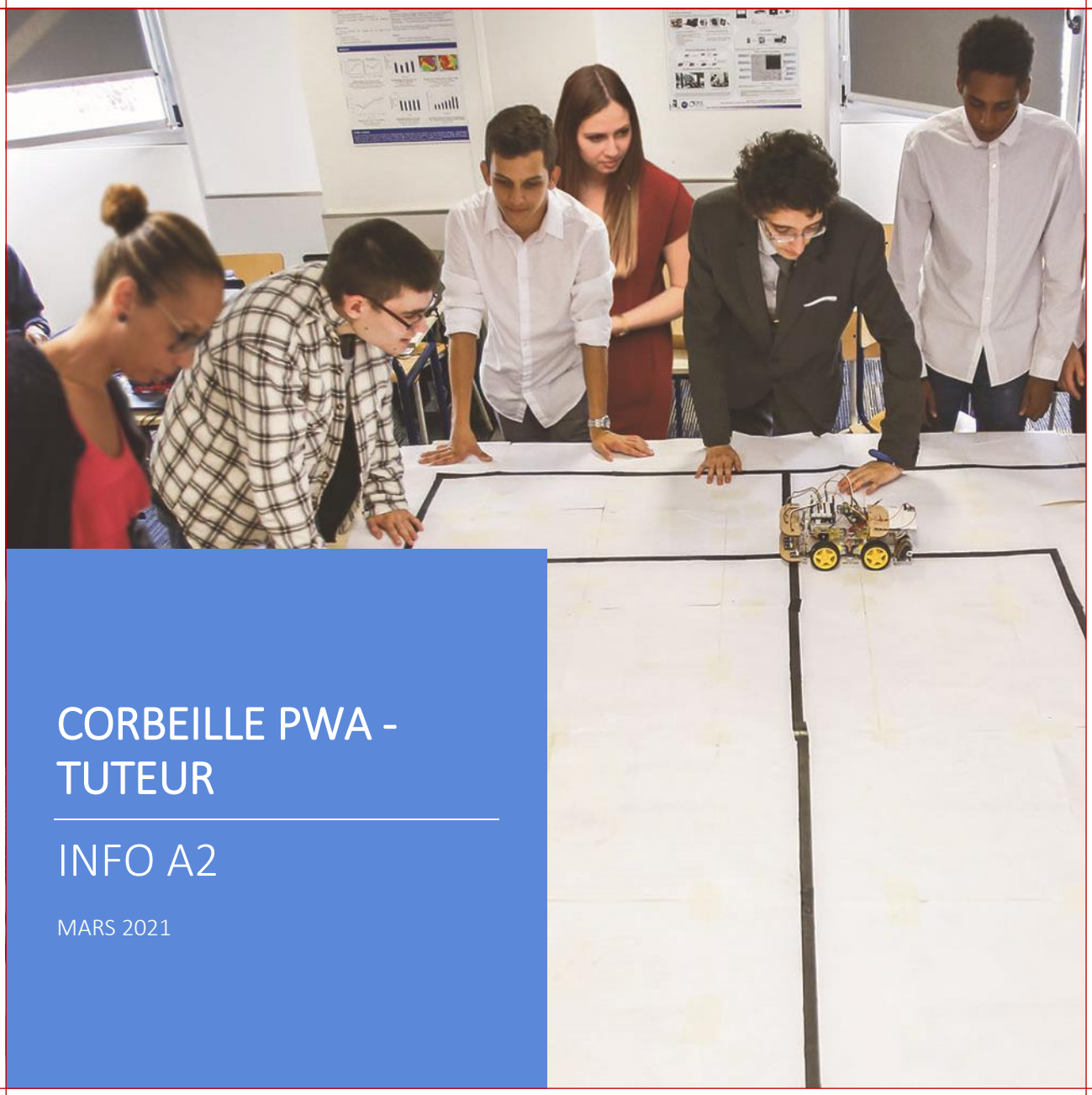


CORBEILLE PWA - TUTEUR

INFO A2

MARS 2021

cesi ÉCOLE D'INGÉNIEURS

SOMMAIRE

INTRODUCTION

Objectif

L'objectif de cette corbeille, c'est de pouvoir transformer un site web responsive (statique ou dynamique) en application mobile (sous Android).

Vous aurez besoin des outils suivant :

- Sublime Text
- Google Chrome
- Lighthouse (vérificateur ou validateur sur Google Chrome).

Structuration des fichiers

Pour pouvoir réaliser cette transformation, vous devez tout d'abord créer deux fichiers qui seront le cœur de votre transformation, à savoir le fichier manifest en JSON + le service worker en JS.

Conseil :

Vous devez structurer votre site web (POO).

Utilisez la fonction *include* de Php.

Ressources

Tutoriel (FR) :

https://www.youtube.com/watch?v=N4iksNmNKSI&list=PLs_WqGRq69UjvtIJ9qhMSL1WfxCFw2nB4

Tutoriel (EN) :

https://www.youtube.com/watch?v=4XT23X0Fjfk&list=PL4cUxeGkcC9gTxqJBcDmoi5Q2pzDusSL7

(Episode 1 à 22)

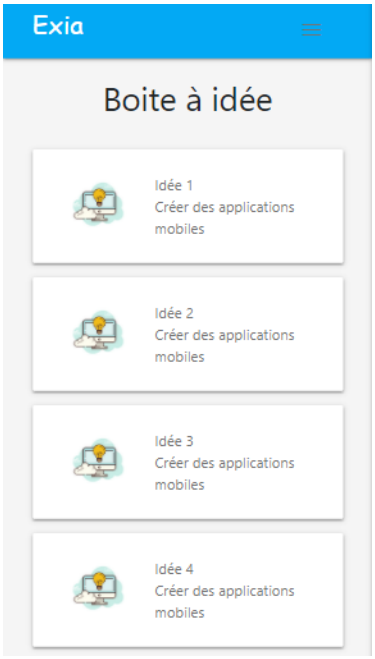
MISE EN PLACE

Etape 1 : vérifier la responsivité de votre site web.

Cette partie consiste à créer un site responsif, qu'on pourra visualiser sur ordinateur, mobile ou tablette. Je vous invite à découvrir la bibliothèque suivante :

https://materializecss.com/

Afin d'avoir plus d'idées, prenez exemple sur ceci :



C'est une petite application web qui fait la fonction d'une boite à idées, mais le choix vous revients en ce qui concerne l'application web à mettre en place, le plus important c'est qu'elle soit responsive.

Etape 2 : Création du manifest.JSON

Le fichier manifest doit contenir les informations nécessaires pour le bon fonctionnement de notre PWA.

Voici à quoi votre manifest doit ressembler :

```
"name": "Exia Boite à Idée",
 "short_name": "Exia BI",
 "start_url": "index.html",
 "display": "standalone",
 "background-color": "#FFFF",
 "theme_color" : "#03a9f4",
 "orientation": "portrait-primary",
 "icons": [
   {
     "src": "./img/logo.png",
     "type": "image/png",
     "sizes": "72x72"
   }
 ]
```

Etape 3 : Sychronisation des pages.

Chaque page doit avoir un lien vers le manifest afin que votre appareil mobile comprenne qu'il existe plusieurs pages dans notre application web.

Lien vers le manifest : **<link rel="manifest" href="manifest.json">**

Et si vous avez plusieurs pages, devriez-vous ajouter le lien dans chaque page ?

La réponse est NON, à vous de découvrir comment faire en sort que si vous voulez rajouter un lien dans l'entête de vos pages web, il vous suffit de le rajouter 1 seule fois.

NB : Php pourra vous aider !

Questions : manifest.JSON

1/ A quoi sert un manifest ?

Il est comme une carte d'identité de l'application web

2/ Donnez une explication pour chaque item.

A voir dans les ressources !! juste pour un rappel.

3/ Vous devez renseigner combien d'icons ?

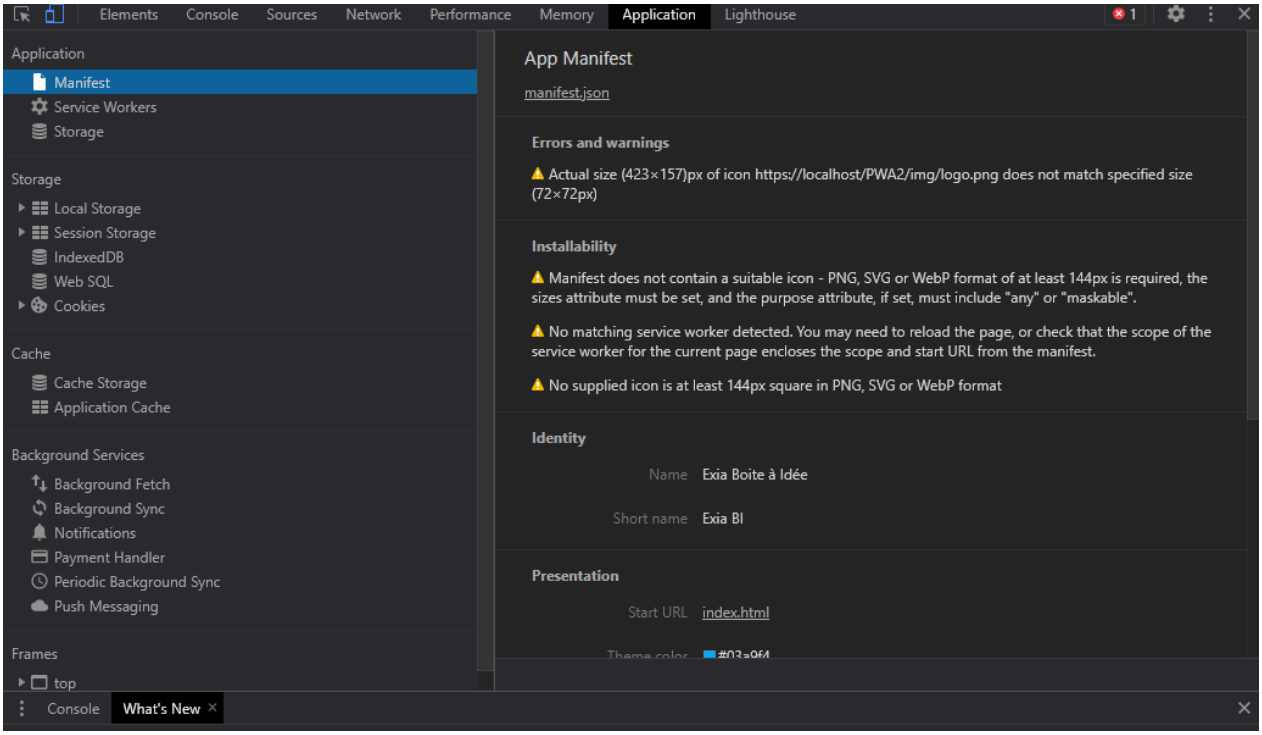
Le manifest doit contenir 7 types d'icons avec les tailles suivantes : "96x96" & "128x128" & "144x144" & "152x152" & "192x192" & "384x384" & "512x512"

4/ Est-il important d'avoir un format minimum du manifest pour que notre PWA soit au point et avec quoi vérifier ?

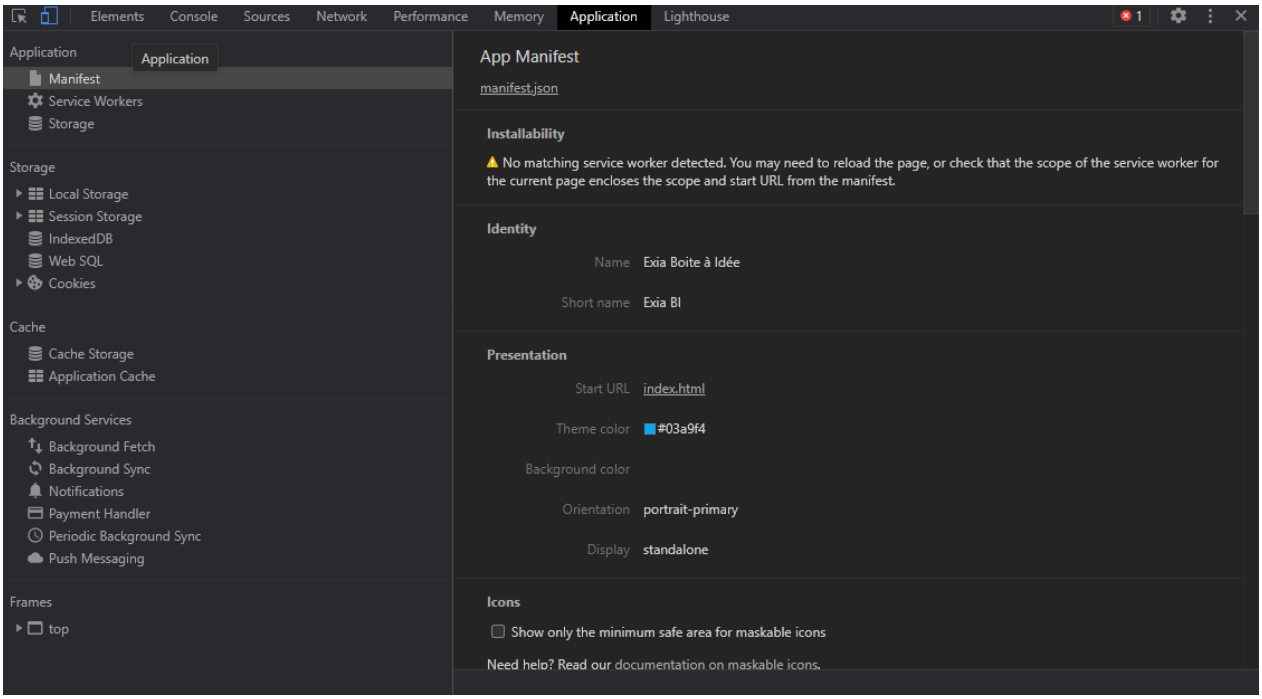
Oui il est important, et on pourra faire une vérification avec lighthouse.

Etape 4 : Vérification du manifest par l'inspecteur d'élément.

Une fois sur votre application web, vous devez aller dans l'onglet application dans inspecter l'élément et cela devra vous afficher ceci :



Vous remarquerez qu'il y a plusieurs warning dans l'onglet « installability », vous devez résoudre chaque warning afin d'avoir le résultat suivant :



Question : Qu'est ce que ça veut dire installability ?

Permet de voir si Google Chrome pourra générer le format APK (pour mobile) de notre application web ou pas.

Une fois que vous avez ce résultat on passera à l'étape 5.

Etape 5 : Création du service worker.js

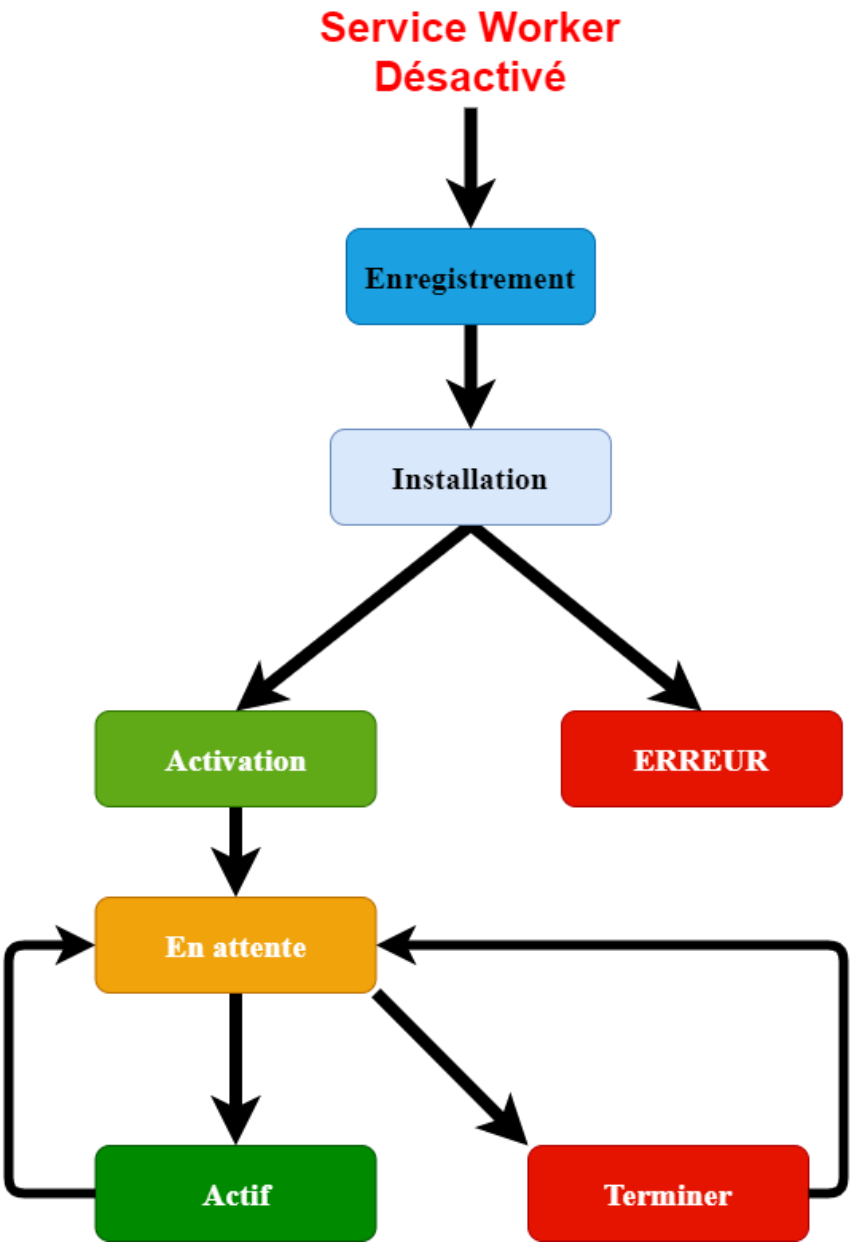
Comme vous l'avez remarqué dans l'image précédente, le Service Worker n'est pas disponible, pour cela vous devrez le créer.

Question : Où (dans l'arborescence du projet) devez vous mettre votre fichier (script) serviceworker.js

Afin qu'il fonctionne parfaitement et pourquoi ?

Il doit être sur la racine afin de pouvoir mettre en cache tous les fichiers appartenant à cette racine.

Rappel : Cycle de vie d'un Service Worker



Pour mieux comprendre je vous invite à voir la ressource suivante :

https://www.julienpradet.fr/fiches-techniques/pwa-declarer-un-service-worker-et-gerer-son-cycle-de-vie/

Etape 6 : Suivre les étapes de création d'un service worker à travers son cycle de vie

- Création d'un script en js, afin d'invoquer ou d'enregistrer votre service worker, voici un exemple de code d'enregistrement du service worker (que votre script doit contenir) :

```
if('serviceWorker' in navigator){
   navigator.serviceWorker.register('ServiceWorker.js')
   .then( (sw) => console.log('Le Service Worker a été enregistrer', sw))
   .catch((err) => console.log('Le Service Worker est introuvable !!!', err));
}
```

- Vous pouvez ajouter le code ci-dessus soit directement dans votre page principale html ou php, sinon en utilisant le lien suivant : **<script src="js/script.js"></script>**

NB : pour l'instant on n'a pas encore créé le Service Worker.

Etape 7: L'installation du service worker.

- Tout d'abord vous devez créer un script qu'on appellera **ServiceWorker.js**.
- Ensuite, vous devez suivre le cycle de vie de service worker, d'où vous arrivez à l'étape installation.
- Pour cela vous pouvez utiliser le code suivant :

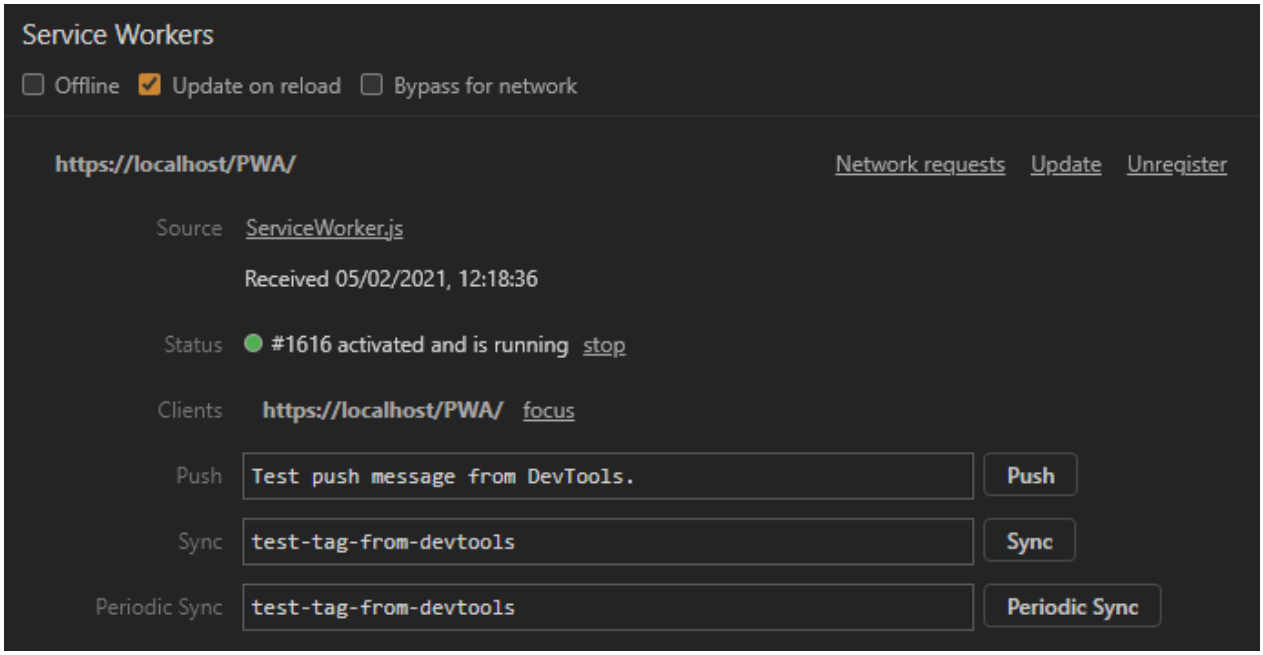
```
//Installation du service worker
self.addEventListener('install', evt => {  //à voir plutard });
```

Etape 8: L'activation du service worker.

- Après l'installation vient l'activation, voici un exemple de code :

```
self.addEventListener('activate', evt => {
   console.log('le Service Worker a été installé ');
});
```

- Après l'activation, vous devez avoir ceci sur inspecter l'élément > Application > Service worker



- Vous remarquerez que le localhost est en HTTPS et non pas en HTTP.
- Un petit exercice, vous devez chercher comment mettre votre Xampp en HTTPS et non pas en HTTP, suivez ce tutoriel afin de mieux comprendre : https://medium.com/@ajtech.mubasheer/setup-https-in-xampp-for-localhost-bc3d01393f31
- Une fois le service worker activé, il se met en attente (IDLE) automatiquement, n'hésitez pas à actualiser votre site afin d'éviter tout problème, ou même d'utiliser le navigateur privé.

Etape 9: Fonctionnement du service worker.

- **Question :** Quelle est l'utilité du service worker ?
  Permettre de faire fonctionner notre application mobile en mode offline.
- Une fois que vous savez répondre à la question précédente, vous passerez au mode Actif du service worker, pour cela vous avez besoin de créer un espace de stockage de données afin que le service worker utilisera plus tard pour sauvegarder ses données, pour cela vous devez revenir à l'étape installation du SW et mettre le code suivant :

```
//Installation du service worker
self.addEventListener('install', evt => {

    evt.waitUntil(  caches.open(NomDuCache).then(cache => {
      cache.addAll(assets);
      })
   )

});
```

- La variable NomDuCache est le nom de votre espace de stockage, et on remarque que la ligne « cache.addAll(assets) ; » ajoute un objet à votre espace de stockage.
- **Question** : Que doit contenir la variable assets ?
  Tous les fichiers que le service worker doit garder en cache.

- Une fois ces étapes passées, vous arriverez au cœur du service worker, où vous allez faire un « **fetch** », afin de répondre aux requête de l'utilisateur, en mode hors ligne.
  Exemple de code :

```
//fetch event afin de répondre quand on est en mode hors ligne.
self.addEventListener('fetch', function(event) {
   event.respondWith(
    caches.open('ma_sauvegarde').then(function(cache) {
      return cache.match(event.request).then(function (response) {
        return response || fetch(event.request).then(function(response) {
          cache.put(event.request, response.clone());
          return response;
        });
      });
    })
  );
 });
```

- Dans ce code vous remarquerez que le service worker en mode hors-ligne devient en quelques sorte le serveur avec qui l'utilisateur échange des requêtes, avec une limitation d'accés, vu qu'il n'a pas accès à votre véritable base de donnée vu qu'elle est en ligne.

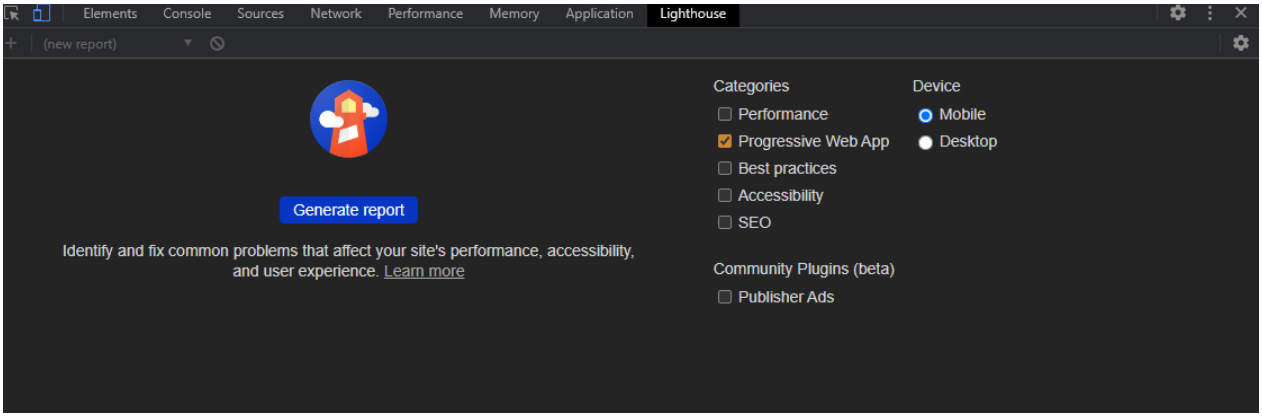
NB : n'oubliez pas d'utiliser les ressources citées en haut.

Etape 10: Validateur Lighthouse

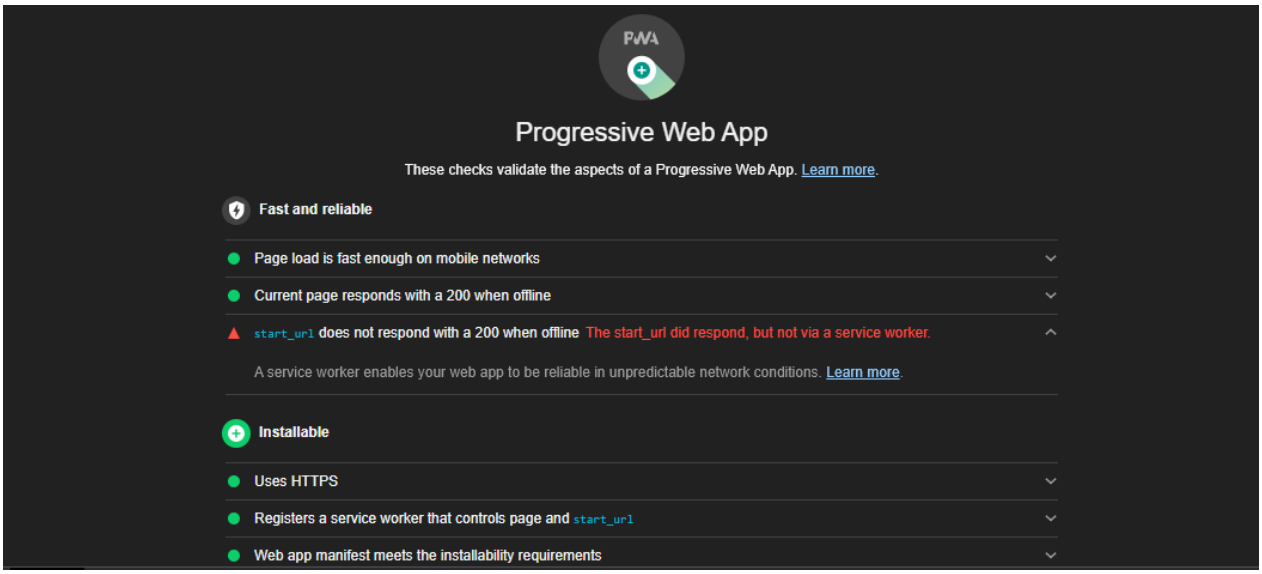
Dans cette étape vous devez d'abord télécharger lighthouse (plugin Google Chrome) sur le lien suivant :

https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk

Une fois Lighthouse installé, vous pouvez voir l'état de votre site en matière de PWA avec un scan de Lighthouse sur inspecter l'élément :



Une fois que vous cliquez sur ***generate report*** :



Vous remarquerez que l'outil vous liste tout ce que vous devez faire pour avoir une PWA fonctionnelle à 100% et pour cela vous pouvez cliquer sur Lean more pour chaque étape non-réalisée, afin de savoir comment faire pour la résoudre.

Vous remarquerez aussi que la section « installable » est en vert donc l'application est prête à être installée sur mobile. (via Google Chrome).

Question : Veuillez définir les différents critères de chaque section de lighthouse afin d'avoir une PWA fonctionnelle à 100%.

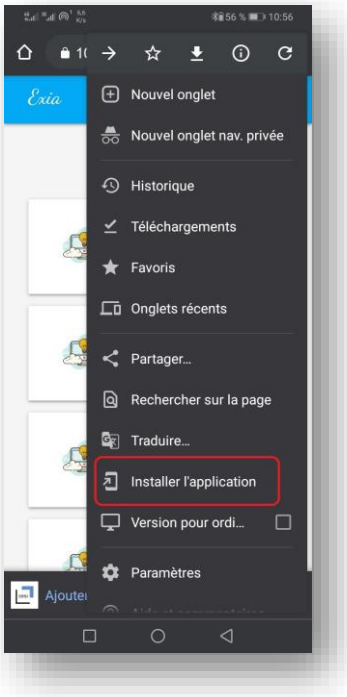
https://web.dev/lighthouse-pwa/

Etape 11: Test sur un emulateur android.

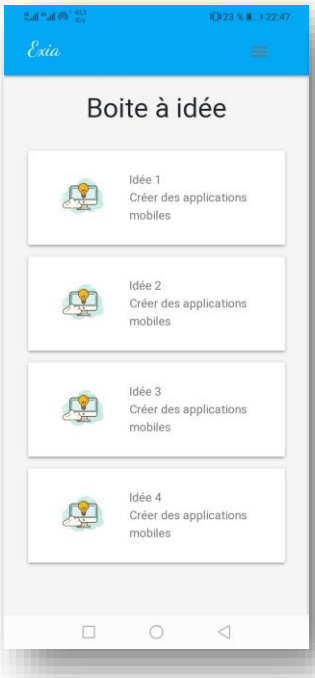
Vous pouvez télécharger Android studio afin de lancer son emulateur mobile et pouvoir tester votre application directement sur un émulateur et voir le résultat de votre travail.

Voici ce que vous devez avoir :

1- Se connecter à l'adresse local de votre émulateur (dans ce cas c'est 10.0.2.2).



Vous remarquerez l'icon de l'application Exia BI est disponible sur le bureau du téléphone comme si une application mobile simple.



Une fois l'application installée et lancée, vous remarquerez qu'elle fonctionne de la même façon qu'une application mobile développée en Native ou avec un Framework.

Il faut savoir que l'implémentation des PWA sur Android est plus rentable que sur IOS d'où l'orientation de cette corbeille d'exercice.

Voila votre application mobile Android, Bravo !!!

CAMPUS
D'ENSEIGNEMENT SUPÉRIEUR
ET DE FORMATION PROFESSIONNELLE