RISK VIEWER

Manuel développeur

RESUME

Ce manuel développeur a pour but d'expliquer les choix de développements lié à l'application Risk Viewer.

Adrien Mousty

Test technique

TABLE DES MATIERES

COMMENT TESTER L'APPLICATION

Il existe plusieurs façon de tester cette application. Néanmoins, ici je ne vais développer que deux façons de faire.

SERVEUR LOCAL

L'installation d'un serveur local est relativement aisé. Il suffit d'entrer trois lignes dans la commande.

1. Appuyer sur la toucher Windows + r
2. Taper « cmd » puis appuyer sur enter.
3. Exécuter la commande «npm[1] install http-server -g »
4. Se rendre dans le dossier du projet via la commande « cd + <chemin pour accéder à l'application[2][3]> »
5. Exécuter la commande « http-server »
6. Laisser tourner la console.
7. Ouvrir un navigateur et taper « localhost :8080 »

A partir de cette étape, vous pouvez profiter de l'application et de la tester en local. Le point d'entrée de cette application est le fichier « index.html ».

APPLICATION HEBERGEE

Cette solution est la plus simple. Il suffit de se rendre sur http://moustyadrien.000webhostapp.com/ et de tester l'application. Néanmoins, cet hébergeur effectue une maintenance une heure par jour. Pendant ce temps, Risk Viewer n'est pas disponible.

[1] https://www.npmjs.com/package/http-server
[2] Téléchargeable ici : https://github.com/amousty/Patient_Risk_Comparison
[3] Par exemple : *cd C:\Users\moust\Documents\GitHub\Patient_Risk_Comparison*

TECHNOLOGIE & LIBRAIRIES UTILISEES

Risk Viewer est développé en JavaScript. J'ai fait ce choix car ce langage est doté de nombreuses librairies et est simple a héberger. Une design pattern a été utilisé, le « Module Design Pattern[4] ». Son avantage est qu'il est relativement simple à mettre en place et permet de protéger une partie du code en ayant des variables privées, ou publiques. C'est aussi une façon « détournée » de faire de l'orienté objet en JavaScript. Avec un système de classe.

LIBRAIRIES

JQUERY

jQuery est une librairie incontournable en JavaScript permettant d'effectuer des effets dynamiques sur les pages WEB.

Version : 3.2.1

Lien : https://jquery.com/download/

BOOTSTRAP

Bootstrap offre un système de design par classe largement utilisé par les développeurs. Il permet aussi de positionner facilement les éléments sur la page, et de manière responsive.

Version : 4.0.0

Lien : https://getbootstrap.com/docs/4.0/getting-started/download/

CHART.JS

Chart JS est largement utilisé dans cette application. Il gère tout ce qui est lié au graphique.

Version : 2.7.1

Lien : https://github.com/chartjs/Chart.js/releases/tag/v2.7.1

DATATABLE

Datatable est l'autre librairie fortement utilisée lors du développement de Risk Viewer. Elle transforme une simple table HTML en un véritable dataset incluant des fonctions de recherche, tri et paging. Elle peut être combinée avec d'autres librairies telle que Boostrap, pour une rendu de meilleur qualité.

Version : 1.10.16

Lien : https://datatables.net/download/index

[4] https://scotch.io/bar-talk/4-javascript-design-patterns-you-should-know#toc-module-design-pattern

FONTAWESOME

Cette petite librairie CSS offre de nombreux choix d'icônes. Une manière simple d'optimiser l'affichage de Risk Viewer. Tout en la rendant plus « user friendly ».

Version : 4.7.0

Lien : http://fontawesome.io/

FAQ

- **Pourquoi avoir tout fait en anglais ?**

Car l'anglais est la langue par défaut lorsque l'on développe. Elle permet aussi de rendre son code plus facilement compréhensible par n'importe qui.

- **Qu'as-tu fais pour optimiser l'application ?**
    - La rendre la plus petite et simple possible
    - Utilisation de fichiers minifiés. Mais les fichiers JS normaux restent accessible car beaucoup plus lisibles.
    - J'ai tenté de diminuer les appels aux « data[5] », car celles sont déjà importantes, 500 enregistrements.
    - J'ai partagé mon application à mon cercle d'ami, développeurs ou non. Afin d'avoir un feed-back objectif. Ceux-ci m'ont permis d'effectuer certaines modifications intéressants (ajout de tooltips, etc.)

- **Pourquoi ce choix de design ?**

C'est un choix personnel. J'aime beaucoup le style « épuré et simple » de manière générale. De plus, complexifier inutilement l'application, sans avoir reçu d'indications au préalable là-dessus aurait rendu cette application un peu « hors sujet ».

Pour ce qui est des couleurs, le logo du GHdC est bleu, c'est pour cette raison qu'une partie de l'application est bleue. Pour rester cohérent.

[5] Fichier JSON