Organisation du code :

Le projet comportera deux applications. L’une d’entre elles sera dédiée au client et comportera donc une interface graphique (HTML, CSS, …). Cette application communiquera à la deuxième application via des requêtes ajax. Cette deuxième application agit comme le backend de notre application, elle recevra des tâches à effectuer venant du client (c’est-à-dire, l’admin système dans le cas de notre projet) utilisant la première application. Ce backend effectuera alors les tâches qui lui a été confié et fera des requêtes sur la base de données postgres si nécessaire. Voici un schéma représentant ces 2 applications :

C:\Users\Corenthin\Downloads\Untitled Diagram (2).png

Organisation des modules Node.js :

C:\Users\Corenthin\Downloads\module.png

C:\Users\Corenthin\Downloads\légende.png

*Italique* = librairie / module externe

Le backend Node.js comportera une organisation précise : l’app.js sera la pièce centrale du backend. Il se chargera d’initier en premier la connexion avec la base de données postgres. Pour cela, il utilisera database.js qui sera un fichier de configuration, il possède la configuration nécessaire pour se connecter à la base de donnée. Le but de ce simple fichier est de permettre de rapidement pouvoir modifier les infos de la base de donnée, si besoin est, et cela sans devoir modifier app.js. Pour permettre une connexion postgres, on utilisera la librairie externe « pg ». En plus de cela, app.js fera appel aussi à des fichiers model pour le json. Par exemple ici, on pourra avoir un model user.js qui définira quelles sont les informations d’un utilisateur.

Enfin, routes.js sera aussi demandé par app.js. Ce fichier contient les différentes méthodes qui seront exécutées en fonction de la requête AJAX envoyée par l’application cliente.