SYNTHÈSE DU PROJET CHALLENGE DESIGN4GREEN 2020

Numéro d'équipe : 25
Lien vers la solution : http://vps-adb8195f.vps.ovh.net/

1. ÉVALUATION PAR LES OUTILS AUTOMATISÉS

1.1. GT METRIX

Score (PageSpeed Score) : …%

score + capture + date

1.2. ECOINDEX

Score (performance environnementale) : …/100

score + capture + date

1.3. ECOGRADER

Score : …/100

score + capture + date

1.4. SONARQUBE

score + capture + date + lien Github

1.5. WAVE

score + capture + date

2. CONCEPTION

2.1. CONCEPTION GÉNÉRALE

Avez-vous réussi à finaliser votre projet ?

Si non, pourquoi et quels éléments sont manquants ?

réponse

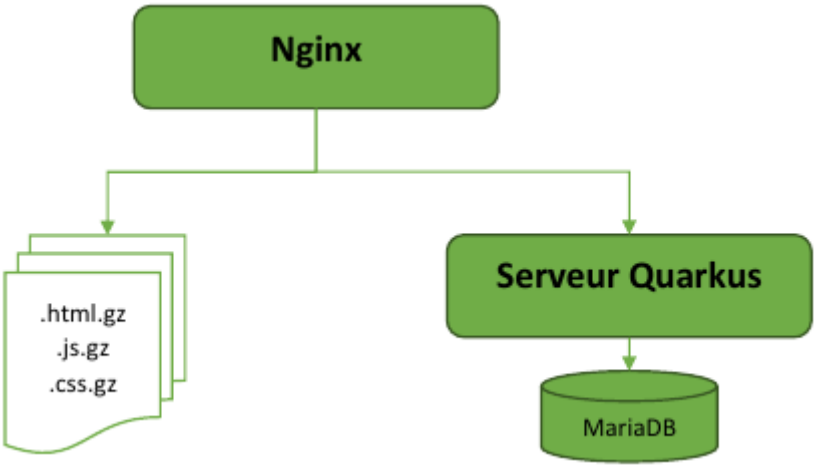
2.2. CONCEPTION TECHNIQUE

2.2.1. QUEL LANGAGE AVEZ-VOUS CHOISI ET POURQUOI ?

L'application se décompose en deux modules :

- le back, réalisant les interactions avec la base de données et retournant les résultats au format ReST au second module…
- le front, servant à mettre en forme les informations afin qu'elles soient facilement compréhensibles et interprétables par l'utilisateur.

Ces deux composants sont exposés par un serveur de fichiers Nginx.



Les critères généraux qui ont guidés nos choix techniques ont été :

- l'ouverture de la solution : nous avons privilégié les solutions libres et ouvertes, afin de ne pas enfermer notre application dans le carcan de formats propriétaires ;
- l'économie de ressource : nous avons cherché les solutions les moins consommatrices de mémoire, processeur et bande passante, afin de tenter de réduire l'empreinte énergétique de notre proposition.

BACK

QUARKUS

Notre back a été développé dans le langage Java, en utilisant le framework Quarkus.

Quarkus est un outil visant à accélérer Java et réduire son empreinte, notamment dans l'univers hautement distribué du *cloud computing*. Il offre notamment la possibilité — non exploitée pour ce challenge — de compiler une **application native**.

Lors de la compilation, Quarkus réalise une analyse exhaustive des chemins possibles et élimine tous les comportements dynamiques comme la réflexion Java, particulièrement consommatrice. Quarkus peut ainsi remplacer tous les codes dynamiques par du code pré-calculer et procéder à **l'élimination de tout code inutilisé**, aussi bien dans l'application que dans le JRE.

Le résultat est un JAR ou une application native plus rapide à lancer et moins gourmandes en ressources.

HIBERNATE

Si les ORM sont généralement regardés comme des ennemis de la sobriété, le couplage d'Hibernate à Quarkus permet de réduire l'empreinte de cette bibliothèque de par son mécanisme de construction.

Nous avons par ailleurs veillé à ne **pas déclarer les jointures au sein des entités**, afin d'éviter toute jointure inutile lors du requêtage de la base de données. Lorsque des requêtes sur plusieurs tables étaient nécessaires, les jointures ont été déclarées manuellement au sein des requêtes.

BASE DE DONNÉES : MARIADB

Nous avons étudié deux possibilités pour la base de données : PostgreSQL et MariaDB. S'il est complexe de trouver des chiffres détaillant les empreintes de l'une et l'autre, la plupart des comparatifs mettent en avant la complétude et la robustesse de PostgreSQL, et plusieurs articles soulignent que l'empreinte disque et mémoire de MariaDB est inférieure. On pourra notamment citer cet extrait :

> MariaDB has a considerably smaller size as compared to that of PostgreSQL in different OS versions. MariaDB is lighter, making it a preferred choice if you are looking for a lighter database and is short on memory allocation.

Le **système de gestion de base de données MariaDB** nous a semblé satisfaisant pour les besoins de ce projet et a donc été retenu.

SÉCURITÉ DU SERVEUR

Afin de sécuriser le serveur, nous avons entrepris plusieurs actions :

- Nous avons mis en place le **pare-feu UFW**, qui permet de s'assurer que seuls les ports et applications autorisés sont exposés.
- Nous avons voulu **exposer la solution en SSL**, mais n'avons pas pu. Le nombre de certificats demandés pour des sous-domaines d'`ovh.net` dépassait en effet la limite autorisée par *Let's Encrypt*.

FRONT

D'un point de vue général, nous avons souhaité construire notre interface web sans nous appuyer sur un framework existant afin de limiter le risque d'importer des ressources inutiles à notre cas. À quelques exceptions que l'on s'efforcera de mentionner, tous les scripts et styles sont nos œuvres.

NODEJS & NPM

Les outils créés dans le but d'exécuter du JavaScript sur les serveurs facilitent grandement la compilation de projets web. Nous nous sommes notamment basés sur NPM pour orchestrer tous les jobs que nous avons mis au point pour préparer le site web.

ASSEMBLAGE DE FICHIERS JAVASCRIPT : ROLLUP

Rollup est moins connu et moins répandu que Webpack. Cependant, des tests passés d'un membre de l'équipe ont permis de montrer des gains de 7 kio sur des scripts de taille raisonnable (gain de 30 % sur l'exemple en question).

Nous en avons conclu que **Rollup serait une solution plus économe** pour notre solution, dont les scripts devaient rester modestes.

RÉTROCOMPATIBILITÉ

Nos styles ont été écrits en CSS 3 et les scripts en ECMAScript 6. Afin de limiter le risque d'exclusion des utilisateurs restreints à des versions plus anciennes de navigateur, nous avons utilisés des outils de transpilation.

Ainsi, **Babel** a été utilisé pour transformer l'ECMAScript 6 en JavaScript plus habituel, et **Autoprefixer** a ajouté des instructions permettant la prise en compte des instructions CSS 3 pour des browsers utilisant des instructions préfixées (`-webkit-, -moz-`). Ces deux outils se sont basés sur la liste `defaults` de **browserslist**, qui inclut tous les navigateurs ayant plus de 0,5 % de parts de marchés et quelques autres.

Ce choix est un compromis entre inclusion et bande passante utilisée.

HARMONISATION DES STYLES

Un *reset CSS* est devenu habituel pour des applications web, afin d'assurer un rendu uniforme dans tous les navigateurs. Il est question de redéfinir tous les styles qui ont une valeur par défaut dans le butineur.

Nous avons préféré nous appuyer sur **normalize.css**. La subtilité est que cet outil ne vise que les styles différents d'un navigateur à l'autre, afin d'harmoniser le rendu en réduisant l'empreinte ajoutée par ce CSS.

RÉDUCTION DE LA BANDE PASSANTE UTILISÉE

Afin de réduire autant que possible la bande passante nécessaire à la consultation de notre solution, nous avons automatisé la minification de toutes les ressources que nous distribuons (HTML, JavaScript et CSS).

Ces ressources sont par ailleurs compressées avec l'algorithme gzip.

SERVEUR DE FICHIERS : NGINX

Nous avons fait le choix d'Nginx plutôt qu'Apache HTTPD pour sa gestion plus efficace des ressources.

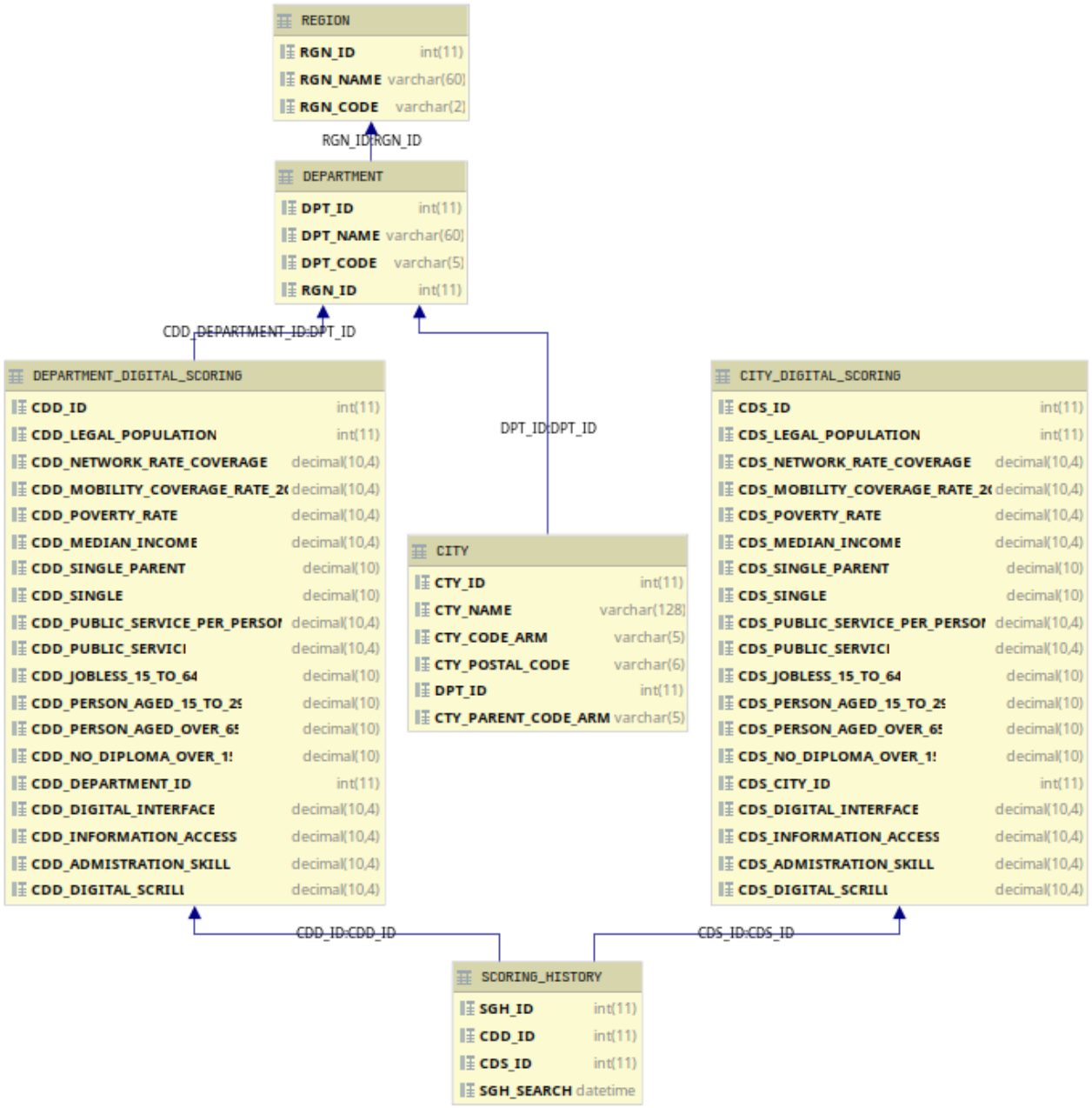
Le module `gzip_static` permet par ailleurs de pré-compresser les ressources avant le déploiement. Nginx peut ainsi servir les ressources compressées sans avoir à réaliser lui-même cette transformation, épargnant du temps processeur lors des réponses aux clients.

Si certaines ressources n'ont pas été pré-compressées, Nginx pourra toujours réaliser la compression à la volée.

2.2.2. COMMENT AVEZ-VOUS OPTIMISÉ VOS REQUÊTES ?

Ainsi qu'évoqué dans la partie précédente, les requêtes sont réalisées par le biais de l'ORM Hibernate. Cependant, contrairement à ce qu'il est habituel de faire, nous n'avons pas déclaré les jointures entre objets afin que l'ORM ne prenne jamais l'initiative de faire ces jointures.

Nous avons par ailleurs pris du temps pour déterminer notre MPD (modèle physique de données) et déterminer les meilleures façon d'optimiser notre schéma. Par exemple, nous avons envisagé d'avoir une seule table pour tous les scores, mais avons finalement préféré séparé les scores de villes des scores de départements afin de réduire le nombre de jointures.



Partout où une jointure est possible, nous avons défini une *foreign key*. Des index ont été également définis sur certains champs qui peuvent être utilisés pour des requêtes, à condition que ces requêtes puissent les exploiter (la recherche de ville par nom ne le permet pas car elle utilise un critère `like %name%` à des fins d'ergonomie).

2.3. CONCEPTION FONCTIONNELLE

Avez-vous choisi d'utiliser un outil de représentation graphique ?

Si oui, pourquoi ?

réponse

Si non, pourquoi ?

réponse

2.3.1. DESIGN

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

réponse

2.3.2. ACCESSIBILITÉ

Qu'avez-vous mis en place pour le respect de l'accessibilité du site ?

réponse

3. QUESTIONS GÉNÉRALES

Qu'est-ce qui fait que votre site est éco-conçu ?

réponse

Avez-vous d'autres remarques pertinentes sur votre projet ?

réponse