

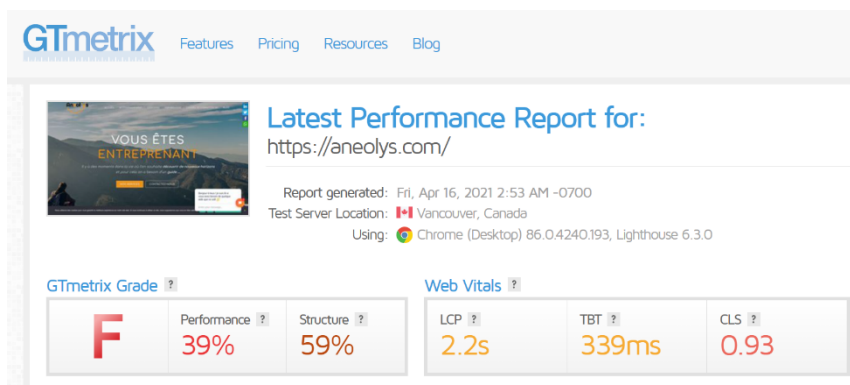
Processus pour améliorer la performance du site

L'amélioration des performances d'un site, va toucher le référencement du site par Google, mais surtout l'expérience utilisateur, qui sera améliorée.

Point sur les performances actuels :

Utilisation de deux analyseurs de performance de page web, **GTmetrix** et **Google Speed Page Insight**, pour avoir un bilan complet des axes d'amélioration. Je me suis concentré sur la page mère/principale du site, l'accueil, qui correspond à la vitrine de l'entreprise sur le Web (<https://www.aneolys.com/>).

Résultat GTmetrix (note de F à A, correspond à une performance globale, utilisateurs sur les mobiles et les ordinateurs) :



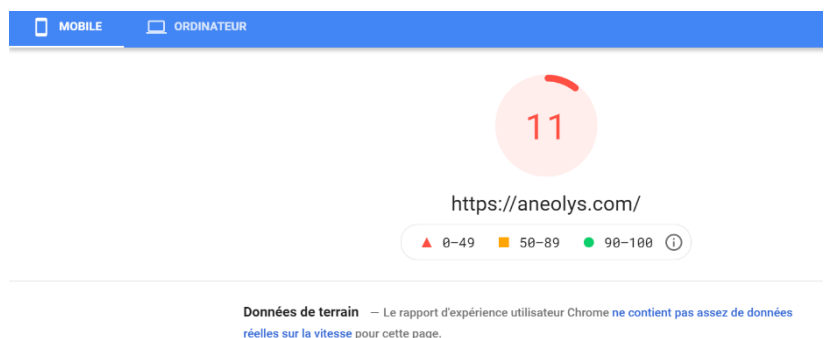
Résultat Google Speed Page Insight (note de 0 à 100):

Performance pour un utilisateur sur un ordinateur



Performance pour un utilisateur sur mobile.

Une place très importante aujourd'hui, on se concentre principalement sur l'utilisation sur mobile.



On voit que les résultats sortent le même constat, une performance utilisateur et système assez mauvaise (temps de chargements long, mauvaise structure du site (les liens, etc.), etc.). Les problèmes principaux qui revenaient à travers les deux analyseurs :

1. Les images bien trop lourdes et mal dimensionnées (correspondant à 2.5 MB sur 3.5 MB au total de la page)
2. Les ressources qui bloquent le rendu, l'affichage des pages (La partie JS, HTML et CSS)
3. Les urls ayant le statut 404 (une url ne menant à une page qui n'existe pas en interne ou en externe).

A savoir, on retrouve couramment avec le CMS Wordpress, le problème de ressources bloquant le rendu. Comme dis précédemment, le test s'est mené sur la page d'accueil, mais les changements ont été effectués sur toutes les pages du site.

1. Compression des images

En général, une image sur un site web ne doit pas dépasser un poids de 200 Ko. Je me suis basé sur cette donnée.

Pour compresser les images, je me suis dirigé simplement vers **des solutions en ligne**, avec :

- [Compressor .io](https://compressor.io)
- [iloveimg](https://iloveimg.com)

A la suite de la compression, plusieurs images dépassaient encore le seuil des 200 Ko. Le redimensionnement des images venait répondre à ce problème, car une grande quantité d'image était affichées dans des dimensions moindres à son origine. Le redimensionnement des images, permet de baisser ou augmenter le poids d'une image. Ce qui me convenait parfaitement, j'avais besoin de baisser leurs dimensions, ainsi que baisser leurs poids.

L'outil **Paint** m'a permis de **modifier les dimensions** efficacement des images.

Pour conclure cette partie consacrée aux images, je me suis concentré sur la **libération de stockage dans la BDD**, en **supprimant les images et médias non utilisés ou en doublons**. Le **plugin Media Cleaner** m'a facilité la tâche en analysant les médias qui ne sont pas utilisés, pour que je puisse les supprimer en suivant.

2. Les ressources bloquantes

Les ressources bloquantes correspondent, au contenu Javascript et CSS (HTML peut compter dedans mais n'est pas le principal) qui sont lu (de haut en bas) pour afficher une page web. L'affichage en simultané de ces trois types de données crée des ralentissements. Pour débloquer les ressources ou fluidifier la lecture et l'apparition des pages, je me suis dirigé vers deux plugins gratuits :

- **Async Javascript**

A savoir lors de la lecture d'une page, par défaut, il télécharge les données HTML, mais s'il rencontre une donnée CSS ou JavaScript, il va alors la télécharger et mettre en pause la partie HTML. Ce qui pose un problème, car il pourrait par exemple afficher le contenu HTML déjà chargé pendant qu'il télécharge le contenu Javascript ou CSS.

Ce plugin va alors permettre de choisir deux types de lecture des scripts JavaScripts seulement, le mode « **Async** » qui permet à l'analyseur HTML (par exemple, le *navigateur d'un visiteur*) de télécharger le JavaScript tout en analysant le reste du HTML. Autrement dit, il ne s'arrête pas

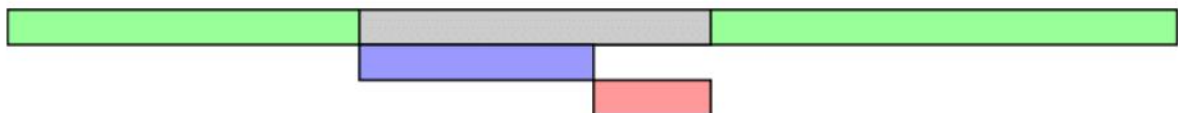
complètement pendant le téléchargement du fichier. Cependant, il mettra en pause l'analyseur HTML pour exécuter le script une fois qu'il aura été téléchargé. Ou le mode « **Defer** », permet à l'analyseur HTML de télécharger le JavaScript pendant l'analyse du reste du HTML **et** attend l'exécution du script jusqu'à ce que l'analyse du HTML soit terminée.

Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution

<script>

Let's start by defining what `<script>` without any attributes does. The HTML file will be parsed until the script file is hit, at that point parsing will stop and a request will be made to fetch the file (if it's external). The script will then be executed before parsing is resumed.



<script async>

`async` downloads the file during HTML parsing and will pause the HTML parser to execute it when it has finished downloading.



<script defer>

`defer` downloads the file during HTML parsing and will only execute it after the parser has completed. `defer` scripts are also guaranteed to execute in the order that they appear in the document.



J'ai choisi le mode « **Async** », qui permet d'exécuter les plusieurs scripts qui composent la page d'accueil lors des premiers affichages du contenu HTML.

- **Autoptimize**

Ce plugin donne la possibilité d'optimiser la lecture des contenus CSS, HTML ou JS. Mais comment ? En concaténant chaque contenu ensemble sur une même ligne, optimisant alors la lecture. J'ai mis en place cette optimisation sur le CSS seulement, car en concaténant les contenus HTML et JS, j'avais des problèmes (plus elementor dans la barre d'administrateur (JS) ou des balises HTML plus prises en compte (par exemple la section contact de la page d'accueil).

En liaison avec l'optimisation de la lecture des contenus, j'ai ajouté un plugin pour la mise en cache des pages pour les utilisateurs dans un temps de visites fixé. Utile pour les sites avec des pages qui se composent de données statiques, ce qui est le cas d'Aneolys. La mise en cache du navigateur peut aider à stocker les fichiers sur le navigateur de l'utilisateur. Lorsqu'un utilisateur va visiter le site Web pour la première fois, il prendra le même temps que d'habitude pour charger. Cependant, lorsque ce même utilisateur revient sur le site, ou va sur une autre page de votre site, il aura déjà certains des fichiers du site stockés sur leur ordinateur. Cela signifie que le navigateur de l'utilisateur doit télécharger moins de données et faire moins de demandes au serveur. Cela entraîne une diminution des temps de chargement de la page.

3. Les Url's avec un statut 404

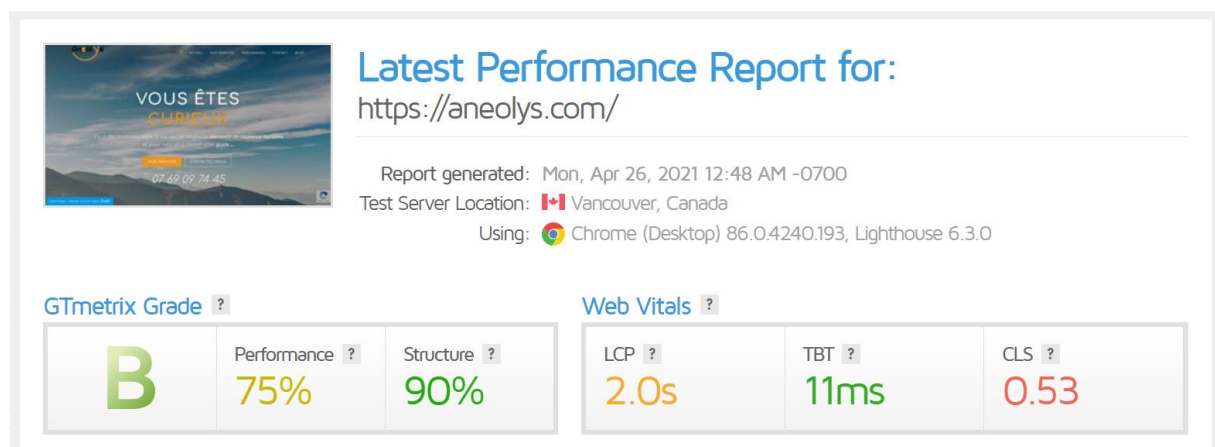
Troisième problème, celui des url's avec un statut 404, ce qui veut dire qu'elle redirige vers une page qui n'existe pas, cela peut concerner un lien interne ou externe. Pour vérifier toutes les redirections du site, je me suis tourné vers le **crawler Screaming Frog SEO**. Il analyse toutes les url's du site, puis indique les différents statuts, ce qui m'a permis par la suite de retrouver les liens morts, pour les mettre à jours.

4. Suppression/désactivation de plugins gourmands en ressources

Dernier problème, les ressources utilisé en trop grande quantité par les plugins (principalement des scripts js). J'ai pris la décision de désactiver Shareaholic (remplacé par le plugin Social Warfare, moins gourmand et s'affiche sur les articles seulement) et Tidio Chat (redondant avec le formulaire de contact).

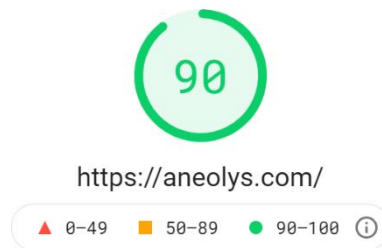
Points sur les performances après intervention :

Résultat GTMetrix (note de F à A, correspond à une performance globale, utilisateurs sur les mobiles et les ordinateurs) :

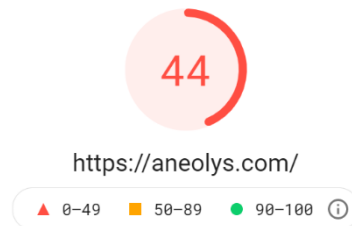


Résultat Google Speed Page Insight (note de 0 à 100):

Performance pour un
utilisateur sur un ordinateur



Performance pour un
utilisateur sur mobile



Des résultats de performance en hausse par rapport à la performance originelle, mais encore une performance assez basse concernant l'expérience d'utilisation sur mobile. Le futur axe d'amélioration doit se tourner vers le support mobile.