

# Étude et mise en place d'une plateforme web facilitant la gestion de projets entre mandants et équipes de développeurs indépendants

Thibaud Alt – 21 octobre 2021

# Contexte et objectifs

*Plateforme web de communication directe entre mandants et équipes de développeurs*

# Contexte



Problèmes rencontrés par les **mandants**

⚠ Dépassement des coûts, non-respect des délais, diminution des fonctionnalité, etc.

Problèmes rencontrés par les **développeurs**

⚠ Cahier des charges, problèmes technologiques, manque de temps, etc.

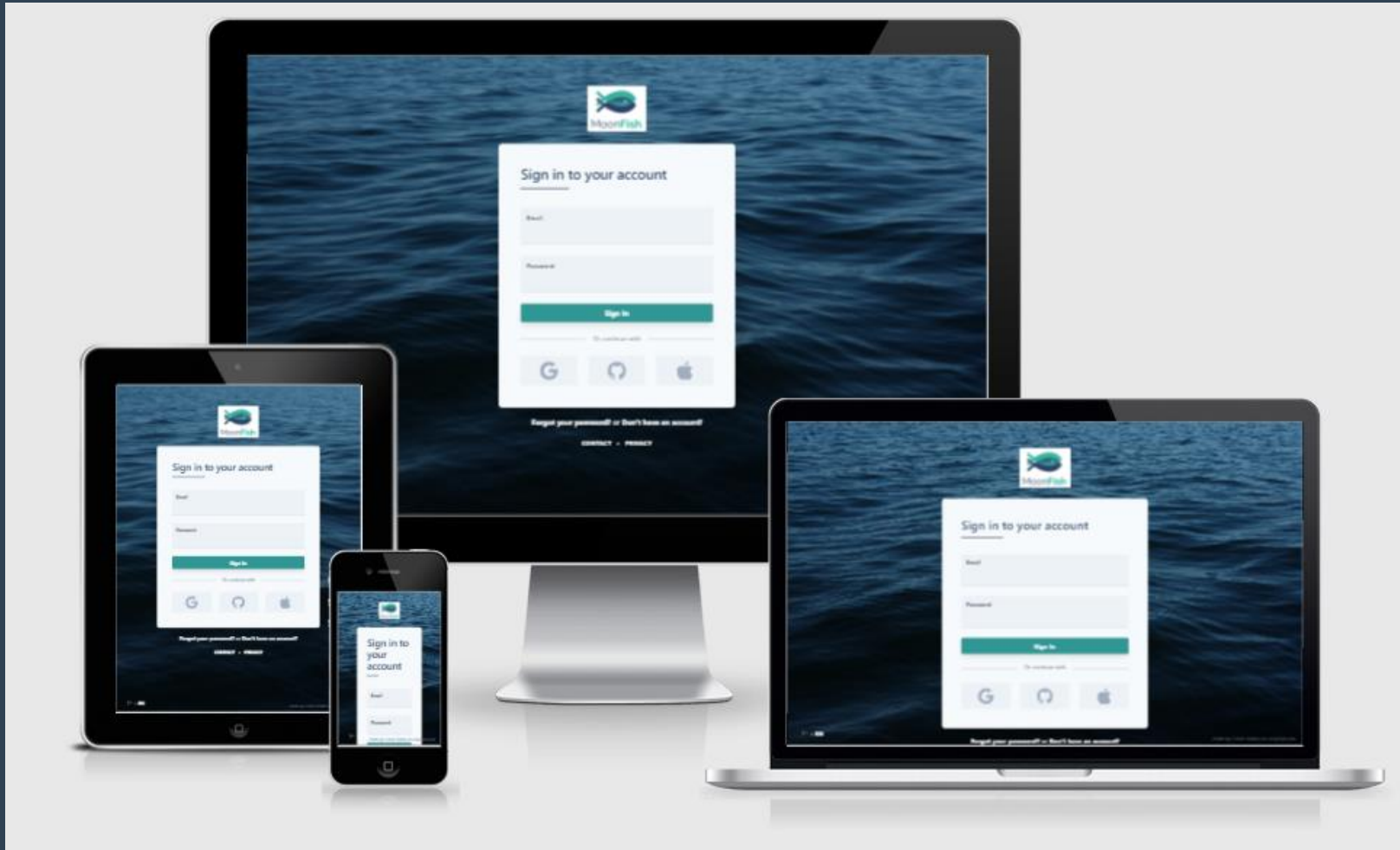
# Objectifs



- ✓ **Communication** directe
- ✓ Compétition sociale
- ✓ Choix des coéquipiers (équipes), des technologies, des prix

... Mise à disposition d'un espace de travail, de professionnels externes (coach, médiateur, consultant, etc.) d'assistance, de ressources, de traductions automatiques, de gestion des versions de sous traitance

# Plateforme web



# Back-end

*Mise en place d'une API REST avec Express.js*

# Persistance des données



# Persistance des données

## MariaDB

*Données n'apportant pas d'information nécessaire aux relations*

- Project

id	27
uuid	95361000-0235-4697-a365-67111de8ce41
createdAt	20/10/2021 12:22
updatedAt	20/10/2021 12:22
projectId	27
lang	en
title	Project eum
description	The Lander is the trademarked name of several series of Nagasaki sport bikes, that started with the ABC800J



## Neo4j

*Données qui apportent des informations pertinentes aux relations*

- Project

```
{  "createdAt": "2021-10-20T12:22:29.278000000Z",  "deadline": "2021-09-26T15:08:36.793000000Z",  "uuid": "95361000-0235-4697-a365-67111de8ce41",  "tags": [ "Claire", "Fantom" ],  "status": 6}
```

uuid

 95361000-0235-4697-a365-67111de8ce41 





# Mapping *objet-relationnel* et *objet-graphe*







SELECT, UPDATE, DELETE...   Data table



MATCH, CREATE, SET...   Data



 Request  JSON Object

Request   JSON Object

express

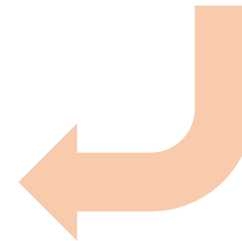
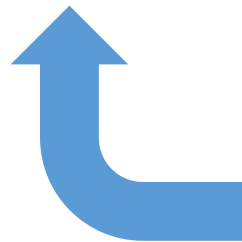
# API REST

express

```
// Example : get all resources linked to a project

router.get(                                // 1. La méthode http (get, post, put, etc.)
  '/projects/:uuid/ressources',           // 2. Le endpoint ou chemin d'accès, il s'agit de l'URL à interroger
  requireAuth,                             // 3. Si nécessaire, l'authentification vérifiée par le middleware
  requiredRole(ROLE_USER),                 // 4. Si nécessaire, le ou les rôles requis par accéder à la requête
  trimRequest.all,                         // 5. Une fonction de « nettoyage » de données reçues
  validateGetProject,                      // 6. Si nécessaire, une fonction de validation des données reçues
  getProjectTeams                          // 7. La fonction à exécuter en cas de validation des étapes précédentes
)
```

1. POST /login
2. GET /projects/uuid/ressources



Header / Status  
JSON Object or error

# Problèmes rencontrés

- Mise en place de la structure (*contollers, middlewares, models*) plus longue que prévue
- *Neode* pas complet, nécessitant d'écrire plusieurs requêtes « à la main »

# Bilan

MariaDB	Neo4j	Sequelize	Neode	Express
✓	✓	✓	✗	✓

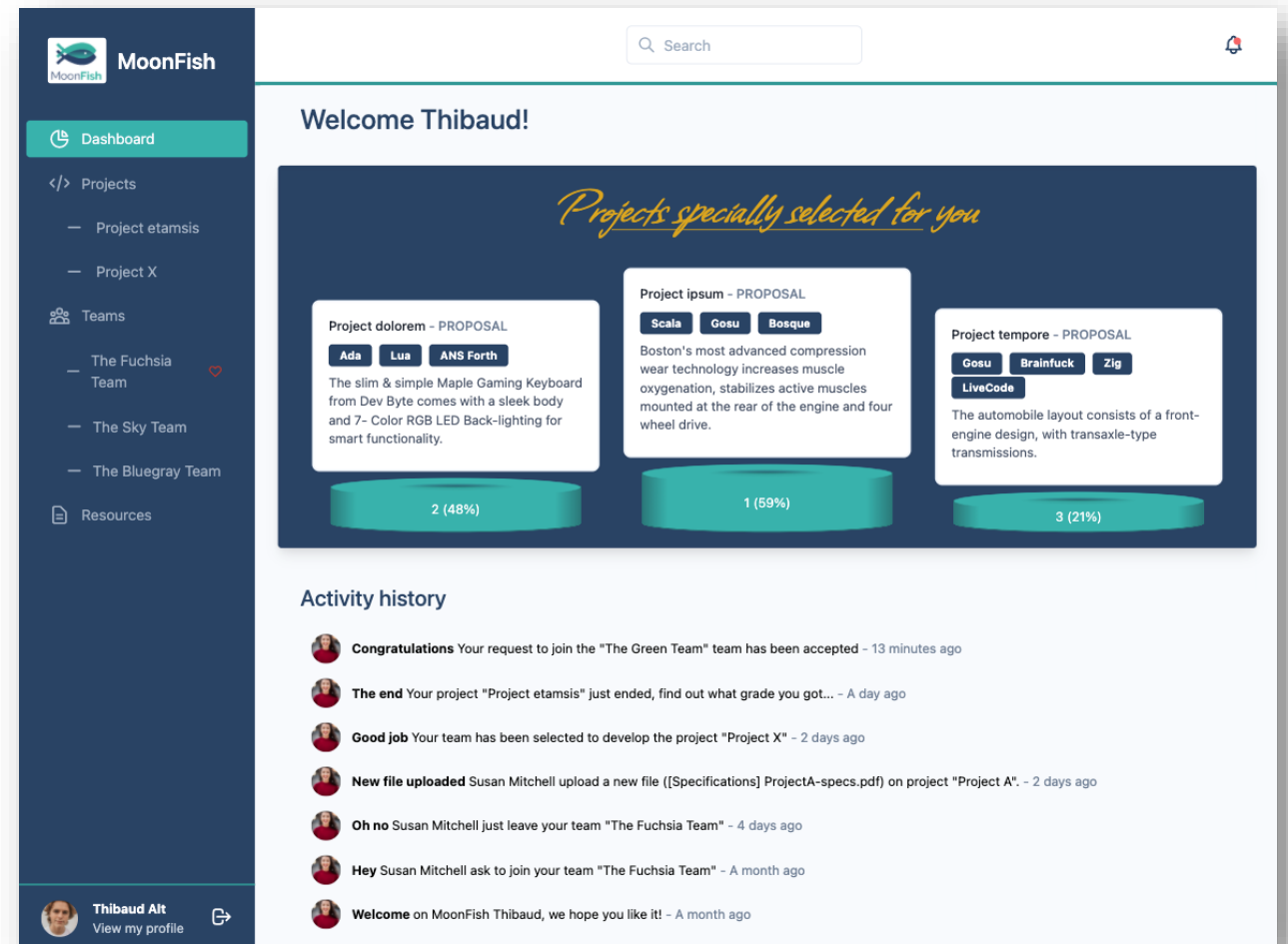
- Express est un *framework* minimaliste mais puissant, facile à prendre en main et pérenne
- Le couple MariaDB / Neo4j fonctionne très bien pour la persistance des données de ce projet
- Il est faudrait trouver une alternative à *Neode*, soit un OGM plus complet soit en écrivant les requêtes *Cypher* directement

# Front-end

*Développement de l'application cliente avec TailwindCSS et Vue.js*

# TailwindCSS

- La prise en main est rapide et super intuitive
- Les interfaces sont propres et complète
- Pas eu besoin d'écrire une seule ligne de CSS !
- L'intégration avec *Vue.js* est transparente



# TailwindCSS

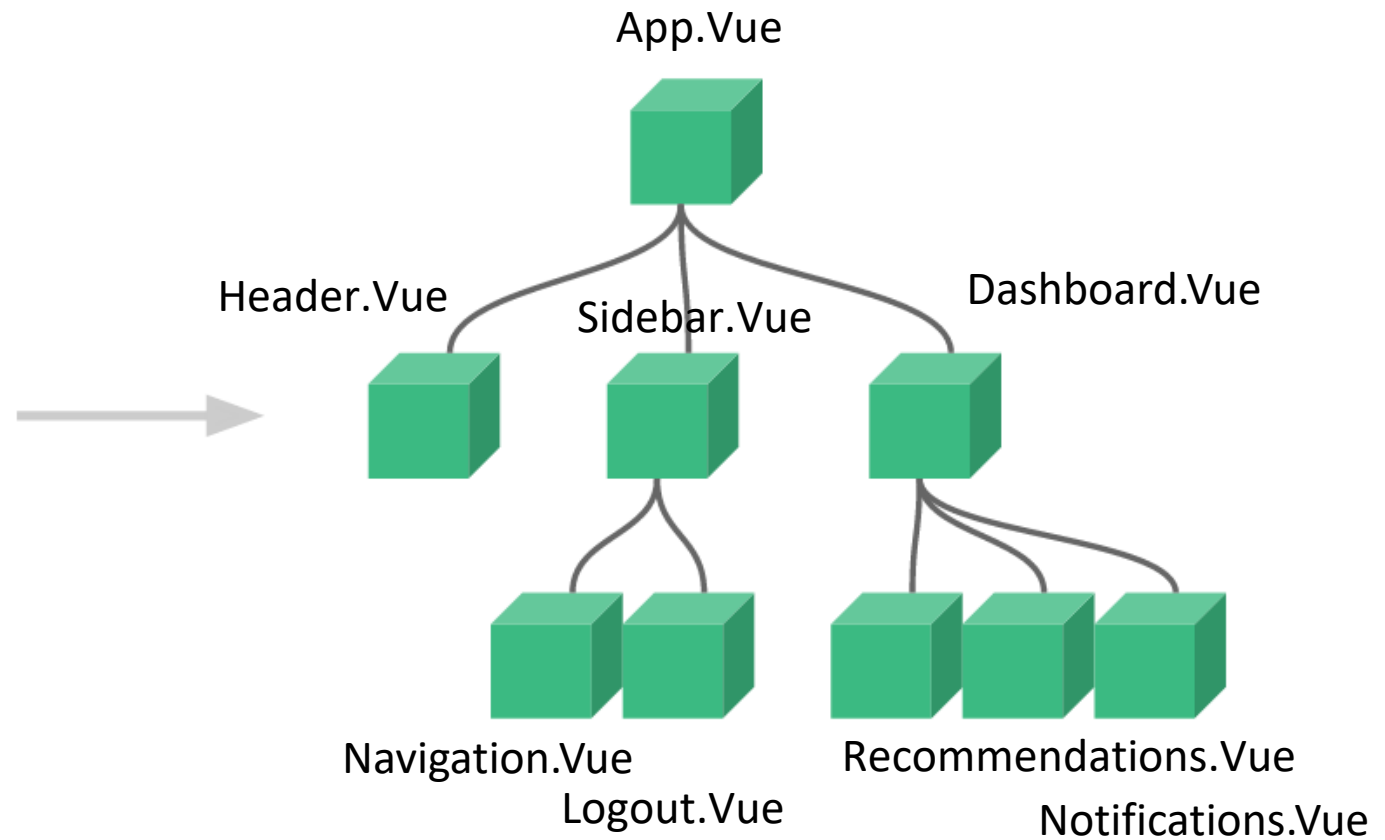
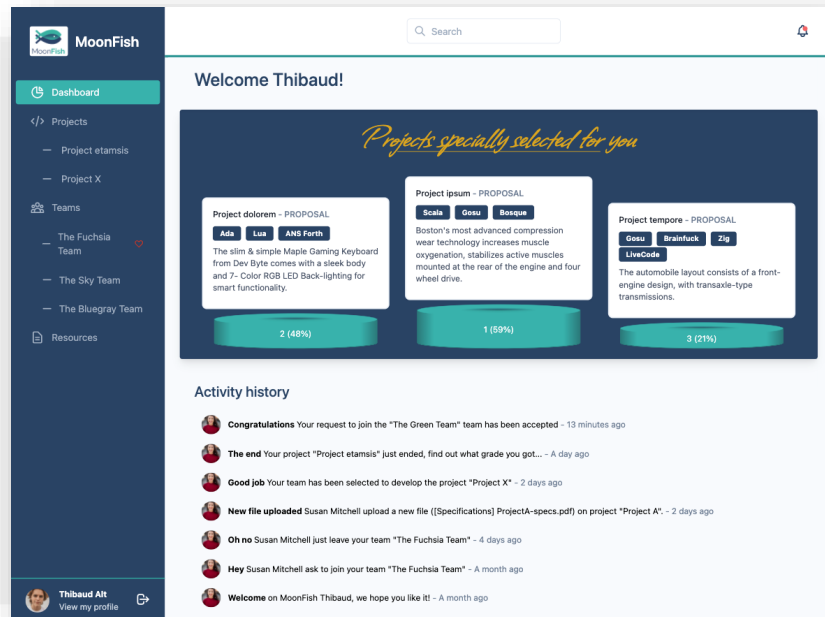
... mais les templates deviennent très vite surchargés

```
<div :class="open ? 'block' : 'hidden'" @click="$emit('show-sidebar', false)"
  class="fixed z-20 inset-0 bg-black opacity-50 transition-opacity lg:hidden"></div>
<div :class="open ? 'translate-x-0 ease-out' : '-translate-x-full ease-in'"
  class="fixed z-30 inset-y-0 left-0 w-64 transition duration-300 transform bg-blue-900 lg:translate-x-0 lg:static lg:inset-0">
  <router-link to="/" class="flex items-center justify-center mt-8">
    
    <span class="text-white text-2xl mx-3 font-semibold">MoonFish</span>
  </router-link>
  <Navigation/>
  <footer class="flex justify-between items-center py-4 px-6 border-t-2 border-teal-600 w-full fixed bottom-0">
    <div class="flex items-center">
      <div class="relative block h-10 w-10 mr-4 rounded-full overflow-hidden">
        
      </div>
      <div class="relative block text-white text-sm">
        <span class="block font-bold">{{ fullName() }}</span>
        <router-link to="/profile" class="text-gray-400 hover:text-teal-500">{{ $t('Profile.view') }}</router-link>
      </div>
    </div>
    <Logout/>
  </footer>
</div>
```

+ de 60% du code ci-dessus!

# Vue.js

- Briques de composants





# Définition d'un composant

## 1. Template HTML

```
<template>
  <main class="flex-1 bg-gray-100">
    <div class="container mx-auto px-6 py-8">
      <h1 class="text-blue-900 text-3xl font-medium">{{ welcome() }}</h1>
    </div>
    <div class="bg-blue-900 rounded shadow-lg">
      <Recommendations :number="3" @msg="transfer"/>
    </div>
    <div class="container">
      <h2 class="py-4 text-blue-900 text-2xl font-medium">
        {{ $t('Notifications.activity') }}
      </h2>
      <NotificationsList :notifications="notifications" :history="true"/>
    </div>
  </main>
</template>
```

## 2. Script JS

```
<script>
import ...;

export default {
  name: 'Dashboard',
  components: { NotificationsList, Recommendations },

  data() {
    return {
      content: '',
      notifications: [],
    };
  },

  mounted() {
    this.retrieveNotifications();
  },

  computed: {
    currentUser() {=}
  },

  methods: {
    transfer(msg) {=},
    welcome() {=},
    async retrieveNotifications() {=}
  }
};
</script>
```

# Variables statiques

```
<h1 class="text-blue-900 text-3xl font-medium">{{ welcome() }}</h1>
```

```
methods: {  
  welcome() {  
    return this.$t('Dashboard.title', {  
      firstName: capitalize(this.currentUser.firstName)  
    })  
  },  
}
```

```
computed: {  
  currentUser() {  
    if (this.$store.state.auth.user) {  
      return this.$store.state.auth.user.user;  
    }  
    return false;  
  }  
},
```

VueX

```
# ----- Pages -----  
# Dashboard  
  
Dashboard:  
  title: "Welcome {firstName}!"
```

Traduction

```
/**  
 * Capitalize a word  
 *  
 * @param {string} word the word to capitalize  
 * @return {string} the word capitalized  
 */  
export default function capitalize(word) {  
  if (typeof word === 'undefined') {  
    return '';  
  }  
  return word.toLowerCase().charAt(0).toUpperCase()  
    + word.toLowerCase().slice(1);  
}
```

Utilitaire

# Variables dynamiques et requêtes

```
<div class="container">
  <h2 class="py-4 text-blue-900 text-2xl font-medium">{{ $t('Notifications.activity') }}</h2>
  <NotificationsList :notifications="notifications" :history="true"/>
</div>
```

```
import NotificationsList from "@components/ui/NotificationsList";
import NotificationsService from "@services/notifications.service";
```

## 1. Importation de composants

```
methods: {
  async retrieveNotifications() {
    this.notifications = await request(NotificationsService.getMine(true), this)
  }
}
```

## 3. Récupération des données

```
data() {
  return {
    notifications: [],
  };
},

mounted() {
  this.retrieveNotifications();
},
```

## 2. Définition des données

# Variables dynamiques et requêtes

## 4. Requête HTTP

```
/**
 * NotificationsService
 *
 * Define all available routes for "Notifications"
 */
class NotificationsService {
  /**
   * getMine Retrieve all notifications linked to the connected user
   *
   * @return {*[]}/Promise<AxiosResponse<any>>}
   */
  getMine(archived = false) {
    const user = JSON.parse(localStorage.getItem('user'))
    let endpoint = `/notifications?filters[userUuid]=${ user.user.uuid }`
    endpoint += (archived ? '&orders[createdAt]=DESC&limit=50'
                  : '&orders[updatedAt]=DESC&filters[read]=0')
    if (user && user.user && user.user.uuid) {
      return http.get(endpoint)
    }
    return []
  }
  ...
}
```

```
// Create axios instance
const http = axios.create({ baseURL: process.env.VUE_APP_API_BASE_URL });

// Add authentication headers to requests via interceptor
http.interceptors.request.use(function(config) {return config});

// Intercept the response to resolve or to manage errors
http.interceptors.response.use(
  (response) => { return Promise.resolve(response.data); },
  (error) => {return error}
);

export default http;
```

*axios*

# Problèmes rencontrés

- Prise en main difficile de *Vuex*
- Débogueur mal pris en charge par l'IDE

# Bilan

Tailwind CSS	Vue.js	axios (Requêtes HTTP)	vue-i18n (Multilinguisme)	yup (Validation formulaires)
⚠	✓	✓	✓	✓

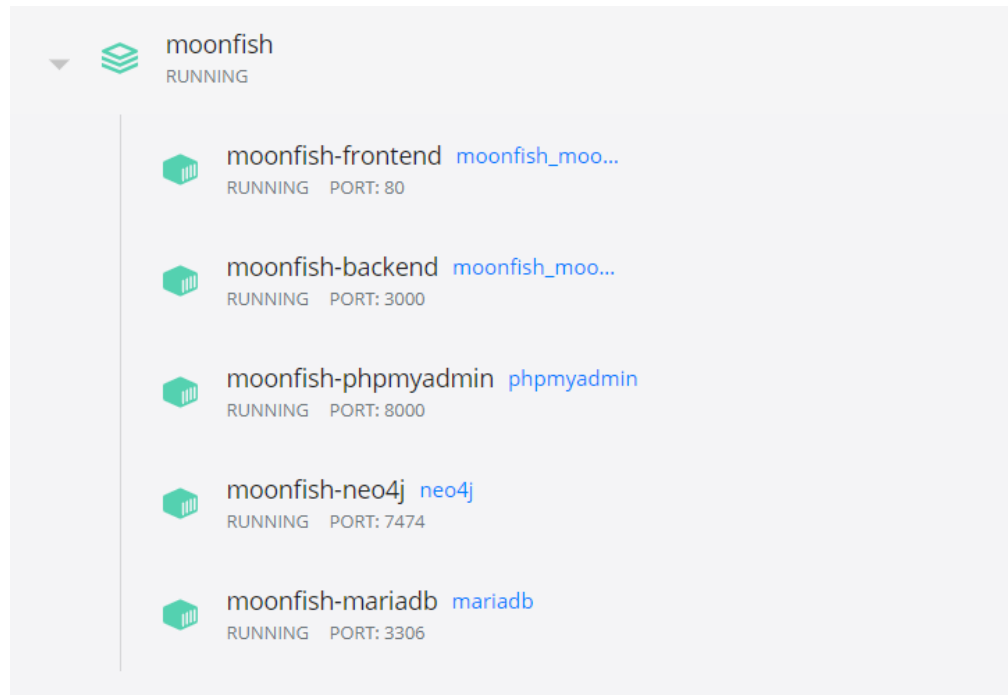
- *Vue.js* adopte les meilleurs compromis entre puissance, simplicité d'apprentissage et plaisir d'utilisation
- *Tailwind* risque d'être difficile à maintenir, une alternative serait de repartir sur *Bootstrap*
- L'utilisation de *TypeScript* avec *Vue.js* serait un bon plus

# Déploiement

*Assemblage et déploiement en une seule ligne avec Docker*

# Docker

```
$ docker-compose up
```



1. Start MariaDB
2. Start Neo4j
3. Run the *back-end*
4. Populate databases
5. Serve the *front-end*



# Problèmes rencontrés

- Service d'hébergements en ligne ne permettant pas de répondre à l'architecture incluant le serveur *Express* et deux bases de données
- La rédaction du fichier *docker-compose* fut laborieuse pour obtenir ~~quatre~~ cinq conteneurs entièrement fonctionnels

# Bilan

Docker	Netlify	Heroku	Synology
✓	✓	✗	⚠

- *Docker*, une fois mis en place, permet de gagner un temps et une facilité de déploiement considérable
- Le déploiement sur un Synology est une solution de développement acceptable mais non pérenne
- Il serait judicieux d'analyser des solutions Cloud pour une mise en production

# Résultats

*Respect des objectives, gestion et planification, améliorations possibles*

# Objectifs du cahier des charges

- ✓ Réaliser une étude de marché sommaire
- ✓ Analyser via un "*State of the art*" les différentes techniques permettant le développement d'applications web en 2021
- ✓ Définir la structure et la technologie de la ou des base(s) de données à utiliser
- ✓ Développer une première version de l'application web client-serveur
- ✓ Proposer des améliorations et/ou d'autres fonctionnalités à développer dans des versions postérieures de l'application web

# Planification et gestion

**Sprint 1 - Complete**

**Sprint 1**

Sprint 1 : Développement de l'API  
🕒 Jul 28 - Aug 6

Mise en place de l'environnement Express

[S1] Mise en place de la structure

[S1] Ajout de Neo4j et de Neode

[S1] Surcharge des endpoints existants avec Neode

**Sprint 2 - Complete**

**Sprint 2**

Sprint 2 : Développement des interfaces utilisateur  
🕒 Aug 11 - Aug 20

Mise en place de l'environnement front-end

[S2] Epic 1 : Création d'un compte et authentification

Mise en place multilingue + gestion

**Sprint 3 - Complete**

**Sprint 3**

Sprint 3 : Intégration des interfaces et de l'API  
🕒 Aug 25 - Aug 27

[S3] Gestion des erreurs, reconnexion, stockage de la session, etc.

[S3] Mettre en place la connexion entre les interface et l'API

[S3] Intégrer les diffé de l'API aux interfaces

**Sprint 4 - Complete**

**Sprint 4**

Sprint 4 : Mise en place des relations et recommandations  
🕒 Sep 1 - Sep 3

[S4] Peupler la base de données avec des données provisoires

[S4] Créer des relations entre les différentes données

**Sprint 5 - Complete**

**Sprint 5**

Sprint 5 : Finalisation du projet  
🕒 Sep 8 - Sep 17

[S5] Peupler la base de données avec des données provisoires

[S5] Tester l'application dans son ensemble et réaliser des ajustements

État	Heures planifiées	Heures réalisées	Dérive
Terminé	4	5	25%
Terminé	8	9.5	19%
Terminé	4	3.5	-13%
<b>TOTAL</b>	<b>450</b>	<b>464</b>	<b>3%</b>

# Résultats

- Les choix technologiques effectués et le résultat obtenu sont convainquant
- Le *JavaScript* reste un langage complexe et long à mettre en place même au travers de *frameworks*
- La réalisation des *epics* sélectionnés ont pris plus de temps qu'espéré et de ce fait aucun *epic* supplémentaire n'a pu être développé
- Il reste plusieurs heures de travail à réaliser pour ajouter des fonctionnalités
- De nombreuses améliorations sont encore envisageables

# Améliorations possibles

- Inscription et connexion via des services tiers
- Adaptation « *responsive* »
- Intégration de solutions de paiement
- Possibilité d'interventions de professionnels externes
- Ajout d'un système de sous-traitance
- Mise en place d'un espace de connaissances
- Améliorations des algorithmes de recommandations
- Sécurité, mise en cache, monter en charge
- ...

# Conclusion

*Acquisition de nouvelles connaissances à exploiter dans le monde professionnel*



# Conclusion

- Projet fortement enrichissant qui ma permis d'acquérir une meilleure vision des différentes technologies *JavaScript* actuelles
- Mise en place d'une architecture avec deux système de bases de données permettant de tirer les avantages de chacun
- Bon aperçu de la complexité de mise en place d'une application en partant de zéro
- Projet prometteur permettant de nombreuses valeurs ajoutées



“

*L'informatique, ça fait gagner  
beaucoup de temps... à  
condition d'en avoir beaucoup  
devant soi !*

- Mireille Sitbon

# Démonstration

*<https://heig-tb-moonfish.netlify.app>*

# Merci de votre attention

*Avez vous des questions ?*