvénient

- Apprentissage du Ruby (RhoMobile) ou du Framework propriétaire Javascript (Appcelerator)

Les principaux outils

AppCelerator Titanium, RhoMobile, JMango

4.3.3. Développer des applications web

Cette solution consiste à développer un site internet optimisé pour un écran de téléphone portable à l'aide des technologies du web classique. L'utilisateur peut accéder au site grâce au navigateur internet de son téléphone portable.

Les avantages

- Développement avec des technologies web (HTML5, CSS3, JS)
- Développement avec des librairies JavaScript populaires
- Développement unique pour toutes les plateformes
- Le code source peut être réutilisé

Les inconvénients

- Aucune distribution possible sur les plateformes de téléchargement d'applications
- Performance moindre, car les pages ne sont pas stockées en local mais doivent être chargées lors de chaque requête.
- L'accès aux fonctions natives du téléphone est impossible ou, dans certains cas, très limité
- Une connexion internet est requise.

Les principaux outils

JQuery Mobile, Sencha, Dojo Toolkit

4.3.4. Développer des applications hybrides

Une application hybride est développée en HTML5, CSS3 et JavaScript comme une application web classique. Le projet web est ensuite inséré dans une coquille, qui est développée en langage natif. Elle a pour but d'afficher les pages web contenues sous forme de vues et de les faire communiquer avec les fonctions natives du téléphone. Ceci grâce à l'aide d'une librairie qui fait office de pont entre le natif et le JavaScript.

Les avantages

- Développement avec des technologies web (HTML5, CSS3, JS)
- Développement unique pour toutes les plateformes
- Accès aux fonctionnalités natives du téléphone
- Coûts et temps de développement faibles
- Portabilité de l'application
- Distribution possible sur les plateformes de téléchargement d'applications

L'inconvénient

- Performances inférieures, car une couche supplémentaire s'intercale entre le natif et les pages web.

Les principaux outils

PhoneGap, Adobe Flash Builder, The M Project

4.3.5. Comparatif des quatre approches et choix

Pour terminer, voici un tableau récapitulatif des quatre approches qui permettent de développer des applications « crossplatform »

	WEB APP	HYBRID APP	NATIVE APP INTERPRETEE	NATIVE APP GENEREES
Coûts et temps de développement	Normal	Normal	Raisonnable (pas de langage web)	Raisonnable (pas de langage web)
Performances	Dépend de la connexion internet	Rapide (couche intermédiaire)	Rapide (couche intermédiaire)	Très rapide
Déploiement sur les plateformes de téléchargement	Non	Oui	Oui	Oui
Développement unique pour toutes les plateformes	Oui	Oui	Oui	Oui
Langages de programmation	HTML5/CSS3/JS	HTML5/CSS3/JS	C++, C# ou JS	Ruby ou JS
Accès aux fonctionnalités natives du téléphone	Non	Oui	Oui	Oui
Code développé est régénéré en langage natif	Non C'est bien ?	Non	Non, mais interprété	Oui

Tableau 2 - récapitulatif des approches "crossplatform"

En respectant les critères minimaux de l'institut Icare, en recherchant les meilleures performances d'exécution et une optimisation du temps de développement, ce sont les applications hybrides qui correspondent le mieux à nos besoins.

Nous avons également remarqué que des grandes entreprises comme Facebook, LinkedIn ou Bing utilisent depuis 2011 cette approche d'application hybride pour la création de leurs applications mobiles.

La technologie est stable et le fait de pouvoir développer l'application avec des langages web, langages que nous maîtrisons, nous a définitivement convaincu que cette approche était la plus adaptée.

4.4. Outils et choix pour un développement hybride

4.4.1. Le Framework hybride de base

Afin de pouvoir développer des applications hybrides, nous devons utiliser un Framework qui permette à notre projet web d'être embarqué à l'intérieur d'une coquille native. Plusieurs entreprises offrent aujourd'hui des Frameworks pour développer ce type d'applications. Cependant, en fixant quelques critères minimaux comme,

- interagir avec la fonction de géolocalisation du téléphone,
- embarquer une base de données,
- intégrer Facebook,
- avoir un Framework stable (pas de version BETA),
- avoir un Framework compatible avec iOS et Android,
- développer avec des langages web (HTML/CSS/JS),

nous retenons les trois outils suivants :

	Sencha	PhoneGap	MoSync
Licence	Commercial	Libre	Commercial
Accès au GPS	Oui	Oui	Oui
Stockage local	Oui	Oui	Oui
Framwork UI intégré	Oui	Non	Oui
MVC / MVVM intégré	Oui	Non	Non
Documentation	++	++++	+

Tableau 3 - Comparatif des Frameworks hybrides

Nous avons décidé d'utiliser PhoneGap comme Framework hybride pour les raisons suivantes : l'outil est gratuit, la documentation disponible est abondante et le team de développement de l'institut Icare nous l'a conseillé. De plus, PhoneGap est le leader dans ce domaine et de nombreuses applications hybrides ont déjà vu le jour avec ce Framework.

4.4.2. La librairie d'architecture JavasScript

Le Framework hybride de base choisi, nous nous sommes penchés sur l'architecture de notre application.

Comme mentionné dans le chapitre précédent, notre projet hybride est basé sur les langages web HTML, CSS et JavaScript. Depuis toujours, JavaScript a causé des difficultés aux développeurs car les différentes variantes de ce langage rendaient souvent des fonctions incompatibles d'un navigateur à un autre. Plusieurs standards ont vu le jour et JavaScript est rapidement devenu un langage sale et peu structuré. Avec le temps, le JavaScript a été standardisé et aujourd'hui, ce langage est interprété de la même manière sur presque tous les navigateurs modernes. Cependant, la difficulté de structurer des projets JavaScript demeure.

Afin d'avoir une architecture d'application structurée, claire et efficace, nous avons choisi des librairies qui permettent le développement d'une architecture de type MVC. Cette approche nous permet de séparer les différentes couches de notre application et ainsi d'organiser notre projet de manière structurée en trois parties :

1) La première partie dite « modèle » s'occupe de la gestion des données (accès aux web services et stockage en base de données)
2) La seconde partie dite « vue » s'occupe du rendu et de l'affichage à l'utilisateur.
3) La dernière partie dite « contrôleur » s'occupe de gérer les différentes actions de l'utilisateur.

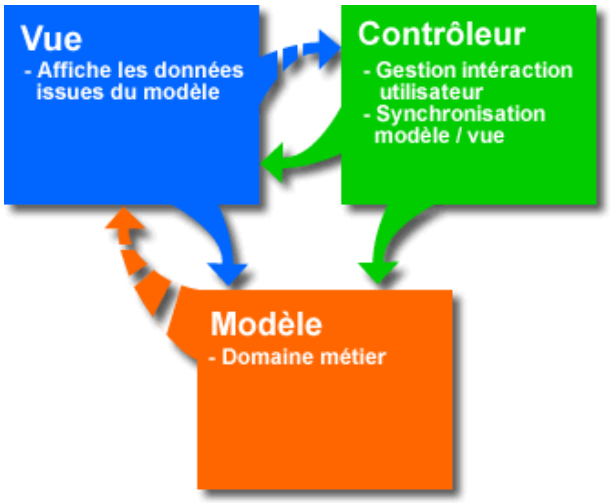


Figure 2 - Interaction MVC, source : http://blog.iteratif.fr

Par souci d'efficacité, nous avons choisi de développer notre architecture MVC à l'aide d'un Framework existant. Nous avons fixé certains critères, effectué des recherches et établi une liste des différents Frameworks existants.

Critères :

- librairie JavaScript légère
- utilise la syntaxe JQuery [13] pour l'interaction avec le DOM
- permet de réaliser du MVC ou équivalent

	batman.js	**Spine.js**	BACKBONE.JS	Ember.js	**Knockout.js**
Langage	JavaScript	JS généré à partir de CoffeeScript	JavaScript	JavaScript	JavaScript
Pattern utilisé	Modèle - vue	Modèle - vue	Modèle - vue	Modèle – vue – contrôleur	Modèle-vue-vue-modèle
Documentation	+	++	++++	+	+++
Commentaire	Manque des composants	Contient des bugs, manque de documentation	L'un des premiers Framework	Très jeune, manque de documentation	Pas de collections, data-binding est mélangé avec le HTML

Tableau 4 - Comparatif des librairies MVC

Le choix s'est rapidement porté sur Backbone.js. Il s'agit d'une librairie JavaScript robuste et légère à la fois qui offre une structure de code intéressante. D'autre part, la documentation pour Backbone.js est riche et abondante.

De plus, des grandes entreprises comme LinkedIn ou 37 Signals utilisent cette librairie pour leurs applications mobiles. Ce choix nous a vivement été conseillé par Icare.

Notons également que Backbone.js fonctionne uniquement avec une librairie de sélection comme JQuery, Zepto.js ou JqMobi. Le choix de la librairie de sélection sera décrit au chapitre suivant.

[13] Glossaire : JQuery

4.4.3. La couche graphique

Afin de pouvoir simuler une expérience « native » sur des vues web, nous avons analysé et testé de nombreux Frameworks graphiques optimisés pour des appareils tactiles.

Aujourd'hui, JQuery Mobile est sans conteste le leader dans ce type de Frameworks graphiques. Plusieurs applications mobiles ont été développées avec ce Framework au sein de l'institut Icare mais avec des résultats peu probants. En effet, JQuery Mobile est très lent. Le Framework a été développé afin d'être supporté par tous les moteurs de rendu[14] des navigateurs du marché. Il est très lourd car de nombreux tests sont effectués afin de garantir un rendu identique sur tous les navigateurs. L'institut Icare voulait une alternative à JQuery Mobile. A l'aide de celle-ci, nous devrions être en mesure de développer rapidement des interfaces graphiques sans que les performances de l'application ne baissent à cause de la lourdeur du Framework graphique.

Nous avons testé un grand nombre de possibilités et pouvons affirmer qu'il existe actuellement deux moyens de procéder. Soit, utiliser un Framework qui combine une librairie graphique avec une librairie de manipulation du DOM[15], (exemple : JQuery Mobile) soit, utiliser une librairie graphique indépendante de la librairie de manipulation du DOM (exemple : TwitterBoostrap combiné avec JQuery).

Des recherches et de nombreux tests nous ont permis d'établir un tableau comparatif[16] qui tient compte des tendances actuelles de développement mais aussi des performances et de la légèreté qu'offrent ces Frameworks graphiques.

Tout semblait désigner JQMobi[17] comme le successeur de JQuery Mobile. Cette librairie est la plus légère et la plus optimisée pour des projets PhoneGap. En effet, elle a été optimisée pour le moteur de rendu webkit utilisé par PhoneGap. Nous avons donc commencé le développement de notre projet en utilisant ce Framework. Rapidement, nous avons remarqué qu'il était très performant, mais qu'il contenait beaucoup d'erreurs et n'était pas assez complet par rapport aux fonctions proposées. Les transitions étaient lentes et souvent des comportements non-souhaités surgissaient. De plus JQMobi est très mal documenté.

---

[14] Glossaire : Moteur de rendu
[15] Glossaire : Librairie de manipulation du DOM
[16] Annexe 3 : Tableau comparatif des Frameworks graphiques
[17] Glossaire : JqMobi

Nous nous sommes donc intéressés à la solution JQuery combinée avec du CSS. Même si la combinaison Zepto.js avec du CSS est plus légère, la librairie de sélection JQuery demeure plus stable et surtout plus rapide. Cette variante nous a permis de développer un outil performant tout en rationnalisant le poids de notre projet.

Afin de simuler une expérience graphique tactile, nous avons utilisé une librairie JavaScript IScroll 4. Celle-ci est utilisée pour simuler des listes déroulantes natives. En ce qui concerne les feuilles de style, elles ont été intégralement développées à la main.

4.4.4. La base de données

Les données que nous allons récupérer depuis le serveur distant doivent pouvoir êtres stockées en local sur le téléphone, afin de pouvoir utiliser l'application en mode « hors-connexion ». La librairie PhoneGap offre la possibilité de créer en natif une base de données SQLITE. Par contre, si dans un avenir proche, nous voulons utiliser le projet *Vinéa* en dehors d'une coquille PhoneGap, toute la base de données serait inutilisable.

Pour cette raison, nous nous sommes intéressés aux récentes options de stockage qu'offre HTML5. Avec ses moyens de stockage novateurs, HTML5 permet d'enregistrer les données d'un projet directement dans la mémoire cache du navigateur.

Il existe acteullement trois principales façons de stocker des données à l'aide d'HTML5 :

Le localStorage

Il s'agit d'un système de stockage simple qui utilise la combinaison clé/valeur.

Avantages : Simple d'utilisation, disponible sur la plupart des nouveaux navigateurs, il existe un adaptateur qui permet de faire communiquer nos modèles développés avec Backbone.js avec le localStorage.

Inconvénients : Pas de langage de requêtes et les normes de sécurité ne sont pas standardisées.

Indexed Database API

Il s'agit d'un fichier plat avec un système de stockage clé/valeur. Possède une indexation des données.

Avantage : Très simple d'utilisation

Inconvénients : non-disponible sur la plupart des navigateurs, ne possède pas de langage de requêtes

Web SQL

Il s'agit d'une base de données Sqlite stockée dans la mémoire cache du navigateur

Avantages : supporte le langage de requêtes SQL, très rapide lors de l'exécution de requêtes

Inconvénients : La W3C a laissé de côté cette pratique et donc non-standardisé.

Pour des raisons de performances, de standardisation et de légèreté, mais aussi de compatibilité avec Backbone.js, nous avons décidé de choisir de stocker nos données à l'aide du localStorage.

4.5. Les librairies supplémentaires

Une fonctionnalité principale à développer pour le projet *Vinéa* était l'affichage d'une carte qui situe tous les stands. Afin de réponde à ce besoin il nous fallait une librairie qui nous permette de mettre en place un moteur cartographique web afin d'afficher de manière fluide le plan de situation. Après quelques recherches nous avons pu lister les trois librairies les plus populaires suivantes :

	OpenLayers™	Google maps	bing maps
Licence	Open source	Payant pour les gros volumes	Payant pour les gros volumes
Possibilité d'utiliser nos propres cartes	Oui	Non	Non
Possibilité de stocker les cartes en local	Oui	Non	Non

Tableau 5 - Tableau comparatif des moteurs cartographiques

Afin d'utiliser cette fonctionnalité sans connexion internet, il était impératif de pouvoir stocker nos propres cartes en local sur l'appareil. OpenLayers est le seul Framework qui offre cette possibilité. s De plus, cette librairie open-source s'intègre à merveille à l'intérieur de projets JavaScript et offre de larges possibilités de personnalisation.

4.6. Synthèse

En résumé, nous utilisons et conseillons, pour un développement hybride efficace, les technologies et outils suivants :

La librairie **PhoneGap** pour la passerelle entre le code natif et les vues web.

Argument phare : Leader dans son domaine, documentation abondante disponible et librairie libre.

La librairie **Backbone.js** pour le squelette et l'architecture de base de notre application.

Argument phare : Librairie la plus robuste et la plus légère. Elle est utilisée par des grandes entreprises telles que LinkedIn ou 37 Signals.

La combinaison de **JQuery, HTML5 et CSS3** pour toute la partie du rendu des vues.

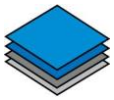
Argument phare : Modulable, stable et léger. Les librairies graphiques web optimisées pour mobiles ne sont pas encore assez performantes.

IScroll 4 pour simuler avec du langage web des listes déroulantes natives.

Argument phare : Extrêmement léger, performant. Utilisé par LinkedIn et d'autres grandes entreprises pour leurs applications hybrides.

Le **localStorage HTML5** pour tout ce qui est du stockage des données

Argument phare : Standardisé par la W3C, performant et léger



Openlayers pour le moteur cartographique web.

Argument phare : Opensource, beaucoup de documentation et offre une personnalisation énorme des cartes.

5. Description de l'application

5.1. Fonctionnalités développées

Cette partie du document, regroupe un aperçu des fonctionnalités réalisées[18]. L'écran d'accueil de l'application permet à l'utilisateur d'avoir rapidement un aperçu des fonctionnalités à disposition. La barre de navigation du bas lui permet de naviguer entre ces différentes fonctionnalités que sont la liste des vins et encaveurs, le plan de la manifestation, sa liste de favoris, des parcours ou alors le programme de la manifestation. C'est au lancement de l'application que cette dernière va vérifier si les données qu'elle contient sont encore à jour par rapport au web-service distant. Si ce n'est pas le cas, les données modifiées ou supprimées sont téléchargées et remplacées en local sur le téléphone mobile.

Figure 3 - Aperçu général de l'application

[18] Annexe 4 : Accès au téléchargement de l'application

Fonctionnalité « Vins et encaveurs »

Cette première vue permet à l'utilisateur de lister les encaveurs et vins présents à la manifestation.

Un champ de saisie permet d'effectuer une recherche sur les caves. Les deux onglets du haut permettent à l'utilisateur de naviguer entre la liste des vins et la liste des encaveurs.

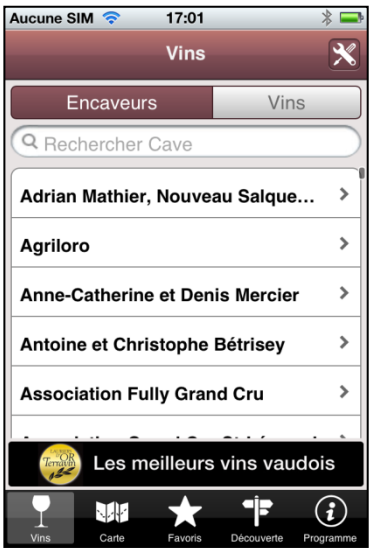


Figure 4 - Liste des encaveurs

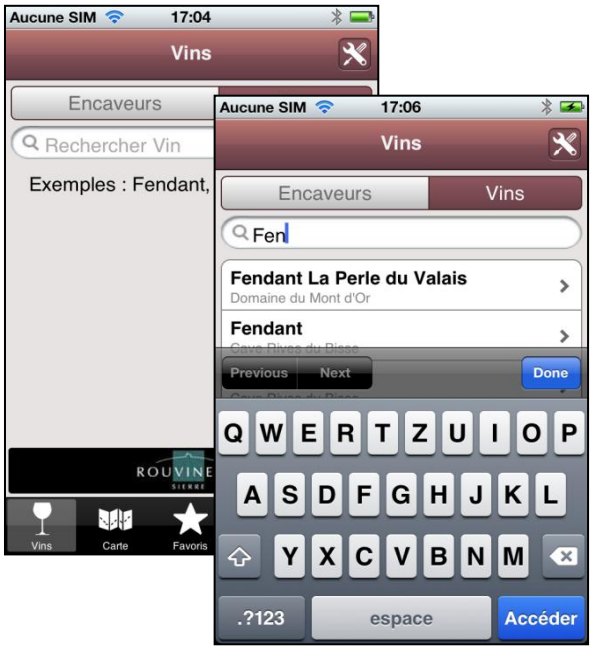


Figure 5 - Recherche d'un vin

En basculant sur l'onglet « Vins », seule cette barre de recherche est disponible. Aucune liste n'est affichée car la base de données contient plus de 2100 vins ce qui aurait considérablement ralenti l'application.

L'utilisateur saisit un terme de recherche et la liste des résultats apparait.

Si l'utilisateur sélectionne une cave, l'écran ci-contre s'affiche. Y figurent des informations sur la cave ainsi que la liste des vins proposés en dégustation. A partir de cette liste, l'utilisateur peut sélectionner un vin qui le conduira vers l'écran suivant.

Dans la barre de navigation du haut, l'utilisateur a la possibilité de visualiser l'emplacement de la cave sur la carte.

Figure 6 - Détail d'un encaveur

Cet écran apparait lorsque l'utilisateur décide d'afficher le détail d'un vin soit depuis l'écran de détail d'une cave, soit depuis l'écran de recherche d'un vin.

Si l'utilisateur a apprécié ce cépage, il a la possibilité de noter ce vin, d'y apporter un commentaire et de l'ajouter à ses favoris.

Dans la barre de navigation du haut, l'utilisateur a également la possibilité de localiser le stand dans lequel il peut déguster le cépage sélectionné.

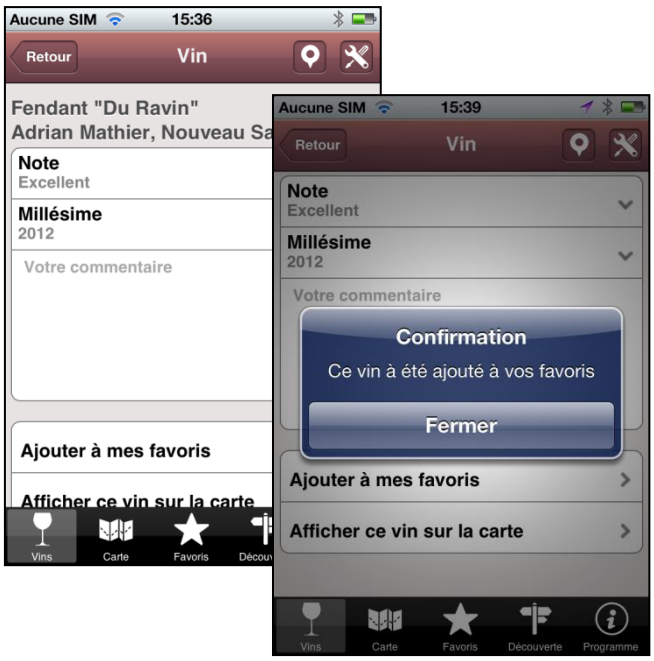


Figure 7 - Détail d'un vin

Lorsque l'utilisateur décide de localiser une cave via l'écran « détail d'un vin » ou « détail d'une cave », la carte ci-dessous met en évidence, à l'aide d'une animation, le stand recherché.

Figure 8 - Localisation d'un stand

Fonctionnalité « Carte »

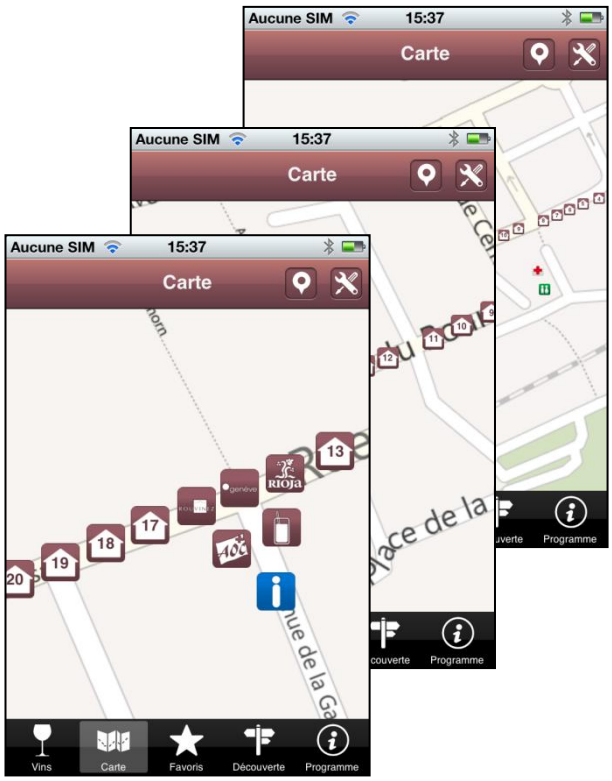


Figure 9 - Fonctionnalité carte

En cliquant sur l'onglet « Carte », un plan de situation avec les stands de *Vinéa* s'affiche. Cette fonctionnalité à été implémentée afin que l'utilisateur puisse l'utiliser sans connexion internet. Les images de la carte sont embarquées au sein du projet mobile.

A l'aide d'un mouvement des doigts, l'utilisateur peut zoomer jusqu'à trois niveaux différents.

En cliquant sur les icônes, différentes informations s'affichent.

Dans la barre de navigation du haut, un bouton permet de se situer sur la carte. La géolocalisation est uniquement possible si l'utilisateur reçoit un signal GPS ou est connecté à un réseau internet mobile. Le signal GPS est toutefois recommandé.

En cliquant sur un stand de type « tente » l'écran ci-dessous apparaît. Ce détail contient une liste des encaveurs qui sont présents sous la tente en question. A partir de cet écran, l'utilisateur peut afficher le détail d'une cave. Si l'utilisateur clique sur un stand de type « générique » une petite fenêtre, avec une description du stand, s'affiche.

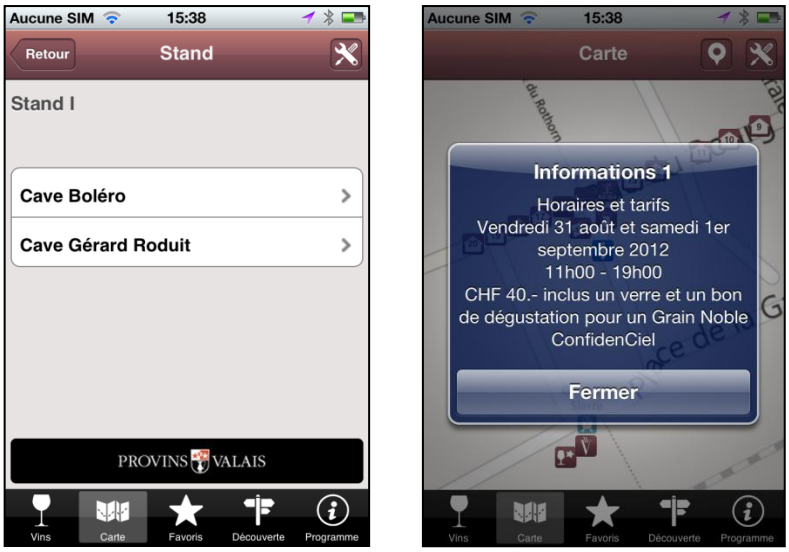


Figure 10 - Détail d'un stand

Fonctionnalité « Favoris »

L'écran « détail d'un vin » permet à l'utilisateur d'ajouter un vin à ses favoris. Les écrans ci-dessous permettent à l'utilisateur de modifier ou de supprimer ses favoris.

Un bouton de partage « Facebook » offre la possibilité de publier ses favoris sur son mur Facebook. Un autre bouton permet d'exporter ses favoris par mail. Un mail contenant ses favoris est automatiquement généré et l'utilisateur et redirigé vers l'application mail de son Smartphone.

Figure 11 - Gestion des favoris

Lorsque l'utilisateur clique sur le bouton de partage « Facebook » il autorise « Vinéa Mobile » à publier sur son mur. Le processus d'autorisation terminé, il peut publier ses favoris et ajouter un commentaire.

Figure 12 - Processus de publication sur Facebook

A partir du statut publié sur Facebook, il est possible de pointer vers une page internet valide. Une page web standard, qui récupère les vins favoris partagés, a été créée. Cette page s'affiche de la manière suivante :

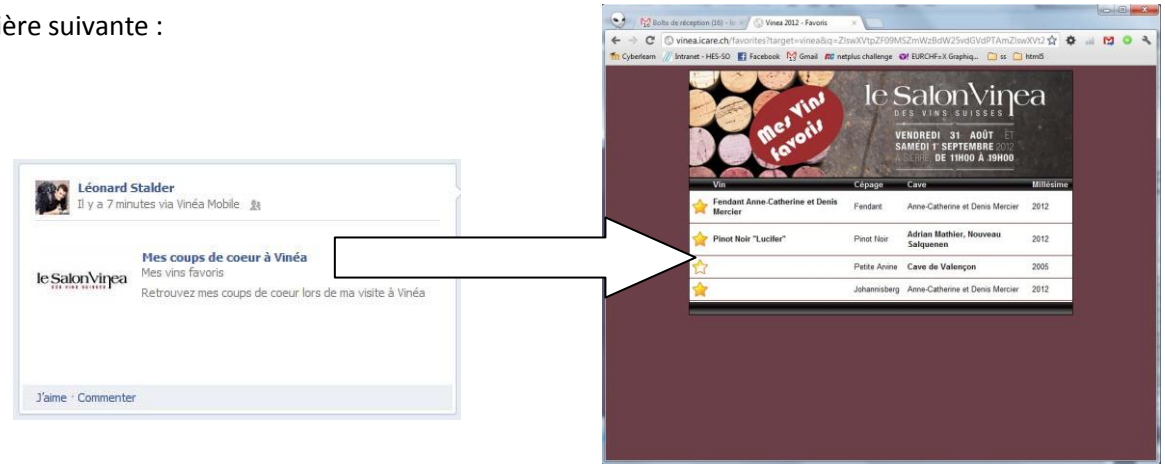


Figure 13 - Partage sur Facebook

Fonctionnalité « Découverte »

Afin d'affiner ses connaissances viticoles, différents parcours œnologiques sont proposés. *Vinéa* propose à ses sponsors de parrainer ces parcours découverte. Le jour de la manifestation, plusieurs parcours seront définis en fonction des vins sélectionnés par les sponsors (par exemple : Parcours Manor).

L'utilisateur peut visionner les vins d'un parcours et en cliquant sur un vin, il sera redirigé vers l'écran « détail d'un vin ». Ces parcours peuvent à tout moment êtres modifiés depuis l'interface d'administration distante.

Figure 14 - Parcours découverte

Fonctionnalité « Programme »

L'onglet « Programme » contient un menu avec des informations générales concernant la manifestation *Vinéa*.

Figure 15 - Informations générales

Fonctionnalité « paramètres »

En cliquant sur le bouton « paramètres » en haut à droite, l'utilisateur est redirigé vers une page de paramètres. A partir de cette page, il peut afficher l'aide de l'application ou la page « à propos ».

Figure 16 - Paramètres

Publicité

Pour financer l'application, des bandeaux publicitaires figurent au bas de l'écran. Ceux-ci apparaissent de manière aléatoire sur certaines pages.

En cliquant sur une publicité, l'utilisateur est redirigé vers une page à disposition du sponsor.

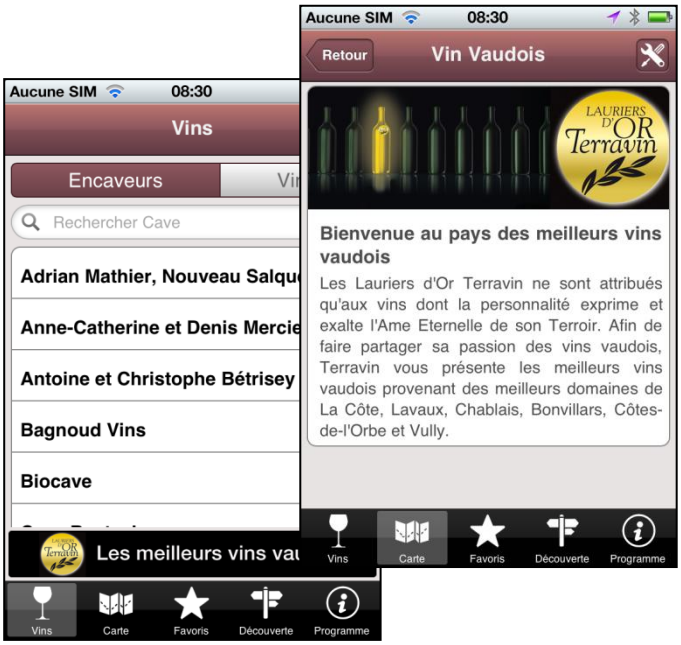


Figure 17 - Publicité

5.2. Fonctionnalités abandonnées

La seule fonctionnalité abandonnée en cours de route a été la géo-localisation de ses amis Facebook.

Une première version de cette fonctionnalité a été développée. Cependant, elle n'était pas assez performante pour une mise en production de l'application début août 2012. Elle contenait des bugs et, après plusieurs tests sur le terrain, nous avons remarqué que la géo-localisation sur un aussi petit site que *Vinéa* n'était pas assez précise.

Pour ces raisons et dans le but de nous concentrer sur l'amélioration des fonctionnalités existantes et ainsi garantir une mise en production début août 2012, nous avons, d'un commun accord avec l'institut Icare, décidé d'abandonner cette fonctionnalité pour cette version de l'application.

5.3. Fonctionnement global

Notre projet web a été embarqué dans deux coquilles natives : une coquille iOS et une Android. Celles-ci nous permettent d'avoir deux applications mobiles qui tournent sur deux systèmes d'exploitation. Dans cette partie du document, nous allons nous intéresser au projet web qui est embarqué dans la coquille PhoneGap et aux différentes interactions qu'il effectue.

Nos deux applications mobiles sont basées sur une architecture de type MVC. Elles fonctionnent soit en mode « hors ligne » soit en mode « en ligne ». Les différentes données de nos applications sont stockées dans une base de données locale et sont synchronisées lors de chaque lancement de l'application avec un web-service[19] distant, si une connexion internet est disponible.

Pour la fonctionnalité « informations générales », les deux applications utilisent des pages html hébergées sur un serveur web distant. Nous avons préféré externaliser ces pages, car les informations qu'elles contiennent sont continuellement modifiées. Si le programme de la manifestation ou si les horaires d'ouvertures changent, il suffit de mettre à jour ces pages sur le serveur web distant et l'application mobile affichera la page corrigée. L'inconvénient de ce système est que la fonctionnalité est uniquement disponible avec une connexion internet.

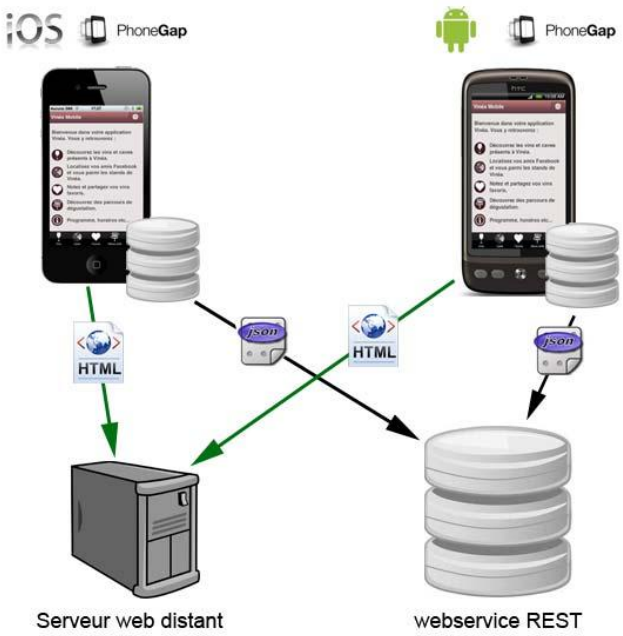


Figure 18 - Fonctionnement global

[19] Glossaire : web service

5.4. Architecture de l'application

Dans une première partie, nous traiterons les concepts de base de l'architecture de l'application hybride. Dans une deuxième partie nous allons nous intéresser à l'organisation du code et à la gestion et à la synchronisation des données. Pour terminer, nous évoquerons les difficultés rencontrées.

5.4.1. Concept d'architecture utilisé

Nous avons décidé d'utiliser la librairie Backbone.js afin de structurer notre application web et d'obtenir une architecture de type modèle – vue – contrôleur. La librairie Backbone.js nous offre les possibilités suivantes :

- Gérer des modèles (équivalents aux classes-objets Java)
- Créer des collections de modèles (tableaux d'objets)
- Créer des vues qui peuvent écouter les actions de l'utilisateur et des modèles
- Créer un routeur qui permet de naviguer entre les différentes vues

La grande force de Backbone.js consiste à pouvoir lier tous ces éléments entre eux. Les vues vont gérer, à l'aide de « templates HTML », l'affichage et la capture des événements. Elles peuvent êtres attachées à des modèles et ainsi, dès qu'un modèle est modifié, déclencher automatiquement des événements. Un routeur permet de gérer la navigation entre nos différentes vues. Il fonctionne, en quelque sorte, comme un aiguilleur. Ainsi, si une vue veut pointer vers une autre, elle indique au routeur la « route » qui correspond à la nouvelle vue à afficher.

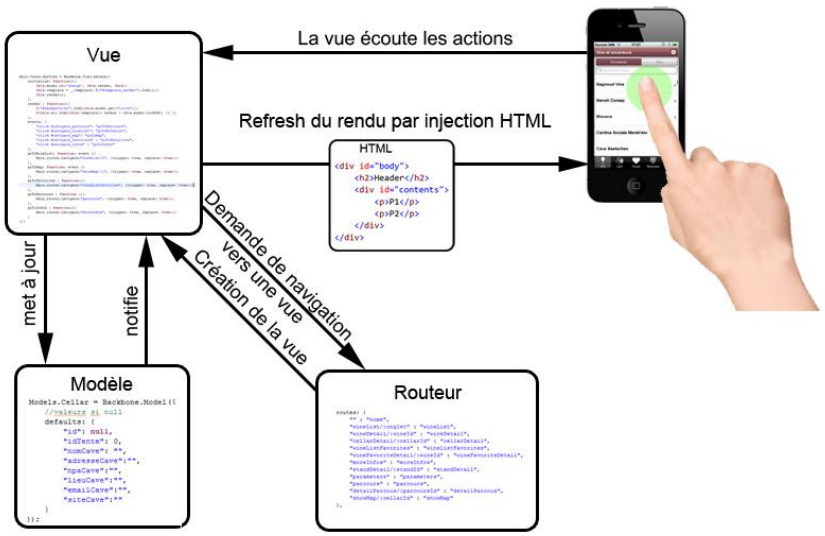


Figure 19 - Fonctionnement de Backbone.js

La compréhension du fonctionnement et la mise en place de Backbone.js a été très fastidieuse. Cependant, une fois en place, sa structure de code lisible permet de créer aisément des vues et des modèles.

Remarque : fonctionnement du « refresh » des vues à l'aide de « templates HTML »

Le fichier index.html d'un projet qui utilise Backbone.js et son système de rendu par injection de « templates HTML » est structuré de la manière suivante :

```
<body>
	<div id="header">
	</div>
	<div id="content">
	</div>
	<div id="navbar">
	</div>
	<script type="text/template" id="template_X">
		<p>Mon template de la vue X</p>
	</script>
	<script type="text/template" id="template_Y">
		<p>Mon template de la vue Y</p>
	</script>
 </body>
```

Des « templates HTML » sont appelés par les vues et insérés dans les différents containers qui seront visibles à l'écran. Lorsqu'on quitte une vue, le contenu est effacé et la nouvelle vue charge son « template HTML ». Les vues modifient cette page en fonction des actions de l'utilisateur.

5.4.2. Organisation du projet et du code

Dans l'optique de pouvoir réutiliser le code, ce chapitre décrit l'organisation de notre projet web.

Afin que l'application puisse être exécutée sur un téléphone mobile, le dossier « www » à été inséré dans une coquille PhoneGap sous « assets ».

Ci-contre, une image de notre projet web qui contient 6 dossiers. Les dossiers « cordovaAndroid » et « cordovaIphone » contiennent les librairies PhoneGap.

Au niveau graphique, les feuilles de styles se trouvent dans le dossier « css ». La feuille de styles « vineamobile.css » contient le CSS général et la feuille de style « styleOpenLayers.css » le CSS pour notre carte off-line.

Les images icônes utilisées dans notre application sont dans le dossier « images ». Les tuiles, qui permettent de recomposer la carte, sont également contenues dans celui-ci.

Le dossier data, inclut les données nécessaires à l'initialisation de l'application. Le fonctionnement sera décrit au chapitre 5.5.3.

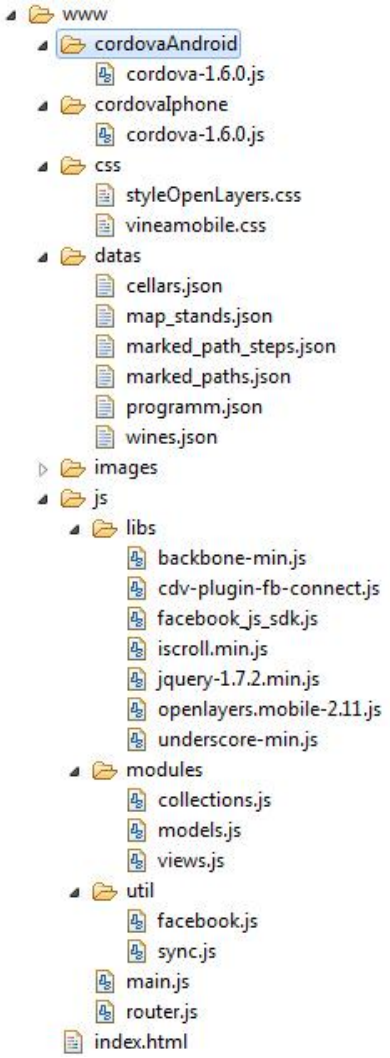


Figure 20 - Structure du projet web

Le dossier « js » englobe l'ensemble du code JavaScript de l'application. Les librairies externes (Backbone.js, JQuery, OpenLayers etc.) sont inclues dans le dossier « jslib ». Nos collections, modèles et vues sont contenues dans le dossier module. Le dossier « util » incorpore les classes d'accès aux données des web-services (sync.js) ainsi qu'une classe facebook.js qui nous permet de connecter l'application à Facebook. La classe « router.js » gère la navigation entre les vues. Une classe « main.js » permet d'instancier l'application ainsi que la mise à jour des données. La page « index.html » contient le HTML qui s'affiche à l'écran.

5.5. Gestion des données

5.5.1. Fonctionnement de l'accès et la synchronisation des données

Les données de notre application mobile sont stockées en local dans une base de données embarquée et sont synchronisées avec un web-service distant. Le web-service ainsi qu'une interface d'administration ont été réalisés en collaboration avec l'institut Icare. A l'aide de nos schémas de données, l'institut Icare a réalisé le web-service. L'accès et la synchronisation des données s'effectuent comme suit :

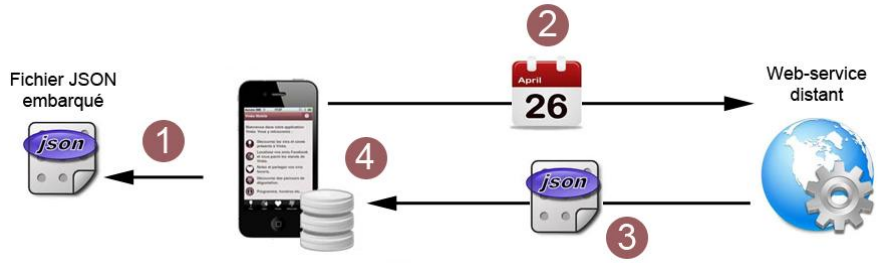


Figure 21 - Fonctionnement global

1) L'utilisateur lance pour la première fois l'application mobile. Nous avons embarqué dans le projet mobile un fichier plat contenant les données de base. Lors de ce premier démarrage, un script récupère les données dans ce fichier plat et les insère dans le localStorage de l'application. Ainsi, l'application n'est pas dépendante d'une connexion internet pour effectuer la première importation des données.
2) Lorsque l'utilisateur relance l'application, celle-ci contient déjà des données dans le localStorage. Elle ne récupère plus les fichiers plats qui sont embarqués, mais elle se connecte à un web-service distant afin de savoir si des nouvelles données sont disponibles. Pour ce faire, l'application mobile envoie au web-service une date de la dernière mise à jour. Cette opération est transparente pour l'utilisateur et elle est effectuée à chaque fois que celui-ci exécute l'application.
3) En fonction de la date de mise à jour reçue, le web-service retourne un fichier JSON qui contient les enregistrements qui ont été modifiés, ajoutés ou supprimés. Voici un exemple de JSON retourné :

```
[{"id":2,"idCave":35,"nomCave":"Anne-Catherine et Denis Mercier",
"cepage":"Johannisberg","nom":"","annee":2010,"created_at":"2012-07-
18T09:06:13+02:00","updated_at":"2012-07-18T09:06:13+02:00", "deleted_at":null}]
```

4) L'application mobile reçoit la réponse sous forme d'un fichier JSON. A l'aide des trois derniers attributs, à savoir "created_at" , "updated_at" et "deleted_at" celle-ci pourra soit insérer un nouvel enregistrement, soit modifier un enregistrement existant ou alors supprimer un enregistrement existant dans le localStorage.

5.5.2. Services distants

Le web-service, réalisé par l'institut Icare, nous permet de récupérer, à l'aide de requêtes AJAX, les données suivantes :

Liste des caves

Ce service retourne la liste des encaveurs. On y trouve les informations relatives à un encaveur. Voici un exemple d'un fichier JSON retourné :

```
[{"id":15,"idTente":1,"nomCave":"Nouveau Salquenen ","adresseCave":"Bahnhofstrasse",
"npaCave":"3970","lieuCave":"Salquenen","emailCave":"info@mathier.com","siteCave":"ht
tp://www.mathier.com","created_at":"2012-07-18T09:06:12+02:00","updated_at":"2012-07-
24T09:58:36+02:00", "deleted_at":null}, {...}]
```

Liste des vins

Ce service retourne la liste des vins. On y trouve le nom du vin ainsi que le cépage et l'encaveur. Voici un aperçu d'un fichier JSON retourné:

```
[{"id":2,"idCave":35,"nomCave":" Provins","cepage":"Dôle","nom":"Dôle de Sion",
"created_at":"2012-07-18T09:06:13+02:00","updated_at":"2012-07-18T09:06:13+02:00",
"deleted_at":null},{...}]
```

Liste des stands

Ce service retourne la liste des stands de *Vinéa*. On y trouve le nom du stand avec ses coordonnées géographiques afin qu'on puisse le placer sur la carte. Voici un aperçu d'un fichier JSON retourné:

```
[{"id":10,"name":"Stand      10","code":"1","latitude":46.29305,"longitude":7.532958,
"description":"","created_at":"2012-07-23T21:51:07+02:00","updated_at":"2012-08-
06T21:21:03+02:00","deleted_at":null},{...}]
```

Liste des parcours

Ce service retourne la liste des parcours que *Vinéa* propose. On y trouve le nom du parcours ainsi qu'une brève description. Voici un aperçu d'un fichier JSON retourné:

```
[{"id":3,"title":"Parcours de la Syrah","comment":"Les meilleures Syrah de Vin\u00e9a
selon       Univerre","created_at":"2012-07-18T18:05:59+02:00","updated_at":"2012-07-
18T18:05:59+02:00","deleted_at":null},{...}]
```

Détail des parcours

Ce service retourne la liste des parcours que Vinéa propose. On y trouve le nom du parcours ainsi qu'une brève description. Voici un aperçu d'un fichier JSON retourné:

```
[{"idParcours":1,"idWine":49,"created_at":"2012-07-18T18:07:27+02:00",
"updated_at":"2012-07-18T18:07:27+02:00","deleted_at":null},{...}]
```

5.5.3. Stockage local

Comme mentionné au chapitre 5.4.2, notre application mobile possède une base de données embarquée qu'elle synchronise avec un web-service distant. Nous utilisons le localStorage HTML5 pour stocker nos données. Le localStorage n'offre pas les possibilités d'une base de données relationnelle et offre uniquement un stockage des données qui utilise la combinaison clé/valeur.

Pour sauvegarder nos collections et modèles créés avec Backbone.js et afin de simuler les comportements d'une base de données relationnelle nous avons utilisé un adaptateur. Celui-ci sert de pont entre nos collections et modèles et le localStorage.

Nos enregistrements sont stockés de la façon suivante :

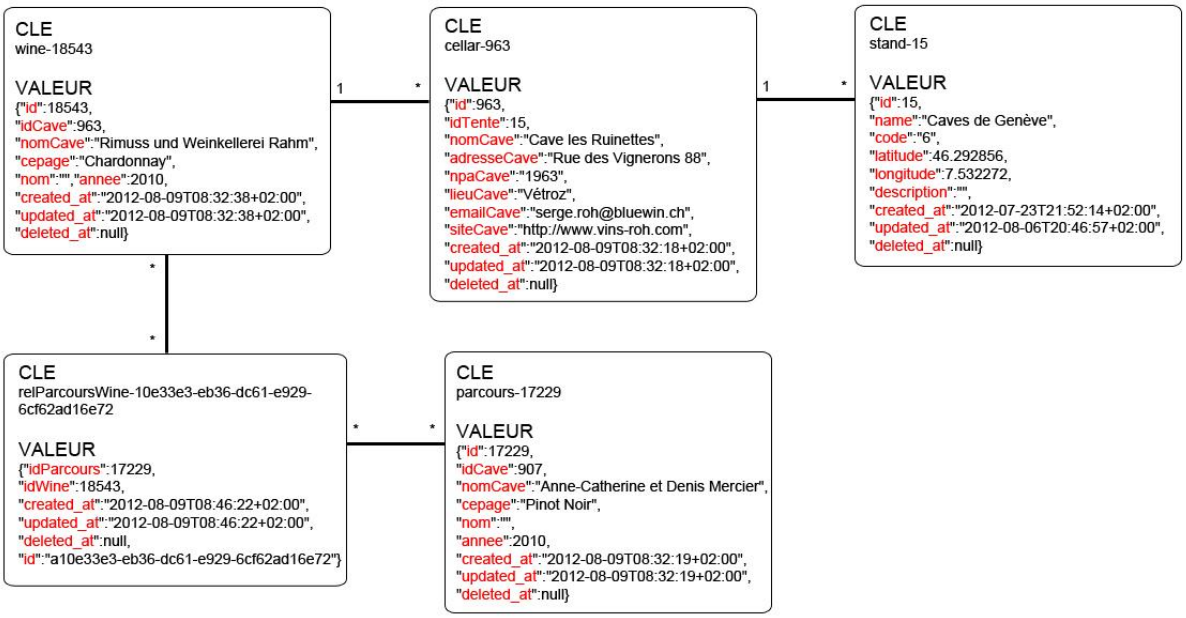


Figure 22 - Schéma de base de données locale

5.6. Difficultés rencontrées lors du développement

Ce chapitre recense les principaux problèmes rencontrés lors du développement de notre application.

Prise en main des nouvelles technologies

L'utilisation du JavaScript comme architecture de base d'une application web était complétement nouveau pour nous. En effet, nous étions habitués à développer des application web avec la notion de client-serveur classique. Un temps de formation que nous avons sous-estimé à été nécssaire.

Fonction Carte

Une des fonctionnalités principales de notre application consiste à afficher le plan de situation de la manifestation.

Figure 23 - Carte

Nous avons utilisé pour cela la librairie gratuite et open-source Openlayers. Cette librairie, très riche en fonctionnalités, nous permet de mettre en place un moteur cartographique web afin d'afficher de manière fluide des cartes.

Différents calques peuvent êtres superposées sur la carte. Nous avons utilisé ce système de claques afin de placer nos stands sur la carte, mais aussi afin de géo-localiser l'utilisateur.

Par défaut, une carte OpenLayers ne fonctionne qu'avec une connexion internet. Cependant, la carte devait être disponible en mode hors-ligne et avec trois niveaux de zooms différents.

Aujourd'hui, que se soit Google Maps, Bing Maps ou OpenLayers, tous utilisent la même projection cartographique [20] afin de représenter la surface de la terre. Le principe est d'afficher des tuiles l'une à côté de l'autre et au final d'obtenir un rendu qui représente l'ensemble de la carte. La seule différence entre ces fournisseurs de moteurs cartographiques est la manière dont ces tuiles sont indexées.

Figure 24 - Structure d'indexation Bing

Afin d'avoir une carte en mode « hors-linge », nous avons téléchargé ces différentes tuiles pour les ajouter, avec la même indexation, à notre projet web en local. Lors de l'affichage de la carte, nous avons modifié la source des tuiles qui pointait vers le serveur cartographique d'OpenLayers afin de la rediriger vers un dossier local contenant les tuiles téléchargées.

[20] Glossaire : Projection cartographique

Rapidement, nous avons remarqué qu'OpenLayers n'offrait que 17 niveaux de zoom sur ces cartes. Cela ne suffisait pas pour pouvoir distinguer les stands sur la carte.

Nous nous sommes alors intéressés au serveur cartographique Bing, qui offrait des tuiles allant jusqu'à 20 niveaux de zoom. Le but étant de récupérer les tuiles du service cartographique Bing pour ensuite remplacer les nôtres avec celles-ci. Comme les différents services de cartographie n'utilisent pas la même manière d'indexer leurs tuiles, nous avons dû convertir l'indexation des tuiles Bing pour qu'elles soient compatibles avec OpenLayers. Ce travail a été réalisé manuellement et a nécessité un investissement en temps important.

Facebook connect

Afin de pouvoir simuler une expérience utilisateur la plus native possible, nous avons décidé d'utiliser un plugin PhoneGap. Il devait nous permettre d'authentifier l'utilisateur auprès de l'application Facebook. Après quoi nous pouvions récupérer les amis Facebook de l'utilisateur et publier des statuts sur son mur.

Un plugin PhoneGap est en fait une extension de la librairie PhoneGap principale. Il s'agit de plugins natifs et sont donc écrits en Java pour Android et en Objective C pour iOS. L'intégration du plugin sous Android a pris passablement de temps.

Intégration de pages html distantes

La fonctionnalité « Programme » contient des informations générales de *Vinéa*. Afin de pouvoir mettre à jour ces pages à n'importe quel moment, elles ne sont pas embarquées dans le projet PhoneGap mais stockées sur un serveur distant. Elles sont récupérées depuis ce serveur et insérées à l'intérieur d'une balise HTML nommée *iframe*. Celle-ci permet d'intégrer un site internet distant à l'intérieur de notre application.

Les pages html externes sont téléchargées sur le téléphone et affichées à l'intérieur de la balise *iframe*. Nous avons rencontré un problème avec cette balise. Son comportement sur-définissait le comportement de certaines balises de notre application et créait des réactions non-attendues. Afin de contourner ce problème, nous avons dû retirer certaines balises lors de l'affichage des pages en question.

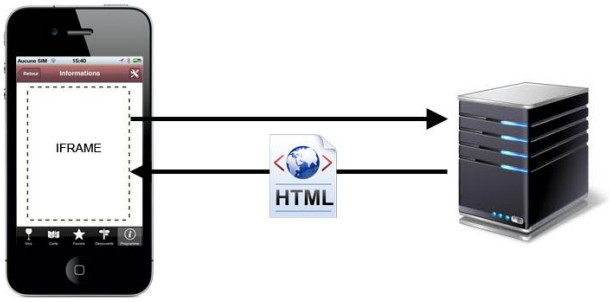


Figure 25 - Intégration de pages html distantes

Graphisme mobile de l'application

Comme mentionné dans notre état de l'art, nous avons décidé ne pas utiliser de librairies graphiques pour la simple et bonne raison qu'elles ralentissent trop les performances des applications hybrides.

Afin d'optimiser notre feuille de styles, nous avons uniquement utilisé des attributs CSS optimisés pour le moteur de rendu webkit. Notre application ne fonctionne donc pas sur d'autres moteurs de rendu web comme Gecko par exemple. Ce choix nous a permis d'optimiser notre feuille de styles pour notre projet web. Ainsi, nous avons pu alléger considérablement les opérations effectuées lors de l'affichage par le moteur de rendu.

Un gros effort à été fourni pour que notre projet web ait un look mobile et surtout que notre feuille de styles soit facilement réutilisable dans d'autres projets hybrides de l'institut Icare. La feuille de style de notre projet a été suffisamment commentée pour qu'elle puisse être réutilisée facilement.

Différences entre IOS et Android

Théoriquement le développement hybride est censé fonctionner. Nous avons effectué notre développement à l'aide d'une coquille Android. Lorsque nous avons voulu embarquer notre projet dans une coquille iOS, quelques fonctionnalités n'avaient pas le même comportement. L'initialisation de l'application était plus lente et les numéros des stands n'apparaissaient pas sur les tentes. Afin de régler ces problèmes, le code d'initialisation de l'application a été optimisé et un calque supplémentaire à été crée pour l'affichage des numéros de tente.

Au niveau de l'ergonomie et de la gestion de l'historique, les applications iOS et Android ne fonctionnent pas de la même manière. Des adaptations du code ont du être effectués afin que nos deux applications aient une ergonomie et une gestion de l'historique personnalisée en fonction du système d'exploitation

Figure 26 - Différences ergonomiques entre iOS et Android

Quantité et qualité des données

Intégration des données liées aux vins et encaveurs

Les données sur les encaveurs et les vins représentent environ 2300 enregistrements. Comme mentionné dans le chapitre 5.4.1, les données sont chargées dans le localStorage lors du premier lancement de l'application. Le temps d'écriture des données prenait beaucoup de temps. D'énormes efforts d'optimisation de code ont été fournis afin de réduire le temps de chargement des données de cette opération.

Intégration des données liées aux stands

De nombreuses informations nous sont parvenues tardivement ou sous une forme non exploitable. Par exemple, le plan du registre du cadastre fourni par Vinéa était inexploitable. Sans la position géographique des stands il nous était impossible de les placer sur la carte. Après de nombreuses recherches nous avons utilisé Google Earth afin de récupérer les coordonnées géographiques des stands. En effet, Google Earth offre la possibilité de superposer des images sur leur carte. Nous avons positionné des marqueurs sur les emplacements des stands pour récupérer les coordonnées géographiques. Toutes ces données ont été entrées manuellement.

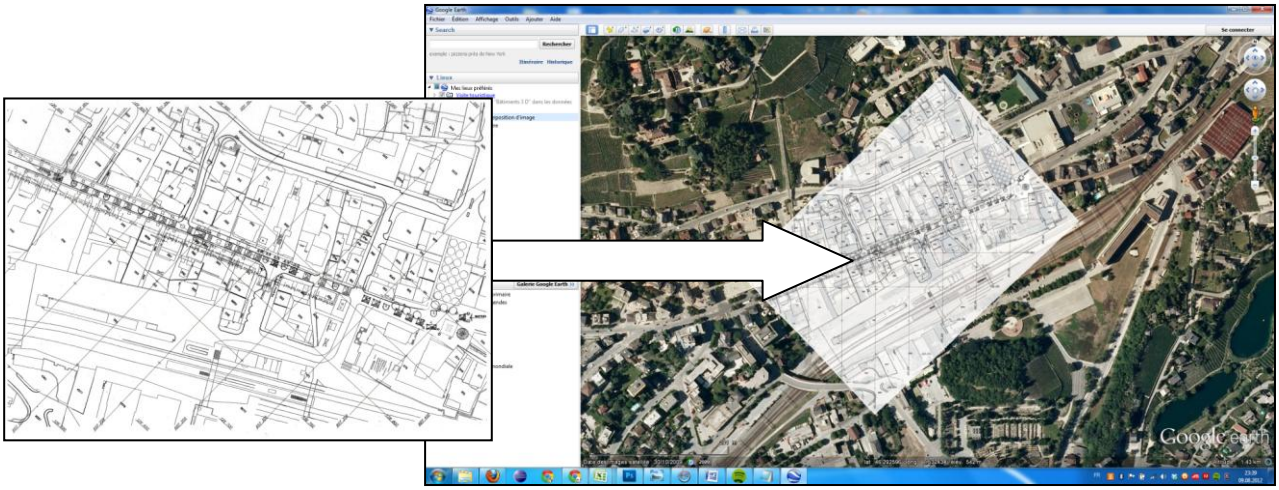


Figure 27 - Récupération des coordonnées géographiques

5.7. Industrialisation du logiciel

Etant donné que le projet allait être mis en production, nous avons cherché, tout au long du développement, d'optimiser notre code afin qu'une fois exécuté il soit le plus rapide possible.

Afin de diminuer au maximum la taille de notre projet mobile nous avons minimisé nos classes JavaScript. Ce procédé consiste à élever les espaces blancs, les commentaires et les noms des variables. Le temps de chargement des scripts et ainsi réduit et les performances de l'application en sont améliorées. Pour minimiser notre code nous avons utilisé le compilateur Google.

En accord avec l'institut Icare, il a été décidé de ne pas mettre en place des tests unitaires. La plupart du code n'était pas assez complexe pour être testé. Les méthodes les plus complexes ont été testées manuellement. Des tests fonctionnels durant lesquels plusieurs utilisateurs vérifient que l'application effectue bien ce qui a été demandé ont étés effectués. Le code a ensuite été réadapté en fonction de bugs relevés.

L'application mobile iOS a été publiée sur l'App Store par l'institut Icare, car un Mac possédant le logiciel Xcode était nécessaire.

L'application Android va être publiée sur Google Play par mes soins.

5.8. Améliorations possibles

L'application existante ne contient aucune erreur et est utilisable. Cependant quelques modifications pourraient la rendre encore plus efficace.

Les images des icônes placées sur la carte sont embarquées dans le projet mobile et ainsi non modifiables. Si l'année prochaine les images des stands veulent être changées, l'application doit être redéployée. Une des améliorations serait d'externaliser le stockage de ces images sur le « back-end » afin qu'elles puissent être gérées à distance.

La fonction géo-localisation des amis Facebook avait été abandonnée en cours de route. Un gros travail avait déjà été effectué afin de mettre en place cette fonctionnalité. Quelques jours de développement suffiraient afin de finaliser cette fonctionnalité et pouvoir la mettre en production.

Les bannières publicitaires de notre application ne sont pas modifiables à distance. En effet, si l'on veut modifier les annonceurs il est nécessaire de redéployer l'application.

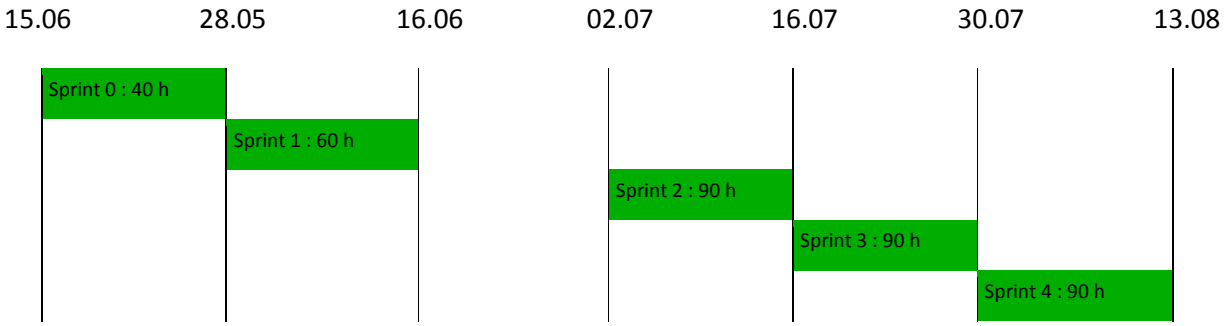
6. Gestion de projet

6.1. Panification des tâches

Une gestion de projet *Agile* a été adoptée afin d'offrir un maximum de flexibilité au suivi du développement de notre application.

Dans un premier temps, un *Product Backlog*[21] a été rédigé avec les différents acteurs du projet. Ce document, qui évolue au fur et à mesure du développement, a pour objectif de regrouper toutes les fonctionnalités qui répondent aux besoins du client. Celles-ci sont exprimées sous forme d'*User-Stories*. Par la suite, des priorités ainsi que des estimations de temps ont été attribuées aux *User-Stories*.

Les fonctionnalités à développer ont été réparties sur les 360h que compte un travail de Bachelor. Le travail a été décomposé en 5 *Sprints*. Voici la répartition de ces heures :



La répartition des tâches sur les différents *Sprints* est décrite dans le cahier des charges en annexe. A la fin de chaque *Sprint*, un *Sprint Review* a été effectué avec les acteurs du projet afin d'adapter, si nécessaire, le *Product Backlog* et de planifier le prochain Sprint.

Le *Product Backlog* ne tient pas compte des tâches de recherche ni du temps de formation. Ces tâches exceptionnelles, en plus des *User-Stories,* ont étés ajoutées à chaque *Sprint Planning*[22].

6.2. Outil de suivi du projet

Un tableau de bord [23]a été conçu et mis à disposition des différents acteurs afin de présenter à tout moment un aperçu général de la progression du projet. Afin de faciliter sa consultation il a été mis sur Dropbox.

---

[21] Annexe 5 : Product Backlog
[22] Annexe 6 : Planification des Sprints

6.3. Ecart par rapport à la planification initiale

L'estimation des tâches à réaliser à été sous-évaluée. Le temps de formation sur les nouvelles technologies ainsi que la mise en production de l'application ont pris plus de temps que prévu. De ce fait, notre planification initiale de 360 heures a passé à 417 heures[24]. Cet écart est principalement dû à la mise en production de l'application.

7. Conclusion

Les trois mois passés sur mon travail de bachelor ont élargi mes connaissances en matière de développement mobile. La philosophie de création d'une application mobile multiplateforme est considérablement différente de celle d'un développement de logiciel prévu pour un ordinateur fixe. Les projets mobiles doivent être légers et pourvus d'une interface graphique qui répond à une utilisation tactile. Ils doivent également fonctionner lorsque l'utiliisateur est situé à des endroits dépourvus de réseau.

Le fait de devoir livrer une application qui ne contienne aucun bug puisqu'elle allait être mise en production était également un exercice nouveau pour moi. En effet, les projets scolaires que nous avions réalisés n'ont jamais été mis en production et pouvaient ainsi contenir des bugs.

Pouvoir travailler au sein du team de développement d'Icare a été une expérience très enrichissante et surtout un premier enseignement dans le monde du travail. D'un point de vue personnel, la collaboration et les nombreux échanges que j'ai eu avec les personnes de l'institut Icare m'ont permis d'acquérir de nouvelles connaissances mais surtout de découvrir des façons innovantes de penser et de concevoir des logiciels.

En fin de compte, j'espère que les visiteurs de Vinéa pourront, grâce à ces différentes fonctionnalités développées, profiter pleinement de la manifestation et ainsi déguster de bons cépages et partager leurs avis avec les amis.

---

[23] Annexe 6 : Tableau de bord
[24] Annexe 7 : Décompte des heures

8. Sources

Etat de l'art

Florent Lamoureux (1.02 2012) - Différences entre PhoneGap, Appcelerator Titanium, Sencha Touch et JQuery Mobile

Consulté le 29.05.2012, sur http://www.florentlamoureux.fr/blog/differences-entre-phonegap-appcelerator-titanium-sencha-touch-et-jquery-mobile/

Robert Raiola (30.11.2012) – La montée en puissance des applications hybrides

Consulté le 29.05.2012, sur http://blog.brightcove.com/fr/2011/11/la-montee-en-puissance-des-applications-hybrides

Angelo Buscerni (23.03.2012) – Trois approches pour le développement d'app mobiles

Consulté le 29.05.2012, sur http://www.ictjournal.ch/fr-CH/News/2012/03/20/Trois-approches-pour-le-developpement-dapps-mobiles.aspx

Peter Friese (2012) – Cross-Platform Mobile Development

Consulté le 29.05.2012, sur http://www.slideshare.net/peterfriese/cross-platform-mobile-development-11239246

Tony Lukasavage (20.04.2011) – A deeper look at Appcelerator and PhoneGap

Consulté le 30.05.2012, sur http://savagelook.com/blog/portfolio/a-deeper-look-at-appcelerator-and-phonegap

François Petitit (17.10.2012) – Applications mobile multi-plateformes : les approches PhoneGap et Titanium Mobile

Consulté le 30.05.2012, sur http://blog.octo.com/applications-mobiles-multi-plateformes-les-approches-phonegap-et-titanium-mobile

Gordon L. Hempton (13.01.2012) – The Top 10 Javascript MVC Frameworks Reviewed

Consulté le 31.05.2012, sur http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed

Dave Feldman (23.01.2012) – Comparing Mobile Web Frameworks : Sench TOuch, JQuery Mobile, jQTouch, Titanium

Consulté le 12.06.2012, sur http://operationproject.com/2011/adventures-in-html5-part-one

Christian Simms (27.06.2011) – HTML5 Storage Wars – localStorage vs. IndexedDB vs. Web SQL

Consulté à le 2.06.2012, sur http://csimms.botonomy.com/2011/05/html5-storage-wars-localstorage-vs-indexeddb-vs-web-sql.html

Aaron Brethorst (6.06.2012) – A primer on hybrid apps for iOS

Consulté le 12.06.2012, sur http://www.cocoacontrols.com/posts/a-primer-on-hybrid-apps-for-ios

Paul Krill (19.01.2012) – JQMobi JavaScript framework boats faster mobile apps

Consulté le 4.06.2012, sur http://www.infoworld.com/d/application-development/jqmobi-javascript-framework-boasts-faster-mobile-apps-184476

Test des outils

Documentation de Batman.js

Consulté à partir du 31.05.2012, sur http://batmanjs.org

Documentation de JQTouch

Consulté à partir du 12.06.2012, sur http://www.jqtouch.com

Documentation de Twitter Bootstrap

Consulté à partir du 12.06.2012, sur http://twitter.github.com/bootstrap

Documentation de JQuery Mobile

Consulté à partir du 12.06.2012, sur http://jquerymobile.com

Documentation de Zetpo.js

Consulté à partir du 13.06.2012, sur http://zeptojs.com

Documentation de JqMobi

Consulté à partir du 13.06.2012, sur http://www.jqmobi.com

Documentation et tutoriaux

Documentation de PhoneGap

Consulté à partir du 3.06.2012, sur http://phonegap.com

Documentation Backbone.js

Consulté à partir du 4.06.2012 sur http://backbonejs.org

Net.tutsplus.com (28.04.2012) – Getting Started with Backbone.js

Consulté le 4.06.2012, sur http://net.tutsplus.com/tutorials/javascript-ajax/getting-started-with-backbone-js

Artur Adib – Hello Backbone.js

Consulté à partir du 4.06.2012, sur http://arturadib.com/hello-backbonejs

Sébastien Chopin (15.04.2012) – Backbone.js : le fonctionnement des modèles

Consulté le 4.06.2012, sur http://www.atinux.fr/2012/04/15/backbone-js-le-fonctionnement-des-modeles

Thierry Stiegler (29.10.2010) – Html5 Backbone.js et localStorage sont dans un bateau

Consulté à partir du 8.06.2012, sur http://thierrystiegler.wordpress.com/2010/10/29/html5-backbone-js-et-localstorage-sont-dans-un-bateau

Rico Sta. Cruz (2011-2012) - Backbone patterns

Consulté à partir du 4.06.2012, sur http://ricostacruz.com/backbone-patterns

Documentation OpenLayers

Consulté à partir du 12.06.2012, sur http://docs.openlayers.org

Mobile UI Patterns (2012)

Consulté à partir du 26.06.2012, sur http://mobile-patterns.com

Facebook Developers

Consulté à patir du 16.07.2012, sur https://developers.facebook.com

MicroImages Inc (12.2009) - Bing Maps Structures

Consulté à partir du 12.07.2012, sur

http://www.microimages.com/documentation/TechGuides/76BingStructure.pdf

Klokan Petr Pridal (2008) – Tiles à la Google Maps : Coordinates, Tile Bounds and Projection

Consulté à partir du 12.07.2012 sur http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection

9. Liste des figures

10. Liste des tableaux

11. Annexes

Annexe I : Cahier des charges

Cahier des charges : Application multiplateforme

Présentation générale du projet

Le projet « Application mobile Vinéa » consiste à réaliser une application multiplateforme (compatible iPhone et Android) afin de répondre aux besoins des utilisateurs dans le cadre l'événement viticole sierrois Vinéa. Une première version de cette application a été réalisée en 2009 par l'institut Icare pour la plateforme iPhone uniquement. Aujourd'hui, celle-ci ne répond plus aux besoins du marché et demande à être repensée, adaptée et redéveloppée. Ce projet est proposé par l'institut Icare et sera réalisé au sein de celui-ci. Les fonctionnalités à implémenter seront à définir dans un Product Backlog livrable à la fin du premier sprint (voir chapitre 5 : planning).

L'application sera réalisée à l'aide des technologies Ruby, RubyOnRails, HTML5/ CSS3 et PhoneGap pour la compatibilité multiplateforme. Une analyse concernant les technologies à utiliser sera également réalisée.

Etapes à réaliser (ordre chronologique)

- Identification de la problématique par rapport à l'application existante
- Identification et analyse des besoins en fonction du marché
- Proposition de solution suite aux analyses
  - concept d'application cliente
  - concept de gestion « back end »
- Planification du développement en fonction des choix du client
  - Réalisation des user-stories (attribution de story-points)
  - Identification de l'importance des user-stories
- Analyse et choix des technologies à utiliser (état de l'art) en fonction du Product Backlog
- Conception et développement de l'application mobile multiplateforme
- Conception et développement du backend (administration)
- Réalisation d'un manuel d'utilisateur et administrateur
- En parallèle, rédaction de documents en rapport à la thématique traitée

Organisation du travail

Parties prenantes

Léonard Stalder: Etudiant

Yann Bocchi : Professeur responsable

Pierre Kenzelmann : Responsable du suivi de l'avancement du projet

Livrables

Phase de spécifications et d'analyse des besoins

- Cahier des charges
- Analyse du marché et de l'existant
- Proposition de concept et de positionnement
- Product Backlog (spécifications)
- User-stories
- Evaluation des user-stories

Phase de recherche et test

- Recherche des technologies à utiliser (état de l'art) en fonction du Product Backlog
- Test des technologies et choix

Phase de réalisation

- Sprint-plannings et reviews
- Réévaluation des user-stories
- Etats d'avancements

Phase de livraison

- Application multiplateforme fonctionnelle et
- Code source de l'application

Phase de rédaction

- Rédaction du travail de Bachelor (mise en commun des différentes recherches)

Méthodologie de travail

L'agilité sera utilisée pour gérer au mieux ce projet. Les différents concepts et outils seront utilisés : Uses-cases, user-stories, product backlog, sprint backlog, sprint-review et calcul de la vélocité.

Planning

Le travail de Bachelor compte 360 heures qui doivent être effectuées entre le 15.05.2012 et le 13.08.2012. Le travail sera décomposé en 5 sprints. Voici comment seront réparties ces heures :

Sprint 0 (40 h)

Le sprint 0 comprend 2 semaines à mi-temps au terme duquel un cahier des charges, une analyse du marché et de l'existant, un concept d'application et un Product Backlog avec évaluation des user-stories seront livrables. Les méthodologies agile de gestion de projet seront également mis en place durant ce sprint.

Sprint 1 (60 h)

Le sprint 1 comprend 2 semaines à mi-temps. Une partie de ce sprint sera consacré à la recherche et au test (essayer d'implémenter des fonctionnalités) des technologies à utiliser. L'autre partie sera consacrée à la prise en main de ses technologies. Au terme de ce sprint un document explicatif des technologies à utiliser sera livrable.

Sprint 2 (90 h)

Le sprint 2 comprend 2 semaines à plein-temps. Ce sprint compte la planification du développement ainsi que le début du développement.

Sprint 3 (90 h)

Le sprint 3 comprend 2 semaines à plein-temps. Ce sprint comprend le développement de l'application.

Sprint 4 (90 h)

Le sprint 4 comprend 2 semaines à plein-temps. La première partie du sprint comprend la fin du développement et la résolution des bugs. L'autre partie sera consacrée à la rédaction du travail de Bachelor.

Signatures ou visa

Léonard Stalder _____

Yann Bocchi _____

Pierre Kenzelmann _____

Annexe II : Scénarios d'utilisation

Afin de présenter au client notre proposition d'application, nous avons décidé de détailler les 6 fonctionnalités principales sous forme de scénarios d'utilisation. Chaque scénario sera ensuite composé de différentes User Stories auxquelles nous allons attribuer une priorité et une estimation de temps.

Scenario « Carte »

Screenshot 1 : Se géo-localiser parmi les stands de *Vinéa*

Cette fonctionnalité permet à l'utilisateur de se géo-localiser sur une carte de type Google Maps et d'afficher les stands et les détails d'un stand.

Scenario « Inviter mes amis Facebook à déguster un cépage »

Screenshot 1 : Géo-localiser ses amis Facebook

Cette fonction est une extension de la fonction « carte ». Elle offre à l'utilisateur la possibilité d'afficher ses amis Facebook sur la carte de Vinéa et de les inviter à boire un verre à un stand.

Scenario « Recherche et lister les vins/encaveurs »

Cette fonctionnalité permet d'afficher et de rechercher les vins et les encaveurs présents à la manifestation. L'utilisateur peut également afficher le détail de chaque vin, le noter et l'ajouter à sa liste de favoris ou alors le partager sur Facebook.

Screenshot 1 : lister les vins

Depuis le menu principal, une icône permet à l'utilisateur d'accéder à un écran « Vins et encaveurs ». Les vins sont affichés sous forme de liste.
Les onglets « vins » et « encaveurs » permettent à l'utilisateur de naviguer entre les différentes vues.

Screenshot 2 : afficher le détail d'un vin

En cliquant sur un vin, l'utilisateur va pouvoir afficher le détail d'un vin. Il pourra également le noter et le commenter avant de l'ajouter à sa liste de favoris ou alors le partager sur les réseaux sociaux.

Screenshot 3 : lister les vins d'un encaveur

Sous l'onglet « encaveurs », l'utilisateur peut afficher le détail d'un encaveur et les vins proposés. Il est également possible d'afficher le détail d'un vin par un clic.

Screenshot 4 : Fonction de recherche

Une barre permet à l'utilisateur de saisir un terme recherché concernant les vins et les encaveurs.

Scenario « Sentiers des vins »

Cette fonctionnalité permet aux utilisateurs de suivre des parcours de dégustation proposés par *Vinéa*. Les visiteurs seront ainsi guidés au travers des stands en fonction du parcours choisi. Par exemple, un parcours « Cornalin » proposera à l'utilisateur un itinéraire qui passera par tous les stands qui offrent du Cornalin. Elle permet également aux sponsors de proposer des parcours.

L'utilisateur peut choisir le parcours qu'il veut suivre. Les stands par lesquels il doit passer sont entourés. Une fois que l'utilisateur est a proximité du stand, une fenêtre apparaît et lui propose d'ajouter le vin à ses favoris ou alors de passer au vin suivant.

Scenario « Afficher et gérer les favoris »

Ce processus permet d'afficher et de gérer les vins favoris. Ils peuvent être ajoutés depuis la liste des vins et encaveurs décrite dans le scénario précédant. Une fenêtre permet d'exporter ses favoris en les envoyant à une adresse mail.

Le bouton « partager sur Facebook » permet de publier sur le mur de l'utilisateur sa liste des favoris. Une demande d'autorisation d'accéder à ses informations Facebook sera effectuée la première fois qu'il utilisera cette fonctionnalité.

Scenario « Plus »

Cette partie de l'application permet à l'utilisateur d'afficher des informations relatives à l'événement et de configurer certains paramètres.

Annexe III : Tableau comparatif des Frameworks graphiques

	Frameworks complets				Frameworks combinés		
	The M project	**Jquery Mobile**	**JQMobi et UI**	**JQtouch**	**Zepto.js avec CSS3**	**Twitter boostrap avec JQuery**	**Jquery avec Css3**
Licence	Gratuit	Gratuit	Gratuit	Gratuit	Gratuit	Gratuit	Gratuit
Taille de la librairie de manipulation du DOM	66 kB	36 ko	3.7 ko		7.8 ko	36 ko	36 ko
Syntaxe	Propriétaire	JQuery	JQuery	Propriétaire	JQuery	JQuery	JQuery
Librairie UI js/css		62 ko	14 ko		Dépend du CSS	14ko	Dépend du CSS
Supports	All browsers	All browsers	Webkit	Webkit	Webkit	All browsers	Dépend du css
Editeur	PanaCoda	JQuery	Appmobi	JQtouch	Zepto	Twitter	JQuery
Rapidité exécution	Très lent	Animations lentes. Réactivité faible. Lourd par le fait que JQuery est cross-browser	3 fois plus rapide que JQuery (sous Android) 5.5 fois plus rapide que Zepto (sous Android)	HTML généré à partir du javaScript. Basé sur JQuery	Bonne combinaison, mais CSS développé à la main. Zepto est moins rapide que Jquery mais plus léger.	Possède une syntaxe particulière. Le design n'est pas adapté pour mobile mais plutôt pour des applications web normales	Bonne combinaison, mais CSS développé à la main. Jquery est plus rapide que Zepto.js
Documentation	Riche	Riche	Très limitée	Limitée	Riche	Riche	Riche
Existe depuis		2006	2012	2010	2010	2010	2006
Téléchargements depuis GIT HUB	558	6552	850	2299	3553	6552	6552

Annexe IV : Accès au téléchargement de l'application

L'application iPhone et iPad peut être téléchargée à partir du lien suivant :

http://ota.icare.ch/vinea2012/

L'application Android peut être téléchargée à partir du 20 août sur Google Play

Annexe V : Product Backlog

US Nr.	Acteur	Tâche	Priorité	Statut	Story Points	Sprint
1	Utilisateur	Afficher le menu d'accueil de l'application	1	OK	1	1
2	Utilisateur	Pouvoir naviguer dans les différentes pages	1	OK	2	1
3	Utilisateur	lister les vins	1	OK	4	1
4	Utilisateur	lister les caves	1	OK	1	1
5	Utilisateur	rechercher un vin ou une cave	1	OK	1	1
6	Utilisateur	afficher le détail d'un vin	1	OK	1	1
7	Utilisateur	ajouter un vin à ses favoris	1	OK	2	1
8	Utilisateur	noter un vin avant de l'ajouter aux favoris	1	OK	1	2
9	Utilisateur	commenter un vin avant de l'ajouter aux favoris	1	OK	1	2
10	Utilisateur	Afficher ma liste des favoris	1	OK	4	1
11	Utilisateur	Sélectionner un vin favori	1	OK	1	2
12	Utilisateur	Modifier mon évaluation d'un vin favori	1	OK	1	2
13	Utilisateur	exporter ma liste de favoris par mail	1	OK	2	2
14	Utilisateur	partager le vin dégusté sur les réseaux sociaux	3	OK	2	3
15	Utilisateur	afficher la carte de l'Avenue Général Guisan	2	OK	8	1
16	Utilisateur	afficher les stands de Vinéa sur la carte	2	OK	6	2
17	Utilisateur	afficher ma position sur la carte	2	OK	4	2
18	Utilisateur	afficher le détail d'un stand en cliquant dessus	2	OK	3	2
19	Utilisateur	Lier mon compte Facebook	3	OK	4	3
20	Utilisateur	Afficher mes amis Facebook présents à Vinéa sur la carte	3	NON	8	3
23	Utilisateur	Inviter un ami Facebook à un boire un verre depuis la carte	3	NON	9	3
27	Utilisateur	Afficher les informations générales sur Vinéa	2	OK	0.5	1
28	Utilisateur	Afficher une page "à propos"	2	OK	0.5	3
29	Utilisateur	Afficher l'aide de l'application	2	OK	0.5	3
30	Utilisateur	Afficher le programme de la manifestation	2	OK	0.5	3
31	Utilisateur	Afficher les animations de la manifestation	2	OK	0.5	3
32	Utilisateur	lister les parcours qui me sont proposés	3	OK	3	2
33	Utilisateur	Lister les vins d'un parcours	3	OK	2	2
34	Utilisateur	Sélectionner un parcours proposé	3	OK	4	2
35	Utilisateur	Géo-localiser le vin depuis le menu "détail d'un vin"	3	OK	4	2
36	Utilisateur	Partager un parcours sur les réseaux sociaux	2	NON	2	3
37	Utilisateur	Afficher des bandeaux publicitaires	2	OK	2	4
38	Utilisateur	Implémentation de la couche graphique	2	OK	8	4
39	Utilisateur	Navigation et gestion de l'historique	1	OK	2	4

Annexe VI : Planification des Sprints

Sprint 1 : 60 h planifiées 28.05.12 - 16.06.12			
US	**Tâche**	**Story point**	**Terminé?**
	Etat de l'art - test des outils de développement	10	oui
	Choix des technologies	2	oui
	Prise en main de la technologie Backbone et JS	5	oui
1	Afficher le menu d'accueil de l'application	1	oui
2	Pouvoir naviguer dans les différentes pages	2	oui
3	lister les vins	4	à moitié
4	lister les caves	1	oui
5	rechercher un vin ou une cave	1	à moitié
6	afficher le détail d'un vin	1	oui
7	ajouter un vin à ses favoris	1	à moitié
15	afficher la carte de l'Avenue Général Guisan	8	presque
27	Afficher les informations générales sur Vinéa	2	oui
10	Afficher ma liste des favoris	4	à moitié
Total		42	

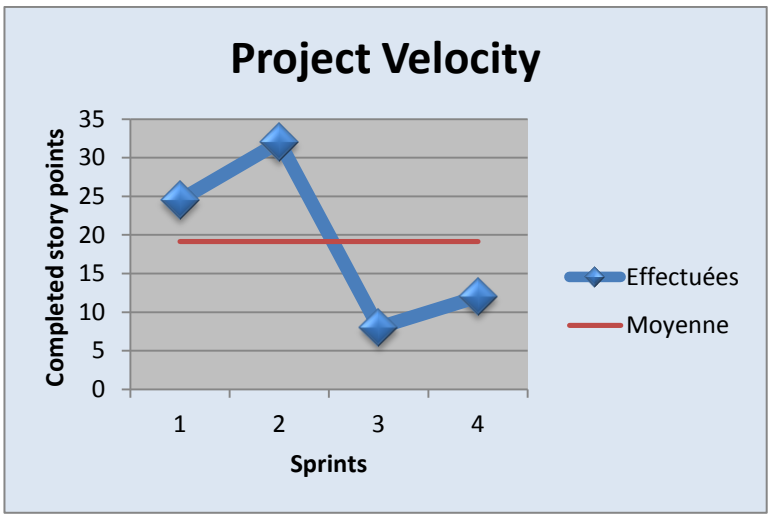
Sprint 2 : 120 h planifiées 16.06.12 - 16.07.12			
US	**Tâche**	**Story point**	**Terminé?**
	Création du schéma de base de données	1	oui
	Mise en place de la base de données local	2	oui
	Mise en place sync des données	3	en cours
	Mise en place du webservice distant	2	en cours
	Liaison entre les models et la base de données	3	oui
	Mise en place du FrameWork UI choisi	2	oui
	Test des différents Frameworks UI et MVC	10	oui
7	ajouter un vin à ses favoris (sprint 1)	0.5	oui
3	lister les vins (sprint 1)	1.5	oui
10	Afficher ma liste des favoris (sprint 1)	1.5	oui
5	rechercher un vin ou une cave (sprint 1)	0.5	oui
15	afficher la carte de l'Avenue Général Guisan (sprint1)	1	en cours
32	lister les parcours qui me sont proposés	3	oui
16	afficher les stands sur la carte	6	oui
18	afficher le détail d'un stand en cliquant dessus	3	oui
34	Sélectionner un parcours proposé	4	oui
	Mise en place des images fournies par Vinéa	3	en cours

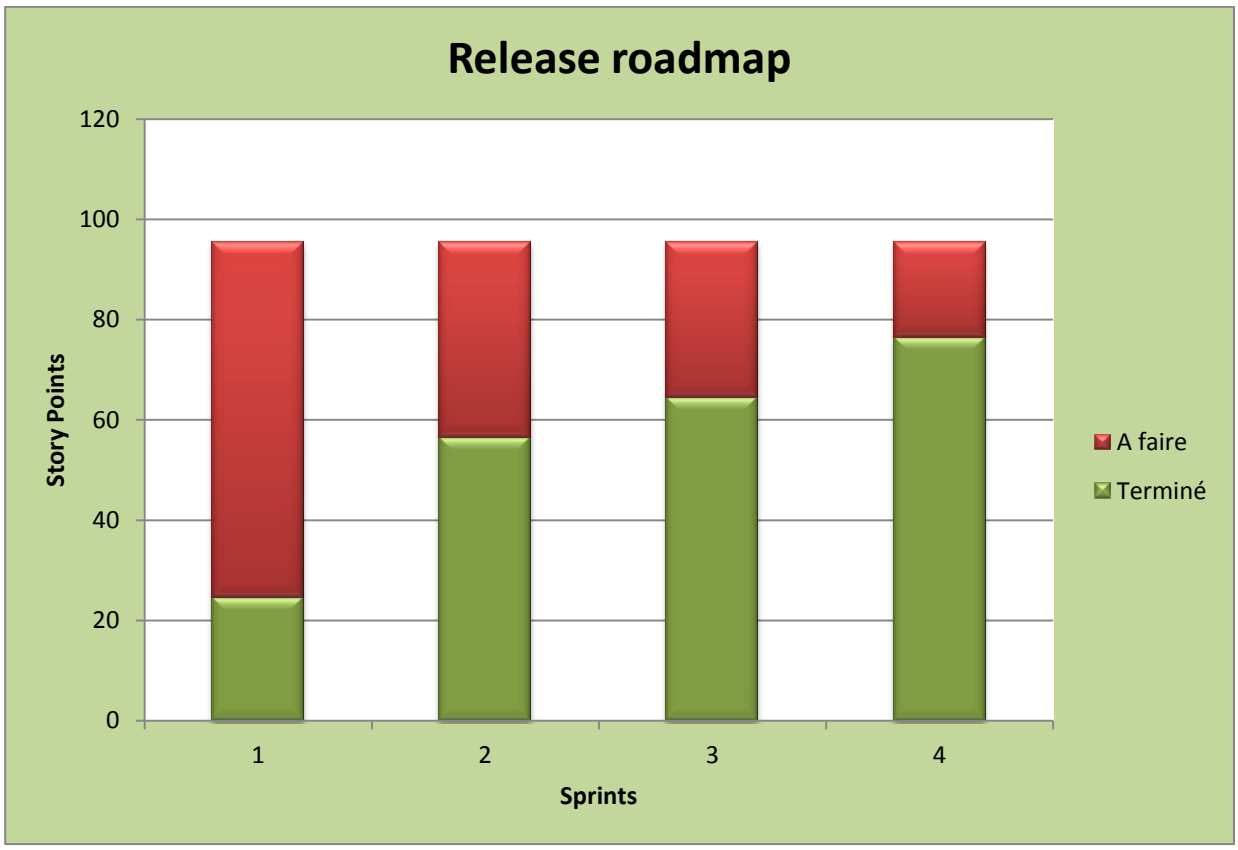
	Tester l'application sous iOS	1	oui
33	Lister les vins d'un parcours	2	oui
17	afficher ma position sur la carte	4	oui
	activer / désactiver le traçage GPS	2	oui
12	Modifier mon évaluation d'un vin favori	1	oui
11	Sélectionner un vin favori	1	oui
13	exporter ma liste de favoris par mail	2	oui
35	Géo-localiser le vin depuis le menu "détail d'un vin"	4	oui
Total		64	

Sprint 3 : 90 h planifiées 16.07.12 - 30.07.12			
US No	**Tâche**	**Story point**	**Terminé?**
	Mise en place sync des données (sprint 2)	2	oui
	Mise en place du webservice distant (sprint 2)	2	oui
15	afficher la carte de l'Avenue Général Guisan (sprint 2)	1	en cours
	Mise en place des images fournies par Vinéa	3	oui
	Bugs avec géo-localisation	2	oui
14	partager le vin dégusté sur les réseaux sociaux	2	en cours
19	Lier mon compte Facebook	4	oui
20	Afficher mes amis Facebook présents à Vinéa sur la carte	8	Abandonné
23	Inviter un ami Facebook à un boire un verre depuis la carte	9	non
29	Afficher l'aide de l'application	0.5	en cours
30	Afficher le programme de la manifestation	0.5	oui
36	Partager un parcours sur les réseaux sociaux	2	non
37	Afficher des bandeaux publicitaires	2	non
	Range de localisation : prévoir une localisation uniquement à l'intérieur d'un range	1	oui
	Ajouter les stands sur la carte	2	oui
Total		41	

Sprint 2 : 90 h planifiées 30.07.12 - 13.08.12			
US	**Tâche**	**Story point**	**Terminé?**
	partager le vin dégusté sur les réseaux sociaux (sprint 3)	2	oui
	Afficher l'aide de l'application (sprint 3)	2	oui
	Afficher carte en off-line (sprint 3)	4	oui
	Page "a propos"	1	oui
	Bugs avec géo-localisation	2	oui
	Système de pubs	2	oui
	Gestion de l'historique de navigation	2	oui
	Intégration des pages Print to Mobile	4	oui
	Réduire temps de chargement de l'application	5	oui
Total		24	

Annexe VII : Tableau de bord





Annexe VIII : Décompte des heures

	Date	Tâche	Heures
SPRINT 0	15.05.2012	Séance de début de projet	1
	15.05.2012	Rédaction du cahier des charges	5
	16.05.2012	Prise en main de l'application et recherche de l'existant	5
	21.05.2012	Mise en place du sondage et récolte d'informations ainsi que recherche de fonctionnalités	3.5
		Rédaction d'un document expliquant les fonctionnalités	4.5
	22.05.2012	Dépouillage du sondage, ajustement du document "concept d'application"	2.5
	23.05.2012	Création de prototypes écrans	5
	24.05.2012	Ajustement du document "concept d'application"	3
		création des supports de gestion de projet et planification des sprints	3
	25.05.2012	Relecture et rédaction Product Backlog	4
SPRINT 1	29.05.2012	Analyse état de l'art -> test des différentes plateformes de développement à utiliser	4.5
	30.05.2012	Etat de l'art-> choix de l'approche de développement mobile	6.5
	31.05.2012	Etat de l'art-> Test des différents Frameworks pour l'architecture JavaScript	8
	01.06.2012	Test -> Mise en place d'Android, PhoneGap et du Framework Backbone.js	4
	04.06.2012	Prise en main de BackBone.js (FrameWork)	8
	06.06.2012	Début du développement de l'architecture de l'application avec Backbone	5
	08.06.2012	Mise en place du MVC avec Backbone	6.5
	11.06.2012	Mise en place du MVC avec Backbone	7
	12.06.2012	Test de JQMobi et autres frameworks Ui	7
	13.06.2012	Test d'OpenLayers	6
	14.06.2012	Mise en place MVC pour les vins et Openlayers	7
	15.06.2012	Rédaction document technologies + Openlayers	5
SPRINT 2	18.06.2012	Séance de sprint review et planification du prochain sprint avec Pierre	1
		Mise en place favoris et détail de vins et possibilité d'ajouter aux favoris	6
	19.06.2012	Planification second sprint	1.5
	20.06.2012	Test des différents Frameworks UI notamment JqMobi	7
	21.06.2012	Mise en place du framework UI JqMobi sur notre projet actuel	5
	22.06.2012	Mise en place de la navigation et la gestion des transitions. Création du menu de navigation	9
	27.06.2012	Abandon de JqMobi pas assez stable. Adaptation de la couche graphique avec CSS3	3

	28.06.2012	Adaptation de la couche graphique avec CSS3 et IScroll4	7
	02.07.2012	Résolution de bugs graphiques et de navigation	4
	03.07.2012	Séance avec Pierre pour la couche graphique et la mise à jour des données	1
		Création du schéma de base de données et mise en place de la base de données locale	5
	04.07.2012	Liaison de la base de données avec la logique backbone.	2
		Mise en place d'objets qui lisent et écrivent en base de données	5
	05.07.2012	Création d'un webservice pour les tests qui renvoie la liste des caves	4
		Mise en place de la synchronisation des données en local	3
	06.07.2012	Synchronisation des données et mise à jour de la liste des vins automatisée	6
		Création des fonctionnalités "favoris" et "recherche"	3
	09.07.2012	Réflexion et rédaction des données à retourner par le webservice définitif	4
		Séance avec David (il va réaliser le backend en Rails) Explication des données dont j'ai besoin	1
	10.07.2012	Mise en place de la synchronisation des données pour les caves, les stands, les parcours	4
		Création d'un mini-webservice test pour les données ci-dessus.	2
	11.07.2012	Création et modification fonctionnalités "ajouter des vins favoris", "lister vins"	4
		Fonctionnalité carte : ajout des stands sur la carte et sélection d'un stand	4
		Possibilité de sélectionner un vin favori	1
	12.07.2012	"Création de la fonctionnalité "Liste des parcours" et la possibilité d'en choisir un	4
		Fonctionnalité carte : ajout de la géo-localisation et terminer "sélection d'un stand"	4.5
		Ajout de stands tests sur la carte	1
	13.07.2012	Création de la fonctionnalité de détail d'un stand.	2
		Fonctionnalité favoris : Possibilité d'exporter par mail	1
		Fonctionnalité carte : Possibilité d'activer/désactiver le traçage GPS	4
	14.07.2012	Fonctionnalité carte : Possibilité de localiser un vin depuis le détail d'un stand ou vin	5
	16.07.2012	Test de l'application sous iOS	1
		Implémentation de la fonctionnalité "lister les vins d'un parcours"	3
		Correction de bugs	2
		planification du troisième sprint	1
		Séance avec Pierre pour la couche graphique et la mise à jour des données	1

SPRINT 3	17.07.2012	Mise en place JSON embarqué pour la première utilisation -> si pas de connexion internet	7
		Mise à jour des données du JSON embarqué -> liens morts, clés étrangères manquantes	2
	18.07.2012	Résolution de bugs liés à la géo localisation et la carte	4.5
		Mise à jour des stands sur la carte - Possibilité de différencier les stands	4.5
	19.07.2012	Mise en place du plugin Facebook connect	5
		Mise en place du partage sur Facebook	2.5
		Création d'une application Facebook avec une landing page pour celle-ci	2
	22.07.2012	Configuration de l'application Facebook	1.5
		Hashage de l'url pour le partage des vins sur Facebook	2
		Création d'une landing page pour les vins partagés sur Facebook	2
	23.07.2012	Terminer le partage Facebook	2.5
		Création des pages "aide" et "a propos"	5
		Résolution de bugs	1
	24.07.2012	Résolution de bugs : mise en place d'un système de versionning pour les données	4
		Dépouillage des données : Récupération des long. / lat. des stands	3.5
	25.07.2012	Mise à jour des stands sur le web-service et des informations	2
		Mise en place "range" de localisation	1
		Intégration des pages "print to mobile" à l'application	4
	26.07.2012	Création du menu dynamique pour les pages "print to mobile"	2
		Modification des stands	1
		Rédaction du document	5
	27.07.2012	Mise en place de la map Off-line : téléchargement des tiles et modification de la librairie OL	7.5
	28.07.2012	Terminer la map off-line : modification des png et transition des notations	4
	29.07.2012	Terminer la map off-line : ajout des stands	3.5
SPRINT 4	30.07.2012	Modification des tiles avec ceux de Bing -> problème des niveaux de zoom	4
		Rédaction du document	3
	31.07.2012	Gestion du bouton de l'historique de navigation : supporté sur iOs et Android	4
		Modification des bugs quant aux pages print to mobile	3
	02.08.2012	Rédaction du rapport	3
		Optimisation du code l'initialisation de l'application	4
	03.08.2012	Possibilité de gérer le menu du programme dynamiquement	2
		Optimisation du code de fermeture des vues. But : s'assurer que la mémoire s'est bien vidée	2

		Rédaction du document	3
	04.08.2012	Modification de l'emplacement de certains stands	2
		Rédaction du document	4
	06.08.2012	Implémentation de la couche graphique - ajout d'icônes haut-résolution	4
		Modification du CSS en fonction des idées d'un graphiste	4
		Rédaction du document	3
	07.08.2012	Rédaction du document	3
		Implémentation d'un système de gestion des publicités	4
		Optimisation du code	2
	08.08.2012	Optimisation du code pour la mise en production	4
		Rédaction du document	6
	09.08.2012	Correction de bugs en fonction des retours de tests fonctionnels	4
		Rédaction du document	6
	10.08.2012	Modification des certains icones et minimisation du JavaScript	2
		Rédaction et relecture du document	9
	11.08.2012	Modification du document	9
	12.08.2012	Modification et impression du document	11
Total des heures			417