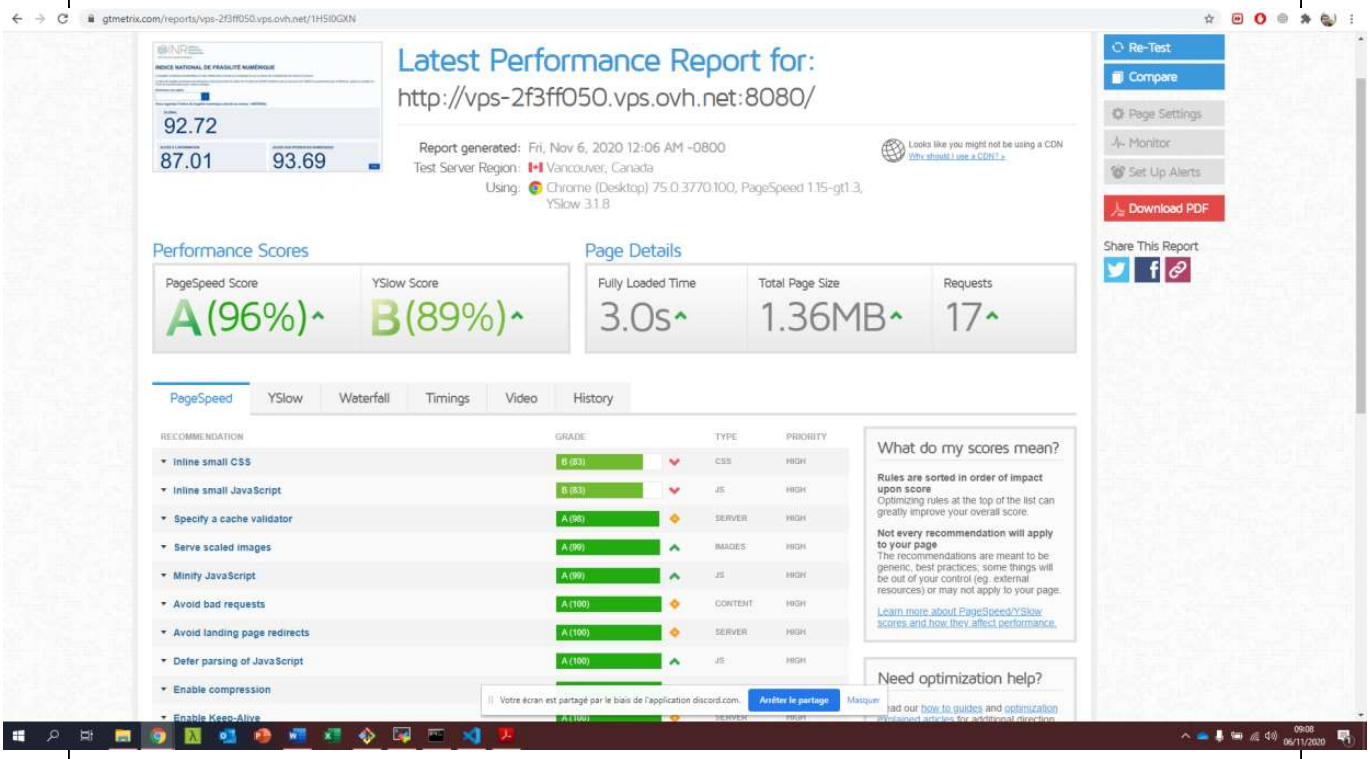
SYNTHESE DU PROJET CHALLENGE DESIGN4GREEN 2020
REPORT

Numéro d'équipe / Team Number : 58

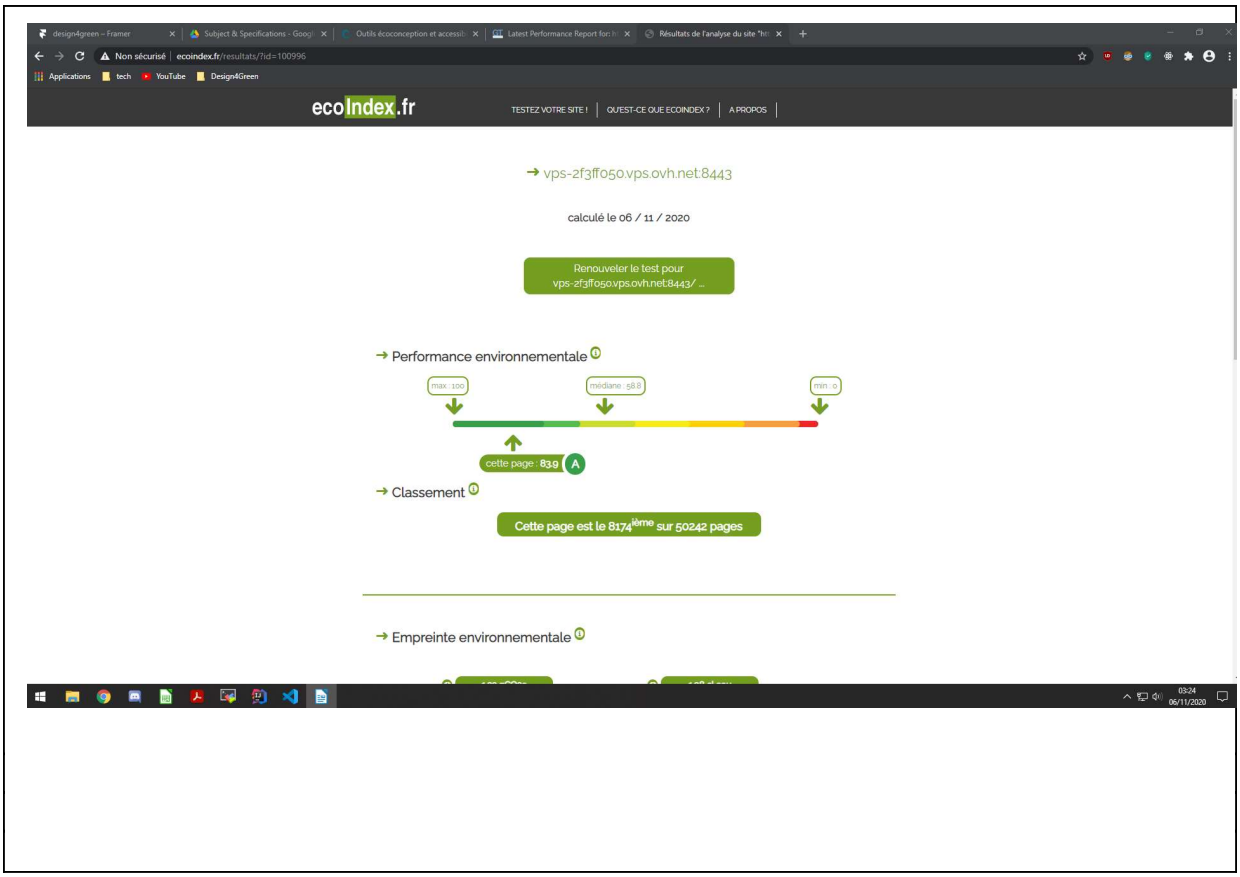
GT MTERIX

SCORE (PageSpeed Score) : 95% (only percentage)
SCREENSHOT (with Day and time)



ECOINDEX

SCORE (Performance environnementale / Environmental performance) : 83,9/100
SCREENSHOT (with Day and time)



ECOGRADER

SCORE : …… / 100
SCREENSHOT (with Day and time)

SONARQUBE

GITHUB URL :

- Front : https://github.com/SGecko-Design4Green/design4green-front-public
- Back : Notre back est codé en RUST, langage non pris en charge par SonarQube.

Conception générale – General conception

Avez-vous réussi à finaliser votre projet ? Did you manage to finish your project ? Oui Yes / Non No

Non.

Si non, pourquoi et quels éléments sont manquants ? if not, why and what is missing ?

Notre projet est globalement finalisé d'un point vue front-end, back-end et stockage de données, seules quelques fonctionnalités n'ont pas été réalisées.

Nous avions pour objectifs de :
- proposer une sélection d'items personnalisés grâce à la géolocalisation (« quels sont les villes autour de moi avec lesquelles je pourrais comparer mes indices »).
- proposer un téléchargement personnalisé des données (en partie implémenté sur le backend).

Nous avons préféré ne pas délivrer ces fonctionnalités au profit de la qualité de l'architecture applicative ainsi que de la performance énergétique associée

Conception technique – Technical conception

Quel langage avez-vous choisi et pourquoi ? which language did you use and why ?

Deux langages ont été utilisés pour concevoir l'ensemble de notre architecture :

- Pour la partie Front-end :
Le langage Javascript avec le support de la librairie **NextJS** a été utilisé. Ce choix a été motivé par la capacité de la librairie a pouvoir délivrer une expérience utilisateur moderne et complète tout en proposant des fonctionnalités limitant la pollution de l'espace numérique. Si l'on devait citer quelques fonctionnalités phares: pages générées en Server-Side-Rendering (le rendu des pages est calculé par le serveur et ensuite délivré au client) et Optimisation des images).

source : https://nextjs.org/

- Pour la partie Back-end :
Le langage **Rust** a été utilisé pour la réalisation de l'API en back-end, du serveur de fichiers ainsi que celle de la base de données. Malgré sa courbe d'apprentissage très élevée, notre équipe a été plus que séduite par ce langage émergeant notamment à cause de ses performances élevées, sa faible consommation énergétique ainsi sa stabilité exemplaire assurant une tolérance zéro sur les fuites de mémoires.

source : https://www.rust-lang.org/

Comment avez-vous optimisé vos requêtes ? How did you optimize the query ?

Nous avons essayé d'optimiser nos requêtes sur chacune des couches techniques traversées afin d'alléger au maximum les interactions nécessaires.

Notre postulat de base était de partir sur un projet autonome : une application capable de servir un site en JS?

- Sur la partie Front-end (NextJS):
- Server-side-rendering
- Requêtes pré-cachées.

- Sur la partie API (RUST):
- Traduction conçise du usecase en API selon le modèle REST afin d'éviter les appels multiples et inutiles.
- Factorisation des resources clés de l'application et définition fine des rôles de chaque composant grâce à une conception en architecture hexagonale et aux principes SOLID.
- Activation du cache duration sur les point d'accès de l'API pour limiter les interactions des clients au strict minimun.

- Sur la partie serveur de fichiers (RUST)
- Activation du cache duration sur les fichiers pour limiter les interactions avec des clients au strict minimum.

- Sur la partie base de données (RUST)
- Analyse du modèle de données pour en extraire les index les plus pertinents.
- La base de données a été créée sur mesure (RUST également) pour un flux de données contrôlée dans son ensemble (index optimisés en mémoire et système de stockage sur disque LSM-Tree).

Conception fonctionnelle – Functional conception

Avez-vous choisi d'utiliser un outil de représentation graphique ? Did you us a graphical representation ? Oui Yes / Non No

Si oui pourquoi ? if yes, why ?

Si non pourquoi ? if not Why ?

Pour ce besoin, et dans une approche minimaliste, il ne nous a pas paru nécessaire de mettre à disposition un moteur de représentation graphique en regard de la quantité d'indicateurs.

Les outils de représentations graphiques bien qu'intéressants pour vulgariser une situation ou un contexte, peuvent se révéler très consommateurs de données si mal utilisés. L'idée d'avoir une application tableau de bord ne nous séduisait pas également à cause de la masse de statistiques sachant que n'étions pas sûr que la majorité des utilisateurs les consomment correctement (selon le principe de Pareto, la règle des 80/20).

Design

Expliquez en quelques mots les choix réalisés au niveau du design du site? Explain your design choices ?

Nous avons donc pris le parti de vouloir en afficher moins afin de sensibiliser plus, tout en limitant la casse énergétique.

Pour ce faire nous proposons un design minimaliste mettant en avant les indicateurs clés en rapport IFN et rappelant la charte graphique du site initial.

Accessibilité

Qu'avez-vous mis en place pour le respect de l'accessibilité du site? How did you manage the accessibility of your site ?

La rigueur imposée par NextJS dans la création de contenu nous assure une accessibilité minimum. Nous avons essayé d'utiliser des composants simples pour une meilleure lecture.

QUESTIONS GÉNÉRALES – GENERAL QUESTIONS

Qu'est ce qui fait que votre site est éco-conçu? Why your solution is ecodesign ?

Lors des premiers échanges de notre équipe autour de l'éco-conception (que nous ne maîtrisions pas), nous avons tous été étonnés de constater que la majorité des informations sur le sujet mettaient essentiellement en avant la partie Web.

Nous avons tous été portés par la simple idée que l'éco-conception ne devrait pas s'arrêter qu'au front-end et c'est à partir de ce postulat que nous avons développé notre site.

Nous avons essayé d'appliquer ces principes sur l'ensemble de la stack applicative (de l'utilisateur jusqu'aux données, en passant par le serveur) afin proposer une solution optimisée à toutes les étapes de la chaînes et même au delà quand on pense à la maintenance.

Pour commencer notre site, comme décrit précèdemment, tourne uniquement grâce à une application autonome développée en **RUST**.

Celle-ci possède trois fonctionnalités essentielles:

- Servir le site Web développé en **NextJS** grâce à un serveur Web embarqué.
- Servir l'API Rest de backend (multi-threadé).
- Servir la base de données (embarqué).

Nous avons donc travaillé sur les axes suivants :

Réduire notre empreinte Web via l'eco-conception du site en lui-même.

Réduire nos dépendances technologiques induites liées au Web en créant une application autonome embarquant ces élèments (pas serveur web externe, pas de serveur de base de données, etc…). Moins d'intermédiaires, moins de configurations, plus de contrôles sur les données, ce qui est synonyme de moins de pollution sur les réseaux.

Réduire la consommation énérgétique liés à la conception à court terme en utilisant des technologies consommant peu de ressources. Par exemple, l'utilisation du RUST (étant défini selon de récentes études comme un des langages le plus eco-friendly, *source en bas de l'encart*), nous a permis de délivrer une application légère (12Mo), consommant peu de mémoire vive et ayant des performances équivalente à un programme développé en C/C++ (sans garbage collector).

Réduire la consommation énérgétique liés à la conception à plus long terme en utilisant développant notre modèle d'application suivant un un modèle d'architecture hexagonale pour une maintenance maîtrisée. Nous avons égalements essayé d'appliquer certains enseignements du craftmanship comme l'application des principes de développements SOLID (par exemple, l'injection de dépendances).

Réduire notre consommation de matériel et avoir une attitude eco-responsable : par exemple sachant que nous étions hébergés chez OVH (disque dur SSD) et en prenant en compte que le jeu de données fourni est immutable à court/moyen terme, nous avons opté pour un type de base de données embarquée utilisant un stockage de type LSM-Tree (moins destructif pour les disques dur SSD que les bases de données traditionnel de type BTree).

Sources : https://thenewstack.io/which-programming-languages-use-the-least-electricity/

Avez-vous d'autres remarques pertinentes sur votre projet ? others comments on your project ?

Notre site propose une approche comparative basée sur certains indicateurs clés fournis.

La stack applicative mise en place nous a permis de recalculé les différents indicateurs sélectionnés agrégés pour les ventiler sur les niveaux suivants :

- National
- Régional
- Départemental
- Communal
- Quartier

Ce recalcul a été possible grâce à notre architecture et ce dans un temps relativement court (<120 secondes). Ces données n'ont été calculé qu'une seule fois pour tous les utilisateurs et pour ensuite être servies à la demande en des temps records grâce aux index.