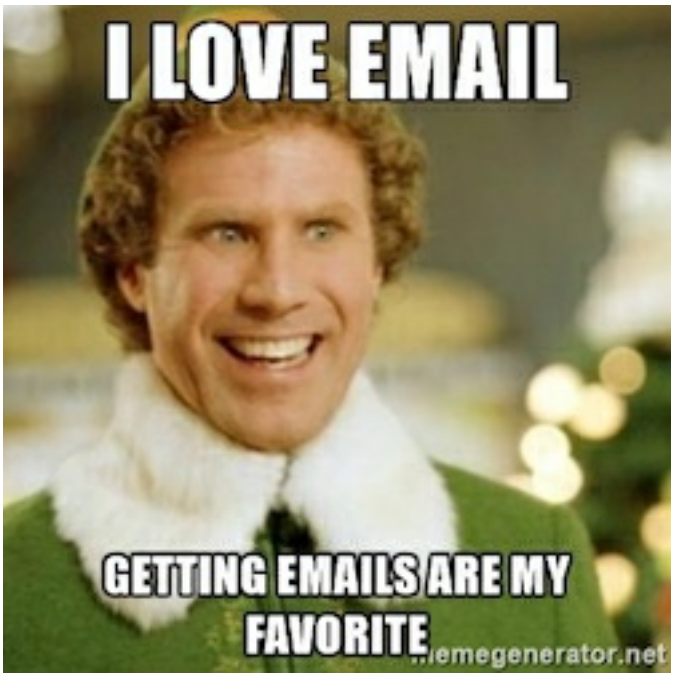
TP 6 : Envoie d'email



Dans le TP de cette semaine, vous allez voir comment envoyer un email à l'utilisateur selon les cas suivants :

- Lors de l'ajout pour lui donner ses informations de login et de mot de passe
- Lors de la modification si le login ou le mot de passe ont changé (juste un email lui disant que ses informations ont changées)
- Après avoir lancé une requête de récupération de mot de passe : Effectivement, vu que les mots de passe sont cryptés en BDD, vous devrez être en mesure de changer uniquement celui-ci via une route prévue à cet effet.
- Rajouter la route `/authent` prenant en `POST` les paramètres login/password et retournant `{ msg : 'ok' }` si les informations sont bonnes et `{ msg : 'ko' }` si les informations sont incorrectes.

Pour réaliser ces emails, vous allez utiliser les modules suivants :

- `nodemailer` : library permetant d'envoyer facilement un ou plusieurs emails au travers de Node.JS
- `mailgen` : Moteur de templating d'email compatible tout client (ou presque).

Attention : Lors de l'édition de l'utilisateur (route `PUT`), si jamais le mot de passe n'est pas saisit ou bien s'il s'agit du mot de passe crypté, veuillez à ne pas le recrypter sous peine de vous retrouver avec un résultat invalide en base de donnée.

Attention : Penser la partie d'envoie d'email (qui devrait être une fonction propre) réutilisable facilement et configurable via ses paramètres d'entrée vu que vous devrez en faire un service hapi indépendant au dernier TP.

Compléments

Lors du TP précédent, certains d'entre vous ont eu du mal à s'en sortir ou comprendre comment certaines choses marchaient :

- Les fonctions de Handler entrantes (celles appelées par les routes) prennent toujours deux paramètres : `request` et `reply`.
- Vous pouvez récupérer la variable `server` au travers de `request.server` pour, par exemple, accéder à un plugin spécifique.
- Pour accéder à un model mongoose, vous pouvez de la manière suivante : `server.database.monmodel`.
- Les models mongoose retourné via la méthode plus haut vous permettent d'utiliser ses fonctions statiques et magiques (find, findAll, findOne, update, insert, ...).
- Si vous voulez créer une instance de votre model, vous devrez faire `let model = new server.database.monmodel()`.

- Au travers de `json-mongoose` toute fonction du model est promisifiée (vous pouvez faire `model.save().then()`, `mommodel.find().then().catch()`, ...)
- Si vous obtenez l'erreur `model monmodel.js is incorrect`, commentez tous les `require` non utilisés, que vous utilisez bien `module.exports` (avec un s) et vérifiez que votre schéma soit correct.
- Niveau organisation du code : Un handler/route pour une partie métier. Par exemple à partir de ce TP vous devriez avoir les routes et handlers suivants :
    - default
    - users
    - mails (uniquement en handler pour le moment)