



MASTER 2 INFORMATIQUE : GÉNIE LOGICIEL

Rapport de stage Cirrusware

Professeur : Laurent Réveillère

Maitre de stage : Vincent Jaulin

Auteur :

Alexandre Erard

24 août 2021

1 Remerciements

Je souhaite remercier toute l'équipe de Cirrusware pour m'avoir accompagné sur les **s** **projet** et pour m'avoir aidé à développer mes compétences techniques et humaines.

merci **Un grand** à Vincent Jaulin et Pascal Tison pour m'avoir pris **et** stage et m'avoir **en** permis d'être sur des projets stratégique pour l'entreprise.

Merci au corps enseignant de l'Université Bordeaux, pour toute l'aide qu'ils m'ont apporté tout au long de mes **étude**. **s**

Merci à toutes les personnes m'ayant permis d'écrire ce rapport. **Au** personnes qui **aux** m'ont relus, en particulier mes amis.

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 pans de l'informatique comme par exemple le développement d'applications mobiles natives, l'intelligence artificielle, la gestion des bases de données et plus largement le Big Data, le développement web. Celui qui m'a le plus passionné est l'architecture logiciel, c'est-à-dire construire des applications scalables et maintenables tout en restant performantes. Pour moi il prend tout son sens dans le développement d'applications web qui ont pour obligation d'optimiser leurs performances, ainsi que leur consommation de ressources 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

1	Remerciments	1
2	Avant propos	2
3	Contexte du stage	4
3.1	Cirrusware	4
3.2	La plateforme Send Up	4
3.3	Le XRM	5
4	Outils et technologies utilisés	6
4.1	Outils	6
4.1.1	Monday	6
4.1.2	Google Workspace	6
4.1.3	Git et GitKraken	7
4.1.4	PHPStorm	8
4.1.5	Postman	8
4.2	Technologies	8
4.2.1	Symfony	10
4.2.2	VueJS	10
5	Missions réalisées	13
5.1	Widget pour le studio Digital	13
5.1.1	Serializations des données des formulaires	13
5.1.2	Compréhension de l'API et implémentation des requêtes	14
5.1.3	Création des widgets	14
5.2	Reprise d'un projet XRM	15
5.2.1	Fonctionnalités	15
5.2.2	Analyse de l'existant	16
5.2.3	Les formulaires des prélèvement	17
5.2.4	Ajout des écarts	18
5.2.5	Reste à faire	18
6	Conclusion	19
7	Anexes	20

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 sein des entreprises ainsi que de **faciliter** la gestion des relations entre clients et contacts.

faciliter

Fondée en 2013, Cirrusware a su convaincre de grands groupes français tels que *RANDSTAD, SAFRAN, MAÏSADOUR, OCEALIA, CARREFOUR*, ou encore *INTERCONTINENTAL - grand hôtel de Bordeaux*.

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

Cirrusware développe 2 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 faciliter leur communication marketing auprès de leurs différents contacts. Les principales fonctionnalités de la plateforme sont :

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

En plus de campagne Emailing **personaliser** en fonction des destinataires. La plate-forme possède également un éditeur PDF afin de réaliser des campagnes papiers.

Le Studio digitale est lui un éditeur WYSIWYG¹ de page web. Il permet en glissant simplement des blocs paramétrables afin d'atteindre le résultat souhaité. Il permet aussi d'accéder facilement au code source de la page afin d'ajouter des fonctionnalités

1. What you see is what you get

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

L'automation lui permet d'automatiser les tâches les plus répétitives de la communication marketing, comme l'envoie d'une campagne 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'accompagner ses clients dans leurs différents projets. Allant de la création de campagnes efficaces à l'intégration de sites web. L'entreprise propose aussi des formations afin de permettre à leurs clients d'être autonomes dans l'utilisation des fonctionnalités les plus poussées de la plateforme.

3.3 Le XRM

Le XRM est un outil permettant d'optimiser la relations entre clients, contacts et fournisseurs. Il est totalement intégré à la plateforme Send-Up.

Il permet :

- une gestion de portefeuille clients.
- une gestion des actions (rendez-vous, notes, comptes-rendus, emails, sms)
- une gestion de diverses opportunités.
- utilisation Hors ligne
- une adaptation au besoin du client.

Le XRM est un outil très modulable qui permet donc pour chaque nouveau client d'intégrer des fonctionnalités uniques. Comme la visualisation de statistiques particulières, ou encore la création et la gestion d'audit².

2. "L'audit est une expertise professionnelle effectuée par un agent compétent et indépendant aboutissant à un jugement par rapport à une norme sur les états financiers, le contrôle interne, l'organisation, la procédure, ou une opération quelconque d'une entité." - *Wikipedia* [2]

4 Outils et technologies utilisés

Lors de mon arrivée dans l'entreprise, j'ai dû 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* [3] pour communiquer, *VSCode* [4] pour développer, et *GitHub Issue* [5] ou *Trello* [6] pour gérer les projets. Durant mon stage j'ai du utiliser des outils plus professionnels 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 [7] 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 quelqu'un, elle possède un état (en cours, terminée, 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.

	Classement	Demandé le	Demandeur	Dev	Statut	Temps estimé	Terminé le	
[Actions] - modale revoir les couleurs des boutons	★★★★★	jull. 26			MA: Fait			
[Drive] - Documents // Contrats et mandats	★★★★★	jull. 26			Clients			
[Timeline d'un client] - on peut voir que des actions liées avec l'utilisateur	★★★★★	jull. 26			1.A faire			
[Modale Email] - L'envoi d'email ne fonctionne pas	★★★★★	jull. 29			6. Push PROD			
[CSS] - Dans les champs de recherche faire apparaître entièrement le mot "Rechercher"	★★★★★	jull. 26			4. Non validé	0,5h		
[Modale] - Agenda // Champ non obligatoire pour le message	★★★★★	jull. 26			3. Push Pre-PROD	0,5h		
[JDS] - Faire la géolocalisation	★★★★★	jull. 26			3. Push Pre-PROD	1h		
[Clients/Contacts] - sauvegarder la recherche + la page	★★★★★	jull. 26			3. Push Pre-PROD	1h		
[Fichier client] - bloc drive // fichier n'apparait pas	★★★★★	jull. 26			2.Beaucoups	2h		
[Fichier client] - Bloc drive // Suppression impossible	★★★★★	jull. 26			2.En cours	2h		
[CRM AVAL] - Voir le mode hors connexion	★★★★★	jull. 26			2.En cours	2h		
[Responsive] - A faire VO	★★★★★	jull. 26			1.A faire			
[Doc] - information sur les droits	★★★★★	jull. 26			1.A faire			
[Modales] - quand on clique sur une action, faire apparaître directement l'édition et pas l...	★★★★★	jull. 29			1.A faire	0,5h		
[Automation] - Revoir le process pour la géoloc	★★★★★	jull. 26			1.A faire	1h		
[Blason] - Fiche client // Header block paramétrage photo	★★★★★	jull. 26			1.A faire	1h		
[Abonné] - Enlever Groups // Vincent	★★★★★	jull. 26			1.A faire	0,5h		

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 en tenant compte de la situation sanitaire actuelle, l'application Google Meet est utilisée comme application de visio conférence, pour les diverses réunions et les rendez-vous avec les clients.

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 [8] 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 branches, etc..).

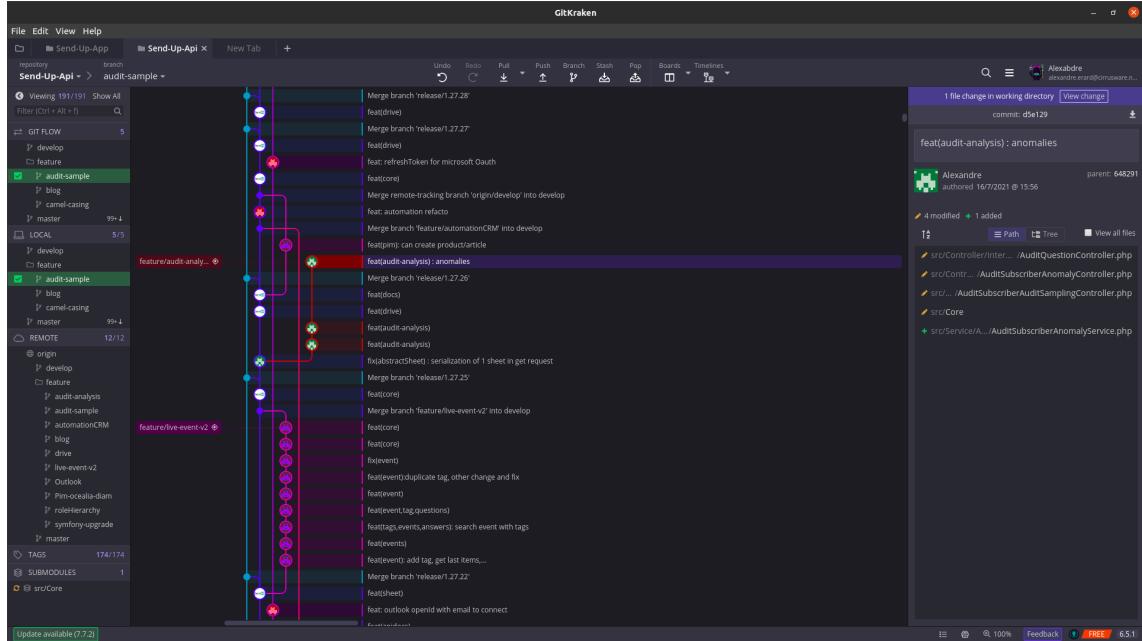


FIGURE 2 – Interface de GitKraken

De plus il permet une totale intégration des git flow. C'est à dire créer facilement une branche par nouvelle fonctionnalités, et simplement la fusioner avec la branche de développement lorsque celle-ci est terminée. Les git flow permettent également une meilleure gestion des HotFix, débogage effectuer en production, et de mise en production mais je ne les ai pas utilisées lors de mon stage.

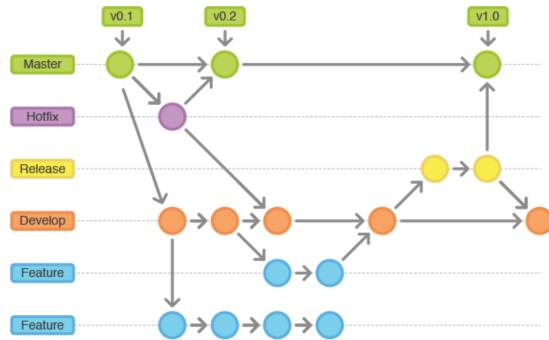


FIGURE 3 – Déroulement d'un git flow

4.1.4 PHPStorm

PHPStorm [9], créé par *JETBRAINS* [10], est l'un des IDE les plus complets pour le développement web. A l'aide de ses nombreux pluggins il permet une totale prise en charge des framework comme *Symfony* [11] et *VueJS* [12]. Il permet de facilement ~~de~~ refactoriser du code ou encore de créer de nouveaux snippets, portions de code paramétrables, afin de faciliter les phases de développement. Malgré ses nombreux points positifs, il est dur à prendre en main au début mais par chance au cours de mes études supérieures j'ai été amené à utiliser *InteliJ IDEA* [13], ou encore *Android studio* [14] qui sont deux IDE de *JetBrain* qui ont exactement la même structure.

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* [15] afin de facilement créer nos requêtes ainsi que des tests associés. L'outil permet de faire hériter des tests à des sous collections ce qui nous permet de gagner du temps lorsque certains tests 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* (Figure 7) qui permet d'agencer l'ordre des requêtes, ainsi que de les lancer n fois, à un intervalle de temps y .

4.2 Technologies

Historiquement la plateforme Send-Up a été développée sur une stack *Symfony 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

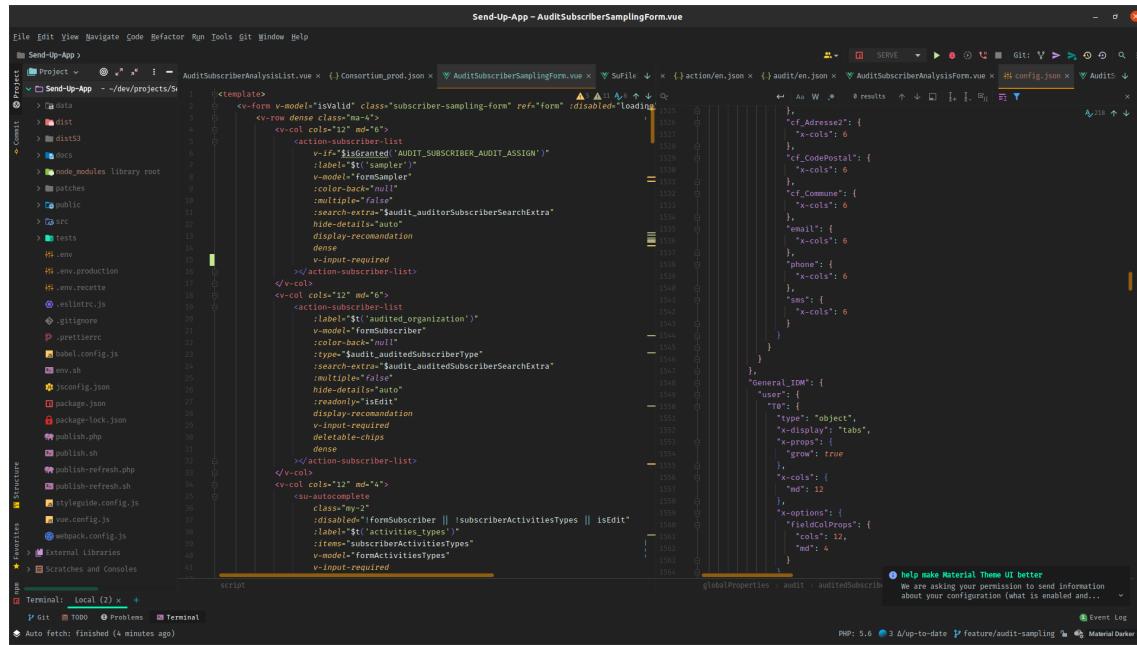


FIGURE 4 – PHPStorm

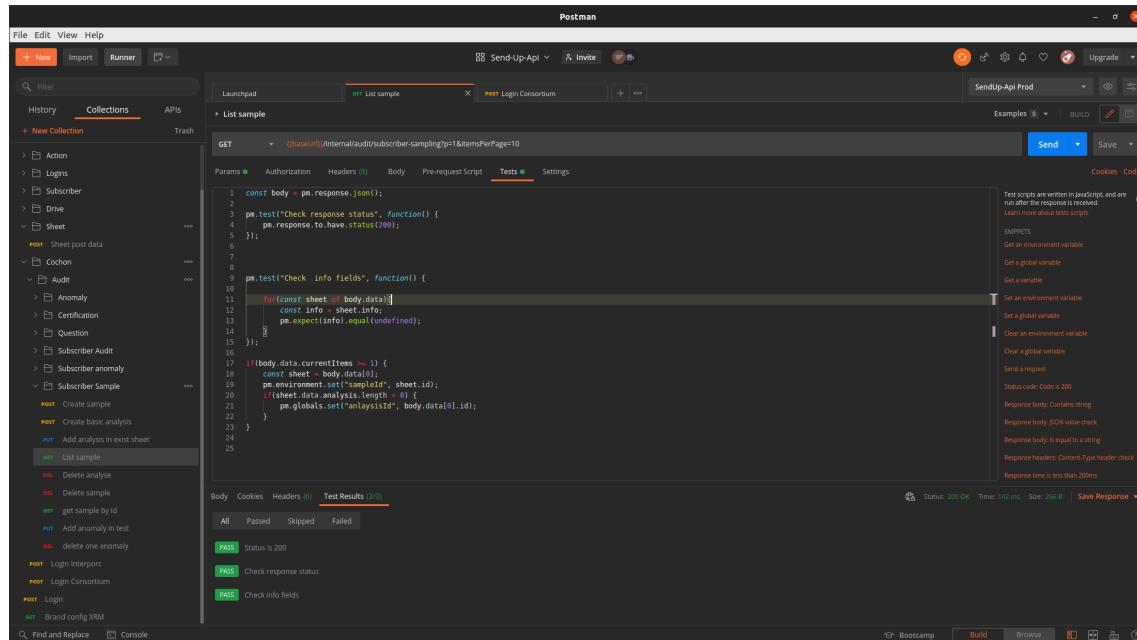


FIGURE 5 – Exemple de test Postman

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", l'héritage des fonctionnalités existante, afin de garder une rétro-compatibilité mais aussi afin de faciliter son usage auprès des clients. Voyons plus en détails les technologies choisies par l'entreprise à savoir Symfony et VueJS.

4.2.1 Symfony

Symfony [11] 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, à ce jour il existe plus de 4 563 bundles. Et énormément de réponses sur les forums de développeur. Il intègre également des outils qui sont extrêmement pratiques dans la gestion des données comme par exemple **Doctrine** [16] 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'interrogation de la base de données nommé DQL. Ce langage permet de rendre plus lisible les requêtes ainsi que l'utilisation de variables. D'autres outils sont également présents comme l'injecteur de dépendances ou encore des outils de debug disponibles en environnement de développement.

4.2.2 VueJS

Pour remplacer la partie Jquery de Send-Up le framework VueJs [12] a été choisi par l'équipe. Il s'agit d'un framework open-source développé à l'origine par Evan You en 2013. Ce framework facilite la création d'interfaces graphiques et de Single Page Application (SPA). Pour présenter rapidement le framework voici un exemple de composant basique.

Un fichier .vue est composé de 3 éléments 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. On peut l'utiliser de deux manière différentes au sein d'un composant. En tant que style générale qui va atteindre le composant et ses **enfant**, ou en porté restreinte qui ne va atteindre que le composant.

La balise script est l'endroit où 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

```
<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() {
      this.count++;
    },
  },
  mounted() {
    console.log("HelloWorld component mounted");
  },
}
</script>
<style></style>
```

FIGURE 6 – Exemple de composant VueJS

son rendu, les paramètres (appelés *props*) que son parent peut lui passer, ici on attend un paramètre *msg* de type *String*. On retrouve également l'état du composant avec toutes les données utilisées. Dans l'exemple ci-dessous, on a un compteur et la chaîne de caractère *hello*, les méthodes qui vont déterminer la logique, les propriétés calculées du composant. Les propriétés calculées peuvent être appelées 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 parlera plus en détails plus tard.

Voici le rendu de ce composant appelé avec *msg = Friends* :

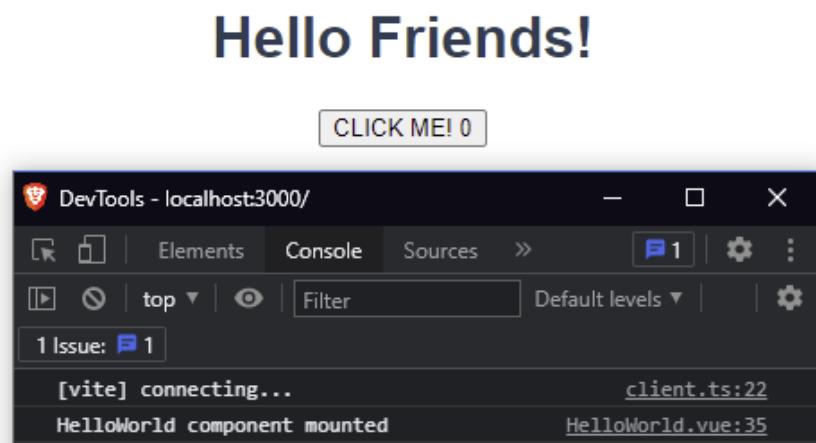


FIGURE 7 – Exemple de composant

On y retrouve également les divers événements du cycle de vie auquel on peut se brancher pour effectuer certains traitements spécifiques, comme faire des requêtes d'API. Pour plus de détails voir l'annexes 7

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-Comté*.

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 aux 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 vues 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 a été d'adapter l'enregistrement des formulaires du studio qui jusque là ne nous permettaient pas d'effectuer simplement des statistiques.

5.1.1 Serializations des données des formulaires

Afin de récolter les données des formulaires, le studio digital passe par un serializer, un outil qui permet de passer les données au travers d'un flux. Celui ci va enregistrer dans un objet json les valeurs des différents champs associés à leurs id. Ce système présente plusieurs avantages comme la facilité et la rapidité d'implémentation. Malheureusement 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.

Celui-ci

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 s directement dans le code html de la page, le DOM³. Les informations qui je vais récupérer sont celles, qui permetrons le calcule de la note, diverses meta-données par rapport au question comme le label, l'id du champ concerné, le résultat obtenu à la question, etc..

récupérées Dès que ces informations sont récupérées il ne reste plus qu'à 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'implémentation actuelle du serveur Send-Up, la partie gérant les *Recordset* - table MySQL comportant un champ info destiné à stocker du JSON - est gérée par une partie de l'API dont les routes pointant vers cette partie sont générées automatiquement à partir du nom de la méthode définie dans le contrôleur. Exemple la méthode *postRecordset* est appelée lors d'une requête POST sur la route */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** étaient acceptés pour la requête existante. Une requête qui permettait de lister tous les Recordset liés 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 champ personnalisé nommé "entreprise" qui correspond au vrai nom de son entreprise. Grâce à cela on a juste à rajouter un paramètre à la requête précédente afin d'avoir une vue réduite sur les données. Pour récupérer les données d'un seul utilisateur, il nous suffit juste de filtre sur l'id de l'utilisateur.

5.1.3 Crédit des widgets

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

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

Après avoir développé certains d'entre eux, il s'est avéré que certains traitements se répéter ou se ressembler fortement. Or VueJS intègre des *Mixins*, ces éléments

3. Document Object Model

possèdent les mêmes caractéristiques dans leurs balises script qu'un composant vueJS à classique. Mais ce dernier a la possibilité d'être intégré à l'intérieur de composants (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 spécial afin de faciliter la lisibilité et la maintenance du code.

Pour voir les résultats de ces widgets, voir les figures : 7, 7, 7, 7, 7

5.2 Reprise d'un projet XRM virgule

Suite au départ d'un développeur de l'équipe. On m'a placé sur la reprise de son projet qui consiste à créer un XRM permettant tous les points de contrôle concernant le *Consortium du Jambon de Bayonne et Interporc Franche-Comté*. 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. s

Toute la partie audit a été réalisée par mon prédécesseur, il me reste donc à implémenter la partie Analyse et prélèvement.

5.2.1 Fonctionnalités

Tout d'abord passons en revue les fonctionnalités principales souhaité par les clients.

s Ici nous avons 2 clients différents avec des besoins différents.

- En tant que Consortium, je souhaite pouvoir créer des fiches prélèvements. 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 analyses à un prélèvement. Chaque analyse possède un type dépendant des champs sélectionnés dans la fiche prélèvement. Ceci conditionera les champs spécifique de l'analyse.
- En tant que Interporc, je souhaite pouvoir créer des fiches analyse. Les champs spécifiques dépendent uniquement du type de produit sélectionner : selectionné Saucisse, Lisier, Lactoserum, Sel.
- En tant qu'Interporc et Consortium, je souhaite pouvoir lever des écarts à chaque analyses si besoin est.
- En tant qu'Interporc et Consortium, je souhaite pouvoir téléverser les résultats des laboratoires et les associer à une analyse.

5.2.2 Analyse de l'existant

- s Après avoir discuter avec mon chef des **projet**, et mon maître de stage, on a **de-déterminé** qu'un audit et un prélèvement **se ressembler** énormément. En effet chacun **se ressemblaient est relié** et **relier** à un auditeur/prélevEUR, une personne audité, un ou plusieurs cahier des charges et un ou plusieurs type d'activités.
- s 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 précédemment. La grande différence est que cette dernière est **gérée** par une classe abstraite nommé *AbstractSheet*, qui définit que le champs info possède au minimum un champ builder, data et files. Chaque entité voulant être **stocker** dans la table *Recordset_app* doit être hérité de cette classe et donc respecter ce schéma. Pour différencier nos sous classe nous **faisons** appel **une fonction** de doctrine appelée le discriminant. Il permet d'associer un entier à une classe. À ce jour nous avons 2 **classe** héritant de **AbstractSheet** :
1. *Recordset_app* : qui correspond à toute les fiches standard.
 2. *SubscriberAudit* : qui correspond au audit créer par nos subscriber.
- l'existant Voilà pour **l'existant** au niveau de l'API. Voyons maintenant comment à était construit les audit côté interface utilisateur afin de pouvoir garder une cohérence dans l'application et de pouvoir réutiliser au maximum les composants déjà créé. créés
- Premièrement commençons par la liste des audits créés. 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.
 - *AuditSubscriberList* : qui va récupérer les données du serveur et les passer au composant enfant avec les en-têtes du **tableaux** 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 [17] de Vue. Un slot permet de réservé une place à un composant ou autre template html qui sera injecter depuis un composant parent. Un composant parent peut lui même définir qu'un slot enfant comprend un autre slot. Ainsi Cela permet une total personnalisation 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 également accompagné d'information complémentaire comme sa gravité ou encore des commentaires associés. Or la création d'écart à déjà était créer dans la partie audit, et les prélevement ont été construit de sorte à respecter la même interface, le même contrat que les audit. Cela rend donc possible la réutilisation du composant. Ceci nécessitera de refactorer une partie de l'api car certaines options devenaient optionnel, selon le context audit ou prélevement. Pour plus de précision, voir l'architectture des recordset7. On repetera cette architectture également pour les contrôleur. contrôleurs

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

5.2.3 Les formulaires des prélevement

Afin de nous indiqué leur besoins, le Consortium nous a envoyé un fichier excel correspondant à toutes les conditions possible pour les formulaire de prélevement et d'analyse. Une configuration de prélevement 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 prélevement. La liste des produits nous a été fournis dans le fichier excel et dépend essentiellement du type d'activité et du cahier des charges. Pour contruire construire facilement des interfaces de formulaire, nous utilisons une librairie nommé Koumoul [18]. Koumoul est une implémentation Vue/Vuetify de la norme JSON Schema Form [19]. Cette norme permet de facilement construire des formulaire à 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 permettant de parser le fichier qu'on nous a mis à 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 selon celons la configuration choisie.

Après avoir construit cela divers problème lié à la gestion des variable de koumoul

virgule

s sont **survenu**. Dans vue, une des notions les plus importante est le **v-model** [20]. Il à s'agit d'une variable **passer** en props par un parents à un composant enfant qui aura la particularité d'être modifiable lors de l'émission de l'**événement** input par l'enfant. évènement Or lorsque je **souhaiter envoyé** le résultat de mon formulaire au serveur celui-ci ne souhaitais s'était pas mis à jour. La **seul** solution pour résoudre ce problème de synchronisation envoyer est l'appel à la méthode *nextTick* [21] de Vue. Cette méthode permet de différer l'appel d'une fonction au prochain cycle de mise à jour du DOM.

passé

seule

Ici je vous présente uniquement le cas du Consortium car il s'agit du plus complexe s 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 éviter pour Interporc car leur composition sont trop différente. Afin d'**éviter a nouveaux** la créée redondance de code **je créer** donc un Mixins pour centraliser la logique comumne.

à nouveau

5.2.4 Ajout des écarts

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

er

clairs

5.2.5 Reste à faire

À l'heure actuelle 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 fonctionnalités comme certains 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 fonctionnalités spécifique à développer.

testée

aux

6 Conclusion

deux

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 connaissances entre développeur. En effet l'une des principales difficulté lors de ce projet à était la reprise d'un code qui sur certains aspect mal documenter.

reprise

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 comprehension mais également d'integrer des tests à l'aide de postman. Ces tests me permettaient intégrer facilement de savoir si une fonctionnalité 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.

s
passé

amené
Point + maj
s
le plus possible
s
aspects était mal documenté.

s
est
s
impactée

7 Anexes

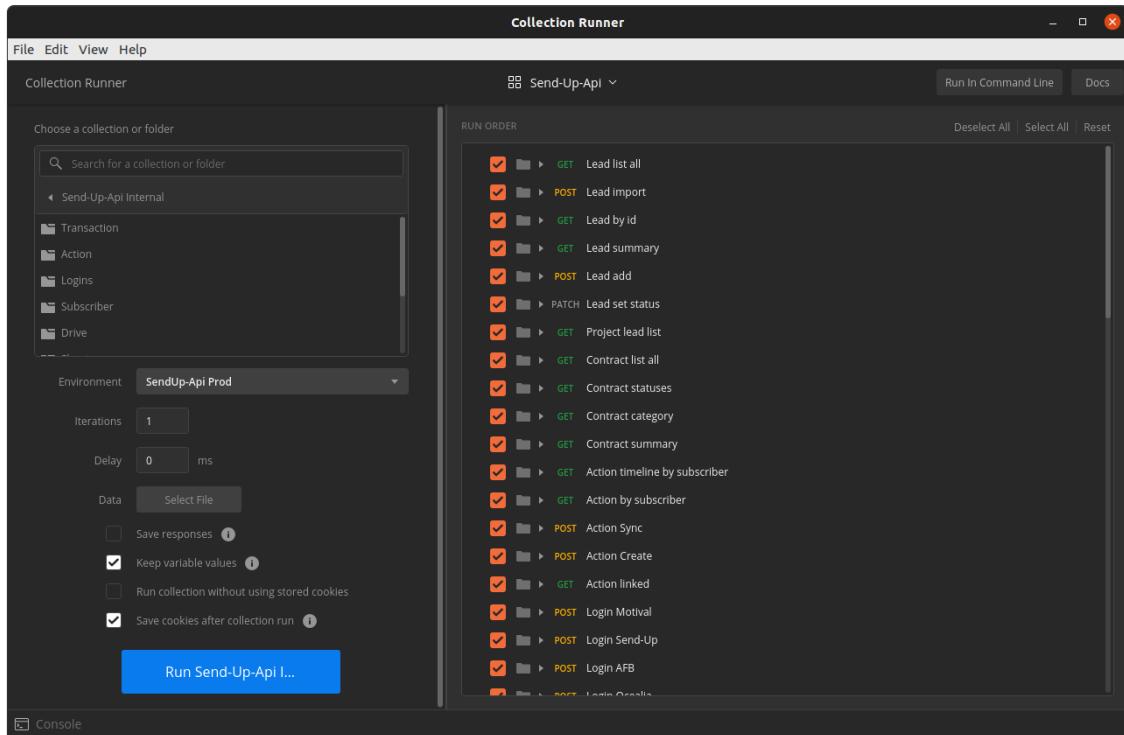
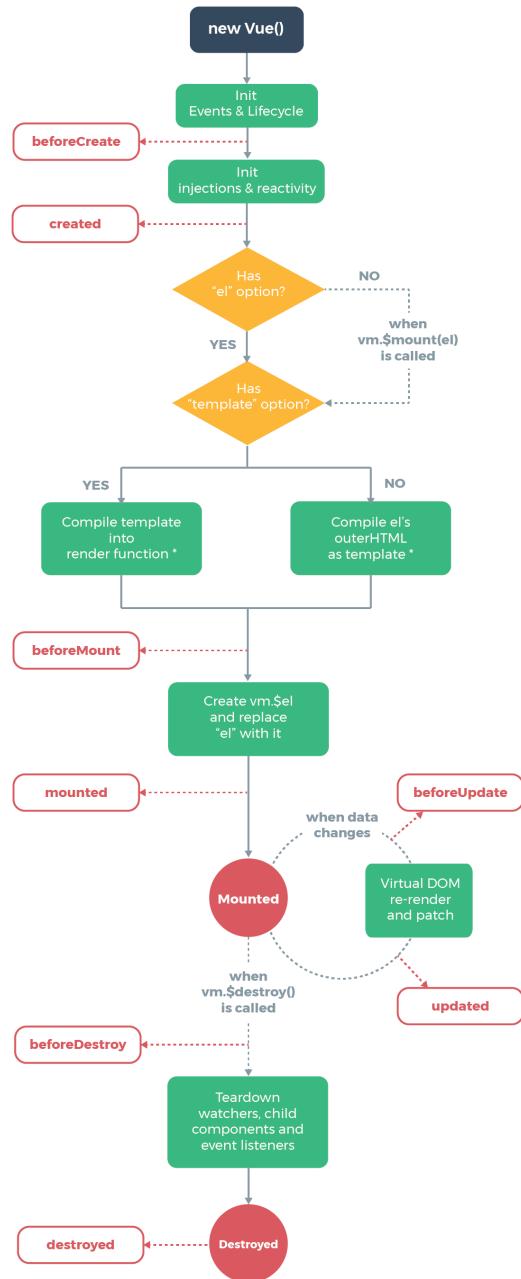


FIGURE 8 – Exemple de collection runner Postman



* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

FIGURE 9 – Schéma du cycle de vie - d'un composant vue [22]

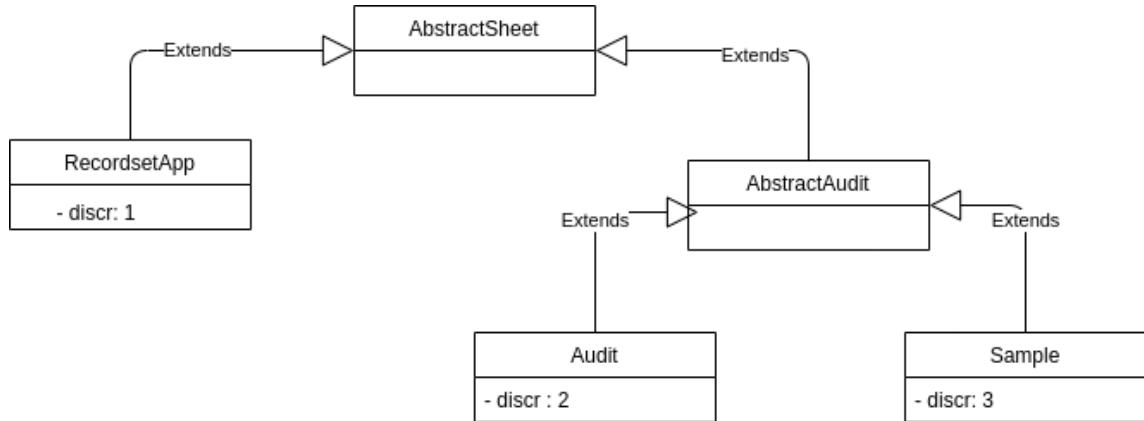


FIGURE 10 – Architecture des recordset

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

FIGURE 11 – Architecture du schéma JSON

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

FIGURE 12 – Statistiques individuelles

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

FIGURE 13 – Statistiques globales

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

FIGURE 14 – Réponses au questionnaire\$

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

FIGURE 15 – Questionnaires de satisfaction

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

FIGURE 16 – Statistiques du questionnaire de satisfaction

Références

- [1] Cirrusware. Send-up. <http://www.send-up.net/>.
- [2] Audit. <https://fr.wikipedia.org/wiki/Audit>.
- [3] Discord. Discord. <https://discordapp.com/>.
- [4] Microsoft. Vscode. <https://code.visualstudio.com/>.
- [5] Microsoft. github. <https://github.com/>.
- [6] Atlassian. Trello. <https://trello.com/home>.
- [7] Monday. Monday. <https://monday.com/>.
- [8] GitKraken. Gitkraken. <https://www.gitkraken.com/>.
- [9] Jetbrains. PhpStorm. <https://www.jetbrains.com/phpstorm/>.
- [10] Jetbrains. JetBrains. <https://www.jetbrains.com/>.
- [11] Symfony. Symfony. <http://symfony.com/doc/current/index.html>.
- [12] Vue. Vue. <https://vuejs.org/>.
- [13] Jetbrains. Intelij idea. <https://www.jetbrains.com/fr-fr/idea/>.
- [14] Google. Android studio. https://developer.android.com/studio?hl=fr&gclid=EAIAIQobChMI246lxPHH8gIVjP1RCh0GvwUnEAYASAAEgJE3PD_BwE&gclsrc=aw.ds.
- [15] Postman Team. Postman. <https://www.postman.com/>.
- [16] Doctrine Project. Doctrine. <https://www.doctrine-project.org/projects/doctrine-orm/en/2.9/index.html>.
- [17] Vue. Slots component. <https://vuejs.org/v2/guide/components-slots.html#Slot-Content>.
- [18] koumoul. Koumoul vjsf. <https://koumoul-dev.github.io/vuetify-jsonschema-form/latest/>.
- [19] json schema org. Json schema. <https://json-schema.org/>.
- [20] Vue. Form input binding. <https://json-schema.org/>.
- [21] Vue. Vue.nexttick. <https://vuejs.org/v2/api/#Vue-nextTick>.
- [22] Vue. The vue instance. <https://vuejs.org/v2/guide/instance.html>.
- [23] Red Hat. What is a rest api? <https://www.redhat.com/fr/topics/api/what-is-a-rest-api>.
- [24] Vutify. Vuetify. <https://vuetifyjs.com/en/>.
- [25] Packagist. Bundle symfony. <https://packagist.org/?query=symfony&tags=symfony>.