

LIVRE BLANC

Le Web Mobile
2015

#viseospirit

WISEO
www.viseo.com

Sommaire

1	Introduction.....	3
2	Pourquoi le Web Mobile ?.....	4
2.1	L'approche « Native ».....	5
2.2	L'approche « Web Mobile ».....	8
2.3	L'approche « Web Mobile Hybride »	11
2.4	En somme, pourquoi le Web mobile ?	14
2.5	Arbre décisionnel.....	14
	Avant 2014	15
	A partir de 2014.....	16
3	Ce que fait / ne fait pas le Web Mobile	17
4	Fonctionnalités par plateforme mobile.....	18
5	Statistiques sur les OS mobile	22
5.1	Sur l'ensemble des smartphones	22
5.2	Pour Android	23
5.3	Pour iOS	24
5.4	Pour BlackBerry	25
6	Contraintes courantes liées a la mobilité	26
7	Ergonomie.....	29
7.1	Proposer une gamme limitée de fonctionnalités	29
7.2	Des icônes / éléments adaptés à nos doigts	30
7.3	La densité des écrans	33
7.4	Les zones accessibles.....	34
7.5	Les gestes usuels.....	35
7.6	Adapter la conception de vos écrans.....	36
8	Modèle d'architecture mobile	39

8.1	Le MultiPage versus le SinglePage	39
8.2	Architecture Web mobile avec une technologie serveur	41
8.3	Architecture Web mobile avec une technologie cliente	42
8.4	Architecture Web mobile avec une technologie cliente et serveur.....	43
9	FrameworkS additionnels	44
9.1	Less / Sass / Compass	44
9.2	MVC / MVVM.....	45
10	Outillage.....	49
11	Conclusion	52
12	A propos de VISEO	53
13	VISEO – Organisme de Formation	54

1 INTRODUCTION

Le livre blanc présent a plusieurs objectifs :

- Vous exposer les différents moyens de concevoir des applications mobiles,
- Vous expliquer les avantages de la solution dite « Web mobile », tout en ne dissimulant pas les inconvénients de celle-ci,
- Enfin, une fois fait le choix de partir sur la solution « Web mobile », le document expose les points importants à connaître, et à aborder, tels que : la sensibilisation sur les plateformes à viser, la conception de son ergonomie, mais aussi l'architecture de son projet et les outils / frameworks incontournables.

2 POURQUOI LE WEB MOBILE ?

La mobilité en tant que telle est un média qui est devenu incontournable.

Faisons une sinologie : il est impensable qu'une société existe sans avoir son propre site Web (même très basique). Pourquoi ? Car depuis des années, le Web est incontournable.

Et cela est en train de se produire de nouveau, mais cette fois-ci dans le monde mobile.

Maintenant, posons-nous les questions suivantes : quelles solutions existent et surtout, que font-elles ?

A l'heure actuelle, trois approches sont mises en avant :

- **L'approche « Native »**, ou comment réaliser une application dédiée à une plateforme,
- **L'approche « Web Mobile »**, ou comment réaliser une application multiplateforme basée sur des technologies Web,
- **L'approche « Web Mobile Hybride »**, ou comment réaliser une application multiplateforme basée sur des technologies Web et utilisant aussi des fonctionnalités natives des smartphones.

Il existe une variante du dernier point que nous pourrions appeler l'approche « hybride », qui consiste à écrire intégralement son application dans un langage pivot, puis de la compiler dans un langage compatible avec les plateformes cibles. Mais elle reste encore très marginale et surtout « exotique ».

Nous pourrions citer comme frameworks faisant cela :

- Titanium d'Appcelerator,
- Rhomobile de Motorola,
- MonoTouch et MonoDroid de Xamarin.

2.1 L'approche « Native »

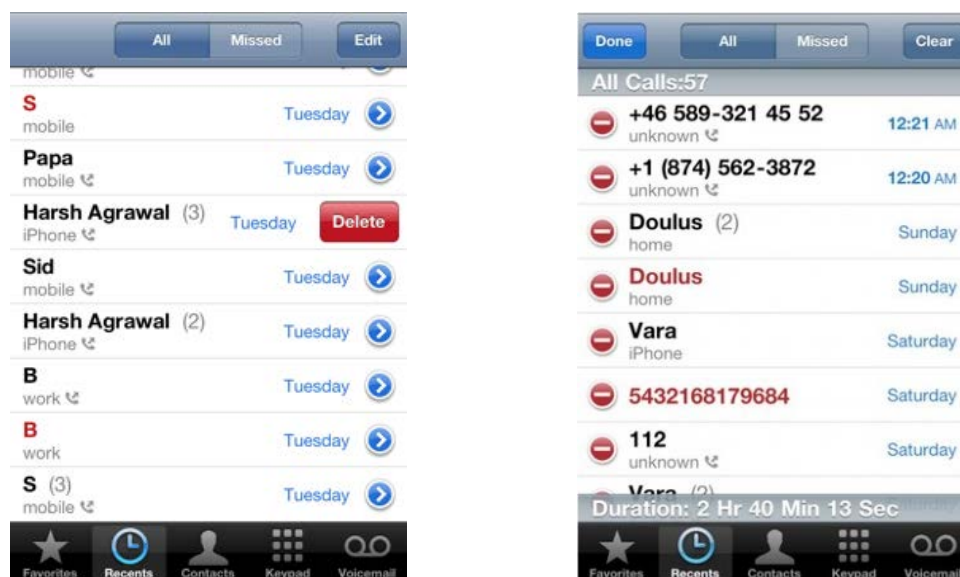
C'est l'approche la plus couramment utilisée. Elle consiste à créer des applications dédiées à un système mobile particulier. A l'heure actuelle, les plus répandus sont Android, iOS, BlackBerry et Windows Phone 7.

Quels sont les avantages ? Un des plus notables est l'intégration fine avec le système qui nous permet de (presque) tout faire, allant de la lecture des SMS à la configuration dudit système.

A cela s'ajoute la cohérence de l'ergonomie de l'application par rapport au système. En effet, malgré des principes d'ergonomie communs aux applications mobiles (voir le chapitre « Ergonomie »), nous avons tout de même des subtilités pour chacune des plateformes.

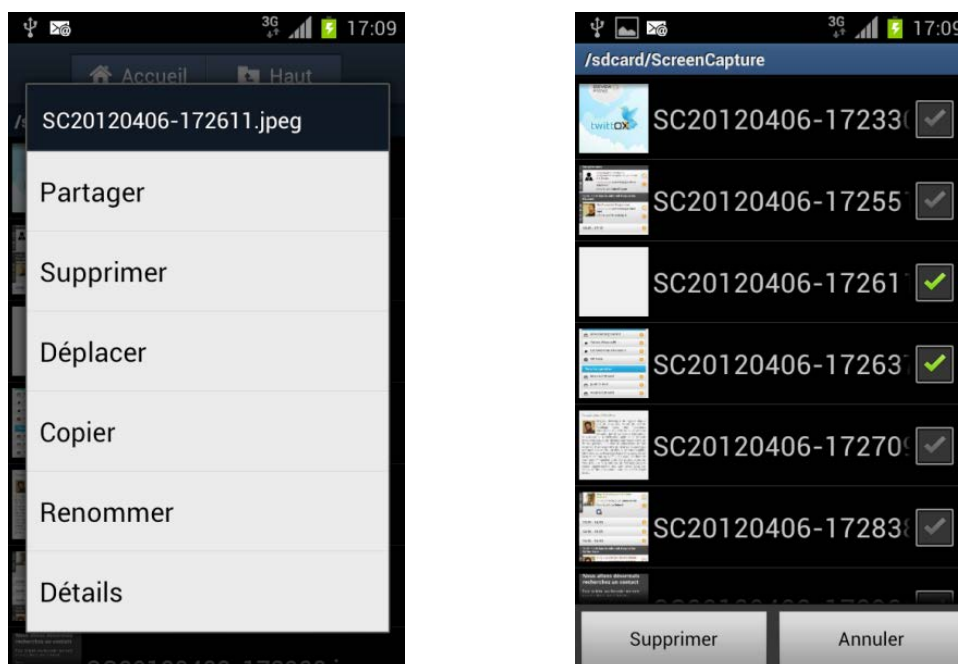
Un exemple concret visant à supprimer un élément d'une liste :

Sous iOS un « swipe » à gauche, laisse apparaître le bouton « Supprimer ». L'annulation se fait en faisant un « swipe » à droite. Ou alors, un bouton permet de basculer l'affichage en présentant une icône de suppression pour toutes les lignes.



Suppression sous iOS

Dans le cadre d'Android, soit il s'agit d'un « tap long », faisant apparaître un menu contextuel / fenêtre proposant de supprimer l'élément, soit il s'agit d'une bascule de l'affichage où nous pouvons sélectionner plusieurs éléments à la fois.



Suppression sous Android

Ainsi, le développement d'une application native permet de garder une cohérence dans l'expérience utilisateur.

Avec cette approche, nous pouvons aussi avoir accès aux couches basses pour les applications nécessitant de hautes performances, comme les jeux, ou encore la réalité augmentée.

D'un point de vue industrialisation, chaque système possède un ensemble d'outils permettant de faciliter la production des applications natives (outils de développement, de test, de déploiement). Et surtout, nous avons accès aux différents « markets », qui permettent ainsi de centraliser les applications, afin de faciliter leurs référencements et leurs accessibilités.

Un des principaux inconvénients vient de la duplication des applications si l'on souhaite adresser plusieurs plateformes. En effet, à l'heure actuelle, pour être le plus visible possible (à hauteur de plus de 95%), il faut viser trois plateformes. Pour l'instant, nous avons iOS, Android et BlackBerry. Néanmoins, Microsoft entreprend des actions agressives afin de faire avancer ses produits phares : Windows 8 et Windows Phone 8. De ce fait, nous devons nous attendre d'ici la fin 2013 à voir soit BlackBerry se faire détrôner, soit avoir un quatrième challenger. En effet, avec sa version 10, BlackBerry espère reconquérir des parts de marché.

De ce fait, nous devons dupliquer la même application pour les différentes plateformes, ce qui induit du temps et de l'argent. Ces duplications entraînent forcément de plus importantes charges de développement et, de ce fait, un budget plus important. Ceci nous amène également aux problèmes de ressources. En effet, s'il est aisé de trouver des compétences Java, où la montée en compétence sur Android / BlackBerry se fait « assez simplement », les compétences pour Windows Phone sont rares, et la montée en compétence reste accessible pour ceux qui connaissent C#. En revanche, dans le cadre

d'iOS, les compétences sont extrêmement rares, et la montée en compétence est longue et laborieuse.

Ainsi, l'approche « native » n'est pas la plus adaptée dans le contexte d'applications multiplateformes.

Il toujours est possible d'en faire, mais il faut alors avoir du temps, de l'argent et des ressources.

2.2 L'approche « Web Mobile »

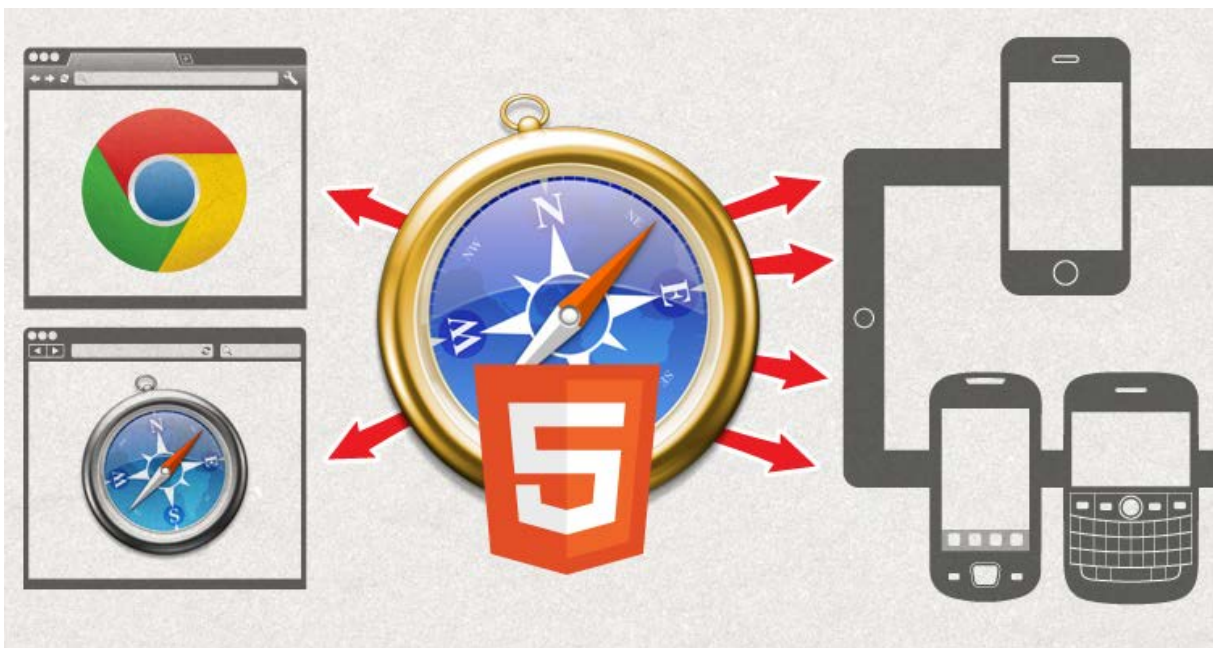
Dans le cas de l'approche « Web Mobile », nous développons avec des technologies Web modernes, comme HTML5 (qui sera validé officiellement courant 2014), CSS3 et JavaScript. De nombreux frameworks, client et serveur, nous servent de facilitateurs dans l'élaboration et l'adaptation de nos sites pour les smartphones et tablettes.

Nous avons ainsi la possibilité de créer une application « quasiment » multiplateforme. Le terme « quasiment » est utilisé sciemment, car nous aurons quelques adaptations à faire pour chacune des plateformes.

Les développeurs Web ont déjà l'habitude de le faire. En effet, pour des navigateurs de bureau, il y a un grand nombre de navigateurs à cibler (ayant des moteurs de rendus différents), avec différentes versions.

En revanche dans le monde mobile, le moteur de rendu d'Apple, Webkit, est légion. Seuls les smartphones tournant sur Windows Phone ne l'ont pas. Ainsi, nous pouvons espérer mutualiser entre 95% et 100 % du code.

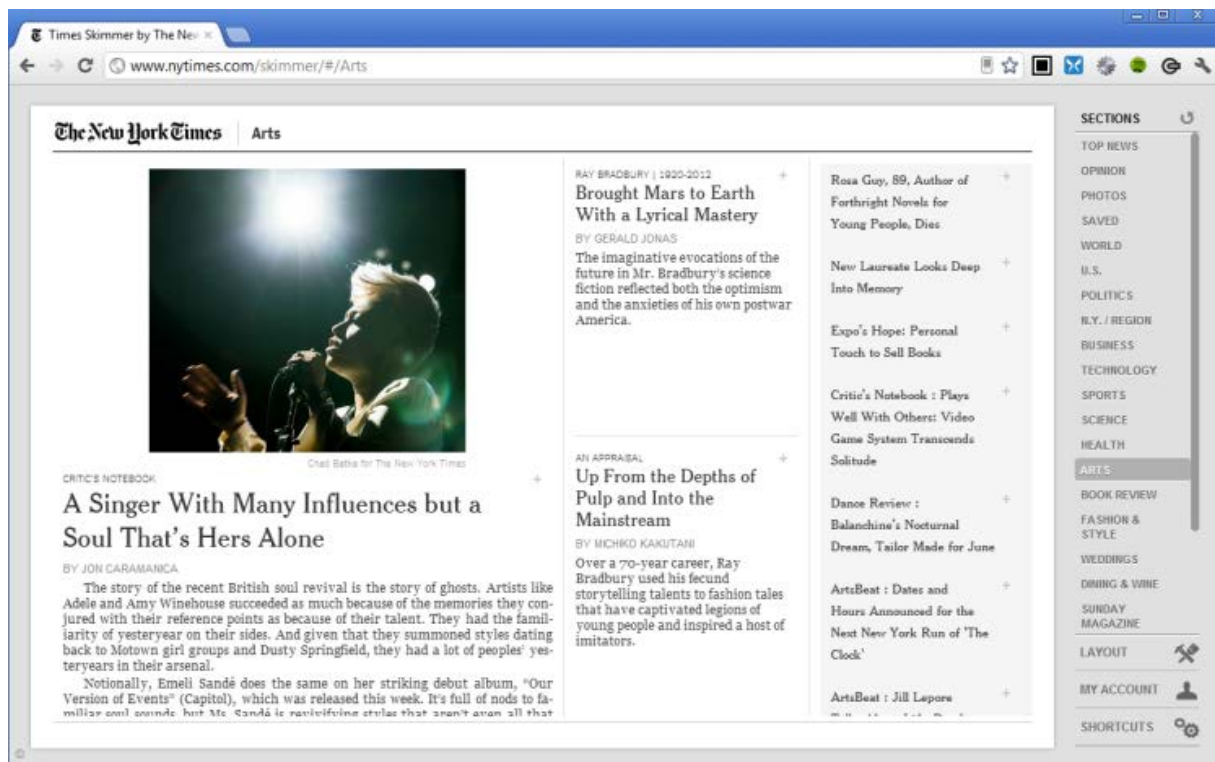
L'emploi de Webkit dans les navigateurs Web Mobile, et la puissance exponentielle de nos smartphones, permettent d'avoir des applications Web mobiles performantes.



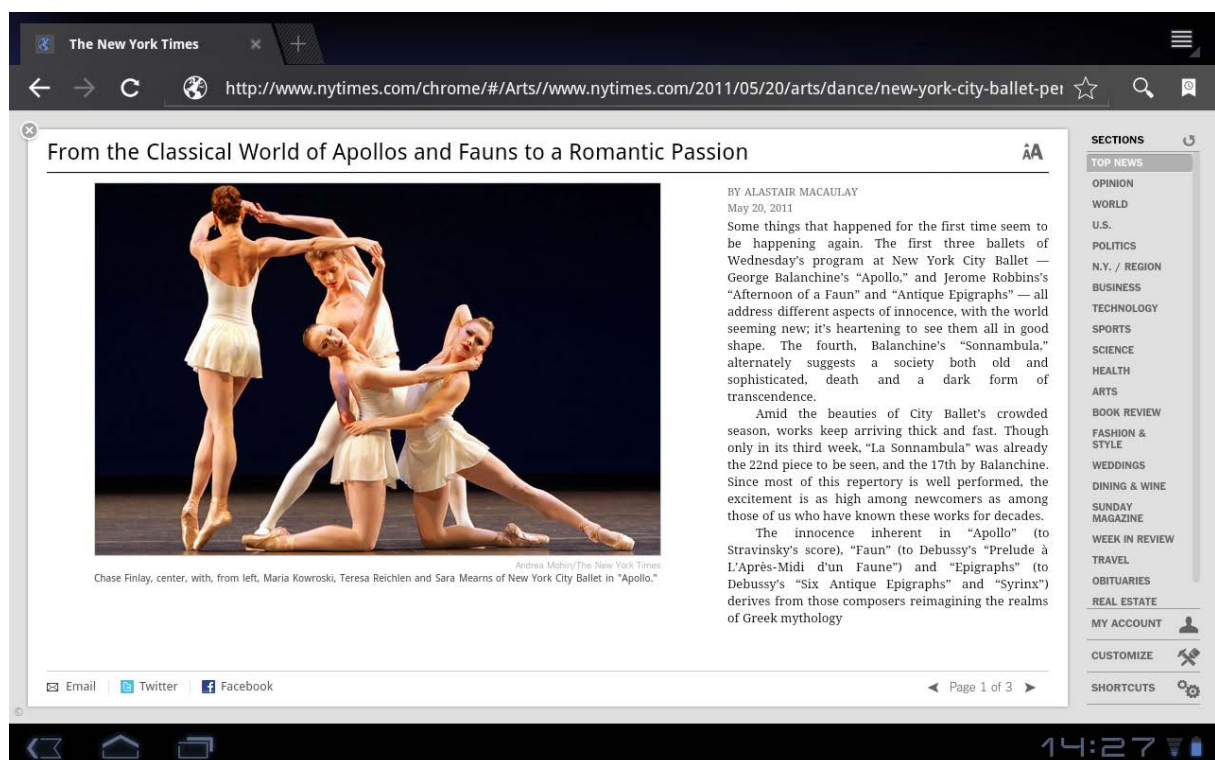
Le moteur Webkit est bien présent sur desktop, et omniprésent sur mobile

Les nouvelles technologies Web ont été pensées afin de fonctionner sur les smartphones et ainsi exploiter en partie leurs capacités. Cela permet d'offrir une bonne expérience utilisateur et une bonne interaction vis-à-vis de son smartphone. De plus, celles-ci permettent également l'élaboration d'interfaces professionnelles et adaptées en fonction du support ciblé.

Prenons l'exemple du New York Times, qui a réussi à faire une application Web accessible, tant sur desktop que sur mobile :



Le New York Times sur desktop



Le New York Times sur tablette

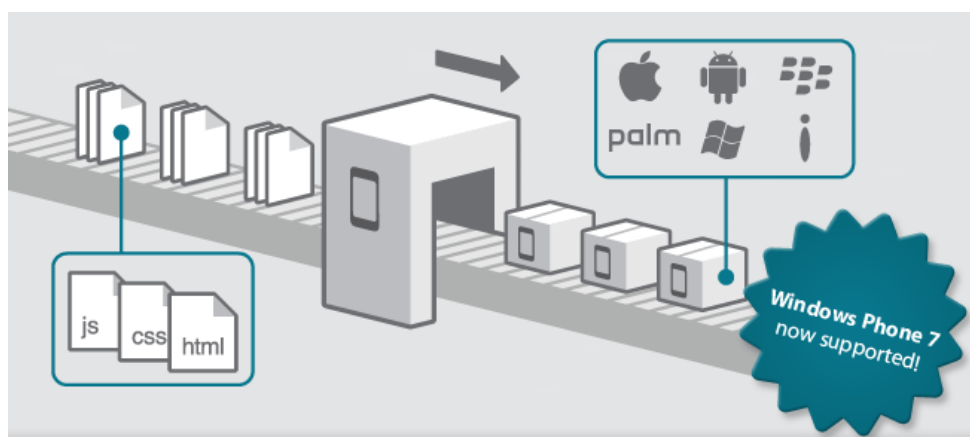
Cette mutualisation permet ainsi de produire des applications mobiles qui sont accessibles au plus grand nombre.

En revanche, nous n'avons accès ni aux hautes performances, ni aux couches basses du téléphone. Nous n'aurons pas non plus une ergonomie en parfaite

adéquation avec le système (mais cela peut se corriger), et surtout, nous n'aurons pas accès aux « markets » (bien qu'il soit désormais de plus en plus possible de référencer une application Web).

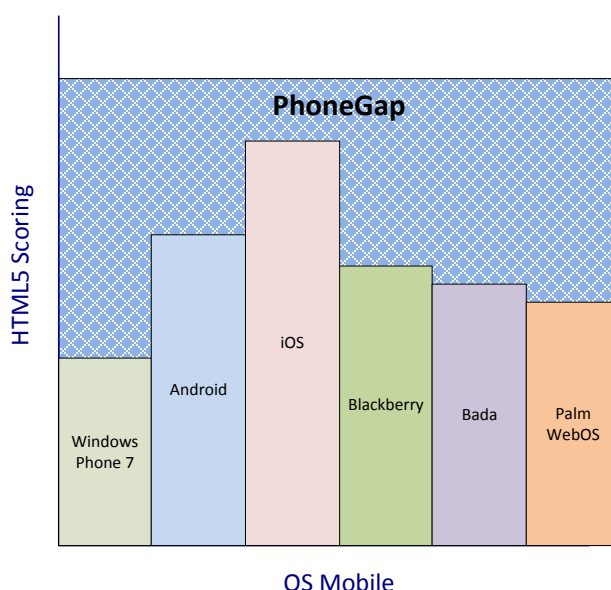
2.3 L'approche « Web Mobile Hybride »

Cette approche est un subtil mariage entre l'approche « Native » et l'approche « Web Mobile ». Elle consiste à définir une application Web Mobile en pure HTML5 / CSS3 et JavaScript, qui sera intégrée dans une enveloppe native. Celle-ci est généralement accompagnée d'un pont entre le JavaScript et les couches basses. Ainsi, elle permet d'accéder à plus de fonctionnalités des smartphones, mais surtout, elle peut être publiée officiellement sur les « markets ». Un framework permet cela : il s'agit de PhoneGap (se basant sur « Cordova »).



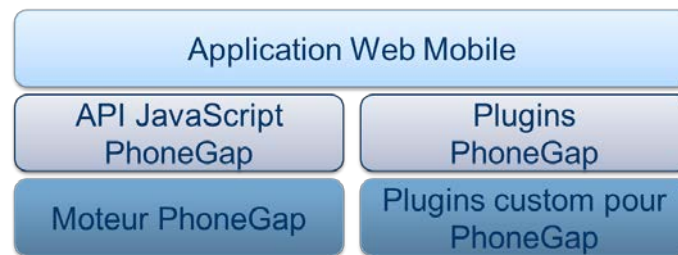
Schématisation de PhoneGap

PhoneGap permet de faciliter la définition de ces enveloppes natives incorporant les applications Web Mobile, et offrant aussi un grand nombre de fonctionnalités de bases (comme les contacts, le système de fichiers, l'état du réseau ...). D'ailleurs, ces fonctionnalités se basent en grande partie sur les spécifications du W3C. Autrement dit, PhoneGap essaie de pallier aux manques des navigateurs mobiles courants.



PhoneGap permet d'avoir des navigateurs sur le même pied d'égalité

PhoneGap repose sur une architecture à plugins, il est donc possible d'ajouter des fonctionnalités manquantes. Autrement dit, il est possible de créer ses propres plugins et ensuite de les intégrer au sein de son projet. Vous pouvez trouver sur Github les plugins déjà existants : <https://github.com/phonegap/phonegap-plugins>



Architecture de PhoneGap

Ainsi, il est possible de créer des plugins assez facilement (PhoneGap fournit une bonne documentation et API pour cela) et de les décliner pour les différentes plateformes.



Plugins PhoneGap pour envoyer un SMS pour Android et Windows Phone 7

Néanmoins il n'est pas possible d'accéder aux hautes performances. S'il manque une fonctionnalité native, il faudra alors écrire un plugin pour chacun des systèmes

mobiles visés, et surtout, il faudra tout de même installer l'environnement de développement pour chacun de ces systèmes, ce qui n'est pas négligeable.

En effet, cela sous-entend une implication sur l'installation, la maintenance et la compréhension de ces systèmes. Ce qui peut apporter un travail conséquent. Certes, il existe des solutions dans le Cloud, comme PhoneGap:Build. Néanmoins, nous dénombrons un certain nombre de problèmes de certifications et d'enregistrements dans les différents markets ...

En conclusion, cette approche est conseillée uniquement pour répondre à un besoin multiplateforme nécessitant plus de fonctionnalités que celles adressées par le Web Mobile classique.

2.4 En somme, pourquoi le Web mobile ?

Maintenant que nous avons vu les différentes solutions, demandons-nous : pourquoi le Web mobile ? Pourquoi pas du natif ou de l'hybride ?

En fait, tout dépend de vos besoins, de vos exigences. Vous trouverez ainsi au prochain chapitre, un arbre décisionnel pour prendre la meilleure décision. . Mais si vous devez faire un choix, le Web mobile est un bon compromis.

En effet, lorsque nous regardons les besoins fonctionnels d'une application mobile, la plupart peuvent être réalisés en Web mobile. Nous avons également un besoin récurrent d'être présents sur la majorité des plateformes mobiles, ce qui nécessite une application multiplateforme. Et pour cela, le Web mobile répond tout à fait à nos attentes.

D'un point de vue ressources, trouver des développeurs Web est très aisé. Le développement Web mobile est devenu monnaie courante, avec énormément de livres / d'articles, de personnes spécialisées dans le domaine (Brad Frost, Luke Wroblewski, Ethan Marcotte ...), d'outils et de frameworks.

Nous avons ainsi la possibilité, à moindre coût, de créer des applications multiplateformes mobiles, à condition bien sûr de respecter l'arbre décisionnel.

2.5 Arbre décisionnel

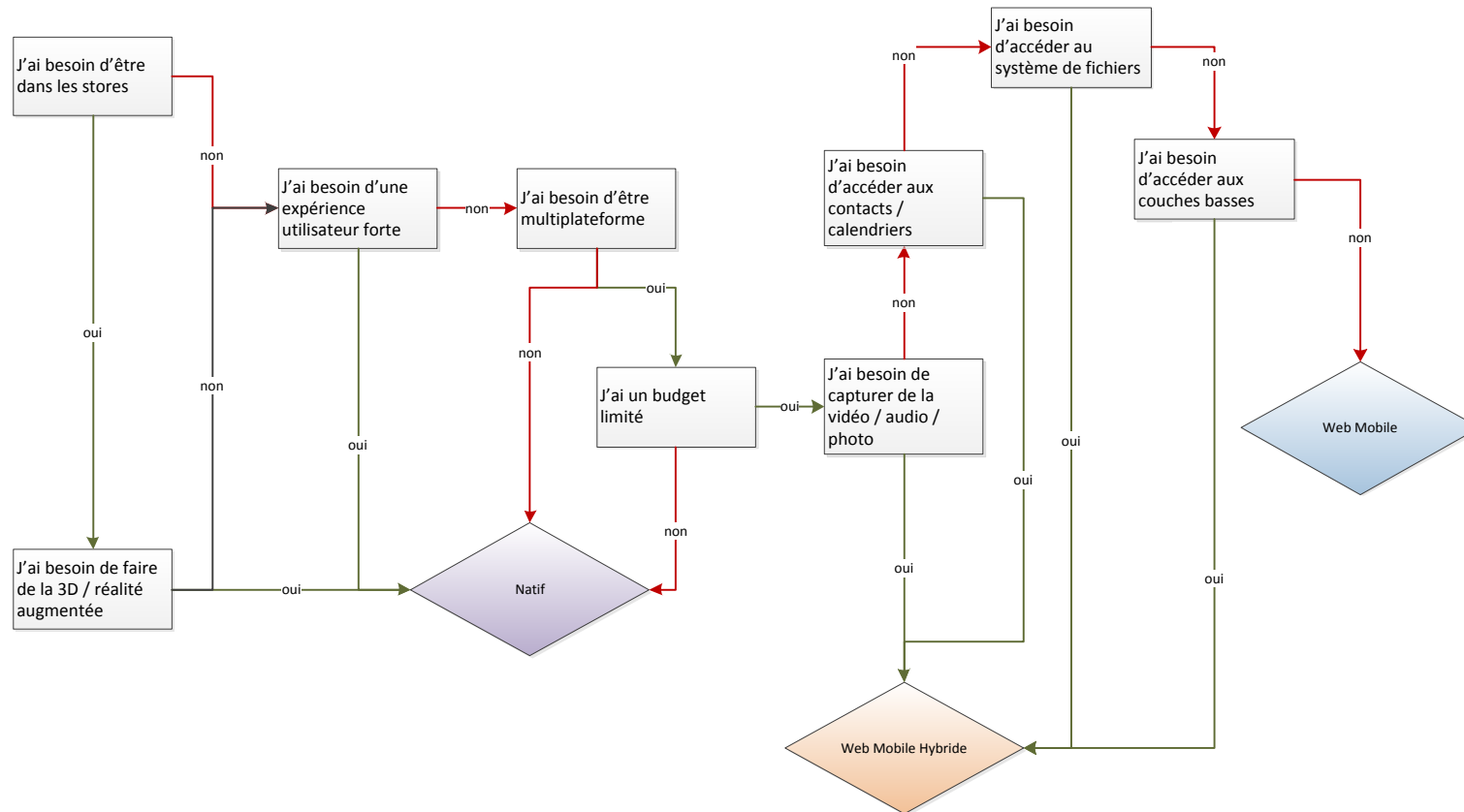
Nous avons ici deux arbres décisionnels : un avant 2014 et un second après 2014.

Cette date correspond au passage de l'état « brouillon » à l'état « final » de la spécification d'HTML5 par le W3C.

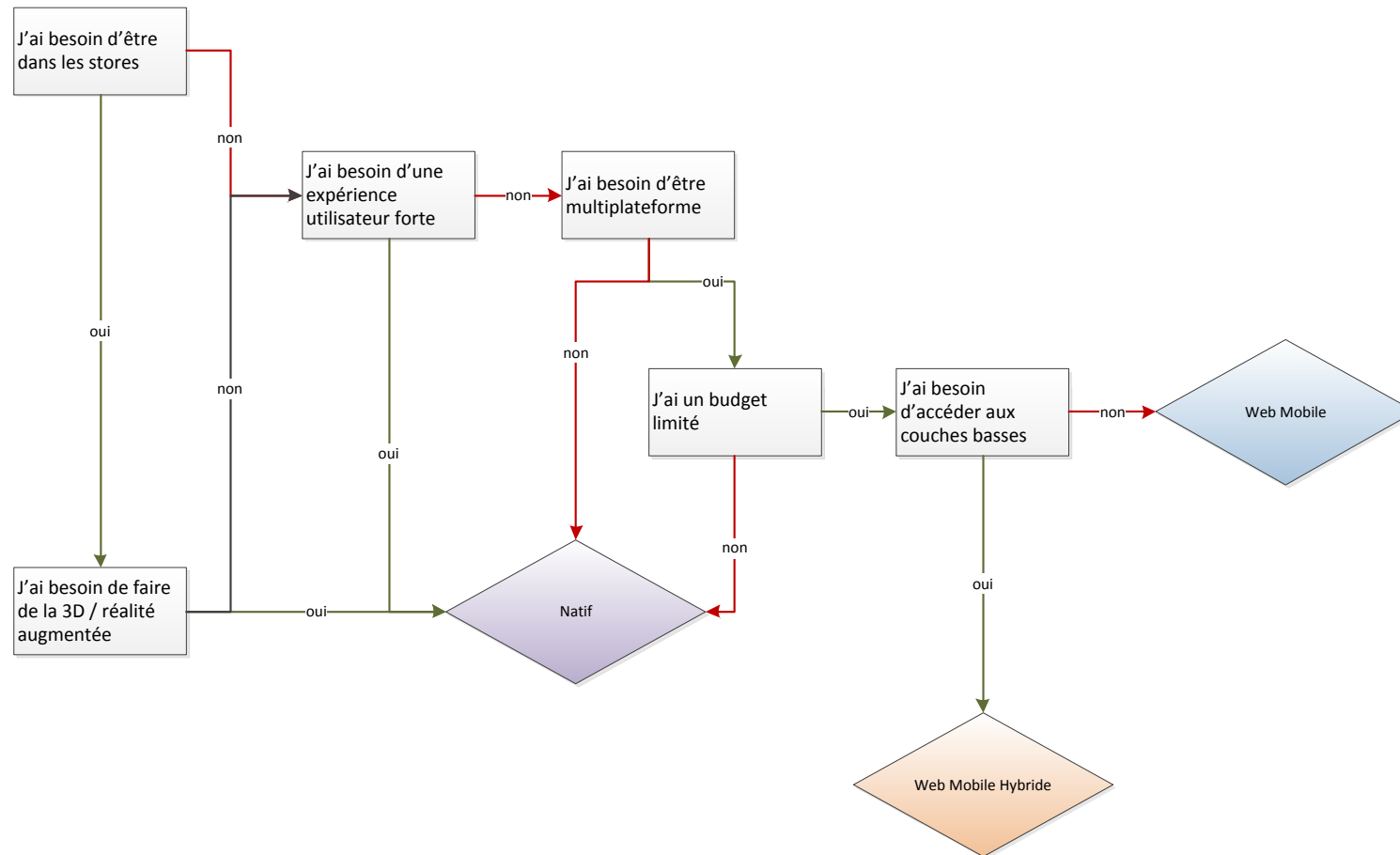
Cette date, prévue depuis début 2011, est très importante car elle va permettre aux grands éditeurs du monde de la mobilité et du Web, de donner leur avis sur les dernières fonctionnalités à valider. Ces derniers pourront ainsi se mettre à jour par rapport aux spécifications, avec les frameworks et les outils.

Néanmoins, grâce à ceux arbres, vous pourrez choisir facilement la bonne solution, en fonction de vos besoins.

Avant 2014



A partir de 2014



3 CE QUE FAIT / NE FAIT PAS LE WEB MOBILE

Libellé	Oui	Non	Prévu
Accéder aux couches basses du smartphone : <ul style="list-style-type: none"> ▪ Envoyer un SMS ▪ Gérer la mémoire ▪ Configurer le smartphone ▪ ... 		✓	
Faire de la réalité augmentée		✓	
Faire des jeux sophistiqués (nécessitant de tourner directement sur la carte graphique, comme des jeux de courses, ou des jeux en 3D)		✓	
Etre présent dans les markets			Cela dépend des markets. BlackBerry le refuse par exemple
Afficher des pages adaptées au monde mobile, ainsi qu'à une utilisation professionnelle	✓		
Gérer des animations / transitions / transformations	✓		
Lire une image / audio / vidéo, et la manipuler	✓		
Faire des graphiques / animations évoluées (via le SVG / Canvas)	✓		
Gérer le mode hors-ligne	✓		
Géolocaliser, mettre en place un GPS	✓		
Gérer le multi-touch	✓		
Reconnaître des mouvements de base	✓		
Gérer le pivotement	✓		
Gérer l'accéléromètre	✓		
Stocker des données (temporairement ou non)	✓		
Gérer des formulaires plus avancés	✓		
Gérer la validation des formulaires	✓		
Consommer aisément des services Rest	✓		
Manipuler le JSON et le XML	✓		
Manipuler / gérer des fichiers (dans son propre espace de stockage)			Uniquement avec les versions récentes des mobiles
WebIntent <ul style="list-style-type: none"> ▪ mailto: envoi de mail ▪ tel: fait un appel 	✓		
Accéder / manipuler les contacts			Dans le draft W3C. Pas avant fin 2014.
Accéder / manipuler le calendrier			Dans le draft W3C. Pas avant fin 2014.
Capturer une image / audio / vidéo (WebRTC)			Dans le draft W3C. Pas avant fin 2014. Android permet déjà de capturer des images et du son
Accéder à l'état du réseau			Dans le draft W3C. Pas avant fin 2014. Déjà présent sur Android
Utilisation des WebIntent			Associer des comportements par rapport à des URLs <ul style="list-style-type: none"> ▪ sms: envoi de sms ▪ mms: envoi de mms ▪ geo : afficher une position, ou un itinéraire sur le logiciel de cartographie <ul style="list-style-type: none"> ○ Présent sur Android
Faire du temps réel (WebSocket)			Dans le draft W3C. Pas avant fin 2014. Présent sur iOS
Faire de la 3D (avec OpenGL ou avec les animations / transitions)			Dans le draft W3C. Pas avant fin 2014.
Paralléliser le traitement (via les WebWorker)			Dans le draft W3C. Pas avant fin 2014. Déjà présent sur BlackBerry
Faire vibrer le téléphone			Dans le draft W3C. Pas avant fin 2014.

4 FONCTIONNALITES PAR PLATEFORME MOBILE

Fonctionnalités HTML5	iOS	Android		BlackBerry		WP7	WP8	Navigateurs	
	Safari	Smartphone	Tablette	Smartphone	Tablette	IE	IE	Chrome	Firefox
	Stockage / Mode hors ligne								
Application Cache	✓	✓ 2.1+	✓	✓ 6.0+	✓		✓	✓	✓
	Mécanisme permettant de télécharger de manière explicite les ressources nécessaires à notre application afin que celle-ci puisse fonctionner en mode « hors-ligne »								
Web storage	✓	✓ 2.0+	✓	✓ 6.0+	✓	✓	✓	✓	✓
	Mécanisme permettant de stocker sous forme de clés / valeurs des données (pouvant être liées à session) côté client. Volume de données possible : environ 2.5 MO								
Web SQL storage	✓	✓ 2.0+	✓	✓ 6.0+	✓			✓	✓
	Mécanisme permettant de stocker dans une base SQL des données côté client. Volume de données possible : 10 à 15 MO								
Web IndexedDB storage					✓		✓	✓	✓
	Mécanisme permettant de stocker dans une base NoSQL des données côté client. Volume de données possible : 10 à 15 MO								
File API	✓ 6.0+	✓ 4.0+	✓		✓		✓	✓	✓ 3.0 Android
	Mécanisme permettant de gérer un ensemble de fichiers (temporaires ou non) et qui soient liés à notre nom de domaine. Utile pour stocker des images, des PDFS, ...								

Tableau récapitulatif inspiré du site <http://mobilehtml5.org/>

Fonctionnalités HTML5	iOS	Android		BlackBerry		WP7	WP8	Navigateurs	
	Safari	Smartphone	Tablette	Smartphone	Tablette	IE	IE	Chrome	Firefox
	Multimédia								
Multimedia	✓	✓ 2.3+	✓	✓ 7.0+	✓	✓	✓	✓	✓
	Mécanisme permettant de jouer une vidéo ou un fichier audio								
Canvas API	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Mécanisme permettant de dessiner des éléments avec une API de bas niveau (un peu comme une table traçante). C'est un objet qui a un coût mémoire faible et est très performant.								
SVG	✓	✓ 4.0+	✓	✓	✓	✓	✓	✓	✓
	Mécanisme permettant de dessiner des éléments avec une API de haut niveau (déclaration à la XML). C'est plus facilement manipulable, mais prend plus de mémoire								
HTML Media Capture	✓ 6.0+	✓ 4.0+	✓					✓	✓
	Mécanisme permettant de faire de la capture audio, vidéo, photo								

Tableau récapitulatif inspiré du site <http://mobilehtml5.org/>

Fonctionnalités HTML5	iOS	Android		BlackBerry		WP7	WP8	Navigateurs	
	Safari	Smartphone	Tablette	Smartphone	Tablette	IE	IE	Chrome	Firefox
	Présentation								
CSS3 Basic	✓	✓	✓	✓ 6.0	✓	✓	✓	✓	✓
	Opacité, couleurs, effets sur les textes, coins arrondis ...								
CSS3 Transforms	✓	✓ 2.0+	✓	✓ 6.0	✓	✓	✓	✓	✓
	Mécanisme permettant de transformer nos éléments (leur appliquer une rotation, une translation ...)								
CSS3 Transitions	✓	✓ 2.0+	✓	✓ 6.0	✓		✓	✓	✓
	Mécanisme permettant d'appliquer des effets sur la modification de la valeur d'une propriété CSS (avoir une transition douce ou lieu d'un changement brutal)								
CSS3 Animations	✓	✓ 2.0+	✓	✓ 6.0	✓		✓	✓	✓
	Mécanisme permettant de définir des scénarios d'animations (changement de propriétés dans le temps)								
Viewport	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Mécanisme permettant de réadapter le contenu de notre page par rapport à la largeur du support								
Position: fixed support	✓ 5.0+	✓ 2.2+	✓ 3.1+	✓ 7.0+	✓		✓	✓	✓ 11+
	Pouvons-nous placer des éléments qui ne changent jamais de place, à travers le viewport vu précédemment ?								

Tableau récapitulatif inspiré du site <http://mobilehtml5.org/>

Fonctionnalités HTML5	iOS	Android		BlackBerry		WP7	WP8	Navigateurs	
	Safari	Smartphone	Tablette	Smartphone	Tablette	IE	IE	Chrome	Firefox
	Interaction système & outils								
Network Information API		✓ 2.2+	✓						✓
	Mécanisme permettant de savoir le type de réseau, son débit et son statut (hors-ligne ...)								
Notifications API					✓ 2.0+				✓
	Mécanisme permettant de notifier les utilisateurs en utilisant le système de notification natif.								
Geolocation	✓	✓ 2.0+	✓	✓ 6.0+	✓	✓	✓	✓	✓
	Mécanisme permettant de géolocaliser ou d'utiliser le GPS du smartphone								
Web Sockets	✓ 4.2+			✓ 6.1+	✓		✓	✓	✓ 7+
	Mécanisme permettant de mettre en œuvre le push pour des applications Web (versus le polling Ajax).								
Web Workers	✓ 5.0+			✓ 6.0+	✓		✓	✓	✓
	Mécanisme permettant de paralléliser le traitement JavaScript								

Tableau récapitulatif inspiré du site <http://mobilehtml5.org/>

Fonctionnalités HTML5	iOS	Android		BlackBerry		WP7	WP8	Navigateurs	
	Safari	Smartphone	Tablette	Smartphone	Tablette	IE	IE	Chrome	Firefox
	Interaction utilisateur								
Motion Sensors	✓ 4.2	✓ 4.0+	✓		✓			✓	✓
	Mécanisme permettant d'exploiter le gyroscope et l'accéléromètre du smartphone								
Touch Events	✓	✓ 2.1+	✓	✓ 6.1+	✓		✓	✓	✓
	Mécanisme permettant de détecter les positionnements des doigts sur l'écran (un peu l'équivalent de la gestion de la souris)								

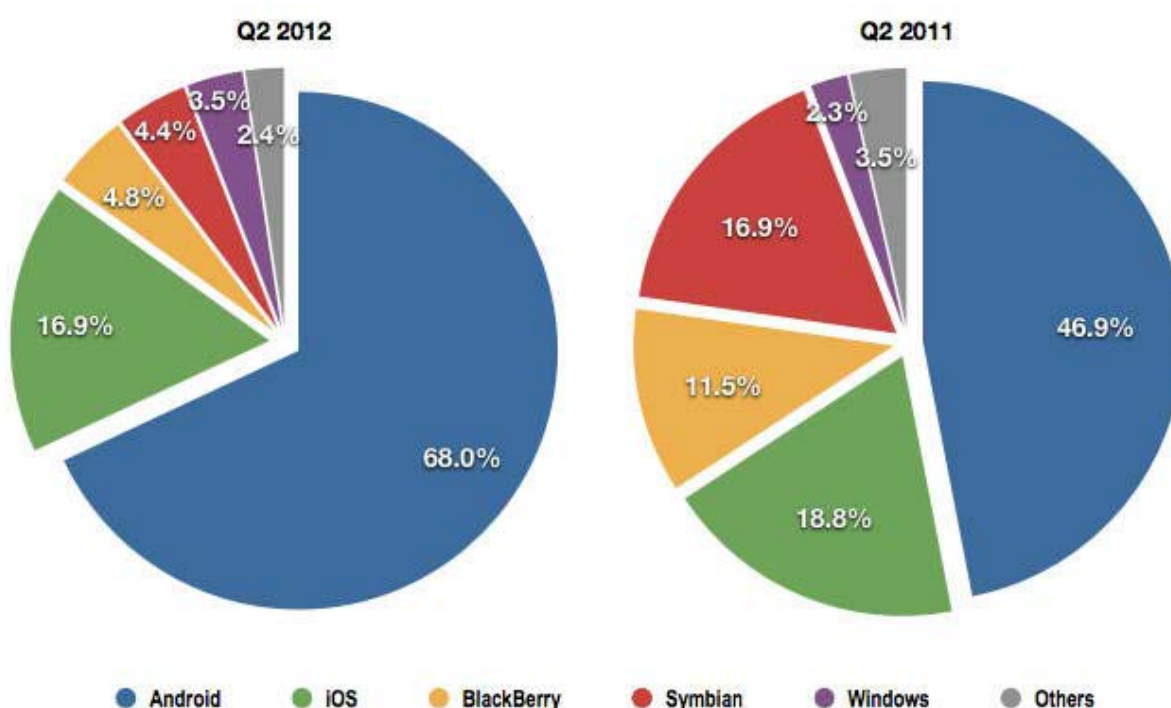
Tableau récapitulatif inspiré du site <http://mobilehtml5.org/>

5 STATISTIQUES SUR LES OS MOBILE

Le but de cette section est de montrer la répartition des différents OS mobile et, pour chaque OS, les versions utilisées. Cela permet de cibler les OS ainsi que les fonctionnalités HTML5 / CSS3 disponibles.

Ce chapitre va vous aider à cibler les OS à supporter, mais aussi les fonctionnalités HTML5 / CSS3 disponibles en présentant la répartition des OS mobiles utilisés et leurs versions.

5.1 Sur l'ensemble des smartphones

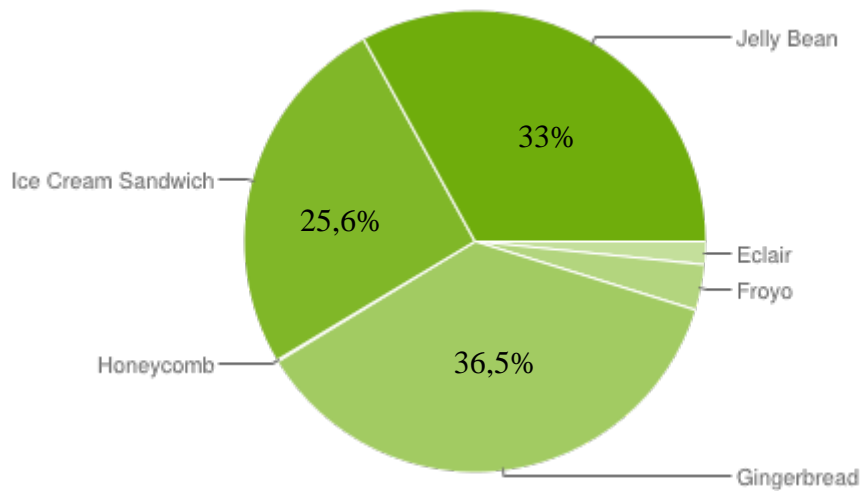


Réf : <http://abuggedlife.com/2012/08/10/android-now-4x-bigger-than-ios-symbian-and-blackberry-each-at-4-market-share/>

En somme, si nous voulons être le plus « vu » par l'ensemble des utilisateurs mobiles, il faut viser comme plateforme :

- Android
- iOS
- BlackBerry

5.2 Pour Android

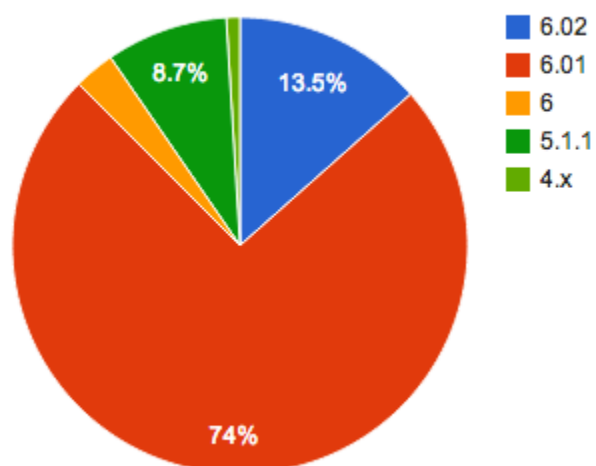


Réf : <http://developer.android.com/about/dashboards/index.html>

En somme, les versions à viser (et donc pour lesquelles il faut tester le bon fonctionnement de son application) sont :

- Jelly bean (4.1.x)
- IceCream Sandwich (4.0.x)
- GingerBread (2.3.x)

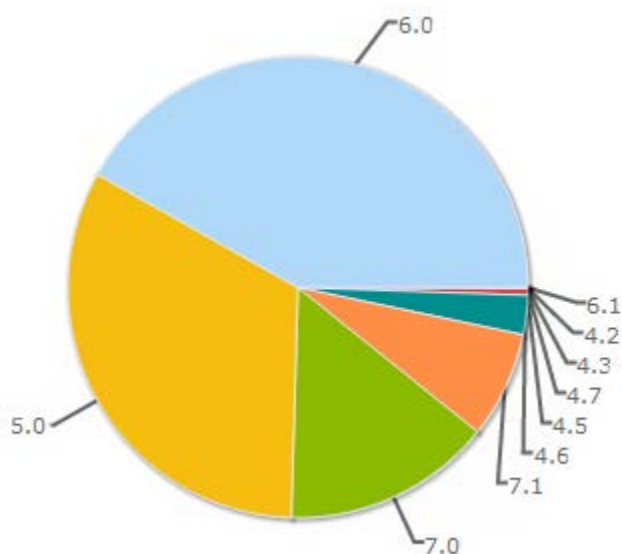
5.3 Pour iOS



Réf : <http://www.14oranges.com/2013/01/ios-version-statistics-january-14th-2013/>

En somme, nous devons considérer que l'application doit être testée sur des versions d'iOS supérieures ou égales à la version 6.0.1.

5.4 Pour BlackBerry



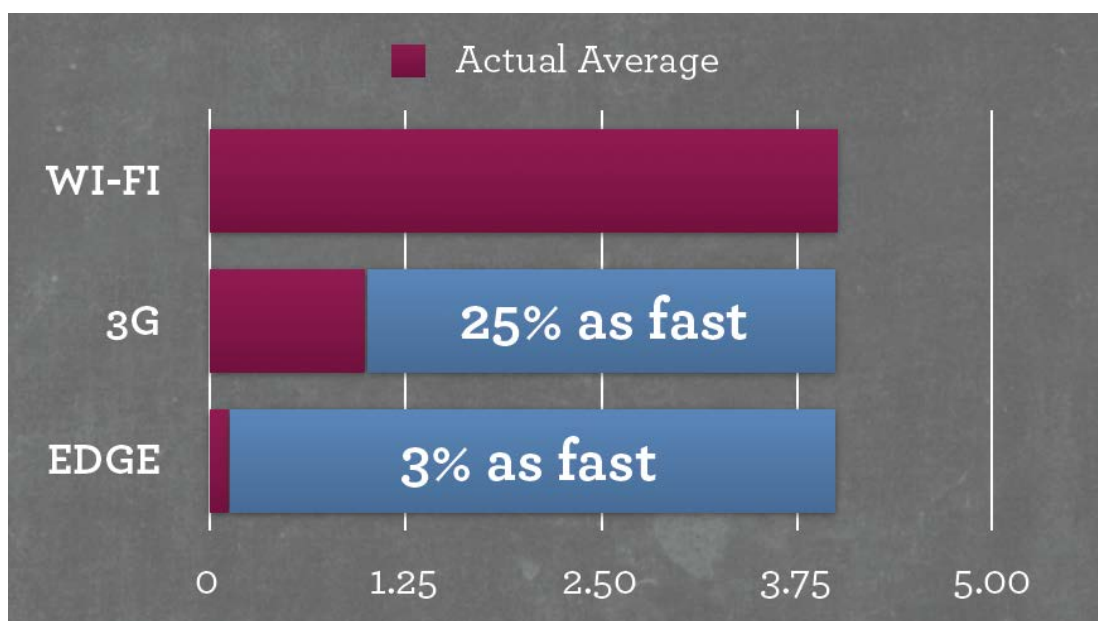
Réf : <http://www.inteist.com/tag/blackberry/>

En somme, nous devons considérer que l'application doit être testée sur des versions de BlackBerry supérieures ou égales à la version 6. A noter d'ailleurs que la majeure partie des fonctionnalités permettant de faire son application sur BlackBerry, viennent de la version 6.

6 CONTRAINTES COURANTES LIEES A LA MOBILITE

Ce chapitre consiste à vous mettre en garde sur certains aspects liés à la mobilité, et qui ne se limitent pas au Web Mobile.

En effet, il ne faut pas oublier que dans le monde mobile, nous sommes ... mobiles ! De ce fait, les pertes de connexions ne sont pas rares, chose qui n'est (quasiment) plus gérée dans le monde desktop. Qui plus est, nous sommes dans un monde où nous communiquons avec de la 3G ... dans le meilleur des cas ! Parfois, nous avons la chance d'avoir un utilisateur en Wi-Fi, mais cela reste occasionnel. De plus, la 3G est un réseau mobile qui est lent par rapport à l'ADSL, et encore plus par rapport à la fibre d'entreprise !



Ratio de transfert entre les différents réseaux

Cela nous amène donc aux contraintes suivantes : limiter au maximum les interactions avec le serveur, et surtout éviter de transiter des données volumineuses ! Privilégier également autant que possible la mise en cache des données, ainsi que le format JSON. Ce dernier étant beaucoup moins verbeux qu'XML, et,, partiellement typé, il permettra de réduire la taille de la bande passante. De plus, l'exploitation du JSON dans le monde Web Mobile est un plus, car il sera interprété de manière optimisée par les navigateurs, qui le comprennent très bien, et sera manipulable aisément par les développeurs.

N'oublions pas que nous sommes sur un support limité. Certes, certains smartphones / tablettes sont plus puissants que des ordinateurs de desktop, mais l'interaction avec eux est beaucoup moins riche ! Nous n'avons ni clavier, ni souris, nous devons nous contenter de nos doigts, sur une surface plutôt restreinte.

Il faut donc :

- proposer des interfaces épurées, simples,

- des fonctionnalités qui soient aisément accessibles (ne nécessitant pas l'emploi de clavier et de souris),
- éviter également d'avoir trop de saisie, ou de sélectionner de la saisie / texte,
- être attentif sur la disposition des éléments : la taille de l'écran l'imposera.

Il faut de plus être conscient que tous les supports ne sont pas sur un pied d'égalité quant à la taille des écrans : il y en a une grande variété, avec une palette de résolutions différentes. De ce fait, prévoir une application qui puisse se réagencer de manière dynamique, est impératif. C'est ce que l'on nomme le « responsive design » (ou le design réactif).



Sur quels supports tourne le Web dans le monde mobile

Et ceci est d'autant plus vrai avec l'hétérogénéité des supports.



Sur quels supports tourne le Web actuellement

Et encore plus dans le futur.

THIS WILL BE THE WEB.



Sur quels supports tournera le Web

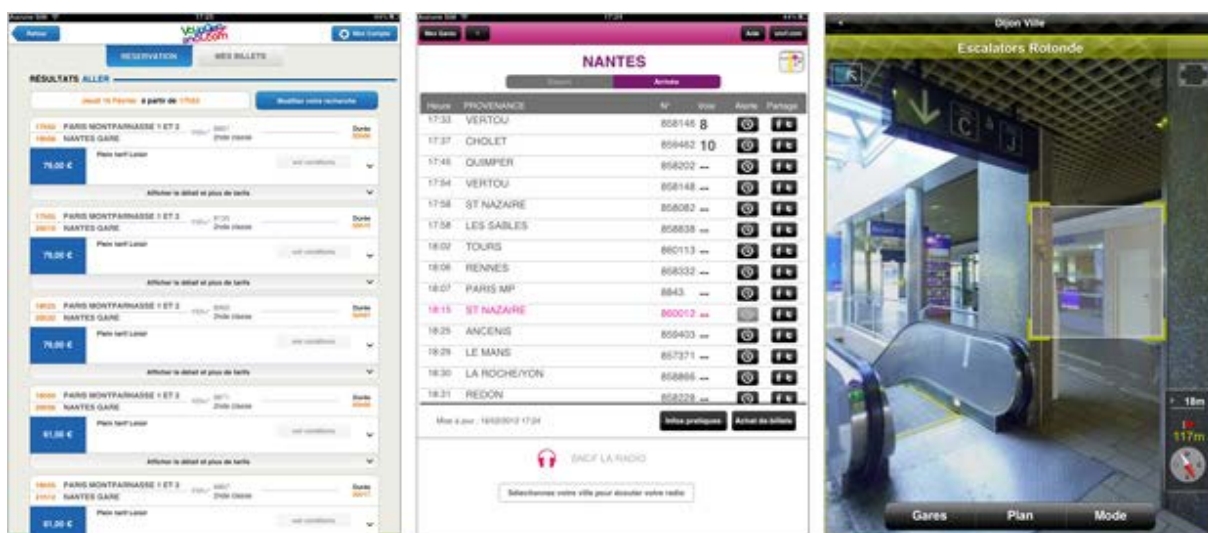
7 ERGONOMIE

7.1 Proposer une gamme limitée de fonctionnalités

Une application doit rester la plus simple possible. De ce fait, non seulement nous ne pourrions pas toujours avoir les mêmes fonctionnalités que sur les navigateurs de bureau (dû aux limitations de matériel), mais en plus, il faudra penser à proposer l'essentiel !

Nous ne passons pas autant de temps sur un smartphone que sur notre ordinateur de bureau. Nous avons l'habitude de faire des actions rapides. De ce fait, nous devons avoir des applications pragmatiques, qui vont à l'essentiel. Dans un contexte de mobilité, où potentiellement, nous pouvons faire nos actions en marchant, et dans un temps plus limité, il faut que l'interface soit la plus épurée possible de fonctionnalités.

D'ailleurs, les grands comptes/entreprises, qui ont un large panel de fonctionnalités, l'ont bien compris : ils préfèrent proposer un ensemble d'applications permettant de répondre à un sous-ensemble de fonctionnalités.

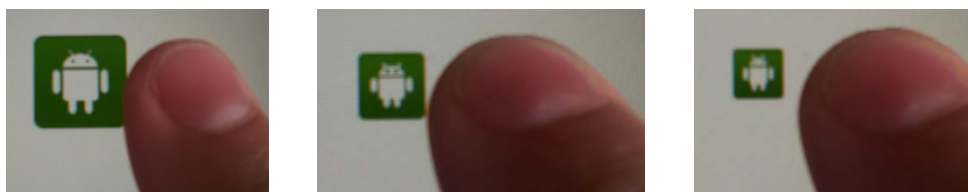


SNCF a réalisé une famille d'applications iPad et iPhone plutôt qu'une application unique trop dense.

7.2 Des icônes / éléments adaptés à nos doigts

Qu'on se le dise : nous n'aurons jamais une souris et un clavier sur nos smartphones et tablettes. De ce fait, nous sommes obligés d'utiliser nos doigts. Et cette obligation a un impact fort sur les interfaces que nous avons à produire.

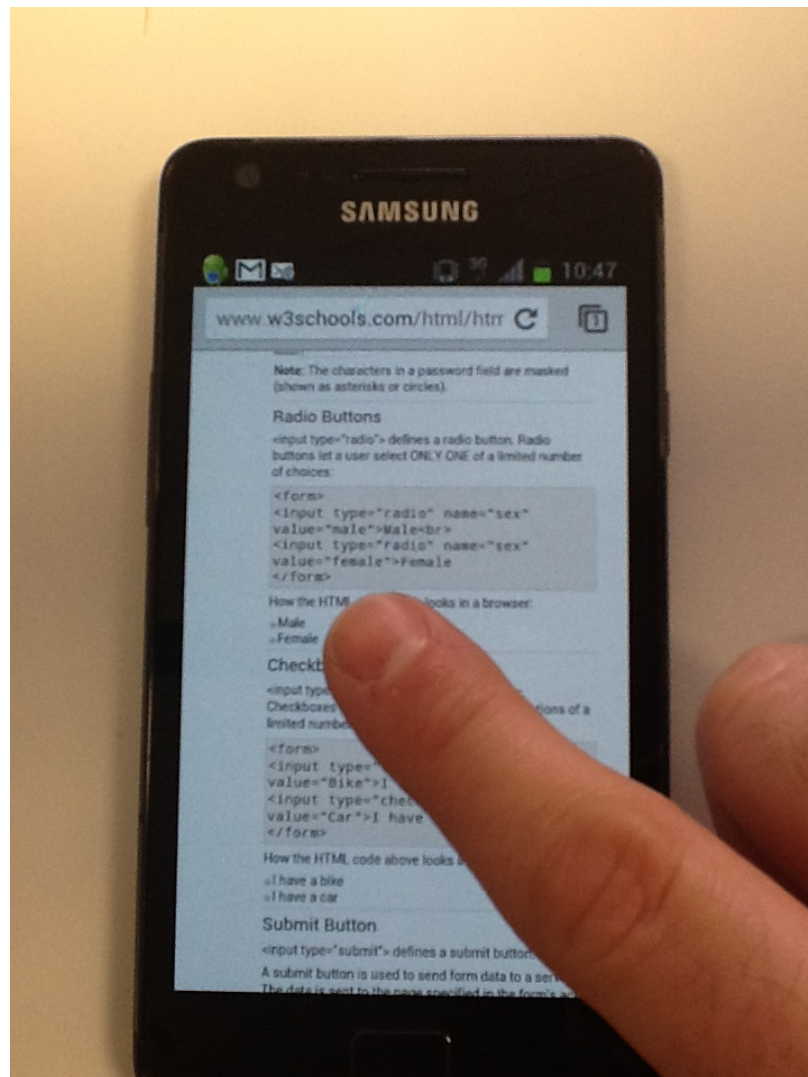
Vous devez vous rendre compte qu'un doigt en règle générale tourne entre 45 et 60 px de diamètre. Il faut donc éviter de reprendre les icônes des sites Web, qui ont tendance à faire 24 / 32px. Il faut également espacer les éléments, surtout lorsque l'on utilise des listes, afin de pouvoir facilement cliquer dessus. Lorsque l'on crée une application Android, le SDK Android nous fournit des assets par défaut (par exemple l'image représentant l'icône de l'application mesure sur un smartphone 48px * 48px).



*De gauche à droite : 48px * 48px, 32px * 32px, 24px * 24px*

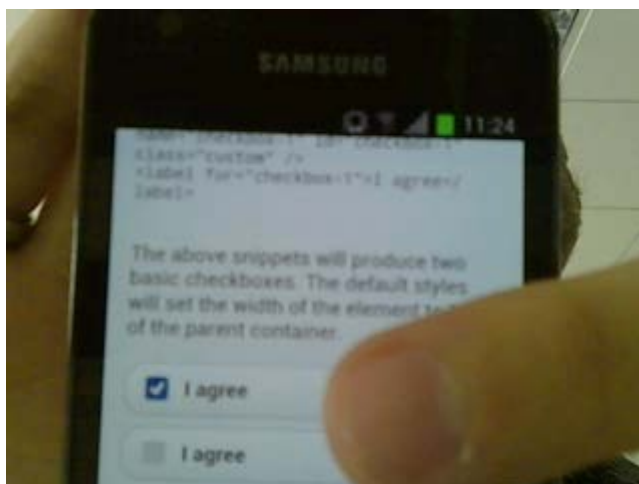
Ainsi, il faut adapter nos composants, afin de les rendre plus accessibles. Mais également faciliter la saisie et l'interaction !

Prenons l'exemple des cases à cocher. Sur un navigateur mobile, par défaut, cela s'affiche comme cela :



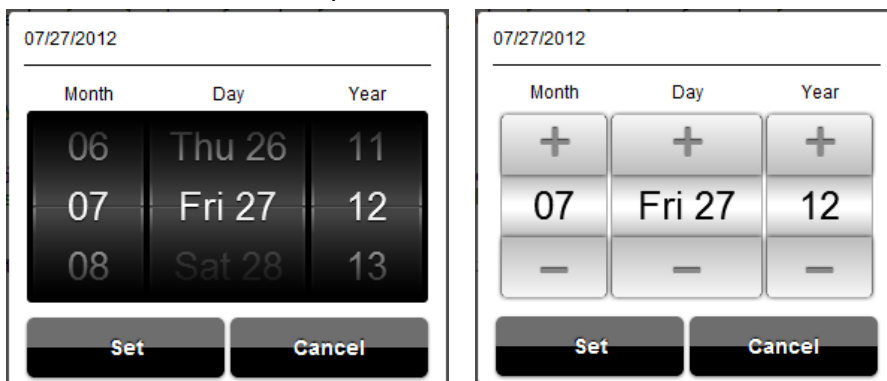
Un peu petit ...

Par rapport à nos doigts, ce n'est pas pratique. Il faut donc penser à les convertir dans un modèle plus accessible :



Beaucoup mieux !

Et cela peut s'étendre à beaucoup de choses :



Un datepicker plus simple d'accès !



Un slider pratique

Example 01

The simplest implementation.

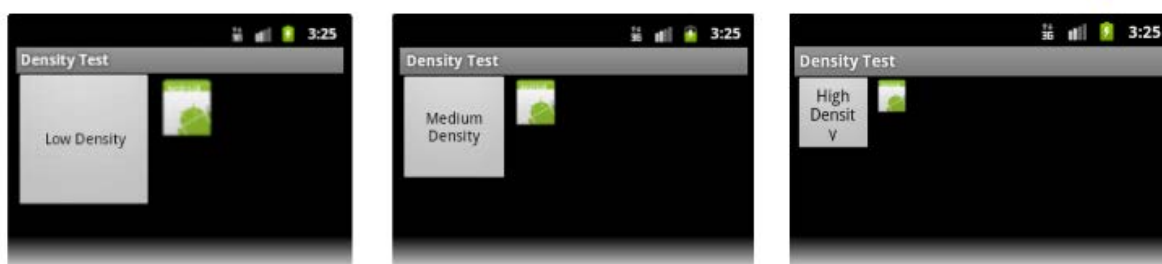


Une case à cocher sympa

7.3 La densité des écrans

Il faut savoir qu'un iPhone 4 a une largeur physique de 640 pixels, mais quand nous affichons nos applications, la largeur est de 320 pixels (probablement pour obliger la mise en place d'images de meilleure qualité). Nous tombons typiquement dans des problèmes de densité d'écran.

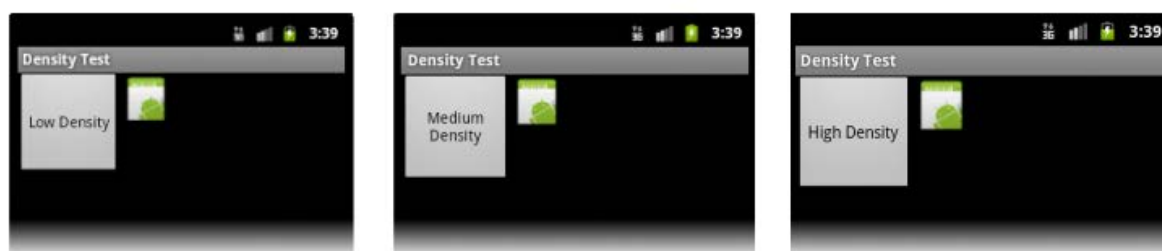
Nous devons nous dire qu'un pixel physique ne correspond pas toujours à un pixel affiché. Ainsi, un élément d'interface utilisateur va sembler physiquement plus grand sur un écran de faible densité, et inversement, plus petit sur un écran de forte densité.



Mauvaise gestion de la densité

De ce fait, il faut toujours prévoir un set d'images identiques avec différentes densités. Le système Android prévoit de base une telle gestion, en plaçant les images adaptées dans des répertoires correspondants pour des écrans de forte densité, faible densité

Mettre les Media Queries de CSS3, permet d'utiliser les bonnes images en fonction de la densité de l'écran.



Bonne gestion de la densité

7.4 Les zones accessibles

L'endroit où sont placés les éléments est primordial ! En effet, il est important de bien prendre en compte la façon dont l'utilisateur tient son smartphone et sa tablette. En règle générale, un droitier tient son smartphone de la main droite, et la plupart des actions se font via le pouce. Ce qui donne la cartographie suivante :



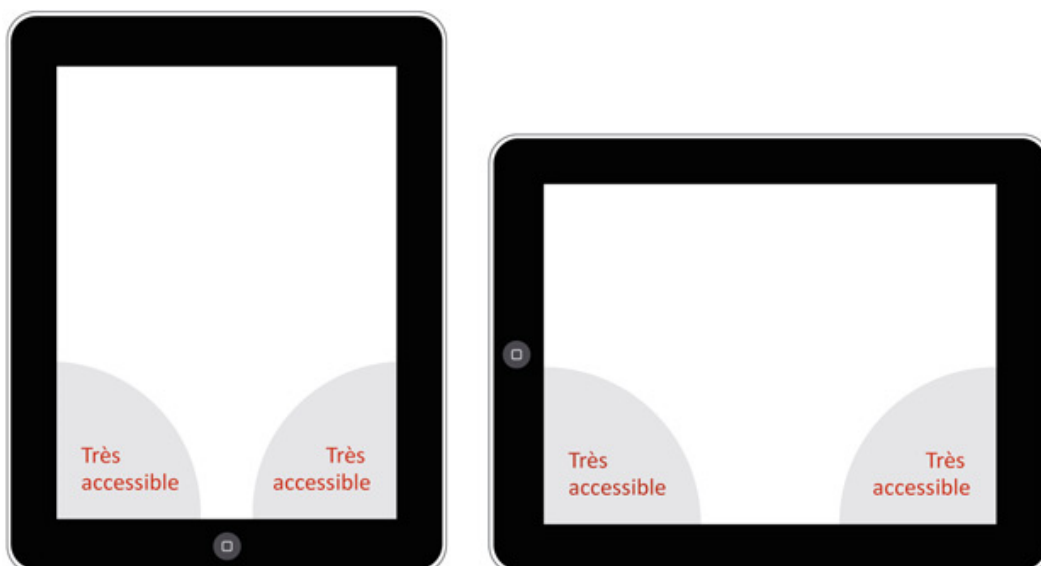
A gauche, les zones accessibles en tenant le smartphone de la main droite. A droite, la fusion des zones par rapport à la droite et à la gauche

Cela indique où placer des éléments en fonction de leur criticité. Par exemple, un bouton qui doit confirmer la suppression de tous les contacts se mettra plutôt en haut pour ne pas être facilement accessible.



Un exemple concret

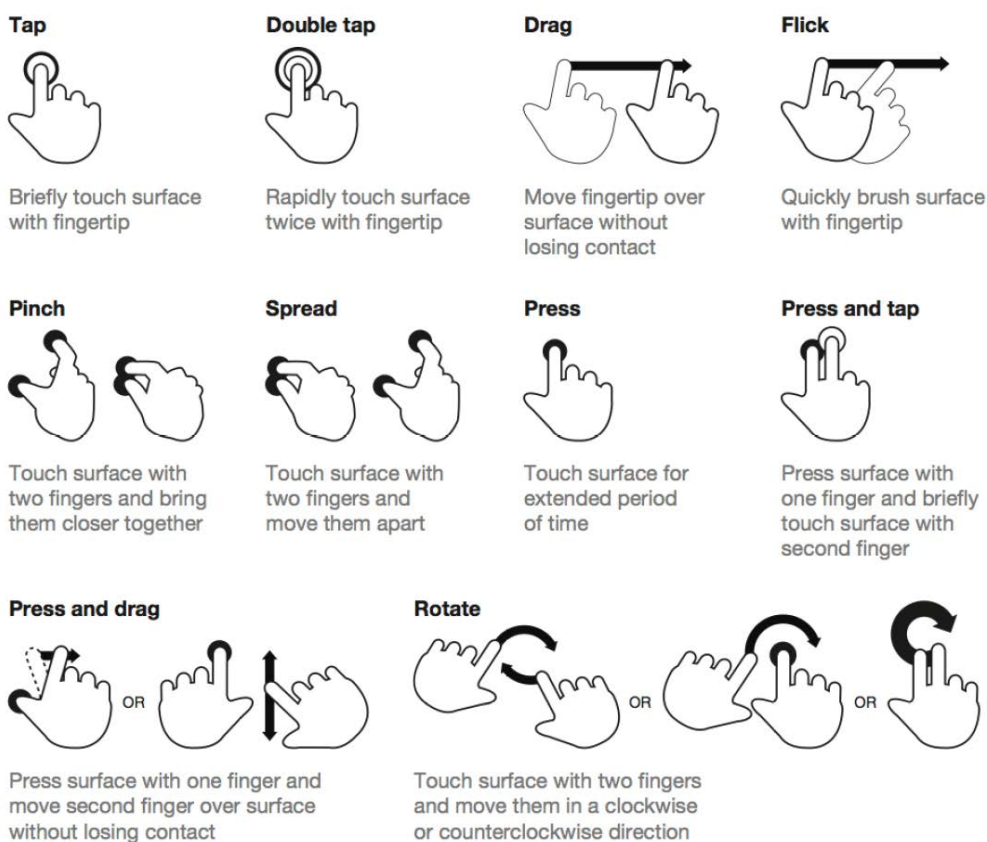
Et cela va de pair avec les tablettes !



Les zones accessibles sur tablettes

7.5 Les gestes usuels

Voici une liste des gestes que les utilisateurs de smartphones et de tablettes ont l'habitude de faire



Gestes usuels

7.6 Adapter la conception de vos écrans

Ceci correspond à l'approche dite « responsive design ».

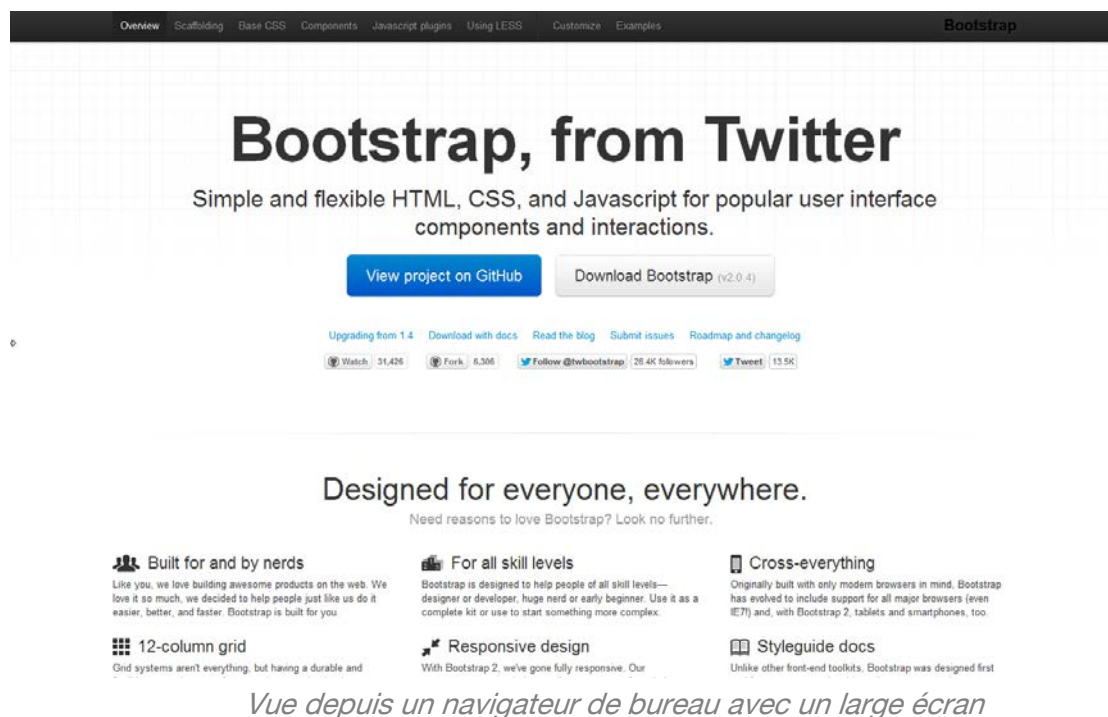
Il s'agit d'agencer le contenu des pages et de réadapter les composants en fonction de la résolution de l'écran, du support (navigateur de bureau, smartphones ...), ou des fonctionnalités du navigateur.

L'objectif est double : mutualiser au maximum le code de l'application, mais également le rendre accessible au plus grand nombre en évitant de le dupliquer en fonction des supports.

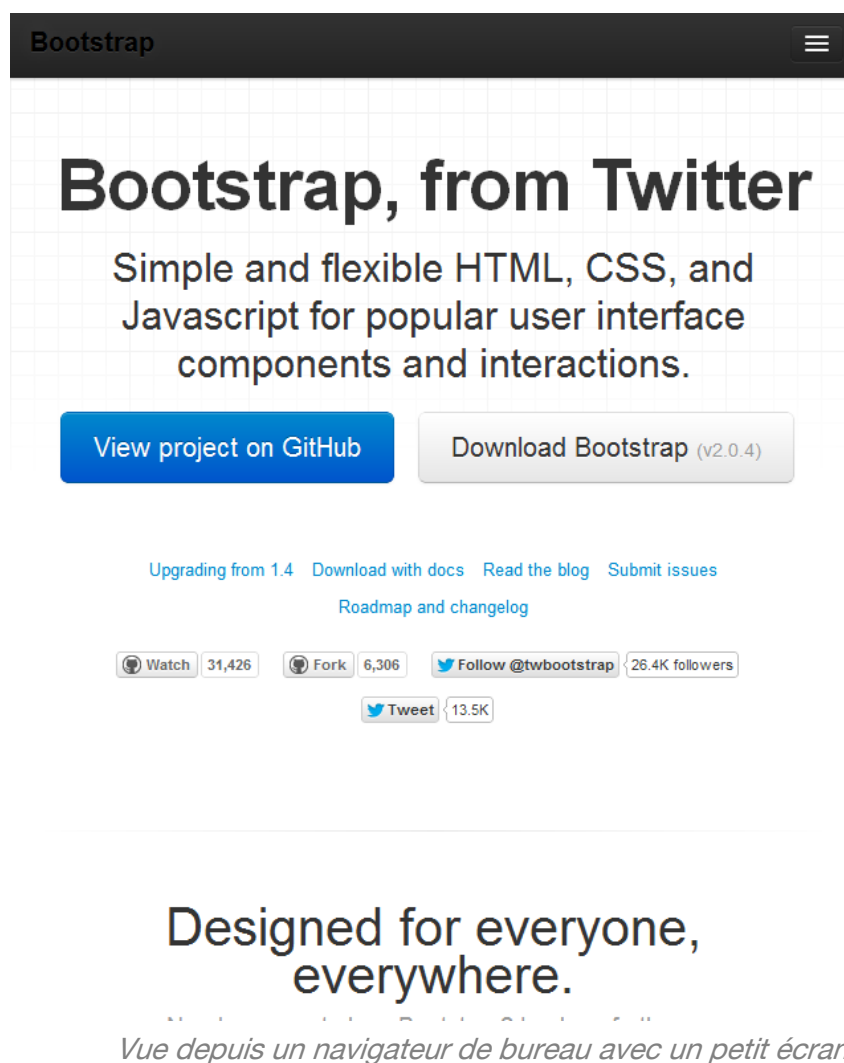
Quelque part, ceci nous pousse à réfléchir à l'interface et aux fonctionnalités de notre application. D'abord pour les smartphones ou les tablettes (où l'interaction et l'affichage sont beaucoup plus restreints), puis ensuite à penser à la version desktop.

Pour illustration, regardons le framework JavaScript Bootstrap de Twitter, qui a été conçu avec jQuery, et qui est orienté responsive design.

Sur un écran large, voici comment s'affiche la page :



Et voici la même page avec une résolution plus petite :



Première constatation : le menu du haut s'est réadapté. Il apparaît uniquement lorsque nous cliquons sur le bouton du haut.

Et si nous continuons à diminuer la résolution :



L'interface s'est de nouveau adaptée. Nous sommes cependant restés sur la même application, nous n'avons pas été redirigés, et le contenu reste le même.

En pratique, il faut définir à l'avance les zones d'affichage de notre site, et appliquer des règles CSS en fonction de la résolution de l'écran (sur nos éléments, nos images), afin de réagencer notre page.

8 MODELE D'ARCHITECTURE MOBILE

Deux approches sont possibles: soit une application en pur Web Mobile, soit une application avec une solution backend. Dans le premier cas, nous devons définir une architecture serveur exposant des Web Services, puis définir la logique métier en pur Web Mobile. Dans le second cas, tout cela se fait côté serveur.

Dans le premier cas, pour une plus grande souplesse, facilitée d'utilisation, d'implémentation et une meilleure maintenabilité, nous préconisons la mise en place d'une architecture stateless, basée sur REST.

REST est une architecture d'applications permettant d'exposer des Web Services. Ces derniers permettent de transmettre des données sous différents formats, aussi bien du JSON, que du XML, du `www-form-urlencoded`, ou encore son propre format de données. De plus, c'est une approche dite « ressource ». De ce fait, pour accéder à une ressource, il faut lancer une requête sur une URL formatée d'une manière spécifique.

Cette approche permet de découper facilement son application, et de très nombreux frameworks (notamment dans le monde Java/JEE) permettent la conversion automatique des données (du format d'échanges au modèle Java).

Dans les deux cas, il faudra utiliser un framework permettant d'agencer convenablement notre site Web Mobile. Pour ce faire, nous pouvons utiliser un framework externe, comme jQuery Mobile, Bootstrap. Dans le cas de l'approche serveur, nous pouvons également utiliser le framework qui est intégré, ou le mixer avec un framework externe.

8.1 Le MultiPage versus le SinglePage

Dans le développement Web Mobile, nous avons deux approches : le MultiPage versus le SinglePage.

Le MultiPage, correspond à la navigation assez classique des applications Web, où nous trouvons des liens vers d'autres pages. Parfois, nous injectons de l'HTML via Ajax en supplément.

Le SinglePage, lui, a pour objectif de ne jamais changer de page ! Le but étant d'éviter de recharger les ressources sur le réseau (même si nous avons les ressources dans le cache, le navigateur a parfois tendance à lancer une requête pour chacune d'entre elles pour voir s'il n'y a pas de mise à jour), de permettre de faire de jolies transitions lorsque nous changeons de page (chose qui n'est pas possible avec l'approche précédente).

Mais le plus important est sans doute de pouvoir préserver son contexte JavaScript ! En effet, en navigant d'une page à l'autre, nous perdons nos instances JavaScript. Nous pouvons toujours stocker des données dans le cache de session

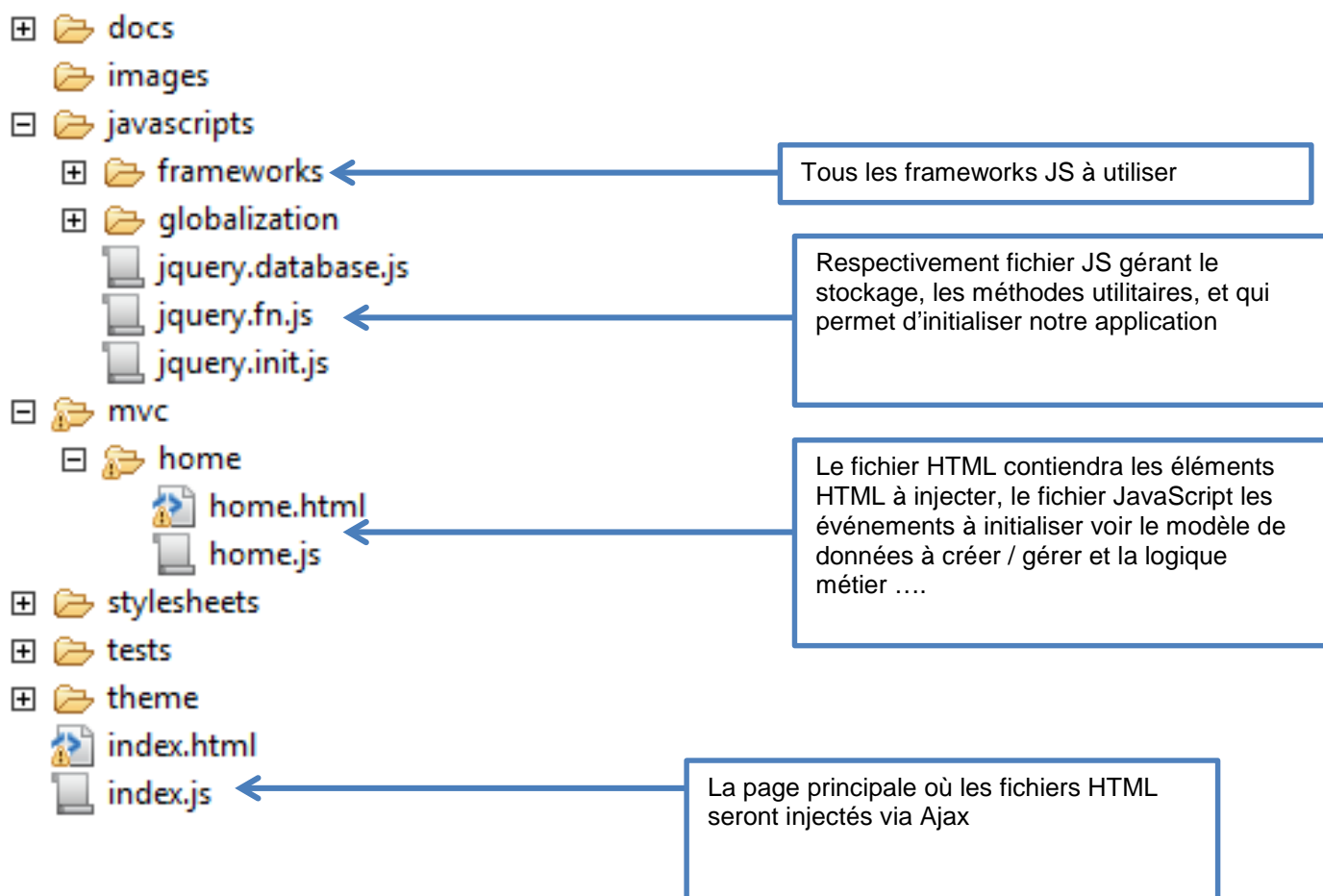
HTML5, mais nous aurons alors à reconstruire nos objets JavaScript, nos écouteurs, nos plugins, ... Et cela peut être coûteux d'un point de vue temps et mémoire.

Il est à noter que le SinglePage a deux approches. La première consiste à tout déclarer dans sa page HTML. La seconde à créer des fichiers HTML qui seront chargés en Ajax, puis injectés dans la page (jQuery Mobile, par exemple, sait gérer ces deux cas en même temps).

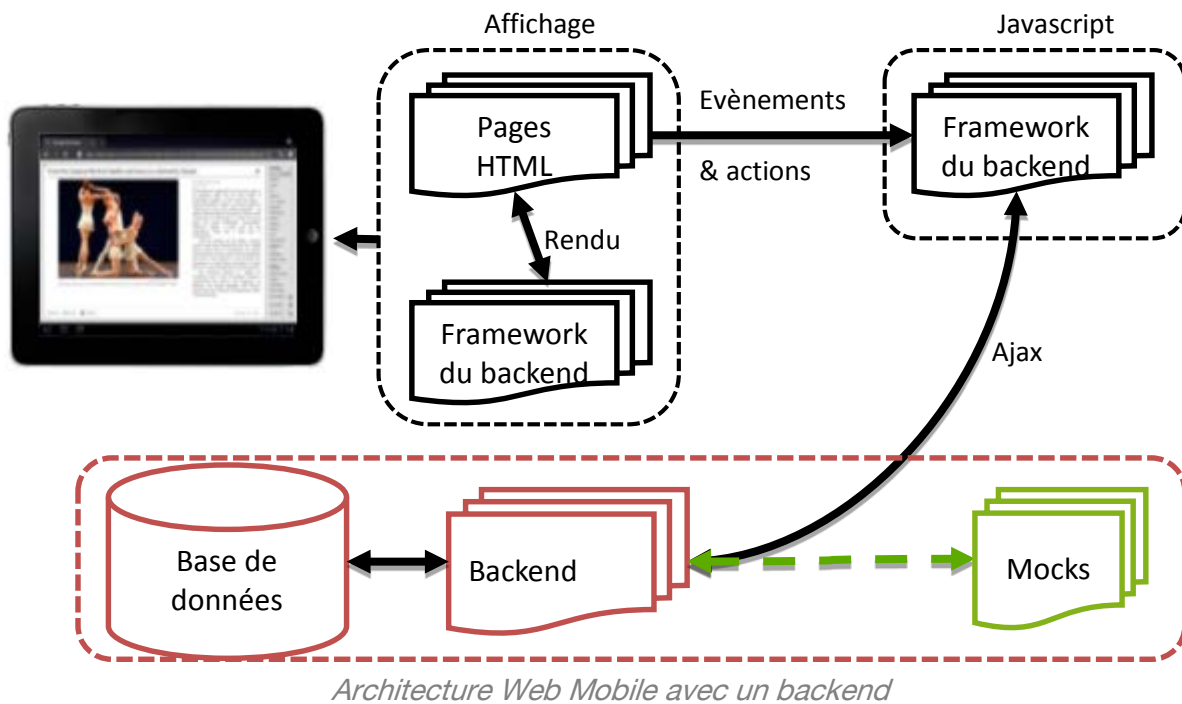
D'un point de vue optimisation, il est généralement déconseillé de tout mettre dans une page pour le cas où une application serait riche en écrans. Plus la page inclut des éléments dans le DOM, plus le navigateur mettra du temps à réagir. Cela est principalement vrai pour les applications dites hybrides, où les composants de « WebView » sont plus lents que les navigateurs Web natifs.

La seconde approche du SinglePage est intéressante dans le cas où nous aimerions découper l'architecture de notre application (tout avoir dans le même document va plutôt rendre la lisibilité ardue et la maintenance compliquée).

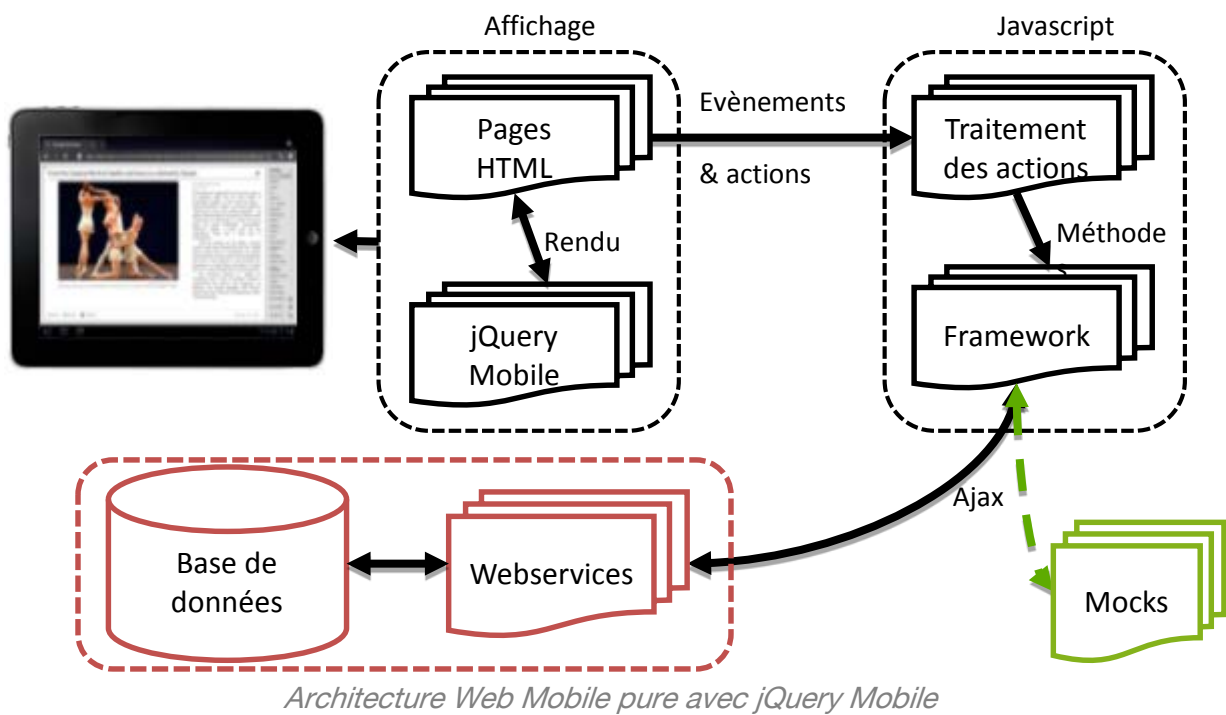
Néanmoins, le SinglePage implique d'avoir des règles de navigation, ce que ne possèdent pas tous les frameworks. Nous pouvons toutefois en définir un, l'étendre pour qu'il puisse importer des fichiers JavaScript, et imaginer, par exemple, de concevoir une architecture de la sorte :



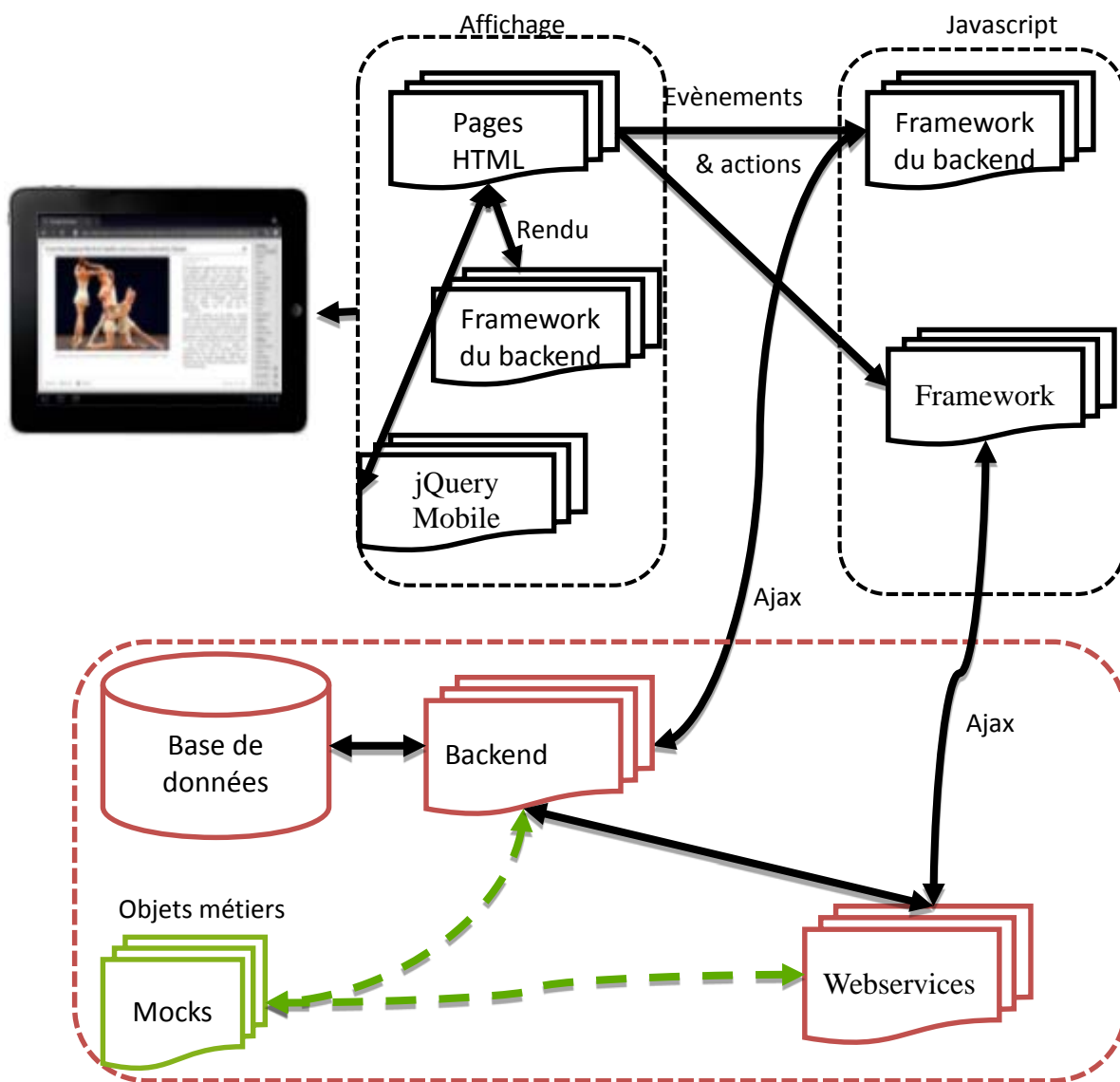
8.2 Architecture Web mobile avec une technologie serveur



8.3 Architecture Web mobile avec une technologie cliente



8.4 Architecture Web mobile avec une technologie cliente et serveur



Architecture Web Mobile frontend et backend

9 FRAMEWORKS ADDITIONNELS

9.1 Less / Sass / Compass

Ces frameworks permettent de créer des feuilles CSS à partir de langages de haut niveau. Nous pouvons alors avoir des constantes, des méthodes, de l'héritage ... et ainsi générer le code CSS correspondant.

Ces outils sont très utiles pour nous permettre de gérer manuellement beaucoup de CSS !

Un exemple concret (avec Less) :


```
@myColor: #000000;


.corner(@radius, @color: @myColor) {
  border-radius: @radius @radius;
  -webkit-border-radius: @radius @radius;
  -moz-border-radius: @radius @radius;
  border: 1px solid @color;
}

div {
  background-color: #eeeeee;
  padding: 15px;
  text-align: center;
  width: 200px;
  .corner(15px);
}

button {
  color: #4D926F;
  .corner(5px, #4D926F);
}
```

Nous obtenons alors :

```
Matched CSS Rules
media="screen"
div {
  background-color: #EEE;
  padding: 15px;
  text-align: center;
  width: 200px;
  border-radius: 15px 15px;
  -webkit-border-radius: 15px 15px;
   -moz-border-radius: 15px 15px;
  border: 1px solid black;
}
```

```
Matched CSS Rules
media="screen"
button {
  color: #4D926F;
  border-radius: 5px 5px;
  -webkit-border-radius: 5px 5px;
   -moz-border-radius: 5px 5px;
  border: 1px solid #4D926F;
}
```

Less propose une approche qui permet d'interpréter le fichier de haut niveau avec du JavaScript (nous pouvons également le générer complètement).

Sass / Compass (Compass étant une extension de SaaS) nécessitent en revanche de le générer au préalable, mais sont beaucoup plus puissants.

Le principe de ce genre de frameworks est de faciliter la mise en place d'une architecture MVC dans nos applications Web, en offrant une approche de data-binding.

A l'heure actuelle, nous avons trois grands leaders :

- Knockoutjs : <http://knockoutjs.com>
- Backbonejs : <http://backbonejs.org/>
- Angularjs : <http://angularjs.org/>

Les deux premiers frameworks existent déjà depuis plusieurs années et fonctionnent plutôt bien.

Backbonejs se pilote entièrement en JavaScript (aucun attribut HTML spécifique), s'intègre bien avec jQuery, et surtout permet de créer des modules (que nous pouvons charger avec des frameworks comme RequireJS). Mais le data-binding n'est pas géré.

Knockoutjs s'utilise à la fois dans l'HTML et le JavaScript. Il a une gestion d'événements qui met à jour notre modèle et les différents éléments HTML (data-binding). Néanmoins, il est plus gourmand en termes de mémoire.

Angularjs est un framework de Google qui est sorti récemment. Il est encore assez jeune et, malgré la documentation, assez difficile à appréhender. C'est pourquoi il faut se cantonner aux deux premiers frameworks.

Voici des exemples permettant de réaliser l'écran suivant :

First name: **Johnfdfd**

Last name: **Doe**

First name:

Last name:

Avec Knockoutjs :

```
<!DOCTYPE html>
<html>
<head>
  <title>Test</title>
  <meta charset="utf-8">
  <script type="text/javascript" src="jquery-1.7.1.min.js"></script>
  <script type="text/javascript" src="knockout-2.0.0.js"></script>
  <script type="text/javascript">
    (function($){
      $(function(){
        ko.applyBindings(new function() {
          this.firstName = ko.observable("John");
          this.lastName = ko.observable("Doe");
        });
      })(jQuery);
    })(jQuery);
  </script>
</head>
<body>
  <p>First name: <strong data-bind="text: firstName"></strong></p>
  <p>Last name: <strong data-bind="text: lastName"></strong></p>
  <br />
  <p>First name: <input data-bind="value: firstName" /></p>
  <p>Last name: <input data-bind="value: lastName" /></p>
</body>
</html>
```

Avec Backbonejs :

```
<!DOCTYPE html>
<html>
<head>
  <title>Test</title>
  <meta charset="utf-8">
  <script type="text/javascript" src="jquery-1.7.1.min.js"></script>
  <script type="text/javascript" src="underscore-min.js"></script>
  <script type="text/javascript" src="backbone-min.js"></script>
  <script type="text/javascript">
    var userInstance, readViewInstance, editViewInstance;

    (function($){
      $(function(){
        var User = Backbone.Model.extend({
          defaults: { firstName: 'John', lastName: 'Doe' }
        });

        userInstance = new User();

        var ReadView = Backbone.View.extend({
          el: $('span'),
          initialize: function(){
            _bindAll(this, 'render');
            this.model = userInstance;
            this.model.on('change', this.render, this);
            this.render(); // not all views are self-rendering.
          },
          render: function(){
            $(this.$el.get(0)).text(this.model.get("firstName"));
            $(this.$el.get(1)).text(this.model.get("lastName"));
          }
        });

        var EditView = Backbone.View.extend({
          el: $('input'),
          events: {
            "keypress": "contentChanged"
          },
          initialize: function(){
            _bindAll(this, 'render');
            this.model = userInstance;
            this.render(); // not all views are self-rendering
          },
          render: function(){
            $(this.$el.get(0)).val(this.model.get("firstName"));
            $(this.$el.get(1)).val(this.model.get("lastName"));
          },
          contentChanged: function(e) {
            this.model.set("firstName", $(this.$el.get(0)).val());
            this.model.set("lastName", $(this.$el.get(1)).val());
          }
        });

        readViewInstance = new ReadView();
        editViewInstance = new EditView();
      })(jQuery);
    }</script>
  </head>
  <body>
    <p>First name: <strong><span></span></strong></p>
    <p>Last name: <strong><span></span></strong></p>

    <br />
    <p>First name: <input type="text" /></p>
    <p>Last name: <input type="text" /></p>
  </body>
</html>
```


Et avec Angularjs :

```
<!DOCTYPE html>
<html ng-app>
<head>
  <title>Test</title>
  <meta charset="utf-8">
  <script type="text/javascript" src="angular-1.0.1.min.js"></script>
</head>
<body>
  <p>First name: <strong>{{firstName}}</strong></p>
  <p>Last name: <strong>{{lastName}}</strong></p>

  <br />
  <p>First name: <input ng-model="firstName" /></p>
  <p>Last name: <input ng-model="lastName" /></p>
</body>
</html>
```

En règle générale, ces frameworks se marient plutôt bien avec les grands frameworks frontend mobile.

Dans le cadre d'un développement Web Mobile, les outils utilisés pour le développement Web classique peuvent être repris.

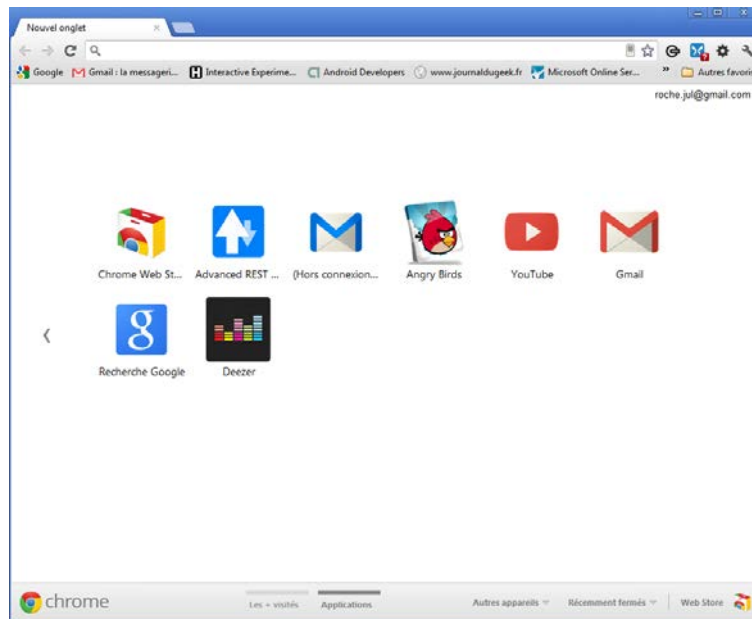
Toutefois, voici ce que nous utilisons chez VISEO :

- L'IDE Aptana 3
 - C'est un bundle Eclipse particulier, gérant les technologies et les frameworks Web modernes, comme HTML5, CSS3, JavaScript, jQuery, Saas, CoffeeScript ...
 - Il est soit téléchargeable en standalone, soit en tant que plugin Eclipse (dans le second cas, nous pourrions également avoir les plugins pour SVN, Git, Maven ...)



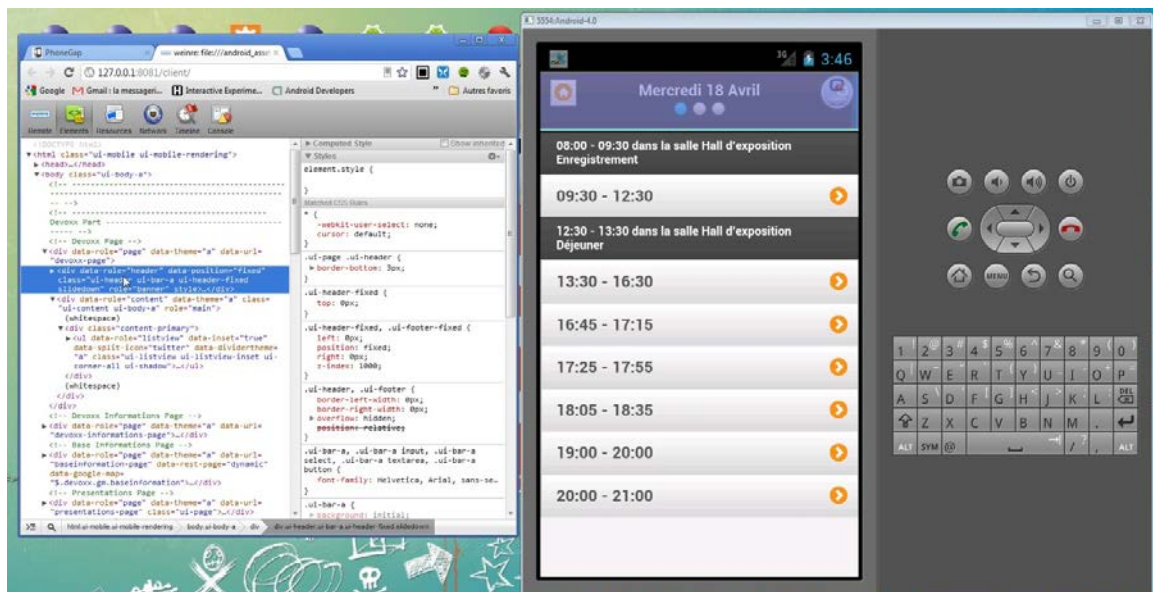
Aptana 3

- Le navigateur Chrome
 - La plupart des smartphones / tablettes ont le moteur de rendu Webkit, moteur qu'utilise Chrome.
 - Ainsi, il est toujours mieux de commencer par développer sur son navigateur de bureau Chrome, et ensuite de voir le rendu sur les différents supports mobiles (via des émulateurs ou des appareils).
 - Il est le navigateur qui possède la meilleure compatibilité HTML5
 - De plus, ce dernier possède une palette d'outils assez riches et fonctionnels (accès / manipulation du DOM et du CSS, analyseur réseau, débogueur JavaScript, analyseur de ressources et de données des cookies, localStorage, WebSql et IndexedDB ...)



Google Chrome

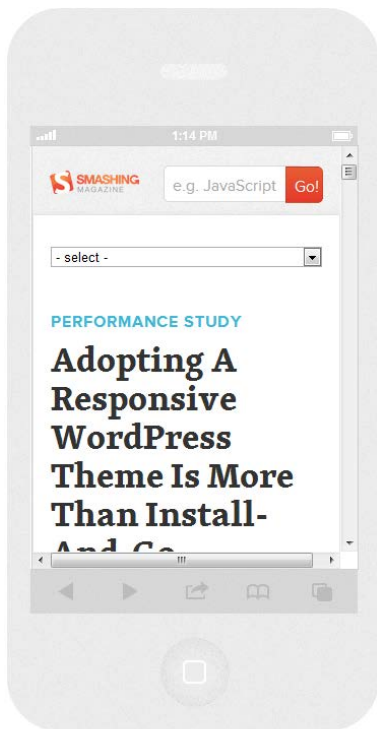
- Weinre
 - C'est un outil OpenSource fournit par PhoneGap
 - Il permet d'analyser les pages à distance, permettant ainsi de trouver les causes de problèmes d'affichage



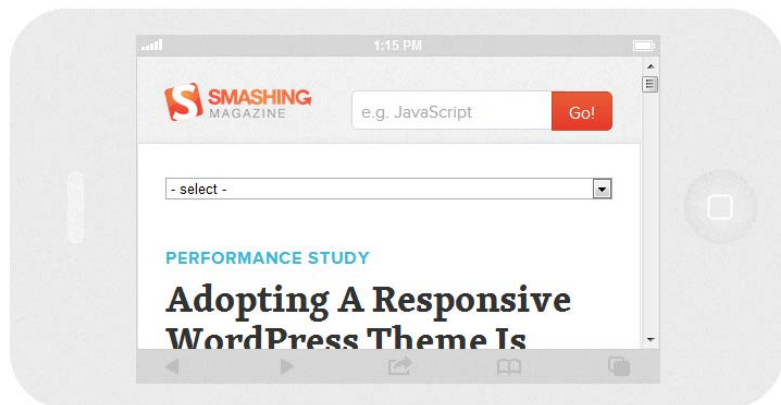
Weinre en action

- Un serveur Web pour héberger les pages, comme Apache, Ngnix, Tomcat ...

En complément, nous utilisons généralement des pages ayant des templates / gabarits de smartphones et de tablettes, afin d'avoir un premier aperçu du rendu. Mais il existe aussi des sites pour cela, comme <http://www.responsinator.com/>

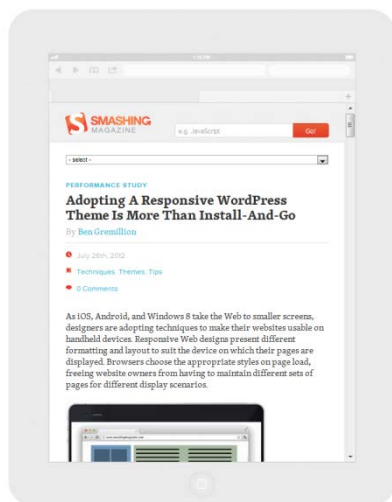


iPhone portrait 320 x 480

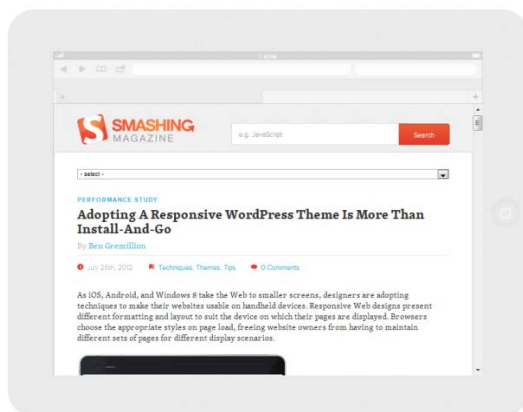


iPhone landscape 480 x 320

Aperçu du site <http://www.smashingmagazine.com> sur smartphones ...



iPad portrait 768 x 1024



iPad landscape 1024 x 768

... et sur tablettes !

11 CONCLUSION

Comme vous pouvez le voir, nous avons abordé beaucoup de points. Nous avons vu :

- Les différentes solutions de conception orientées mobile,
- Les « pour » et les « contre » du développement Web mobile,
- Les points essentiels pour tout développement mobile, tels que :
 - Les plateformes et les versions de ces dernières à viser,
 - Les points d'ergonomies à connaître,
- Et enfin, les architectures, les frameworks et les outils tournant autour du développement Web mobile.

Certes, tout cela est riche et condensé. Néanmoins, cela fait partie d'une expérience mobile nécessaire et incontournable, quelle que soit la solution visée, afin de mener à bien son projet mobile.

Disons-nous cela : la documentation sur le Web mobile est légion, les acteurs nombreux, et les frameworks en nombre et de qualité.

Ainsi, nous pouvons considérer qu'il n'existe aucun frein à la réalisation de son application Web mobile, à condition de respecter l'arbre décisionnel vu dans ce livre blanc.

12 A PROPOS DE VISEO

VISEO est une société de conseil, de services et de formation, experte depuis 1998 dans les architectures objet (Java, JEE, .Net) et web, les applications mobiles (Android, iPhone, HTML5...), les méthodes agiles, la modélisation (UML) et l'assistance à maîtrise d'ouvrage (AMO).

Nous accompagnons les plus grandes entreprises dans la construction et l'évolution de leur Système d'Information grâce à la complémentarité de nos activités : conseil technologique (définition d'architectures, choix d'outils ou de technologies), conseil méthodologique (mise en place de méthodes agiles Unified Process, XP, Scrum et amélioration du cycle en V), conseil en urbanisation (définition d'architecture d'entreprise, cartographie métier, cartographie applicative, urbanisation du système d'information), formation aux nouvelles technologies, édition et distribution de logiciels.

VISEO est une filiale de VISEO, groupe international indépendant et premier spécialiste en France à avoir développé un niveau d'expertise aussi élevé sur toutes les dimensions du système d'information (progiciels de gestion, outils de gestion de la relation client, Business Intelligence, développement web et objet, génie logiciel).

Découvrez l'offre d'accompagnement VISEO : [*Web et mobile : expertise technique & développement*](#)

Pour en savoir plus : www.VISEO.com

Vous avez un projet ?
Contactez-nous !

Avec 12 années d'expertise dans l'animation et la commercialisation de formations spécialisées dans les nouvelles technologies, VISEO a développé une offre de très haute qualité basée sur un socle théorique solide et proposée au niveau national au sein de ses centres de formation à Paris, Lyon, Grenoble et Toulouse.

- Les formations VISEO vont à l'essentiel de la technologie et de la méthode afin de vous apporter un savoir-faire immédiatement opérationnel.
- Les formations VISEO ont été conçues afin que le rythme et la progression de l'apprentissage soient parfaitement optimisés.

VISEO propose 47 formations sur les nouvelles technologies et les méthodes agiles susceptibles d'être implémentées dans vos projets.

VISEO adapte le format de ses cours en fonction de vos besoins:

- Les formations sont animées en mode Interentreprises (à partir des dates catalogue pour différentes sociétés) ou Intra-entreprise (pour une seule société à la date de son choix).
- Notre équipe de consultants-formateurs peut adapter les formations standards selon le contexte client (métier ou technologique).
- Notre équipe vous propose des « formations coaching » afin d'aider l'équipe au démarrage du projet dans la prise en main des nouvelles technologies acquises.

L'équipe Formation VISEO est à votre écoute et sera ravie de vous accueillir dans l'un de ses 4 centres de formation.

Inscriptions et informations:

formation@viseo.com

Tel : 01 41 22 13 13 ou 04 72 33 68 67

Découvrez nos formations mobilité :

- [Architectures mobiles : principes et implémentation](#)
- [Développement Web Mobile : HTML5, PhoneGap](#)
- [Android : développement mobile](#)
- [IOS8 – Développement iOS 8 \(iPhone, iPad\)](#)



À PROPOS DE VISEO

VISEO EST LA PREMIÈRE SOCIÉTÉ DE SERVICES INFORMATIQUES EN FRANCE À AVOIR DÉVELOPPÉ UN NIVEAU D'EXPERTISE AUSSI ÉLEVÉ SUR TOUTES LES DIMENSIONS DU SYSTÈME D'INFORMATION (PROGICIELS DE GESTION, OUTILS DE GESTION DE LA RELATION CLIENT, BUSINESS INTELLIGENCE, DÉVELOPPEMENT WEB ET OBJET, GÉNIE LOGICIEL).

#viseospirit

VISEO
www.viseo.com