

Table des matières

0.1	Introduction	5
1	Présentation de l'Entreprise	6
1.1	Ingima	6
1.2	Ingima Lab	7
1.2.1	Le directeur de Lab	8
1.2.2	Chercheur	9
1.2.3	Développeurs/Stagiaires	9
1.2.4	Consultant	10
1.2.5	Fonctionnement du Lab	10
2	Le Stage	11
2.1	Description de mon rôle	11
2.2	Projet PackDiff	11
2.2.1	Description	11
2.3	Architecture	14
2.3.1	Front-End	16
2.3.2	Back-End	21
2.3.3	Core	24
2.4	développement de PackDiff	25
2.4.1	Gestion de code	25
2.4.2	Workflow	26
2.4.3	Processus de Livraison	29
2.4.4	Rôle et tickets	30
2.5	Conclusion	32

Glossaire

Front-End La partie Front-End d'une application concerne le développement de l'interface utilisateur et de toutes les interactions que pourra avoir celui-ci. En développement web, les langages HTML, CSS et JavaScript sont utilisés.

Back-End Le Back-End d'une application va s'occuper de toutes les tâches que l'utilisateur ne voit pas. Cela peut être la communication avec une base de données ou même le traitement de données.

Full-stack Un développeur Full-stack est une personne capable de travailler sur toutes les parties d'un logiciel.

DevOps Le DevOps (développement et opérations) est une manière de développer un produit en intégrant un déploiement continu. Cela permet notamment d'automatiser des phases de tests et de déploiement.

Framework un framework désigne un ensemble de composants de logiciel qui servent à créer les fondations ainsi que la structure d'un logiciel.

API Une API (Application Programming Interface) est un ensemble de classes, de fonctions, de données qui sert de d'interface par laquelle un logiciel offre des services à d'autres logiciels.

Sprint Un sprint en développement logiciel est une période de temps limitée dans laquelle l'équipe se fixe des objectifs à réaliser.

Git Git est un logiciel de gestion de versions décentralisé. Il fonctionne par système de « branches », qui correspondent chacune à une version différente d'un projet. La branche principale d'un projet est généralement appelée master.

Chapitre 1

Présentation de l'Entreprise

1.1 Ingima

Fondé en 2012, Ingima est une ESN(Entreprise de Services du Numérique) spécialisée dans le conseil informatique et le management de projets. Sa principale activité est de recruter des talents spécialisés dans les métiers de l'informatique afin de vendre leurs services à ses clients. Le nom « Ingima » est né de la fusion d ' « Ingéniosité » et « Imagination ». le groupe comporte aujourd'hui près de 250 collaborateurs répartis entre la France, Belgique et Israël.

La stratégie du groupe tend à développer ses activités dans les domaines infrastructures et sécurité, et web et digital. Des investissements stratégiques ont également lieu dans e big Data,l'Inot(Internet des Objets), la Vision par Ordinateur et la Réalité Virtuelle/Réalité Augmentée, qui sont considérés comme des domaines porteurs de grandes transformations. Les principaux secteur d'intervention sont la défense (39 Les équipes sont majoritairement composées de RH et Business Manger. Les Business Manager sont là pour garder un portefeuille de clients, avec lesquels ils vont démarcher différents projets. Lorsqu'un client émet le besoin d'avoir un développeur en plus, les Business travailleront avec les RH pour trouver un talent correspondant à leurs attentes.

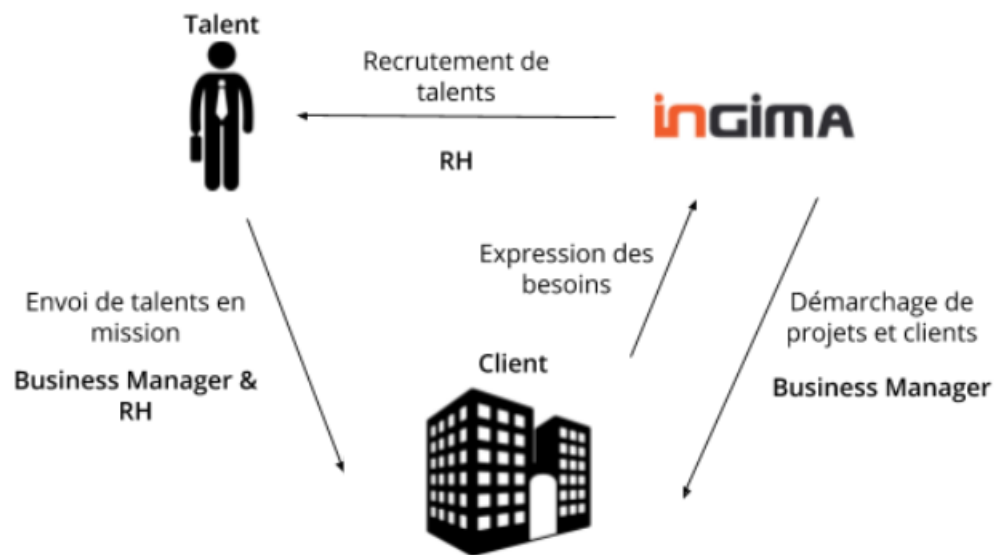


FIGURE 1.1 – Fonctionnement Ingima France

1.2 Ingima Lab

En Janvier 2017, Ingima a crée un pole R&D et innovation nommé Ingima Lab afin de pouvoir être proactif sur les demandes et challenges technologiques du marché actuel. Depuis Septembre 2019, le Lab est dirigé par Antonin Goude, mon tuteur de stage. c'est dans cette structure que j'ai travaillé pendant mes 4 mois de stage. Le Lab est composé d'une dizaine de personnes environ et est organisé comme ceci :

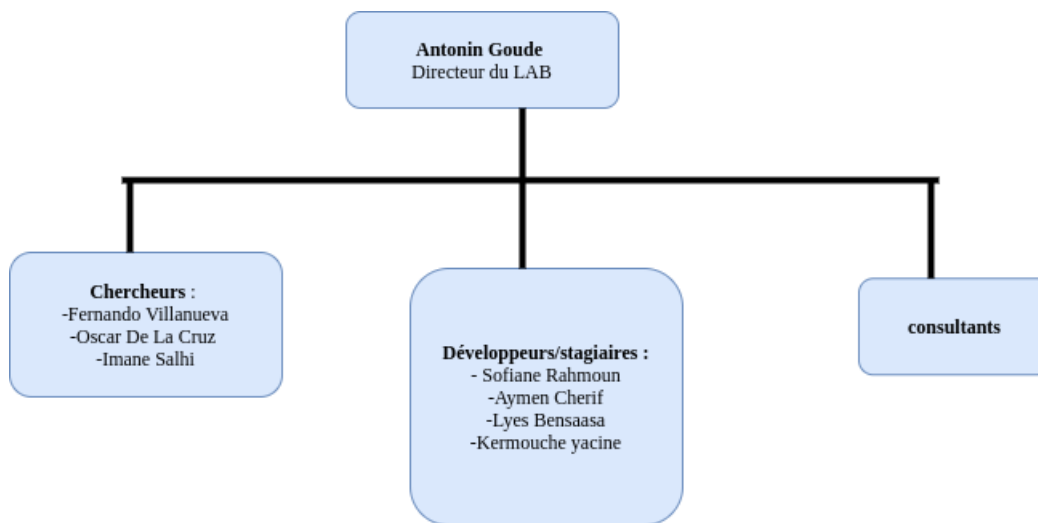


FIGURE 1.2 – Organisation du Lab

1.2.1 Le directeur de Lab

Le directeur du Lab a pour rôle d'organiser toutes les opérations et la stratégie du Lab. Son rôle est de gérer les différents projets à haut niveau et d'en démarcher de nouveaux. Pour cela, Les business manager peuvent aider afin de faire profiter de leur portefeuille ou alors le Lab peut démarcher ses propres clients.

1.2.2 Chercheur

Les chercheurs sont l'ADN même du Lab, En effet, c'est en fonction de leurs domaines d'expertise et leurs compétences que le Lab va pouvoir définir ses sujets de prédilection et ainsi définir sa feuille de route et sa stratégie. Les projets dépendent donc de ces personnes, Aujourd'hui', le Lab se concentre sur les sujets suivants :

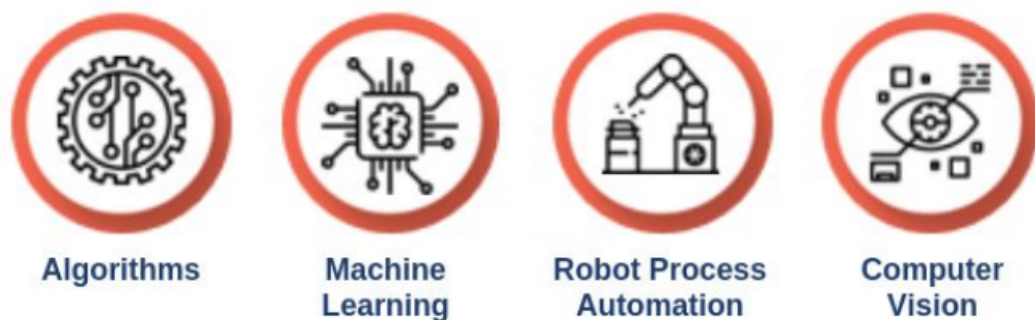


FIGURE 1.3 – Domaine du Lab

Afin de pouvoir réaliser des projets qui peuvent être industrialisés et proposés à des clients, les chercheurs doivent collaborer avec des développeurs pour créer des applications complètes.

1.2.3 Développeurs/Stagiaires

Les développeurs sont majoritairement des personnes qui viennent au laboratoire pour une durée déterminée, En effet, Cela dépend du besoin des différents projets en cours, Par exemple, si un projet machine learning demande le développement d'une application web, le Lab recrutera des personnes spécialisées dans ce domaine pour la durées de celui-ci. Les stagiaires peuvent avoir différents rôles : un rôle de développeur d'application ou un rôle de chercheurs en mettant en avant des compétences comme l'intelligence artificielle, le machine learning etc ...

1.2.4 Consultant

Les consultants ne font pas partie intégrante du Lab. En effet, il s'agit des personnes envoyées en missions chez les clients. Cependant, entre deux missions un consultant peut être amené à aider le Lab sur certains sujets, de part son expertise technique.

1.2.5 Fonctionnement du Lab

Pour pouvoir fonctionner, le Lab doit gagner des projets et essayer de les mener jusqu'à l'industrialisation. Pour ce faire, un principe de Proof Of Concept (PoC) est utilisé. Voici la démarche dans son ensemble :

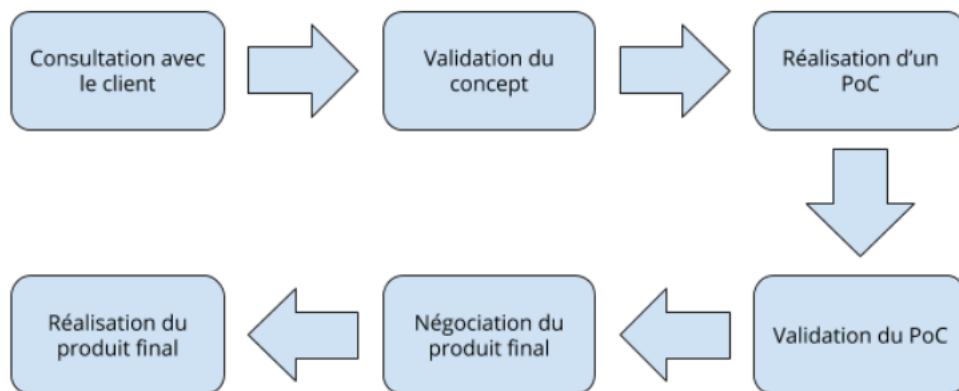


FIGURE 1.4 – Fonctionnement d'un projet du Lab

Un PoC est une version minimale(sans toutes les fonctionnalités) du produit final afin que le client valide l'idée du Lab . Comme les projets sont innovants, cela permet d'être sûr que les deux parties se comprennent bien et ont les même attentes.

Chapitre 2

Le Stage

2.1 Description de mon rôle

Durant mes premiers jours dans l'entreprise mon rôle était de bien comprendre le projet PackDiff qui assez complexe, et d'assister aux réunions avec le client pour que je puisse m'adapter avec le fonctionnement du projet. Par ailleurs, mon rôle était le rôle de développeur Web, À partir du cahier des charges rédigé par le client j'analyse les besoins et je choisis la solution technique afin de satisfaire le besoin dans les meilleurs délais.

Aussi faut faire des tests unitaires afin que les modifications apportées sur le code n'impactent pas le code existant et le bon fonctionnement de l'application.

2.2 Projet PackDiff

2.2.1 Description

Ce projet est celui sur lequel j'ai travaillé tout au long de mon stage. C'est un projet qui était en cours depuis plus de d'un an et demi déjà dans le Lab. Je suis donc arrivé à une étape bien avancée de la vie du produit. PackDiff est un projet destiné à la grande distribution. En effet, le client est un des plus grands acteurs Français de ce secteur. **Contexte** aujourd'hui le client vend des produits sous sa propre marque. Pour cela trois acteurs différents sont impliqués de la création du produit :

-
- Le producteur, qui s'occupe du produit en lui-même. ;
 - Une plateforme qui s'occupe de l'emballage du paquet, c'est à dire aussi bien design que textes, normes, etc. ;
 - Une plateforme qui s'occupe de la traduction des textes créés pour l'emballage du produit..

Ces acteurs sont des intervenants extérieurs, il y a donc besoin d'une vérification du travail surtout dans ce domaine dans lequel les normes sont très strictes et les informations se doivent d'être précises. Ce projet a pour but de simplifier et de rendre plus rapide la phase de vérification de l'emballage. Trois types d'employés sont impliqués dans cette vérification : **Les chefs de produits(PM)** : ils sont responsables du marketing du produit et vont donc superviser toute l'opération. **Les responsables qualité(QM)** : ils sont responsables de toutes les parties concernant la loi sur un produit. C'est un travail très important car il permet au client d'éviter d'avoir de gros problèmes quant au respect des normes. **Les responsables emballages (PPC)** : ils sont responsables de l'emballage général du produit et de doivent vérifier que celui corresponde bien à la demande et qu'il contienne toutes les informations.

Voici le processus d'un produit sans l'outil PackDiff :

1. Avant de faire la vérification, il faut définir quels sont les éléments qui doivent être présents sur l'emballage(images et textes) et mes trier en deux catégories : demander par la loi et marketing. 1 PC, 2QM ET 1PPC écrivent toutes les spécifications de ce qui doit se retrouver dans cet emballage. Ces spécifications sont triées par catégories qui sont enregistrées dans un outil interne au client que l'on nommera Alpha.
2. Les spécifications sont envoyées aux traducteurs en fonction des langues dont le paquet a besoin (7 langues possibles).
3. Après validation, les spécifications sont envoyées à la plateforme externe qui s'occupe de l'emballage des produits et qui va mettre toutes les informations demandées sur celui-ci.
4. L'un après l'autre, 1 PC , 1 QM, 1 PPC, font la vérification du produit en regardant chacun si les critères liés a leur métier sont respectés ou non.
5. Le processus recommence au point numéro 2 jusqu'à ce que l'ensemble des métiers ait validé l'emballage.
6. Le produit validé peut partir en supermarché.

Le projet PackDiff vise à améliorer la phase 4 de cette procédure. En effet, C'est un processus qui prend énormément de temps car les employés

doivent comparer des fichiers textes avec PDF mais aussi PDF avec PDF. C'est un vrai travail manuel, mais surtout les employés sont obligés de comparer les éléments qui ne sont pas différents et doivent vérifier l'entière des spécifications à chaque vérification. Ce n'est pas donc pas du tout un process automatisé et le fait que ce soit manuel peut éventuellement apporter des erreurs humaines lors de la validation. Le but était donc d'optimiser le processus en utilisant un outil qui permet la vérification automatique des emballages en fonction des spécifications entrées par les différents métiers.

Besoin Pour ce faire le client a demandé à Ingima Lab de créer une application web utilisable par tous les acteurs de processus (PC, QM, PPC) répondant au problème posé par l'étape 4.

Voici les différentes caractéristiques de base que l'outil est donc censé mettre en place :

- Récupération du fichier des spécifications pour un emballage
- Récupération de l'emballage sous format PDF
- Comparaison des spécifications sous format PDF
- Affichage des différences entre les deux à l'utilisateur

Les fichiers de spécifications seraient de deux types différents :

- **Trame Texte (appelé TT)** C'est ce que contient tout le texte affiché sur l'emballage : dénomination commerciale, composition, nom du produit, etc
- **Tableau de Valeur Nutritionnel (appelé TVN).**
Pour les aliments, un tableau de valeur nutritionnel correspondant à des normes différentes et donc structuré de manière différente. Ce sont les tableaux généralement présentés sous cette forme : ce sont les tableaux généralement présentés sous cette forme :

Comme expliqué précédemment, les différents corps de métier écrivent les spécifications et les stockent grâce à l'outil interne au client appelé dans ce rapport de stage Alpha. Ces spécifications sont sauvegardées au format XML. Le XML(ou Extensible Markup Language) est un langage de balisage qui permet de structurer des données selon les règles que l'on peut personnaliser. Cela permet de créer une norme pour ces fichiers et rend leur utilisation par un programme informatique beaucoup plus simple si le schéma des données est connu.

Valeur nutritive	
par 175 g	
Teneur	% valeur quotidienne
Calories 130	
Lipides 0,5 g	1 %
saturés 0,3 g + trans 0 g	2 %
Cholestérol 4 mg	
Sodium 125 mg	5 %
Glucides 26 g	8 %
Fibres 0 g	0 %
Sucres 26 g	
Protéines 8 g	
Vitamine A 8 %	Vitamine C 4 %
Calcium 25 %	Fer 0 %

FIGURE 2.1 – Exemple tableau valeur nutritionnelle

— Un fichier XML est souvent défini de la manière suivante :

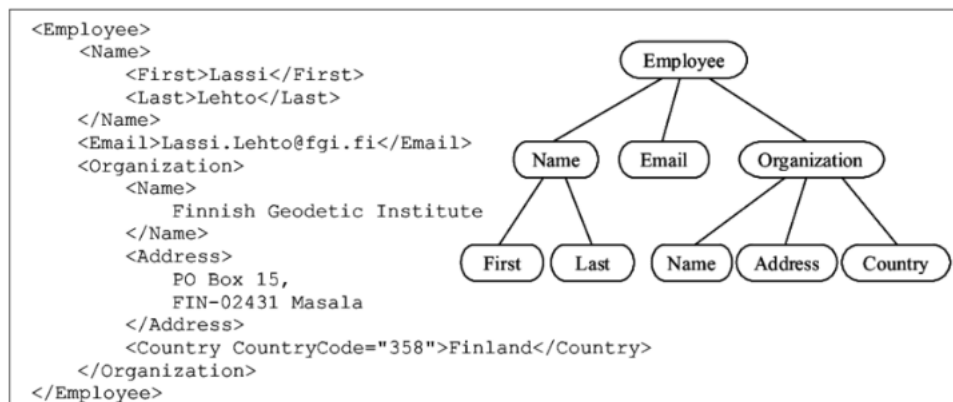


FIGURE 2.2 – Structure d'un fichier

2.3 Architecture

Étant arrivé plus d'un et demi après le démarrage du projet, je n'ai pas participé à l'élaboration de celui-ci. J'ai donc passé la première partie de mon stage à essayer de comprendre l'architecture de ce projet complexe qui

comprend trois parties différentes :

Front-end C'est la partie interface utilisateur qui est accessible via un navigateur internet (chrome, Firefox etc..)

Back-end le back-End est quelque sorte le moteur de toute l'application. C'est cette partie qui va s'occuper de toutes les communications, comme la récupération des données Alpha ou encore de certaines données avant de les envoyer aux différentes parties de l'application.

Core Le Core (ou cœur) est la partie algorithmique de l'application qui va traiter toutes les comparaisons demandées par l'utilisateur. Cette partie effectue donc des calculs plus intensifs qui ne pourraient pas être traités directement dans le Back.

Comme expliqué précédemment, le Lab fonctionne avec le tandem docteurs-développeurs. La partie Full-Stack (qui regroupe donc le Front-End et le Back-End) est donc prise en charge par les développeurs tandis que la partie Core est développée par les docteurs. Évidemment, les rôles sont flexibles et chaque personne peut intervenir sur chaque partie si nécessaire. Pour comprendre cette architecture, je me suis aidé de la documentation créée par les développeurs précédents. Voici un schéma de l'architecture globale de PackDiff :

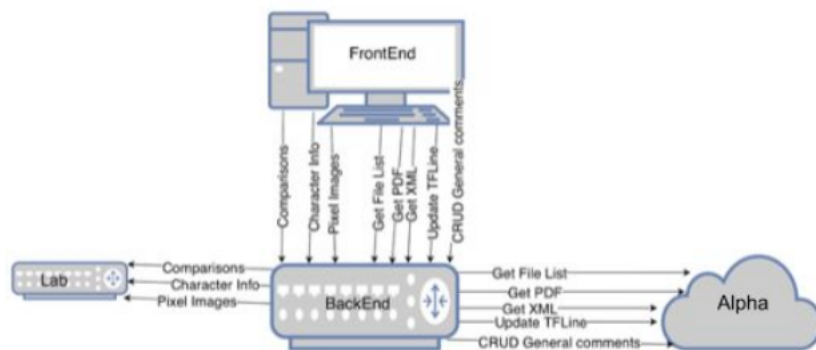


FIGURE 2.3 – Architecture PackDiff

2.3.1 Front-End

Cette application a été développée avec le framework VueJS. C'est un framework JavaScript permettant de construire des interfaces utilisateurs. VueJS est basé sur la notion de composants. Ce sont des éléments qui peuvent être apparentés à des classes. Cela permet de stocker des données pour chaque composant de la page et de définir des actions pour chacun d'entre eux selon interactions de l'utilisateur. Voilà comment une page VueJS en général : Cela permet une gestion dynamique de la page web. En

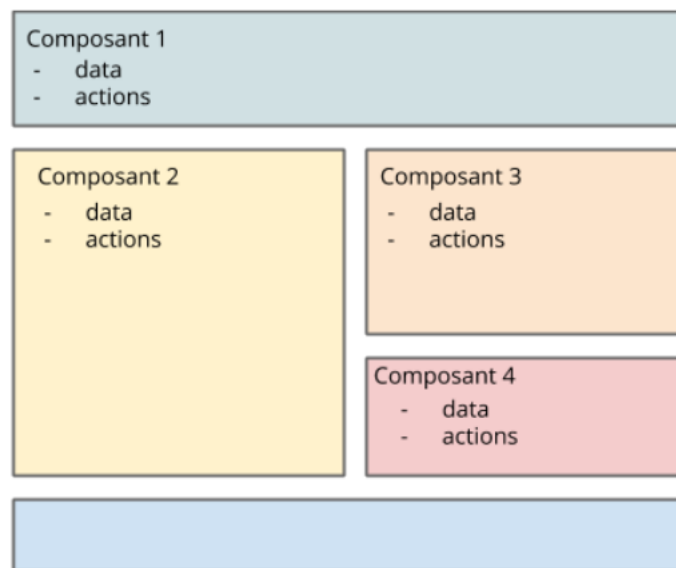


FIGURE 2.4 – Organisation d'une page VueJS

effet, dans configuration HTML/CSS classique, mettre à jour l'interface suite à une action utilisateur demanderait un rechargement de la page entière. Ici, si les données d'un composant change suite à une action, celui-ci se met à jour automatiquement.

Pour la gestion des ces données de composant et de ses actions , l'application utilise une librairie appelée Vuex. C'est un gestionnaire d'état qui sert de zone de stockage centralisée pour tous les composants Vue, avec des règles pour s'assurer que l'état ne puisse subir de mutations que d'une manière prévisible. C'est une librairie utilisée pour la création d'application web monopages, qui utilisent différents composants avec des états différents. Voici comment donction la vie d'un composant avec Vuex : On observe

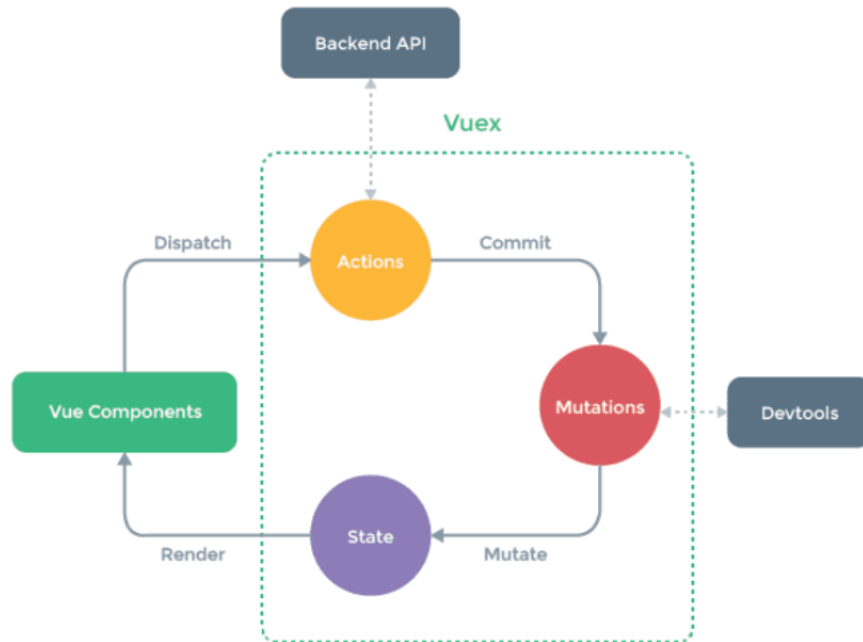


FIGURE 2.5 – Principe d'un composant Vuex

bien les trois parties d'un composant VueJS :

- La View (ou vue), qui est l'affichage du composant à l'utilisateur.
- Le State (ou état), qui comporte les données du composant et leurs valeurs à un instant T.
- Les Actions, qui définissent les interactions possibles avec ce composant

On observe aussi une partie Mutations, celle-ci est spécifique à Vuex et c'est ce qui assure la bonne modification de l'état du composant suite à une action. Cela peut aussi entraîner la modification d'un autre composant. Comme vu sur le schéma, les actions utilisateurs peuvent entraîner une communication avec le Back-End. Par exemple, l'utilisateur peut entraîner une comparaison, le Front-End va donc notifier le Back-End pour que celui-ci lance le traitement de données.

L'interface de l'outil se présente comme ceci : ici, il s'agit de l'affichage

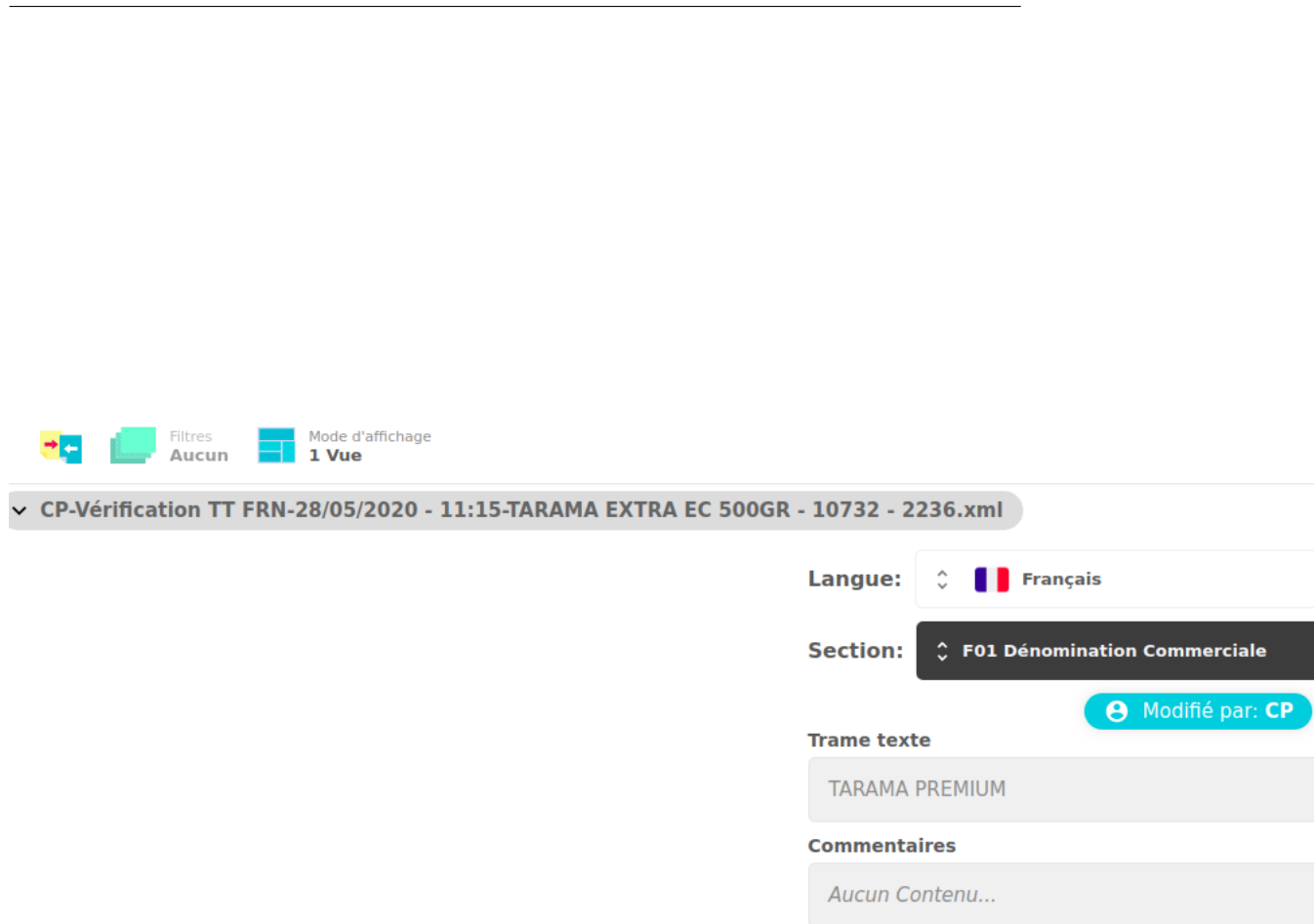


FIGURE 2.6 – Interface PackDiff n1

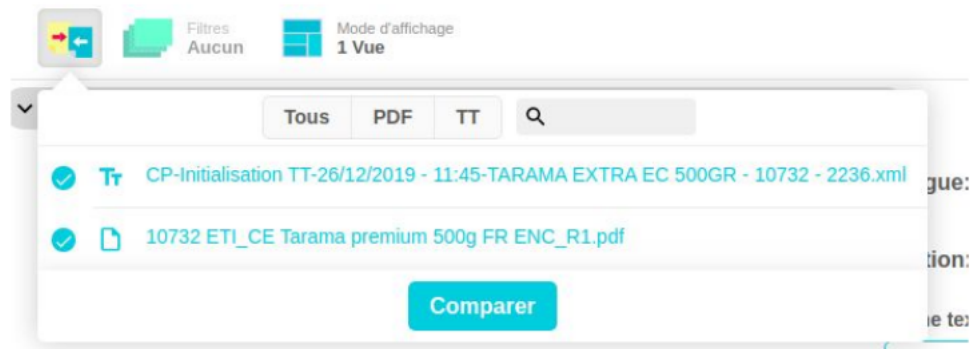


FIGURE 2.7 – Interface PackDiff n2

d'une Trame Texte, qui regroupe toutes les données qui doivent se trouver sur un certain emballage, il y a donc différentes langues et différentes sections afin de trier les données rentrées par le métier lors des spécifications. La fonction principale de cet outil est la comparaison, il en existe deux types :

- La comparaison Trame Texte-PDF
- La comparaison PDF-PDF L'utilisateur choisi les deux fichiers à comparer de cette manière :

Le résultat de la comparaison s'affiche de la manière suivante : Il y a donc




FIGURE 2.8 – Interface PackDiff n3

panels différents : l'un comportant les spécifications définies par le métier, l'autre contenant le PDF produit par le service externe.

Dans le panel de gauche on observe qu'il y a cette fois ci 2 champs avec du texte, il s'agit de celui de la Trame Texte(donc celui des spécifications) et celui extrait du PDF. Le surlignage orange indique à l'utilisateur les différences entre les deux textes. Dans le panel de droite, on observe des pastilles de couleurs. Ce sont les endroits où l'outil a détecté une erreur sur le PDF. Les couleurs traduisent différentes erreurs(texte différent, texte absent...). L'utilisateur a ensuite moyen de choisir de traiter la pastille(en bas à gauche) une fois qu'il considère qu'il s'est occupé de cette différence. Il s'agit là de la fonctionnalité principale de l'outil. La comparaison PDF-PDF affiche deux panels avec les PDF ainsi qu'un autre affichant le texte extrait.

Le front-End utilise donc Vue.JS avec le langage TypeScript. C'est une surcouche du JavaScript qui pour but d'améliorer et de sécuriser la production de code de ce type. Lors de la compilation, le TypeScript est compilé en JavaScript ce qu'il le rend utilisable par tous les navigateurs web. Les dépendances sont gérées par Yarn, un outil qui va télécharger toutes les librairies nécessaires pour le fonctionnement de l'application. Les dépendances sont définies dans un fichier package.json qui se présente comme ceci :



```
{
  "name": "dummy",
  "version": "1.0.0",
  "description": "",
  "main": "",
  "dependencies": {
    "bluebird": "^3.4.0",
    "config": "^1.20.4",
    "events": "^1.1.0",
    "express": "^4.13.4",
    "http": "0.0.0",
    "lodash": "^4.13.1",
    "lru-cache": "^4.0.1",
    "middleware": "^3.0.3",
    "node-static": "^0.7.7",
    "path": "^0.12.7",
    "request": "^2.72.0",
    "serve-favicon": "^2.3.0",
    "util": "^0.10.3"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "MIT"
}
```

FIGURE 2.9 – Exemple le package.json

Le serveur HTTP est défini grâce à NodeJS. En production, on utilise un reverse proxy nginx pour l'accès au Front. Cela permet d'accéder à l'application sur un serveur interne et non sur Internet directement.

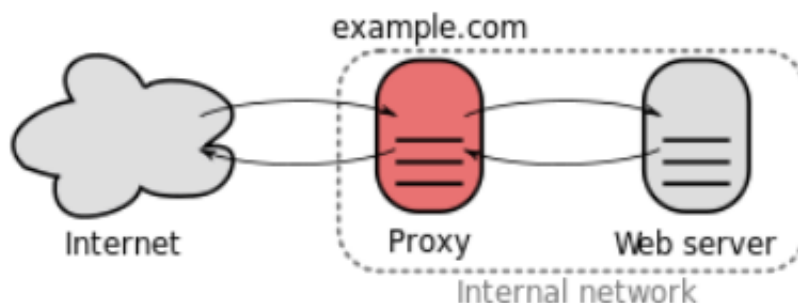


FIGURE 2.10 – Principe d'un proxy

2.3.2 Back-End

La partie Back-End est développée en Java grâce au framework SpringBoot. c'est un framework basé sur Spring qui facilite le développement d'applications java. Cela permet la création d'un fichier .jar ou .war lors de la construction afin que l'application puisse être lancée facilement. Les dépendances et la définition de la structure de l'application sont gérés par Maven. Le Back-End fonctionne comme une API REST pour que le Front-End puisse lui envoyer des requêtes. Le fonctionnement d'une API REST s'illustre comme ceci :

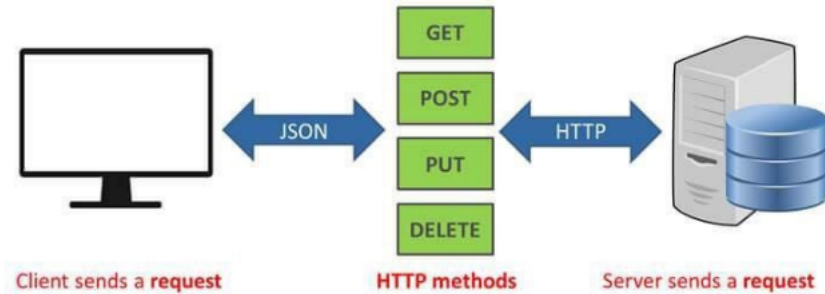


FIGURE 2.11 – API REST

Le Client (ici le Front-End) envoie une requête avec une méthode HTTP. Pour cela il faut que le Front-End connaisse l'URL de l'API du Back-End. Une fois celle-ci connue le Front-End peut appeler des fonctions du Back-End grâce à des requêtes du type :

```
return await Request<ComparisonResult>('GET', 'compare/pdf', {  
  referenceName,  
  referenceUrl,  
  comparedName,  
  comparedUrl,  
});
```

FIGURE 2.12 – requêtes Get

ici on retrouve la méthode GET qui sera envoyée à l'URL `http://APIURL/api/compare/pdf` avec différents arguments. Cette requête est transformée en format JSON et envoyée au Back-End qui comporte un fichier spéciale pour traiter les requêtes. C'est la partie dans le Back-End qui s'occupe de gérer les requêtes. Par exemple, ici la fonction s'occupe des requêtes GET sur l'URL `API-URL/api/compare/pdf`. Ainsi d'autres fonctions peuvent être appelées et le traitement des données peut être fait. Comme expliqué précédemment, le Back-End s'occupe de liaison avec l'outil interne du client appelé Alpha. C'est là se trouvent toutes les informations et les fichiers nécessaires pour les comparaisons (Trame Texte, Tableaux de valeur Nutritionnel, PDF). Pour cela, Alpha a mis en place des web services (WS) qui permettent à PackDiff de pouvoir envoyer des requêtes à leurs applications. Les web

```

@RestController
@RequestMapping(value = "/api/compare")
public class ComparisonResource {

    @RequestMapping(value = "/pdf", method = RequestMethod.GET)
    public ResponseEntity<ComparisonGlobalResult>
compare(@RequestParam("referenceName") String referenceName,
@RequestParam("referenceUrl") String referenceUrl,
@RequestParam("comparedName") String comparedName,
@RequestParam("comparedUrl") String comparedUrl) {
    }
}

```

services ont un fonctionnement un peu similaire aux API REST, il s'agit d'un ensemble de fonctionnalités exposées sur Internet et accessibles par d'autres applications. Pour cela, PackDiff doit d'abord s'identifier avant de pouvoir faire des requêtes. La communications entre le Back-End et Alpha s'organise de la sorte :

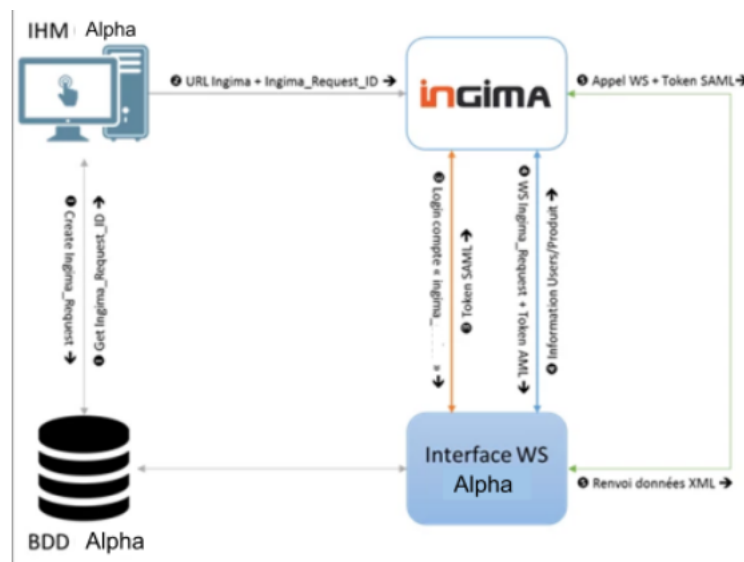


FIGURE 2.13 – communication PackDiff-Alpha

2.3.3 Core

Le Core c'est lui aussi codé en Java et fonctionne lui aussi comme une API REST pour que le Back-End puisse lui envoyer des requêtes. Une librairie principale est utilisée pour le traitement des PDF, il s'agit d'une librairie open source appelée PDFBox. Cette librairie permet la manipulation et la création de PDF, ainsi que l'extraction de données de celui ci (texte, image). Le Core effectue ces différentes tâches pour la réalisation d'une comparaison :

- Décomposition des données du fichier XML de la TT récupérée avec le Back-End
- Création de paragraphes avec le texte extrait du PDF
- Création d'une chaîne de caractère unique
- Exclusion des éléments ne devant pas être comparés
- Correspondance des textes venant des deux fichiers comparés
- Traitement des correspondances pour trouver les différences
- Traitement des différences à renvoyer au Back-End pour l'affichage final (position de l'erreur sur le PDF, ...)

La question qui se pose alors est : pour quoi y aurait-il besoin d'une partie cœur s'il s'agit d'une simple comparaison de texte ? Tout simplement parce qu'il ne s'agit pas d'une comparaison binaire. En effet, on ne veut pas seulement savoir si les textes correspondent ou non . On veut pouvoir identifier les différences, les catégoriser et les afficher directement à l'utilisateur. Cette partie nécessite donc de l'analyse sémantique pour comprendre d'où viennent ces différences, chose qui peut être très subtile parfois. C'est donc pour cette raison que les docteurs ont du développer des algorithmes capables de répondre a cette problématique.

Le Core utilise un algorithme d'alignement ainsi qu'un algorithme hongrois pour traiter ce problème. Nous appellerons match les correspondances entre les deux lignes de texte.

- Une TextLine est l'unité de base pour un match, elle contient le texte ainsi que les données annexes tel que la position dans le PDF .
- Un match est une pair de TextLine de chaque document avec score
- Un « True Match » est un match qui a été sélectionné parmi la liste des matchs possible.
- Un match véritable est le match à la fin de l'algorithme.

Une matrice est remplie avec les différents matchs et leur score. Une fois la matrice remplie, on peut sélectionner le meilleur match. Il s'agit donc d'une méthode heuristique puisque la solution trouvée n'est pas forcément la bonne à chaque coup et l'algorithme doit s'exécuter plusieurs fois avant de trouver le match qui semble le meilleur.

2.4 développement de PackDiff

2.4.1 Gestion de code

Une fois l'architecture et le principe du projet bien compris, je me suis familiarisé avec la méthode de développement du Lab. Le code est hébergé sur BitBucket, un service web d'hébergement et de gestion de logiciel utilisant le logiciel de version Git. Plusieurs personnes étant sur ce projet, il faut donc respecter une procédure lorsque l'on effectue des changements dans le code. Chaque nouvelle modification ou feature nécessite la création d'une nouvelle branche afin de travailler sur un environnement qui ne changera pas le comportement de l'outil pour les autres développeurs.



FIGURE 2.14 – git workflow

Comme montré dans le schéma ci-dessus, chaque nouvelle feature crée une

nouvelle branche. Pour que cette feature oit intégrée au master (c'est a dire à la branche finale qui peut-être potentiellement livrée au client), il faut faire ce qui appelé une « merge request ». Cela permet que quelqu'un vérifie la conformité du code avant que celui ci soit importé dans la branche principale.

2.4.2 Workflow

Comme indiqué précédemment je suis arrivé à une étape de vie avancée du projet. Du coup le projet était déjà en étape de garantie. Il s'agit d'une phase négociée par le Lab qui donne un certain laps de temps, 3 mois en l'occurrence, au client pour tester l'outil et remonter le plus de problème possibles afin que l'outil soit conforme à leurs attente finale.

Du coup, de puis mon arrivé on attendaient les retours du client qui était entrain de tester le produit.

Pour ce faire, une méthode de travail a du être mise en place, il a été décidé de travailler de maniéré Agile avec les équipes du client. En effet, du coté du client une équipe spéciale a été mise en place pour s'occuper du projet, faire le test de celui-ci, définir les fonctionnalités, C'est cette équipe qui joue le rôle de Product Owner de ce projet.

Les sprints sont des sprints d'une semaine. Il y a donc une livraison hebdomadaire du produit apportant des modifications en fonction des retours du client. Pour les remontées client, l'outil Jira est utilisé. Jira est un logiciel de gestion de projet et de suivi de bugs. Cela fonctionne par un système de tickets. Le processus de remonté de bugs se passe ainsi :

- Le client repère un bug ou veut une modification de l'outil
- le client créer un ticket sur Jira avec la description du bug et comment le reproduire
- L'équipe de développement du Lab est notifiée de l'arrivée du ticket et commence à faire une estimation du temps de résolution
- Le développement de la solution commence avec la création d'une nouvelle branche Git
- Le développement de la solution commence avec la création d'une nouvelle branche Git
- La résolution sera livrée lors de la prochaine livraison hebdomadaire

Les tickets Jira son organisé sur un Board et triés en différentes catégories comme cet exemple on voit ici que les tickets organisés par colonne . Les colonnes définie pour le projet PackDiff sont les suivantes :

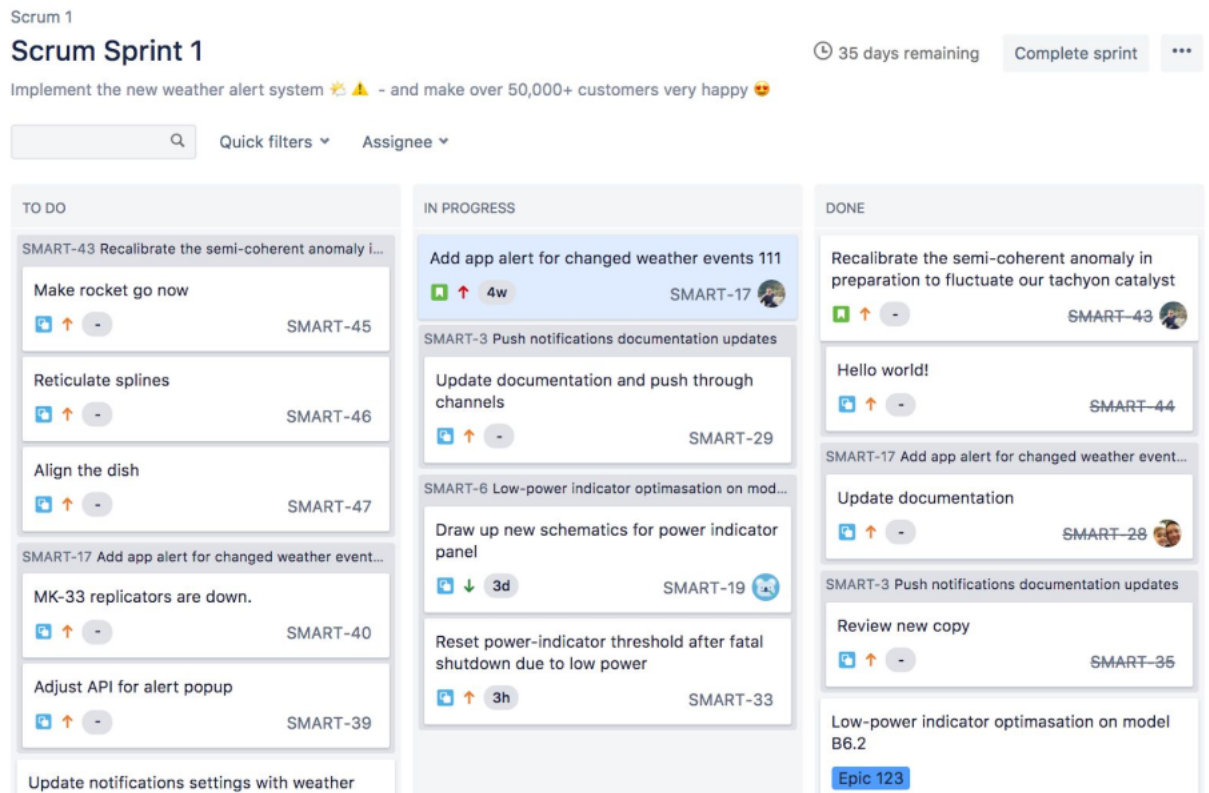


FIGURE 2.15 – Exemple tableau Jira

To DO

Cela correspond aux tickets qui sont arrivés récemment et qui n'ont pas encore été traités par l'équipe.

In Work

ce sont les tickets en cours de traitement par l'équipe. Le but est de définir une date de livraison le plus vite possible .

Missing Information

Cette catégorie regroupe les tickets qui ont commencé à être traité mais où le développeur s'est rendu compte qu'il manquait des informations du côté client pour pouvoir s'attaquer à la résolution.

In Review

Ce sont les tickets qui ont été livrés et qui attendent donc la vérification du ticket par le client pour que celui-ci s'assure que la résolution est bien valide.

Done Les tickets résolus et validés par le client.

Les tickets peuvent être de deux catégories :

Tasks

Ce sont des tickets qui correspondent à un changement de l'outil ne correspondant pas à un bug.

Bugs

Ce sont des comportements ne correspondant pas à celui attendu par le client. Il y a donc une manière pour le reproduire.

Afin que l'équipe de développement sache quel ticket traiter en premier un système de priorité a été mis en place. Les priorités de la plus grande à la plus petite, s'organise dans ce sens :

Important

Ce sont des tickets qui gênent l'utilisation de l'outil.

Mineur

Les tickets mineurs ne gênent pas l'utilisation de l'outil à des petits bugs. des priorités peuvent être définies entre les tickets mineurs.

Le client met à jour priorités des tickets toutes les semaines en s'accordant avec ses équipes internes. Leur compte rendu est tableau de type :

ID ticket	Catégorie	Description du ticket	Priorité	Criticité
213	Bug	Problème d'affichage PDF	1	Bloquant
215	Tâche	Changement message pop-up	2	Mineur

2.4.3 Processus de Livraison

Le processus de livraison de PackDiff a subi plusieurs changements depuis mon arrivée jusqu'à l'écriture de ce rapport. Lors de mon arrivé, le processus était automatisé via un agent Jenkins et cela déployait directement l'outil chez le client.

En effet, pour le déploiement, l'utilisation de Docker est un outil qui permet de lancer des applications dans des conteneurs logiciels. Un conteneur va comprendre l'application ainsi que toutes ses dépendances afin qu'elle puisse se lancer sur n'importe quelle système sans installation requise. il faut cependant que le système possède un agent Docker pour cela. Une

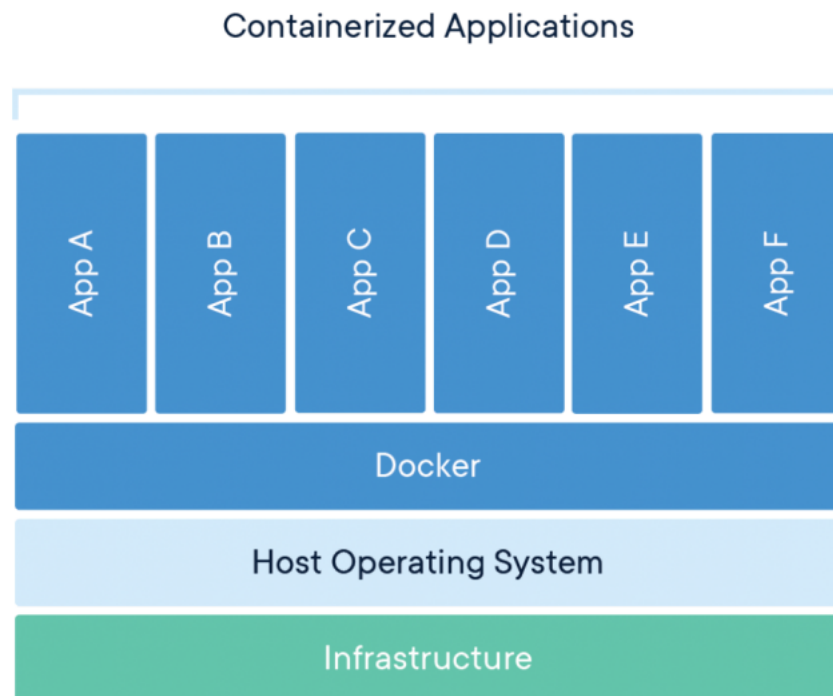


FIGURE 2.16 – fonctionnement d'une application Docker

image Docker est un fichier permettant la création de ces conteneurs. Elles sont créées lors d'une construction et s'occupent de la compilation du code. il n'y a plus ensuite qu'à les lancer. Une image Docker est créée pour chaque partie du logiciel : Une image Back-End, Front-End et Core

2.4.4 Rôle et tickets

Mes rôles pour ce projets ont été diverses. avec le renouvellement de l'équipe au mois de Septembre, je me suis retrouvé seul développeur Full-stack avec un autre développeur pour la partie Core. J'avais donc la prise en charge de tous les tickets touchant au Back-End et au Front-End, et ce jusqu'à l'écriture de ce rapport.

Gestion des tickets

Une fois l'équipe du Lab a vu que je maîtrisais le projet, je me suis vu attribuer le rôle de gestionnaire des tickets. En effet, lors de l'arrivée de nouveaux tickets par le client il faut que l'équipe s'organise pour traiter ceux-ci le plus rapidement possible ou pouvoir donner une estimation. J'ai donc jouer ce rôle la plus grande partie de mon stage.

Cela comportait les action suivantes :

- Vérification du Backlog des tickets voir si de nouveaux sont arrivés.
- Attribution des tickets à l'équipe en fonction de la partie concernée (front-End, back-End, Core).
- Vérification des estimation de résolution pour les tickets en cours.
- Déplacement des tickets dans le Board lorsque ceux-ci sont résolus, vérifiés, manque d'information, etc...

j'avais aussi le rôle de revoir l'anciennes documentation de PackDiff et de faire une mise a jours pour cette documentation.

Résolution de tickets

Afin de sauvegarde tout les traitements des pastilles le client nous à livrer un ticket pour la mise en place d'une base de données,
La description du ticket livré par le client :

Description

CR Point prérequis BDD (21/02/2020) :

Personnes présentes : Fernando Villanueva, Nicolas Viénot, Yacine Kermouche, Loris Présent, Wissem Mahjoub, Said Zane, Sébastien Morineau, Victoria Jallat

Décision :

- Postgree 11,
- Managée chez GCP
- 2 BDD : 1 Prod, 1 non Prod (REC / PPRD)
- Liquidbase pour permettre l'automatisation complète du déploiement
- **Purges / Shutdown à valider équipe projet (soir / week end ?)**
- Budget : 100€/mois environ pour chaque BDD, soit 200€/mois pour tous les environnement

/!\ Risques potentiels identifiés :
une bonne coordination Devops indispensable


Avantages de la solution choisie :

- persistance des données
- synchronisation
- migration facilitée (en phase avec la migration de l'appli dans l'AF)

Next steps :

- ouverture des deux tickets (1/BDD) par Loris Présent pour la création ==> environ 1 semaine d'attente après réception du ticket
- Accès ==> 2 jours
- Développements Ingima

Rapporteur

 ykermouche

Développement

4 commits

1 pull request

Étiquettes

Aucun

IMA Client Issue Summary

Aucun

Priorité

↓ Simple

▼ Afficher 6 champs supplémentai
Story Points, Estimation originale, Sui

Date de création 25 février 2020 à 16:18
Date de mise à jour 17 mars 2020 à 11:11

FIGURE 2.17 – extrait d'un ticket Jira

Après avoir fait une réunion avec notre client dans ses locaux, pour parvenir à bien saisir l'importance de cette évolution et de faire le choix de la technologie et des versions pour entamer cette partie.

voici quelques grandes lignes de cette réunion :

- Pour résoudre ce ticket fallait déjà installer une base de donnée PostgreSQL 11 sur la machine local pour faire des testes, avant d'utiliser un serveur de base de donnée Docker pour faciliter les migration.
- intégrer liquibase afin de contrôler l'évolution de cette base de donnée.
- Ajouter le framework open source Hibernate .
- créer une nouvelle entité pour sauvegarder la persistance de donnée.

L'immense travail concernant ce ticket était fait sur la partie Back-End, et ce qui concerne le front j'ai créé des appels vers Back-End afin de lancer les fonctions à chaque traitement d'une pastille.

Après j'ai eu aussi beaucoup de tickets concernant le front, Ajustement de la barre de navigation et l'ajout d'une page help, qui va contenir un guide pour les utilisateurs de PackDiff.

2.5 Conclusion

Durant ce stage j'ai acquis une expérience dans le monde professionnel du travail j'ai pu avoir échangé avec un client et de répondre à ses besoins, et cela permis surtout d'améliorer mes connaissances dans le développement web et de suivre les différentes parties dans la gestion d'un projet.

Et à travers ce Stage j'ai notamment approfondi mes connaissances en Java Jee et VueJs et comprendre aussi le processus Devops qui est utilisé dans l'entreprise afin de automatiser le processus de livraison cette dernière est une pratique technique qui accélère le mouvement de livraison, le Lab me permet aussi d'assister à des réunions avec des chercheurs sur différents thèmes tel que l'intelligence artificielle et sur des projets innovants qui peuvent intéresser le marché.

J'ai eu la chance de travailler dans un cadre agréable et professionnel, ce que m'a permis de voir le monde de travail de plus près, et d'acquérir une expérience professionnelle.

Table des figures

1.1	Fonctionnement Ingima France	7
1.2	Organisation du Lab	8
1.3	Domaine du Lab	9
1.4	Fonctionnement d'un projet du Lab	10
2.1	Exemple tableau valeur nutritionnelle	14
2.2	Structure d'un fichier	14
2.3	Architecture PackDiff	15
2.4	Organisation d'une page VueJS	16
2.5	Principe d'un composant Vuex	17
2.6	Interface PackDiff n1	18
2.7	Interface PackDiff n2	19
2.8	Interface PackDiff n3	19
2.9	Exemple le package.json	20
2.10	Principe d'un proxy	21
2.11	API REST	22
2.12	requêtes Get	22
2.13	communication PackDiff-Alpha	23
2.14	git workflow	25
2.15	Exemple tableau Jira	27
2.16	fonctionnement d'une application Docker	29
2.17	extrait d'un ticket Jira	31