Institut Universitaire de Technologie
Aix*Marseille Université

RAPPORT DE STAGE

Diplôme Universitaire de Technologie Spécialité Réseaux et Télécommunications

Développeur web chez MAIN (BatterieDePortable.com)

Alexandre NEGREL

06/04/2020 - 14/06/2020

Responsable entreprise: Frédéric Quere
Responsable académique: Corinne Houssain

Table des matières

1 - Introduction

De nos jours, la majorité des personnes possédant un smartphone ou un ordinateur se connecte au moins une fois par jour à internet. C'est ce réseau, qui nous permet d'accéder à des services tels que le streaming, les courriers électroniques, et bien plus encore. Ces services sont appelés des **services web**, ils sont développés par les développeurs web qui chaque jour améliorent et facilitent notre vie.

La création de services qui révolutionnent notre quotidien est l'une des raisons qui m'a poussé vers le développement logiciels avec l'envie de mieux comprendre les outils qui nous entourent. J'ai choisi de faire mon stage chez MAIN (BatterieDePortable.com) afin de mettre à profit mes connaissances et d'en acquérir de nouvelles.

Mon employeur qui est aussi mon responsable de stage est chef et créateur de l'entreprise **MAIN**. M. Quere, était intéressé par mes compétences pour développer des applications web, de mon côté, les missions à effectuer répondaient à mes attentes en termes de stage et étant attiré par l'entreprenariat, le choix était naturel.

Dans un premier temps, nous aurons une présentation en détails de l'entreprise **MAIN**, ensuite nous verrons les objectifs à réaliser durant les 12 semaines de stages. Ensuite, une comparaison entre le travail réalisé et le cahier des charges sera apportée et nous conclurons par une synthèse du travail, des résultats et des problèmes rencontrés.

2 - L’entreprise

Figure 1: Logo BatterieDePortable.com

MAIN est une entreprise 100% française basée à Aubagne, elle est spécialisée dans la vente en ligne de batteries et chargeurs pour ordinateurs portables, smartphones, outillages portatifs et caméras embarquées. Fondée en 2007, l’entreprise de monsieur Quere est aujourd'hui un grand acteur en Europe avec plus de 500 000 clients, particuliers et entreprises.
L’**EURL* MAIN** réalise un chiffre d’affaire d’environ 800 000 € chaque année avec seulement un employé. Le chef d’entreprise s’occupe de :

- Développer le site
  - Mise en page du site pour qu’il soit visuellement beau et intuitif
  - Gestion de la base de données qui contient le catalogue de produit et les comptes clients
- Web marketing
  - Le **SEO*** afin d’avoir une meilleure visibilité et donc plus de clients
  - La publicité
- Gérer le serveur qui comprend :
  - Un **serveur web Apache2** pour envoyer les pages web aux clients
  - Une **base de données MySQL** pour stocker le catalogue et les comptes clients
- La comptabilité

L’employé de son côté est responsable de :

- Gérer le service après-vente pour satisfaire les clients en toute circonstance
- La préparation des commandes

BatterieDePortable.com est donc une **TPE***, cette dernière est très soucieuse de ses clients

avec un service après-vente de très bonne qualité. J'ai d'ailleurs moi aussi contribué à améliorer le service client comme nous le verrons par la suite.
Malgré la situation exceptionnelle que nous vivons en France actuellement à cause de la crise sanitaire du COVID-19, monsieur Quere a accepté de maintenir mon stage en télétravail et je tiens à le remercier.

3 - Objectifs du stage

Comme nous l'avons vu précédemment, l'entreprise cherche constamment à améliorer son service et cela passe par une meilleure communication avec les clients. Ainsi ma première mission est d'ajouter le support des **notifications push** afin de pouvoir, par exemple, envoyer une notification dès l'arrivée d'un colis en point relais. Ces dernières doivent fonctionner sur le site version mobile indépendamment du smartphone et du navigateur utilisé. Pour mener à bien cette mission, outre les compétences classiques requises pour toutes les missions, **PHP* HTML***, **CSS***, **JavaScript*,** il est nécessaire de savoir utiliser les **API*** standard du web ou, le cas échéant, lire et comprendre la documentation.

Ma seconde mission consiste à ajouter la barre de recherche sur les pages **AMP*** qui n'était pas présente jusqu'à présent à cause des restrictions qu'elles imposent. Avec cet ajout, même les clients mobiles peuvent effectuer une recherche sur n'importe quel produit présent dans le catalogue depuis la page d'accueil. Afin d'ajouter cette fonctionnalité, il faut connaître la technologie **AMP,** savoir faire une requête **AJAX*** et **SQL*** (en **PHP**) pour récupérer les produits qui correspondent à la recherche dans la base de données.

La troisième mission est de transformer le site en ce qu'on appelle une **PWA***, ce n'est ni un standard ni une norme mais un "concept". Les applications web dites *progressives* sont des applications qui peuvent apparaître à l'utilisateur comme des applications natives en combinant les différentes fonctionnalités offertes par les navigateurs modernes. Par ailleurs, mes projets personnels (dont des **PWA**) m'ont permis d'être plus facilement accepté en tant que stagiaire chez **MAIN**. En plus des compétences des missions précédentes, il est nécessaire de savoir utiliser les **API** modernes des navigateurs.

Enfin, ma dernière mission consiste à ajouter un nouveau mode de livraison, les points relais **Chronopost**. Le client pourrait ainsi sélectionner un point relais proche de chez lui et suivre l'avancement de la livraison. Pour ce faire, il faut des connaissance en **PHP**, connaitre le protocole **SOAP*** pour interagir avec leur service web. Pour la mise en page, des connaissances en **HTML**, **CSS** et **JavaScript** sont requises.

4 - Résultats

4.1 - Notifications Push

Les notifications push sont le futur du web, elles permettent une meilleure communication avec le client, que les e-mails qui ne perçoivent que 2 à 3% de réponse de nos jours. Pour intégrer cette fonctionnalité à notre site web, nous n'avons pas d'autre choix que d'utiliser les **API** standard du web car c'est le navigateur qui permet à l'utilisateur d'accéder à notre site et c'est aussi ce dernier qui affiche la notification lorsque notre code le demande. Ainsi, si le navigateur n'implémente pas cette **API** standard, il nous est **impossible** d'afficher des notifications chez le client. Voyons maintenant comment intégrer les notifications au site.

L'intégration nécessite donc l'utilisation de ces 3 **API** standard du web :

- Notification **API**
- Service Worker **API**
- Push **API**

La première permet **l'affichage de notification** peu importe l'appareil (ordinateur, smartphone, ...) et le navigateur (chrome, firefox, safari, ...) utilisé, tant que l'on est sur la page du site web, les notifications peuvent être affichées.

La seconde permet d'**exécuter en arrière-plan** du code (**des workers***), l'exécution se fera indépendamment de la page web, même si elle est fermée, ainsi on peut, par exemple, **localiser** un utilisateur et son appareil tant que son navigateur est ouvert et peu importe le site web auquel il se connecte.

Enfin, la dernière permet l'envoi de **messages push** (messages envoyés depuis le serveur web destiné à un seul appareil précis) à un **worker** qui utilisera le message comme bon lui semble. Dans notre cas le message push contiendra le contenu de la notification. Nous avons désormais les pièces pour ajouter cette fonctionnalité au site, il nous reste à les assembler.

La première étape est de demander à l'utilisateur s'il souhaite autoriser notre site web à afficher des notifications, lorsque la demande est faite, l'utilisateur voit apparaître une pop-up du navigateur similaire à celle ci-dessous.

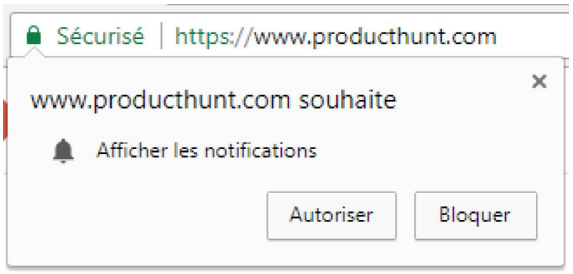


Figure 2: Pop-up Google Chrome pour afficher des notifications.

L'utilisateur a le choix et s'il refuse, le navigateur ne permettra pas l'affichage de notifications. La majorité des utilisateurs y voit un moyen supplémentaire d'être noyé de notifications à l'instar des spams d'e-mails, c'est pourquoi, il est recommandé d'expliquer à quoi servent les notifications que nous allons envoyer. Ainsi, nous avons ajouté une autre pop-up qui s'affiche avant celle du navigateur (*note:* le texte sera peut-être remplacé par un autre plus explicite une fois en production).

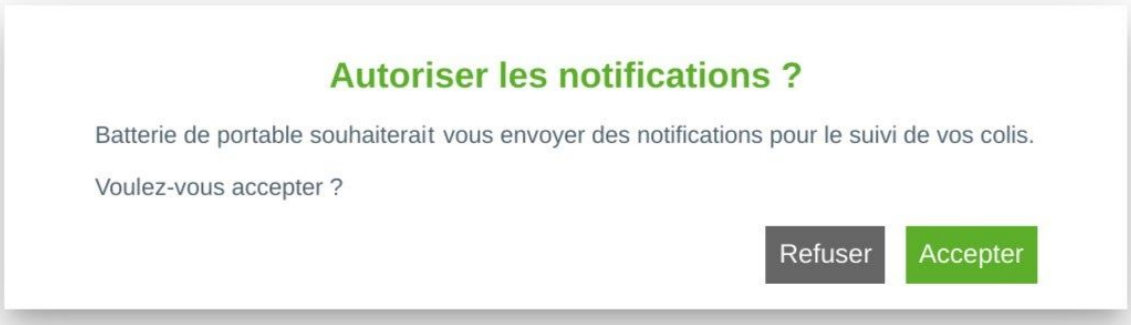


Figure 3 Pop-up BatterieDePortable.com pour afficher des notifications.

Une fois que l'on est capable d'afficher des notifications, nous devons maintenant permettre l'envoi de messages **push** entre le serveur et le client. Pour des raisons de **sécurité**, l'**API** push a été conçue pour ne pas nous donner un accès direct à l'appareil du client mais à un **service push** (cf. RFC 8291) qui sert de lien anonyme entre le client et le serveur. Chaque navigateur a son propre service push, ce qui peut engendrer des complications si le standard n'est pas respecté à la lettre (e.g. firefox mobile). Le fonctionnement est assez simple, si le client n'est pas déjà abonné (l'abonnement est propre à chaque site web) au service push et que son navigateur est compatible, on prend un abonnement. Ce dernier est lui aussi **chiffré** avec la **clé publique unique du serveur Apache2** pour que les services push (détenus par Google et autres géants de l'Internet) ne puissent lire le contenu des messages push ou même identifier l'appareil abonné. Voici un schéma récapitulatif:

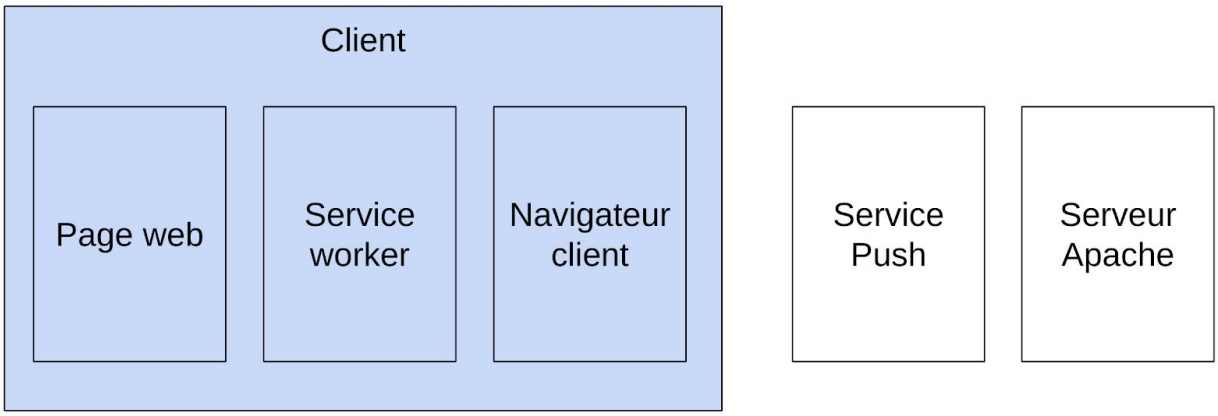


Figure 4: Vue d'ensemble des push notifications

Sur la figure ci-dessus on peut voir un **service worker** chez le client, en réalité c'est le service worker que l'on abonne au **push service**. C'est lui qui continuera de fonctionner et recevra les messages push une fois l'onglet du site fermé, c'est d'ailleurs sa seule utilité. En s'abonnant au push service, comme décrit dans l'API push, un abonnement nous est retourné, ce dernier contient une **clé de chiffrement**, un **endpoint** qui est simplement une **URI*** unique à laquelle on peut envoyer des messages chiffrés avec la clé de chiffrement. L'abonnement retourné est ensuite envoyé à notre serveur **Apache2** qui le sauvegarde ensuite dans la **base de données** pour pouvoir l'utiliser plus tard.

Ainsi, si l'on souhaite afficher une notification push, il suffit d'envoyer un message chiffré au **endpoint**, si l'utilisateur est hors-ligne, le message est sauvegardé par le service push jusqu'à la prochaine connexion de l'appareil utilisateur qui récupérera le message (s'il n'a pas expiré) et affichera une notification en utilisant le contenu de ce dernier. Mis à part les cas particuliers, l'ensemble fonctionne comme décrit ci-dessus. Evidemment, une page d'administration est dédiée à l'envoi de notifications push.

L'ensemble du système décrit ci-dessus a été réalisé par mes soins et j'ai rencontré quelques difficultés. Pour la partie client, étant relativement expérimenté, l'implémentation s'est passée sans accroche, les **API** que j'ai utilisé sont toutes très bien documentées et de multiples exemples existent sur internet. Malgré tout, il se trouve que l'application firefox pour smartphone ne recevait pas les messages push à cause d'un bug de leur côté. Pour résoudre ce problème, il existe une "technique" qui réduit légèrement la sécurité du chiffrement du message push mais permet l'envoi de ce dernier. Finalement, pour le côté serveur, ça a été plus compliqué, il se trouve que le serveur qui fait fonctionner le site utilise une ancienne version de **PHP** (5.5). Afin d'envoyer des messages push il est nécessaire de chiffrer ce dernier, des bibliothèques existent pour faciliter ce processus. N'étant pas un cryptographe, l'utilisation de ces dernières m'est

donc indispensable. Malheureusement, elles ne sont pas compatibles avec la version de **PHP** utilisée en production et il nous est impossible de mettre à jour **PHP** car cela nécessite la réécriture d'une grande partie du site. Une seule solution, ajouter un deuxième serveur web avec la dernière version de **PHP**, évidemment on ne peut exécuter un second serveur **Apache2** sur la même machine, il nous faut donc une **VM***. L'inconvénient de la **virtualisation** est la lourdeur et la perte de performance, comparé à une machine physique, la virtualisation est tellement lourde qu'elle n'est pas *rentable* dans notre cas. J'ai donc proposé l'utilisation d'un autre procédé *similaire* à la **virtualisation** nommé la **conteneurisation.** Contrairement à la virtualisation, cette dernière s'appuie sur le système d'exploitation hôte et n'en simule donc aucune (voir ci-dessous).

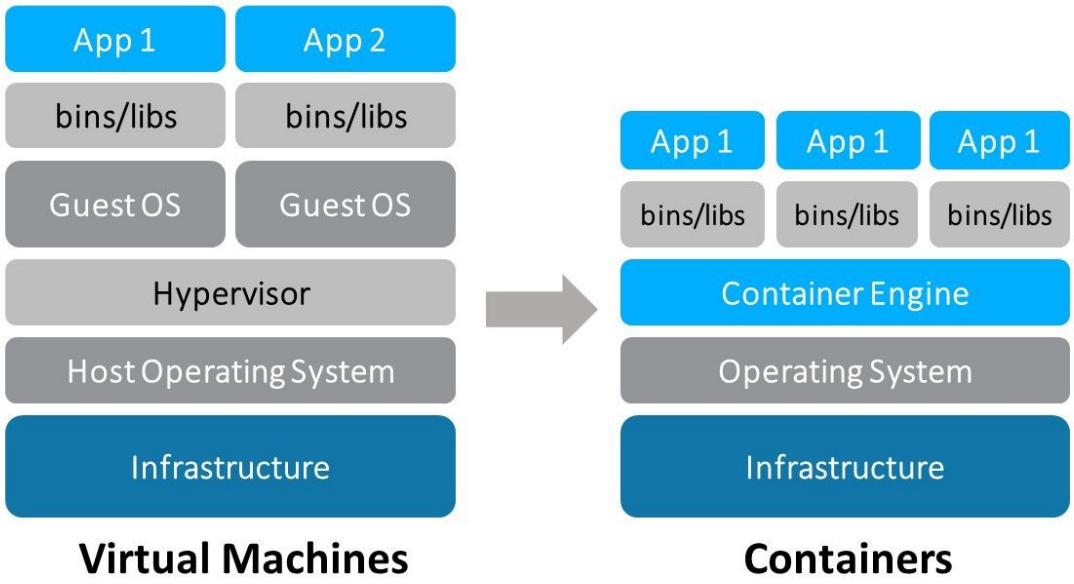


Figure 5: Schéma de machine virtuels et conteneurs

Cette unique différence fait que les deux technologies ne fonctionnent pas du tout de la même manière, en réalité les conteneurs sont extrêmement légers et performants. Au repos, le conteneur consommait environ **30 Mb** de mémoire vive soit **moins de 1%** pour le serveur de 32 Gb utilisé, avec une machine virtuelle il faudrait compter au **minimum** entre **512 Mb et 1 Gb** de mémoire vive par machine virtuelle. Pour déployer un conteneur, j'ai suggéré l'utilisation de **Docker** un outil qui est littéralement en train d'exploser, sa popularité n'a pas cessé de croître depuis 2013 lorsqu'il est devenu open-source. C'est l'un des outils les plus utilisés pour le cloud de nos jours car il **facilite le déploiement** d'applications vers n'importe quelle machine qui supporte Docker (Windows, Linux, MacOS). Par ailleurs, Google utilise en interne une technologie similaire depuis plus d'une décennie et ils ont également contribué à l'écosystème Docker. Après une courte introduction à Docker, mon maître de stage a accepté, à

condition que je rédige une documentation, que l'on essaie cette solution, actuellement un conteneur avec la dernière version de PHP tourne sur le serveur de test. Comme les deux serveurs ne peuvent pas écouter sur le même port, le vieux serveur PHP continue d'écouter sur le port **HTTP*** (port 80) et **HTTPS** (port 443) tandis que le conteneur écoute sur le port 3000. Par soucis d'homogénéité, j'ai mis en place un ***reverse proxy*** (ou mandataire inverse) pour que les requêtes clients utilisent aussi le port **HTTP** standard. Le proxy redirige toutes requêtes commençant par "https://www.batteriedeportable.com/**php**7" ainsi, si à l'avenir d'autres parties du site sont développées en **PHP** 7, le code sera facile à maintenir et relire car il n'est plus nécessaire de se souvenir du port du conteneur et l'utilisation de ce dernier est explicitement écrite.

Une fois le conteneur déployé, le développement s'est passé sans accroche grâce aux formidables bibliothèques disponibles. Nous avons donc vu comment j'ai réalisé l'intégration des notifications push sur le site web maintenant passons à la barre de recherche.

4.2 - Barre de recherche

Les barres de recherche sont des composantes essentielles pour les sites web car elles facilitent la navigation et l'accès aux articles. Sur le site version mobile, le ***framework*** **AMP** est utilisé pour de meilleures performances de rendu et un meilleur référencement. Premièrement, nous verrons un peu plus en détails ce qu'est **AMP** puis le fonctionnement d'une barre de recherche classique que nous adapterons pour le site mobile.

Le projet **AMP** est un projet de **Google**, annoncé le 7 Octobre 2015 avec de multiples partenaires tels que **Twitter**, **LinkedIn** et **WordPress**. À l'origine, les pages **AMP** étaient affichées en haut des pages résultats de **Google Search** ce qui augmentait grandement la visibilité des sites utilisant cette technologie. Cette dernière se concentre sur les performances, ainsi, par exemple, tous les éléments des page **AMP** sont dimensionnés statiquement, et donc la mise en page (ou **layout**) est déjà réalisée avant le téléchargement des ressources (images, script, etc). Le schéma ci-dessous explique les différentes étapes par lesquelles un navigateur passe entre la réception d'une page HTML et l'affichage de cette dernière.

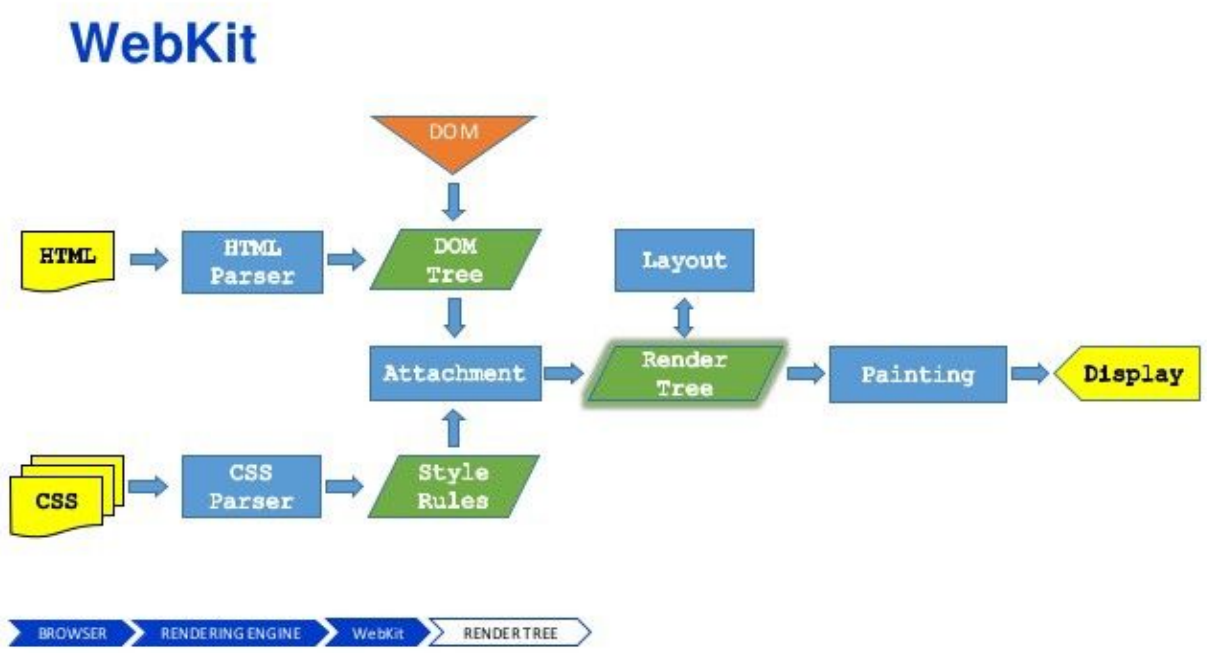


Figure 6: Étape d'un navigateur pour afficher une page

On dit qu'il n'y aura pas de "**reflow**" ou "**relayout**", ces opérations sont considérés comme *coûteuses* en terme de performances puisqu'elles correspondent à une remise en page du site web lorsqu'un élément est inséré ou change de taille par exemple. Pour qu'une page soit considérée comme valide, elle doit respecter de multiples règles afin de renforcer les performances, des composants **HTML** sont fournis par le framework pour

essayer de répondre aux besoins des utilisateurs. Je dis bien essayer, en effet les pages **AMP** restreignent les scripts **JavaScript** et donc la personnalisation de la page, c'est d'ailleurs la raison pour laquelle la barre de recherche n'était pas présente sur la version mobile. Depuis la sortie d'**AMP**, la création du site version mobile et mon stage, de nouveaux composants sont sortis, dont le composant ***amp-script*** que nous avons utilisés.

Tout d'abord, voyons comment fonctionne une barre de recherche en général. L'utilisateur entre le nom ou la référence du produit recherché dans la barre, puis une requête **HTTP** est envoyé au serveur (en **AJAX**) qui va comparer le catalogue de produit et la requête puis retourner une liste produit qui correspond à la recherche. Dans notre cas, la partie serveur était déjà écrite, une barre de recherche est déjà présente sur le site en version bureau, ainsi, initialement nous devions juste ajouter la barre en utilisant les composants **AMP**. Mais il s'est avéré que le serveur retournait des résultats incompatibles avec **AMP.** Les réponses du serveur correspondaient à du code **HTML** qui devait ensuite être inséré dans la page, or comme nous l'avons vu précédemment les scripts **JavaScript** sont limités. L'insertion direct **HTML** dans la page n'est pas autorisée car il existe une alternative plus performante : créer des éléments DOM depuis le navigateur client puis les insérer dans la page. Cette méthode est plus rapide et est même considérée comme une bonne pratique par les développeurs web, en effet elle évite au navigateur de lire (ou **parser** si on francise le mot) le code **HTML,** pour le transformer en élément DOM (voir figure 7). Ces derniers sont ensuite insérés dans la page si il n'y a pas d'erreurs. En insérant des éléments créés à l'aide de code **JavaScript** dans la pages et non de l'**HTML** on supprime toute erreur de syntaxe potentielle et on réduit le travail du navigateur. Nous avons donc changé le format de réponse du serveur pour le standard actuel : le **JSON***. La transition fût très simple car une fonction inclue dans PHP permet la conversion d'un **tableau associatif** (un type d'objet PHP) en **JSON**, le code final est même plus court. Après cette modification, la barre de recherche fonctionnait à l'exception des images des produits qui étaient retirées par **AMP**. Nous avons aussi supprimé l'ancien code pour la version bureau afin d'unifier la barre de recherche mobile et bureau, ainsi à l'avenir, il sera plus facile d'effectuer des modifications.

Hormis la découverte de la technologie **AMP** et ses contraintes, le développement fût relativement rapide. Je n'ai pas rencontré de grosses difficultés car la documentation **AMP** est très complète.

4.3 - Conversion/adaptation en PWA

Les **PWA** sont des applications web (donc des sites web) qui visent à se rapprocher voir **remplacer** les applications **natives** des smartphones (qui sont installables depuis le **Play Store** sous android ou l'**App Store** sous IOS). Pour ce faire, elles doivent atteindre le même niveau de qualité en fonctionnant et en s'adaptant aux fonctionnalités disponibles du navigateur et de l'appareil, elles doivent fonctionner en utilisant les **API Web modernes** tels que les **Service Workers**, être rapides, performantes, installables, re-engageables (par le biais de notifications par exemple) et fonctionner même **sans connexion** internet. Le site est déjà performant et re-engageable depuis la fin de ma première mission, nous devions donc le rendre installable et utilisable hors-ligne. Dans un premier temps, nous verrons comment rendre le site fonctionnel hors-ligne puis comment le rendre installable.

L'accès au site sans connexion internet fait partie des fonctionnalités qui sont requises par le navigateur pour rendre un site, installable. Nous avons donc ajouté le support du hors-ligne dans cet unique but mais cela reste tout de même utile comme nous le verrons par la suite. Pour permettre au site de fonctionner sans connexion, nous allons utiliser les **Service Workers**, déjà utilisés auparavant (première mission). Ces derniers ne seront pas utilisés pour réceptionner des messages push mais pour **intercepter** toute requête destinée au serveur. Si la réponse de la requête est présente dans le cache, le contenu du cache est renvoyé et la requête n'atteindra jamais le serveur. À l'inverse, si la réponse de la requête n'a jamais était mise dans le cache, cette dernière est envoyé au serveur. Cette **stratégie** de cache est appelée "***cache-first***", on priorise l'utilisation du cache au réseau (voir figure ci-dessous).

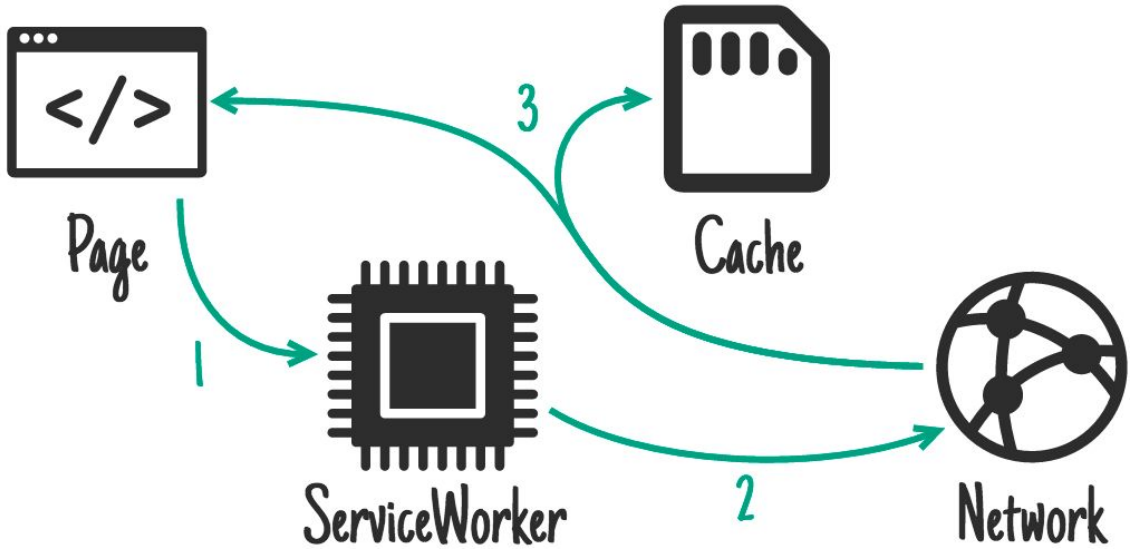


*Figure 7: Utilisation du cache avec la stratégie **cache-first***

Au-delà de rendre le site web installable, le cache permet de soulager les serveurs en limitant le nombre de requête et rendre l'accès ou la navigation plus rapide. Ainsi, le site est plus accessible, performant et fiable. Si une page n'est pas présente dans le cache, une page générique est utilisée pour prévenir l'utilisateur qu'il n'a pas d'accès internet. Nous avons beaucoup parlé de l'utilisation, de la lecture du cache mais nous n'avons pas vu comment les pages sont écrites dans ce dernier. Lorsqu'une requête est envoyé et que le cache ne contient pas la réponse à lui transmettre (ce qui est le cas lorsque l'utilisateur visite pour la première fois une page du site) une requête est envoyée au serveur, la réponse émise est ensuite écrite dans le cache. La figure ci-dessous illustre l'écriture dans le cache.

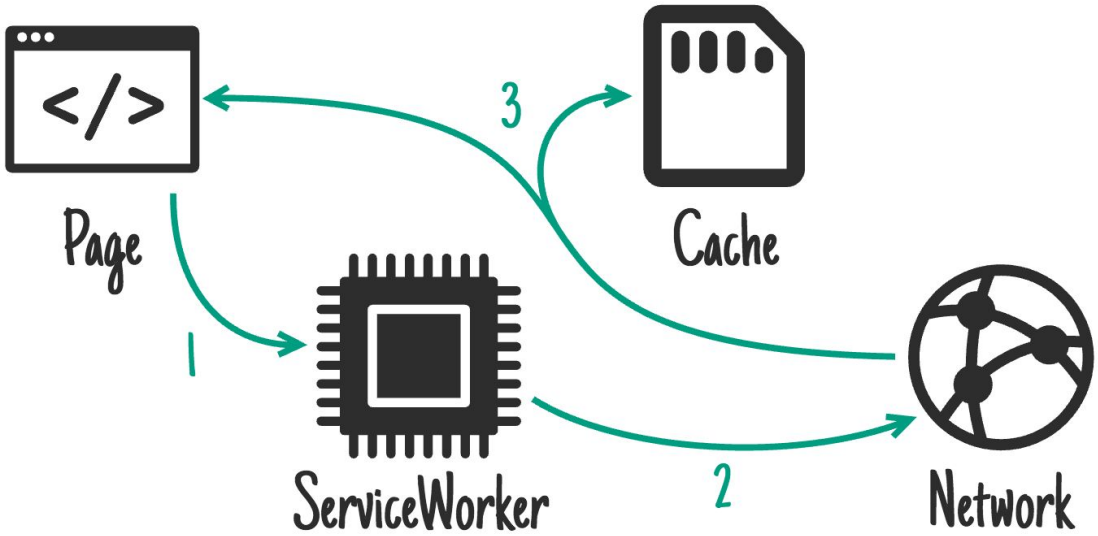


Figure 8: Écriture dans le cache

Maintenant que nous avons vu le fonctionnement du cache, voyons comment rendre le site installable .

Les **PWA** se veulent être le future du web, il faut donc satisfaire certains critères pour qu'une application web soit installable. Cette dernière doit avoir une connexion sécurisée (via **HTTPS**), fonctionner indépendamment du réseau (grâce au **cache** notamment), contenir un fichier descriptif de l'application (plus connu sous le nom de "***manifest.json***"), avoir une interface **adaptative** (peu importe le format d'écran utilisé) et fonctionner quelque soit le navigateur. Évidemment, le navigateur ne peut vérifier l'ensemble des critères ci-dessus, seul les 3 premiers critères sont nécessaires, toutefois, c'est la responsabilité du développeur de tous les respecter. Le site possède déjà une connexion sécurisée car c'est indispensable pour les paiements cependant le serveur de test n'était plus aux normes de sécurité puisque le **certificat TLS/SSL*** (qui certifie que la

connexion est sécurisé) avait expiré. Ce certificat est envoyé par le serveur lorsqu'une requête est émise par un client, il est ensuite utilisé pour initialiser une connexion sécurisé à l'aide du protocole **SSL**. Cette connexion ne peut être initialisée sans le certificat qui fait le lien entre une adresse web et une clé de chiffrement. Afin de vérifier l'intégrité du certificat (qui peut être faux ou avoir était révoqué) lui même, le navigateur vérifie l'authenticité de ce dernier auprès de l'autorité de certification qui l'a émis. Une fois le certificat vérifié, la connexion sécurisée est mise en place (cf. annexe page 36**)** pour plus de précision). Afin de pouvoir utiliser le serveur de test, il est nécessaire de renouveler le certificat. À l'origine, ils étaient payants et coûtaient entre 150 et 300 € par an, aujourd'hui il existe **Let's Encrypt,** un organisme à but non lucratif qui a fourni 180 millions de certificats **TLS** en 2019. C'est donc celui que nous avons choisis pour générer un nouveau certificat et l'installer sur notre serveur **Apache2**. Enfin, la création d'un fichier "***manifest***" au format **JSON**, ce dernier contient les informations nécessaires à l'installation du site, à savoir, l'icône, le nom de l'application, la couleur principale et d'autres informations essentiels. Voilà, les conditions nécessaires ont été remplies, nous pouvons maintenant installer notre site web.

Ayant déjà travaillé à de multiples reprises sur des **PWA** pour des projets personnels, je n'ai pas eu de problèmes à adapter le site. À noter que je n'avais tout de même jamais implémenté manuellement le cache.

4.4 - Chronopost relais

Ma dernière mission consistait à ajouter une option de livraison en **point relais** avec le service **Chronopost**. Chronopost est une société spécialisée dans la livraison express de colis aux entreprises et particuliers. En plus d'offrir un autre moyen de livraison, ils sont les n°2 en Europe et proposent un très bon service de livraison à l'internationale. Nous verrons dans un premier temps comment interagir avec le service web de **Chronopost** puis comment présenter les résultats avec une interface simple mais efficace.

L'interaction avec les services **Chronopost** se fait à l'aide du protocole **SOAP*** qui utilise le format **XML** pour structurer les données. Les messages sont composés d'une enveloppe qui contient des informations sur le message en lui-même afin de permettre son traitement et d'un modèle de données définissant le format du message à transmettre. Pour former correctement le message, on peut lire la documentation ou bien consulter le fichier **WSDL*** qui décrit l'ensemble des requêtes et réponses des services web proposés par Chronopost. C'est ce que nous avons utilisé pour former un message valide afin de récupérer les points relais disponibles et proches de l'adresse du compte client. Une fois la liste récupérée, il faut extraire les informations des points relais (présent dans la réponse au format **XML**). **PHP** intègre dans sa bibliothèque standard un client **SOAP** qui nous facilite grandement l'interaction avec les serveurs en générant la requête **XML** proprement. Le client facilite aussi la récupération des résultats. Initialement, une liste de point relais devait être affichée pour le client mais j'ai proposé, en plus de la liste, d'afficher sur une carte les points relais et le logement du client. Cette carte permettrait donc au client de mieux se repérer et comparer les distances avec son domicile. De plus, elle serait interactive, le client pourrait sélectionner le point relais en interagissant avec la carte ou la liste. Commençons par afficher la carte. À l'origine, **Google Maps** devait être utilisé mais nous avons finalement choisis d'utiliser une alternative gratuite et open source: **OpenStreetMap**. Pour créer et afficher la carte nous utilisons **Leaflet.js** qui est une bibliothèque **JavaScript** open source pour des cartes interactives adaptées aux mobiles. Ainsi, on utilise **Leaflet.js** pour créer la carte puis la télécharger chez **OpenStreetMap** et enfin l'afficher comme sur la figure ci-dessous.

Figure 9: Affichage d'une carte

La seconde étape consiste à implémenter le domicile du client sur la carte. L'adresse est sauvegardée dans le compte client or la carte fonctionne uniquement en coordonnées géographiques: latitude et longitude. Afin de récupérer les coordonnées géographiques d'une adresse, l'utilisation d'un autre service web s'impose, celui du gouvernement français qui met à disposition des **API** gratuites pour interroger les référentiels géographiques plus facilement. Ces dernières sont bien plus récentes et n'utilisent pas le protocole **SOAP** mais sont basées sur un style d'architecture logicielle nommé **REST***. Les services web dit "***RESTful***" sont extrêmement simples à utiliser et sont de nos jours plus populaires que le standard **SOAP**, les paramètres de la requête passent par l'**URL** et la réponse est généralement disponible sous de multiples formats. Nous avons choisi une réponse au format **JSON** car il est est beaucoup plus léger (donc performant) et plus facile à manipuler que les services **SOAP**. Nous pouvons désormais afficher le domicile du client et les points relais à proximité (voir figure ci-dessous).

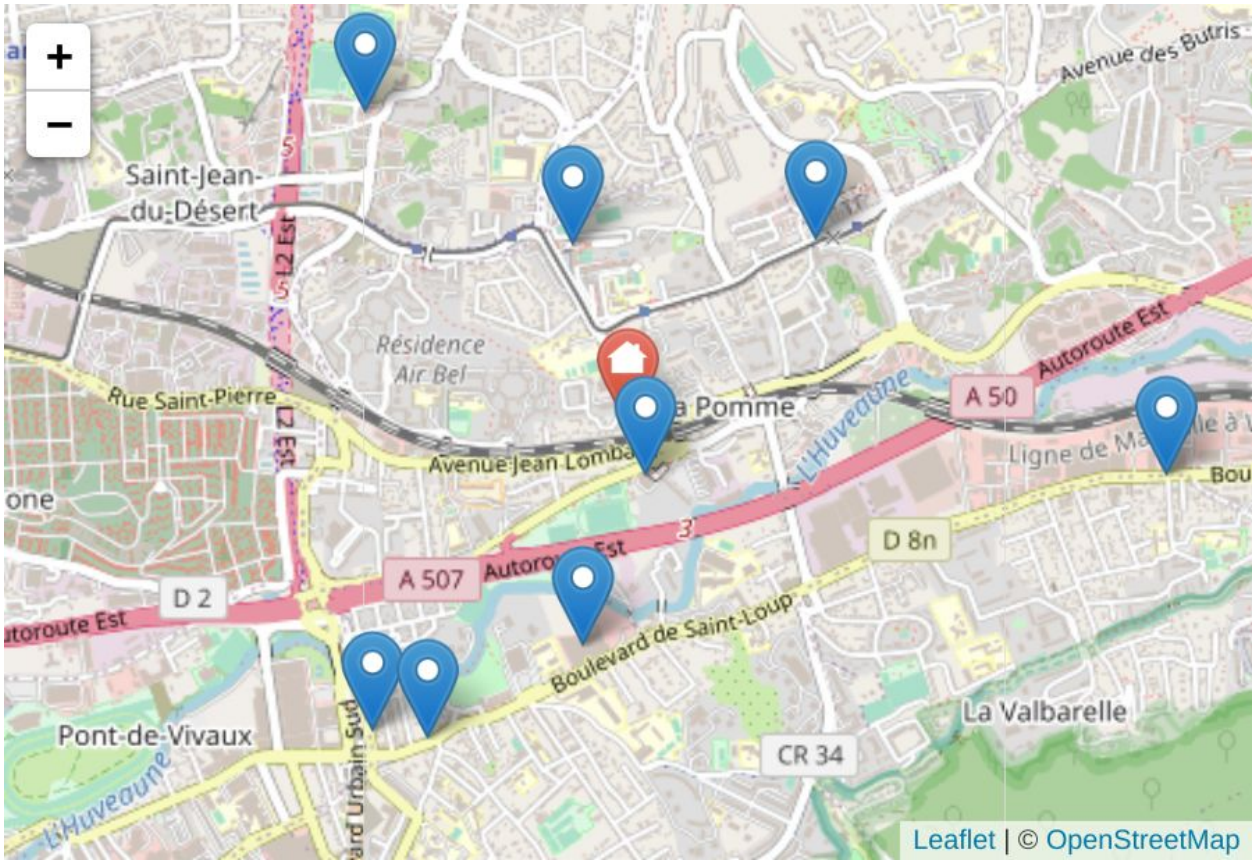
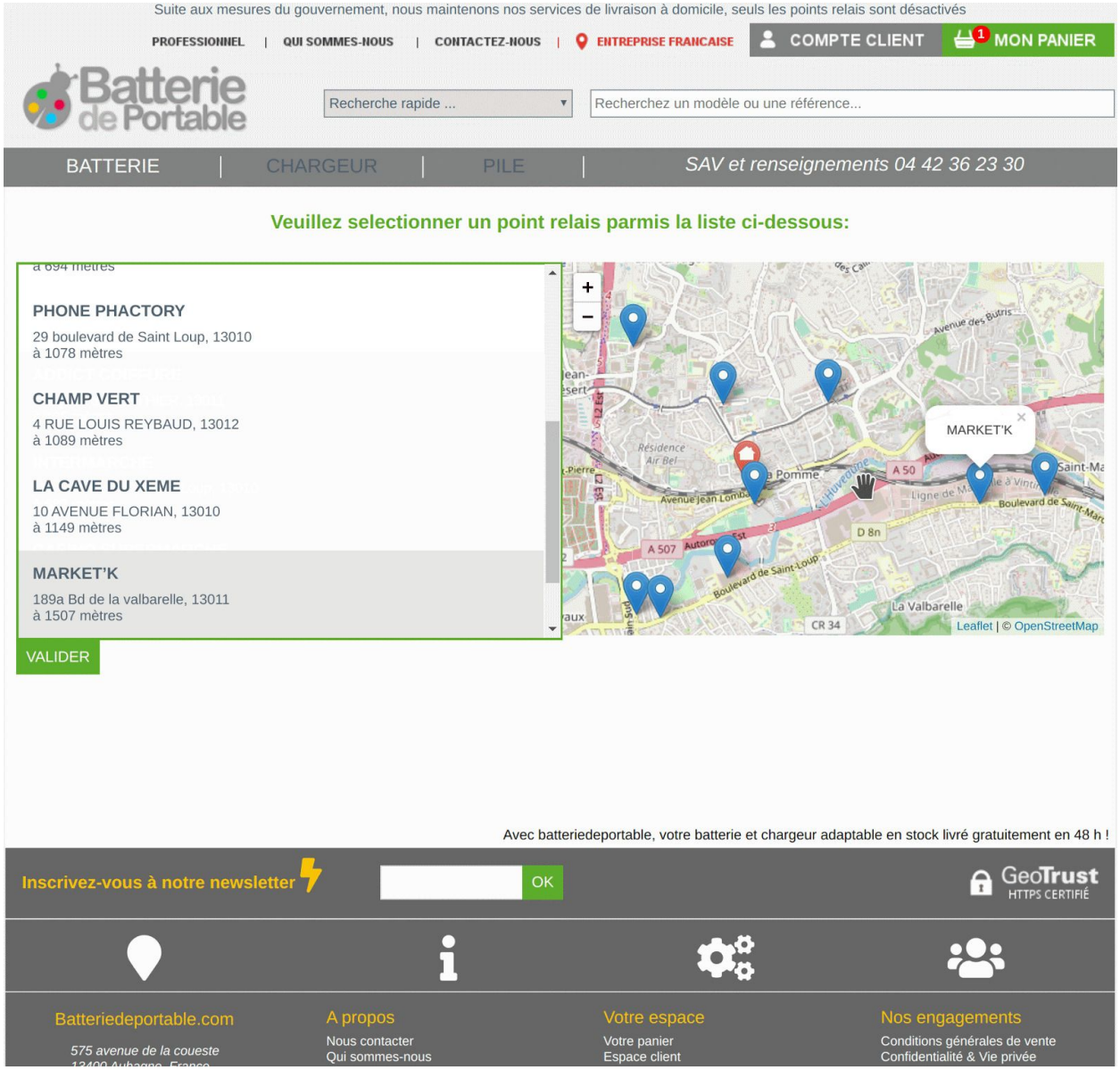


Figure 10: Affichage d'une carte avec les point relais et l'adresse du client.

Après quelques ajustements, la carte est interactive avec la liste (voir figure ci-dessous). Désormais lorsque l'on clique sur un point relais sur la carte, le point relais correspondant est sélectionné dans la liste.


Suite aux mesures du gouvernement, nous maintenons nos services de livraison à domicile, seuls les points relais sont désactivés
PROFESSIONNEL | QUI SOMMES-NOUS | CONTACTEZ-NOUS | ENTREPRISE FRANCAISE
COMPTE CLIENT
MON PANIER
Batterie de Portable
Recherche rapide ...
Recherchez un modèle ou une référence...
BATTERIE | CHARGEUR | PILE | SAV et renseignements 04 42 36 23 30
Veuillez selectionner un point relais parmis la liste ci-dessous:
PHONE PHACTORY
29 boulevard de Saint Loup, 13010
à 1078 mètres
CHAMP VERT
4 RUE LOUIS REYBAUD, 13012
à 1089 mètres
LA CAVE DU XEME
10 AVENUE FLORIAN, 13010
à 1149 mètres
MARKET'K
189a Bd de la valbarelle, 13011
à 1507 mètres
VALIDER
MARKET'K
Avenue des Butris
Résidence Air Bel
A 50
D 8n
A 507
Boulevard de Saint-Loup
La Valbarelle
CR 34
Leaflet | © OpenStreetMap
Avec batteriedeportable, votre batterie et chargeur adaptable en stock livré gratuitement en 48 h !
Inscrivez-vous à notre newsletter
OK
GeoTrust HTTPS CERTIFIÉ
Batteriedeportable.com
575 avenue de la couveste
A propos
Nous contacter
Qui sommes-nous
Votre espace
Votre panier
Espace client
Nos engagements
Conditions générales de vente
Confidentialité & Vie privée

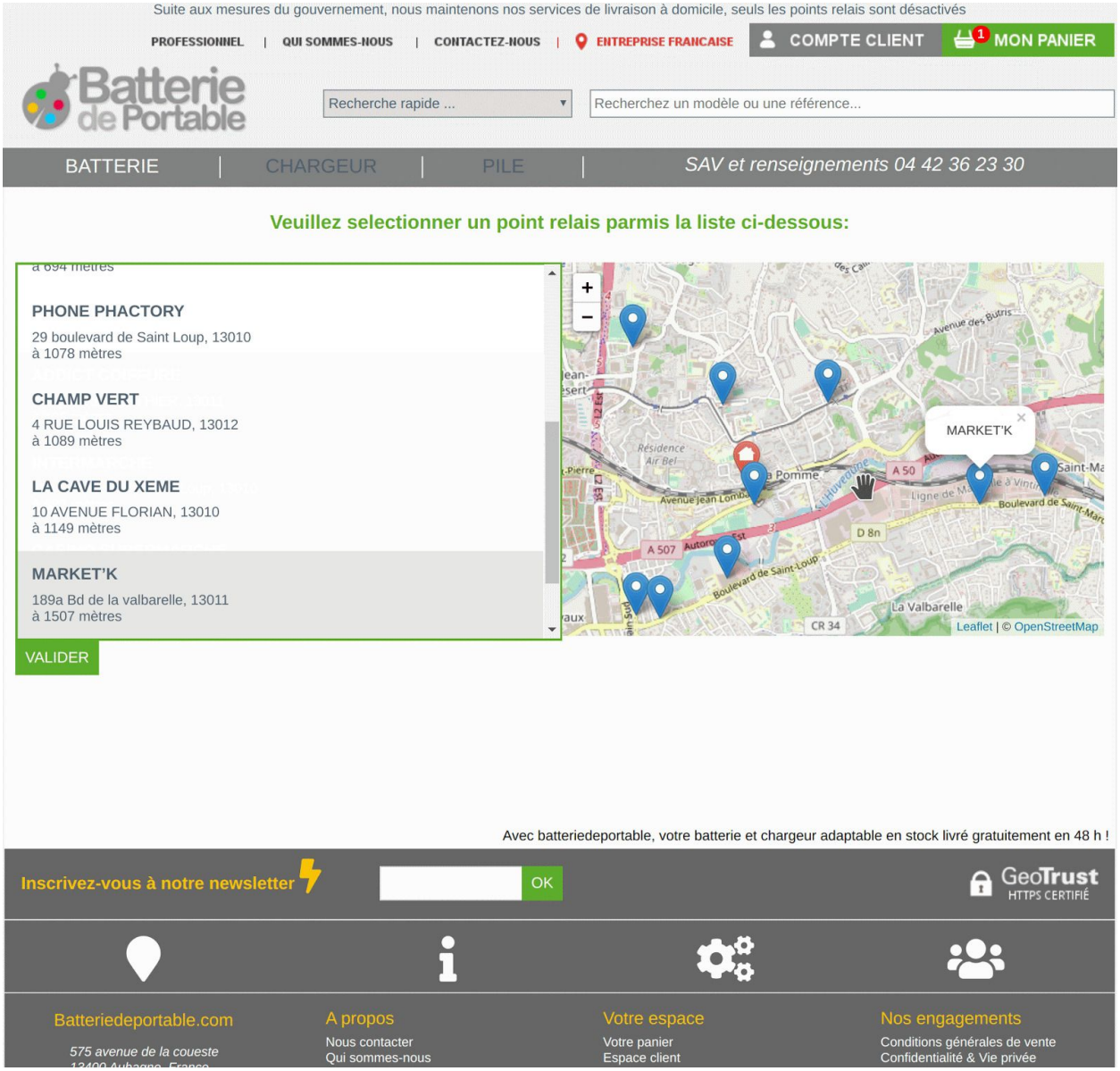


Figure 11: Carte de point relais intéractive.

Malheureusement, nous arrivons à la fin de mon stage et je n'ai pas eu le temps de finaliser le support des points relais **Chronopost**. Une fois le point relais sélectionné, le site est supposé leur transmettre la commande et leur demander son coût. Une fois que le client à payé, nous aurions pu, par exemple, effectuer un suivi de colis régulier et notifier le client (via email ou push notification) que sa commande est bien arrivée. En réalité, ce n'est pas un problème car M. Quere m'a proposé un **CDD** d'un mois ou deux pour que je puisse continuer à travailler sur son site web.

J'ai rencontré plus de difficultés sur cette mission, je n'avais jamais travaillé avec le protocole **SOAP**, de plus, de mon point de vue les **Chronopost Web Services** sont mals documentés et ne facilitent pas la tâche du développeur. Premièrement, il n'y a pas de documentation en ligne, il a fallu contacter l'entreprise afin d'obtenir une

documentation au format **PDF**. La documentation contient un peu plus d'une dizaine de services et environs 150 pages. Une documentation en ligne aurait été bien plus pratique pour trouver ce que l'on cherche. Hormis le support, la documentation reste assez claire et concise. Quant à l'affichage de la carte, **Leaflet.js** est très bien documenté, je n'ai donc pas eu de problème même si c'était ma première utilisation.

5 - Conclusion

Pour conclure, ce stage était pour moi une première dans le monde du travail où j'ai découvert le fonctionnement d'une **TPE** familiale. J'ai eu la chance de pouvoir demander de l'aide ou des conseils sans problème, afin de dissiper mes doutes et produire un meilleur travail. De plus, ma proposition d'utiliser **Docker** et la **conteneurisation** fût considérée et acceptée par mon maître de stage. Ce stage a donc était très instructif pour moi, j'ai désormais des connaissances solides en **PHP** alors que je n'avais que les bases. J'ai aussi grandement appris sur les fonctionnalités modernes tels que les **push notifications**, le **cache** et les **services workers,** et mes connaissances en **JavaScript** se sont grandement approfondies. L'utilisation d'**AMP** m'a permis de découvrir un nouveau **framework** optimisé pour les performances et les mobiles. Les contraintes qu'impose **AMP** pousse le développeur à écrire du meilleur code et mieux comprendre le fonctionnement des navigateurs si ce dernier s'intéresse aux contraintes. Enfin, j'ai découvert et utilisé le protocole **SOAP** pour interagir avec les services web Chronopost alors que j'avais toujours travailler avec des **API REST**. Enfin, j'ai appris à travailler en autodidacte, je me suis énormément documenté sur l'ensemble des **technologies**, **protocoles** et **API** utilisés.

Dans l'ensemble, je pense avoir atteint les objectifs du stage, à savoir que quelques missions ne sont pas présentes dans le rapport car elle n'apportent rien de plus comparé aux autres. J'ai amélioré des pages d'administration en intégrant le support de l'**AJAX**, j'ai ajouté une section administration pour éditer les pages articles du site ainsi que le support des notifications push. Le site est désormais installable et fonctionne hors-ligne. J'ai donc contribué à l'amélioration du site même si la barre de recherche sur les pages **AMP** ne peut pas afficher d'image et les point relais Chronopost ne sont pas encore finalisés. Pour ce qui concerne la barre de recherche **AMP**, il n'y a pas de solution actuellement, à voir si à l'avenir **AMP** devient plus permissif. Mais pour ce qui est de développer le support de Chronopost, je finaliserai cela lors de mon CDD.

Personnellement, ce stage a été très bénéfique, j'ai dû être assidu et régulier pour fournir des résultats dans les temps. J'ai aussi eu la chance de travailler avec un entrepreneur qui s'est construit seul et qui a pu m'apporter son expérience. Ce stage correspondait exactement à mon projet professionnel : être développeur et entrepreneur. Après avoir travaillé pendant deux mois et demi sur un site d'e-commerce, je projette également d'en ouvrir un, ce qui représentera peut être un début d'entreprenariat mais qui surtout sera instructif techniquement. Enfin, j'ai appris que j'étais accepté à l'**IMERIR** (**Institut Méditerranéen d'Études et de Recherches en Informatique et Robotique**) pour poursuivre mes études en alternance dans le domaine de l'informatique. Cette école me

semble être la parfaite continuité puisqu'elle forme aussi à l'entreprenariat avec en prime, un module à l'étranger.

6 - Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidée lors de la rédaction de ce rapport.

Je voudrais dans un premier temps remercier, mon maître de stage M. Frédéric Quere, pour, sa disponibilité, son partage d'expérience et surtout pour m'avoir accorder sa confiance tout au long de mon stage.

Je tiens à exprimer toute ma reconnaissance à ma tutrice académique, Mme Houssain. Je la remercie de m'avoir orienté, aidé et conseillé.

Merci à Paola pour sa patience et le temps consacré à la relecture de ce rapport.

Enfin, je tiens à remercier toutes les personnes qui m'ont permis de réaliser ce stage malgré les contraintes sanitaires.

7 - Glossaire

AJAX, Asynchronous Javascript And XML, permet une communication asynchrone entre le client et le serveur.

AMP, Accelerated Mobile Pages, est une technologie open source proposée et soutenue par Google pour créer des pages **HTML** avec des restrictions pour assurer des performances fiables.

Apache2, est un serveur HTTP créé et maintenu au sein de la fondation Apache. Jusqu'en avril 2019, ce fut le serveur HTTP le plus populaire du World Wide Web.

API, Application Programming Interface, les API permettent à votre produit ou service de communiquer avec d'autres produits et services sans connaître les détails de leur mise en œuvre (exemple: de multiple navigateur web permettent l'envois de notification à l'utilisateur mais le fonctionnement interne des navigateurs est différent).

CSS, Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents **HTML.** Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C).

DOM, Document Object Model, est une interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web, de page web.

DUT, Diplôme Universitaire de Technologie

ECMAScript, est un ensemble de normes concernant les langages de programmation de type script et standardisés par Ecma International dans le cadre de la spécification ECMA-262. Il s'agit donc d'un standard, dont les spécifications sont mises en œuvre dans différents langages de script, comme JavaScript ou ActionScript.

Framework, infrastructure logicielle, désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

EURL, Entreprise Unipersonnelle à Responsabilité Limité, il s'agit en fait juridiquement d'une société à responsabilité limitée (SARL) mais qui ne compte qu'un seul associé.

HTML, HyperText Markup Language, est le langage de balisage conçu pour représenter les pages web.

HTTP, HyperText Transfer Protocol, est un protocole de communication client-serveur

développé pour le World Wide Web. HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS.

JavaScript, est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs.

JSON, JavaScript Object Notation, est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML mais en plus léger.

VM, Virtual Machine, une machine virtuelle est une illusion d'un appareil informatique créée par un logiciel d'émulation ou instanciée sur un hyperviseur. Le logiciel d'émulation simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée.

PHP, Hypertext Preprocessor, est un langage de programmation libre (de droit), principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP.

PWA, Progressive Web Apps, applications/site web visant à remplacer les applications natives des ordinateurs, smartphones et tablettes (exemple: https://outlook.office.com).

REST, REpresentational State Transfer, est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web.

SAV, Service Après-Vente pour le service de livraison ou autre.

SEO, Search Engine Optimizations, définit l'ensemble des techniques qui visent à optimiser le positionnement d'une page, d'un site ou d'une application web dans la page des résultats de recherche d'un moteur de recherche (exemple: google).

SOAP, Simple Object Access Protocol, est un protocole d'échange d'information structurée dans l'implémentation de services web bâti sur XML.

SQL, Structured Query Language, est un langage informatique normalisé servant à exploiter des bases de données. Il permet l'insertion, la modification et la suppression de donnée dans la base.

TLS/SSL, Transport Layer Security (TLS) ou Secure Sockets Layer (SSL), sont des protocoles de sécurisation des échanges sur un réseau informatique, en général, mais en particulier, sur Internet.

TPE, Très Petite Entreprise, est une entreprise de faible taille, employant moins de 10 salariés

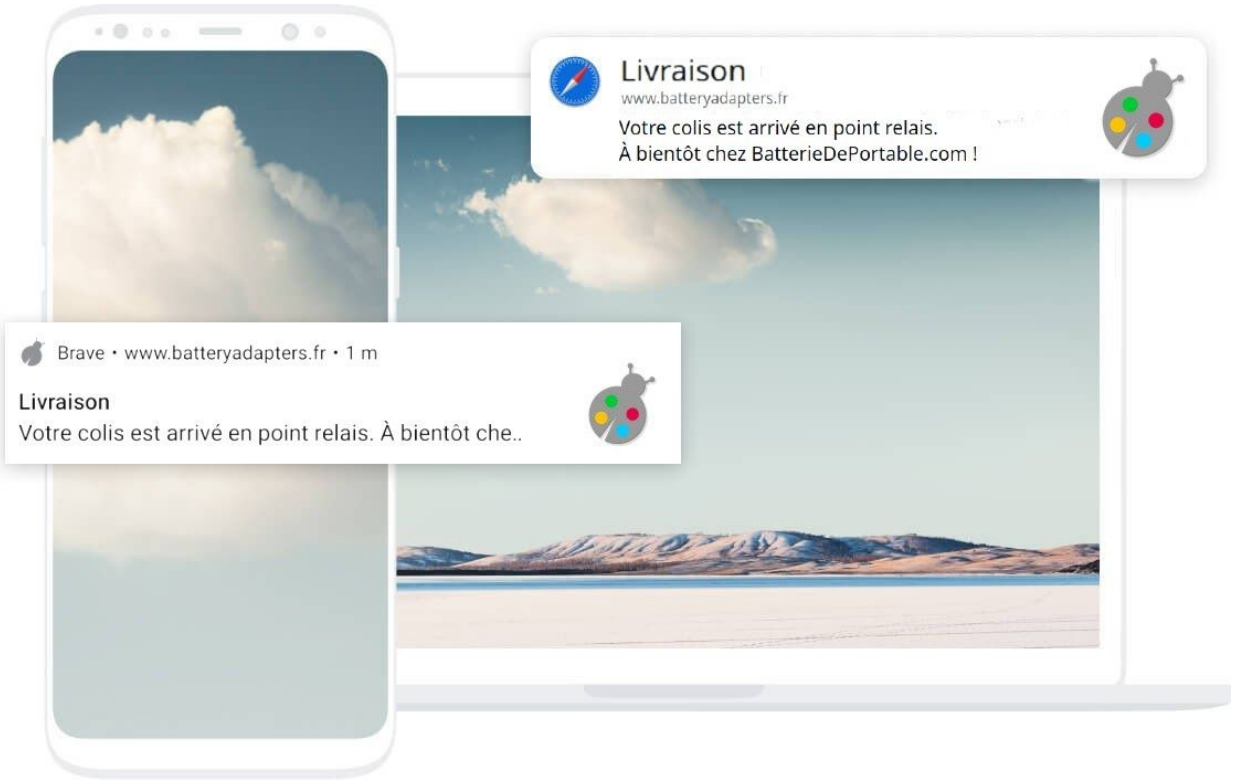
URL, Uniform Resource Locator, couramment appelée adresse web, est une chaîne de caractères uniformes qui permet d'identifier une ressource du World Wide Web par son emplacement et de préciser le protocole internet pour la récupérer (par exemple http ou https).

Worker, les workers sont des tâches (du code) exécutées en arrière-plan indépendamment des pages web par un navigateur.

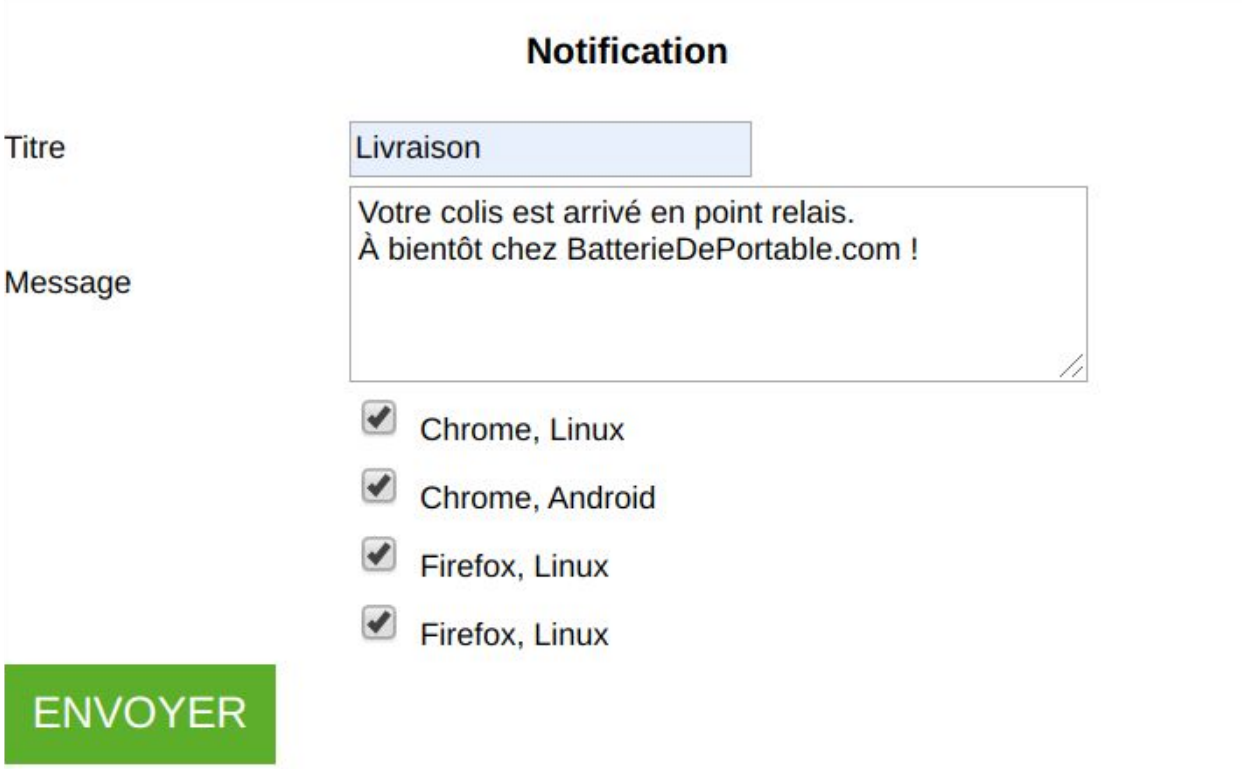
WSDL, Web Services Description Language, est une grammaire XML permettant de décrire un service web.

8 - Annexes

Affichage de notification Push (Android & MacOS)

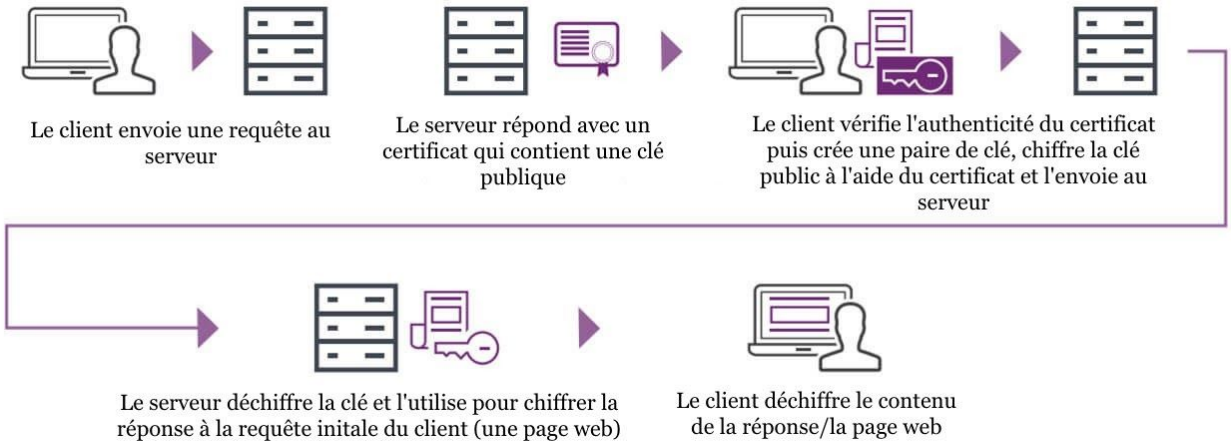


Formulaire notification push

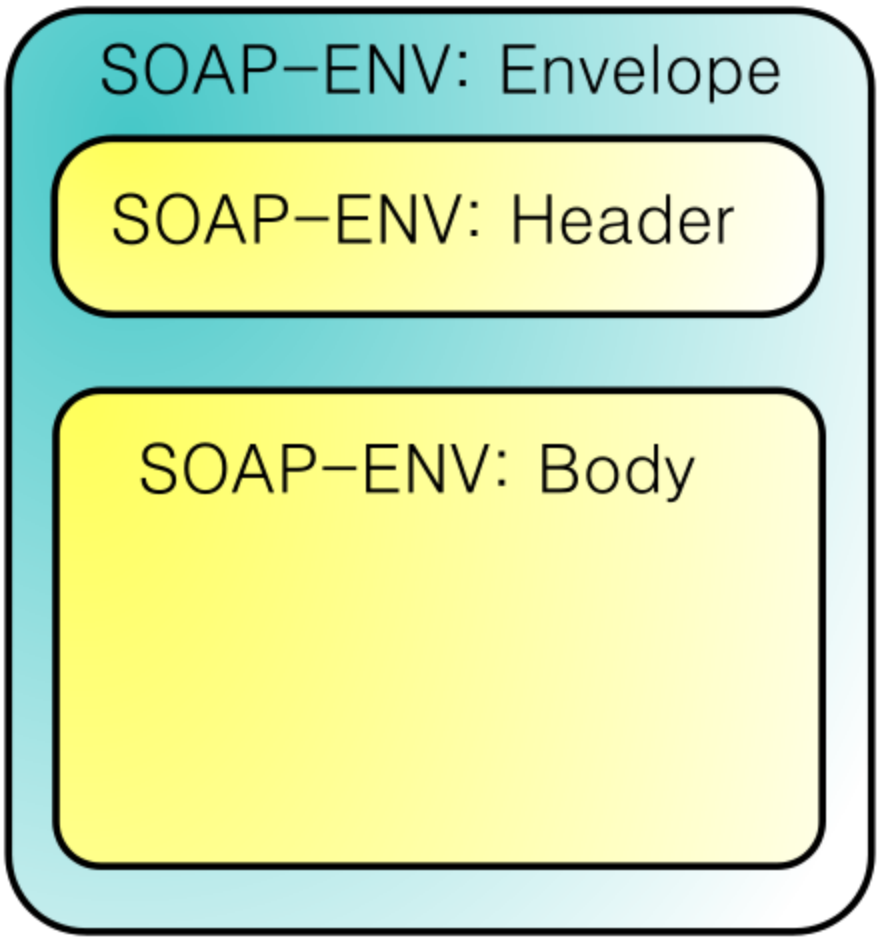


Barre de recherche AMP

Requête sécurisée en HTTPS



Structure de SOAP



Exemple d'une requête SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://cxf.rechercheBt.soap.chronopost.fr/">
  <SOAP-ENV:Body>
    <ns1:recherchePointChronopostInterParService>
      <accountNumber>[illegible]</accountNumber>
      <password>[illegible]</password>
      <address>[illegible]</address>
      <zipCode>13011</zipCode>
      <city>Marseille</city>
      <countryCode>FR</countryCode>
      <type>P</type>
      <service>L</service>
      <weight>1000</weight>
      <shippingDate>26/06/2020</shippingDate>
      <maxPointChronopost>25</maxPointChronopost>
      <maxDistanceSearch>25</maxDistanceSearch>
      <holidayTolerant>1</holidayTolerant>
      <language>FR</language>
    </ns1:recherchePointChronopostInterParService>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Carte point relais mobile