Master 2 Informatique : Génie Logiciel

Rapport de stage
Cirrusware

Professeur : Laurent Réveillère
Maitre de stage : Vincent Jaulin

Auteur :
Alexandre Erard

18 août 2021

1 Remerciment

2 Avant propos

Ce rapport s'inscrit dans le cadre du stage de fin d'études d'informatique de l'Université de Bordeaux, plus précisément dans la spécialité Génie Logiciel.

Au cours de mon cursus, j'ai pu m'intéresser à divers pan de l'informatique comme par exemple le développement d'application mobile native, l'intelligence artificielle, la gestion des bases de données et plus largement le Big Data, le devellopement web. Celui qui m'a le plus passionné est l'architecture logiciel, c'est à dire construire des applications scalable, maintenable tout en restant performante. Pour moi il prend tout son sens dans le développement d'application web qui ont pour obligation d'optimiser leur performances, ainsi que leur consommation de ressource tout en étant capable d'ajouter facilement de nouvelles fonctionnalités afin de s'adapter aux nouveaux besoins.

J'ai donc recherché un stage dans ce domaine et l'entreprise Cirrusware m'a offert l'opportunité de participer à leur projet afin que je développe mes compétences de développeur full stack.

Table des matières

3 Contexte du stage

3.1 Cirrusware

Cirrusware est une PME informatique française spécialisée dans le développement d'applications web ayant pour objectif de faciliter les communications numériques au seins des entreprises ainsi que de facilter la gestion des relations entre clients et contacts.

Fonder en 2013, Cirrusware a su convaincre de grands groupes français tels que *RANDSTAD*, *SAFRAN*, *MAÏSADOUR*, *OCEALIA*, *CARREFOUR*, ou encore *INTERCONTINENTAL - grand hotel de Bordeaux.* (Pour plus d'information, voir)

Aujourd'hui l'entreprise compte 13 employés dont 6 développeurs 5 chefs de projet et 2 commerciaux.

Cirrusware développe 4 type principaux de produits :

La plateforme Send-Up [1]

Un XRM (CRM).

3.2 La plateforme Send Up

Send-Up est le premier produit de Cirrusware et permet au client de facilité leur communication marketing auprès de leurs différent contacts. Les principales fonctionalités de la plateforme sont :

- La création de campagne Emailing, SMS.
- La création de Site/Landing Page via leur Studio digitale.
- L'automatisation de campagne ou autre tâches répétives via l'automation.
- Stockage cloud

En plus de campagne Emailing personaliser en fonction des destinataire. La plateforme possède également un éditeur PDF afin de réaliser des campagne papier.

Le Studio digitale est lui un éditeur WYSIWYG [1] de page web. Il permet en glissant simplement des block paramétrable afin d'atteindre le résultat souhaité. Il permet aussi d'acceder facilement au code source de la page afin d'ajouter des fonctionalité

1. What you see is what you get

et comportement à la page pour les clients qui font une utilisation plus poussé de l'outil.

L'automation lui permet d'automatiser les tâches les plus répétives de la coommunication marketing, comme l'envoie d'une capagne toutes les semaines, ou encore l'envoie d'un mail à chaque inscription sur le site d'un client.

En plus de fournir la plateforme. Cirrusware propose d'accompagné ses clients dans leur différents projet. Allant de la création de campagne efficace à l'intégration de site web. L'entreprise propose aussi des formations affin de permettre à leur client d'être automome dans les fonctionalité les plus poussé de la plateforme.

3.3 Le XRM

L'XRM est un outil permettant d'optimiser la relations entre client, contact et fournisseur. Il est totalement intégré à la platforme Send-Up.

Il permet :
- une gestion de portefeuille client.
- une gestion des actions (rendez-vous, notes, compte-rendu, email, sms)
- une gestions de diverse oportunitées.
- utilsation Hors ligne
- une adaptation au besoin du client.

Le CRM est un outils très modulable qui permet donc pour chaque nouveaux client d'intégrer des fonctionalités unique. Comme la visualisation de statistiques particulières, ou encore la création d'audit.

4 Outils et technologies utilisés

Lors de mon arrivé dans l'entreprise, j'ai du utiliser les outils internes de gestion de projet, de communication, mais aussi de développement. Lors des projets d'études, nous étions une équipe de 6 personnes maximum et nous utilisions principalement des outils tels que Discord pour communiquer, VSCode pour développer, et GitHub Issue ou Trello pour gérer les projets. Durant mon stage j'ai du utiliser des outils plus professionels et complets afin de pouvoir travailler sur des projets de plus grandes tailles. Voici la liste exhaustive de ces outils.

4.1 Outils

4.1.1 Monday

Monday est un outil de gestion de projet simple qui permet de créer des tableaux et y ajouter des tâches. Chaque tâche peut être attribuée à un quelqu'un, elle posséde un état (en cours, terminé, etc.) et une priorité. De plus chaque tâche possède un espace "discussion" qui permet de faire état de l'avancement ou encore de noter certaines informations importantes.

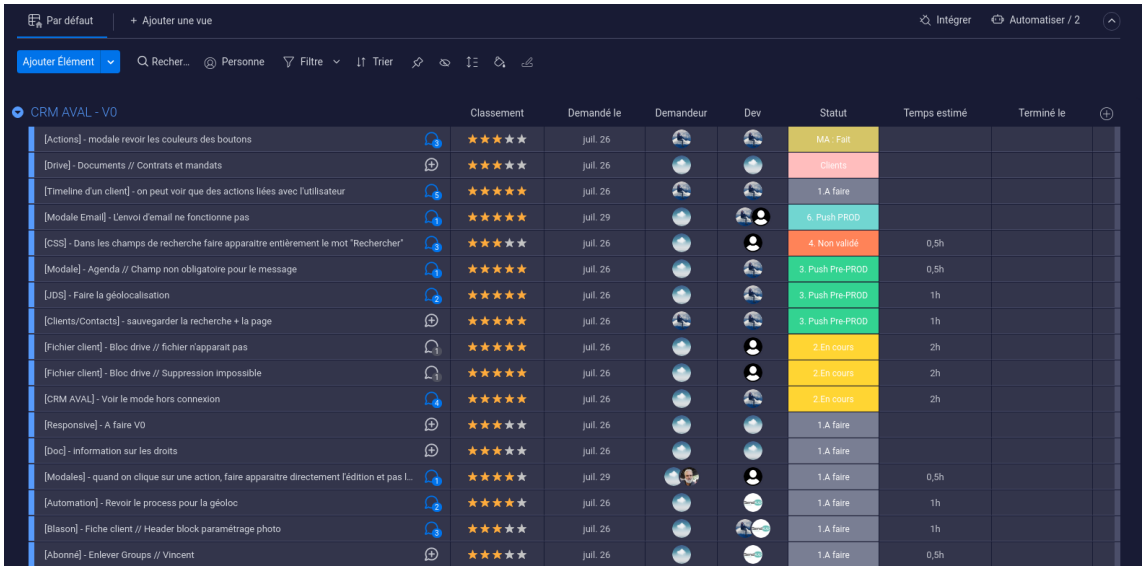

FIGURE 1 – Exemple de tableau Monday

4.1.2 Google Workspace

Afin de faciliter la communication au sein des équipes, l'entreprise utilise Google Workspace pour produire de la documentation (Cahier des charges, etc...), GMail et

Google Chat afin d'échanger rapidement, Google Agenda pour planifier les rendez-vous. De plus, l'entreprise ayant des clients se trouvant dans toute la France, et tenant compte de la situation sanitaire actuelle l'applications Google Meet est utilisée comme application de visio conférence, pour les diverses réunions et rendez-vous projets.

4.1.3 Git et GitKraken

Git est un logiciel de gestion de version, il permet de créer des projets et de les versionner. Il permet également le partage du code ainsi que le développement sur plusieurs branches, afin de ne pas empiéter sur le travail des autres membres de l'équipe.

GitKraken lui est une interface graphique utilisant git. En effet git s'utilisant principalement par ligne de commande il est parfois difficile à utiliser. GitKraken permet de faciliter l'utilisation de git et permet de tout faire en passant par l'interface (pull, merge, se déplacer entre les branche, etc..).

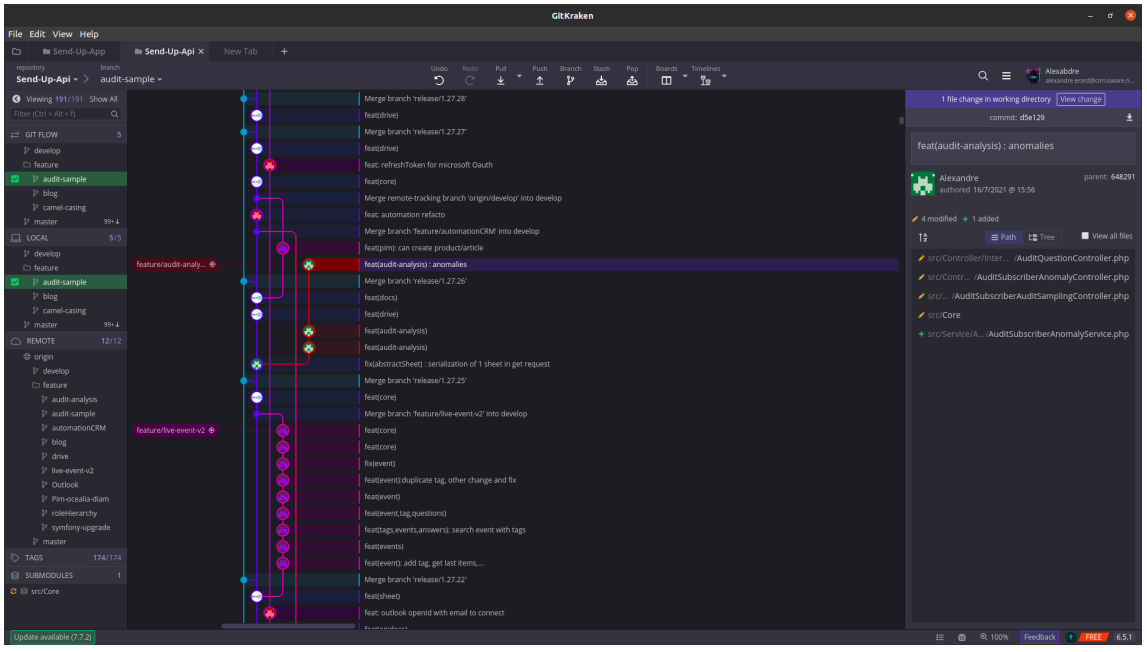

FIGURE 2 – Interface de GitKraken

De plus il permet une totale intégration des git flow. C'est à dire créer facilement une branche par nouvelle feature, et simplement merge avec la branche de développement lorsque celle ci est terminée. Les git flow permettent également une meilleure gestion des HotFix et des releases mais je ne les ai pas utilisé lors de mon stage.

FIGURE 3 – Déroulement d'un git flow

4.1.4 PHPStorm

PHPStorm créé par JETBRAIN et l'un des IDE les plus complet pour le développement web. A l'aide de ses nombreux pluggins il permet une totale prise en charge des framework comme Symphony et VueJS. Il permet de facilement de refactoriser du code ou encore de créer de nouveaux snippets afin de faciliter les phases de développement. Malgré ses nombreux points positifs, il semble dur à prendre en main au début mais par chance au cours de mes études supérieures j'ai été amené à utiliser InteliJ IDEA, ou encore Android studio qui sont deux IDE de JetBrain qui ont exactement la même structure.

//TODO : Screen PHPStorm

4.1.5 Postman

En ce moment l'entreprise ouvre son API de plus en plus à ses clients afin qu'ils puissent eux même intégrer les différents services de la plateforme. Et c'est pour cette raison que l'API doit devenir de plus en plus robuste. Dans cette optique nous utilisons l'application Postman afin de facilement créer nos requêtes ainsi que des tests associés. L'outil permet de faire hériter des test à des sous collections ce qui nous permet de gagner du temps lorsque certains test sont génériques comme par exemple un temps de réponse inférieure à 200ms ou encore un statut de réponse différent de 500.

De plus il intègre un *collection runner*[7] qui permet d'agencer l'ordre des requêtes, ainsi que de les lancer n fois, à un intervalle de temps y.

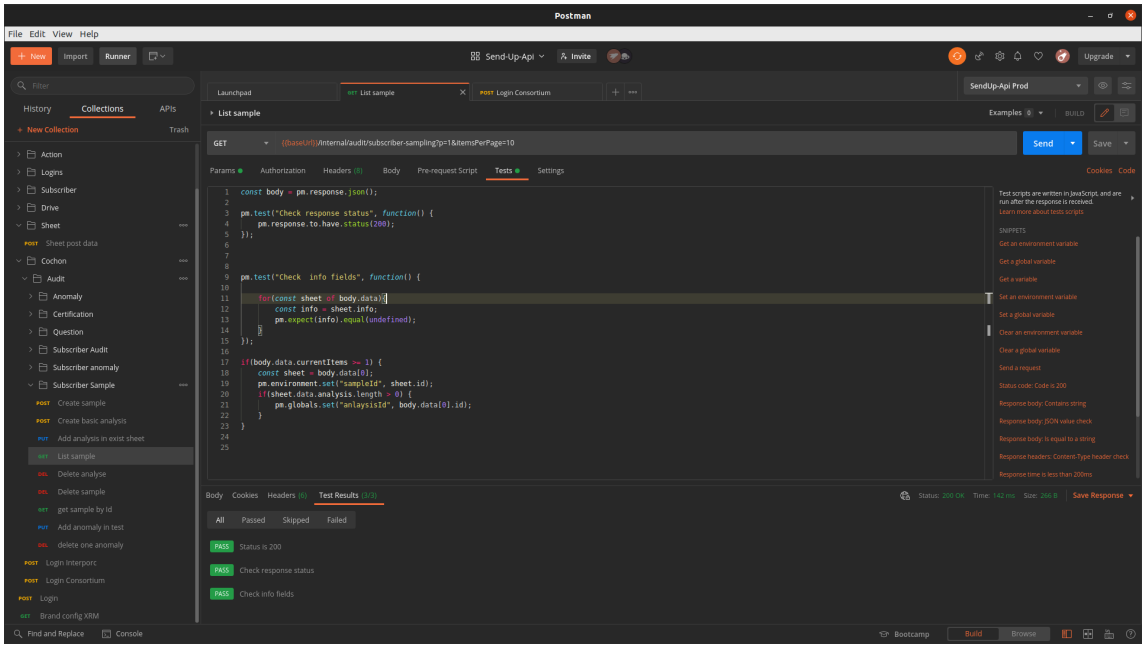

FIGURE 4 – Exemple de test Postman

4.2 Technologies

Historiquement la plateforme Send-Up à était développé sur une stack Symphony 2.8/Jquery/Bootstrap. Aujourd'hui l'entreprise est en pleine transition technologique. Son objectif est de se passer de Jquery et Bootstrap au profit des nouvelles technologies du web et plus particulièrement du framework VueJS. Cette transition reste un défi pour l'entreprise car certains outils du site comme le *studio digital* doivent garder la "legacy" afin de garder une rétro-compatibilité mais aussi afin de facilité son usage au prés des clients. Voyons plus en détails les technologies choisies par l'entreprise à savoir Symfony et VueJS.

4.2.1 Symfony

Symfony [2] est un framework PHP français développée en 2005 par Fabien Potencier. Basé sur le pattern Modèle-Vue-Controller (MVC), il est actuellement en version 5.x.x Le gros avantage de Symfony sur les autres framework php du même type comme Laravel, ou CakePHP est sa documentation qui en plus d'être riche et également traduite en français. De plus, il possède une communauté très importante et réactive. En effet a ce jour il existe plus de X bundle sur la version 2.8. Et énormément de réponses sur les forums de developpeur. Il intègre également des outils qui sont extrêment pratiques dans la gestion des données comme par exemple

Doctrine [3] un ORM (Object Relation Manager).

Un ORM permet de facilement gérer les données en transformant directement les résultats des requêtes en Entité php. Il intègre également un langage d'intérrogation de la base de données nommé DQL. Ce langage rend plus permet de rendre plus lisible les requêtes et l'intégration de variable dans celle-ci. D'autre outils sont également présents comme l'injecteur de dépendance ou encore des outils de debug disponnibles en environement de developement.

4.2.2 VueJS

Pour remplacer la partie Jquery de Send-Up le framework VueJs [4] a était choisi par l'équipe. Il s'agit d'un framework open-source développer à l'origine par Evan You en 2013. Ce framework facilite la création d'interface graphique et de Single Page Application (SPA). Pour présenter rapidement le framework voici un exemple de composant basique.

Un fichier .vue est composé de 3 élements principaux.

- **template** : contient le code HTML de l'interface graphique.
- **script** : contient la logique et le comportement du composant.
- **style** : contient les styles CSS du composant. Le style peux être utilisé avec une props *scoped* afin de limiter les styles à l'intérieur du composant.

La balise script est l'endroit ou on va déclarer le composant et tout ce dont il a besoin pour fonctionner. On y retrouve donc les sous composants qu'il utilise dans son rendu, les paramètres (appelé *props*) que son parent peut lui passer, ici on attend un paramètres *msg* de type *String*. On retrouve également l'état du composant avec toutes les données utilisées,un compteur et la chaîne de caractère *hello*, les méthodes qui vont déterminer la logique, les propriétés calculées du composant. Les propriétés calculées sont appelables comme les variables indiquées dans data. Mais ne sont calculées qu'une seule fois à la création du composant puis lors de la modification de variable utilisée à l'intérieur. De plus le framework possède une catégorie d'objet nommé *Mixins* dont on parelera plus en détails plus tard.

Voici le rendu de ce composant appeler avec $msg = Friends$:

On y retrouve également les divers événement du cycle de vie au quelle on peut se brancher pour effectuer certains traitement spécifiques. Pour plus de détails voir l'annexes 7

```vue
<template>
  <div>
    <h1>
      {{ helloworld }}
    </h1>
    <button @click="count">CLICK ME</button>
  </div>
</template>

<script>
export default {
  name: "HelloWorld",
  components: {},
  mixins: [],
  props: {
    msg: String,
  },
  data() {
    return {
      hello: "Hello",
      count: 0,
    };
  },
  computed: {
    helloworld() {
      return this.hello + " " + this.msg + "!";
    },
  },
  methods: {
    count: function () {
      this.count++;
    },
  },
  mounted() {
    console.log("HelloWorld component mounted");
  },
};
</script>

<style></style>
```

FIGURE 5 – Exemple de composant VueJS

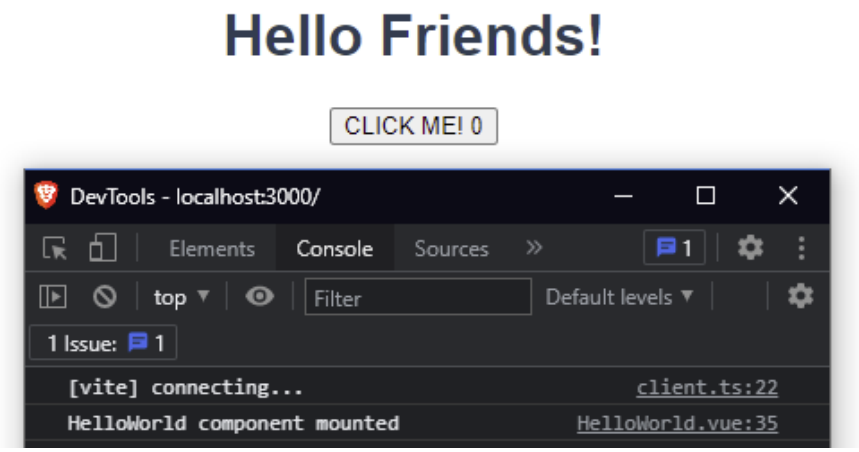

FIGURE 6 – Exemple de composant

5 Missions réalisées

Au cours de mon stage, j'ai pu intervenir sur différents projets allant d'un simple script permettant de dynamiser une page web d'un client à la reprise d'un projet complet. Ici nous allons détailler les deux principales missions de mon stage à savoir le développement de widget en vueJS pour le studio Digital et la finalisation des XRM du *Consortium du Jambon de Bayonne* et d'*Interporc Franche-Compté*.

5.1 Widget pour le studio Digital

Lors de mon arrivé dans l'entreprise, *CIRRUSWARE* cherchait à valoriser ses formations en devenant Organisme de formation certifié **QUALIOPI**. QUALIOPI est une certification qui garantie des formations avec :

- des bons résultats
- des adaptations au besoins
- des formateurs qualifiés
- une bonne prise en compte des retours face au formations

Afin de correspondre à ces critères, nous avions besoin de construire un site web qui permettrait d'effectuer les questionnaires des différents modules de la formations, ainsi que le formulaire de satisfaction. Puis par la suite de pouvoir consulter les différentes statistiques des différents modules sous 3 vue différentes, formés, qui doit voir uniquement leurs résultats, entreprise qui ne voit que les statistiques de ses employés formés, une vision globale qui permettra de voir toutes les statistiques des personnes formées.

La première étape réalisée afin de pouvoir réaliser ceci à était d'adapter l'enregistrement des formulaires du studio qui jusque là ne nous permet pas d'effectuer simplement des statistiques.

5.1.1 Serializations des données des formulaires

Afin de récolter les données des formulaire, le studio digital passe par un serializer qui va enregistrer dans un fichier json les valeurs des différents champs associés à leurs id. Ce système présente plusieurs avantage comme la facilité et la rapidité d'implémentation. Maleureusement ceci ne nous permet pas de savoir si une réponse est correcte ou non dans notre cas. Ou encore pouvoir associer le label d'une question à son résultat.

Pour atteindre le résultat souhaité, j'ai dû implémenter un nouveau widget pour le studio digital en VueJS. Ce widget est un simple bouton de correction qui va devoir récupérer tout le contenu nécessaire du formulaire et de le stocker dans des champs cachés afin de servir le serializer. Le reste du formulaire étant coder en JQuery/html,

la seule solution était donc de parcourir le formulaire et de récupérer les information dans le DOM. A savoir les information permettant le calcule de la note, diverses meta-données par rapport au question comme le label, l'id du champ concerné,le résultat obtenu a la question, etc..

Dès que ces informations fut récupérer il ne rester plus qu'a établir les différentes requêtes nécessaire à la récupération des données en fonction du périmètre défini (formé, entreprise, admin)

5.1.2 Compréhension de l'API et implémentation des requêtes

Dans l'actuel implémentation du serveur Send-Up, la partie gérant les *Recordset* - Table MySQL comportant un champ info destiné à stocker du JSON - est gérer par une partie de l'API dont les routes pointant vers cette partie sont générer automatiquement à partir du nom de la méthode définie dans le controlleur.Exemple la méthode postRecordset est appelée lors d'une requête POST sur /Recordset. De plus l'une des complications a été de comprendre le format des données à envoyer dans la requête. En cherchant plus profondément dans le code je me suis rendu compte que seuls les **FormData** était accepter pour la requête existante. Une requête qui permetter de lister tous les Recordset lié à l'entreprise qui les a créé, Dans notre cas il s'agit de *Send-Up Formation*. Cette requête permet donc de répondre à la demande de vue administrateur. Afin d'avoir une vue "entreprise", on associe chaque utilisateur à un champs customisé nommé "entreprise" qui correspond au vrai nom de son entreprise. Grâce à cela on à juste a rajouter un paramètre à la requete précédente afin d'avoir une vue réduite sur les données. Pour récuppérer les données d'un seul utilisateur, il nous suffit juste de de filter sur l'id de l'utilisateur.

5.1.3 Création des widgets

Une fois les données récupérées et les requêtes créées il fallait que je créais 5 widgets différents qui permettait de :

- visualiser les résultats d'une personne formée pour chaque tentative.
- visualiser les réponses avec correction d'une personne formée.
- afficher les statistiques globales de la formation c'est a dire le nombre de bonne réponse par rapport au nombre de personnes.
- afficher les statisques globales pour le formulaire de satisfaction client.
- afficher les réponses d'une personne formé au questionnaire de satisfaction.

Après avoir développer certains d'entre eux, il s'avéra que certains traitements se répéter ou se ressembler fortement. Or VueJS intègre des *Mixins*, ces éléments possède les mêmes caractéristique dans leur balise script qu'un composant vueJS classique. Mais ce dernier à la possibilité d'être intégré à l'intérieure de composant (ou autre Mixins). Chaque composant hérite donc des données, méthodes et autres propriétés

calculées. Ceci empêche donc la redondance et facilite la maintenabilité. C'est donc pour cela, que j'ai créé un mixins spéciale afin de facilité la lisibilité et la maintenance du code.

5.2 Reprise d'un projet XRM

Suite au départ d'un développeur de l'équipe. On m'a placer sur la reprise de son projet qui consite a creer un XRM permetant tous les points de contrôle concernant le *Consortium du Jambon de Bayonne* et *Interporc Franche-Compté*. Un XRM est un logiciel de gestion des relations tiers. C'est à dire des clients, fournisseur, revendeur etc... En plus de réaliser ceci le notre doit permettre de pouvoir de créer et de réaliser des audits, en fonction du type d'activité et du cahier des charges de la personne audité, et de pouvoir saisir les données issue d'analyse laboratoire.

Toute la partie audit a était réaliser par mon prédécésseur, il me reste donc à implémenter la parite Analyse et prélévement.

5.2.1 Fonctionnalités

Tout d'abord passons en revus les fonctionalités principales souhaité par les clients. Ici nous avons 2 client différents avec des besoins différents.

- En tant que Consortium, je souhaite pouvoir créer des fiches prélevements. En fonction du cahier des charges, type d'activité de la personne audité il est possible d'avoir plusieurs champs différents définis dans un fichier de configuration.
- En tant que Consortium, je souhaite pouvoir ajouter des des analyses à un prélévement. Chaque analyse possède un type dépendant des champs sélectionnés dans la fiche prélévement. Ceci conditionera les champs spécifique de l'analyse
- En tant que qu'Interporc, je souhaite pouvoir créer des fiches analyse. Les champs spécifiques dépendent uniquement du type de produit sélectionner : Saucisse, Lisier, Lactoserum, Sel.
- En tant qu'Interporc et Consortium, je souhaite pouvoir lever des écarts à chaques analyses si besoin est.
- En tant qu'Interporc et Consortium, je souhaite pouvoir téléverser les résultats des laboratoires et les associé à une analyse.

5.2.2 Analyse de l'existant

Après avoir discuter avec mon chef des projet, et mon maitre de stage, on a determiner qu'un audit et un prélévement se ressembler énormément. En effet chacun

et relier à un auditeur/préleveur, une personne audité, un ou plusieurs cahier des charges et un ou plusieurs type d'activités.

Pour cela les audits sont stocké dans une table nommé *Recordset_ app*.Cette classe possède un champ info au format JSON, comme la classe *Recordset* décrit précédement. La grande différence est que cette dernière est gérér par une classe abstraite nommé *AbstractSheet*, qui définis que le champs info possède au minimum un un champ builder, data et files. Chaque entité voulant être stocker dans la table Recorset_app doit étre hérité de cette classe et donc respecter ce schéma. Pour différencier nos sous classe nous faisont appel une fonction de doctrine appelée le discriminant. Il permet d'associé un entier a une classe. A ce jour nous avons 2 classe héritant de AbstracSheet :

1. *Recordset_ app* : qui correspond à toute les fiches standard.
2. *SubscriberAudit* : qui correspond au audit créer par nos subscriber.

Voila pour léxistant au niveaux de lápi. Voyons maintenant comment à était construit les audit côtés interface utilisateur afin de pouvoir garder une cohérence dans l'application et de pouvoir réutiliser au maximum les composants déjà créer.

Premierement commençons par la liste des audits crées. Cette liste comporte 3 composant principaux :

- *AuditSubscriberListGrid* : Les composants suffixer par **Grid** dans la CRM sont des composant paramétrable dans le fichier de configuration d'une CRM. Ces composant integre un composant integre un **GridBlock** qui permette d'integrer un composant dynamiquement dans l'écrand d'accueil ou encore dans la barre de navigation latéral. Des routes dynaque seront générer à partir de la configuration.
- *AuditSubscriberList* : qui va récupérer les données du serveur et les passer au composant enfant avec les en-têtes du tablaux souhaité.
- *AuditSubscriberListSimple* qui va intégré le composant data-table de la librairie Vuetify.

Afin de rendre ces composants les plus réutilisable possible il a utilisé le système de **slots** [5] de Vue. Un slot permet de réserver une place à un composant ou autre template html qui sera injecter depuis un composant parent. Un composant parent peu lui même définir qu'un slot enfant comprend un autre slot. Ainsi Cela permet une total personalisation et donc une réutilisation total du composant. Les clients souhaitaient pouvoir créer des écarts lorsqu'une analyse est non conforme. Un écarts est une sorte de ticket qui explique la non conformité d'une question d'audit ou d'une analyse. Il est égalment accompagné d'information complémentaire comme sa gravité ou encore des commentaire associe. Or la création d'écarts à déjà était créer dans la partie audit, et les prélévement ont été construit de sorte a respecter la

même interface, le même contract que les audit. Cela rend donc possible la réutilisation du composant. Ceci nécécittera de refactorer une partie de l'api car certaines options devenaient optionnel, celon le context **audit** ou **prélèvement**. Pour plus de précision, voir l'architectecture des recordset7. On repetera cette architectecture également pour les controlleur.

Nous souhaitons également pouvoir téléverser des fichier lié a nos analyse. Il s'agit d'une Fonctionnalités très utiliser dans la CRM est un composant exite donc déjà.

5.2.3 Les formulaires des prélévement

Afin de nous indiqué leur besoins, le *Consortium* nous a envoyé un fichier excel correspondant a toutes les conditions possible pour les formulaire de prélévement et d'analyse. Une configuration de prélévement dépend de trois grands critères :

- Type d'activité
- Cahier des charge
- Le type de produit

Le type d'activité et le cahier des charges dépendent de l'organisme chez qui on effectue le prélévement. La liste des produits nous a été fournis dans le fichier excel et dépend essentielement du type d'activité et du cahier des charges. Pour contruire facilement des interfaces de formulaire, nous utilisons une librairie nommé *Koumoul* [6]. Koumoul est une implémentation Vue/Vuetify de la norme JSON Schema Form [7]. Cette norme permet de facilement construire des formulaire a partir d'un fichier JSON. De nombreuse implémentation existe pour les différent framework javascript, comme React ou Angular. L'un des avantage en plus de la rapidié et la facilité de construction, il est donc réutilisable au travers différent composant d'une applications mais également au travers de plusieurs application quelque soit la technologie utilisé. Koumoul à la particularité d'utilisé la librairie graphique Vuetify qui permet donc d'utiliser les composants de cette librairie.

Pour me faciliter le travail, j'ai donc écrit un script permetant de parser le fichier qu'on nous a mis a disposition afin de le rendre exploitable par notre application.

Au vue de toute les conditions nécessaire j'ai donc choisi d'architecturer mon fichier à la manière de l'annexe 7. Cette configuration me permet de facilement choisir le schéma souhaité à l'aide des information choisie par l'utilisateur. Il me permet également de facilement proposer les différents produit ou type d'analyse possible celons la configuration choisie.

Après avoir construit cela divers problème lié à la gestion des variable de koumoul sont survenu. Dans vue, une des notions les plus importante est le **v-model** [8]. Il s'agit d'une variable passer en props par un parents a un composant enfant qui aura la particularité d'être modifiable lors de l'émition de l'évenement input par

l'enfant. Or lorsque je souhaiter envoyé le résultat de mon formulaire au serveur celui-ci ne s'était pas mis à jour. La seul solution pour résoudre ce problème de synchronisation est l'appel à la méthode *nextTick* [9] de Vue. Cette méthode permet de différer l'appel d'une fonction au prochain cycle de mise à jour du DOM.

Ici je vous présente uniquement le cas du Consortium car il s'agit du plus complexe des deux et les éléments technologique utilisé dans la résolution de celui-ci sont également utilisé lors du développement du cas d'Interporc. En effet, la configuration pour Interporc, est beaucoup plus simple et ne dépend que d'un seul élément. Ceci me force donc à créer un composant spécifique pour le Consortium et un autre pour Interporc car leur composition sont trop différente. Afin d'eviter a nouveaux la redondance de code je créer donc un Mixins pour centraliser la logique comumne.

5.2.4 Ajout des écarts

Une fois la configuration et les interfaces des formulaire d'analyse/prélèvement effectuer. La prochaine étapes vu d'intégré l'ajout des écarts à une analyse. Ici je n'avais pas de nouveaux composant à créer, mais la difficulté résida dans la compréhension de ce qu'avait fais mon prédécésseur. Cette conpréhension fut d'autant plus compliqué qu'aucune documentation n'était livrer avec ce composant. Les endroits où il réccupérer les données n'était pas claire et cela m'a donc fait prendre du retard sur mon avancer.

5.2.5 Reste à faire

A leur actuel cette partie est en train d'être tester par les clients qui devraient faire leur retour sous peu. Il manque malgré cela quelques fonctionalités comme l'export dans un fichier csv. de toutes les analyses, en laissant la possibilité de filtrer sur certain critères. Au niveaux globale de la CRM le projet devrait se poursuivre jusqu'en décembre car il reste encore 2 grandes parties communes au clients et plusieurs petites fonctionalités spécifique à developper.

6 Conclusion

Après avoir participé à plusieurs petits projets et 2 gros (développer ci-dessus), j'ai appris l'importance de la communication entre les équipes. J'ai été amener à réaliser plusieurs petit script pour aider les chefs de projets dans le studio digital, lors de ces missions, j'ai fais tout mon possible pour expliquer le fonctionnement du code pour qu'il puisse si nécessaire le réutiliser dans d'autre projet pour des situations similaires. En plus de ces missions, j'ai essayé le plus de partager avec mon chef de projet, afin de lui faire pars de mes avancés, mes problématiques et mes idées. La reprises d'un projet existant m'a également beaucoup appris sur le passage de connaisances entre développeur. En effet l'une des principales difficulté lors de ce projet à était la reprinse d'un code qui sur certains aspect mal documenter.

Du côté plus technique, j'ai beaucoup appris sur le fonctionnement général de javascript, en plus d'avoir approfondis mes compétences technique en VueJs tout en améliorant mes compétences en veille technologique. Pour ce qui ai du backend, j'ai essayé au plus possible de documenter mes requête afin de faciliter leur compréhension mais également d'integrer des tests à l'aide de postman. Ces tests me permetter facilement de savoir si une fonctionalité existante a été impacter par les modifications du code.

Pour conclure, ce stage m'a donc permis d'approfondir mes compétences sociales et technique. Il m'a conforter dans l'idée de faire du développement web mon métier. De plus le déroulement de mon stage s'étant bien passer j'ai donc choisie de continuer à travailler au poste de développeur fullstack dans l'entreprise Cirrusware.

7 Anexes

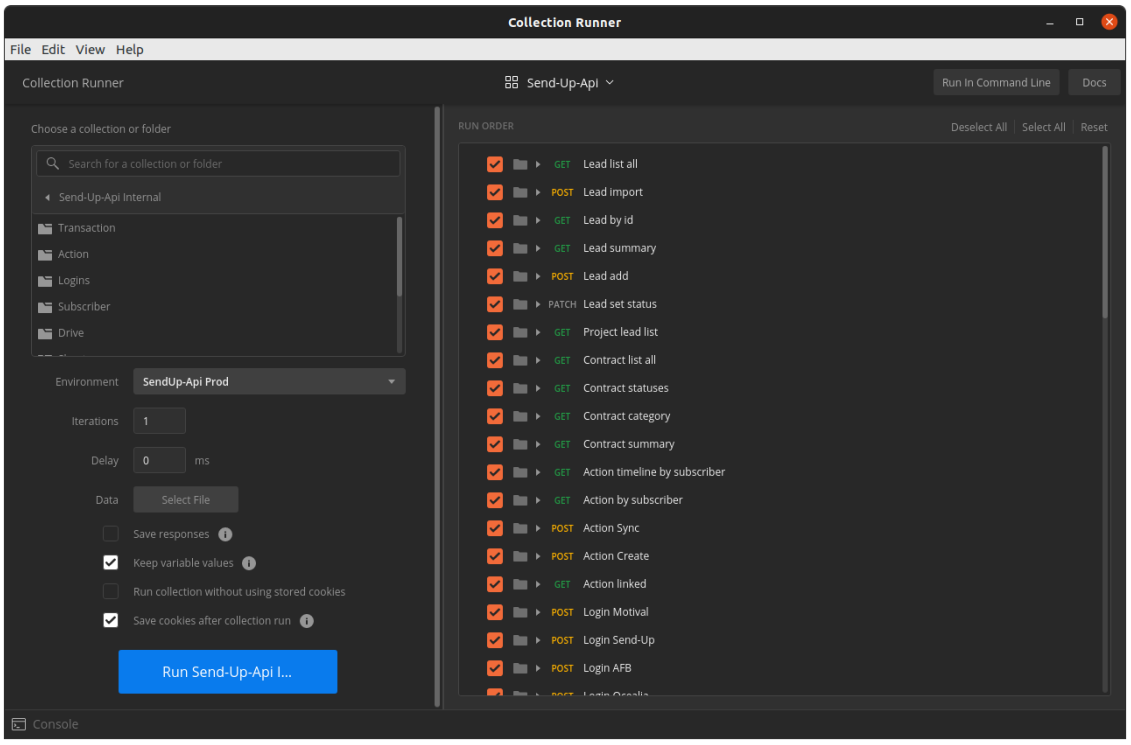

FIGURE 7 – Exemple de collection runner Postman

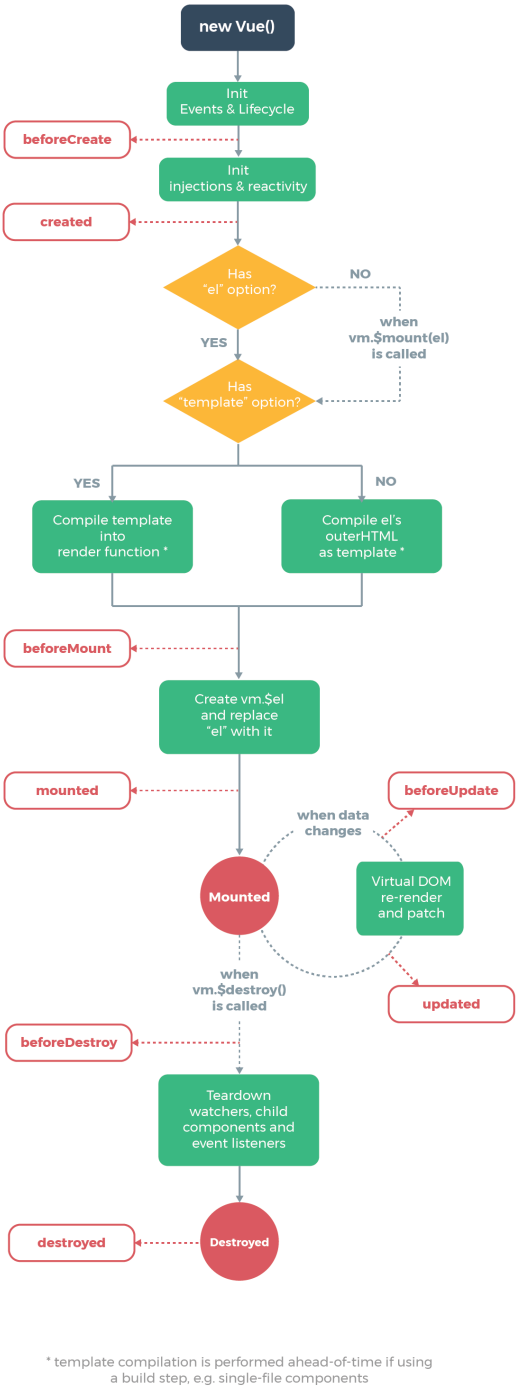

FIGURE 8 – Schéma du cycle de vie - d'un composant vue [10]

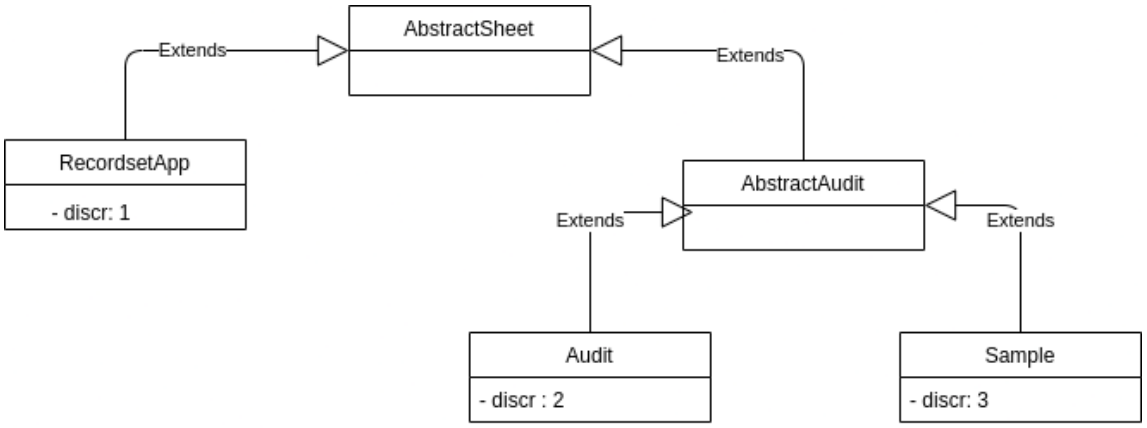

FIGURE 9 – Architecture des recordset

```
{
    "Cahier des charge" : {
        "Type d'activité" : {
            "Produit" : {
                "prélèvement": {
                    ... JSF ...
                },
                "analyse": {
                    "Type d'analyse": {
                        .... JSF ...
                    }
                }
            }
        }
    }
}
```

FIGURE 10 – Architecture du schema JSON

Références

[1] Cirrusware. Send-up. http://www.send-up.net/.

[2] Symfony. Symfony. http://symfony.com/doc/current/index.html.

[3] Doctrine Project. Doctrine. https://www.doctrine-project.org/projects/doctrine-orm/en/2.9/index.html.

[4] Vue. Vue. https://vuejs.org/.

[5] Vue. Slots component. https://vuejs.org/v2/guide/components-slots.html#Slot-Content.

[6] koumoul. Koumoul vjsf. https://koumoul-dev.github.io/vuetify-jsonschema-form/latest/.

[7] json schema org. Json schema. https://json-schema.org/.

[8] Vue. Form input binding. https://json-schema.org/.

[9] Vue. Vue.nexttick. https://vuejs.org/v2/api/#Vue-nextTick.

[10] Vue. The vue instance. https://vuejs.org/v2/guide/instance.html.

[11] Red Hat. What is a rest api ? https://www.redhat.com/fr/topics/api/what-is-a-rest-api.

[12] Vutify. Vuetify. https://vuetifyjs.com/en/.