Université de Mons
Faculté des sciences
Département d'Informatique

Energenius

Rapport de projet - prototype

Professeur :
Tom Mens
Assistants :
Pierre Hauweele
Sébastien Bonte

Auteurs :
Godwill Louhou
Gilles Jaunart
Jérémy Delnatte
Louis Dascotte

Année académique 2022-2023

Table des matières

1 Introduction

1.1 Préambule

Ce rapport présentera l'application client/serveur Energenius, ainsi que les fonctionnalités disponibles et implémentées. Il sera également présenté un petit mode d'emploi afin de permettre de lancer l'application localement.

1.2 Répartition des tâches

Pour réaliser ce projet, nous avons décidé de séparer les tâches de la manière suivante :

- **Godwill Louhou** : Développement de l'application client pour les clients.
- **Gilles Jaunart** : Développement de l'application client pour les fournisseurs.
- **Jérémy Delnatte** : Développement de l'application serveur/API.

De par cette séparation des tâches, il a été possible de se concentrer sur un aspect du projet à la fois et par personne, ce qui nous a permis d'améliorer nos compétences dans les domaines choisis.

1.3 Choix techniques

1.3.1 Langage

Pour le développement de l'application serveur, il nous a été imposé d'utiliser Java. Cependant, le choix était plutôt libre vis à vis de l'application client. Après avoir envisagé différentes technologiques, nous avons opté pour JavaScript couplé au framework ReactJs. Nous avons choisi ce framework, car en plus d'être très populaire, il permet de développer des applications web de manière très efficace et rapide. De plus, il permet de développer des applications web de manière modulaire, ce qui permet de séparer les différentes parties de l'application et de les réutiliser facilement. Le fait qu'il soit si populaire a également facilité la recherche de solutions aux divers problèmes rencontrés, ainsi que la recherche de documentation.

1.3.2 Base de données

Pour la base de données, nous avons choisi d'utiliser MongoDB Atlas, qui est une base de données NoSQL hébergée sur le Cloud. Cela nous a donc permis d'éviter de devoir héberger la base de données nous même, et de par l'application MongoDB Compass, il a été aisé de créer les différentes collections et de les gérer.

1.3.3 Outils

Durant le développement de ce projet, certains outils ont été indispensables pour nous permettre de travailler et collaborer de façon efficace. Ainsi, l'utilisation de GitHub nous a permis de travailler facilement sur le même code, et

de pouvoir le partager sans avoir à faire de transferts de fichiers. Pour nous organiser, nous avons également créé un Trello afin de suivre l'avancée dans nos différentes tâches, mais également d'en ajouter pour les autres lorsque nécessaire. Nous avons majoritairement utilisé Discord pour communiquer entre nous. L'outil Postman nous a également été très utile pour tester les différentes routes de l'API, sans avoir à tout implémenter sur l'application client.

1.3.4 Structure des applications

Les deux applications client sont basées sur le même schéma de fonctionnement, avec comme majoritaire différence les pages qui sont disponibles.

2 Application client

2.1 Application client

2.1.1 Application pour les clients

Vidéo de présentation :
Il est possible de retrouver une vidéo de présentation de l'application pour les clients via ce lien : .

Fonctionnalités implémentées :
L'application pour les clients permet la création de portefeuilles, la visualisation des données de consommation, la gestion des contrats ainsi que l'entrée de données de consommation. Le développement de l'interface graphique suit la maquette proposée pendant la phase de modélisation, à quelques différences près, à cause de contraintes techniques ou de changements d'avis. L'utilisateur peut donc se rendre sur le site et se connecter à l'aide du compte qu'il a créé dans la page de création de compte. Une fois son compte créé, il est redirigé vers la page de connexion, où il peut alors se connecter et utiliser l'application.

Avantages et inconvénients
L'application est visuellement agréable et intuitive d'utilisation. Elle a été pensée de sorte à ce que n'importe quel utilisateur puisse l'utiliser sans soucis, peu importe son expérience sur Internet. Elle est très réactive, permettant de limiter le temps d'utilisation au maximum. Le fait d'utiliser React a permis de réutiliser certaines pages pour en créer d'autres, et donc accélérer la vitesse de développement, permettant également d'ajouter de nouvelles pages dans le futur sans trop de soucis. L'utilisation des JWT (JSON Web Tokens) permet de garder une sécurité sur le site, mais également de permettre au client de ne pas avoir à se connecter à chaque changement de page.

Malheureusement, l'application contient quelques défauts, non résolus par manque de temps.

Fonctionnalités manquantes

Il n'a pas été possible d'utiliser le protocole HTTPS pour sécuriser les échanges de données entre l'application client et l'application serveur. Ainsi, les requêtes sont envoyées en clair, ce qui peut poser problème si un attaquant arrive à intercepter les données.

3 Serveur

3.1 API

L'api est séparée en trois parties : la partie Repository, la partie Service et la partie Controller.

- **Repository :** Cette partie permet d'interagir avec la base de donnée MongoDB, et de céer toutes les requêtes pour récuperer les documents dans la DB.
- **Service :** Cette partie sert à tous ce qui est de la logique de l'api. Elle permet aussi la gestion des exceptions et des erreurs en envoyant des exceptions à la couche supérieur.
- **Controller :** Cette couche permet a l'application client d'interagir avec toutes la partie api de l'application. Elle gère aussi la gestion des exceptions envoyer par la couche service.

3.2 Base de données

La base de données ne respecte pas totalement la modélisation de l'ERD éffectuer dans la première partie du projet. Ces changements ont été effectués pour mieux fonctionner avec la base de données qui est du NoSQL.

3.3 Tests unitaires

Des tests unitaires ont été réalisés pour pratiquement toute la couche service de l'application. Malheuresement, par manque de temps certaines fonctions de la couche n'ont pas pu être tester. Pour la couche Repository, nous n'avons pas réussi à faire fonctionner une base de données embarqué pour tester la couche Repository de l'app.

4 JWT

Pour ne pas devoir faire des requêtes d'authentification pour chaque requêtes à l'api, nous avons choisi d'utiliser JWT qui va permet d'envoyer un token lors de la connexion d'un utilisateur. Ce token va être envoyer dans chaque requêtes et va permettre à l'api de vérifier que la requête viens bien de l'utilisateur connecté.

5 Extensions

5.1 Extension 5 - Auto-production d'énergie - Jérémy Delnatte

5.1.1 Introduction

L'extension d'auto-production d'énergie permet de surveiller la production d'électricité et de gérer les certificats verts.

Pour surveiller la production d'électricité, l'utilisateur peut ajouter un point de production d'électricité qui est associé a un compteur numérique. Pour fonctionner le point de production d'électricité doit être accepter par le fournisseur. Le point de production d'énergie fonctionne en toute transparence avec les fonctionnalités des points de fournitures de l'application de base.

Pour les certificats verts, l'utilisateur peut en demander lorsque le seuil de production d'éléectricité à depassser les 1000 kWh, dès lors une requête est envoyé au fournisseur et peut accepter le certificat vert ou non. L'application permet de voir tous les certificats verts demandés et acquis.