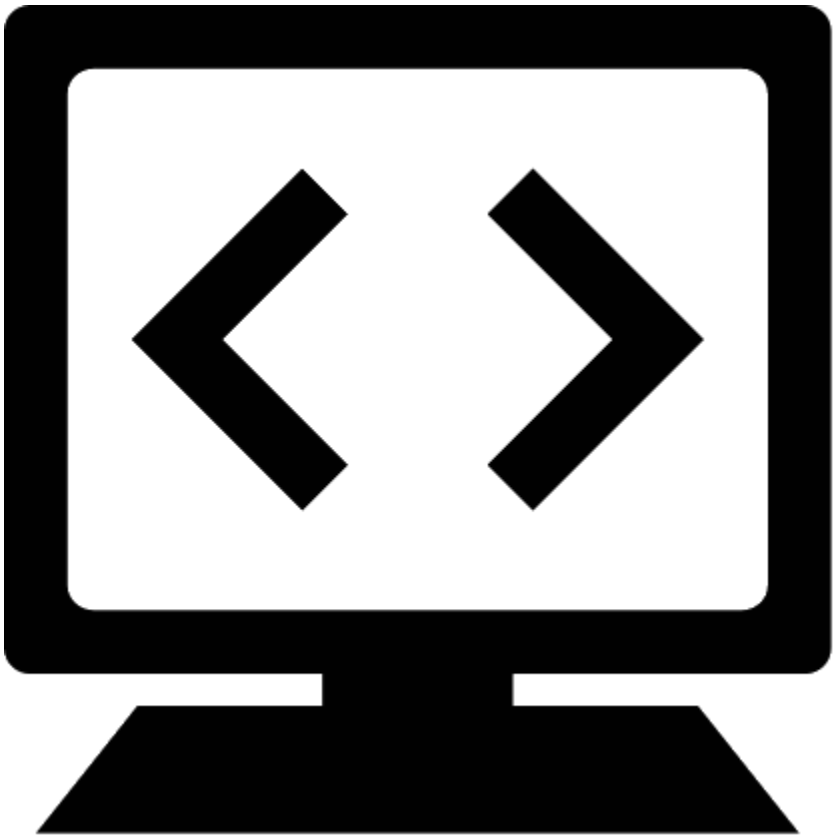
Dossier de synthèse

Projet:

MyThread



FRINOT Xavier

Table des matières

Introduction 4
Qui suis-je ? 4
Liste des compétences du référentiel 5
Présentation du projet 6
Le contexte 7
Mon histoire 7
Mon Parcours de formation 7
Mes Objectifs 7
Choix de développement 8
Langage de développement 8
Outils supplémentaires 8
Environnement supplémentaire 8
Open source 8
Conditions de Travail 11
Architecture, ergonomie et maquettage 12
Ergonomie et maquettage 12
Architecture du code 13
Node Js 14
Modèle OSI 14
Serveur Node Js 15
Synchrone / Asynchrone 16
Cahier des charges 18
Base de données 19
MCD (Modèle Conceptuel de données) 19
MLD (Modèle logique de données) 20
Express.JS 21
Connexion à la base de données 22
La partie Modèle (MVC) 23
La partie Controller (MVC) 24
La partie Routeur (MVC) 25
La partie Vue (MVC) 26
Sécurité des formulaires 28
Sécurité et Protection des données (RGPD) 29

Ajout de futures fonctionnalités et améliorations 30
Futures fonctionnalités 30
Améliorations 30
Difficultés rencontrées 31
Conclusion 32

Introduction

Qui suis-je ?

Je m'appelle Xavier FRINOT je suis actuellement en formation de développeur web à ELAN Formation pour un titre de qualification de niveau III.

Passionné d'informatique je me suis mis à développer tout seul chez moi avant de rejoindre cette formation.

Suite à mes diverses expériences professionnelles dans différents domaines j'ai pu acquérir de l'autonomie, de la rigueur ainsi que l'envie de travailler dans ce que j'aimais faire.

Mon expérience en tant que développeur mon permis d'approfondir ma méthodologie d'apprentissage pour connaître de nouveaux langages de développement.

Liste des compétences du référentiel

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

Présentation du projet

Avant toutes choses pour mieux comprendre la suite de ce dossier je vais vous résumer en quoi consiste mon projet de fin d'année.

MyThread

MyThread est un réseau social comportant uniquement du texte où l'utilisateur peux raconter des anecdotes qui lui sont arrivés durant la suite du dossier nous appellerons cela un "Thread", chaque utilisateur a son propre profil où il peut ajouter une photo de profil (seul photo possible sur le réseau). Chaque Thread est composé de manière suivante :

- Le profil de la personne qui a posté le Thread
- Un bouton "Follow" / "Unfollow"
- Un Titre
- Un Texte de maximum 255 caractères
- La date de la publication
- Un hashtag (#)
- Un bouton J'aime

Les autres utilisateurs peuvent interagir avec les Threads de tous les autres utilisateurs en les aimant grâce à la fonctionnalité "Like". Ils peuvent aussi suivre la personne qui poste le "Thread" pour ensuite avoir toutes les nouvelles et anciennes publications dans leurs fil d'actualité.

La catégorie "Hashtag" (#) dans la publication d'un thread quant à elle sert de thème à cette publication.

Prenons un exemple:

Le contexte

Mon histoire

Mon Parcours de formation

A l'entrée de la formation nous avons commencé par apprendre les langages HTML 5 et CSS 3 ensuite nous sommes directement parti dans la direction du langage de programmation PHP en faisant de petits exercices de bas niveaux. Par la suite nous avons appris ce qu'était, à conceptualiser et à modéliser une base de données. Nous avons ensuite appris la méthode de programmation orienté objet puis fais différents exercices afin d'approfondir notre connaissance dans la programmation informatique. Nous avons aussi un peu vu le langage JavaScript pour comprendre à quoi servait ce langage pour pouvoir faire des requêtes Ajax (nous en parlerons plus loin).

Nous sommes ensuite allés apprendre à quoi servait et à utiliser un Framework. Pour cela nous avons travaillé avec le Framework PHP Symfony.

Mes Objectifs

Pour mes objectifs à la sortie de la formation j'espère pouvoir intégrer une entreprise en tant que développeur JavaScript Full-Stack. Un développeur Full-Stack est un développeur qui s'occupe de la partie front-end d'un site ainsi que de la partie back-end.

Choix de développement

Langage de développement

Je vais vous lister maintenant les différents outils et langages de développements que j'ai utilisés pour la réalisation de mon projet.

Mon projet est réalisé uniquement en JavaScript en utilisant la plateforme logicielle libre Node.js, le moteur de Template EJS et le Framework ExpressJS. (Embedded JavaScript)

J'ai aussi utilisé le langage CSS3 pour la mise en page des différentes informations reçu par EJS

Un Framework, appelé en français cadres d'applications est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel. (Architecture)

Outils supplémentaires

Pour les outils supplémentaires que j'ai utilisé nous avons :

- Visual Studio Code pour l'IDE (Environnement de développement)
- Google Chrome pour le navigateur internet
- SQlite / SQliteStudio pour toute la partie base de données.
- JMerise pour le MCD
- Le Framework Bootstrap pour faciliter la mise en page et le responsive
- JQuery, une bibliothèque JavaScript facilitant l'utilisation de requêtes Ajax
- Font Awesome, une bibliothèque d'icones

Environnement supplémentaire

Pour l'organisation de mon Travail (Diagramme de classes, maquettage de l'application), pour cela j'ai utilisé ArgoUML et draw.io

Pour l'organisation entière et éviter de se perdre ou de perdre tout mon projet j'ai utilisé Git et Github. Git qui est un gestionnaire de visionning et Github qui est un serveur de stockage de code utilisant Git.

Open source

La désignation open source, ou « code source ouvert », s'applique aux logiciels (et s'étend maintenant aux œuvres de l'esprit) dont la licence respecte des critères précisément établis par l'Open Source Initiative, c'est-à-dire les possibilités de libre redistribution, d'accès au code source et de création de travaux dérivés. Mis à la disposition du grand public, ce code source est généralement le résultat d'une collaboration entre programmeurs.

Tous les outils que j'ai utilisé pour ce projet sont open sources.

Node.JS. Utilisation pour comme plateforme de serveur Web.

SQLite. Utilisation pour moteur de base de données relationnelle.

Bootstrap. Utilisation pour la création du design.

Visual Studio Code (IDE). Utilisation pour éditer le code.

Express.JS. Utilisation pour construire des applications web basées sur Node.js. C'est un FrameWork JavaScript.

JQuery. Utilisation pour faciliter l'écriture de scripts côté client, surtout pour les requêtes AJAX. C'est une bibliothèque JavaScript.

HTML 5. Utilisation pour afficher du Front. C'est un langage de balisage.

JavaScript. Langage principal de l'application utilisé pour tout.

CSS 3. Utilisation pour la mise en page.

Embedded JavaScript. Utilisation pour permettre la manipulation de variables données par le routeur.

Conditions de Travail

J'ai choisi d'utiliser JavaScript pour mon projet car c'est un langage qui permet de tout faire en écrivant le code une seul fois (Application Bureau, site web et applications Mobile). En utilisant JavaScript cela me permet d'implémenter Node.js plus facilement.

Node.js est une plateforme logicielle libre et événementielle en JavaScript orientée vers les applications réseau qui doivent pouvoir monter en charge.

Elle utilise la machine virtuelle V8 et implémente sous licence MIT les spécifications CommonJS.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Il est donc possible de se passer de serveurs web tels que Nginx ou Apache lors du déploiement de sites et d'applications web développés avec Node.js.

Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur.

Node.js est utilisé notamment comme plateforme de serveur Web, elle est utilisée par Groupon, Vivaldi, SAP, LinkedIn,6, Microsoft, Yahoo!, Walmart, Rakuten, Sage et PayPal et bien d'autres encore.

Architecture, ergonomie et maquettage

Ergonomie et maquettage

Avant de commencer la programmation de mon projet, il fallait que je sache vers où j'allais alors j'ai débuté par réfléchir à quoi ressemblerait le projet et quelles seront ces fonctionnalités. Dès que je savais ce que je voulais je me suis mis au travail.

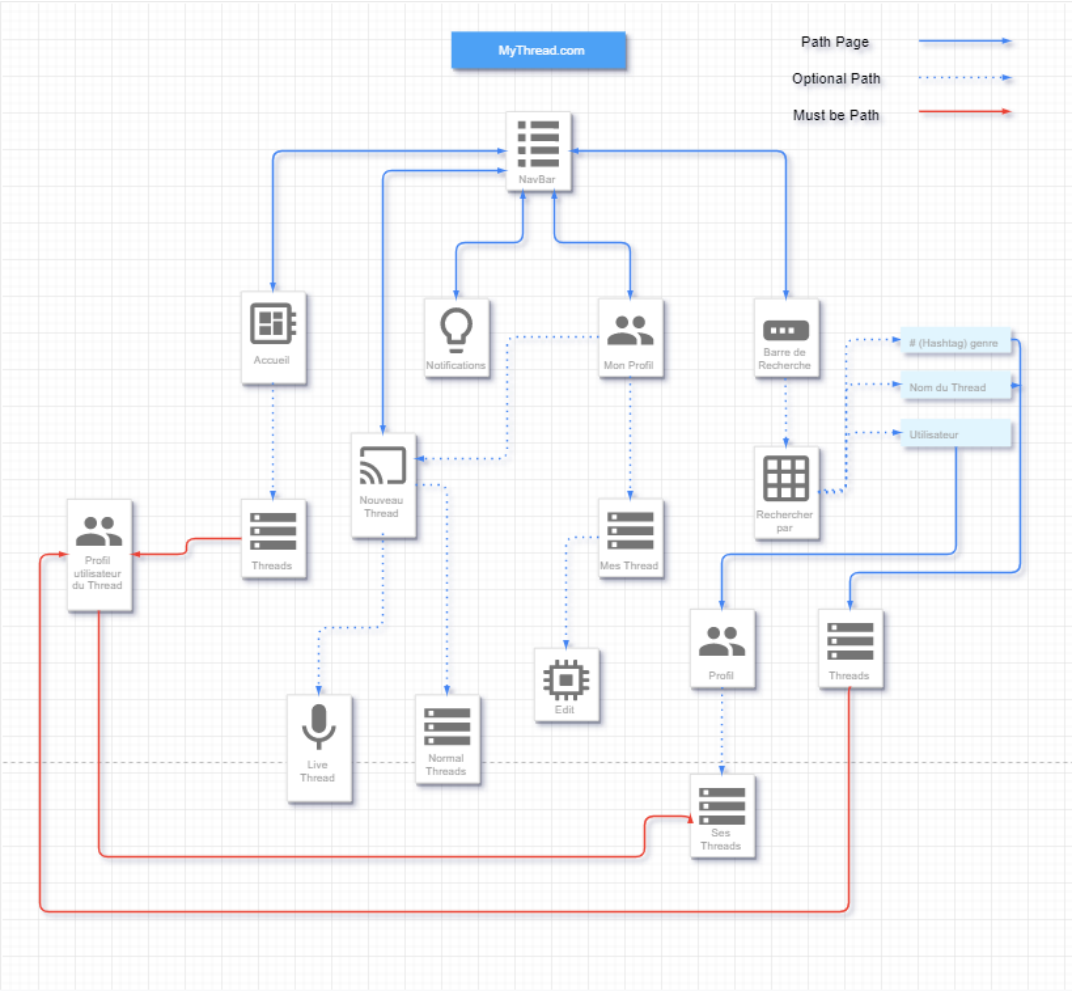
J'ai commencé par réaliser un cahier des charges pour énumérer toutes les tâches que j'avais à faire.

J'ai décidé de commencer par faire un maquettage de mon application et de son ergonomie pour éviter d'avoir à refaire l'architecture du projet plus tard. J'ai ensuite décider de faire un diagramme de classe UML pour savoir quels étaientt mes besoins dans le MCD.

MCD est une abréviation qui signifie modèle conceptuel de données, il s'agit d'une représentation logique de l'organisation des informations et de leurs relations.

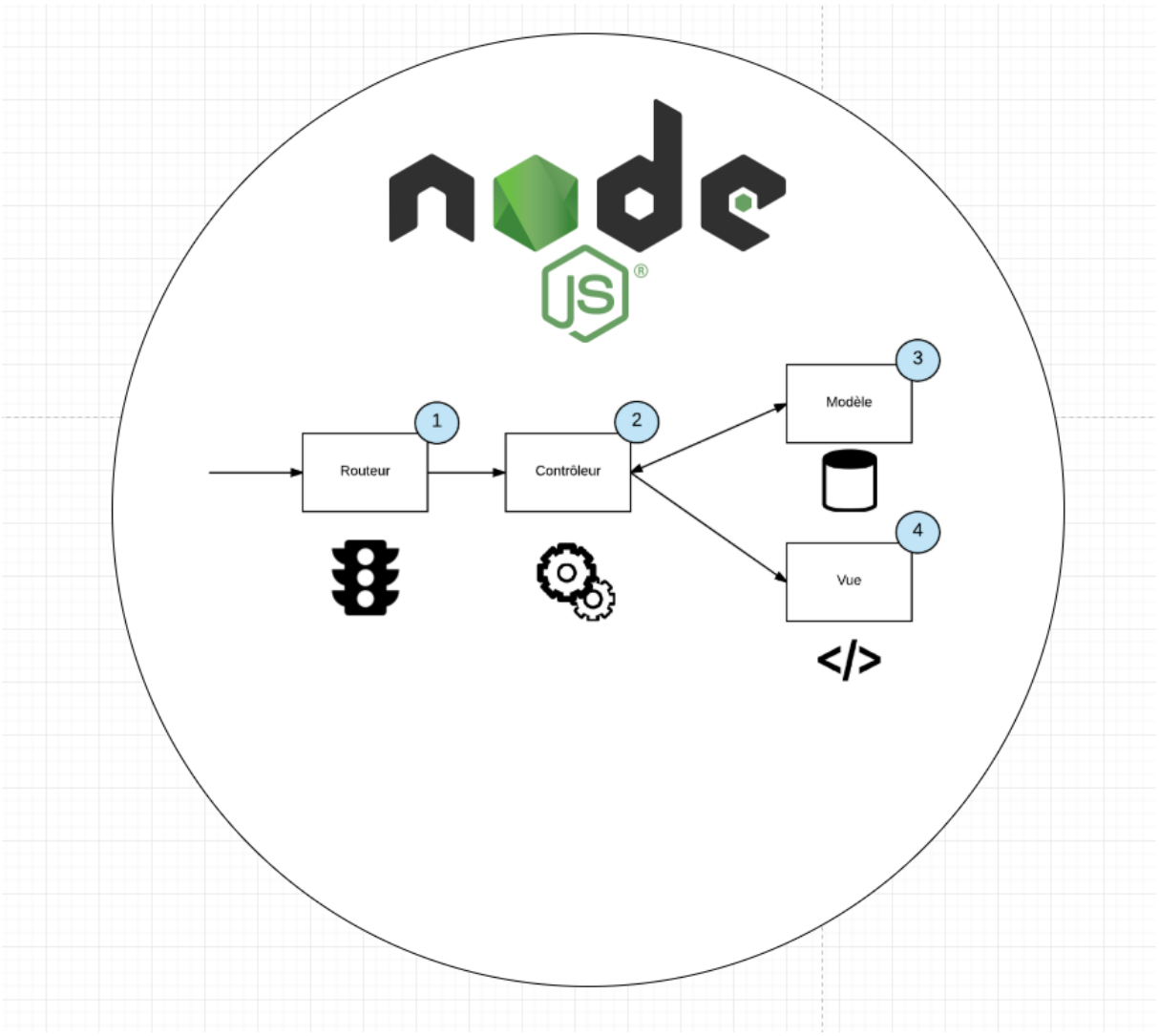
Je me suis ensuite attaquer à la base de données et au MCD. Seulement une fois que tout cela était terminé je pouvais commencer à initialiser mon projet.

Voici un schéma démontrant comment j'ai pensé l'ergonomie de mon application avant de commencer à coder.



Architecture du code

Pour l'architecture de mon code j'ai décidé d'utiliser la Design Pattern MVC, je vous explique cela avec un schéma.



Nous avons Node.js qui englobe notre application, la requête avance vers le routeur, passe par le contrôleur, cherche les données dans le partie modèle, retourne dans le contrôleur et renvoie la vue.

Node Js

Modèle OSI

Node.js est une plateforme logicielle libre et événementielle en JavaScript orientée vers les applications réseau qui doivent pouvoir monter en charge.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Il est donc possible de se passer de serveurs web tels que Nginx ou Apache lors du déploiement de sites et d'applications web développés avec Node.js.

Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur, Node.js est utilisé notamment comme plateforme de serveur Web.

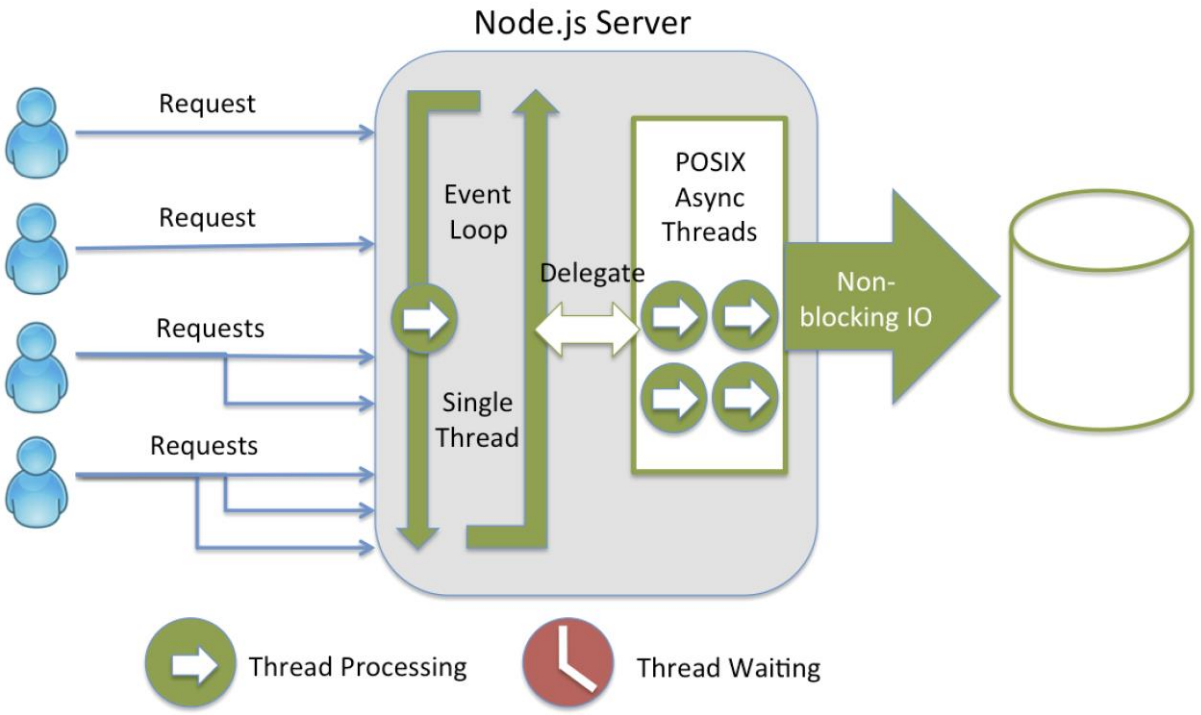
Le modèle OSI est une norme de communication, en réseau, de tous les systèmes informatiques.

	PDU	Couche		Fonction
Couches hautes	Donnée	7	Application	Point d'accès aux services réseau
		6	Présentation	Gère le chiffrement et le déchiffrement des données, convertit les données machine en données exploitables par n'importe quelle autre machine
		5	Session	Communication Interhost, gère les sessions entre les différentes applications
	Segment / Datagramme	4	Transport	Connexion de bout en bout, connectabilité et contrôle de flux ; notion de port (TCP et UDP)
Couches matérielles	Paquet	3	Réseau	Détermine le parcours des données et l'adressage logique (**adresse IP**)
	Trame	2	Liaison	Adressage physique (**adresse MAC**)
	Bit	1	Physique	Transmission des signaux sous forme numérique ou analogique via un câble, une onde...

Sur le modèle OSI Node.Js se trouve sur la couche numéro 4.

Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.

Serveur Node Js



Ceci est un schéma pour expliquer son aspect asynchrone, soit une des raisons pourquoi j'ai utilisé Node js pour mon projet.

Nous pouvons constater, sur le schéma, que nous avons plusieurs requêtes simultanément, nous avons aussi une boucle d'évènement qui tourne en continue et qui va intercepter toutes ces requêtes qui délègue tout ça aux POSIX Threads qui va permettre de les exécuter en même temps et de renvoyer le premier qui a une réponse.

Synchrone / Asynchrone

Serveur Apache (PHP)

```php
$Array = [];
$String = "Hello";
$i = 0;
For($i=0; $i<10; $i++) {
        array_push($Array, $String);
}
Var_dump($Array);
//output
array(10) {
[0]=> string(5) "Hello" [1]=> string(5) "Hello" [2]=> string(5)
"Hello" [3]=> string(5) "Hello" [4]=> string(5) "Hello" [5]=>
string(5) "Hello" [6]=> string(5) "Hello" [7]=> string(5)
"Hello" [8]=> string(5) "Hello" [9]=> string(5) "Hello"
}
```

Serveur Node (JavaScript)

```javascript
let Array = []
let String = "Hello"
for(let i = 0; i < 10; i++) {
   Array.push(String)
}
console.log(Array)
//output
Array = []
```

Comme nous pouvons le constater sur l'extrait de code ci-dessus le var_dump() (Debug en php) nous renvoi bien un tableau rempli, tandis que le code écrit en JavaScript nous renvoi un tableau vide car il a d'abord renvoyé une réponse du console.log(Array) avant d'avoir terminé la boucle for().

L'Asynchrone est donc indirectement plus rapide mais peut créer quelques difficultés dont nous parlerons plus tard dans le dossier.

Les promesses

```
deleteaccount: (id_user) => {
    const dao = new AppDAOfs('./db/mythread.db')
    const UserDAO = new UserDAOfs(dao)
    return new Promise((resolve, reject) => {
        UserDAO.deleteaccount(id_user)
            .then(() => {
                dao.db.close(() => {
                    console.log("BDD CLOSE !")
                })
                resolve('ITS OKAAAAAY')
            })
    })
}
```

Quand nous allons faire une requête basique à la base de données nous allons avoir le problème suivant :

- La réponse nous renvoie un objet vide

Pour régler ce problème nous avons la solution de la promesse écrite dans l'extrait de code ci-dessus

Nous créons une promesse, tout le code qui va être exécuté à l'intérieur de cette promesse deviendra synchrone et attendra qu'elle soit résolue avant d'exécuter la suite.

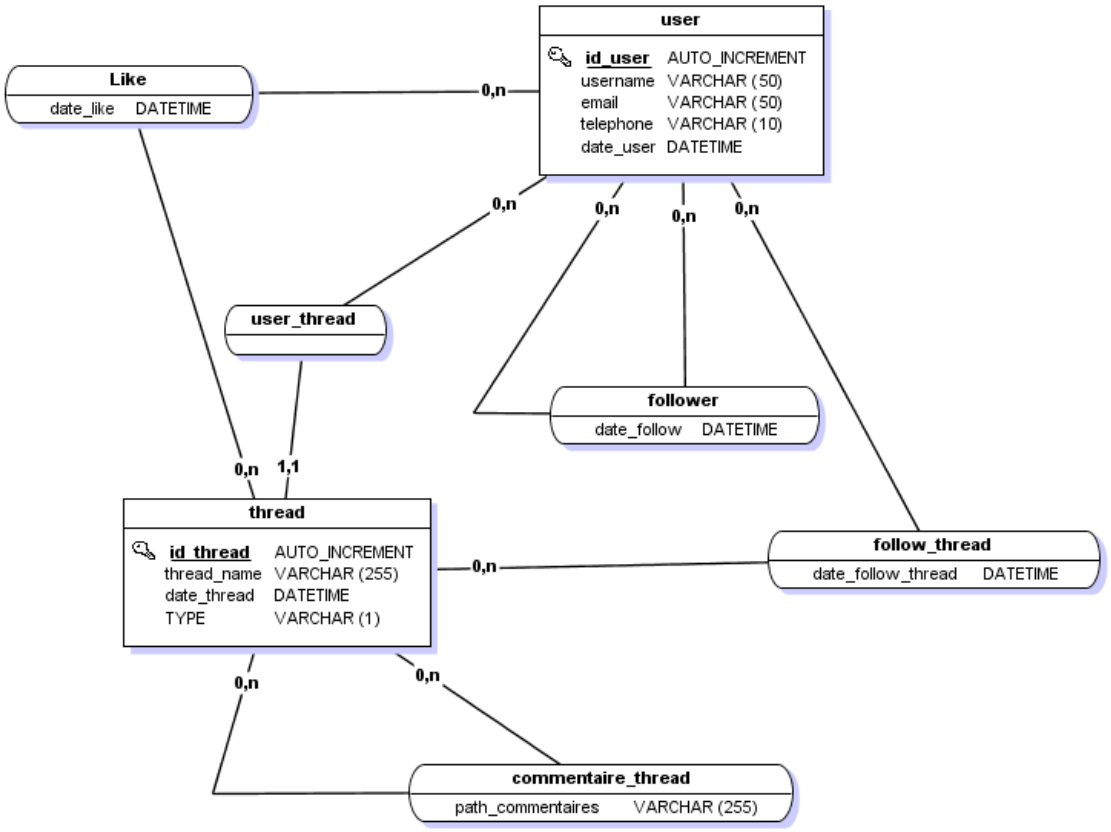
Cahier des charges

Voici mon cahier des charges

- Maquettage de l'application
- Ergonomie de l'application
- Diagramme de classe UML
- Modélisation du MDC
- Migration du MDC vers SQliteStudio en SQlite
- Initialisation du projet
- DAO (Connection a la base de données)
- La classe User (Utilisateur)
- Inscription, hashing du mot de passe
- Connexion, comparatif de mot de passe
- Session de l'utilisateur
- Droit d'accès aux pages si l'utilisateur est connecté
- CSRF Protection
- Message d'erreur
- Barre de navigation
- Algorithme page d'accueil
- Affichage Thread Page d'accueil
- Responsive Design (page d'accueil)
- Front et Design
- Page Profil de l'utilisateur connecté
- Publication d'un Thread
- Page profil des utilisateurs non connectés
- Affichage des Threads sur les pages profils
- Modification d'un Thread
- Fonctionnalité d'affichage d'un Thread aléatoire
- Bouton J'aime
- Fonctionnalité Hashtag (#)
- Fonctionnalité Follow / Unfollow
- Sécurité des formulaires
- Messages d'erreur pour la publication d'un Thread
- Nombre de Follow / Follower
- Affichage des Followers / Following
- Page "Découvrir"
- Fonctionnalité des notifications
- Respect de la loi RGPD

Base de données

MCD (Modèle Conceptuel de données)



La table User et la table Thread sont les tables principales de mon projet, elles sont reliées par toutes les autres relations.

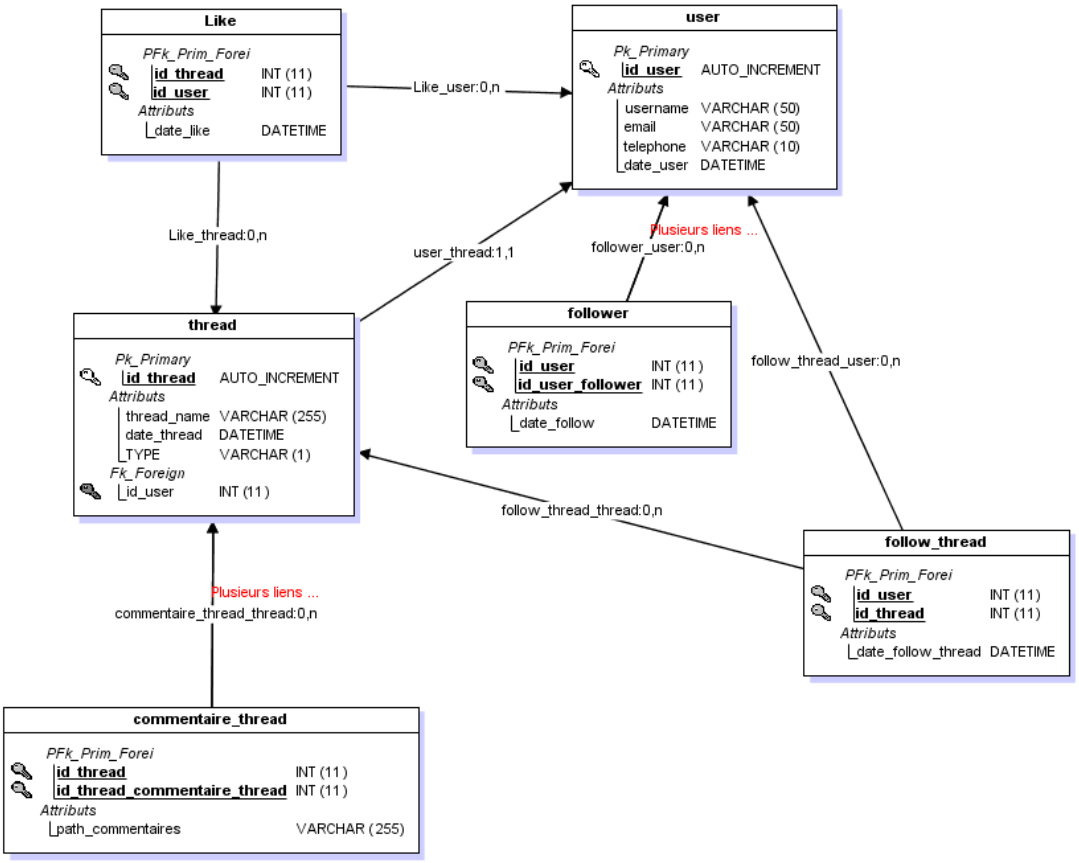
La relation Follower et la relation commentaire_thread sont des relations récursives.

Toutes les autres relations sont des relations normales.

Chaque utilisateur peut "Liker", "Follow" et écrire plusieurs Threads comme indiqué dans chaque début de relation avec les notations o, n.

Pour la table Thread c'est la même chose hormis le fait qu'un thread ne peut être écrit que par un seul utilisateur.

MLD (Modèle logique de données)



Voici le Modèle logique de données, comme expliqué auparavant, nous voyons ce que donne les relations récursives. Elles deviennent des tables avec deux clefs étrangères nécessitant les mêmes informations, les informations de l'ID de table user.

Les deux tables participiales sont dotées chacune d'une clef primaire en auto incrémentation. C'est-à-dire que dès qu'un tuple (ligne d'une table) est rempli, cette clef, représenté sous formes d'ID se remplira tout seul en nombre entier.

Express.JS

Pour mon projet j'ai utilisé le Framework (Cadre de travail) Exress .JS. Express.js est un Framework pour construire des applications web basées sur Node.js. C'est de fait le Framework standard pour le développement de serveur en Node.js.

Voici la ligne de commande à écrire pour installer ce Framework.

```
$ npm install express-generator -g
```

Puis pour choisir notre moteur de Template.

```
$ express --view=pug myapp
```

Le Framework va nous initialiser un serveur http sur le port 3000, il va aussi nous crée le routeur et un contrôleur.

Connexion à la base de données

Connexion à la base de données

```
class AppDAO {
  constructor(dbFilePath) {
    this.db = new sqlite3.Database(dbFilePath, (err) => {
      if (err) {
        console.log('Could not connect to database', err)
      } else {
        console.log('Connected to database')
      }
    })
  }
```

Ici nous avons l'extrait de code à la connexion à la base de données grâce au npm sqlite3. N'oublions pas que sur un serveur Node après chaque ouverture de connexion à la base de données nous devons la refermer aussitôt la requête terminée.

```
all(sql, params = []) {
  return new Promise((resolve, reject) => {
    this.db.all(sql, params, (err, rows) => {
      if (err) {
        console.log('Error running sql: ' + sql)
        console.log(err)
        reject(err)
      } else {
        resolve(rows)
      }
    })
  })
}
```

Le code ci-dessus est une fonction qui prend en paramètre une requête SQL vas nous permettre d'accéder aux données de celle-ci dans la base de données.

La partie Modèle (MVC)

```
getUserByUsername(username, id_user_connect) {
    const User = new UserClass()
    return new Promise((resolve, reject) => {
        this.dao.all("SELECT id_user, username, email, telephone, date_user,
            .then((data) => {
                data.forEach(element => {···
                });
            })
    })
}
```

Pour la partie Modèle, ceci est un exemple avec la classe "UserDAO", est une classe qui va nous permettre d'accéder aux données concernant l'utilisateur.

Nous avons une fonction qui prend en paramètre le pseudo de l'utilisateur et l'id de l'utilisateur connecté, ensuite nous créons un objet User puis nous créons une promesse qui à l'intérieur va exécuter une connexion à la base de données puis exécutera la fonction de la page précédente pour récupérer les données.

La partie Controller (MVC)

Pour la partie contrôleur je vous ai mis un extrait de code.

```
getUserByUsername: (username, id_user_connect) => {
    const dao = new AppDAOfs('./db/mythread.db')
    const UserDAO = new UserDAOfs(dao)
    return new Promise((resolve, reject) => {
        UserDAO.getUserByUsername(username, id_user_connect)
            .then((data) => {
                dao.db.close(() => {
                    console.log("BDD CLOSE !")
                })
                resolve(data)
            })
    })
},
```

La fonction getUserByUsername() qui prend en paramètre le pseudo de l'utilisateur et celui de l'utilisateur connecté comme pour la partie modèle, cependant c'est lui qui va créer l'objet dao (Data Access Object).

Nous initialisons deux objets, le premier 'dao" est un objet de la classe "AppDAOfs" pour la connexion à la base de données qui prend en paramètre le chemin vers la base de données, le deuxième est un objet de la classe 'UserDAO", classe de la partie modèle (voir page précédente).

Nous créons une promesse qui va exécuter la fonction "getUserByUsername()" de l'objet "UserDAO" (classe de la page précédente).

Une fois que tout cela a été exécuté et que nous avons récupéré les données dont nous avons besoin nous pouvons fermer l'ouverture à la base de données que nous avons fait.

La partie Routeur (MVC)

Dans le Design Pattern que j'ai vous ai expliqué auparavant nous avons un routeur, c'est-à-dire que dès que nous recevons une requête http celle-ci vas être renvoyer vers le routeur qui se trouve dans le dossier route généré par Express.JS.

```
//PROFILE
router.get('/profile', isLoggedIn, (req, res, next) => {
  var messages = req.flash('errorThread'); //Display the error message in passport.js
  const getUser = () => {
```

Dans l'extrait de ce code nous recevons une requête http qui est écrite comme cela :

http://mythread.com/thread/random

Alors le routeur va automatiquement exécuter le code qui se trouve à l'intérieur

Une fois que les contrôleurs ont faits leur travail, alors le routeur nous retourne une vue avec comme paramètres les différentes variables utiles que nous allons afficher dans la vue comme

```
ThreadControl.getThreadCountByIdUser(req.session.passport.user)
  .then((threadcount) => {
    Threads.threadcount = threadcount
    res.render('pages/profile', { messages: messages, User: User, threads: Threads, csurfToken:
  })
```

ceci.

Une fois cela terminé nous pouvons passer au code qui nous permet d'afficher la vue.

La partie Vue (MVC)

```
<% for(var i = 0; i < threads.length; i++) { %>
<div class="thread" ondblclick="Like(icon<%- x -%>,
```

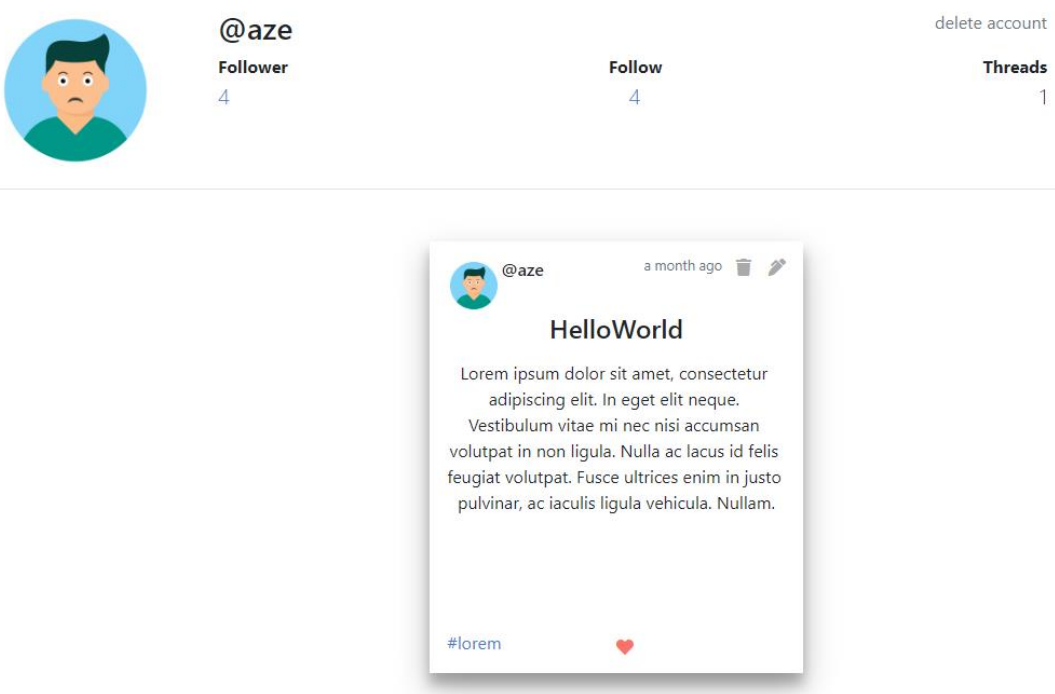
Une fois que nous avons reçu la variable envoyé depuis le routeur dans la vue, en l'occurrence, là nous avons un tableau, alors pour l'afficher nous allons boucler l'affichage EJS dans une boucle for que nous allons terminer un peu plus loin.

```
<i class="fas fa-pen-alt edit" onclick="EditThread(<%- threads[i].id_thread -%>)"···
<i class="fas fa-trash edit"···
<small class="text-muted date-thread"><%= threads[i].date_thread %></small>
<h6>@<%= User.username %></h6>
<h4 class="text-center thread-name"><%= threads[i].thread_name %></h4>
<p class="wordwrap text-center"><%= threads[i].text %></p>
<a href="/thread?hashtag=<%- escape(threads[i].hashtag) -%>"···
<% if (threads[i].liked === false) { %>
<i onclick="Like(icon<%- x -%>, <%- threads[i].id_thread -%>)" id="icon<%- x -%>"
    class="bottom-heart fas fa-heart font-awesome"></i>
```

```
            <% } %>
        </div>
        <% x++  %>
        <% } %>
    </div>
```

Ouvrir une balise EJS (<%= %>) nous permet d'écrire du JavaScript une des raisons pourquoi j'ai choisi de prendre ce moteur de Template plutôt qu'un autre.

Au final tout ce code du Design pattern nous permet d'avoir ceci (voir page suivante)



Sur cette page, qui est la page profil de l'utilisateur connecté, nous avons recueilli les informations suivantes :

- Toutes les données liées à l'utilisateur
    - Son nom d'utilisateur
- Les Followers de l'utilisateur
- Les Follows de l'utilisateur
- Toutes les données liées aux Threads qu'il a écrit
    - Le Titre
    - L'Hashtag
    - La date de publication
    - Le contenue du Thread

Sur cette page nous pouvons noter que l'utilisateur connecté peut à tout moment modifier ses threads, les supprimer ou bien il peut même supprimer son compte comme demandé par la loi RGPD.

Sécurité des formulaires

Express-Validator

Pour la sécurité de mes formulaires j'ai utilisé le module "express-validator" que j'ai initialisé aux niveaux des routes.

```
const { check, oneOf, validationResult } = require('express-
validator/check');
```

```
check('title', 'Title must have Alphanumeric character and contain
no more than 20 characters').isAlphanumeric().isLength({ max: 20 }),
  check('thread_content', 'Thread Content must contain between 0 ans
255 characters').isLength({ min: 1, max: 255 }),
  check('hashtag', 'Hashtag must begin with "#" and contain
Alphanumeric character and contain no more than 25
characters').matches(/^[#][a-zA-Z0-9.()-]/).isLength({ max: 25 })
```

Comme nous pouvons constater sur l'extrait de code ci-dessus, le module va vérifier si le titre est en alphanumériques et contient un maximum de 20 caractères. Si le contenu du Thread contient entre 1 et 255 caractères et pour l'hashtag j'ai utilisé une Regex (expression régulière) pour obliger l'utilisation du caractères "#" en première position.

CSRF Protection

Pour chaque formulaire j'ai utilisé le système du CSRF Protection (Cross-site request forgery), cette protection sert à contrer les attaques d'un utilisateur authentifié une requête HTTP falsifiée qui pointe sur une action interne au site.

Anglais

Pour mes recherches j'allais sur les sites où la communauté était vraiment très active comme par exemple le site de StackOverflow, OpenClassRoom…, ou bien j'allais directement regarder la documentation de la technologie que j'utilisais comme par exemple :

https://www.npmjs.com/package/sqlite3

La documentation en anglais

Main

new sqlite3.Database(filename, [mode], [callback])

Returns a new Database object and automatically opens the database. There is no separate method to open the database.

filename: Valid values are filenames, ":memory:" for an anonymous in-memory database and an empty string for an anonymous disk-based database. Anonymous databases are not persisted and when closing the database handle, their contents are lost.

mode (optional): One or more of sqlite3.OPEN_READONLY, sqlite3.OPEN_READWRITE and sqlite3.OPEN_CREATE. The default value is OPEN_READWRITE | OPEN_CREATE.

callback (optional): If provided, this function will be called when the database was opened successfully or when an error occurred.

The first argument is an error object. When it is null, opening succeeded.

If no callback is provided and an error occurred, an error event with the error object as the only parameter will be emitted on the database object.

If opening succeeded, an open event with no parameters is emitted, regardless of whether a callback was provided or not.

Ma traduction en français

Principal

"new sqlite3.Database(filename, [mode], [callback])"

Retourne un nouvelle Objet de base de donnée et ouvre celle-ci automatiquement. Il n'existe pas d'autre méthode pour l'ouverture de la base de donnée.

nom du fichier : "nom de fichier valide", ":memory:" pour une base de donnée anonyme en mémoire et une chaîne de caractères vide pour une base de donnée anonyme basé sur le disque.

Une base de donnée anonyme n'est pas persistant et quand on la ferme tout le contenue est perdu.

mode (optionnel) : Une ou plus (d'option) comme sqlite3.OPEN_READONLY, sqlite3.OPEN_READWRITE et sqlite3.OPEN_CREATE. La valeur par défaut est OPEN_READWRITE | OPEN_CREATE.

(callback) Rapelle (optionnel) : si il ya une condition, cette fonction vas être appeler seulement si la base de donnée a bien été ouverte ou qu'une erreur a été trouver.

Le Premier argument est un objet d'erreur. si il est nul alors la base de donnée a été ouverte avec succès.

Si il n y a pas de rappelle (callback) et que le code contient une erreur alors , un événement sera passer en paramètre comme objet d'erreur et ce sera le seul paramètre émis à la base de donnée.

Si la base de donné s'est ouverte avec succès, un événement d'ouverture sans paramètres est emit, sans regarder si un appel a été fourni ou pas.

Sécurité et Protection des données (RGPD)

Règlement général sur la protection des données

Voici les différentes obligations de loi RGPD :

- Information
- Demande d'effacement et droit à l'oubli
- Droit à la portabilité
- Protection et sécurité

Ajout de futures fonctionnalités et améliorations

Futures fonctionnalités

Les fonctionnalités supplémentaires que j'aimerai importer dans mon projet sont les suivantes :

- Pouvoir faire des Live Thread, c'est-à-dire que nous allons écrire un Thread sur plusieurs jours voir plusieurs mois comme par exemple : Une histoire d'amour, un projet en cours, ...
- Pouvoir Follow ces Live Thread pour avoir des notifications dès que la personne poste la suite de son Thread
- Pouvoir commenter les Threads de tout le monde
- Améliorer l'algorithme de la page découvrir
- Faire des suggestions d'utilisateur ou de Thread à suivre suivant ce que l'on à aimé
- Crée une API pour pouvoir exporter ses Thread sur Twitter ou sur son propre site
- Mettre des comptes certifiés
- L'ajout de photos ou images sur sa publication

Améliorations

Pour l'amélioration globale de mon projet je pourrai utiliser la technologie MERN (Mongo, Express, React, Node).

L'utilisation de MongoDB, qui est un système de gestion de base de données orientée documents serai meilleur après une requête à la base de données il nous renvoi un objet JSON (JavaScript Notation). Ce qui facilite amplement le code entièrement et l'affichage des objets que nous apportons dans la vue grâce à React qui est une bibliothèque JavaScript libre développée par Facebook.

Au début de mon projet je voulai effectivement utiliser ces différentes technologies, mais après réflexion je me suis rendu compte que je n'avais pas le temps d'apprendre l'utilisation de React et de MongoDB sachant que je devais déjà apprendre à utiliser Node.js et le langage JavaScript en lui-même. J'ai donc utilisé une technologie un peu moins lourde au niveau apprentissage mais qui a une certaine limite et qui n'est pas optimal pour une application.

Difficultés rencontrées

Au niveau des difficultés rencontrés je pense pouvoir dire que l'utilisation de Node.JS est extrêmement compliquée.

L'une des difficultés que j'ai eu à affronter était le langage JavaScript en lui-même suite aux différents changement de syntaxes tout aux long de sa durée de vie.

J'ai aussi rencontrés quelques difficultés à comprendre Node.js, son utilisation, pourquoi le choisir, qu'est ce qui fais sa force.

Mais la plus grosse difficulté que j'ai eu à affronter est la connexion à la base de données en passant par SQLite et en sortir des données, pour cause, l'asynchrone.

Je ne comprenais pas le système de CallBack qui se trouve dans le langage Javascript, j'ai donc régler ce problème en utilisant des promesses, ce que je trouve, personnellement plus simple à comprendre, je pouvais aussi utiliser le système de Async/Await qui fonctionne de la même manière que le système des promesses mais avec un code plus propre, je ne l'ai pas utilisé souvent durant mon projet car j'ai appris son existence plus tard après le début de mon projet.

Conclusion

En conclusion ce projet m'a apporté énormément de connaissance en JavaScript, que ce soit pour son utilisation pour le Back-End ou pour le Front-End. Il m'a aussi permis de faire un projet que j'ai choisis, grâce à quoi j'ai pu garder la motivation d'apprentissage dans le domaine de JavaScript. Je vous remercie d'avoir lu ce dossier et j'espère que vous avez pris plaisir à le lire.