Green Jiraration

Technique :

Pour le front nous avons choisis du Javascript natif (minifié avec Parcel) compilé à partir de TypeScript :

- Afin d'être le plus léger possible nous avons opté pour un framework css minimaliste (milligram.css faisant 5 Kb).
- Pour économiser du chargement d'image inutiles, nos ressources sont en webp et svg qui sont très légères.
- L'application est une single page application ce qui nous évite d'avoir à recharger de nombreuses fois des documents html.
- Cela nous donne un document unique contenant HTML,CSS,JS de 20.7Kb (GZIP) et ce malgré un coût initial plus important en éléments HTML.

Pour le back nous avons choisis Actix (actix-web, actix-files, actix-web-actors, …) qui est du Rust:

- Après étude des différents framework étant les plus performant, d'après https://www.techempower.com/benchmarks/, nous avons décidé de prendre actix étant donné qu'il est celui ayant le meilleur score.
- Le stockage de nos données se fait sur un PostgreSQL qui est simple et léger.
- Le back-end embarque la version compilée du front-end afin de n'utiliser qu'un seul serveur http.
- Les compressions acceptées sont dans l'ordre de préférence: brotli, gzip, deflate

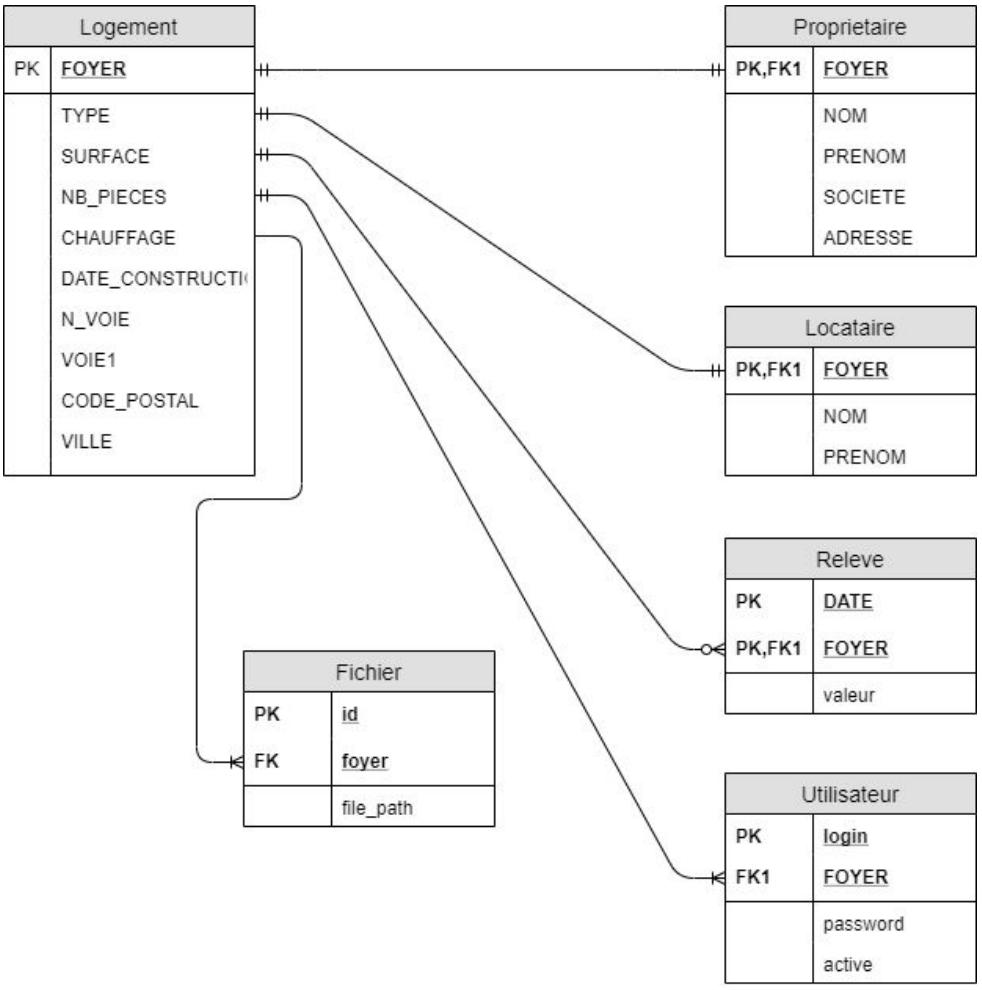
Afin d'optimiser les échanges entre le back et le front toute notre API est basé sur les websocket (sauf envoi de fichier) ce qui nous permet de :

- Créer un tunnel de conversation entre la session d'un utilisateur et le back ce qui évite de refaire les échanges superflus (handshake, header, token d'authentification et autres) à chaque envoi et réception de données.
- Faire un affichage en temps réel car il suffit d'un simple message pour tout mettre à jour.

Nous avons implémenté un Système de cache qui fait en sorte qu'on ne charge et renvoi jamais une donnée qui a déjà été servie à une socket. Elle est stockée dans la ram du back.

Base de données:

Voici le schémas de notre base de donnée :



Design :

Comme indiqué précédemment, nous avons choisis la légèreté, de ce fait le design que nous avons fait est simple mais complet.

Nous n’avons pas eu le temps de faire d’étude empathique sur les différents personas qui peuvent utiliser notre application mais nous avons ajouté l’évolution de consommation des utilisateurs. Ce qui a pour objectif d’encourager, graphiquement, la réduction de cette dernière.

Reste à faire :

Par manque de temps nous n’avons pas eu le temps d’implémenter les notifications quand modification. Actuellement, toutes les 3 secondes, le front demande au back si il y a eu des modifications via le tunnel de socket. Le système de notification nous aurait permis d’inverser cela et de faire que ce soit le back qui informe le front qu’il y a eu une modification. Cela permet d’éviter des requêtes sans réponses.

Sécuriser les échanges avec HTTPS et WSS.