**Scalian**

TODO List

**Avril 2019**

# Introduction

Le but d'origine est de créer une application sous VueJS afin d'évaluer le framework. Le context choisie est de réaliser une TodoList, utilisant une API REST pour sauvegarder les actions. Il devra être possible d'effectuer les opérations CRUD sur les tâches de la liste. Le présent document décrit les choix techniques significatifs ayant été effectués

## Choix techniques supplémentaires

# 

Utilisation de Typescript lors des développements. Le Typescript est une surcouche de javascript, il permet notamment l'utilisation du typage statique. Les fonctionnalités d'ECMAScript2015 comme les classes ou les modules sont égalements incluses. Typescript est devenu le langage obligatoire pour l'utilisation de Angular2, et son intégration avec VueJS est de plus en plus significative. La prochaine version majeure de Vue sera elle-même écrite en Typescript.



JSON-Server est un utilitaire permettant de créer des API REST à partir de simples fichier json, principalement pour des tests et des prototypes. Nous l’utiliserons pour simuler un serveur basique permettant d’interagir avec les tâches de notre application.

**AXIOS**

Librairie utilisée côté client pour faire les appels REST avec gestion asynchrone des retours.

## Ecosystème

Plusieurs outils sont largement recommandés pour du développement front VueJS, certains sont des standards, d’autres sont plus spécifiques à la technologie:

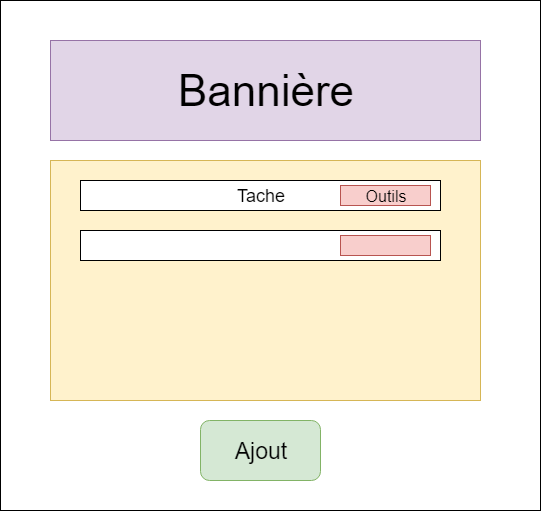
* **npm** : Outil de partage des packages javascript.
* **Visual Studio code** : Éditeur de texte recommandé pour Vue, avec les extensions Vetur et TSLint.
* **Vue.js devtools** : extension disponible pour Chrome et firefox.

La version de base de VueJS va à l’essentielle en terme de fonctionnalitées proposées, il est nécessaire d’ajouter des packages pour la gestion d’aspect supplémentaires. Dans notre cas :

* **Vue CLI** : Interface de scafolding pour Vue, permet de générer des templates de projets, de nombreuses options sont possibles selon le projet.

# Ergonomie

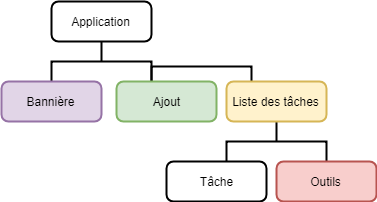
Avant de se lancer directement sur le développement de l’application, une version basique de l’interface souhaitée est réalisé afin de guider la démarche :



Avec cette interface un affinement des besoins métiers est nécessaires, plutôt qu’un nouvel écran permettant les opérations CRUD classique, on utilisera les capacités d’application single page offertes par Vue. L’idée est que ces opérations soient proposées sur la page principale via des contrôles spécifique, plutôt que de nécessiter des rechargements de pages. Cette approche permet de mieux tester les capacités d’évènements et de réactivité de Vue.

Les composants Vue sont liés les uns aux autres, cette vision permet de planifier la gestion des évènements.

Graphe simplifié des composants :



# Application

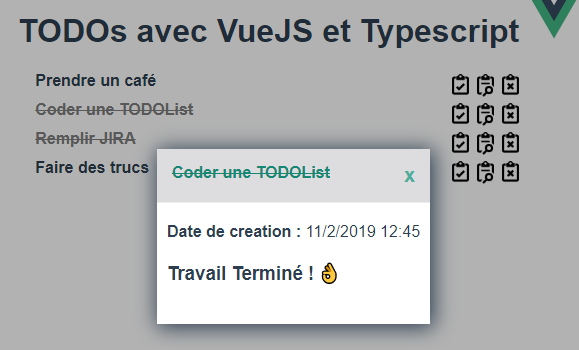
## Présentation et capture d’écrans

Interface de base :



Ajout d’une tâche :

# 

Clic sur l'icône de détails :

## Lancement de l’application

Une fois l’application récupérée sous Gitlab :

* Ce placer dans le répertoire *src/server* et lancer la commande : ***json-server --watch tasks.json*** Cette commande permet de lancer le serveur json qui répondra aux requêtes REST de notre application. Le serveur est accessible sur <http://localhost:3000/tasks>.
* Ce placer dans le répertoire principal, *TODOList*, et lancer la commande : ***npm run serve***  Cette commande lance l’application Vue.js, accessible sur <http://localhost:8080/>.

# Retours

Les retours que je peux faire sont basés sur l’expérience que j’ai pu avoir en réalisant ce prototype, mais également sur la partie veille que j’ai effectuée avant de me lancer sur l’application.

**Positif**

* Prise en main et flexibilité
* Vue CLI pour la génération de projets
* Documentation complète et à jours
* Capacités de manipulation du DOM similaire à Angular/React
* Librairie annexes supportées officiellement pour les problématiques majeurs (Vuex, vue router, vue-test-utils, ...)

**Négatif**

* Moins utilisé que React et Vue
* Taille de la communauté, moins de questions et exemples
* Trop permissif, nécessité de rechercher les bonnes pratiques pour rester sur une architecture pérenne

# Conclusion

Vue est une alternative crédible à Angular et React, le framework répond aux mêmes problématiques que ces deux derniers.

Similaire à Angular dans l’approche, la proximité limite d’autant plus la phase d’apprentissage pour des développeurs habitué celui-ci. Dans le cas de projets plus simples, mais voués à évoluer, Vue permet une approche plus efficace, avec un ajout progressif des fonctionnalités nécessaires.

Plus récent, le nombre de projets utilisants actuellement Vue n’est pas encore aussi élevé, mais sa popularité au niveau des développeurs explique sa progression rapide et lui assure une place de choix en tant que framework Front pour les prochaines années.