William-Arno CLEMENT IUT de Fontainebleau

**Projet Tuteuré**

**Application web de Micro-blogging**

# Présentation

Hapshot est une application de microblogging développée avec NodeJS pour le serveur et MySQL pour la base de données.

La problématique est la suivante : **Comment créer une application de réseautage social, et surtout comment la mettre en production ?**

# Concept

On peut partager sur Hapshot des textes courts accompagnés (ou pas) de photos. Le logiciel est construit en suivant le pattern MVC. Côté design, le logiciel est inspiré de Twitter et Tumblr.

Un utilisateur inscrit au service peut publier un article sur son compte. Ce dernier peut être accompagné ou non d’une image. Il peut également accéder à son fil de messages personnalisé pour voir un journal des publications des personnes suivies dans l’ordre anti-chronologique.

La page profil contient les derniers posts d’un utilisateur ainsi que sa photo, sa biographique, son pseudonyme et son nom en haut de page.

Les pages profils sont accessibles depuis le web à l’adresse suivante : [www.hap.sh/ot/nom\_d\_utilisateur](http://www.hap.sh/ot/nom_d_utilisateur) .

# Choix techniques

Hapshot se veut être une plateforme sociale. Pour cela, elle doit répondre aux codes des réseaux sociaux actuels.

Le choix de NodeJS est pour moi un avantage dans la réalisation de ce projet : il permet un chargement de page rapide : il est notamment utilisé chez de nombreux médias en ligne pour sa rapidité d’exécution et ses performances SEO. NodeJS supporte également les fortes montés en charge : il peut donc gérer un grand nombre de requêtes simultanément. Il est un environnement de développement stable, sécurisé et javascript est un langage que j’aime particulièrement.

Le logiciel sera architecturé à l’aide du standard MVC pour plusieurs raisons :

* La segmentation du code sous forme Modèle/Vue/Contrôleur permet plus de clarté
* Il permet de rendre le code évolutif

La gestion des packages sera performée par npmJS.

Pour le back-end, Express sera utilisé comme serveur. Tandis que la base de données MySQL sera connectée à notre application à l’aide du module officiel NodeJS. Je pense utiliser des modules tels que bodyParser, cookieParserJS, JSWebToken ainsi que HBS pour les vues.

# Objectifs

* Apprendre et comprendre le fonctionnement d’une application sociale
* Maitriser NodeJS et le système de développement avec packages (npm)
* Approfondir mes connaissances sur MySQL
* Créer une interface utilisateur conviviale et une expérience à forte valeur ajoutée

Enfin ce projet sera pour moi l’occasion de travailler sur un projet doué de sens : à la fois expérimentale et sociale.

# L’expérience utilisateur, ou l’interactivité au service du partage

## La charte Graphique

La première étape du développement du projet a été la création d’une charte graphique. Car c’est à partir de cette ligne directrice que j’ai eu l’opportunité de filtrer le public qui s’apprête à utiliser l’application.

L’idée est partie d’un constat simple : à quoi ressemblerait Twitter s’il était pensé pour une audience jeune, par exemple celle de Snapchat ou Tumblr ?

Mon objectif était donc de cibler une tranche de la population jeune, volatile et qui partage plus vite que son ombre chaque moment marquant.

Pour m’accorder au mieux à ce public, il a été tout à fait naturelle de m’inspirer les chartes graphiques les plus populaires sur les magasins d’application. A partir de cela, je me suis appliqué à la création d’une marque claire, efficace : Hapshot. Hap pour « Happy », « Happening » concaténé au mot « shot » pour l’esprit jeune, pour l’idée d’addiction.

L’application sera représentée par un smiley armé de dents de vampire, le tout sur un fond bleu uni.



*Logo de l’application*

Le site web sera accessible à partir de l’url [www.hapshot.com](http://www.hapshot.com), mais j’ai finalement décidé de raccourcir cette url à [www.hap.sh](http://www.hap.sh) afin que le site soit accessible plus rapidement.

Lors de mes recherches graphiques, j’ai effectué quelques itérations pour la création du logo. Celle-ci représente bien ma vision du projet.

## L’interface utilisateur : la page d’accueil

La première page du site internet est pensée pour être minimaliste, et proposer à l’utilisateur la vision du produit.

Une image contenant texte

Description générée automatiquement

*Page d’accueil de l’application web*

Sur la capture d’écran ci-dessus, vous pouvez constater que l’interface fait part belle à l’image d’un concert, qui doit selon moi montrer le potentiel de l’application : partager le moment.

A droite se trouvent les premières ébauches de ce à quoi ressemblera l’interface utilisateur : le logo de l’application, suivie du slogan « Tweet less, do more ».

A partir des boutons en dessous, l’utilisateur va pouvoir se connecter à son compte, ou en créer un nouveau.

# Inscription au service

# Une image contenant texte Description générée automatiquement

*Page de création d’un nouveau compte*

Une fois son compte créé, l’utilisateur aura l’occasion de se connecter à son compte, mais son interface sera inaccessible car son compte n’a pas encore été activé. Une image contenant texte

Description générée automatiquement

*Compte non activé, un mail d’activation a été envoyé au mail fourni*

Une image contenant texte

Description générée automatiquement

*Le mail d’activation envoyé au mail fourni*

Un mail de confirmation sera alors envoyé au mail fourni par l’utilisateur afin d’activer son compte, et lui permettre d’accéder aux services.

# Les services et fonctionnalités

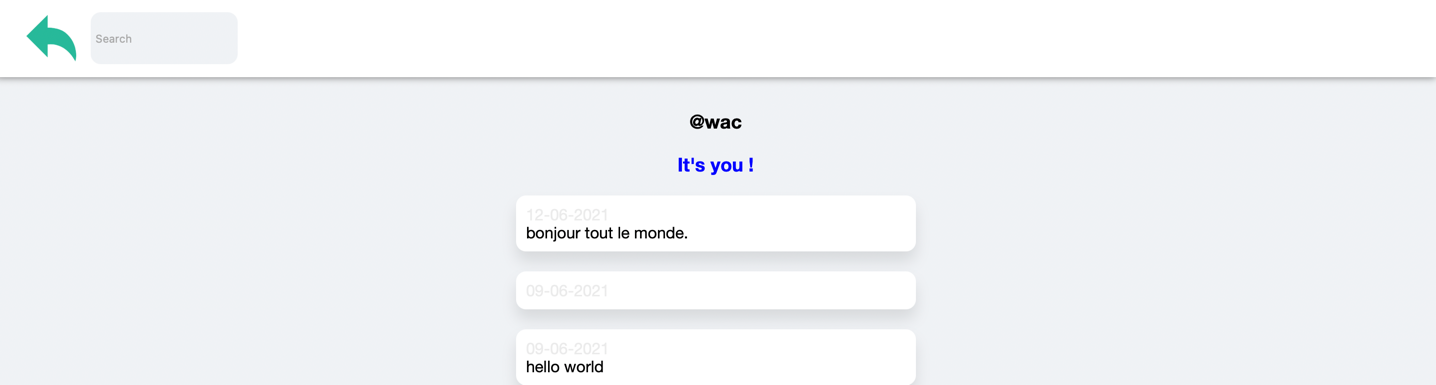
Une image contenant texte

Description générée automatiquement

*La page d’accueil sur Hap.sh après connexion au service*

Une fois connecté à son compte, l’utilisateur va pouvoir entrer dans le vif du sujet avec le fil d’actualité. Y seront disposés dans l’ordre anti-chronologique les différents posts des « hapshooters » suivis par l’utilisateur.

L’utilisateur pourra interagir de deux manières avec un post : lui laisser une mention « j’aime » ou bien accéder au profil de son émetteur.



*La première version du profil (en tant qu’utilisateur consulté)*

Pour accéder au profil, il y a plusieurs possibilités : la première est de cliquer sur le nom de l’utilisateur à consulter. La deuxième est la consultation de l’utilisateur de son propre profil à l’adresse [www.hap.sh/ot/](http://www.hap.sh/ot/)

Dans le cas où le profil consulté n’est pas celui de l’utilisateur, un bouton « Suivre/Follow » va apparaitre à côté du nom d’utilisateur.

Une image contenant texte

Description générée automatiquement

*La première version du profil (en tant qu’autre utilisateur)*

# Le développement

Après avoir imaginé l’architecture du logiciel en avril 2021, je me suis appliqué au développement du projet.

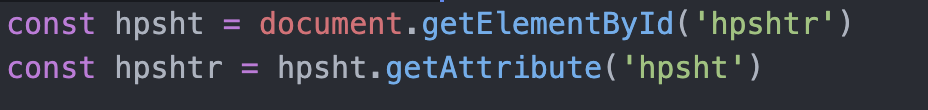
L’idée est la suivante : rendre le projet le plus modulaire possible afin de pouvoir développer une API et peut-être plus tard greffer une application mobile au projet.

Le développement est toujours en cours à l’heure où j’écris ces lignes. Néanmoins, plusieurs obstacles se sont dressés sur ma route durant le développement du projet.

## Comment passer une variable HBS à un fichier Javascript du client

L’un des problèmes sur lequel je suis tombé est le transfert de variables depuis handlerbar.js vers un fichier javascript client.

Je pensais au début passer par une méthode native à javascript, mais je n’ai jamais trouvé de solution propre au problème. C’est pourquoi j’ai trouvé une alternative en me servant d’une div cachée permettant de servir de fichier temporaire entre les deux partis.



*On récupère depuis le document HTML l’attribut dans la div caché*

Une image contenant texte

Description générée automatiquement

## HBS, le moteur de vues

Handlebar.js est un moteur de vue pour serveur express. J’avais le choix entre ce dernier et un autre moteur de templates : Pug. J’ai néanmoins choisi HBS car je me sentais vraiment à l’aise avec dans mes projets personnels, et étant adepte des balises de styles au sein du HTML, il m’a semblé nécessaire d’utiliser un outil permettant la manipulation de code HTML directement dans la vue.

## MySQL, ma base de données

J’ai longtemps hésité entre MySQL et Moongoose pour la gestion de la base de données. Mon choix s’est finalement porté sur MySQL car le fournisseur que j’utilise pour l’hébergement NodejS, à savoir OVH, propose une base de données MySQL et l’outil de gestion PHPMYADMIN.

## Nodemon, l’outil efficient

Afin de gagner du temps lors de mes différentes itérations, j’ai utilisé un outil NodeJS nommé Nodemon. Ce framework permet un gain de temps considérable dans le développement du serveur.

A chaque modification de fichier, le serveur redémarre automatiquement, et ainsi, je peux m’abstenir de relancer le serveur Node manuellement.

J’ai gagné en efficacité et en efficience grâce à cet outil, et j’ai donc pu me concentrer sur le produit plutôt que sur le côté administration.

## Nodemailer : le mailer simple d’utilisation

Afin d’automatiser l’envoie de mail de confirmation/validation de compte, j’ai utilisé une dépendance au Mailer Nodemailer. Ce dernier possède une approche assez simple, et il suffit d’entrer l’identifiant, le mot de passe ainsi que le serveur SMTP du webmail que l’on veut utiliser pour permettre l’envoie de mails automatiques.

Par la suite, il suffisait de générer un entier unique pour valider le compte. Dans mon code, j’utilise les routes « verify » et « send » pour respectivement vérifier si le lien est correct, et pour envoyer le mail auprès du courrier électronique fourni par l’utilisateur venant de s’inscrire.

# Documentation en vue de la maintenance logicielle

Le projet est architecturé selon les normes du modèle MVC.

Une image contenant texte

Description générée automatiquement

Tout d’abord, commençons par le dossier « db » : il contient juste un module d’export permettant de se connecter à la base de données de manière unique au travers du framework MySQL de NodeJS.

Nous avons également le dossier « controllers » contenant à la fois des fonctions utilitaires comme par exemple getProfileFromUsername(), qui permet de récupérer les informations du profil d’un utilisateur depuis son nom d’utilisateur ; mais aussi les modules d’exports dédiés à l’authentification utilisateur ( login et register).

Puis, intéressons-nous au dossier public : il s’agit d’un dossier qui a été inscrit au format static dans notre serveur express. L’idée est de pouvoir accéder à son contenu directement par son url.

Par exemple, si nous sommes dans la vue (que je vais évoquer quelques lignes en dessous), je pourrais invoquer une image à partir de son url relative au dossier public, à savoir /img/logo.png.

Ce dossier public est donc un fourre-tout dans lequel se retrouveront images, feuilles de styles CSS ainsi que fichier javascript destinés au client.

Nous avons ensuite le dossier routes, dans lequel se retrouveront les fichiers gestionnaires des routes. Ce sont ces fichiers qui vont déterminés quelles url mènent à quel contrôleur. Mais ce n’est pas tout : c’est ici que vont être gérés les services liés aux API pour permettre la récupération de données au format JSON à partir d’une url décrivant notre demande.

Par exemple, pour récupérer la liste des posts d’un utilisateur, il suffit de faire un appel à l’url hap.sh/api/posts/nom-d-utilisateur. C’est d’ailleurs de cette manière que le client va chercher les posts à afficher sur la page profil.

Enfin, le dossier views est le dossier qui va stocker les vues pour l’affichage au client. Les vues sont au format hbs. De plus, vous remarquerez qu’il s’y trouve un sous dossier « partials » : il s’agit en fait du répertoire dédiés aux morceaux de vues réutilisables telles quel le header, le footer…