MEMOIRE DE STAGE

Les échanges de données informatisés : clé de la stratégie de l'entreprise ?

Alors les entreprises cherchent à accroître davantage leur productivité et performance, les échanges de données informatisés sont-ils la clé de la stratégie de l'entreprise ?

Nathalie Durassier-Bonheur
EPSI B2 – 2019-2020

En premier lieu, je tiens à remercier M. Anh DAO-DUY, Directeur des Systèmes d'Information pour le groupe DLJ. En tant que maître de stage, il a su m'accompagner tout au long de cette période. Il a su se rendre disponible malgré ses obligations et a su partager ses nombreuses connaissances dans le domaine informatique.

Je souhaite remercier aussi Mme. Maëliss ABLAIN, Mme. Lisa ROLLAND, M. François LAVERGNE, M. Charles LEGROS et Mr. Pierre DEGUIGNE, pour leur accueil chaleureux au sein de leur bureau. Je les remercie aussi pour leur disponibilité et la qualité de leur encadrement en entreprise. Je remercie aussi tous les employés du groupe DLJ que j'ai pu rencontrer et avec qui j'ai pu échanger durant ces cinq semaines de stage.

Je remercie aussi Mr. Nicolas DURET, avec qui j'ai partagé ce stage et ai pu collaborer sur ce même projet.

Je tiens aussi à remercier Mr. Didier LAURY, directeur général du groupe DLJ, pour m'avoir permis de vivre cette expérience professionnelle très enrichissante.

J'adresse enfin mes remerciements au corps professionnel et administratif d'EPSI Nantes pour la qualité de leur enseignement ainsi que pour leur disponibilité.

SOMMAIRE

INTRODUCTION

Les données sont au centre de tout. Quelles soient informatisées ou non, elles constituent l'essence même des entreprises. Dans un monde qui se veut de plus en plus numérique, les entreprises cherchent sans cesse à augmenter leur performance et leur productivité. Vient alors la question de la gestion des flux de données qui ne cessent de croître. Une mauvaise gestion des données peut s'avérer fatale, car celle-ci impacte la compétitivité, la réactivité et l'efficacité d'une entreprise. Les enjeux sont nombreux et l'entreprise se doit d'être responsable de sa gestion des données. Au travers de la gestion des données, on inclut leur stockage, leur qualité, leur sécurisation, leur optimisation ainsi que leur conformité aux obligations légales.

Le groupe DLJ, société holding dans laquelle le stage s'est déroulé, doit, comme toute autre entreprise, faire face à cette problématique. Constituée de filiales de secteurs divers et variés, l'hétérogénéité des données a conduit le groupe à un besoin précis : mettre en place une application permettant de gérer et traiter les données. L'objectif du stage était donc de participer à ce projet et de réaliser une application web permettant de répondre à ce besoin. L'exploitation de données traitées sera un second objectif de ce stage.

Les échanges de données informatisés (souvent appelés par l'acronyme EDI) désignent tout échange d'informations utilisant l'informatique, comme un échange de documents d'une entreprise à une autre par voie électronique. Ce processus a été mis en place dans l'optique de supprimer l'utilisation de papier et afin d'automatiser le traitement de l'information. L'EDI peut prendre diverses formes mais a une réelle importance pour une entreprise, que ce soit en termes de gain de temps, d'efficacité ou d'ordre financier. Le Web-EDI sera ici utilisé : une application accessible depuis internet sera mise à disposition des employés, leur donnant accès aux données et leur permettant de les traiter. Toute entreprise est définie par des objectifs, des buts à accomplir.

La stratégie d'entreprise est le moyen que celle-ci emploie pour les atteindre. Les choix et l'orientation des actions que mènent l'entreprise compose aussi sa stratégie. Cette-dernière permet à l'entreprise d'acquérir un avantage-concurrentiel sur le marché qu'elle occupe. La stratégie qu'elle emploie a donc son importance dans la direction et l'évolution de la société.

Face au besoin et au contexte de la société, nous pouvons nous questionner sur l'importance qu'ont les échanges d'informations dans la stratégie d'une entreprise. La problématique que nous traiterons dans ce mémoire est la suivante : les échanges de données informatisés sont-ils la clé de la stratégie d'une entreprise ?

La réalisation d'une application web permettant de gérer les données de l'entreprise peut s'avérer être nécessaire et indispensable. Ce besoin constituait la première mission du stage. La seconde était d'exploiter ces données afin d'obtenir des visuels, tels que des graphiques, permettant aux sociétés d'avoir un retour sur leurs chiffres ou encore sur leur clientèle.

Avant d'évoquer l'état de l'art qui a été réalisé, nous allons présenter et préciser le contexte dans lequel le stage s'est déroulé. L'étude technologique sera ensuite détaillée, précisant ainsi les outils et les processus travaillés. A la suite de ces deux premières parties nous évoquerons la réalisation technique de la mission et ferons une évaluation de ces-dernières. La méthode de gestion de projet employée durant cette période de stage sera enfin détaillée.

II – CONTEXTE

Cette partie a pour but de mettre en lumière le contexte dans lequel s'est déroulé le stage.

Le stage s'est déroulé dans l'entreprise Abelys Services. Cette entreprise appartient à la société holding DLJ. Ce groupe est un dirigé par D. LAURY. Cette société intervient dans différents secteurs d'activités comme l'immobilier, la cosmétique, l'automobile, l'hôtellerie. Il s'agit d'une holding, que l'on peut définir comme étant un groupe de sociétés. Les sociétés holding détiennent des actions, des titres dans plusieurs entreprises pouvant appartenir à différents secteurs. Le principal objectif d'une telle organisation est d'obtenir une unité de direction, simplifiant ainsi sa gestion. Il existe deux types de sociétés holding : les sociétés holding passives, qui se content simplement de détenir des participations dans d'autres sociétés ; et les sociétés holding actives, qui, de plus, fournissent des services à leurs filiales. DLJ est une société holding passive. En effet, les sociétés membres du groupe fournissent des services entre elles. Par exemple, l'entreprise Au Bio Jardin a fait appel à Abelys Services pour la création de son internet. Autrement appelée « société mère » ou « société consolidante », la holding donne de nombreux avantages juridiques, opérationnels et fiscaux entre autres. Elle permet notamment la mutualisation de certains services ou encore l'isolation des fonctions supports de la holding.

Sous le nom de DLJ, la holding regroupe ainsi différentes filiales telles qu'Au Bio Jardin (entreprise marchande dans le domaine du jardinage), le supermarché E.Leclerc de la galerie commerciale Océane (à Rezé, commune de la région de Nantes), Italy's diner (restaurant rapide situé dans le centre commercial), Akena Hotels (chaîne hôtelière), Fitness Prime (salle de sport située dans la galerie marchande) et Abelys Services (prestataire de services). Vous trouverez en annexe n°1 (voir p. 26), l'organigramme du groupe DLJ qui reprend la description faite ci-dessus.

Situés au sien du centre commercial Océane à Rezé (commune limitrophe de Nantes), les bureaux rassemblent les différentes filiales de la holding. On y retrouve alors divers métiers, du comptable au directeur des systèmes d'information. Le stage s'est donc déroulé dans la société Abelys Services. Le poste occupé était celui de développeur. Dans un open-space d'une dizaine de personnes, on y retrouve des graphistes, des chargés de communication marketing, un web master ainsi qu'un directeur des systèmes d'information.

Abelys Services est récente puisque la société a été créée en novembre 2019. Il s'agit d'un prestataire de services, autrement dit, la société offre, entre autres, des services aux différentes sociétés du groupe DLJ. Abelys Services est ainsi spécialisée dans le conseil marketing, commercial et digital. Sa principale mission est donc de conseiller les filiales du groupe DLJ au niveau de leur stratégie commerciale et digitale. L'entreprise dispose ainsi d'un pôle marketing et un pôle information. Le pôle marketing est constitué d'une responsable marketing, de graphistes ainsi que d'une chargée de marketing digital en alternance. Pour ce qui est du pôle informatique, l'équipe est formée d'un webmaster et d'un directeur des systèmes d'informations.

Les missions d'Abelys Services tiennent donc principalement des besoins des autres sociétés du groupe DLJ. Les sites internet sont, par exemple, créés par Abelys Services. La société cherche sans cesse des moyens, des outils afin d'optimiser et de simplifier les tâches quotidiennes des salariés. C'est un ce contexte que s'inscrit la mission de stage. L'entreprise avait besoin d'un logiciel permettant de rassembler les données des différentes entités, et ainsi les traiter sur une unique plateforme avant de

les stocker et de les utiliser à des fins comptables (analyse de chiffres d'affaires, de clientèles, etc…). Le besoin était d'autant plus grand avec la reprise de la salle de sport Fitness Prime. Auparavant, il s'agissait d'un complexe Fitness Park rattaché à cette SARL. Le groupe DLJ a récupéré les locaux afin d'y créer sa propre salle de sport. Il était donc nécessaire d'avoir un visu sur les chiffres et données de cette nouvelle enseigne – comme le nombre de nouveaux adhérents, les adhérents perdus à la suite du changement. L'application créée durant le stage devait donc répondre à ce besoin.

Deux missions ont été définies. Le projet a été défini, encadré et géré par Mr. DAO-DUY. Deux développeurs juniors ont été missionnés sur ce projet, Nicolas DURET et Nathalie DURASSIER-BONHEUR. La première mission consistait en le développement d'une application web métier ETL (Extract-Transform-Load) permettant la gestion des données. Ces données, une fois traitées, font l'objet d'une exploitation sous Redash, qui représentait la seconde mission. Nous reviendrons plus précisément sur la réalisation de ces deux missions.

L'interface d'administration devait donc répondre à des besoins précis. Sa conception représentation la première et la plus longue mission. L'outil devait permettre d'administrer les sociétés, les types de sources de données (CVS, API, XML, par exemple), les catégories des données, les utilisateurs et leur rôle, le téléchargement de données (fichiers CSV, XML entre autres), la visualisation des données téléchargées en attente de traitement, l'exécution des opérations d'importation d'un Data Warehouse. Traduit de l'anglais, un Data Warehouse signifie un entrepôt de données. Il s'agit plus précisément d'une large base de données stockant des données structurées (propres et fiables) visant à être analysées. Pour ce qui est du choix de la solution technique, il a été demandé de choisir des solutions fiables et de préférences Open Source, disposant d'une communauté active. Une présélection a été établie dans le cahier des charges avec une liste de références à privilégier. L'objectif final pour l'entreprise est de disposé d'un outil de management unique à toutes les filiales du groupe DLJ. Cet objectif fait référence au concept appelé Master Data Management (ou encore un outil de gestion des données de référence, en français). Il s'agit d'un ensemble de méthodes, d'outils et de processus assurant la qualité des données de référence. Autrement, il permet de nous assurer que les données sont bien identifiées, de bonne qualité, sans erreur et utilisables sans aucun risque. Les données sont ainsi centralisées et sont partagées entre les différents services et entités du groupe DLJ, dans notre cas. Ce concept de Master Data Management (MDM) apporte de nombreux avantages à l'entreprise, notamment l'unicité et la fiabilité des données. Le cycle de vie des données peut ainsi être centraliser, de leur création à leur suppression en passant par leur mise à jour. Il est d'autant plus utile et avantageux pour les sociétés comptant un grand nombre de services ou de filiales, comme c'est le cas pour DLJ.

Concernant la seconde mission, qui consistait en l'exploitation des données traitées, celle-ci se fera avec Redash. Cette mission a pour but de réaliser un tableau de bord à partir des données. C'est à travers de ces tableaux de bords que la précédente mission prend tout son sens. Les données traitées vont pouvoir être utilisées à des fins comptables et financières. Les employés et responsables auront, par exemple, un visu clair et en temps réel de l'évolution de leur clientèle ou de leur chiffre d'affaire. Pour la nouvelle salle de sport, Fitness Prime, il est important et nécessaire d'avoir le nombre de clients réinscrits (ceux qui ont repris leur abonnement malgré le changement d'enseigne), les clients qui se sont désinscrits et les nouveaux clients. Un seul graphique permet de donner rapidement toutes ces données. Ces-dernières peuvent être, en un clic, réactualisées. Nous détaillerons la réalisation de cette mission dans la suite du document.

Une dernière mission a été donnée : rédiger une documentation d'utilisation, permettant l'accompagnement des futurs utilisateurs de l'application. Un dossier de mise en exploitation de la plateforme créée était attendu. Cette dernière mission n'a malheureusement été effectuée, faute de temps. Le chef de projet se chargera de la réaliser. Nous ne l'évoquerons donc pas davantage.

Comme dans tout projet, il existe des contraintes temporelles, financières, matérielles et logicielles. En termes de temporalité, la mission devait respecter une durée définie de cinq semaines,

du lundi 6 janvier au vendredi 7 février 2020. La mission confiée devait donc être faisable sur cette courte période et a donc dû être prise en compte lors de la mise en place d'un planning et fait partie des contraintes du projet. Aucune contrainte financière n'a été définie, si ce n'est la volonté de préférer l'utilisation de logiciels open source, qui renvoie aussi à une contrainte logicielle. Cela évite l'achat de logiciel. Cependant cette préférence n'a pas été respectée puisque l'entreprise a eu recours à l'achat d'un logiciel pour la réalisation de l'application de gestion de données. Nous reviendrons sur cet achat lors de l'étude de l'art réalisée avant la prise de décision du choix de l'outil de développement. Pour ce qui est des contraintes matérielles, l'entreprise n'avait pas de quoi fournir aux stagiaires un poste fixe de travail. Les machines des stagiaires ont donc été utilisées tout au long du stage. Les applications et logiciels nécessaires au bon déroulement de la mission ont donc été installés à leur arrivée et durant toute la période de stage. Les stagiaires ont été accompagnés par une même et seule personne, le directeur des systèmes d'information, Mr DAO-DUY. Le personnel de l'entreprise se tenait, cependant, disponible pour toute interrogation ou incompréhension dans la mesure du possible et dans le cadre de leurs savoirs et savoir-faire.

Nous reviendrons sur la réalisation de ces missions lors de la description de leur réalisation (cf. p. 15).

* * *

III – ETAT DE L'ART ET ETUDE TECHNOLOGIQUE

Cette partie consiste à analyser les techniques et outils correspondant aux missions données et déjà existants. Les outils et méthodes, qui seront utilisés pour la réalisation de l'application web métier ETL, seront décrits et expliqués dans l'étude technologique.

A – ETUDE DE L'ART

Avant de se lancer dans la réalisation des missions, il était nécessaire et demandé d'établir un état de l'art, c'est-à-dire une étude de ce qui existe déjà et qui est déjà mise en place et fonctionnel. L'état de l'art permet de faire une analyse des connaissances, des technologies utilisées à un instant donnée. Une étude technologique a ensuite été réalisée, précédant elle aussi, la réalisation technique.

Un état de l'art a d'abord été réalisée sur les outils et les méthodes ETL (Extract-Transform-Load) existante. Nous reviendrons sur l'analyse des méthodes ETL lors de l'étude technologique (cf. p. 11). Il a été constaté que de nombreux logiciels et applications ETL existent. Quelques-uns ont été étudiés et testés.

Un tableau comparatif a été établit rassemblant les avantages et inconvénients des différents langages ou outils testés.

ReactJS	Laravel	Symfony
+ Framework JS + Définition simple des composants et des vues + Mise à jour des composants automatisée + Technologie récente + OpenSource + Communauté présente et tutoriels en ligne - Mais langage moins bien maîtrisé que le PHP	+ Framework PHP + Langage PHP connu + Syntaxe facilitée + Bibliothèque orientée objet + Outil de ligne de commande intégré (Artisan) + Prise en charge de l'architecture MVC + Permet de générer des urls + Tutoriels en ligne + OpenSource + Beaucoup de modules disponibles pour couvrir les besoins du projet - Framework jamais utilisé	+ Framework PHP + Langage PHP connu + Framework connu et déjà utilisé + OpenSource + Bibliothèque orientée objet + Nombreux modules disponibles pour couvrir les besoins du projet + Tutoriels en ligne + OpenSource - Communauté moins présente que pour le framework Laravel
Laravel Nova	**CRUD Admin Generator**	**ForestAdmin**
+ Laravel Nova : permet de créer une page d'administration avec diverses fonctionnalités de gestion et d'administration + Tutoriels en ligne + Communauté présente + De nombreux modules disponibles couvrant les besoins du projet - Mais payant	+ Outil Open Source en PHP + Permet de créer facilement une interface d'administration + Liaison facile avec MySql + Couvre en partie les besoins du projet - Mais beaucoup de fonctionnalités superflues et non nécessaires au projet	+ Framework OpenSource permettant de créer un panneau d'administration + Couvre en partie les besoins du projet - Mais beaucoup de fonctionnalités superflues et non nécessaires au projet

D'autres frameworks et outils ont été étudiés, comme Django ou JetAdmin, mais ces-derniers sont basés sur le langage Python. Ce langage nécessitait une prise en main et un temps d'apprentissage plus long que les autres solutions à dispositions.

Des solutions ETL ont été testées, sur recommandation du référent du projet, Mr DAO-DUY. Ces solutions, plus complexes, offrent de réels avantages pour ce projet. Malheureusement, ces solutions n'ont pas été exploitées au sein du projet. Celles-ci ont tout de même été testées et ont fait l'objet de démonstrations.

Trifacta est un logiciel Open Source créé en 2012 par la start-up éponyme installée à San Francisco. Ce logiciel est dédié à la transformation et préparation de données avant leur éventuel traitement et stockage dans un Data Warehouse, si l'on suit le processus ETL. Trifacta automatise le

processus d'organisation des données, il permet de « nettoyer » les fichiers. Ce logiciel a donc été testé avec un fichier CSV qui nécessitait la suppression de guillemets, entre autres. Le logiciel nous permet, par exemple, de reformater une date, de supprimer des caractères ou encore de remplacer un point par une virgule. Ce travail pourrait être fait à la main mais plus difficilement et dans un plus large laps de temps. Dynamique et pratique, Trifacta est un outil facile à prendre en main et qui fait gagner un temps précieux dans la phase de traitement, de transformation de données.

Google a créé une API à partir de Trifacta. Cloud Dataprep de Google permet ainsi d'explorer visuellement, de nettoyer et de préparer les données avant de les transférer sur notre Data Warehouse.

Existante depuis 2005, la société Talend crée des logiciels ETL. Leur logiciel Talend Open Studio for Data Integration permet de créer des flux. Chaque flux, créé sur l'interface graphique, représente un transfert de données d'une source vers une autre. Logiciel Open Source, Talend peut s'utiliser dans n'importe quel contexte dans lequel des données sont utilisées et nécessitent un traitement ETL. Talend facilite ainsi le processus ETL, il permet d'extraire des données d'une source, de modifier les données et enfin les recharger vers une destination, comme un Data Warehouse.

L'outil Redash a été, lui aussi, testé et sera utilisé afin de réaliser la seconde mission. Redash est un logiciel permettant d'exploiter des données qui proviennent de plusieurs sources différentes ou non et de les visualiser en créant des dashboards. Ce software nous permet d'écrire des requêtes facilement en profitant de la puissance et du confort d'un client SQL tout en gardant les avantages d'un service basé sur le cloud. Une fois les requêtes créées, Redash nous permet de les visualiser sur des tableaux de bord. On peut ainsi planifier le rafraichissement des données et partager les tableaux de bord avec nos collaborateurs ou au public.

L'étude de l'art a montré qu'il existait de nombreuses applications ETL semblables aux attentes du projet. Une étude technologique est nécessaire avant toute réalisation afin de décider des outils qui seront utilisés pour réaliser le projet.

B – ETUDE TECHNOLOGIQUE

Dans cette étude technologique, figure le choix du langage de programmation ainsi que l'architecture fonctionnelle mise en place.

Une longue étude a été réalisée avant de définir le langage de programmation de l'application web métier ETL. Les processus des méthodes ETL ont ainsi été détaillés et expliqués durant cette phase de projet que l'on peut intituler la phase d'étude de conception ou encore la conception préliminaire. Nous détaillerons quelles ont été les différentes phases du projet lorsque nous évoquerons la gestion de projet (cf. p. 22).

L'acronyme ETL (Extract-Transform-Load) est utilisé pour désigner un type de logiciel qui permet la collecte des données en provenance de diverses sources ainsi que leur traitement avant leur transfert dans un Data Warehouse. Un logiciel ETL permet donc d'extraire des données brutes à partir d'une base de données. Ces données peuvent ensuite être traitées, restructurées, transformées au sein du logiciel. Une fois la transformation effectuée, les données peuvent être chargées dans une Data Warehouse. Existant depuis les années 1970, ce type de logiciel a beaucoup évolué ces derniers temps

notamment avec l'essor du Cloud, des SaaS (Software as a Service) et l'émergence du Big Data. A son apparition, l'outil ETL servait à rassembler et intégrer des données diverses, provenant de multiples sources. D'années en années, les sources et les types de données ont augmenté.

L'objectif premier de cette étude était donc de comprendre le fonctionnement de l'ETL. Repartons du contexte : DLJ regroupe plusieurs sociétés qui possèdent chacune, entre autres, un chiffre d'affaire et des clients. Le groupe a besoin d'analyser facilement et en même temps ces données. Cependant, le type de données collectées pour chacune des entreprises ne sont pas systématiquement les mêmes. C'est à ce niveau que le logiciel ETL intervient : son rôle est de collecter les données venant de sources différentes et de les transformer afin de leur rendre compatibles avec le Data Warehouse, la cible finale. Il existe trois phases dans le processus ETL :

- la phase d'extraction, consistant à collecter les données qui proviennent des différentes sources (des différentes sociétés, par exemple) ;
- la phase de transformation, consistant à traiter, à transformer les données afin qu'elles puissent être transférées dans le Data Warehouse ;
- la phase de chargement (loading) consistant à charger, à transférer les données transformées vers le Data Warehouse, la cible finale.

Un schéma représentant le processus ETL et résumant les explications précédentes se situe en annexes (Annexe n°2).

A l'issu de l'état de l'art réalisé et décrit précédemment, Laravel a été choisi comme langage de programmation pour la conception de l'application web.

Laravel est un framework PHP. Un framework peut se définir comme étant un ensemble cohérent de composants logiciels structurels, servant à créer les bases et les grandes lignes d'un logiciel. Il offre donc une base cohérente avec les premières briques du projet à disposition. Le framework permet de gagner du temps de développement sur des tâches qui ont déjà été faite par d'autre, et souvent mieux réalisées et validées par la communauté. PHP est un langage populaire et accessible. Facile d'installation et riche en fonctionnalités, un code en PHP peu cependant vite devenir complexe et confus. PHP n'encourage pas à organiser son code et rien ne l'oblige à le faire. Des bonnes pratiques sont bien souvent prises, par exemple pour gérer les pages d'un site internet de façon dynamique, en créant des dossiers. Divers frameworks PHP sont disponibles. Parmi eux, on peut citer les deux frameworks les plus connus : Symfony et Laravel.

Même si le framework Symfony avait été pratiqué auparavant par les stagiaires, le choix s'est porté sur Laravel. Il s'agissait d'abord d'une opportunité pour apprendre à utiliser un nouveau langage, bien que les deux frameworks soient similaires. Sa première version étant sortie en 2013, Laravel reste un framework plus récent que Symfony, dont la première version est sortie en 2005. Les deux frameworks sont cependant tous deux mis à jour régulièrement avec de nouvelles versions et sont suivis par une grande communauté. Laravel dispose néanmoins de fonctionnalités supplémentaires. Il est bon de noter que le créateur de Taylor Otwell s'est inspiré du framework Symfony afin de créer Laravel. Il a, entre autres, amélioré le système de routage déjà initié dans Symfony. Le framework se base lui-même sur des fonctionnalités déjà existantes, comme l'envoi de mail qui se fait avec la bibliothèque SwiftMailer. On retrouve ainsi diverses fonctionnalités : la validation, la pagination, l'authentification, le système de migrations pour les bases de données, etc…

Une interrogation subsistait encore autour de Laravel Nova et notamment sur son achat puisqu'il s'agit d'un outil payant. Conçu par le créateur de Laravel, Laravel Nova est un outil d'administration (Administration Panel) basé sur le framework. Il dispose alors de toutes les fonctionnalités dispensées par ce cadriciel PHP, en plus de proposer un CRUD complet. Un CRUD est un acronyme anglais des quatre opérations de base de la gestion des données et applications :

- Create : créer ;
- Read : lire ;
- Update : mettre à jour ;
- Delete ou Destroy : supprimer.

Il résume les principales fonctions utilisateurs nécessaires pour créer et gérer des données. Les framework CRUD, comme Laravel Nova, facilitent le développement et l'utilisation des applications. L'admin panel permet ainsi de mettre en place différents éléments de façon simplifiée. En effet, Nova implémente de nombreuses fonctionnalités qu'il ne reste qu'à mettre en place et à adapter aux besoins du projet. La phase de développement de l'interface utilisateur est largement écourtée, économisant donc du temps pour d'autres tâches importantes du projet. Nova propose donc un CRUD complet qui n'attend qu'à être configuré et personnalisé. Cette alternative apporte un gain de temps significatif au vu de la courte durée du stage.

Nous le disions, les frameworks favorisent l'organisation du code et tendent à le rendre moins complexe et plus lisible. Laravel et Laravel Nova utilisent le modèle MVC (Modèle-Vue-Contrôleur), qui permet d'organiser le code. Cette architecture de développement permet de séparer le code source en modules. Le modèle MVC, comme son nom l'indique, est constitué de trois couches :

- Le modèle, qui contient la logique métier. Il assure la gestion des données manipulées par le programme, par exemple, dans le cas général d'une base de données, cette-dernière est contenue dans le modèle. Le modèle offre donc la possibilité de mettre à jour ces données, d'en insérer de nouvelles ou encore de les supprimer ;
- La vue, qui constitue la partie graphique de l'application et regroupe tout ce qui a trait à la présentation des données ou des interactions utilisateur. Il fait l'interface avec l'utilisateur. La vue affiche les données récupérées auprès du modèle et reçoit les actions de l'utilisateur (clic de souris, sélection, boutons…). Ces actions sont ensuite transmises au contrôleur ;
- Le contrôleur, qui répond à des interactions utilisateur en provenance de la vue. Il est donc chargé de la synchronisation du modèle et de la vue : le contrôleur reçoit les événements de l'utilisateur et enclenche les actions à effectuer. Si l'action enclenchée est liée aux données, le contrôleur fait appel aux méthodes pour modifier, ajouter ou supprimer les données concernées. Les appels aux traitements se font au travers des méthodes constituant les contrôleurs.

Le schéma reprenant et synthétisant les différentes interactions entre les trois couches de l'architecture MVC, le modèle, la vue et le contrôleur, se situe en annexes (Annexe n°3).

En plus d'intégrer le modèle MVC, Le framework est capable de communiquer avec les tables de la base de données, en passant par les modèles qui sont une représentation objet de chacune des tables. Ce passage, entre le code et la base données, utilise un concept appelé le mapping objet-relationnel ou ORM (acrocyme anglais pour object-relationnal mapping). Le système ORM intégré à Laravel s'appelle Eloquent. Un modèle créé dans Laravel permettra ensuite de créer une table sur la base de données. Chaque table correspond à un modèle qui permet d'intéragir avec la table depuis le code. Le modèle permet d'interroger la table, on peut ainsi ajouter, supprimer et modifier les données de la table correspondante. Un seul fichier de configuration suffit à établir une connexion à la base de données, le fichier *config/database.php*.

Diverses architectures fonctionnelles ont été proposées dans le cahier des charges. Trois architectures ont été décrites et détaillées avant de réaliser les missions. Celles-ci ont été utilisées en référence tout au long du projet. La première architecture est celle considérée comme idéale et qui sera à atteindre à long termes. La seconde est une alternative, dans le cas de figure où des webservices ou API sont utilisés. Enfin, la dernière architecture, pensée à court termes, est celle de référence pour ce stage. Comme précisé précédemment et dans le cahier des charges, l'objectif du stage est de

concentrer sur le téléchargement des données, leur traitement ainsi que leur stockage dans le Data Warehouse. Vous trouverez les schémas de ces architectures, extraites du cahier des charges, en annexes (Annexe n° 4).

L'architecture idéale et complète propose ainsi le téléchargement (ou *upload*) des données, se faisant sous l'action humaine ou automatisée. En effet, la tâche d'upload des fichiers des différentes entreprises est régulière. Cette tâche pourrait donc être automatisée. Une potentielle utilisation de l'outil UI Path avait été évoquée. Ce logiciel permet, en effet, de créer des robots exécutant les automatisations souhaitées. Les tâches à automatiser sont alors définies au sein du logiciel. Les robots créés peuvent être ainsi déployés dans les applications de l'entreprise. Une tâche automatisant le téléchargement des fichiers de toutes les sociétés du groupe DLJ aurait pu être créée et déployée ensuite dans l'application web métier ETL. A long termes, cette solution serait à envisager afin de maximiser les avantages en termes de coûts et de performances des activités de l'entreprise. Les données téléchargées sont ensuite stockées sur une base de données temporaire. L'utilisation d'un web service Amazon (AWS), dans cette architecture, est envisagé. Un service de stockage simple, tel que le service Amazon S3, permet aux données d'être stockées et récupérées de manière simple et pratique. Ce service performant offre une disponibilité et une sécurité des données, faisant de ce service une solution efficace et avantageuse. Pour ce qui est du traitement des données, l'architecture utilise le logiciel Talend que nous avons décrit auparavant. Enfin, le Data Warehouse, plutôt que d'être basé sur Maria DB ou MySql laisserait place à Snowflake, un Data Warehouse basé sur le cloud. Le Data Warehouse est une solution parfois complexe, coûteuse et qui peut être mal sécurisée. Snowflake propose alors une solution SaaS (les utilisateurs n'ont pas recours à l'installation, la configuration et la gestion du logiciel). Snowflake prend en charge le tout et une offre un Data Warehouse sécurisé sur une infrastructure de Cloud. Notons que cette solution est compatible avec les Clouds AWS, donc avec le service Amazon S3 utilisé pour stocker les données uploadées. Cette architecture complète représente la cible idéale à mettre en place afin de répondre parfaitement aux besoins du projet.

La seconde architecture est très proche de la précédente. Il s'agit simplement d'une alternative, où les données sources sont pourvues d'Api ou de web services.

Enfin, l'architecture fonctionnelle utilisée pour la réalisation des données ne fait intervenir aucune automatisation pour la récupération des fichiers sources, qui se fera donc manuellement. Le stockage des données uploadés se fait sur un hébergeur de données en cloud, Digital Ocean. Le traitement des données se fait au travers de scripts composant l'application web métier ETL. En réalité, le système de gestion de base de données (SGBD), MariaDB, remplace le Data Warehouse basé sur le cloud.

Pour toutes les architectures définies, les données traitées seront exploitées avec le logiciel Readsh que nous avons présenté précédemment. Des graphiques pourront ainsi répondre aux besoins de l'entreprise.

Après analyse du cahier des charges et étude l'existant, en conclusion de l'étude technologie, les étapes du projet ont été clairement définies. Le projet a pu être découpé en cinq étapes principales :

- La première, qui est en fait une étape préliminaire : la récupération des données. Les fichiers sources doivent être récupérés des différentes sociétés ;

- La seconde : le téléchargement des données. L'application devant être développée doit implémenter une fonction d'upload permettant de stocker les fichiers sources avant qu'ils soient traités. Ces fichiers seront stockés dans une base de données MySql gérer sous MariaDB reliée à l'application. La conception de la base de données sera évoquée dans la partie réalisation technique (cf. p. 15) ;

- Le traitement constitue la troisième étape du projet. Il s'agit du plus gros travail. Une fois les données uploadées, il faut les transformer, les traiter et les nettoyer. C'est ici qu'on peut faire appel aux processus ETL. Les données doivent être extraites des fichiers (on doit pouvoir les

lire), avant de pouvoir les modifier afin de les stocker ensuite. Par exemple, il est envisageable de créer une vue dans l'application, sous forme de formulaire, permettant à la fois de lire et de modifier les données ;

- Une fois les données traitées, l'avant-dernière étape est de stockée ces données « propres » dans un Data Warehouse. Celui de l'entreprise sera utilisé. Une base de données a été créée à cet effet ainsi que deux utilisateurs. Un premier utilisateur, avec des droits restreints – droit de lecture uniquement, sera utilisé pour l'exploitation de données sur Redash.
- La dernière étape, deuxième mission du stage, correspond à l'exploitation des données traitées sous Redash. La base de données avec les données traitées est reliée à Redash. Pour des raisons de sécurité, seule la lecture des données sera possible. Cette action suffit à l'affichage des données sous formes de diagrammes, graphiques ou autres représentations disponibles sous le logiciel de dashboarding.

Vous trouverez en annexe n°5, un schéma récapitulant les étapes de la réalisation technique. Nous détaillerons comment ces étapes ont été réalisées dans la partie qui suit.

* * *

IV – REALISATION TECHNIQUE ET MESURES

Cette partie consiste à expliquer comment les missions ont été réalisées, avec quels procédés. Les réalisations seront ensuite évaluées et validées, ou non.

A – REALISATION TECHNIQUE

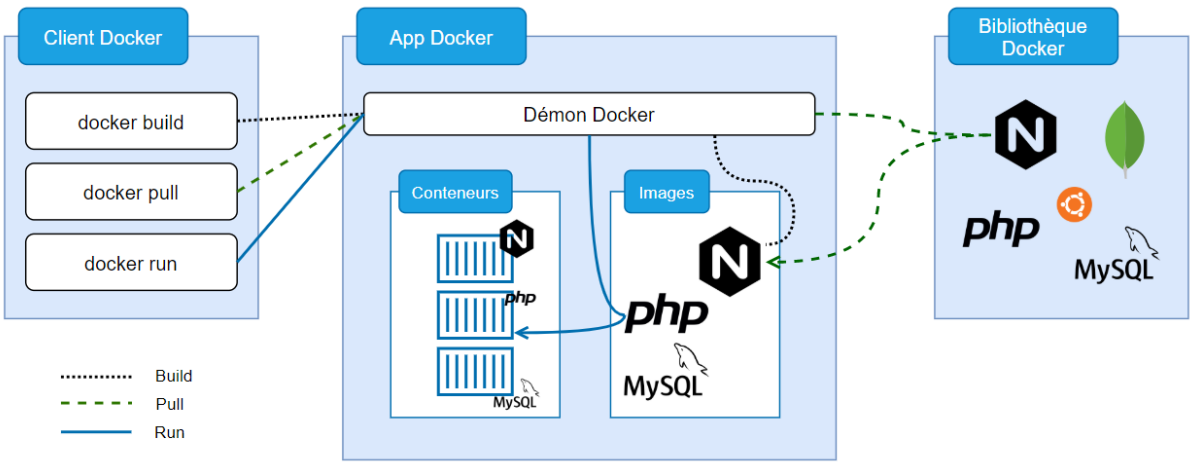
Après avoir longuement conceptualiser l'application et son architecture, la phase de réalisation peut débuter.

La première mission donnée résidait dans la conception d'une application web métier ETL permettant une gestion de données. L'application a été développée avec l'environnement de développement Visual Studio Code. Cet IDE (acronyme anglais pour Integrated Development Environment) offre de nombreuses fonctionnalités avantageuses comme une gestion facilitée des containers Docker (avec une extension à ajouter), la gestion du versionnement sous Git (toujours avec une extension à ajouter). Facile à prendre en main et proche de l'IDE Visual Studio (IDE déjà utilisé), aucun temps d'adaptation a été nécessaire. Cet environnement de développement a été choisi car il était déjà installé sur les machines des stagiaires. Le DSI et référent du projet utilise cet IDE, cela permettait donc d'obtenir une unité et simplicité dans le développement de l'application, en plus de profiter des fonctionnalités intéressantes du logiciel.

Le système gestion des versions, Git, a été utilisée pour ce projet. Git est logiciel de gestion de versions décentralisé. Il permet de gérer les différentes versions d'un même fichier. Ainsi, on obtient

une réelle traçabilité des fichiers tout au long du projet. L'équipe a travaillé avec le logiciel GitLab. Il existe d'autres logiciel de gestion des versions utilisant le logiciel Git, comme Azure DevOps (proposé par Microsoft), GitHub ou encore BitBucket, qui proposent principalement les mêmes services. Cependant, GitLab est celui utilisé dans l'entreprise, d'où l'utilisation de ce logiciel plutôt qu'un autre.

Afin de gagner en stabilité et en rapidité, le projet s'est appuyé sur la plateforme de containérisation Docker. Docker est un logiciel open source permettant d'embarquer une ou plusieurs applications et leurs dépendances dans des conteneurs. Cela facilite leur déploiement. Un des avantages à utiliser Docker, si ce n'est son principal avantage, est que l'on peut facilement transférer et partager les conteneurs d'un environnement à l'autre. L'application fonctionne rapidement et de manière fiable d'un environnement informatique à l'autre. Si l'on veut comprendre le fonctionnement de Docker, il est essentiel de comprendre ce que sont les conteneurs et les images. Un conteneur correspond à une unité d'un logiciel. Il contient du code ainsi que les dépendances nécessaires à son bon déploiement. L'ensemble de logiciels léger, autonome et exécutable comprenant tout ce qui est nécessaire pour exécuter l'application est ce qu'on appelle l'image d'un conteneur. L'image comprend ainsi le code, l'exécution, les outils système, les bibliothèques système et les paramètres. Une fois exécutée avec la commande *docker run*, l'image devient un conteneur. La conception de l'architecture Docker a été réalisée avec le référent du stage. Le schéma ci-dessous présente l'architecture Docker, qui s'applique au projet.



L'application a besoin d'un serveur web, d'un serveur de base de données ainsi que l'application PHP pour fonctionner correctement. Ces trois services correspondent chacun à un container Docker. Docker rassemble un très grand nombre de logiciels dans sa bibliothèque (Ubuntu, PHP, Nginx, MySql, MongoDB, …). Docker dispose de son propre outil de gestion de package : Docker Compose. Cet outil permet de lancer nos différents conteneurs à partir d'un même fichier de configuration écrit en yaml. Docker Compose simplifie l'utilisation de Docker. Docker construit (build) les images automatiquement à partir des instructions lues dans le *dockerfile* (fichier texte) – cf. un extrait du fichier dockerfile en annexe n°6. Les images Docker que l'on souhaite mettre en place sur notre application sont donc référencées dans le fichier dockerfile. Le fichier *docker-compose.yaml* rassembles les conteneurs de l'application et permet de lancer l'ensemble des conteneurs en une seule commande *docker-compose up -d*. Cette commande permet de lancer, en arrière-plan les conteneurs Docker décrits dans le fichier *docker-compose.yaml*. Un extrait du fichier dockerfile se trouve en annexe n°6.

Nous le disions, trois conteneurs ont été créés à partir de trois images Docker de services Open Source :

- Le serveur web, il s'agit d'un serveur Nginx. Open Source et utilisé par de nombreuses entreprises comme GitLab, Google ou Microsoft, Nginx offre une faible utilisation de la mémoire et utilise une approche asynchrone et événementielle. En effet, l'avantage de ce serveur web est qu'il traite les requêtes web en une seule tâche, un seul processus. Le choix du serveur web a été fait par Anh DAO-DUY.

- Le serveur de base de données, un serveur MySql. Open Source et largement répandu, MySQl est un système de gestion de bases de données. Ce système permet de stocker et gérer les données des bases de données qu'il contient.

- Le service PHP. Rappelons que PHP est un langage de scripts Open Source et généralement utilisé pour le développement d'applications web. Il était donc nécessaire d'intégrer le service PHP à l'architecture du projet afin de pouvoir exécuter les scripts PHP sur l'application.

Laravel Nova dispose d'un système de ressources. Les ressources sont créées à partir d'un modèle. Chaque ressource Nova correspond donc à un modèle Eloquent (définit précédemment, voir p. 13). Les ressources Nova se trouvent dans le dossier *app/Nova*. Pour créer une ressource, il faut effectuer la commande suivante :

```
php artisan nova:ressource [NomDeLaRessource]
```

Avant de commencer le développement même de l'application, il a fallu réfléchir au schéma de la base de données. La base de données devait répondre aux besoins de l'application :

- L'ajout, la modification et la suppression d'un utilisateur ;
- La gestion des rôles des utilisateurs ;
- L'ajout, la modification et la suppression d'une société ;
- L'ajout, la modification et la suppression d'un upload ;
- L'ajout, la modification et la suppression d'un type de fichier (extension du fichier téléchargé) pour un upload ;
- L'ajout, la modification et la suppression d'un tag ;
- Attribuer un type de fichier à la création d'un upload ;
- Attribuer un ou plusieurs tag(s) à un même fichier uploadé.

Six tables étaient donc nécessaires afin de mettre en place les premières fonctionnalités de l'application :

- *users* : cette table est créée initialement par Laravel Nova ;
- *data_source_entries* : table correspondant un upload ;
- *data_source_types* : il s'agit du type de fichier uploadé, trois types seront ajoutés par défaut (CSV, TXT, XML) ;
- *tags* : table correspondant à un tag, c'est-à-dire une catégorie, auquel l'upload renvoie. Cette table sera à relier à la table *data_source_entries* en passant par une table relationnelle puisqu'il s'agit d'une relation many-to-many. Un même fichier uploadé peut avoir plusieurs tags et un même tag peut être attribué à plusieurs fichiers téléchargés ;
- *data_source_entry_tags* : cette table est le table pivot entre les tables *data_source_entries* et *tags* ;
- et *companies* : table correspondant aux sociétés du groupe DLJ dont les données seront traitées. Un upload sera donc relié à une société.

Le schéma de base de données est disponible en annexes, annexe n° 7 (voir p. 30).

Une fois la modélisation de la base de données réalisée, les ressources ont pu être créées. L'authentification des utilisateurs est gérée par Nova, elle peut simplement être personnalisée. La gestion du rôle de l'utilisateur (administrateur ou simple utilisateur) a été faite par le seconde développeur de l'équipe. Ainsi seul un administrateur a donc accès à la ressource *User* sur l'application et peut ajouter, modifier ou supprimer un autre utilisateur. Les autres ressources ont été créées.

Pour chaque ressource, Nova intègre des fonctions basiques : *create*, *read*, *update* and *delete* ; qui rappellent l'acronyme CRUD désignant les quatre opérations de base pour stocker et gérer les données en base de données. L'interface utilisateur est construite à partir des fonctions et de la ressource créées. La fonction *fields* de la ressource permet d'ajouter les données que l'utilisateur devra remplir pour ajouter une nouvelle entrée. Vous trouverez en annexe n° 8 (voir p. 30) la fonction *fields* de la ressource *DataSourceEntry* ainsi que le formulaire permettant de créer une Data Source Entry (un nouvel upload).

Afin d'ajouter les tables dans la base de données il faut réaliser une migration. Les migrations sont comme un contrôle de version de la base de données. Ce système permet à toute l'équipe d'un même projet de modifier et de partager le schéma de base de données de l'application. Les migrations sont associées aux ressources et donc aussi aux modèles. On peut ainsi ajouter une colonne manquante en cours de projet à l'aide d'une migration. C'est, par exemple, ce qui a été fait pour ajouter la fonction de soft delete à la table *companies*. Le soft delete permet d'enregistrer la suppression d'un élément et en le rendant indisponible à la sélection ou à l'affichage. L'élément n'est pas pour autant supprimé de la base de données. Le soft delete est très bien géré par Laravel Nova, avec une fonction « prête à l'emploi » qu'il suffit d'ajouter à la ressource. On peut cependant toujours supprimer définitivement un élément, une autre fonction de suppression est disponible. Ainsi, sur l'interface de l'application, deux boutons de suppression sont disponibles : un pour supprimer l'élément en le gardant enregistré dans la base de données et un second pour le supprimer définitivement des données.

Une seconde tâche importante a été attribuée : la gestion des relations de tables par Laravel. En effet, après avoir créé les ressources *DataSourceEntry*, *DataSourceType*, *Tag* et *Company*. Il a fallu les lier afin de permettre à l'utilisateur d'ajouter un type, une société et un tag à un même fichier uploadé. Les relations entre chaque ressource sont définies dans leur modèle respectif. Le système Eloquent intégré à Laravel Nova gère automatiquement quatre principales relations :

- **One To One** : relation la plus basique, cette relation n'existe pas dans l'application ;
- **One To Many** : cette relation est utilisée, par exemple, entre les ressources *DataSourceEntry* et *DataSourceType*. Un fichier uploadé appartient à un seul type mais un type appartient à plusieurs fichiers uploadés. Le modèle *DataSourceType* utilisera la fonction *hasMany()* (voir ci-dessous) signifiant ainsi que le modèle a plusieurs entités de type *DataSourceEntry* :
- **One To Many (Inverse) :** cette relation permet, par exemple, au modèle *DataSourceEntry*, d'accéder à son *DataSourceType* parent. Le modèle *DataSourceEntry* utilise la fonction *belongsTo()*, appliquée au modèle *DataSourceType*, signifiant ainsi qu'une *DataSourceType* appartient à plusieurs entités *DataSourceEntry*. Ci-dessous, les deux fonctions permettant de créer la relation complète One To Many entre les tables data_source_entries et data_source_types.

```php
//app/DataSourceType.php
public function data_source_entries()
{
    return $this->hasMany('App\DataSourceEntry');
}
```

```php
//app/DataSourceEntry.php
public function dataSourceType()
{
    return $this->belongsTo('App\DataSourceType');
}
```

Cette même relation existe entre les tables *companies* et *data_source_entries*.

- **Many To Many** : cette relation est plus complexe. Elle existe entre les tables *data_source_entries* et *tags* : on peut ajouter plusieurs tags à un même fichier uploadé, mais un même tag peut être ajouté à plusieurs fichiers uploadés. Le modèle *DataSourceEntry* utilise alors la fonction belongsToMany(). Appliquée au modèle *Tag*, elle signifie qu'une même entité DataSourceEntry peut avoir plusieurs tags. Quant au modèle *Tag*, il implémente la même fonction *belongsToMany()* appliquée cette fois au modèle *DataSourceEntry*. Ainsi, l'utilisateur peut ajouter plusieurs tags pour un même fichier uploadé. Les tags attribués à une DataSourceEntry s'affichent sur la vue de détails de la ressource dans une card (dans un encadré prédéfini par Laravel Nova) à part.

```php
//app/DataSourceTag.php
public function dataSourceEntries()
{
    return $this->belongsToMany(DataSourceEntry::class);
}
```

```php
//app/DataSourceEntry.php
public function tags()
{
    return $this->belongsToMany('App\Tag');
}
```

Vous trouverez en annexe n°9, les différentes interfaces de l'application ainsi que le résultat graphique des fonctions appliquées ici.

La méthode permettant le téléchargement d'un fichier a été fait par le deuxième développeur de l'équipe. La tâche suivante consistait à lire et afficher le contenu d'un fichier CSV. Plusieurs méthodes ont été employées afin d'ajouter cette fonctionnalité à l'application. Des packages Nova existent et permettent de lire les données d'un fichier, cependant aucun d'entre eux n'a pu être parfaitement intégré à l'application. Le choix de tout faire à la main a donc été pris, mais tardivement, diminuant ainsi le temps de développement de cette fonctionnalité. Un script en PHP a été réalisé. Ce script implémente une fonction PHP permettant de traverser et lire les données d'un fichier PHP. Cependant le rendu n'a pas été optimisé (cf. annexe n°9, pour voir le visuel). Afin d'afficher le retour de la fonction, il a fallu créer une nouvelle card Nova. Un package Nova, permettant de créer des cards sous forme de tableau, a été testé. Cependant, il était compliqué d'insérer le résultat de la fonction dans le tableau. Un autre package php, php-etl, a été testé. Celui-ci rassemblait toutes les fonctions recherchées pour mettre en place les fonctionnalités au sein de l'application. Il intègre une fonction extract(), capable d'extraire les données d'un fichier csv. Une fonction permettant de transformer le fichier, d'enlever ou de modifier des caractères. Enfin, il permet, grâce à sa dernière fonction *load()*, de charger les données du fichier dans une table de base de données. Ce package répond parfaitement aux besoins de l'application et reprend les principes de base de l'ETL. Malheureusement, son intégration au sein du projet n'a pas été concluante. Le script PHP final est visible en annexe n°10. Le résultat de la fonction s'affiche dans une *card* personnalisée, sur la page de détails de la DataSourceEntry.

La deuxième mission attendue est celle de l'exploitation des données traitées sous Redash. Celle-ci s'est faite à partir de données nettoyées, sans utiliser l'application en développement, puisque cette-dernière ne le permet pas encore. A partir des données du Data Warehouse, des requêtes sql ont pu être réalisées. La prise en main de Redash n'a pas été longue, l'outil est simple d'utilisation et très intuitif. Plusieurs requêtes sql ont été réalisées avec toute l'équipe du projet. Chaque développeur a ensuite été missionné d'une requête à exécuter. Des requêtes sql ont été écrites afin d'obtenir des données et statistiques sur la nouvelle salle de sport du groupe DLG. Ayant ouverte récemment et étant un changement d'enseigne, il était important d'avoir des statistiques sur cette société, c'est pourquoi le choix s'est porté sur la société Fitness Prime. Des requêtes existaient déjà sur le nombre d'hommes et de femmes inscrits, sur le nombre d'inscrits auparavant et réinscrits ou encore le nombre

de désinscrits et de nouveaux inscrits. L'objectif de cette mission était donc d'écrire de nouvelles requêtes afin d'obtenir des graphiques apportant de nouvelles informations.

Chaque société du groupe peut avoir un dashboard sur Redash : Akenat et Fiteness Prime en possède chacun un. Le dashboard de DLJ a été créé afin d'y intégrer une requête. Un premier graphique apparaît donc sur Redash et affiche le nombre d'heures passées par activités au sein de l'entreprise. Cette première requête a été faite avant de créer de nouveaux graphiques sur le dashboard de Fitness Prime. Chacun des deux développeurs devaient choisir une requête à faire. Nicolas DURET, développeur sur le projet, a choisi d'obtenir le nom de la personne, hommes et femmes séparés, qui a le plus participer à chacun des cours collectifs proposés. Il a choisi de renvoyer le top dix des clients. Ces informations ont fait naître certaines idées, comme la création d'un palmarès : élire le client et/ou la cliente du mois ou de la semaine. Le requête choisie et réalisée devait afficher le nombre d'inscrits à chacun des cours collectifs dans l'ordre décroissant. Une seconde requête, plus complexe, a été réalisée. Elle devait renvoyer le nom, prénom, civilité et identifiant unique des clients ainsi que le nombre d'activités auquel le client à participer. Les données devaient être affichées par ordre décroissant en fonction du nombre d'activités. Cependant, la complexité résidait dans la récupération de l'identifiant du client. En effet, l'enregistrement d'un client à un cours collectif s'affiche dans une colonne avec le nom, prénom et identifiant du client, chacun séparé par des caractères tels qu'un tiret, une barre oblique ou encore un dièse devant l'identifiant. Sa nomenclature demandait donc un travail de transformation afin de pouvoir afficher l'identifiant des clients ainsi que leur nom et prénom séparément. Cette transformation est possible en utilisant les fonctions SQL suivantes :

- *TRIM()* : cette fonction permet de supprimer des caractères au début et en fin de chaîne de caractères. Cette fonction est très utilisée pour supprimer des caractères tels qu'une tabulation, un espace ou un retour à la ligne ;
- *SUBSTRING_INDEX()* : cette fonction est utilisée pour segmenter une chaîne de caractères. Elle permet ainsi d'extraire ou partie d'une chaîne. Elle passe en paramètre une chaîne de caractères, un délimitant (le caractère recherché) et un nombre (nombre d'occurrence du délimitant). Ici, cette fonction permet de tronquer la chaîne de caractères regroupant le nom, prénom et identifiant des clients séparés par un tiret. Elle permet aussi de ne récupérer seulement l'identifiant unique du client sans le dièse ;
- *REPLACE()* : cette fonction permet tout simplement de remplacer des caractères alphanumériques dans une chaîne de caractères. Elle comporte trois paramètres : la chaîne d'entré, le texte à remplacer et le texte servant de remplacement. Cette fonction est principalement utilisée pour mettre à jour des données dans une base de données ou pour afficher des résultats personnaliser comme c'est le cas ici.

Afin d'obtenir le résultat souhaité, il a fallu créer un vue SQL. Une vue peut être considérée comme une table virtuelle dans la base de données, définie par une requête. Une vue intitulée *clients_by_activities* a été créée revoyant les données des clients inscrits pour chaque cours collectif. C'est dans cette vue que les fonctions citées précédemment ont été utilisées. La requête permettant d'obtenir le nom, prénom, identifiant et nombre de cours effectués pour chaque client, n'a donc plus qu'à utiliser cette table virtuelle contenant les données recherchées. On utilise une fonction d'agrégation SQL *COUNT()*, capable de compter le nombre d'enregistrement dans une table, pour compter le nombre de d'activités auxquelles chaque client à participer. Vous retrouverez la requête réalisée, ainsi que les graphiques créés, en annexe n° 11.

Nous venons de détailler comment les missions ont été réalisées. Ces réalisations ont été, à la fin du stage, mesurées et évaluées en fonction des besoins définis au début du projet.

B – MESURES

Rappelons que le projet devait respecter un délai de cinq semaines. Le langage PHP était connu des deux développeurs mais le framework Laravel Nova ne l'était pas. Une période d'apprentissage et de montée en compétence a donc été nécessaire. La période de conception, comprenant l'état de l'art et l'étude technologique a permis de mettre en place le projet et chercher les solutions permettant de répondre au mieux aux besoins de l'application à développer. L'application a été rendue fonctionnelle mais avec les premières fonctionnalités demandées. Il manque des fonctionnalités à l'application afin qu'elle remplisse entièrement le cahier des charges. La partie ETL de l'application demande un travail supplémentaire.

La première amélioration qui sera à apporter au projet est sur la partie de traitement des données. L'affichage des données n'est pas fonctionnel, il sera nécessaire de revenir sur la technique d'affichage en utilisant une autre méthode. Il serait nécessaire d'accentuer le travail sur le package php-etl, qui, malgré son implémentation et les tests ayant été réalisés, semble répondre aux besoins. Nous verrons dans le prochain chapitre quels ont été les outils mis en place afin de planifier au mieux ce projet et de respecter le délai.

Pour ce qui est de la seconde mission, c'est-à-dire l'exploitation des données, les attentes ont été remplies. La prise en main sur l'outil Redash a été très rapide. Des requêtes permettant d'avoir des chiffres sur l'activité du club de sport ont, entre autres, été faites. L'objectif de cette mission, rappelons-le, était de permettre aux équipes de l'entreprise d'avoir un retour sur les activités des sociétés (le nombre de clients par cours de sport, le nombre de femmes et d'hommes inscrits, etc…).

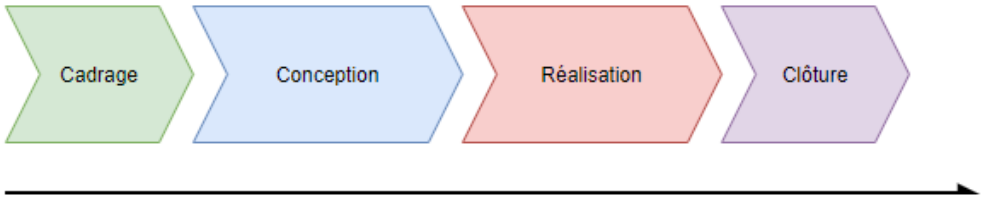
Cependant, outre les fonctionnalités manquantes, l'application est fonctionnelle et est performante. Elle répond aux premiers besoins essentiels : un utilisateur peut se connecter à l'application et peut, par exemple, créer un upload. La gestion des rôles utilisateurs a été gérée, permettant de différencier un administrateur et un simple utilisateur. L'administrateur peut ajouter, modifier ou supprimer un utilisateur. Les interfaces de l'application sont disponibles en annexe n°9.

* * *

V – GESTION DU PROJET

Cette partie reprend les outils et méthodes mis en place au cours du stage afin de gérer le projet. Leurs emplois seront expliqués et justifiés.

Comme dans tout projet, une gestion de projet a été mise en place afin d'organiser au mieux le déroulement du projet et atteindre les objectifs. Rappelons que le projet était constitué de trois personnes : un chef de projet et deux développeurs juniors.



Le projet est constitué de 4 phases principales :

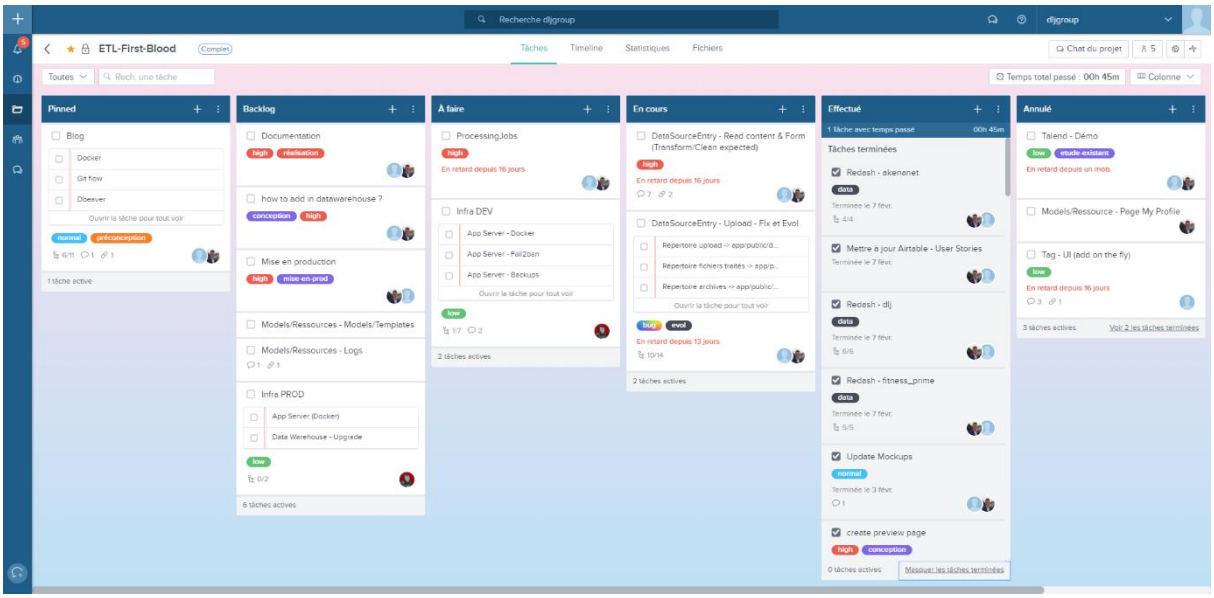
- La phase de cadrage : le projet est présenté à l'équipe. Les outils de gestion de projet sont mis en place. On dresse le cahier des charges qui rassemble les besoins de l'application à réaliser.
- La phase de conception, durant laquelle on étudie le projet : les outils pouvant être utilisés, ce qui existe déjà, les besoins dont il est question. On peut réaliser un backlog. La constitution du backlog débute par la matérialisation et la définition des objectifs de l'application, des utilisateurs cibles (c'est-à-dire les employés du groupe DLJ) et les acteurs du projet. On dresse ensuite une liste d'exigences fonctionnelles ou non. Notons que le backlog évolue au cours du projet. A partir de ce backlog, des user stories peuvent ainsi être réalisées, rassemblant les diverses fonctionnalités que l'application doit contenir.
- La phase de réalisation, où le développement a réellement débuté. Les outils pris en main durant la phase de conception ont été utilisés.
- La phase dernière phase qui est celle de la clôture. Un bilan est réalisé avec toute l'équipe du projet. On évalue ainsi ce qui a été fait ou non, et on dresse le bilan des réalisations.

Divers outils ont donc été utilisés afin de gérer et planifier au mieux ce projet.

Un premier outil a été mis à disposition avant même le début du stage : Slack. Il s'agit d'une messagerie professionnelle. Slack permet ainsi d'échanger avec l'ensemble des membres du groupe DLJ (toutes sociétés confondues). Cet outil a donc permis, en plus de faciliter l'intégration au sein des équipes et la communication, de demander de l'aide en cas de besoin et de partager des ressources essentielles. Cet outil a notamment servi dans la gestion du projet.

Les users stories ont été faites avec l'outil Airtable. Cet outil est un outil de gestion, il permet de facilement gérer des données dans des tableaux proches d'Excel. Les tableaux de données peuvent facilement être exportés en différents formats, en CSV entre autres. Cet outil est utilisé par les équipes comme outil de management, permettant aux employés de remplir leur nombre d'heures passées sur les projets actuels. Airtable propose des templates de tableaux ppur réaliser des User Stories. Simple d'utilisation, les users stories ont pu être réaliser facilement. Vous trouverez en annexe n°12 (voir p.34) les user stories réalisées.

Pour la planification du projet, l'outil Taskworld a été utilisé. De plus en plus d'outils de gestion de projets existent. Taskworld est la solution de gestion de projet utilisée par les salariés de DLJ. Cet outil est complet et pratique à utiliser : il permet de gérer tous les aspects de la planification de projet. Une fois le projet créé sur Taskworld, on peut ajouter les membres de l'équipe. L'interface principale propose un tableau de projets, de tâches. Il permet d'organiser et de prioriser les tâches. Ces-dernières peuvent ensuite être attribuées aux membres du projet. L'efficacité de cet outil réside aussi dans sa messagerie instantanée intégrée. On peut ainsi partager des fichiers, commenter les tâches, donner un avis sur le travail d'un collègue ou encore aider ses collaborateurs. Tout se fait avec un même outil. Ajoutons que Taskworld, offre tableau analytique des projets : le temps passé par tâche, les retards sur les missions assignées. On a ainsi une vision d'ensemble du projet en termes de missions réalisées et de temps, ce qui permet d'évaluer la réalisation et la planification du projet. Il s'agit ci-dessous du tableau dans lequel toutes les tâches définies ont été répertoriées.

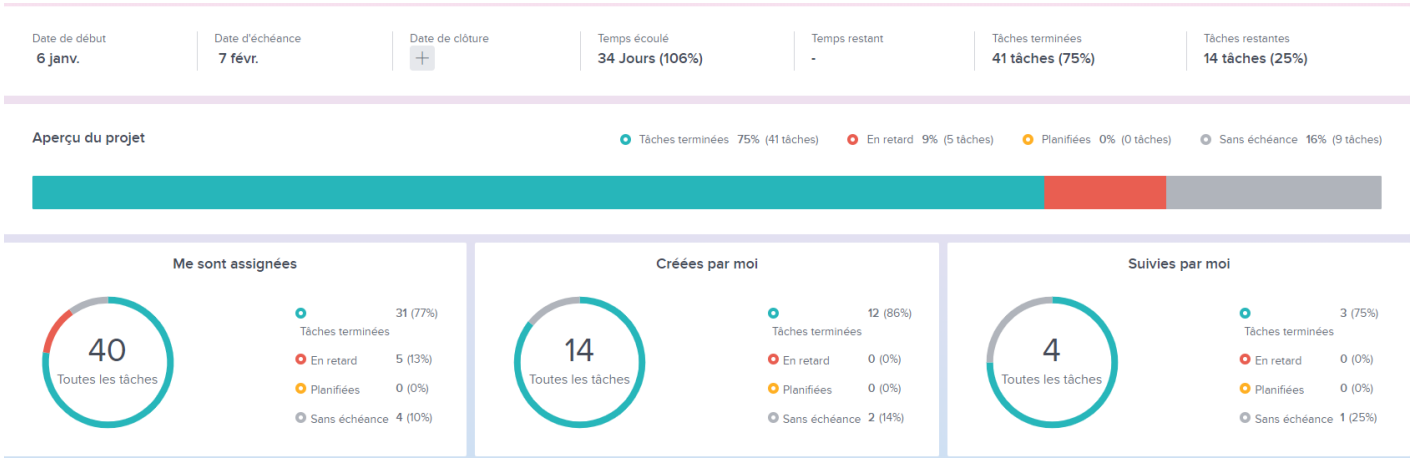


Concrètement, les tâches ont été distribuées entre les deux développeurs. Le chef de projet supervisait et validait les tâches. Il a choisi de diviser le tableau en six colonnes :

- Epinglé (ou Pinned), pour les tâches de fond à réaliser tout au long du stage ;
- Backlog, qui liste les besoins définis dans le cahier des charges que recueille backlog constitué en début de stage ;
- A faire, qui liste les tâches qu'il reste à faire ;
- En cours, pour les tâches qui sont en cours de réalisation ;
- Effectuées, où on y glisse les tâches effectuées ;
- Annulées, pour les tâches qui ont été annulées, faute de temps ou de moyens, pour une tâche trop complexe par exemple.

Si l'on regarde de plus près la timeline créée par Taskworld, on peut voir que certaines missions ont été réalisées avec quelques jours de retard. D'autres ont été annulées. Taskworld permet de tirer facilement des conclusions sur la gestion de projet. Si des tâches ont été rendues avec du retard c'est que le temps estimé n'était pas suffisant.

Le logiciel de gestion de projet donne accès à d'autres graphiques et d'autres statistiques comme l'aperçu du projet renseignant sur le pourcentage de tâches terminées ou rendues en retards. Selon lui, 75% des tâches ont été finies, laissant 9% de tâches en retards. Ces tâches n'ont pas été finies avant la fin du projet. Pour le reste des tâches sans échéance (16%), il s'agit de tâches de fond comme le remplissage du journal de bord, l'ajout d'article sur le blog ou autres. Ces dernières tâches ont été terminées après la fin du stage.



Les chiffres à retenir sont les pourcentages de tâches terminées (75%) et restantes (25%). Ces chiffres nous renseignent sur le taux de réalisation du projet. Le projet est considéré comme étant finit à 75%. D'autres diagrammes sont visibles en annexes (annexe n°13).

Pour rappel, il manque des fonctionnalités à l'application afin qu'elle réponde entièrement aux besoins. Il s'agissait de mettre en place les bases d'un projet à long termes : cet objectif est atteint. Cependant, la phase de traitement de données n'a pas été réalisée, d'où les 25% restant.

Malgré des retards et certaines fonctionnalités manquantes, les tâches accomplies remplissent les besoins de l'application et les fonctionnalités mises en place sur opérationnelles. Le projet a été rendu à temps. L'application reste à être développée afin de répondre entièrement au cahier des charges. Ce projet est de grande envergure et requiert une phase de développement plus longue.

IV – CONCLUSION

Ce mémoire avait pour ambition de résumer les missions faites durant cette période de stage et de traiter le vaste sujet que sont les échanges de données. Nous nous sommes alors demander si ces échanges de données informatisés était la clé de la stratégie d'entreprise. Le contexte dans lequel s'est déroulé le stage nous amener à cette problématique.

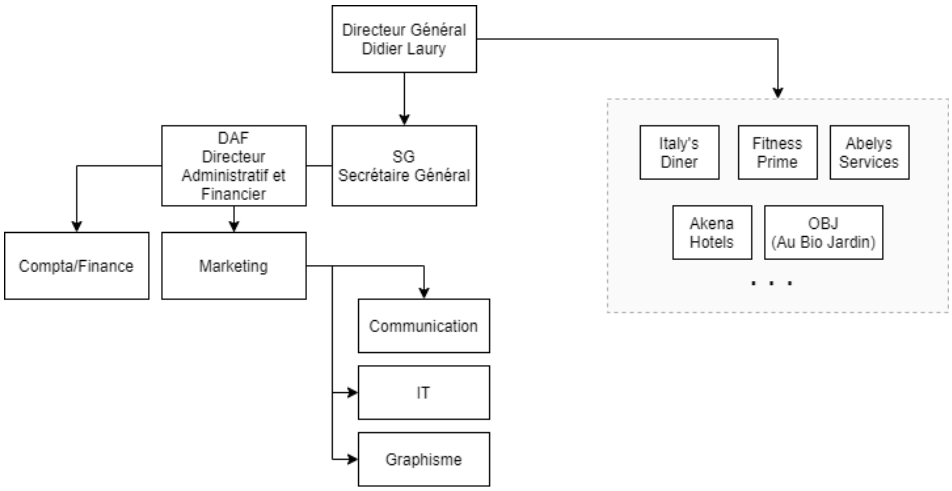
Les données sont aujourd'hui au cœur de toute entreprise et de toute application. Le nombre de données transitant au sein d'une entreprise par jour est considérable. Quant aux échanges ils représentent l'essence même de l'interaction, qu'elle soit humaine ou virtuelle. Les entreprises se doivent échanger, entre elles (d'une entreprise à une autre) mais aussi en interne (au sein de leur propre entreprise). Les échanges de données informatisés sont alors au cœur des entreprises. Le nombre de données transitant au sein d'une entreprise par jour est considérable.

Les données utilisées au sein des applications proviennent de sociétés et constituent leur identité. Les échanges de données informatisés sont effectivement, une des clés de la stratégie de l'entreprise, si ce n'est la plus importante. Les entreprises se doivent d'utiliser des données fiables, c'est donc à chaque entreprise d'instaurer et de mettre en place des outils permettant de gérer et traiter leurs données. DLG Groupe, en cherchant à développer cette application web, a cherché à répondre à ce besoin d'optimisation. Cette recherche d'efficacité et d'optimisation rejoint leur stratégie d'entreprise.

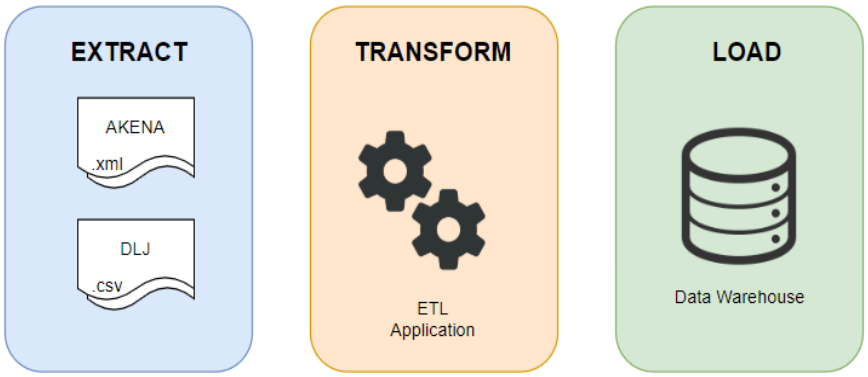
Ce stage a eu un impact plus que positif pour la poursuite d'études. De grands thèmes, comme la gestion des données et les processus ETL, ont été vus et traités. Ces thèmes seront retrouvés tout au long de la carrière de développeur. Être dans une société holding a permis de découvrir différents secteurs d'activités qui doivent chacun répondre à des besoins différents. Le travail d'équipe est aussi un point important qui a été valorisé au sein de ce projet. Les choix des technologies et des outils utilisés ont été faits en équipe, en collaboration. Rappelons que le projet a été commencé *from scratch* (c'est-à-dire depuis rien). Le projet naissait simplement d'un besoin et d'idées : tout restait à faire.

C'est aussi une satisfaction de faire partie d'un aussi grand projet, de plus que celui-ci s'est très bien déroulé. Mr. DAO-DUY, à la fois maître de stage, chef de projet et directeur du système d'informations, a su mener à bien ce projet et a su encadrer son équipe au mieux malgré ses obligations. Il a su se rendre disponible et apporter ses nombreuses connaissances techniques.

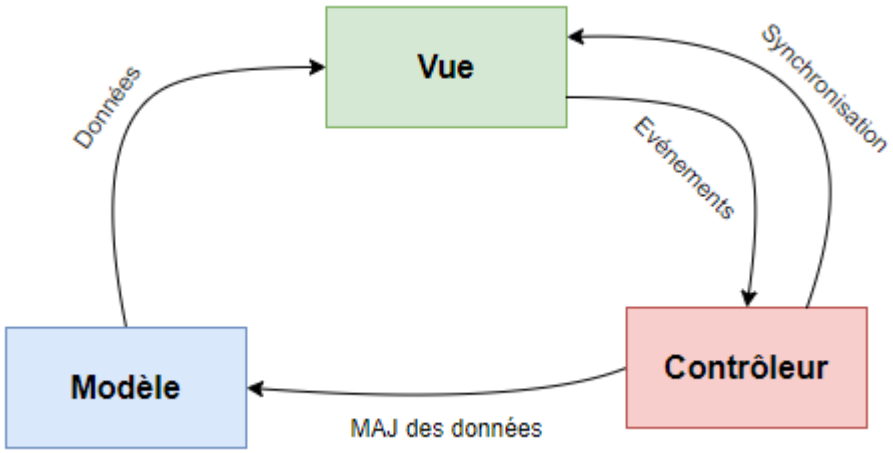
Annexe n° 1 – Organigramme de l'entreprise



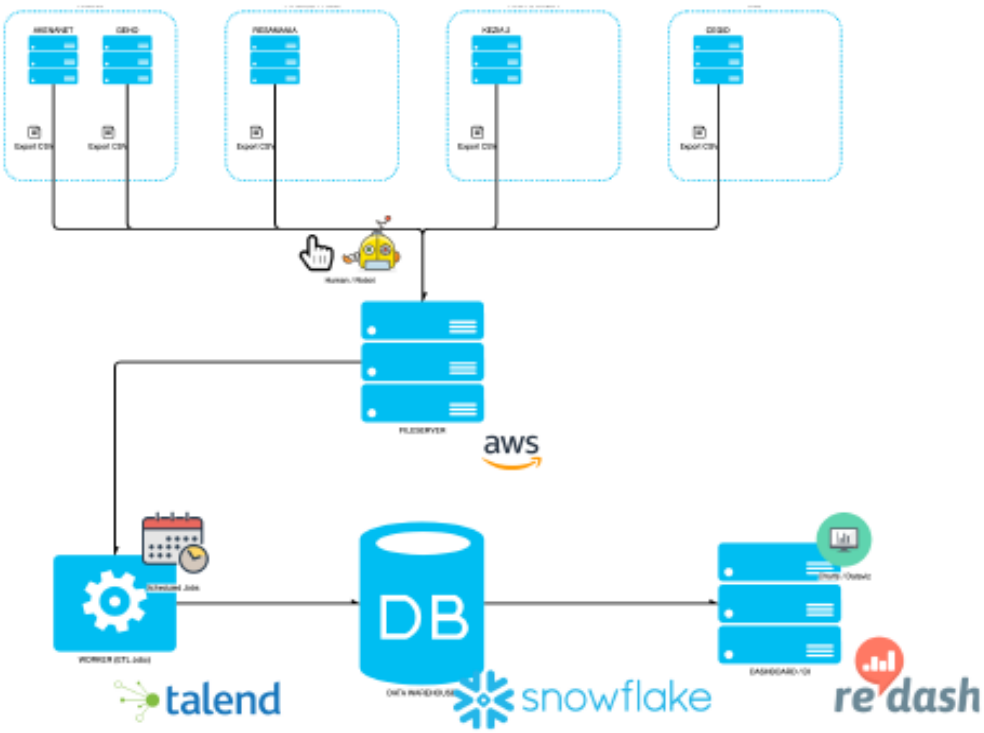
Annexe n°2 – Schéma récapitulatif du processus ETL



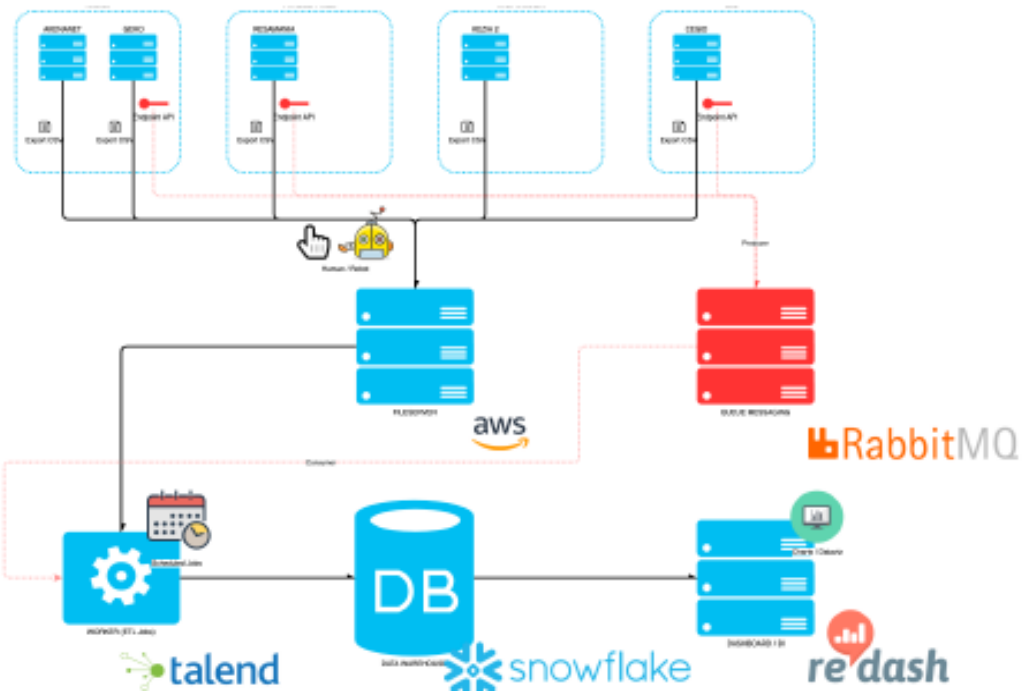
Annexe n°3 – Schéma explicatif du modèle MVC



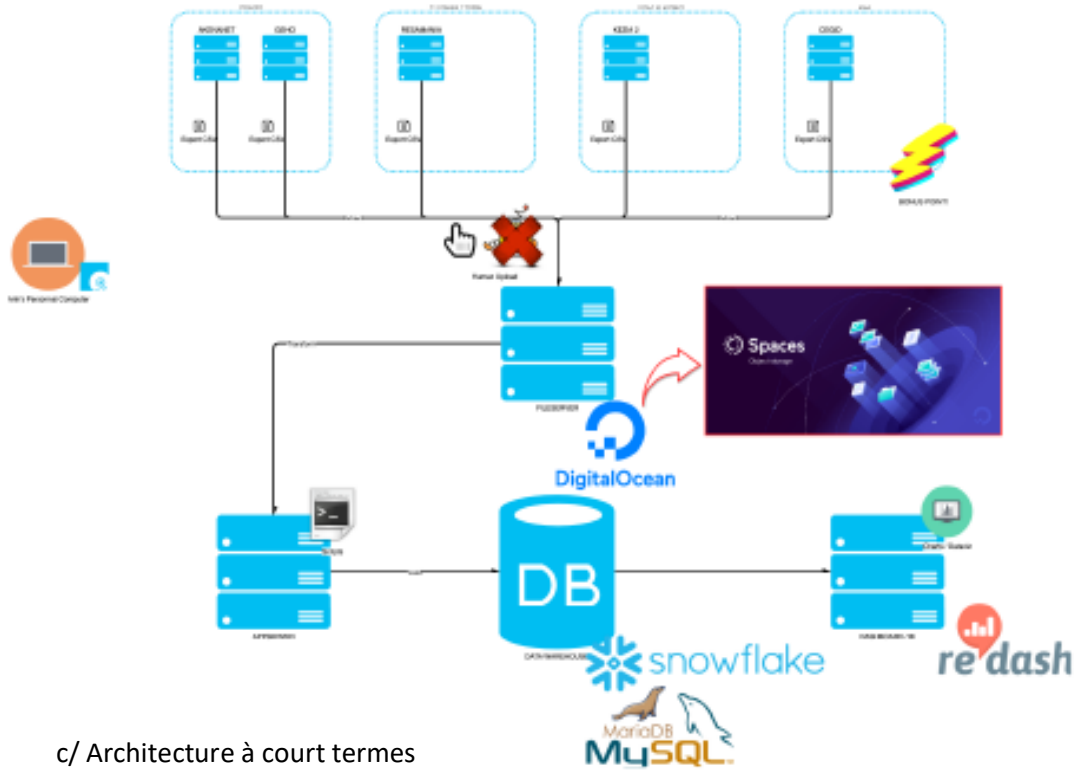
Annexe n°4 – Architectures proposées



a/ Architecture idéale

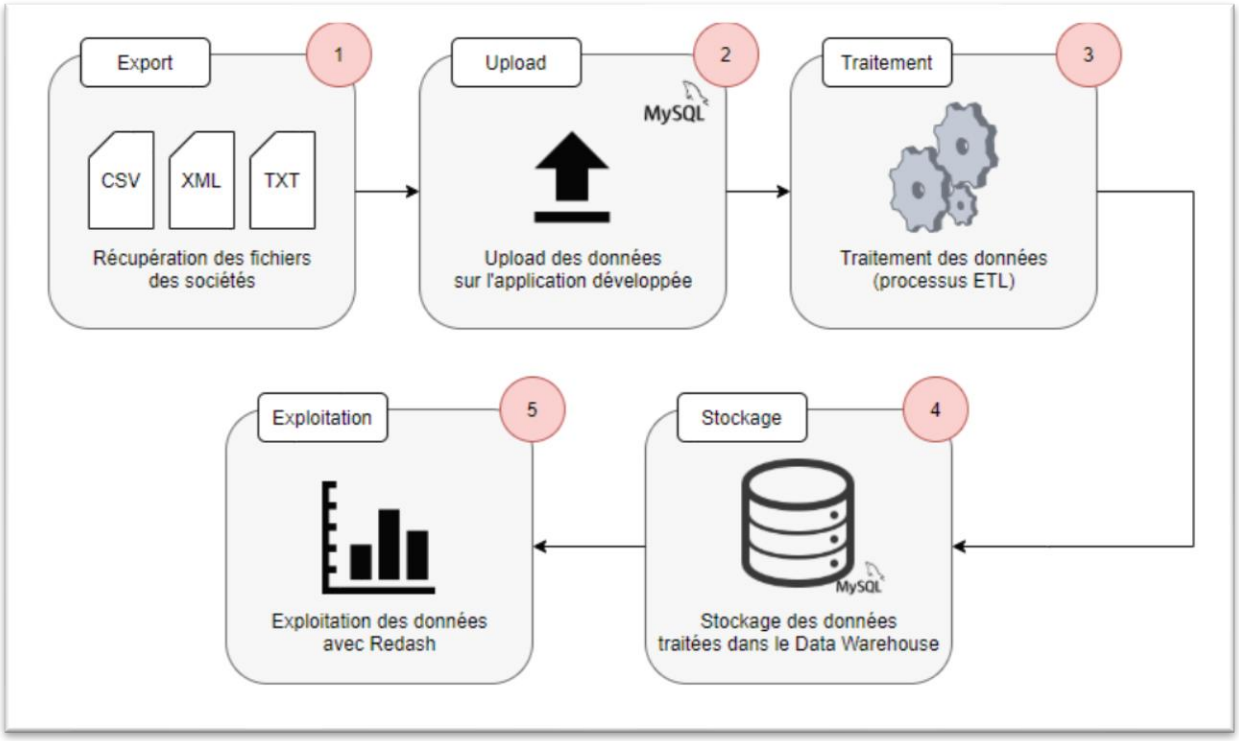


b/ Architecture Webservices Compliant



c/ Architecture à court termes

Annexes n°5 – Schéma récapitulatif des étapes du projet



Annexes n°6 – Fichiers de configuration Docker

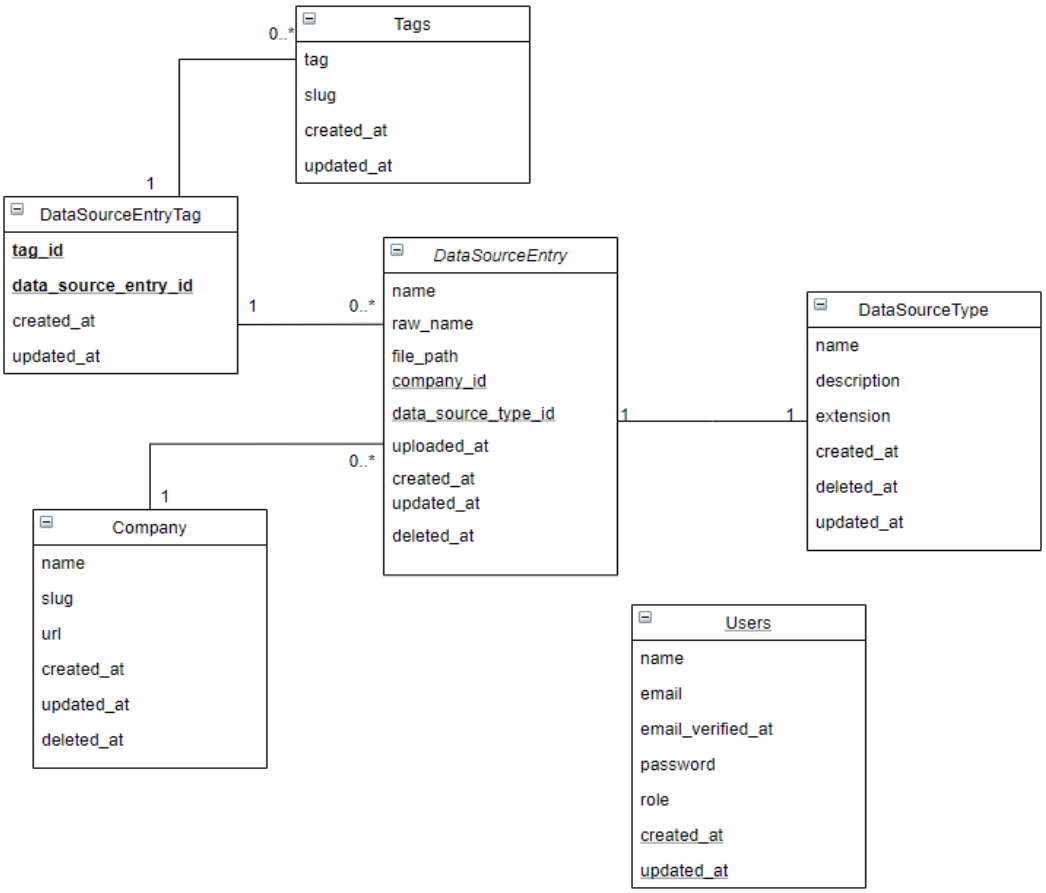
Fichier *docker-compose.yaml*

```yaml
version: '3.5'

services:

 # PHP Service
 app:
  build:
   context: .
   dockerfile: Dockerfile
  container_name: app
  image: ${APP_IMG}
  restart: unless-stopped
  tty: true
  environment:
   SERVICE_NAME: app
   SERVICE_TAGS: ${SERVICE_TAGS}
  working_dir: /var/www/html
  volumes:
   - ./laravel/:/var/www/html
   - ./docker/php/local.ini:/usr/local/etc/php/conf.d/local.ini

 # Nginx Service
 webserver:
  container_name: webserver
  image: ${WEBSERVER_IMG}
  restart: unless-stopped
  tty: true
  ports:
   - ${EXPOSED_PORT}:80
  expose:
   - ${EXPOSED_PORT}
  environment:
   VIRTUAL_HOST: ${DOMAINS}
   LETSENCRYPT_HOST: ${DOMAINS}
   LETSENCRYPT_EMAIL: ${LETSENCRYPT_EMAIL}
  volumes:
   - ./laravel/:/var/www/html
   - ./docker/nginx/conf.d/:/etc/nginx/conf.d/

 # MySQL Service
 db:
  container_name: db
  image: ${MYSQL_IMG}
  restart: unless-stopped
  tty: true
  ports:
   - 3306:3306
  environment:
   MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
   MYSQL_DATABASE: ${MYSQL_DATABASE}
   MYSQL_USER: ${MYSQL_USER}
   MYSQL_PASSWORD: ${MYSQL_PASSWORD}
   SERVICE_NAME: mysql
   SERVICE_TAGS: ${SERVICE_TAGS}
  volumes:
   - ${DB_PATH}:/var/lib/mysql
   - ./docker/mysql/my.cnf:/etc/mysql/my.cnf

# Docker Networks
networks:
 default:
  external:
   name: ${NETWORK}
```

Fichier *dockerfile.txt*

```
FROM php:7.3.13-fpm

# Copy composer.lock and composer.json
COPY ./laravel/composer.lock ./laravel/composer.json /var/www/html/
# Set working directory
WORKDIR /var/www/html
# Install dependencies
RUN apt-get update && apt-get install -y \
   git \
   zip \
   curl \
   sudo \
   unzip \
   libicu-dev \
   libbz2-dev \
   libpng-dev \
   libzip-dev \
   libjpeg-dev \
   libmcrypt-dev \
   libreadline-dev \
   libfreetype6-dev \
   g++
# Clear cache
RUN apt-get clean && rm -rf /var/lib/apt/lists/*
# Install extensions
RUN docker-php-ext-install \
   bz2 \
   intl \
   iconv \
   bcmath \
   opcache \
   calendar \
   mbstring \
   pdo_mysql \
   zip

RUN docker-php-ext-install pdo_mysql mbstring zip exif pcntl
RUN docker-php-ext-configure gd --with-gd --with-freetype-dir=/usr/in-
clude/ --with-jpeg-dir=/usr/include/ --with-png-dir=/usr/include/
RUN docker-php-ext-install gd

# Install composer
# RUN curl -sS https://getcomposer.org/installer | php -- --install-
dir=/usr/local/bin --filename=composer
COPY --from=composer:1.9.1 /usr/bin/composer /usr/bin/composer

# Install nodejs
RUN curl -sL https://deb.nodesource.com/setup_13.x | bash -
RUN apt-get install -y nodejs

# Add user for laravel application
RUN groupadd -g 1000 www
RUN useradd -u 1000 -ms /bin/bash -g www www

# Copy existing application directory contents
COPY ./laravel /var/www/html

# Copy existing application directory permissions
COPY --chown=www:www ./laravel /var/www/html

# Change current user to www
USER www

# Expose port 9000 and start php-fpm server
EXPOSE 9000
CMD ["php-fpm"]
```

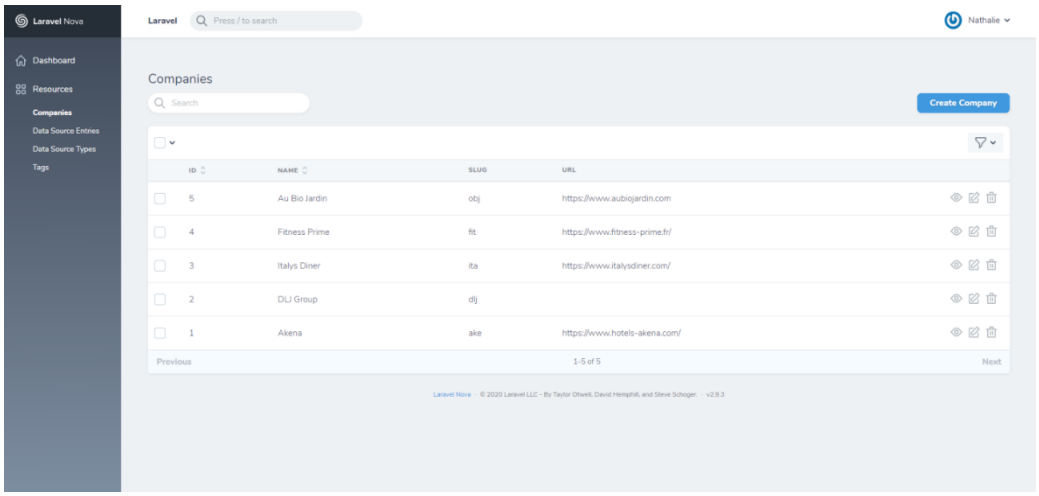
Annexes n°7 – Schéma de base de données



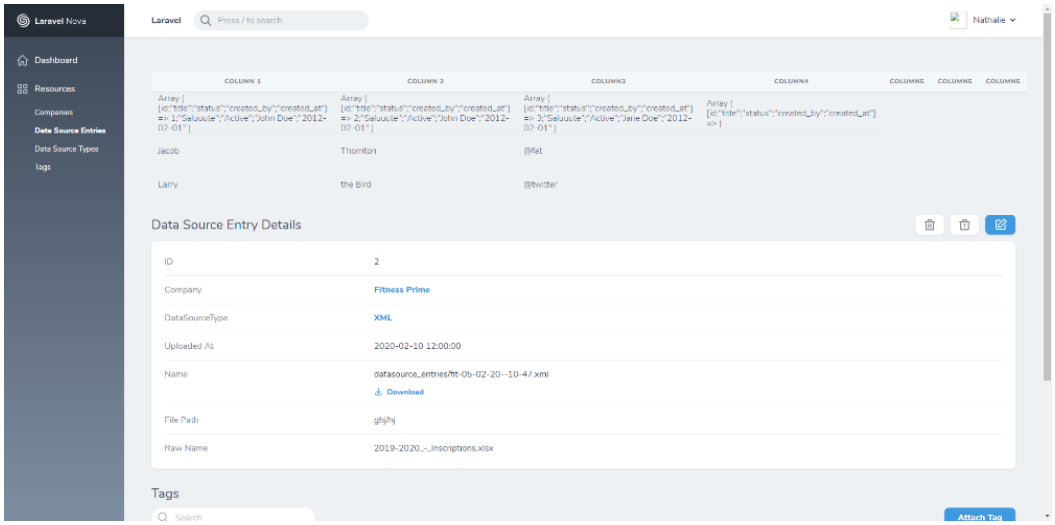
Annexes n°8 – Fonction fields() de la ressource Nova *DataSourceEntry*

```php
/**
     * Get the fields displayed by the resource.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array
     */
    public function fields(Request $request)
    {

        return [
            ID::make()->sortable(),
            BelongsTo::make('Company')->rules('required')->display('name'),
            BelongsTo::make('DataSourceType')->rules('required')->display('name'),
            BelongsToMany::make('Tags'),
            DateTime::make('Uploaded At'),
            File::make('Name')
                ->disk('public')
                ->path('datasource_entries')
                ->storeAs(function (Request $request) {
                    return $this->company->slug.'-'.date('d-m-y--h-i').$this->dataSourceType->extension;
                })
                ->storeOriginalName('raw_name'),
            Text::make('File Path'),
            Text::make('Raw Name')->onlyOnDetail(),
        ];
    }
```

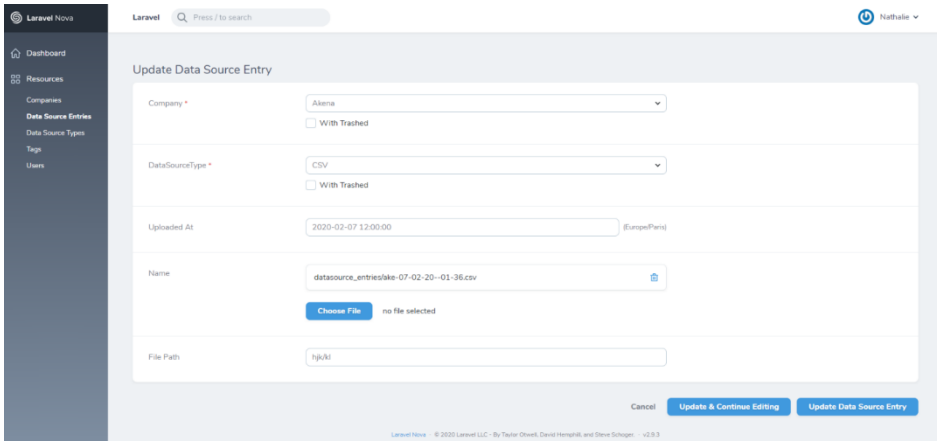
Annexes n°9 – Interfaces de l'application développée



Page d'accueil Ressource Company : liste des company + fonctionnalités de base (voir les détails, ajouter, modifier, supprimer)



Page de détails d'une DataSourcEntry (d'un upload) : détails, affichage d'un contenu de fichier CSV, bouton d'ajout d'un ou de plusieurs tag(s) + boutons modifier et supprimer)



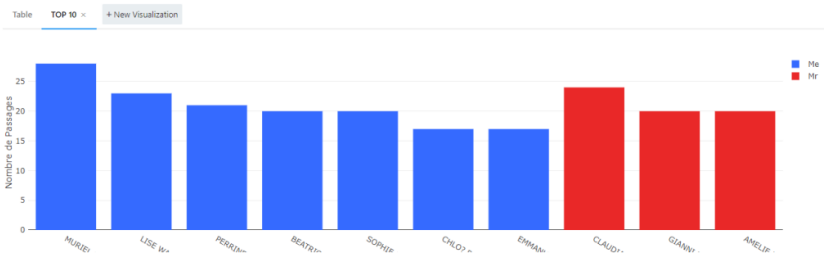
Page de modification type d'une DataSourcEntry (d'un upload)

Annexes n°10 – Script PHP permettant l'affichage d'un fichier CSV

```php
<?php

namespace App\Nova\Cards;

use Marquine\Etl\Etl;
use Illuminate\Database\Seeder;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\Log;
use League\Csv\Reader;
use League\Csv\Statement;

class ReaderCSV extends \Mako\CustomTableCard\CustomTableCard
{
    public function __construct()
    {
        $csv = Reader::createFromPath(storage_path('../storage/app/public/datasource_entries/test.csv'));
        $csv->setHeaderOffset(0);

        // Boucler sur les données du fichier csv
        $records = (new Statement())->process($csv);
        foreach ($records->getRecords() as $record) {
            return $record;
        }
    }
}
```

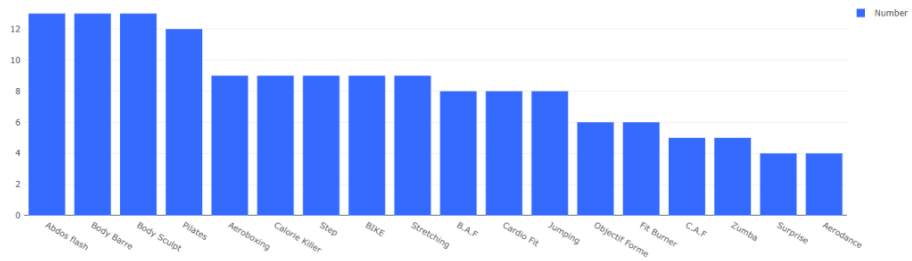
Annexes n°11 – Graphiques de Redash

```sql
SELECT S.Number,
       S.id_client,
       concat( clients.prenom, ' ', clients.nom ) AS IDENTITY,
       clients.civilite
FROM
  (SELECT Count(*) AS Number,
          clients_by_activities.client_id AS id_client
   FROM clients_by_activities
   WHERE clients_by_activities.client_id != ''
   GROUP BY client_id
   ORDER BY Number DESC
   LIMIT 10) S
JOIN clients ON clients.numero_client = S.id_client
ORDER BY S.Number DESC
```



Top 10 des clients (hommes et femmes confondues)

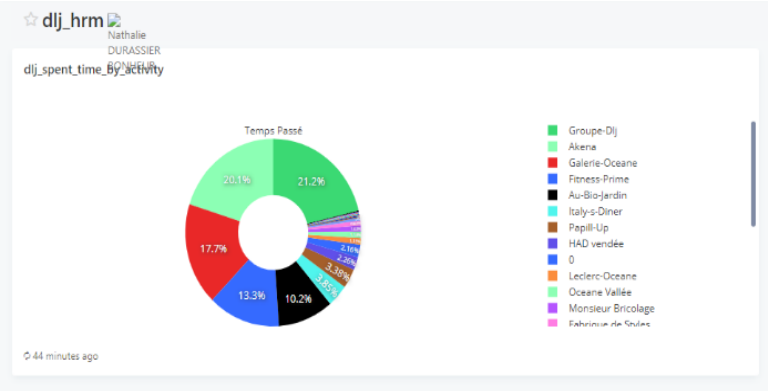
```sql
SELECT activity,
       count(booking_id) AS Number
FROM group_courses
GROUP BY activity
ORDER BY number DESC
```



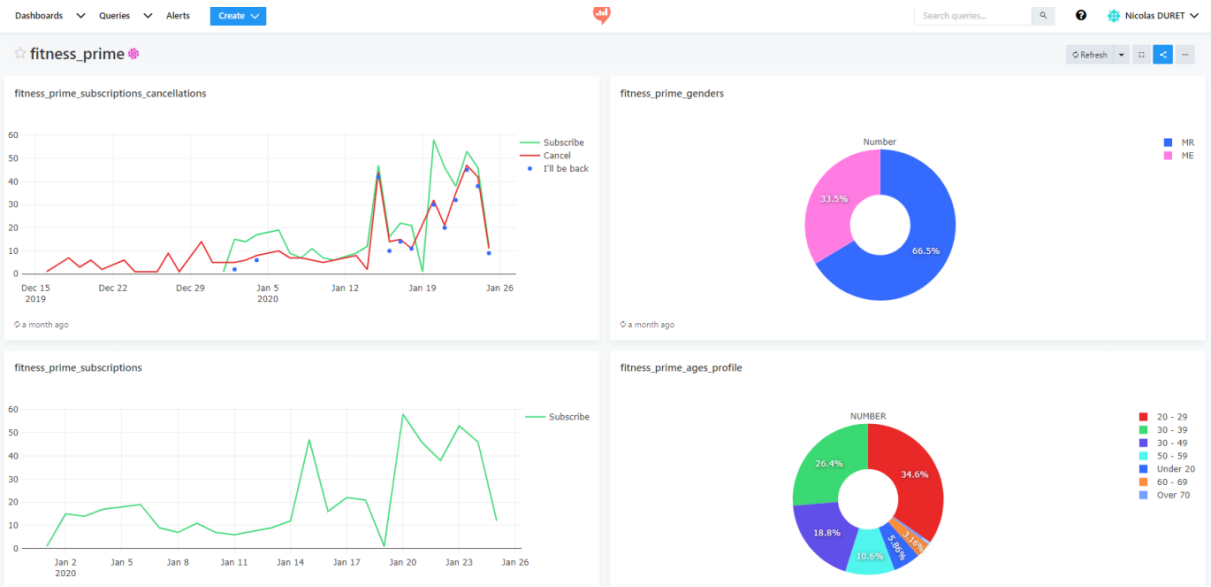
Nombre de participants à chaque cours collectif

```sql
Create view clients_by_activites AS SELECT
  booking_id,
  TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(registered, '/', n.digit+1), '/', -1), '#', -1), '-', 1))
FROM
  group_courses
  INNER JOIN max_clients_by_activity n
  ON LENGTH(REPLACE(registered, '/' , '')) <= LENGTH(registered)-n.digit
ORDER BY
  booking_id,
  n.digit;

SELECT S.Number, S.id_client, clients.nom, clients.prenom, clients.civilite FROM
(SELECT Count(*) As Number, clients_by_activities.client_id As id_client
FROM clients_by_activities
WHERE clients_by_activities.client_id != ''
GROUP BY client_id
ORDER BY Number DESC
LIMIT 10) S
JOIN clients ON clients.numero_client = S.id_client
ORDER BY S.Number DESC
```

Requête SQL renvoyant le nom, prénom, identifiant et nombre de cours collectifs effectués par client

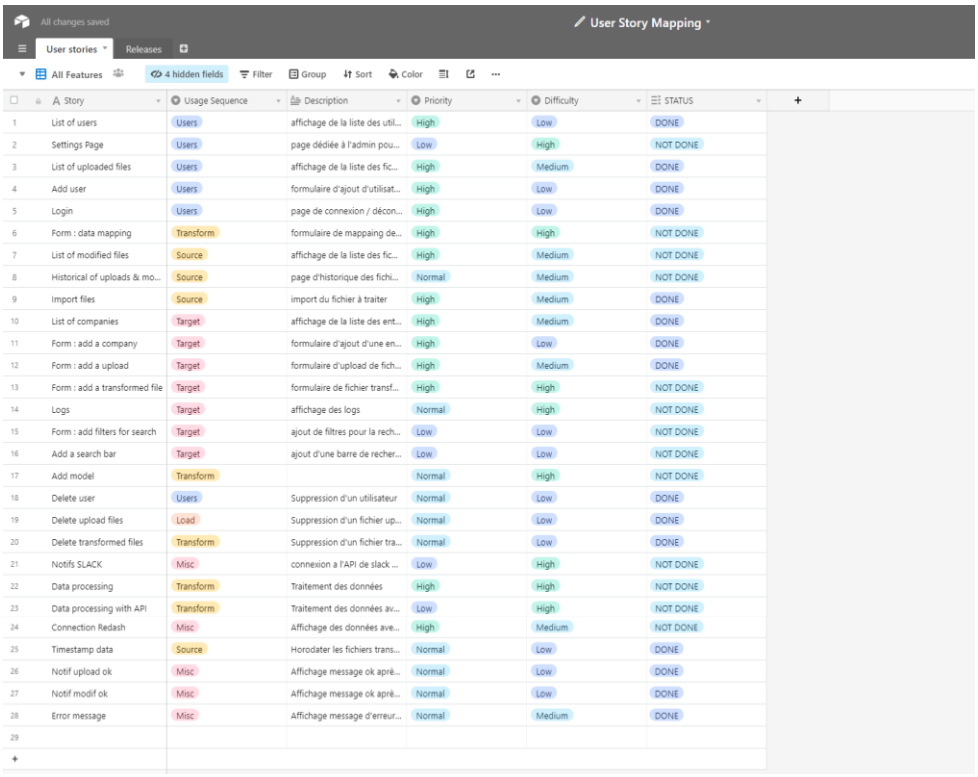


Pourcentage de temps passé pour chaque activité du groupe DLJ



Dashboard existant de Fitness Prime

Annexes n°12 – User Stories définies sur l'outil Airtable



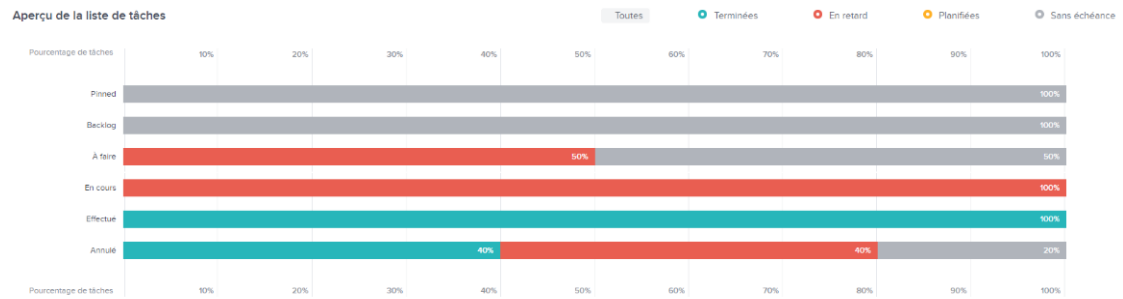
All changes saved — User Story Mapping

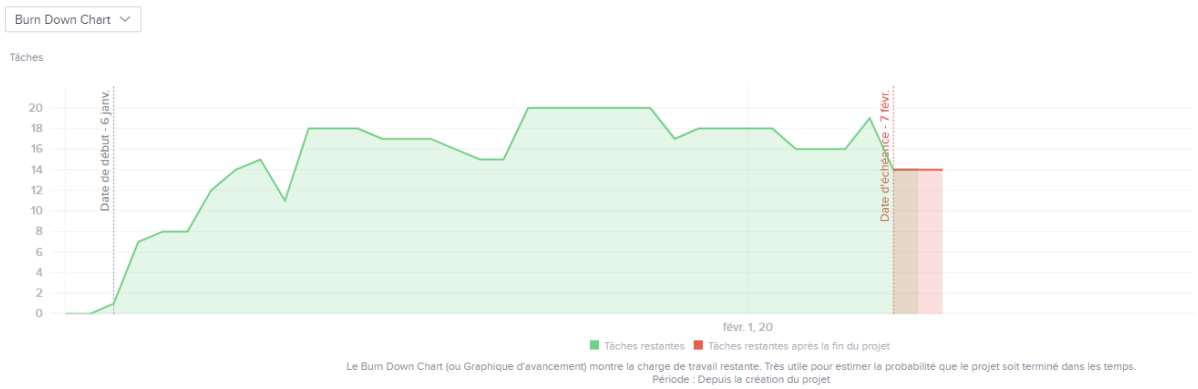
User stories | Releases

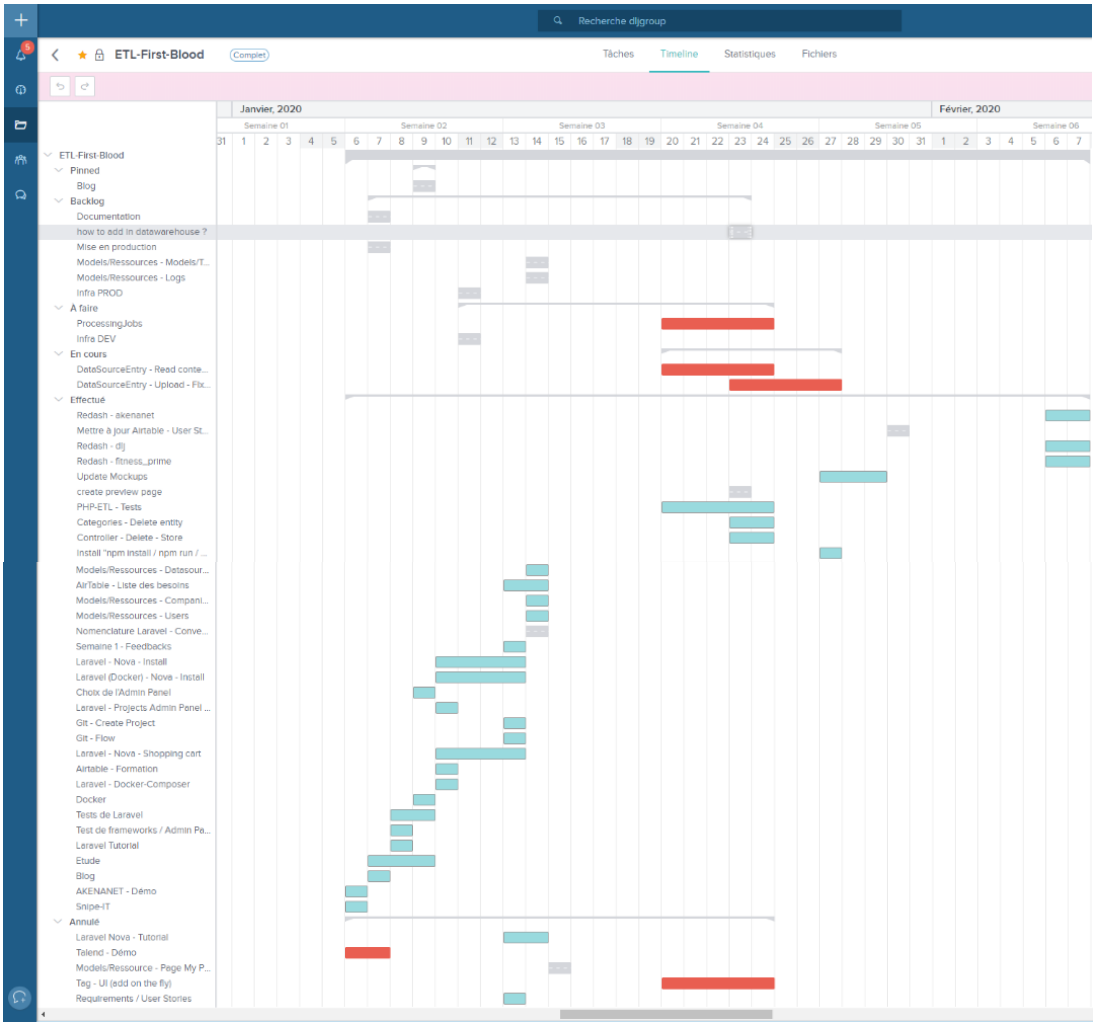
All Features | 4 hidden fields | Filter | Group | Sort | Color

	Story	Usage Sequence	Description	Priority	Difficulty	STATUS
1	List of users	Users	affichage de la liste des util...	High	Low	DONE
2	Settings Page	Users	page dédiée à l'admin pou...	Low	High	NOT DONE
3	List of uploaded files	Users	affichage de la liste des fic...	High	Medium	DONE
4	Add user	Users	formulaire d'ajout d'utilisat...	High	Low	DONE
5	Login	Users	page de connexion / décon...	High	Low	DONE
6	Form : data mapping	Transform	formulaire de mappaing de...	High	High	NOT DONE
7	List of modified files	Source	affichage de la liste des fic...	High	Medium	NOT DONE
8	Historical of uploads & mo...	Source	page d'historique des fichi...	Normal	Medium	NOT DONE
9	Import files	Source	import du fichier à traiter	High	Medium	DONE
10	List of companies	Target	affichage de la liste des ent...	High	Medium	DONE
11	Form : add a company	Target	formulaire d'ajout d'une en...	High	Low	DONE
12	Form : add a upload	Target	formulaire d'upload de fich...	High	Medium	DONE
13	Form : add a transformed file	Target	formulaire de fichier transf...	High	High	NOT DONE
14	Logs	Target	affichage des logs	Normal	High	NOT DONE
15	Form : add filters for search	Target	ajout de filtres pour la rech...	Low	Low	NOT DONE
16	Add a search bar	Target	ajout d'une barre de recher...	Low	Low	NOT DONE
17	Add model	Transform		Normal	High	NOT DONE
18	Delete user	Users	Suppression d'un utilisateur	Normal	Low	DONE
19	Delete upload files	Load	Suppression d'un fichier up...	Normal	Low	DONE
20	Delete transformed files	Transform	Suppression d'un fichier tra...	Normal	Low	DONE
21	Notifs SLACK	Misc	connexion a l'API de slack ...	Low	High	NOT DONE
22	Data processing	Transform	Traitement des données	High	High	NOT DONE
23	Data processing with API	Transform	Traitement des données av...	Low	High	NOT DONE
24	Connection Redash	Misc	Affichage des données ave...	High	Medium	NOT DONE
25	Timestamp data	Source	Horodater les fichiers trans...	Normal	Low	DONE
26	Notif upload ok	Misc	Affichage message ok aprè...	Normal	Low	DONE
27	Notif modif ok	Misc	Affichage message ok aprè...	Normal	Low	DONE
28	Error message	Misc	Affichage message d'erreur...	Normal	Medium	DONE

Annexes n°13 – Taskworld : outil







Timeline de Taskworld : liste des tâches