**Guide du développement.**

Table des matières

[Organisation des fichiers 2](#_Toc388523603)

[Pour reprendre le développement. 3](#_Toc388523604)

[L’architecture et l’organisation du code sources 3](#_Toc388523605)

[BD 3](#_Toc388523606)

[Les principales actions 4](#_Toc388523607)

[Ajout d’une nouvelle vue 4](#_Toc388523608)

[Ajouter un nouvel appel 4](#_Toc388523609)

[Code source 4](#_Toc388523610)

[Utilisation du framework Sencha Touch 5](#_Toc388523611)

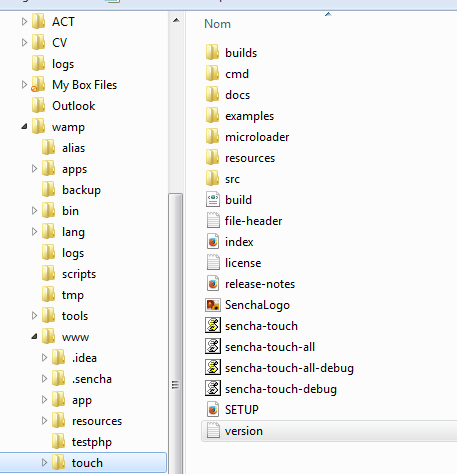
[Remarques 5](#_Toc388523612)

# Organisation des fichiers

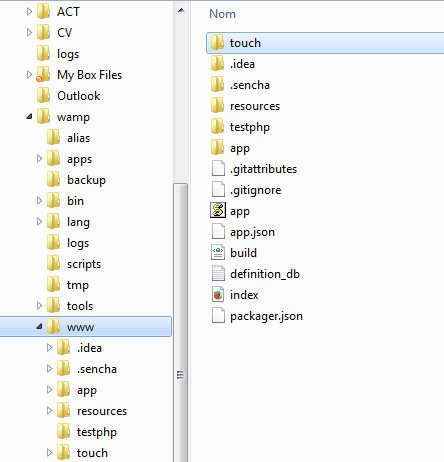
La structure de base était générée automatiquement par Sencha Touch.

Tous les fichiers concernant le projet sont commité et maintenu à jour sur GitHub. Sauf le répertoire touch\ qui comprend le framework Sencha Touch (version 2.2.1). Ce répertoire est assez volumineux, et nous ne sommes pas censé modifier son contenu, c’est la raison pour laquelle il n’est pas commité comme le reste du projet.

Le projet est présent sous wamp/www. Le répertoire touch/ doit être présenté sous wamp/www/.



Voici l’organisation du répertoire www.



# Pour reprendre le développement.

Pour reprendre le projet en cours de développement, il faut suivre les étapes suivantes :

* Démarrer Wampserver, l’icône de l’application doit être vert.
* Pour tester l’application sur navigateur, utiliser Chrome.
  + Lancer Chrome.
  + Entrer l’url <http://localhost>
  + Pour débugger, appuyer sur F12 pour voir les traces de l’exécution
* Pour tester l’application sur téléphone. Il faut entrer l’url du l’adresse IP de l’ordinateur

(<http://192.168.1.17> par exemple.) Pour connaitre l’ip exacte, vérifier dans le console de windows 🡪 ipconfig. Puis trouver l’adresse Adresse IPv4

Les codes sources sont stockés sous C:\wamp\www

# L’architecture et l’organisation du code sources

## BD

La base de données est sur le serveur Wamp dans une BD MySQL. Pour y accéder, utiliser l’url <http://localhost/phpmyadmin/>

La base est nommée « propection ». La table est nommée pcall.

Elle comprend 6 colonnes : ID (auto-incrémentale), pname, type, ctime(l’heure actuelle par défaut), result, comment.

## Les principales actions

### Ajout d’une nouvelle vue

Les vues sont toutes créées au moment où le controler de Main view est chargé.

Ensuite en fonction du bouton cliqué, la vue Main va lancer la fenêtre correspondante.

Chaque fenêtre se ferme avec Ext.Viewport.remove(Ext.Viewport.getActiveItem(), **false**) avec leur propre controler. Attention, l’argument **false** signifie que le vue est juste enlevée du pile des vues, mais elle n’est pas détruite.

### Ajouter un nouvel appel

pcall

Add

testphp

Call model

Call controler

Call view

Coté client Coté serveur

## Code source

Les fichiers javascript qui concernent l’application sont stockés sous C:\wamp\www\app

On y retrouve les quatre types d’élements : **Vue/Controleur/Model/Store**

Les « views » :

* Main : la fenêtre du démarrage
* Call : la vue qui permet de consulter ou de créer un nouvel appel.
* Stat : la vue du graphe des statistiques.
* CallListPanel : la liste des appels déjà passés.

Les vues utilitaires :

* TimePicker : le composant intégré dans la vue Call qui permet de choisir l’heure d’appel.
* Time

Les contrôleurs :

* Main : Le controleur associé à la vue « Main »
* Call : Le controleur associé à la vue « Call »
* Stat : Le controleur associé à la vue « Stat »

# Utilisation du framework Sencha Touch

Avec le framework sencha touch, il est possible de générer automatiquement la squelette d’une application web avec une organisation MVC.

Les éléments principaux de l’application sont définis dans app.js

# Remarques

* Pourquoi les modifications dans app.js n’est pas prises en compte ?
* Il faut vider le cache du navigateur pour que les modifications dans app.js soient prises en compte
* Il faut éviter de donner les mêmes noms de variables même dans les fichiers .js différents. Ce ne sont pas des objets comme dans le langage java. (Ça peut créer de confusion pour l’interpréteur ?)
* Pour accéder à un attribut d’un objet, il vaut mieux utiliser le plus possible les setters et les getters. En effet, il y a certains traitements supplémentaires qui sont ajoutés qui sont nécessaires, même si nous pouvons récupérer directement les références des objets (C’est le cas pour le rafraîchissement du CalllistStore)
* A savoir que pour le langage javascript, les affectations « = » passent toujours la référence de l’objet mais jamais une copie. Donc les actions sur l’objet pointé par une référence concernent donc l’ensemble de références qui lui pointent dessus.