

Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence « Développer une application de mobilité
numérique »

**Développement mobile avec
Cordova / PhoneGap**

Apprentissage

Mise en pratique

Evaluation



APACHE
CORDOVA™

1. PRESENTATION DE CORDOVA

1.1 APPLICATIONS MOBILES

Actuellement, **les sites Web sont accédés majoritairement depuis des terminaux mobiles** (smartphones ou tablettes; voir <http://gs.statcounter.com>) et **les tailles et résolutions d'écrans des périphériques fixes ou mobiles sont de plus en plus variés et de moins en moins standards.**

Les techniques Web associées au Responsive Design permettent une certaine auto-adaptation des pages Web aux caractéristiques physiques d'un mobile mais le design et l'ergonomie de l'application restent tout de même conçus '*façon Web*'.

Les terminaux mobiles disposent de fonctionnalités qui ne sont pas, aujourd'hui, toutes exploitables par les techniques Web (téléphonie, gestion des contacts...).

Une application mobile à destination du grand public doit correspondre aux critères de design et d'ergonomie de chacune des plateformes. Si un seul design '*passer-partout*' peut parfois suffire, le succès de l'application grand public est bien souvent lié à son intégration avec le '*look and feel*' du mobile. En conséquence, **il est bien souvent nécessaire de concevoir design et ergonomie pour chacune des plateformes ciblées.**

En raison de la diversité des technologies adoptées par les constructeurs d'appareils mobiles, **une application spécifique pour mobile doit être développée dans différents langages et technologies** (iOS, Android, Windows Phone... langages C#, Java... et Frameworks spécifiques...) et doit être déposée sur des **plateformes de téléchargement coûteuses** (Google Play, App Store...) si on souhaite s'adresser au grand public.

Bien entendu, **ces développements multiples ne concernent que la partie cliente de l'application**, mais cela représente en général une grande partie de l'effort de développement.

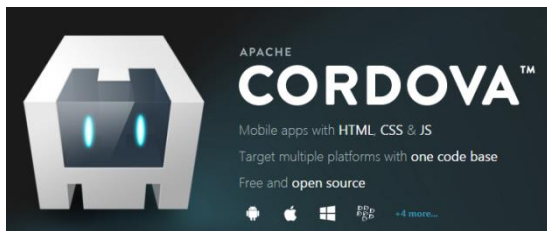
Face à cette complexité de développement, certains éditeurs cherchent des **solutions techniques multiplateformes (encore appelées '*cross-platforms*')**. Actuellement, deux solutions commencent à être largement utilisées :

- **Xamarin**, qui s'adresse plutôt aux développeurs en technologies **Microsoft .Net** car il repose sur le **langage C#** ;
- **Cordova/PhoneGap**, qui s'adresse plutôt aux **développeurs Web** car il repose sur les langages standards du Web (**HTML/CSS et JavaScript**).

1.2 CORDOVA/PHONEGAP

A l'origine, **Cordova** s'appelait **PhoneGap**, c'est la société **Nitobi Software** qui a développé ce **Framework** en **2009**. **Nitobi** a été racheté par **Adobe** en **2011**, puis Adobe a cédé PhoneGap au monde libre (Fondation **Apache**) sous le nom de **Cordova**.

Mais **Adobe** continue quant à lui à faire évoluer sa propre version de **PhoneGap**; d'où la confusion fréquente entre les 2 noms de produits. Pour simplifier, **PhoneGap** et **Cordova**, c'est '*blanc bonnet et bonnet blanc*' !



La solution **Apache Cordova** part d'un principe tout simple : **tout appareil mobile contient un navigateur capable de jouer des pages HTML/CSS/JavaScript qui couvrent déjà l'essentiel des besoins ; il 'suffit' donc d'étendre les capacités standards de JavaScript afin de pouvoir**

utiliser les fonctionnalités spécifiques des mobiles (téléphonie, gestion des contacts...) et de présenter le tout sous forme d'icônes de lancement intégrées aux autres applications du mobile. C'est ce que fait Cordova !

Avec **Cordova/PhoneGap**, le design est réalisé en **HTML/CSS** et **toute la logique est programmée en JavaScript**. Le Framework Cordova/PhoneGap fournit des **bibliothèques JavaScript permettant de faire le lien en JavaScript entre la partie purement Web de l'application et les fonctionnalités spécifiques mobiles** non couvertes en standard ; dans le jargon Cordova/PhoneGap, on parle de '**plugins**' ; il en existe des centaines couvrant tous les besoins. Ces plugins sont disponibles (pas tous, pas toujours) pour les 3 plateformes mobiles standards mais aussi pour d'autres types de mobiles comme les BlackBerry.

A la compilation, le développeur choisit la plateforme cible (Android, Windows, iOS...) et le 'compilateur' Cordova/PhoneGap s'occupe du reste : réaliser les liens avec les modules d'accès aux fonctions natives du mobile, empaqueter les composants nécessaires sous forme d'une application native et créer une icône de lancement. Ainsi, 1 seul code et 3 compilations suffisent à générer une application pour les 3 plateformes standards avec Cordova/PhoneGap, à condition là encore que le développement utilise des fonctionnalités communes et une ergonomie standard.

Pour avoir un aperçu de la manière de développer avec Cordova/PhoneGap, visionnez la vidéo en ligne suivante sans chercher à mémoriser les manipulations mais en vous focalisant sur le processus (car vos manipulations à travers **Visual Studio Code** seront nettement différentes et il est fait usage de Frameworks supplémentaires dans cette vidéo) :



<http://www.grafikart.fr/tutoriels/cordova/cordova-angular-454> (environ 1h 8mn).

1.3 VISUAL STUDIO ET LES EXTENSIONS APACHE CORDOVA

Le Framework Cordova/PhoneGap est disponible sous diverses formes :

- Framework **Adobe PhoneGap Build** ;
- A intégrer et configurer dans un IDE comme Eclipse ; il faut alors adjoindre et configurer les émulateurs nécessaires à la mise au point ; tout ce paramétrage du poste de développement reste assez périlleux ;
- **Déjà totalement intégré à Visual Studio (2013 et plus) sous le nom Visual Studio Tools for Apache Cordova** ; dans ce cas il ne reste qu'à choisir et configurer les émulateurs pour tester et compiler à destination des différentes plateformes mobiles :
 - Pour les cibles **Android**, on peut aussi bien tester à travers **l'émulateur Web Ripple**, un **émulateur Visual Studio Microsoft** ou **l'émulateur Android Google** ;
 - Pour les cibles **Apple**, le développeur **doit connecter une machine Apple** équipée de l'IDE XCode, ou il peut aussi installer un simulateur iOS pour Windows équipé d'un émulateur iPhone/iPad ;
 - Pour les cibles **Windows**, **tout est plus simple** car Visual Studio est conçu à l'origine pour des développements d'applications Windows ;
 - **Dans tous les cas, le développeur peut connecter un appareil mobile via un port USB. C'est cette solution que nous retiendrons car les émulateurs proposés sont gourmands en ressource mémoire et peuvent ralentir l'exécution de vos programmes.**

NB : Le Framework **Cordova** permet de cibler encore d'autres types de mobiles, en ce sens qu'il propose des modules d'interface vers des fonctionnalités natives pour d'autres plateformes.

Il y a plusieurs façons pour utiliser le **Framework Apache/Cordova**.

La première solution que vous verrez en suivant le TP : [A La Découverte des Frameworks JQuery Mobile et Cordova](#) consiste à installer, d'une part, le **PhoneGap Desktop** sur votre station de travail et d'autre part, l'installation du logiciel **PhoneGap Developer** sur votre smartphone.

La deuxième solution consiste à utiliser le package **npm CLI PhoneGap** qui vous permettra de créer et d'enrichir votre application **Cordova** en **ligne de commande**.

Pour cela, vous pourrez lire le document : [Présentation_Cordova.pdf](#) qui vous présentera les différentes lignes de commande pour **créer** un projet, pour le **modifier**, pour **cibler** une plateforme donnée (**Android**, **iOS**, **BlackBerry**, ...), pour **installer** les **plugins** (Interface **JavaScript** qui permet de communiquer avec les composants natifs de l'appareil mobile) nécessaires à votre application, pour **compiler** votre projet ... Il vous présentera aussi les plugins les plus utilisés.

Vous pourrez aussi trouver un complément d'informations en vous rendant sur le site : <https://cordova.apache.org/docs/fr/latest/guide/cli/index.html>

1.4 STRUCTURE D'UN PROJET APACHE CORDOVA






Introduction

Apache Cordova permet de créer des **applications mobiles hybrides**. Ces applications sont développées en utilisant les langages du Web et compilées pour un ou plusieurs environnements mobiles.

La création d'un projet Apache Cordova via la commande « **cordova create** » entraîne la création d'un répertoire qui porte le nom du projet. Celui-ci contient tout une arborescence et un ensemble de fichiers qui sont utilisés par Apache Cordova pour créer l'application mobile hybride.

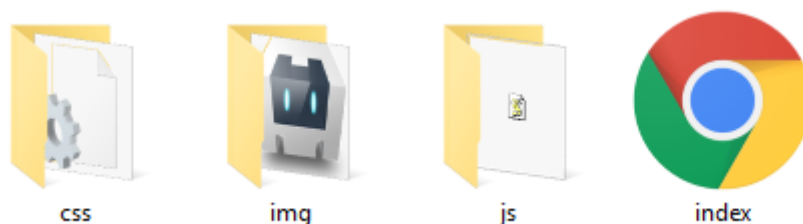
Structure d'un projet Cordova

La création d'un nouveau projet Apache Cordova va donner naissance à une arborescence dont la racine est le répertoire qui porte le nom du projet. Celui-ci à son tour contient initialement quatre sous-répertoire : **www**, **plugins**, **platforms** et **hooks**. Egalement dans ce répertoire racine on va trouver un fichier intitulé « **config.xml** » qui permet de configurer l'application mobile.

 hooks	15/08/2016 13:05	Dossier de fichiers	
 platforms	16/08/2016 10:15	Dossier de fichiers	
 plugins	16/08/2016 10:15	Dossier de fichiers	
 www	16/08/2016 18:00	Dossier de fichiers	
 config.xml	15/08/2016 13:05	Document XML	1 Ko

Répertoire www

Le développement de l'application mobile se fait essentiellement au sein du répertoire « **www** ». En effet, celui-ci est destiné à accueillir le **code source** de l'application. Il contient par défaut trois sous répertoires : **css**, **img** et **js**. En plus, il contient le fichier « **index.html** » qui est par défaut le point d'entrée de l'application. En effet, c'est lui qui sera exécuté par la **WebView** lorsque l'application sera lancée.



Le répertoire « **js** » est destiné à accueillir tous les fichiers JavaScript relatifs à l'application mobile. Par défaut, il contient le fichier « **index.js** » qui est invoqué dans le fichier « **index.html** ».

Le répertoire « **img** » est destiné à accueillir toutes les images utilisées dans l'application mobile. Par défaut, il contient le fichier « **logo.png** » qui représente le logo d'apache cordova utilisé dans l'application créée par défaut lors de la création d'un nouveau projet.

Le répertoire « **css** » est destiné à accueillir tous les fichiers de style css. Il contient par défaut le fichier « **index.css** » utilisé dans le fichier « **index.html** » de l'application générée par défaut. Toutes les feuilles de styles de l'application mobile hybride doivent résider dans ce répertoire.

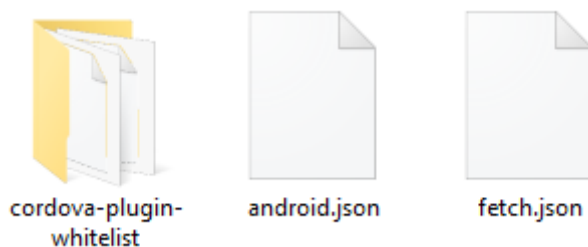
Respecter cette arborescence permet d'avoir un code source organisé et facile à maintenir. L'emplacement des fichiers est ainsi standardisé ainsi que les noms des fichiers principaux (index.html, index.js et index.css).

D'autres répertoires et fichiers peuvent être créés et ajoutés par la suite par le développeur. C'est le cas du répertoire « **res** » par exemple qui va contenir les **icônes** et les **splash screens**.

C'est également le cas des fichiers de certains **Frameworks** ou autres **bibliothèques** qui peuvent être utilisés dans le projet à l'instar des fichiers de « **JQuery Mobile** ».

Répertoire plugins

Pour pouvoir profiter pleinement des fonctionnalités natives d'un dispositif mobile il faut utiliser des **plugins**. Selon l'application mobile hybride il se peut que le développeur ait besoin d'un plugin pour utiliser une fonctionnalité supportée par celui-ci. A ce moment il doit utiliser la commande « **cordova plugin add** » suivie par le nom du **plugin**. L'exécution de cette commande va permettre l'ajout du plugin au projet en cours dans le répertoire réservé aux plugins intitulé « **plugins** ».



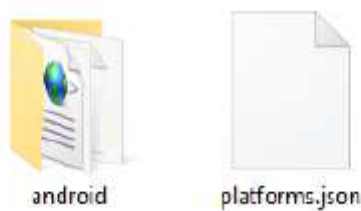
Le développeur ne doit pas toucher au contenu du répertoire **plugins**. En effet, la gestion des plugins (tels que l'**ajout** ou la **suppression**) doit se faire à travers les **commandes cordova**.

Répertoire platforms

Une **application mobile hybride** a la particularité de pouvoir être compilée pour plusieurs plateformes mobiles à partir d'un même code source. Toutefois, pour pouvoir compiler l'application pour un environnement mobile particulier il faut tout d'abord installer son **SDK**.

Puis, il faut ajouter le ou les plateformes ciblées via la commande « **cordova platform add** ».

Celle-ci va permettre d'ajouter les fichiers nécessaires pour pouvoir compiler l'application pour l'environnement souhaité. Ces fichiers résident dans un répertoire qui porte le nom de l'environnement sous le répertoire « **platforms** ».



Ce répertoire peut être utile au développeur dans plusieurs cas. Les deux cas les plus fréquents sont la récupération de l'exécutable et la récupération du code source.

Pour le premier cas, si le développeur souhaite récupérer l'exécutable (fichier d'extension **APK** pour le cas de la plateforme **Android**) pour le tester, pour le partager ou pour le publier.

Le deuxième cas c'est lorsqu'il n'est pas possible de compiler le code source sur la même machine. C'est le cas si le développement s'effectue sur un **PC** et que la plateforme cible est « **IOS** ». Dans ce cas, il faut récupérer le code source et le compiler sur un ordinateur de la marque **Apple** avec le système d'exploitation « **Mac OS X** » avec l'environnement de développement « **XCode** ».

Répertoire hooks

Ce répertoire est destiné à accueillir des scripts « **hooks** ». Il s'agit de scripts qui sont développés par les développeurs des applications mobiles ou des développeurs de plugins pour personnaliser les commandes **cordova**.

Par exemple, vous pouvez définir un « **hook** » qui permet de vérifier systématiquement le formatage du code dans vos fichiers JavaScript avant chaque appel à la commande « **cordova build** » pour générer l'application.

Pour automatiser l'exécution de ce script il suffit d'indiquer son type (Exemples : **before_build**, **after_build**, **before_run** et **after_run**).

Cependant, l'usage du répertoire « **hooks** » pour créer ce type de script est devenu obsolète du moment qu'il est possible de faire la même chose dans les fichiers « **config.xml** » et « **plugin.xml** ». Il existe pour des raisons de compatibilité avec des anciens plugins qui utilisent encore ce répertoire.

1.5 CONCLUSION

Le développement d'applications mobiles devient une tâche récurrente pour une multitude de plateformes. Trois solutions, plus ou moins complexes se présentent pour les environnements mobiles :

1. Une **application native** est développée pour **une seule plateforme** avec des outils qui lui sont propres. Le langage de développement est spécifique au système d'exploitation. La distribution et la mise à jour se fait par l'intermédiaire d'un store dédié.

2. Une **WebApp** est développée avec les technologies web **HTML5**, **CSS3** et **JavaScript**. L'application est distribuée via un **serveur web** et est **exécutée** via le **navigateur web du mobile**. Contrairement aux applications natives, une WebApp est développée une seule fois pour être exécutée sur n'importe quelle plateforme mobile via Internet. C'est une économie importante de temps et du coût de développement.

L'inconvénient majeur toujours mis en avant est **l'absence** d'accès aux **fonctions natives** du **mobile**, ce qui réduit le champ fonctionnel d'une WebApp.

3. Une **application hybride** est un mix des deux premières solutions. Le développement combine des éléments web sous forme de **WebApp** et des éléments de l'application native en compilant une exécutable compatible avec le système d'exploitation. Les plateformes **PhoneGap** et **Cordova** permettent de créer une application indépendante à partir de **pages web** et l'utilisation des **fonctions natives de l'appareil mobile**. L'approche hybride permet de mutualiser le développement sur plusieurs systèmes d'exploitation. Développer en hybride contribue à l'optimisation du temps et du coût de développement.