MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

Projet d'Approfondissement

WEB APP INVENTAIRE HE-ARC

Réalisé par

Armand Delessert

Sous la direction de
Prof. David Grunenwald

À la
Haute École Arc Ingénierie (HE-Arc),
HES-SO

HES-SO//Master, Lausanne, le 19/01/2018.

MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Accepté par la HES-SO//Master (Suisse, Lausanne) sur proposition de Prof. David Grunenwald, conseiller du projet d'approfondissement.

Lausanne, le 19/01/2018.

Prof. David Grunenwald
Conseiller

Prof. Didier Rizzotti
Responsable MRU

PROJET D'APPROFONDISSEMENT

Web App Inventaire HE-Arc

Table des matières

1 But du projet

La HE-Arc dispose d'un inventaire de son matériel disponible en prêt pour ses collaborateurs. Cet inventaire contient toute sorte de matériel électronique, tels que des ordinateurs portables, des téléphones portables, des appareils photo, des oscilloscopes, des multimètres ou encore des cartes électroniques de développement. Chaque collaborateur peut effectuer un emprunt de matériel dans cet inventaire selon ses besoins. Actuellement, le seul moyen d'effectuer des emprunts de matériel, c'est en passant par le site web de l'inventaire. Malheureusement, ce dernier n'est pas adapté aux téléphones mobiles et il ne permet pas d'utiliser les codes-barres ou les codes QR pour identifier un appareil rapidement.

Le but de ce projet est de développer une application web multiplateforme permettant d'enregistrer un emprunt ou un retour de matériel. Cette application web doit pouvoir s'adapter aux différentes tailles d'écran afin d'être utilisable autant sur ordinateur que sur smartphone (design responsive). L'application devra également proposer un scanner de code-barres et codes QR afin de rapidement lire le numéro d'inventaire d'un appareil. Enfin, le développement se focalisera sur une *Progressive Web Application* (PWA).

Pour plus d'information sur le but et la portée du projet, consulter le cahier des charges en annexe.

2 Exemples de cas d'utilisations

Pour une meilleure visualisation du projet, quelques cas d'utilisation de l'application sont présentés ci-dessous.

2.1 Recherche de matériel à emprunter

L'utilisateur a besoin d'un Windows Phone pour effectuer des tests :

- Il ouvre l'application et se connecte.
- Il effectue une recherche de « Windows Phone » dans l'application.
- Il sélectionne un des résultats.
    - Si celui-ci est déjà loué, il sélectionne un autre résultat jusqu'à trouver un appareil disponible.
- L'utilisateur entre ensuite une date de début et une date de fin d'emprunt et clique sur « Loan » pour effectuer l'emprunt.

2.2 Retour du matériel emprunté

L'utilisateur souhaite rendre le matériel emprunté via le numéro d'inventaire :

- Il ouvre l'application et se connecte.
- Il entrer le numéro d'inventaire de l'appareil et valide.
- Il clique enfin sur « Return » pour enregistrer le retour du produit.

L'utilisateur souhaite rendre le matériel emprunté via le code QR :

- Il ouvre l'application et se connecte.
- Il clique sur « Start scan WebCodeCamJS » et scan le code de l'appareil grâce à la caméra de son appareil.
- Lorsque le code est scanné, il clique sur « Return » pour enregistrer le retour du produit.

3 API

L'application communique avec le serveur de l'inventaire via une API (« *Application Programming Interface* », soit interface de programmation en français). Cette API doit proposer toutes les commandes nécessaires au client pour effectuer les tâches désirées comme emprunter ou rendre un matériel.

L'API devrait également respecter les principes de l'architecture REST (« *Representational State Transfer* »). L'architecture REST impose de respecter plusieurs principes (Wikipédia, s.d.). Voici ceux qui nous intéressent ici :

1. Le serveur ne mémorise pas d'informations sur la session, chaque requête effectuées par le client doit donc contenir toute l'information nécessaire pour être traitée par le serveur.
2. Chaque ressource doit être identifiable via une URL propre.
3. Les requêtes doivent utiliser une méthode HTTP appropriée (`POST` pour créer une donnée, `PUT` pour mettre à jour une donnée, `GET` pour lire une donnée, etc.) (REST API Tutorial, s.d.).

3.1 API actuelle

Actuellement, l'API en place n'a pas été destinée à un vrai usage en production et n'a pas été totalement testée. Elle comporte des bugs et ne respecte pas complètement les consignes imposées par l'architecture REST, vues plus haut.

L'adresse de l'API actuelle pour l'inventaire publique est la suivante :

```
https://inventory.ing.he-arc.ch
```

L'adresse de l'API actuelle pour l'inventaire de test et développement est la suivante :

```
https://inventory-dev.ing.he-arc.ch
```

Les différentes commandes de l'API actuelle sont décrites en détail dans l'annexe 9.2. Quelques exemples d'utilisation sont présentés ci-dessous.

3.1.1 Exemples d'utilisation de l'API actuelle

Les URL des commandes ci-dessous sont indiquées sans les noms de domaines présentés ci-dessus. Toutes les commandes s'effectuent avec la méthode HTTP `GET`.

Authentification

L'URL à utiliser pour l'authentification est la suivante :

```
/api/login/<username>/<password>
```

Cette commande retourne la réponse suivante en cas de succès :

```
{
  "success":true,
  "message":"OK",
  "result":"token"
}
```

Le paramètre `result` est l'identifiant de la session de l'utilisateur (le « `token` »).

Récupérer un produit par son identifiant

Pour récupérer un produit et afficher ses détails, il faut utiliser la commande `searchById` de la classe `Product`. Voici la commande :

URL : `/api/rest/<token>/Product/searchById?productId=<productId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.

Résultat : Produit correspondant à l'ID indiqué.

Exemple :

Pour récupérer les détails du produit avec l'ID 200 dans l'inventaire de développement, il faut appeler la commande suivante avec la méthode HTTP `GET` :

```
https://inventory-dev.ing.he-arc.ch/api/rest/token/Product/searchById?productId=200
```

Enregistrer un emprunt de produit

Pour enregistrer un emprunt de produit, il faut utiliser la commande `loan` de la classe `Product`. Voici la commande :

URL : `/api/rest/<token>/Product/loan?productId=<productId>&beginDate=<beginDate>&endDate=<endDate>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.
- `<beginDate>` : Date de début de l'emprunt.
- `<endDate>` : Date de fin de l'emprunt.

Résultat : Enregistre un emprunt pour le produit correspondant à l'ID indiqué.

Exemple :

Pour effectuer un emprunt du produit avec l'ID 200 pour le mois de janvier 2018 dans l'inventaire de développement, il faut appeler la commande suivante avec la méthode HTTP `GET` :

```
https://inventory-dev.ing.he-arc.ch/api/rest/token/Product/loan?productId=200&beginDate=2018-01-01&endDate=2018-01-31
```

Récupérer la liste des produits empruntés par un utilisateur

Pour récupérer la liste des produits empruntés par un utilisateur, il faut commencer par récupérer l'identifiant de l'utilisateur. Pour cela, il faut utiliser la commande `getList` de la classe `User`. Voici la commande :

URL : `/api/rest/<token>/User/getList`

Paramètres :
- `<token>` : Identifiant de session.

Résultat : Liste des utilisateurs enregistrés.

Il faut ensuite récupérer l'identifiant de l'utilisateur dans la liste des utilisateurs retournée par la commande précédente. Enfin, il faut effectuer une requête avec la commande `asearch` de la classe `Product`. Voici la commande :

URL : `/api/rest/<token>/Product/asearch?loanedBy=<userId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<userId>` : Identifiant de l'utilisateur.

Résultat : Liste des produits empruntés par l'utilisateur correspondant à l'ID indiqué.

Exemple :

Pour récupérer la liste des produits empruntés par l'utilisateur `devweb.user` (identifiant de l'utilisateur : 124), il faut appeler la commande suivante avec la méthode HTTP `GET` :

```
https://inventory-dev.ing.he-arc.ch/api/rest/token/Product/asearch?loanedBy=124
```

3.1.2 Conclusion sur l'API actuelle

Le constat est que l'API actuelle se calque sur le fonctionnement interne de la base de données en appelant des « fonctions » et en y passant des paramètres. L'architecture d'une API REST doit normalement se baser sur la structure des données plutôt que sur les actions effectuées (Fielding, 2012), un principe qui n'est pas respecté ici. De plus, toutes les commandes de l'API utilisent la méthode HTTP `GET`, ce qui est contraire au principe REST selon lequel les requêtes doivent utiliser la méthode HTTP appropriée (`POST`, `PUT`, `GET`, etc.).

3.2 Nouvelle API

Pour corriger les manques de l'ancienne API, une nouvelle API a été spécifiée, plus respectueuse des principes REST.

Cette nouvelle API permet les actions suivantes :

- Authentification de l'utilisateur
- Affichage des détails d'un produit
- Enregistrement de l'emprunt d'un produit
- Enregistrement du retour d'un produit
- Récupération de tous les emprunts de l'utilisateur

Par exemple, l'accès aux détails d'un produit se fera avec la méthode HTTP `GET` (lecture) sur l'URL suivante :

```
/api/rest/<token>/products/<id>/details/
```

Concernant l'emprunt d'un produit, ça se fera avec la méthode HTTP `PUT` (mise à jour) sur l'URL suivante :

```
/api/rest/<token>/products/<id>/loan/<begin_date>/<end_date>/
```

Pour la spécification complète de la nouvelle API, voir l'annexe « Spécification de la nouvelle API REST ».

Avec ces changements, la nouvelle API respecte mieux les standards d'une API REST. Cependant, elle n'est pas parfaite non-plus. En effet, lorsque l'utilisateur se connecte au travers de cette API, le serveur doit lui renvoyer un identifiant de session (un « *token* »). Ce *token* doit ensuite être envoyé dans chaque requête futures à l'API afin que le serveur puisse identifier l'utilisateur. Ce mode de fonctionnement viole le premier principe REST énoncé au chapitre 3 ci-dessus selon lequel le serveur ne mémorise pas d'informations sur la session du client et chaque requête effectuées par le client doit donc contenir toute l'information nécessaire pour être traitée par le serveur. Cependant, l'aspect « *full stateless* » sert surtout lorsqu'un serveur doit subvenir à de grandes quantités de connexions et gérer l'état pour chacune de ces connexions est compliqué. Ainsi, l'aspect *stateless* permet une grande montée en charge. Nous ne sommes pas confrontés à cette problématique dans notre cas, l'application n'étant pas amenée à toucher un large public.

3.2.1 Outils de conception d'API REST

mockable.io

La mise en place de l'API a été faite au travers du site de *mock* d'API mockable.io. Ce site permet de coder en dur les URL de test et la réponse associée. La méthode HTTP peut également être spécifiée.

L'avantage de cette solution est de permettre de tester notre API très facilement. Le désavantage, c'est qu'il n'est pas possible d'effectuer des modifications sur les données retournées par l'API. Par exemple, il est possible de définir une URL en dur pour l'emprunt d'un produit avec un identifiant défini et pour une durée définie mais il ne sera pas possible de modifier son statut « `is_loan` » pour le passer à « `true` » et ainsi enregistrer que le produit a été emprunté.

Swagger

Swagger est un autre outil de conception d'API. Il permet de décrire une API avec un langage prévu à cet effet. Il était prévu d'utiliser cet outil pour développer la nouvelle API REST mais après quelques

essais et des difficultés à tester l'API décrite, le choix s'est tourné vers mockable.io, plus basique mais suffisant.

Mocky

Mocky est un autre site de *mock* d'API, à l'image de mockable.io, mais contrairement à ce dernier, Mocky ne propose pas de personnaliser l'URL des requêtes.

4 Choix des outils de conception

4.1 Langage de programmation et *framework*

Le choix du développement d'une application web était imposé afin de permettre à l'application d'être multi-plateformes. Mais le choix des outils de conception était laissé libre.

Le recours à un *framework* JavaScript tel que Angular, React, Bootstrap, Ionic ou autre pour le développement de l'application était envisagé au début du projet. Mais n'ayant pas vraiment d'expérience dans le développement d'un site web et après avoir suivi quelques tutoriels sur les bases du développement web, le développement de l'application s'est naturellement tourné vers le couple standard des technologies web, le HTML et le JavaScript, sans *framework* particulier. En effet, le développement du projet a commencé en reprenant le code présenté dans les divers tutoriels consultés.

Le choix d'un « *boilerplate* », un squelette de site web, a également été envisagé. Le choix s'était tourné vers Initializr, un *template* basé sur HTML5 Boilerplate et qui permet de ne conserver que les éléments qui nous sont utiles. Cependant, après quelques tests avec ce *template*, des problèmes ont été rencontrés et pour les résoudre il a fallu isoler le code de l'application dans un projet vide, sans *template*. Et c'est ainsi que l'application a finalement été développée sans utiliser de *template* de départ.

4.2 Choix de la bibliothèque de scan de codes-barres et de codes QR

Le site web de l'inventaire permet de générer des codes-barres et des codes QR à partir des numéros d'inventaire des produits afin de permettre de rapidement obtenir le numéro d'inventaire du produit via un scan (lien du générateur).

Les codes sont de 2 types comme mentionné plus haut :

- Les codes-barres (type **Code 128**[1], code à 1 dimension), qui sont surtout dédiés aux lecteurs par laser (appareils de scan spécifiques, lecteurs de codes à 1 dimension).
- Les codes QR (code à 2 dimensions), qui sont plus adaptés aux lecteurs par caméra (application sur smartphone, etc., lecteurs de codes à 2 dimensions).

4.2.1 Liste des bibliothèques JavaScript de scan de codes-barres et codes QR

La liste ci-dessous énumère les différentes bibliothèques JavaScript de scan de code 1 dimension et 2 dimensions trouvées ainsi que leurs principales caractéristiques. La bibliothèque JavaScript choisie pour notre application devra être capable de scanner ces 2 types de code.

QR Code scanner (sans nom précis), LazarSoft

- Scan de codes QR seulement.
- Sources sur GitHub.

jsQRScan

- Scan de codes QR seulement.
- Basé sur QR Code Scanner de LazarSoft.

Instascan, schmich

- Scan de codes QR seulement.
- Sources sur GitHub.

QuaggaJS, serratus

- Scan de codes-barres seulement.
- Sources sur GitHub.

BarcodeReader, EddieLa

- Scan de codes-barres seulement.
- Sources sur GitHub.

JavaScript Barcode Scanner (sans nom précis)

- Scan de codes-barres seulement.

WebCodeCamJS, andrastoth

- Scan de codes-barres et de codes QR.
- Basé sur BarcodeReader de EddieLa pour la lecture de codes-barres et sur QR Code scanner de LazarSoft pour le scan de codes QR.
- Sources sur GitHub.

Au final, **WebCodeCamJS** est la seule bibliothèque JavaScript trouvée permettant le scan des 2 types de code qui nous intéressent (codes-barres et codes QR). C'est donc cette bibliothèque qui a été choisie pour mettre en place la fonction de scan de l'application.

[1] Selon la documentation du site permettant la génération des codes.

4.3 Aspect PWA

Les « *Progressive Web App* » (PWA, applications web progressives en français) sont des sites web tirant parti des nouvelles technologies web supportées par les navigateurs web récents.

Les *Progressive Web App* possèdent plusieurs avantages par rapport aux sites web classiques. Elles sont :

- **Installable :** Il est par exemple possible d'« épingler » un raccourci de l'application sur l'écran d'accueil de l'appareil.
- **Indépendante du réseau :** Contrairement à un site web classique, il est possible d'y accéder sans avoir de connexion Internet active ou avec une mauvaise connexion Internet.
- **Progressive :** L'application fonctionne sur tous les navigateurs, quel que soit leur niveau de support des technologies web.
- **Adaptative :** L'application s'adapte à tous les formats d'écrans, que ce soit sur PC ou téléphone.
- **Sécurisée :** En utilisant exclusivement HTTPS par exemple.

Pour acquérir ces nouvelles connaissances, le suivi du tutoriel de Google sur le sujet (Your First Progressive Web App) a été nécessaire.

Le travail effectué sur l'application pour lui donner les caractéristiques des PWA a été d'une part de lui donner une interface adaptative selon la taille de l'écran. Cette tâche a été effectuée assez simplement en utilisant un fichier CSS respectant les consignes du « *responsive web design* ». Le fichier CSS utilisé est celui du *framework* **Foundation**.

D'autre part, la mise en cache des fichiers de ressources statiques (HTML, CSS, JavaScript, etc.) a été réalisée grâce à un « *Service Worker* ». En théorie, le *Service Worker* permettrait également de mettre en cache les requêtes « `GET` » faites à l'API et ainsi permettre à l'utilisateur de consulter les produits déjà consultés, même en cas d'absence de connexion Internet. Cependant, la gestion du cache est une tâche délicate et cette fonctionnalité n'a pas été développée par manque de temps.

La mise en cache des fichiers de ressources statiques doit également se faire dans la configuration du serveur. En effet, pour indiquer au navigateur qu'un fichier ne sera pas modifié prochainement et qu'il peut donc être conservé en cache, le serveur web doit ajouter un entête (« *header* ») dans la réponse HTTP à la requête du navigateur sur ce fichier. L'entête ajouté peut être soit l'entête « `Cache-Control` » pour définir une durée maximale de mise en cache, soit l'entête « `Expires` » pour définir une date d'expiration du fichier conservé en cache. Pour une ressource qui n'est pas amenée à être modifiée, il est recommandé de configurer une durée de mise en cache d'1 an. Il n'est cependant pas recommandé de configurer une durée de mise en cache de plus d'1 an. Malheureusement, GitHub.io, le système d'hébergement de pages web utilisé pour tester l'application, ne permet pas de configurer les entêtes HTTP.

5 Tests effectués

5.1 W3C Markup Validation Service

Ce test a permis de mettre en évidence plusieurs erreurs, dont une balise HTML fermée incorrectement. Toutes les erreurs signalées ont pu être corrigées.

Pour consulter les résultats du test, voir l'annexe 9.5.1, 9.5**W3C HTML Checker.pdf**. Le test peut être exécuté sur le site web validator.w3.org.

5.2 Google PageSpeed Insights

Ce test a permis d'apporter plusieurs améliorations et optimisations à l'application, notamment l'ajout de l'attribut « `defer` » aux lignes d'importation des fichiers JavaScript dans l'entête du fichier HTML afin de différer leur chargement et ainsi améliorer la rapidité de l'affichage de la page.

Pour consulter les résultats du test, voir l'annexe 9.5.2, **PageSpeed Insights Desktop.pdf** et **PageSpeed Insights Mobile.pdf**. Le test peut être exécuté sur le site web developers.google.com/speed/pagespeed/insights/.

5.3 Test de la rapidité du site sur mobile

Ce test permet de constater que le site se charge rapidement sur une connexion mobile. Le test met en avant 3 éléments à améliorer :

- Compresser les fichiers de ressources (HTML, JavaScript, CSS, etc.).

C'est effectivement un point intéressant à optimiser pour un site en production. Dans notre cas, le site est encore en développement et il est nécessaire de conserver une mise en forme du code facile à maintenir.

- Mettre en cache dans le navigateur les fichiers de ressources (HTML, JavaScript, CSS, etc.).

La mise en cache dans le navigateur des fichiers de ressources est un point très important pour qu'un site se charge rapidement lorsque l'utilisateur visite le site plusieurs fois. Il est recommandé, pour une ressource qui n'est pas amenée à être modifiée dans un futur proche, de configurer la durée d'expiration du cache à 1 an.

Cette durée de mise en cache doit être configurée sur le serveur. Il s'agit d'un entête à configurer dans les messages HTTP lorsque le serveur envoi un fichier de ressource au client. Malheureusement, dans notre cas il ne nous est pas possible de configurer les entêtes HTTP envoyée par GitHub.io, le site sur lequel est hébergé l'application web actuellement.

- Supprimer les codes JavaScript qui bloquent l'affichage du contenu au-dessus de la ligne de flottaison.

Concernant ce point, il s'agit d'un comportement voulu. En effet, la page est constituée de plusieurs parties, le log in, l'affichage d'un produit, la recherche d'un produit, etc. Lorsque l'utilisateur effectue une action, l'affichage change en fonction de l'action. Par exemple, lorsque l'utilisateur se log, la partie de login disparaît et la partie permettant d'afficher ou de recherche un produit apparaît.

Pour consulter les résultats du test, voir l'annexe 9.5.3, **Test My Site.pdf**.

5.4 Google Lighthouse (Audits Google Chrome DevTools)

Ce test a permis d'apporter plusieurs corrections et optimisations concernant l'aspect PWA, notamment l'ajout du fichier `manifest.json`. D'après ce test, toutes les pratiques PWA sont respectées.

Pour consulter les résultats du test, voir l'annexe 9.5.4, **Chrome DevTools Audits.png**.

6 Travail restant et développement futur

6.1 Signaler les produits déjà empruntés

Lors d'une recherche de produits par mots-clés, tous les produits correspondants à la recherche sont présentés dans la liste des résultats. Cependant, certains des produits présentés peuvent être déjà empruntés par un autre utilisateur. Il serait intéressant d'ajouter une signalisation sur les produits déjà empruntés et donc non disponibles en affichant leur nom avec une couleur différente par exemple.

6.2 Afficher la liste des produits empruntés après la connexion de l'utilisateur

Lorsque l'utilisateur se connecte, il serait intéressant de tout de suite afficher la liste des produits empruntés, avec un bouton « Return » pour pouvoir facilement retourner un produit emprunté sans devoir le rechercher à nouveau.

6.3 Sécurisation du mot de passe de login

Lors de la connexion de l'utilisateur, le mot de passe est passé en clair au serveur par l'URL. Bien que la communication entre le client et le serveur utilise le protocole sécurisé HTTPS et qu'il est nécessaire de passer par le VPN de l'HE-Arc pour accéder à l'inventaire, ce n'est jamais la meilleure façon de faire de communiquer un mot de passe en clair. Cependant, transmettre un simple hash du mot de passe à la place du mot de passe en clair ne résout pas le problème. La solution conventionnelle est de passer par du HTTPS, ce qui est déjà le cas. Un moyen en plus pour améliorer la sécurité est de passer par le salage du mot de passe :

- Le client effectue une requête de connexion au serveur.
- Le serveur renvoi un sel (une série de bytes aléatoires).
- Le client génère également un sel.
- Le client concatène le mot de passe et les 2 sels et en calcul un hash.
- Le client renvoi le hash avec les 2 sels au serveur.
- Le serveur, qui connais le mot de passe original, est capable de recalculer le hash de la même manière et ainsi vérifier que le mot de passe transmis est le bon.

Pour encore mieux sécuriser le système, le mot de passe doit être hashé avant d'être combiné aux sels, ce qui donne la configuration suivante :

```
Identifiant envoyé au serveur = hash(hash(motDePasse) + selServeur + selClient)
```

Sources :

- Password encryption at client side
- Is it worth hashing passwords on the client side
- How to Make a Secure Login Form with SSL

6.4 Enregistrer le login de l'utilisateur

Actuellement, l'application demande à l'utilisateur de se logger à chaque ouverture. Il serait intéressant que l'application demande à l'utilisateur s'il souhaite que sa connexion soit maintenue pour les ouvertures futures de l'application.

6.5 Ajuster la gestion du cache par le Service Worker

Actuellement, si l'un des fichiers de l'application (fichier HTML, JavaScript, etc.) est mis à jour, le Service Worker ne détectera pas automatiquement ce changement et ne mettra pas à jour le fichier en cache. Il sera nécessaire de recharger la page manuellement (touche F5) ou forcer le rechargement complet de la page (touches Ctrl + F5) selon le navigateur ou, dans certains cas, désenregistrer le Service Worker (sous Google Chrome, ouvrir les DevTools avec F12, section « Application », cliquer sur « Unregister » en face du Service Worker puis recharger la page avec F5).

6.6 Configurer la mise en cache des fichiers du côté du serveur

Le service utilisé pour héberger l'application actuellement est GitHub.io. Ce service est relativement simple à mettre en place mais il ne propose pas de paramétrer son fonctionnement. Ainsi, il n'est pas possible d'ajouter les entêtes HTTP « `Cache-Control` » ou « `Expires` » afin de configurer la mise en cache des fichiers. Sans ces entêtes, la durée de conservation des fichiers en cache par défaut est de 10 minutes, ce qui peut avoir un impact négatif sur les performances. Il serait donc intéressant de changer de système d'hébergement afin de pouvoir configurer cette mise en cache. Pour des fichiers qui ne sont pas amenés à être modifiés régulièrement, la mise en cache peut être configurée pour 1 an.

6.7 Ne pas afficher la section de scan sur les machines dépourvues de caméra

Lorsque le site est utilisé sur une machine sans caméra, la section de scan affiche un cadre blanc à la place de la vidéo de la caméra. Il serait bien de ne simplement pas afficher cette section si la machine ne possède pas de caméra.

7 Bugs connus

7.1 Bugs de l'API actuelle

7.1.1 Emprunts effectués sur une plage de date passées ou futures

Avec l'API actuelle, lorsque l'utilisateur effectue un emprunt sur une plage de dates passées ou futures, le produit emprunté ne peut plus être retourné. Le fait d'enregistrer un retour de ce produit n'annule pas son statut « emprunté ». Il s'agit d'un bug du côté du serveur de gestion de l'inventaire.

7.1.2 Erreur lors de certaines recherches

Certaines recherches de produits par mots-clés retournaient un message d'erreur. Par exemple, la recherche du mot clé « lum » retournait une erreur alors que la recherche du mot clé « del » fonctionnait et retournait bien la liste de produits contenant ce mot-clé.

Il s'agissait d'un bug du côté du serveur. Il a été corrigé par M. Christophe Bolinhas.

7.2 Bug de l'application

7.2.1 Scan en boucle

Lorsque l'utilisateur scan un code-barres ou un code QR, le scan lit le code tant que la caméra point sur lui, ce qui a pour effet de biper continuellement jusqu'à ce que le code ne soit plus visible par la caméra ou que l'utilisateur arrête le scanner via le bouton.

Pour résoudre ce problème, il faudrait comparer le code qui vient d'être lu avec le dernier code lu et s'il s'agit du même code, il faut arrêter le traitement ici.

8 Conclusion

Je me suis lancé dans ce projet dans le but d'avoir une introduction au monde du développement web. Je n'avais jusque-là que peu d'expérience dans ce domaine et je souhaitais en apprendre davantage.

Le fait que ce projet soit pour moi une première introduction au développement web a rendu la phase de recherche des outils et technologies de développement particulièrement longue et fastidieuse. En effet, ne connaissant que très peu les outils à disposition, j'avais beaucoup de mal à me représenter quel avantage offre l'un par rapport à l'autre, notamment au concernant les *framework* JavaScript.

Le débug et l'utilisation des outils de développement de Google Chrome ou Firefox n'a pas été quelque chose d'évident tout de suite pour moi. J'ai eu quelques difficultés lors des premières phases de début, c'est entre-autre pour cette raison que j'ai abandonné l'utilisation d'un « *boilerplate* » et que je suis parti sur un code *from scratch*, plus simple à débuger.

Ce projet m'aura permis de grandement développer mes connaissances dans ce domaine qui est de plus en plus important à maitriser. L'aspect PWA rendait le travail d'autant plus intéressant du fait qu'il s'agisse d'une technologie d'avenir.

9 Annexes

9.1 Cahier des charges

Le cahier des charges est disponible sur le dépôt GitHub à l'adresse suivante :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-
App/blob/master/doc/Cahier%20des%20charges.md
```

9.2 Spécification de l'API actuelle

La spécification ci-dessous ne couvre pas toutes les commandes de l'API actuelle mais les plus importantes. Pour consulter toutes les commandes offertes par l'API actuelle, référez-vous à sa documentation à l'une des adresses suivantes :

API de l'inventaire publique :

```
https://inventory.ing.he-arc.ch/api/commands
```

API de l'inventaire de test et développement :

```
https://inventory-dev.ing.he-arc.ch/api/commands
```

9.2.1 URL des noms de domaines

L'adresse de l'API actuelle pour l'inventaire publique est la suivante :

```
https://inventory-dev.ing.he-arc.ch
```

L'adresse de l'API actuelle pour l'inventaire de test et développement est la suivante :

```
https://inventory-dev.ing.he-arc.ch
```

Les URL des commandes ci-dessous sont indiquées sans les noms de domaines ci-dessus. Toutes les commandes s'effectuent avec la méthode HTTP `GET`.

9.2.2 Authentification

Structure de l'URL pour l'authentification :

```
/api/login/<username>/<password>
```

Paramètres de la commande :

- `<username>` : Nom d'utilisateur.
- `<password>` : Mot de passe (en clair).

Réponse en cas de succès :

```
{
  "success":true,
  "message":"OK",
  "result":"token"
}
```

Le paramètre `result` est l'identifiant de la session de l'utilisateur (le « `token` »).

Noms d'utilisateurs des comptes de développement :

- Utilisateur simple : **`devweb.user`**
- Manager : **`devweb.manager`**
- Administrateur : **`devweb.admin`**

Mot de passe (pour chacun des 3 comptes ci-dessus) : **123456**

9.2.3 Liste des commandes

Classe `Category`

Commande `getList` :

La commande `getList` permet d'obtenir la liste des catégories (Informatique, Électronique, etc.).

URL : `/api/rest/<token>/Category/getList`

Paramètres :
- `<token>` : Identifiant de session.

Résultat : Liste des catégories.

Classe `Domain`

Commande `getList` :

La commande `getList` permet d'obtenir la liste des domaines.

URL : `/api/rest/<token>/Domain/getList`

Paramètres :
- `<token>` : Identifiant de session.

Résultat : Liste des domaines.

Classe `Location`

Commande `getList` :

La commande `getList` permet d'obtenir la liste des localisations (bureaux, laboratoires, etc.).

URL : `/api/rest/<token>/Location/getList`

Paramètres :
- `<token>` : Identifiant de session.

Résultat : Liste des localisations.

Classe `Product`

Commande `searchById` :

URL : `/api/rest/<token>/Product/searchById?productId=<productId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.

Résultat : Produit correspondant à l'ID indiqué.

Commande `search` :

URL : `/api/rest/<token>/Product/search?term=<term>`

Paramètres :
- `<token>` : Identifiant de session.
- `<term>` : Terme de la recherche.

Résultat : Liste des produits correspondants au terme recherché.

Commande `info` :

URL : `/api/rest/<token>/Product/info?productId=<productId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.

Résultat : Infos du produit correspondant à l'ID indiqué.

Commande `canBeLoaned` :

URL : `/api/rest/<token>/Product/canBeLoaned?productId=<productId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.

Résultat : Indique si le produit correspondant à l'ID indiqué peut être emprunté.

Commande `loan` :

URL : `/api/rest/<token>/Product/loan?productId=<productId>&beginDate=<beginDate>&endDate=<endDate>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.
- `<beginDate>` : Date de début de l'emprunt.
- `<endDate>` : Date de fin de l'emprunt.

Résultat : Enregistre un emprunt pour le produit correspondant à l'ID indiqué.

Commande `returnProduct` :

URL : `/api/rest/<token>/Product/returnProduct?productId=<productId>`

Paramètres :
- `<token>` : Identifiant de session.
- `<productId>` : Identifiant du produit.

Résultat : Enregistre le retour du produit correspondant à l'ID indiqué.

9.2.3.1 Classe `User`

Commande `getList` :

URL : `/api/rest/<token>/User/getList`

Paramètres :
- `<token>` : Identifiant de session.

Résultat : Liste des utilisateurs enregistrés.

9.3 Spécification de la nouvelle API REST

La spécification de la nouvelle API REST est disponible sur le dépôt GitHub à l'adresse suivante :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Sp%C3%A9cification%20API%20REST.md
```

9.4 Sources de l'application web

Les sources de l'application web sont disponibles sur le dépôt GitHub à l'adresse suivante :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App
```

9.5 Rapports des tests effectués

9.5.1 W3C Markup Validation Service

Le fichier **W3C HTML Checker.pdf** est disponible sur le dépôt GitHub à cette adresse :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Tests/W3C%20HTML%20Checker.pdf
```

9.5.2 Google PageSpeed Insights

Le fichier **PageSpeed Insights Desktop.pdf** est disponible sur le dépôt GitHub à cette adresse :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Tests/PageSpeed%20Insights%20Desktop.pdf
```

Le fichier **PageSpeed Insights Mobile.pdf** est disponible sur le dépôt GitHub à cette adresse :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Tests/PageSpeed%20Insights%20Mobile.pdf
```

9.5.3 Test de la rapidité du site sur mobile

Le fichier **Test My Site.pdf** est disponible sur le dépôt GitHub à cette adresse :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Tests/.pdf
```

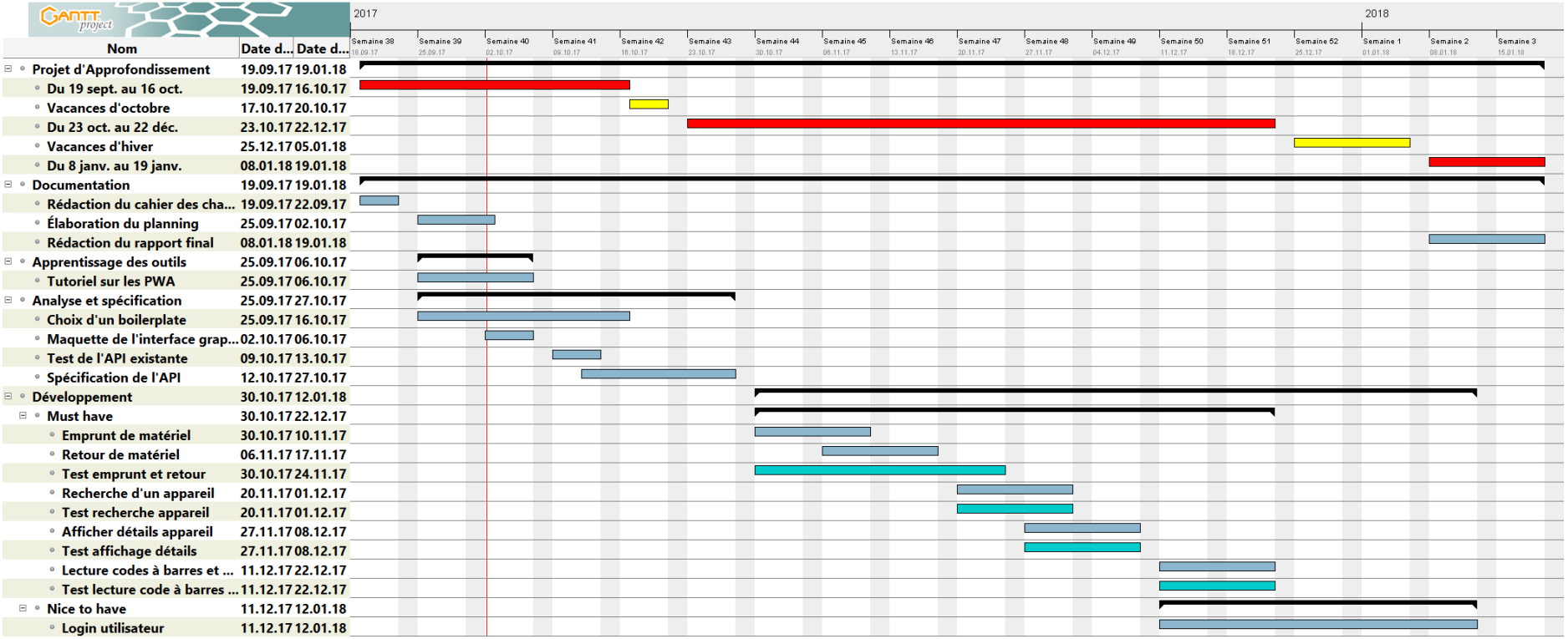
9.5.4 Google Lighthouse (Audits Google Chrome DevTools)

Le fichier **Chrome DevTools Audits.png** est disponible sur le dépôt GitHub à cette adresse :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-App/blob/master/doc/Tests/Chrome%20DevTools%20Audits.png
```

9.6 Planning

Ceci est le planning original prévu au début du projet.



Nom	Date d...	Date d...
Projet d'Approfondissement	19.09.17	19.01.18
Du 19 sept. au 16 oct.	19.09.17	16.10.17
Vacances d'octobre	17.10.17	20.10.17
Du 23 oct. au 22 déc.	23.10.17	22.12.17
Vacances d'hiver	25.12.17	05.01.18
Du 8 janv. au 19 janv.	08.01.18	19.01.18
Documentation	19.09.17	19.01.18
Rédaction du cahier des cha...	19.09.17	22.09.17
Élaboration du planning	25.09.17	02.10.17
Rédaction du rapport final	08.01.18	19.01.18
Apprentissage des outils	25.09.17	06.10.17
Tutoriel sur les PWA	25.09.17	06.10.17
Analyse et spécification	25.09.17	27.10.17
Choix d'un boilerplate	25.09.17	16.10.17
Maquette de l'interface grap...	02.10.17	06.10.17
Test de l'API existante	09.10.17	13.10.17
Spécification de l'API	12.10.17	27.10.17
Développement	30.10.17	12.01.18
Must have	30.10.17	22.12.17
Emprunt de matériel	30.10.17	10.11.17
Retour de matériel	06.11.17	17.11.17
Test emprunt et retour	30.10.17	24.11.17
Recherche d'un appareil	20.11.17	01.12.17
Test recherche appareil	20.11.17	01.12.17
Afficher détails appareil	27.11.17	08.12.17
Test affichage détails	27.11.17	08.12.17
Lecture codes à barres et ...	11.12.17	22.12.17
Test lecture code à barres ...	11.12.17	22.12.17
Nice to have	11.12.17	12.01.18
Login utilisateur	11.12.17	12.01.18

9.7 Journal de travail

Le journal de travail est disponible sur le dépôt GitHub à l'adresse suivante :

```
https://github.com/HE-Arc/Inventory-HE-Arc-Web-
App/blob/master/doc/Journal%20de%20travail.md
```