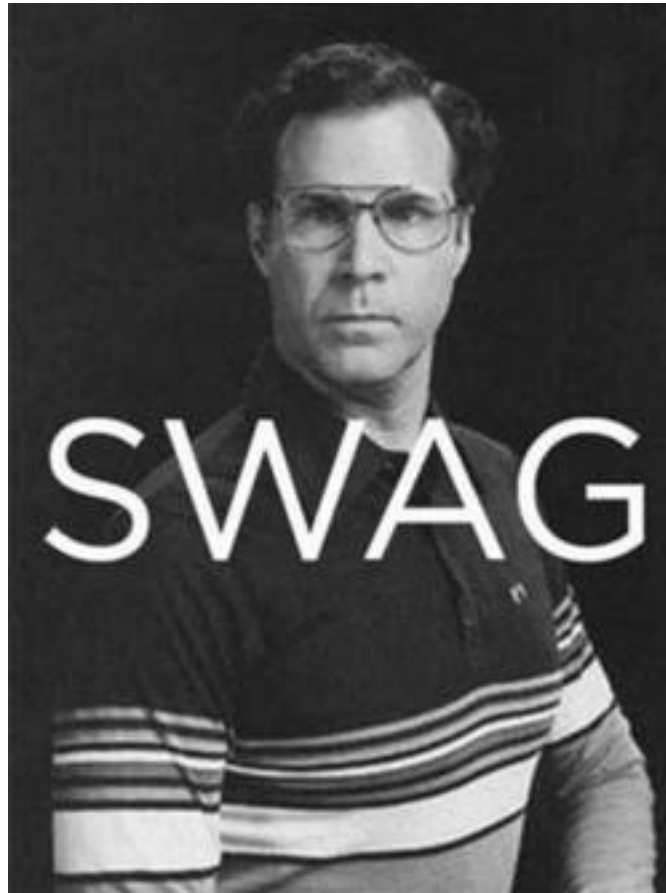


Hapi : Swagger et Joi



Avant de commencer

Avant de commencer ce TP, vous devrez terminer le cours nodeschool `make-me-hapi` (mis à part les deux derniers cours qui sont cookies et authentication) et récupérer dans votre repository git le projet `hapi-boilerplate` servant de base à ce TP et au projet fil rouge. (trouvable à cette adresse : <https://github.com/throrin19/hapi-boilerplate>)

Swagger

Swagger est un outil permettant de générer la documentation REST de votre service avec, pour chaque route, la possibilité de requêter le serveur depuis la documentation avec les champs correspondant aux données attendues.

Vous trouverez des exemples complets de ce que permet Swagger ici :

[Demo live Swagger](#)

Mise en place

Afin de mettre en place Swagger, vous devez savoir qu'un plugin hapi est disponible. Il s'agit du module `hapi-swagger`.

Votre travail est d'intégrer ce module à votre projet Hapi de base (basé sur le boilerplate fournit en tp3) et renseigner l'entrypoint de base dans la documentation swagger.

Joi

Comme vous l'avez vu dans `make-me-hapi`, Joi permet de valider des données par rapport à un schéma. Mais Joi peut aussi facilement contrôler dans Hapi les données entrantes au niveau de vos requêtes, que ce soit dans les params GET ou POST et ce directement au niveau de la configuration de vos routes.

Afin de bien ranger tout ça, nous allons rajouter un répertoire `schemas` dans le répertoire `app` afin de stocker nos différents schémas Joi (un fichier pour un schéma). Ce qui donne l'architecture suivante :

```
Hapi-boilerplate
├── app                                # Partie applicative de l'API
│   ├── endpoints                    # Entrées des traitements par les req
│   │   └── default.js              # Entrées de toutes les requêtes sur
│   ├── handlers                    # Traitement des différentes entrées
│   ├── plugins                     # Plugins internes au projet
│   └── schemas                     # Schémas Joi
```

Mise en place

Comme vous avez vu dans `make-me-hapi`, vous devrez utiliser le module `joi` au sein de votre projet.

Votre travail est de créer un endpoint pour une gestion d'utilisateurs. (Pour le moment juste une création sur la méthode `POST`) en vous basant sur la structure suivante :

```
{
  login : "Mon identifiant",
  password : "Mon mot de passe non crypté",
  email : "mon@adresseemailvalide.fr",
  firstname : "Mon prénom requis",
  lastname : "Mon nom",
  company : "Ma société",
  function : "Ma fonction",
  nir : "numéro de sécurité social valide"
}
```

Vous devrez effectuer les tests de validation suivants :

- login : champ requis
- password : champ requis d'au moins 8 caractères
- email : champ requis et doit être un format d'email valide
- firstname : champ requis
- lastname : champ requis
- nir : champ non requis mais avec un numéro de sécurité sociale valide

Tips : Vous verrez qu'en intégrant sur vos routes des validations Joi, Swagger va automatiquement remplir ses formulaires en fonction, ce qui vous permettra de tester facilement votre route.

Le résultat de cette requête sera l'utilisateur envoyé mais sans les champs suivants :

- password
- nir

Tips : Pour simplifier la suppression des champs, je vous conseille de regarder ce qu'il est possible de faire via le module `lodash` .