# Cahier des charges final

mercredi 21 août 2024 09:22

**Rappel premier échange :**

*Le besoin:*
*Refonte d'une application Web existante : suivi d'avancement de la facturation et logiciel de caisse de type Saas. Actuellement, application sous HTML et PHP, base de données SQL (MariaDb).*

*Application fonctionnelle mais vous cherchez à améliorer l'ergonomie de l'application, l'expérience utilisateur (design plus moderne avec des standards d'interface utilisateur plus récents) et à développer de nouvelles fonctionnalités.*

*Cahier des charges à rédiger mais j'ai déjà noté les points suivants :*
*- l'utilisateur doit pouvoir injecter dans l'application un export du logiciel de banque pour limiter la saisie*
*- l'application doit être utilisable sur un ordinateur mais aussi sur un appareil mobile / tablette*
*- gestion avancée des droits utilisateurs (compte admin, droits lecture et/ou écriture sur certaines ressources, etc.)*

*Architecture envisagée:*
*1- Un client Web sur React*
*2- Une API REST sous Node.js*
*3- Une base de donnée SQL MariaDB (déjà existante, à voir l'état actuel des données: migration sur une nouvelle BDD ?)*

*Technologies proposées:*
*Front-End:*
*React https://react.dev/*
*Librairie de composants Shadcn https://ui.shadcn.com/*

*Back-end:*
*Serveur sous Express: https://expressjs.com/ (librairie Node.js)*
*Module d'authentification avec Passport.js: https://passportjs.org/*
*Interaction API <--> BDD avec Prisma: https://www.prisma.io/*
*Authentification des utilisateurs grâce aux JWT: https://jwt.io/introduction*

*Déploiement et intégration continue :*
*Hébergement: Github avec configuration des alertes en cas de mise à jour de sécurité importantes à faire sur une ou plusieurs librairies Javascript (https://docs.github.com/fr/code-security/dependabot/dependabot-security-updates/configuring-dependabot-security-updates)*

*Conteneurisation des services sous Docker pour isoler :*
*-l'environnement de test à celui de production*
*-le(s) client(s) Web de l'API*

*On n'en a pas parlé et c'est peut-être surdimensionné mais à voir s'il sera nécessaire de mettre en place un outil de monitoring ou à minima un outil de gestion et de stockage des logs sur le serveur Node.js.*

Demandes commerciales :

- Demande 1 - Règlements multiple
  Pouvoir enregistrer des règlements multiples sur une seule facture. (Espèce + chèque ou CB + Chèque ou CB + CB + ESPECE + CHEQUE)
- Demande 2 – Fond de caisse reste enregistré pour le lendemain
  Quand la gestionnaire de caisse commence la journée, il faut pouvoir rappeler la caisse de la veille en validant le contrôle des coupures et pièce avec possibilité de modifier. (Pour cause d'erreur de la veille par exemple)
- Demande 3 – Transfert du jour avec détails de caisse.
  Il faut que le document transfert du jours, indique **le contrôle de la caisse** MOINS **le transfert** égale **le fond de caisse**. (Pour chaque coupure, pour chaque pièce).
- Demande 4 – Interface de supervision
  Il faut avoir un type de compte superviseur permettant de voir l'activité et les montants de chaque caisse.

Demandes comptables :

- Demande 5 – Interface plus simple permettant de modifier n'importe quelle information sur une ligne
- Demande 6 – Importation des virements
  Suivant un template précis, pouvoir importer en masse des virements sous format Excel ou csv.
- Demande 7 – Exporter les espèces
  Pouvoir exporter les paiement en espèces enregistrés par caisse ou par règlements client d'un utilisateur précis et d'une date précise à la fois.