Résultat final du test technique :

1. Sujet du test

- Construire une mini-app VueJS ou Angular 2 (ou autre framework) qui affiche une liste, puis une page pour chaque Item (avec nom, supplier, prix par exemple).

- Afficher cette liste des items avec un filtre par prix/croissant/décroissant en JS et une recherche par nom de produit.

- Stocker le JSON en asset dans l'architecture de l'appli elle-même (chargé en ajax par exemple).

- Utiliser le localstorage pour se souvenir de la dernière fiche consultée (dans ce cas afficher un lien en haut de la liste).

- Committer le projet expliqué sur Github.

- L'app se compose donc de deux pages maximum : une vue liste et une vue détail produit, d'une recherche et d'un filtre par prix.

2. Réalisation :

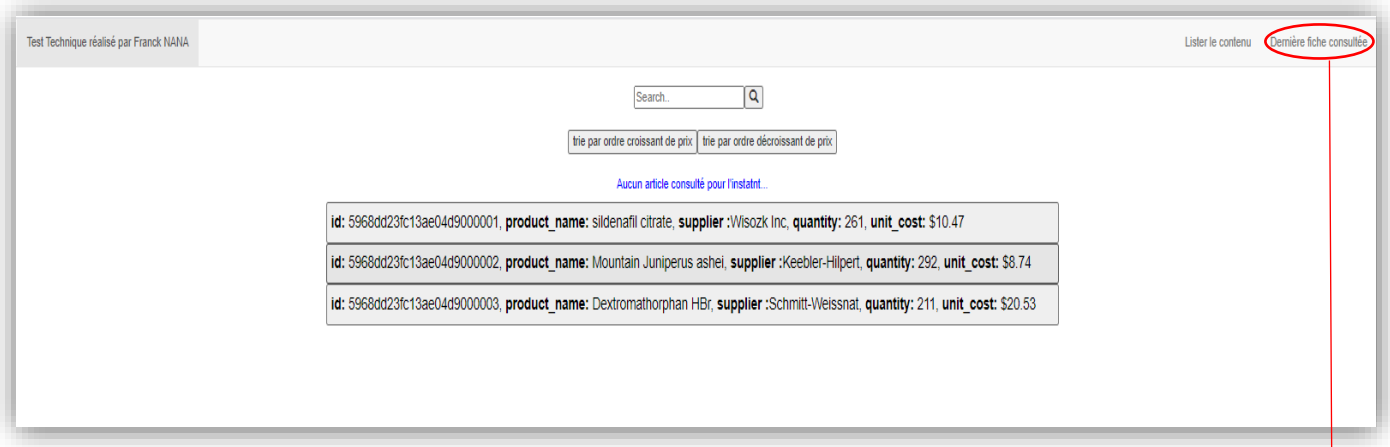
Pré-requis

- Angular 8.1.3
- Angular CLI 8.1.3
- bootstrap 3.3.7
    * commande d'installation: npm install bootstrap@3.3.7 --save
    * Ajouter qussi la ligne suivante dans Angular.json => style
        ./node_modules/bootstrap/dist/css/bootstrap.css"
- installer le package rxjs
    * commande d'installation: npm install rxjs-compat --save
- Version Node 10.16.3

3. Présentation

En lançant l'application on aboutit à la page d'accueil suivante

En cliquant sur **Lister le contenu** on obtient la liste total des articles disponibles dans le fichier json.



Dans ce nouvel affichage on peut cliquer sur un article et avoir un affichage détaillé de l'article dans une nouvelle page avec un bouton de retour.

On peut aussi filtrer les articles par ordre croissant ou décroissant du prix et aussi faire une recherche sur un nom d'article.

Le bouton **dernier fichier consulté** permet de retrouver le dernier article consulté préalablement

4. Détails techniques

- Copier le fichier json dans l'asset de angular et l'ajouter de façon statique dans angular.json => assets.
- Créer un model class contenant des champs représentant les champs décrivant un article. Cela permettra de pouvoir récupérer les données du fichier json
- Créer deux components représentant les deux pages indiquées par le sujet et configurer le routing.
- (facultatif) créer une barre de navigation avec les options : **lister le contenu et dernier fichier consulté.**
- Créer des services pour gérer la lecture et le partage des données issues du fichier json. Ces services contiendront d'autres méthodes utiles comme les tris et la recherche