

Passage d'informations sur le projet du Bus Web Service



Table des matières

Objectif :	2
Installation du web Service :	2
Technologies :	2
Fonctionnement :	3
Format JSON	4
TCP	4
Limites de fonctionnement :	5
Tests sans carte	5
Objectifs de chaque page	6
Diagramme global	7
Aperçu des pages	8

Objectif :

Récupérer/afficher/envoyer des données vers un circuit électronique via un web service.

Installation du web Service :

Installer un serveur Apache si vous voulez installer le web service en local. J'ai installé pour ma part Xampp : <https://www.apachefriends.org/fr/> ou ajouter les fichiers sur un serveur web déjà existant, auquel cas sauter cette étape d'installation web service.

Une fois le logiciel installé, démarrez le serveur Apache dans le panel de contrôle. Vous aurez à démarrer le serveur MySQL quand il faudra cette partie là.

Pour l'emplacement des fichiers, le plus simple est de mettre le répertoire du web service sous xampp/htdocs/

Il suffira donc d'insérer le lien dans votre navigateur le lien <http://localhost/bus/> ou <http://127.0.0.1/bus/>

Un problème à ce stade ? Il ne devrait pas y avoir d'erreur dès le lancement. Si toutefois il y a des éléments qui ne chargent pas :

- Soyez certain que le serveur apache a été lancé, aller sur <http://127.0.0.1/> , il devrait s'afficher une page.
- Sous le navigateur Chrome, faites Ctrl+Maj+I, puis regardez dans l'onglet « console » les erreurs affichées. Agissez en conséquence, peut être n'arrive t'il pas à charger un fichier, modifiez donc le chemin dans le code source.

Technologies :

Langages: HTML5, CSS3, JavaScript, Framework JQuery, PHP. Il faudra écrire des requêtes SQL pour les futurs développements.

Starter : J'ai choisi Web Starter Kit développé par Google

<https://developers.google.com/web/starter-kit/> . Pour le centre de contrôle, avoir un site web responsive design n'a pas d'utilité, mais j'ai choisi ce starter pour avoir une bonne base.

Framework JQuery : pas indispensable, mais j'ai préféré l'utiliser au JavaScript notamment pour l'animation (<http://api.jquery.com/animate/>) et l'AJAX (<http://api.jquery.com/jquery.ajax/>)

Icônes/bouton : <https://www.polymer-project.org> pour le jolie design et sans charger d'image. Cependant sous IE, les icones ne se chargent pas.

Carte : Map Google JavaScript V3 <https://developers.google.com/maps/documentation/javascript/>

Barre d'avancement du bus : HTML5 Canvas <http://www.alsacreations.com/tuto/lire/1484-introduction.html> et <http://www.html5canvastutorials.com/>

Fonctionnement :

La page map.html (redirigé sur `index.php?p=map.php`, Cf les dernière ligne du « **.htaccess** » RewriteEngine et la page « **index.php** » pour comprendre la manipulation dynamique des pages) affiche sur une carte google toutes les minutes via une requête AJAX vers `Position_ajax.php`, la position des bus en fonction de leur longitude et latitude. J'ai utilisé l'API Javascript V3 qui correspond au plus proche à mon besoin : afficher des repères.

La page ligne.html affiche 6 données de chaque numéro de bus. Toutes les minutes via une requête AJAX, les nouvelles données sont récupérées et celles qui doivent être rafraîchies sur la page, une icône en forme de bus apparaît et s'anime pour disparaître au bout de 2 secondes, afin d'attirer l'attention à l'utilisateur. Cette page affiche aussi l'état d'avancement du bus sur sa ligne à l'aide d'un dessin en utilisant des canvas. J'ai essayé d'utiliser d'autres méthodes pour réaliser une barre d'avancement comme la balise `<progress>` mais elle ne m'a pas convenue et n'est pas aussi bien supportée sur divers navigateurs internet (<http://html5demo.braincracking.org/demo/progress.php>) que `<canvas>` (<http://html5demo.braincracking.org/demo/canvas.php>). De plus les canvas répondent bien au besoin que je me suis fixé d'afficher l'avancement du bus sur sa ligne.

Pourquoi utiliser de l'AJAX ?

Autant pour la page `map.php` que `ligne.php`, nous avons besoin d'avoir à minima toutes les minutes, les nouvelles valeurs des bus. Afin de ne pas recharger toute la page mais seulement les éléments (DOM) à modifier, nous utilisons des requêtes AJAX, c'est-à-dire

1. Un code javascript qui va faire une requête sur une autre page (`Position_ajax.php`).
2. `Position_ajax.php` va télécharger les fichiers depuis le FTP ayant en leur sein du texte en format JSON.
3. `Position_ajax.php` va rassembler les résultats et l'afficher sur sa page.
4. Le code javascript n'a plus qu'à lire ce qu'il y a sur `Position_ajax.php` et traite le résultat JSON.

Mode TestLocal/FTP/Socket : Le Web service a plusieurs modes de fonctionnement pour pouvoir être testé à plusieurs niveaux de développement, par défaut il est en mode FTP. Un mode peut être utilisé pour faire des tests sans passer par le FTP, avec un fichier dans le répertoire TestBench. Voir « **config.php** » et « **Position_ajax.php** » pour choisir le mode. J'ai écrit aussi une partie du code pour la connexion par socket, mais si cette dernière méthode est utilisée, il faudra l'intégrer.

Format JSON

Chaque fichier correspondant à un idBus est composé d'une série de données. La meilleure mise en place pour un traitement simple des données et l'évolution du nombre des données est d'utiliser le format JSON (<http://www.json.org/jsonfr.html>)

Les clés utilisés avec pour une valeur assigné :

```
{"state": char, "fillingRatio": int, "speed": int,  
"longitude": float, "latitude": float, "tempAir": int,  
"lastStation": char, "tempMotor": int, "idBus": char}
```

Le fichier Position_ajax.php, une fois les données des différents fichiers reçus concatène les résultats.
[{..., "idBus": "C1"}, {..., "idBus": "C2"}]

TCP

Utiliser le protocole FTP est une des solutions qui peut être utilisé pour le projet de bus, il est cependant certain que les technologies mis en place pour ce genre de projet n'utilisent pas cette solution.

A terme si possible, il faudrait que les différents systèmes (bus et/ou station) envoient des informations directement au serveur, car faire tourner un serveur en envoyant des requêtes à toute la flotte me paraît inconcevable et c'est un processus utilisant des ressources inutiles.

Le Web service doit pouvoir également envoyer des requêtes et en attendre les réponses.

Il est également nécessaire de crypter, sécuriser les données et les commandes.

Pour cela la solution possible en php est de passer par des sockets (<http://php.net/manual/fr/ref.sockets.php>). Il est aussi possible de passer par la méthode cURL (<http://php.net/manual/fr/book.curl.php>) simple d'utilisation par ses options (login,password par exemple), mais cela dépend comment les systèmes électroniques pourront gérer les requêtes.

Un exemple de génération d'une requête du web service vers un système électronique a été créé. Dans le fichier « config.php » mettre la valeur 'socket' à 'MODE_COMM'. Un nouvel onglet 'TCP' apparaît sur le web service. Le cœur de la fonction se situe dans le fichier « **ajax.php** »

Il s'agit d'envoyer à l'adresse IP du système sur le bon port

```
$fp = fsockopen(IP SOCKET, PORT SOCKET)
```

Puis envoyer une commande et attendre la réponse

```
$response = fwrite($fp, "$query\n");
```

Enfin reste à traiter/afficher la réponse

```
echo fgets($response,4096);
```

Limites de fonctionnement :

Je n'ai évidemment pas tout implémenté, le web service a donc ses limites et n'est pas optimisé, notamment n'ayant pas mis en place de base MySQL.

Du côté de la carte, il peut y avoir un nombre illimité de bus, tous les bus ont cependant le même icône.

Du côté des lignes, il ne peut gérer que 2 bus. Si on en veut plus ou moins, il faudra générer le code différemment en php (en fonction du nombre de bus dans la base mysql par exemple) et toutes ont les mêmes arrêts.

Tests sans carte

Il est possible de tester le web service sans que la carte ne soit en fonctionnement.

En l'état, vous pouvez ajouter/modifier manuellement les fichiers sur le FTP (ou dans le répertoire TestBench si vous n'utilisez pas le mode FTP)

Pour un test plus simple, un outil écrit en Python V2 « TestBench/writeFile.py » peut être lancé (installer auparavant <https://code.google.com/p/pyscripter/> par exemple) et lancez le code qui simulera le déplacement d'un bus en longitude toutes les minutes. Il se connecte au FTP et met à jour le fichier. Libre à vous de modifier le code.

Objectifs de chaque page

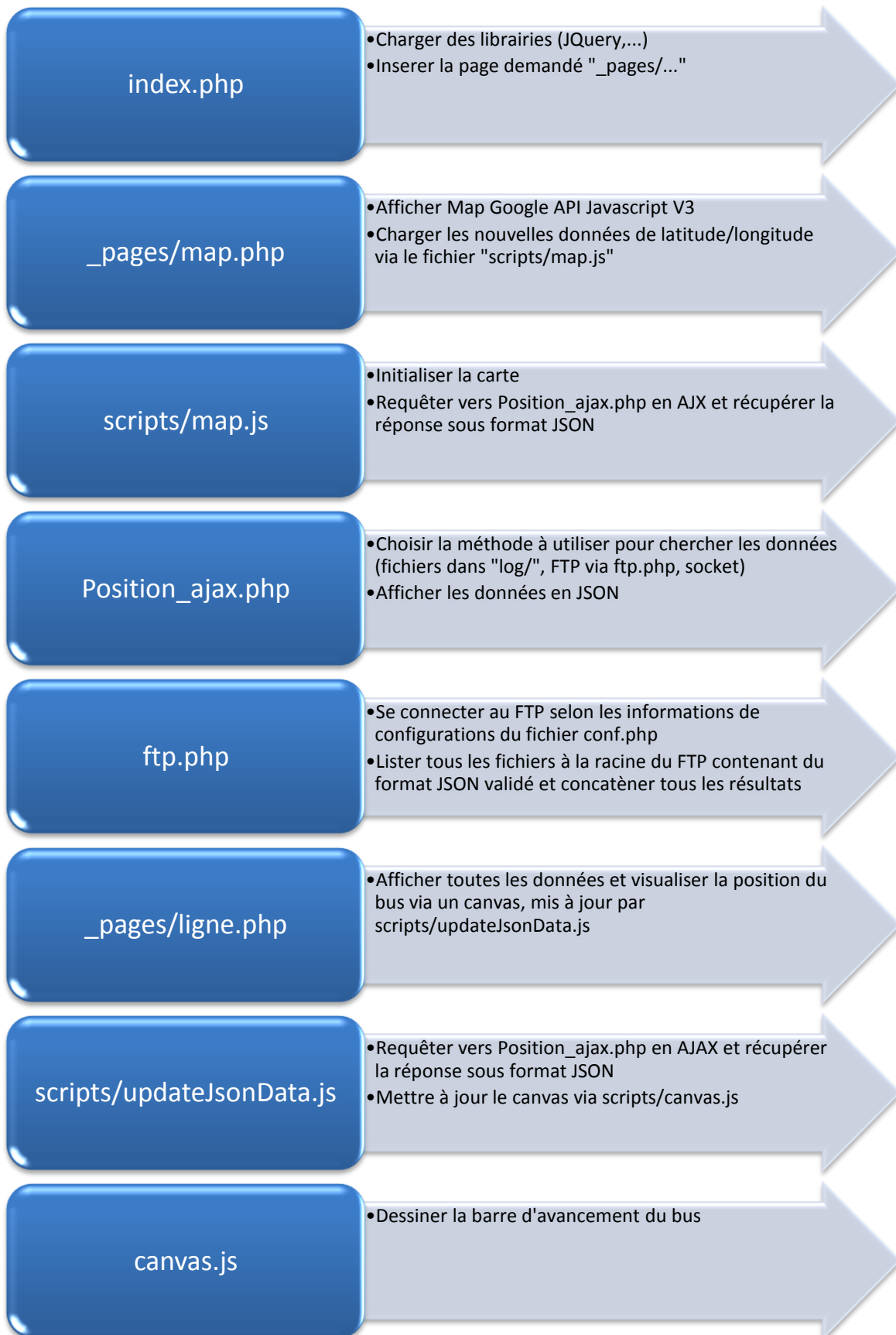
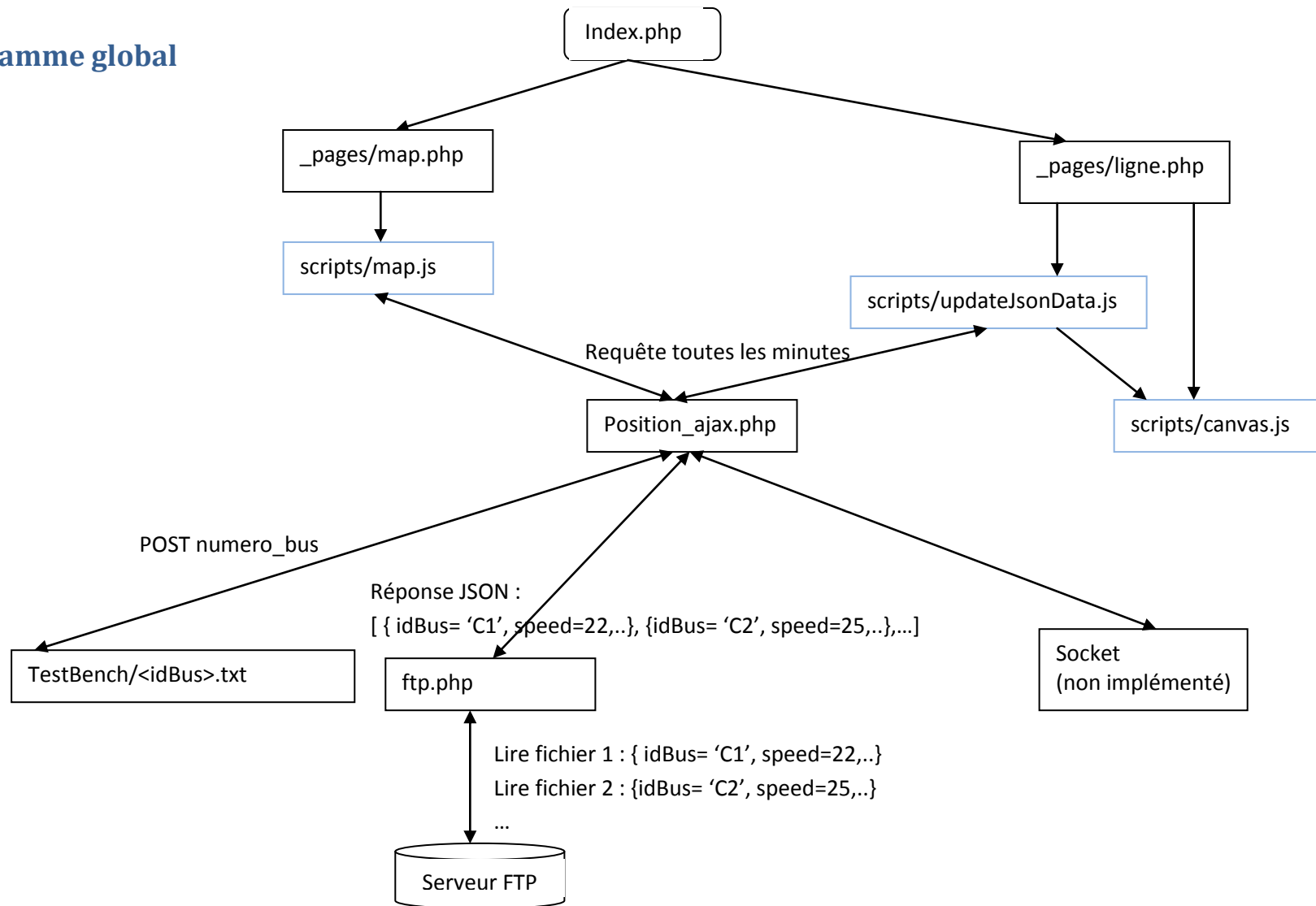


Diagramme global

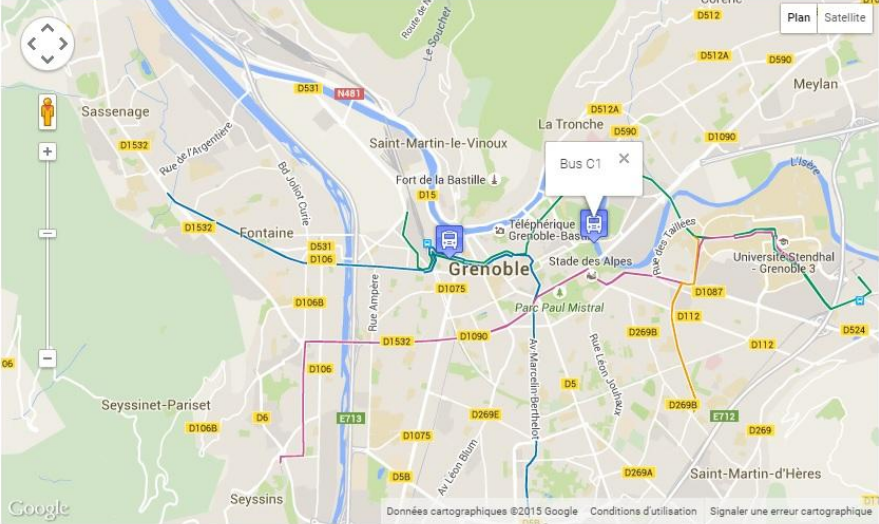


Aperçu des pages

_pages/map.php

Centre de contrôle Bus

CarteLigneTODOAide



PLEIN ÉCRAN


_pages/ligne.php

Centre de contrôle Bus

CarteLigneTODOAide

C1

- Etat : En service
- Mesure géographique : lat:45.189429800000006 long:5.741541300000009
- Taux d'occupation : 11%
- Vitesse et accélération : 25km/h
- Température moteur : 43°C
- Température climatisation : 18°C



MaupertusLes BâillèresMalacherGranierMeylan MariePiscine des BudoisLe BrêtLa raviréeAiguinards - HexagonePleine fleurieCarromerie - Jlle d'amourSablonGrenoble Hôtel de VilleChavantDocteur MartinVictor HugoDocteur MazetFélix VialletGrenoble GaresEmile GuenymardAragoJean Macé

C2