Hes·so VALAIS WALLIS

Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2018

Novelis - Données "Crash-test"

Étudiant : Aleksandar Lazic

Professeur : Nicole Glassey Balet

Déposé, le : 30 juillet 2018

www.hevs.ch

RÉSUMÉ

Ce travail a été réalisé en collaboration avec le laboratoire de recherche et développement de Novelis. Il a pour but de créer une plateforme de gestion des données centralisées pour les « crashs-tests ». En effet, les lots d'aluminium fabriqués à l'usine de Sierre sont soumis à des tests visant à caractériser un alliage et à améliorer la résistance et la déformation de ces derniers. Les données résultantes de ces tests sont ensuite analysées par les employés de l'entreprise qui pourront modifier certaines propriétés dans le processus de fabrication de l'aluminium afin d'optimiser les propriétés du matériel et de sa production.

Nous avons tout d'abord analysé les besoins de l'entreprise par rapport à leur gestion actuelle des données. Ensuite, nous avons comparé des outils tout-en-un, des technologies de base de données, des langages de programmation ainsi que des librairies graphiques pour trouver une solution facilement intégrable dans l'entreprise.

Une solution web a été développée sur la base des résultats précédents. L'utilisateur a la possibilité d'insérer toutes les données allant du lot d'aluminium et ses propriétés jusqu'aux résultats des « crashs-tests ». L'insertion est automatisée pour certaines parties du processus grâce à l'extraction directe des données de fichiers machines générés par les machines effectuant le test. Une fois celles-ci stockées dans la base de données, l'utilisateur peut filtrer les différents résultats selon ses besoins puis les exporter dans un fichier CSV[1].

Notre application a finalement été déployée au sein de l'intranet de Novelis afin que celle-ci ne soit accessible que par les employés ayant accès au réseau sécurisé interne de l'entreprise.

Mots-clés : crash-test, base de données, nodejs, intégration

[1] CSV : Comma-separated values

AVANT-PROPOS

Dans le cadre du travail de Bachelor 2018, proposé par Mme Nicole Glassey Balet de l'Institut de recherche en Informatique de Gestion et Mme Fumeaux de l'entreprise Novelis, il a été demandé à l'étudiant de développer une base de données centralisée pour la gestion des données « crash » au sein du laboratoire de recherche et développement de Novelis.

Le but de ce travail est de faire une étude des besoins en analysant le contexte actuel de l'entreprise afin de trouver la façon optimale de stocker les données ainsi que de visualiser ces dernières. C'est la raison pour laquelle, nous établissons un état de l'art des outils existants, des technologies de base de données, des langages de programmation et des librairies graphiques. Nous allons ensuite développer la solution la plus adaptée pour l'entreprise en faisant particulièrement attention à plusieurs critères comme l'intégration ou la sécurité des données.

L'idée de travailler en collaboration avec une entreprise d'une telle renommée était une grande source de motivation. De plus, le fait que ce projet soit très concret nous a également confortés dans notre choix.

La plus grande difficulté de ce projet était surtout de comprendre les processus internes à l'entreprise. En effet, n'ayant aucune connaissance préalable en métallurgie, il a été très compliqué de démarrer ce travail avant de comprendre le processus de fabrication de l'aluminium.

REMERCIEMENTS

Nous tenons à remercier Mme Nicole Glassey Balet de nous avoir encadrés et aidés tout au long de ces trois mois.

Nous remercions également M. Fabian Cretton pour ses conseils techniques lors de la modélisation de la base de données.

Un grand merci à Mme Maude Fumeaux qui a su nous donner toutes les informations nécessaires pour mener à bien ce projet. Nous la remercions également pour sa grande patience et son temps consacré.

Nous remercions M. Roger Schaer pour ses explications sur le déploiement d'une application NodeJS.

Enfin, nous aimerions remercier toutes les personnes ayant pris le temps de relire et corriger ce document pour en améliorer la qualité globale.

STRUCTURE DU CD

Au dos de ce rapport est joint un CD-ROM contenant les dossiers et fichiers suivants :

- 00_LAZIC_RAPPORT.pdf
  - Ce fichier correspond au rapport écrit du travail de Bachelor.
- 01_LAZIC_DOC_TECHNIQUE.pdf
  - Ce fichier correspond à la documentation technique de l'application.
- 02_LAZIC_LOGBOOK.xlsx
  - Ce fichier correspond aux heures réalisées durant ce travail.
- 03_LAZIC_CODE_SOURCE
  - Ce dossier contient le code source de notre application
- 04_LAZIC_POSTER.pptx

TABLES DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES ILLUSTRATIONS

LISTE DES ABRÉVIATIONS

Ajax	Asynchronous JavaScript And XML
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CSV	Comma-separated values
GIS	Geographic Information System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MVT	Modèle – Vue - Template
NIST	National Institute of Standards and Technology
ORM	Object-relational mapping
PDF	Portable Document Format
PHP	Hypertext Preprocessor
R&D	Recherche et Développement
SQL	Structured Query Language
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
XML	Extensible Markup Language

INTRODUCTION

Les bases de données occupent une place prépondérante dans le système informatique d'une entreprise. En effet, celles-ci permettent de centraliser les données à un seul et même endroit afin que toutes personnes ayant les autorisations nécessaires puissent y accéder ou les modifier à tout moment. Ces dernières ont en partie remplacé des moyens de stockage comme les dossiers et les fichiers dossiers d'un ordinateur (Cheiney, Picouet, Saglio, & Abdessalem, 2011).

L'utilisation d'un système de fichiers pour le stockage peut entrainer plusieurs problèmes pour l'entreprise comme la redondance des données ou l'uniformité de ces dernières. En effet, plusieurs utilisateurs n'auront pas la même façon d'enregistrer les données. Ce manque de cohérence peut surtout poser des problèmes au moment d'une recherche.

De plus, la modification simultanée d'un fichier par plusieurs utilisateurs est également impossible à moins d'utiliser un outil de collaboration comme Dropbox. Ces diverses raisons nous ont poussés à implémenter une base de données pour la gestion des informations de l'entreprise.

Actuellement, Novelis qui est le premier producteur mondial en produits laminés en aluminium, gère ses données à l'aide de plusieurs fichiers Excel où sont stockées diverses informations sur les lots d'aluminium fabriqués dans les usines de production de Sierre. Ceux-ci seront ensuite soumis à des « crashs-tests » visant à déterminer la résistance et la déformation de la matière. Les données résultantes de ces tests sont ensuite analysées par les employés de l'entreprise qui pourront modifier certaines propriétés dans le processus de fabrication de l'aluminium afin de rendre ce dernier plus performant. Toutes ces données se répètent dans les différents fichiers utilisés et il est particulièrement complexe de retrouver une information avec le système actuel de stockage.

L'objectif de ce travail a surtout été de développer une solution web permettant l'insertion et la visualisation des données de Novelis. Les utilisateurs ont ensuite la possibilité de filtrer les résultats puis de les exporter afin de comparer les différents « crashs ».

La première phase de notre travail consistait à comprendre la partie « métier » des employés pour ensuite pouvoir analyser les besoins afin de trouver la solution adéquate. Plusieurs séances ont donc été organisées avec les personnes en charge du projet à Novelis pour comprendre la nature des fichiers ainsi que leurs utilisations.

La seconde phase correspondait à l'analyse des outils existants sur le marché. En effet, nous avons comparé des outils « tout-en-un » (insertion et visualisation), des technologies de base de données, des langages de programmation ainsi que des librairies graphiques dans le but de déterminer la meilleure solution. Cette dernière devait être facilement intégrable au sein du système informatique actuel de l'entreprise.

La troisième partie de ce travail représente la modélisation d'une base de données ainsi que le développement d'une application web en NodeJS. Notre base de données permet aux utilisateurs d'insérer les données pour tout le processus de fabrication de l'aluminium en usine jusqu'aux « crashs-tests » effectués au laboratoire de recherche et développement. Afin d'automatiser certaines parties du processus, l'insertion des résultats des crashs se fait à l'aide d'importation de fichiers. Une fois ceux-ci insérés, l'utilisateur peut filtrer les résultats selon certains critères prédéfinis dans le but d'exporter le tout pour comparer les résultats. Nous avons ensuite déployé notre application dans le système informatique de Novelis. Notre solution n'est accessible qu'à l'interne afin de garantir la sécurité des données.

Tout au long du développement, les employés de Novelis ont pu tester notre solution localement sur notre ordinateur personnel. En effet, les rendez-vous hebdomadaires nous permettaient de présenter le travail accompli la semaine précédente et surtout de fixer les priorités pour la semaine à venir. Les différentes remarques ont également été prises en comptes afin d'améliorer l'expérience utilisateur.

Finalement, nous pouvons constater que cette application est une première base sur laquelle les potentiels futurs développeurs de l'entreprise pourront rajouter des couches supplémentaires afin d'implémenter de nouvelles fonctionnalités. Dans l'idéal, cette application devrait permettre d'afficher des graphiques pour les différents « crashs-tests » avec la possibilité d'exporter des PDF[2] comportant tous les résultats et graphiques souhaités.

[2] PDF : Portable Document Format

1. Analyse des besoins

1.1. Présentation de l'entreprise

Novelis est le premier producteur mondial de produits laminés en aluminium ainsi que l'un des plus importants recycleurs d'aluminium au monde. Ils collaborent avec de grandes entreprises dans différents domaines d'activités comme Audi, BMW, LG ou Coca-Cola. (Novelis, 2018). Novelis est une entreprise présente dans le monde entier et compte près de 11'500 employés répartis dans 11 pays et sur quatre continents (Novelis, 2017).

Le site de Sierre est composé de deux usines de production ainsi que d'un centre européen de recherche & développement (R&D). Ces usines sont spécialisées dans le développement et la production de tôle de carrosserie en alliage d'aluminium pour le domaine de l'automobile (VSlink, s.d.).

Figure 1 : Usine Novelis à Sierre

Source : http://oukas.info/?u=Geographic+Locations++Novelis

Les usines de Sierre couvrent l'ensemble du processus, de la coulée de lingots jusqu'à la finition. Elles disposent de nombreuses machines telles que les laminoirs à chaud et à froid.

1.2. Présentation de la problématique

Le laboratoire de R&D de Novelis Sierre effectue des crashs-tests sur l'aluminium fabriqué par l'entreprise. Leur but est d'améliorer la résistance et la déformation sur la base des résultats des différents tests. Trois types de tests sont effectués dans leur laboratoire.

- 3 points tests : simulation d'un crash latéral d'une voiture
    - Directions de laminage : longitudinale et de travers.

- Crush Test : simulation d'un crash frontal d'une voiture
    - Directions de laminage : longitudinale et de travers.
- Essai Tony : correspond à des essais mécaniques faits au laboratoire de production
    - Directions de laminage : longitudinale, de travers et diagonale

Avant d'analyser les résultats de ces tests. Il est important de connaître la composition de l'aluminium, l'alliage, l'épaisseur, etc. Actuellement, toutes ces données sont stockées dans différents fichiers Excel. Ci-dessous est présenté un aperçu de tous les fichiers utilisés dans l'entreprise.

1.2.1. Formulaire crash.xlsx

Il regroupe un historique de tous les lots d'aluminium qui ont été testés par le laboratoire. Il y a de nombreuses propriétés qui sont liées à un lot comme l'alliage interne, l'alliage international, l'épaisseur ou encore les traitements thermiques. Ce fichier représente surtout une vue globale de tous les lots sans avoir les résultats exacts des tests. Cependant, il contient un numéro de fichier Tony (cf. point 1.2.2) où nous pouvons retrouver les résultats de certains tests.

1.2.2. Fichier Tony

Ce fichier Tony est spécifique à un lot et reprend donc les informations concernant le processus de fabrication du lot d'aluminium déjà contenu dans le formulaire crash.xlsx ainsi que les valeurs des propriétés mécaniques liés aux différents tests.

Les résultats de ces tests sont « copier-coller » dans le fichier Tony depuis un CSV généré par la machine ayant effectué le test. Différentes propriétés mécaniques y sont listées comme le rp02, le rm ou encore l'ag.

1.2.3. Data machine.csv

Ce fichier CSV regroupe les valeurs des propriétés mécaniques pour chaque test effectué.

1.2.4. Test result.xlsx

Ce fichier comprend d'autres résultats pour les tests comme le nombre de plis, le nombre de plis devant, la présence de vis, etc. Les données de chaque tube testé seront également stockées dans un fichier comme celui-ci.

1.2.5. Graphiques horizontaux.xlsx / Graphiques verticaux.xlsx

Ces fichiers listent les résultats des crush-tests ainsi que des tests de flexion en 3 points pour chaque tube testé. Les données sont « copier-coller » depuis un fichier CSV reçu d'une machine. Ils permettront également de générer des graphiques afin d'analyser les résultats.

1.2.6. Images et vidéos

Finalement, des images et vidéos sont également liées à chaque crash-test afin d'avoir un aperçu visuel.

1.2.7. Synthèse

Pour résumer, il n'est pas facile de se retrouver avec tous ces fichiers. Pour trouver une information, il est souvent nécessaire d'ouvrir les fichiers un par un avant de trouver ce que l'on souhaite.

De plus, étant donné que l'enregistrement des données est fait manuellement, le risque d'erreur est plus élevé qu'avec un processus automatisé ou semi-automatisé.

Ainsi, une base de données centralisée permettrait de visualiser toutes ces données depuis une seule et même plateforme. La solution doit être facilement intégrable chez Novelis et surtout accessible uniquement à l'interne d'un point de vue de sécurité.

Après discussion avec le service informatique de Novelis, nous avons été informés qu'un serveur virtuel Linux nous sera mis à disposition pour le déploiement d'une solution Web dans l'intranet de l'entreprise. Cette dernière utilise la technologie MariaDB pour leurs projets en cours, qui est un système de gestion de base de données relationnelle « *open-source* ». Il nous a été demandé de respecter certaines conventions afin d'être le plus proche possible des pratiques internes. Voici une liste non exhaustive des règles à respecter pour l'intégration de la base de données :

- Pas de majuscule dans les champs de base de données (afin d'assurer la transition entre Windows et Linux)
- Utilisation d'un souligné pour séparer les mots : « **date_modification** » par exemple.
- Le nommage des tables se fait comme suit : « **tbl_**exemple »
- Dans une table, la clé primaire commence par le préfixe « id_ ».

Il est essentiel de respecter ces différentes conditions si le projet devait être repris dans le futur par un employé de l'entreprise.

Maintenant que nous avons compris la problématique de l'entreprise et les critères à respecter, nous pouvons passer à l'état de l'art qui va nous permettre d'analyser les outils existants sur le marché.

2. État de l'art

Avant de développer un outil, il est important de faire une analyse de l'existant. En effet, il ne faut pas concevoir une solution qui n'aurait en apparence aucune plus-value par rapport à celles déjà présentes sur le marché. Deux scénarios distincts se présentent donc à nous.

La première étape consiste à déterminer s'il existe un outil « all-in-one » permettant d'assurer la récolte des données de façon intuitive ainsi que la visualisation de ces dernières avec des graphiques. L'avantage de cette solution est qu'elle ne requiert aucun développement de notre part. Avant d'entamer cette analyse, il est important de définir certains critères :

- Récolte : l'outil propose une interface intuitive pour la récolte des données
- Visualisation : les graphiques disponibles sont pertinents
- Intégration : l'outil est facilement intégrable à Novelis

Notre second scénario comprend le développement de notre propre alternative. Dans l'éventualité où aucune solution « all-in-one » adéquate ne serait trouvée, nous développerions notre propre application. En effet, celle-ci devra comporter une technologie de base de données pour la récolte, un langage de programmation web ainsi qu'une librairie graphique pour la visualisation.

2.1. Scénario 1 : outils « tout-en-un » de récolte de données et de visualisation

Cette première étape nous permet de constater s'il existe un outil « tout-en-un », qui assurera le stockage des données de façon persistante ainsi que la visualisation de ces dernières grâce à des graphiques. Pour ce faire, nous allons analyser plusieurs solutions puis les comparer selon certains critères.

Premièrement, une base de données centralisée devra être intégrée afin que les données importées/stockées soient accessibles par tous les employés à tout moment. Celle-ci devra répondre à plusieurs contraintes liées à la complexité des données dans l'entreprise. Ensuite, l'outil devra nous permettre d'ajouter de nouvelles données au fur et à mesure afin de pouvoir garder une trace de tous les crashs-tests ayant été effectués. Enfin, pour ce qui est de la visualisation, l'utilisateur devra être en mesure de comparer plusieurs spécificités d'un crash et de les visualiser avec un graphique.

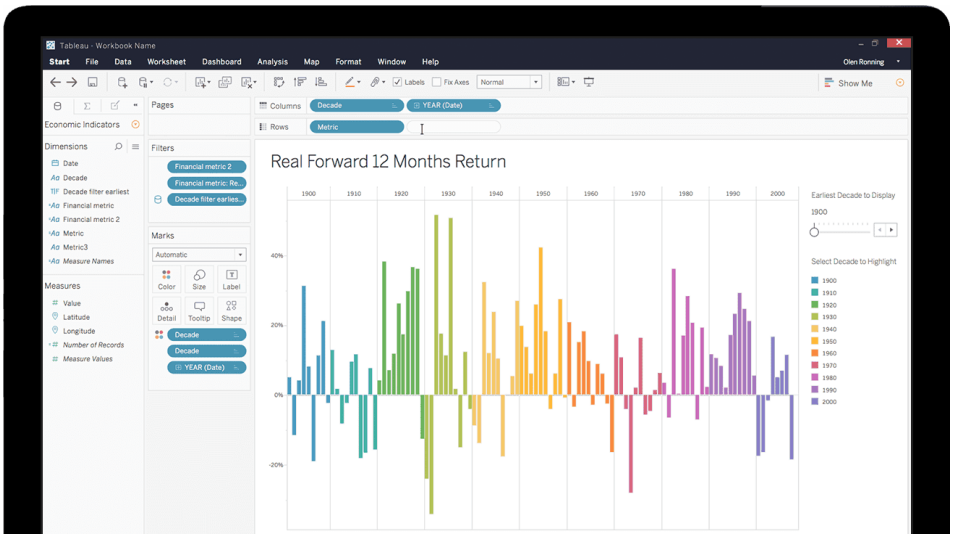
Après plusieurs recherches, nous avons ressorti trois outils répondant plus ou moins à nos besoins. Dans un premier temps, nous allons comparer ces outils puis décider si une solution pourrait être choisie en tant que solution finale à intégrer chez Novelis.

2.1.1. Tableau

Présentation

Tableau est un outil de gestion, mais également de visualisation de données. Il permet aux utilisateurs de voir et comprendre leurs données de façon dynamique. Sa force principale repose sur le glisser-déposer, qui permet de visualiser toutes les informations de l'entreprise en temps réel. Plus de 50'000 clients utilisent ce logiciel dans différents domaines tels que l'aérospatial, les assurances ou bien les banques. Il a été conçu de façon à pouvoir gérer de grands jeux de données pouvant changer régulièrement (Tableau Software, 2017).

Figure 2 : Visualisation de données sur Tableau



Source : https://www.tableau.com/fr-fr/about/mission#power-empower

Base de données

Tableau nous permet de nous connecter à une base de données de type SQL[3] ou des applications cloud telles que « *Google Analytics* » ou « *Salesforce* ». Cependant, les données devront être entrées directement dans la base de données. C'est-à-dire qu'il n'y a pas d'interfaces utilisateurs ergonomiques permettant d'insérer de nouvelles données. Les personnes, voulant ajouter de nouvelles entrées dans la base de données SQL, devront connaître ce langage informatique.

Visualisation

Les tableaux de bord de ce programme sont présentés comme des moyens d'explorer et de comprendre notre modèle de données (Tableau Software, 2017). Cet outil nous permettra d'obtenir

[3] Structured Query Language

rapidement des réponses à nos questions grâce aux différents graphiques. De plus, il est possible de se connecter à plusieurs types de données comme les feuilles de calculs ou les bases de données SQL (Tableau Software, 2017).

Intégration

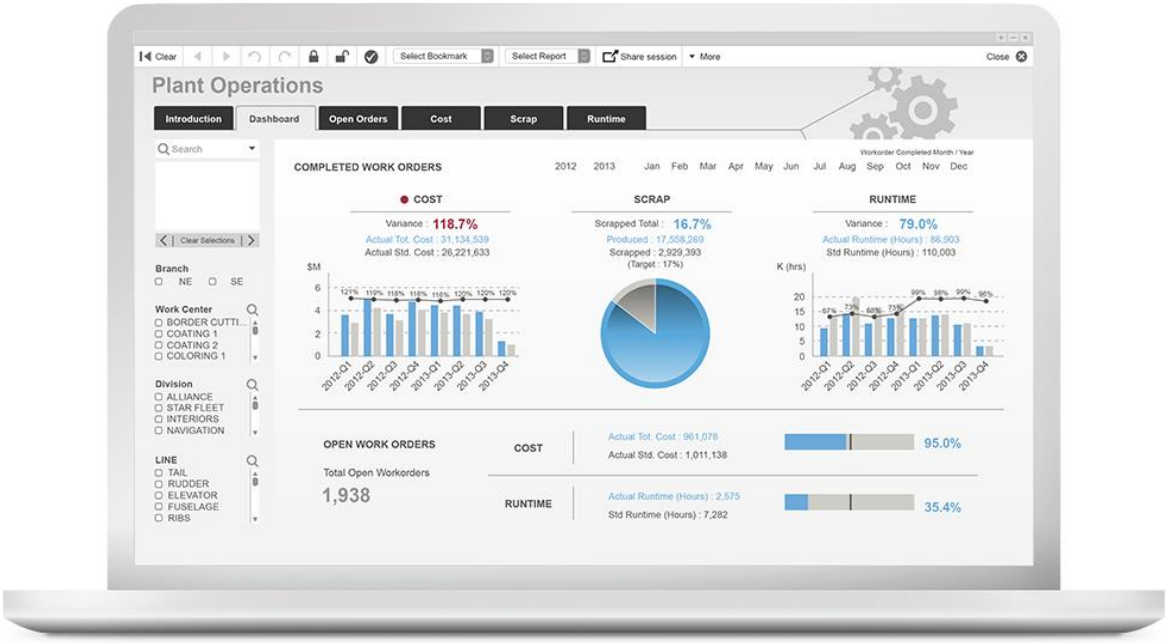
Tableau requiert beaucoup de configurations pour être intégré dans une entreprise de la taille de Novelis. En effet, le logiciel devra être installé et configuré sur tous les postes des employés ayant la nécessité d'utiliser ce programme.

2.1.2. QlikView

Présentation

L'entreprise Qlik avec leur outil QlikView est un des plus grands concurrents de Tableau. Leur solution est utilisée par plus de 40'000 clients à travers plus de 100 pays dans le monde. Il a pour avantage d'être facilement personnalisable. Ses nombreuses fonctions en font également un des « leaders » sur le marché. QlikView nous permet de créer et déployer rapidement des applications d'analyse ainsi que des tableaux de bord interactifs. Toute la structure ainsi que les différentes données nécessaires à la création des rapports sont chargées en mémoire, ce qui garantit une grande rapidité d'exécution (Qlik, 2018).

Figure 3 : Visualisation de données sur QlikView



Source : https://www.qlik.com/fr-fr/products

Base de données

QlikView nous permet de nous connecter à plusieurs types de bases de données à l'aide d'un connecteur différent selon les types de bases de données (Microsoft SQL Server, MySQL…). Les données peuvent être lues directement depuis les bases de données ou en important des fichiers de type Excel, CSV ou XML[4].

Visualisation

De nombreux graphiques sont disponibles dans QlikView, cependant l'utilisation de ces derniers dépend de nos besoins. En effet, un graphique à barres sera plus approprié pour des comparaisons côte à côte qu'un nuage de points.

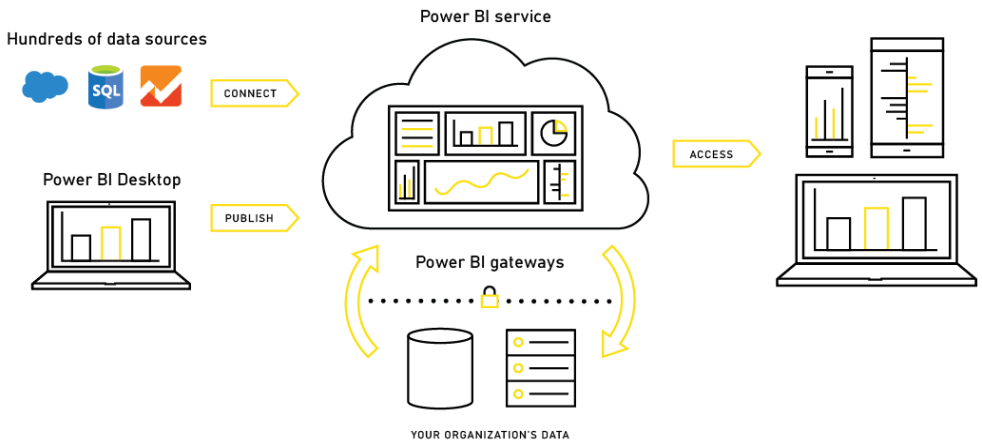
Intégration

L'intégration de QlikView est semblable à celle de Tableau.

2.1.3. Power BI

Power BI est une suite d'outils d'analyse développée par Microsoft en 2014. Les tableaux de bord Power BI garantissent une vue globale des données avec des indicateurs de mesure ainsi que des mises à jour en temps réels. Les rapports sont également accessibles où que nous soyons à l'aide de Power BI Mobile. Il garantit également une centralisation des données qu'elles se trouvent sur un cloud ou sur une base de données SQL (Microsoft - Power BI, 2018)

Figure 4 : Architecture de Power BI



Source : https://powerbi.microsoft.com/fr-fr/what-is-power-bi/

[4] XML : Extensible Markup Language

Base de données

Grâce à Power BI, nous pouvons intégrer des données depuis différentes sources (Excel, CSV, SQL Server…). Les données peuvent également être importées depuis une page web en insérant l'URL[5] de cette dernière.

Visualisation

De nombreux types de visualisation sont disponibles selon nos besoins. Les graphiques en aires ainsi que les histogrammes sont bien évidemment les plus connus, mais il est également possible de combiner différents types de graphiques tels qu'un histogramme et un graphique en courbes. Comme pour les autres logiciels vus précédemment, le choix du graphique se fera selon des objectifs définis (affichage une progression, recherche d'une valeur spécifique…) (Microsoft - Power BI, s.d.).

Intégration

À l'instar de ses concurrents QlikView et Tableau, PowerBI requiert les mêmes types de configuration pour une intégration dans l'entreprise.

2.1.4. Synthèse des outils de récolte de données et de visualisation

Comme nous avons pu le voir, tous les outils de récolte de données et de visualisation sont assez similaires d'un point de vue global. Ils proposent tous des fonctionnalités semblables que ce soit pour la connexion à la base de données ou pour la diversité des graphiques proposés.

Les outils de visualisation pourraient être intéressants pour notre projet contrairement à la récolte de données où l'on ne peut pas insérer des données de façon intuitive avec une interface utilisateur.

Nous passons maintenant à notre deuxième scénario où nous analysons les différentes technologies de base de données, les frameworks de programmation web ainsi que les librairies graphiques.

[5] URL : Uniform Resource Locator

2.2. Scénario 2 : technologies de base de données

Dans ce chapitre, nous listons les technologies de base de données relationnelles les plus connues.

2.2.1. MariaDB

MariaDB est un système de gestion de base de données relationnelle. À la suite du rachat de MySQL par Oracle, un des fondateurs de MySQL décida de fonder MariaDB qui est juste une reprise de ce dernier. C'est un projet « *open-source* » développé en C et C++. Les accès à la base de données se font avec le langage SQL. Les dernières versions de MariaDB incluent également les formats de données GIS[6] et JSON[7].

Figure 5 : MariaDB

Source : https://mariadb.org/

En général, les bases de données MariaDB et MySQL sont très compatibles, c'est-à-dire que nous pourrions désinstaller MySQL et utiliser MariaDB avec le même projet sans trop de problèmes (MariaDB Foundation, 2018).

De plus, il est possible de l'utiliser avec les langages de programmation les plus connus (C, C++, Java, PHP[8], NodeJS, Python…).

MariaDB propose également des correctifs liés à la sécurité plus fréquemment que MySQL. De plus, les différents « *benchmarks* » nous prouvent que MariaDB est plus performant que MySQL (https://mariadb.com/resources/blog/mariadb-53-optimizer-benchmark).

[6] GIS : Geographic Information System

[7] JSON : JavaScript Object Notation

[8] PHP : Hypertext Preprocessor

2.2.2. PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle et objets open-source. Le langage SQL est utilisé pour l'accès à la base de données (requêtes SQL). PostgreSQL a été créé en 1986 en faisant suite au projet POSTGRES de l'université de Californie. Cela fait maintenant plus de 30 ans que des développeurs travaillent activement sur cette plateforme (PostgreSQL, 2018).

Figure 6 : PostgreSQL

Source : https://www.postgresql.org/

PostgreSQL est compatible avec les principaux systèmes d'exploitation (Windows, macOS, Linux), et est depuis 2001 conforme aux propriétés ACID (atomicité, cohérence, isolation et durabilité) qui sont les conditions requises d'une transaction réussie dans une base de données.

En plus d'être gratuit et open-source, PostgreSQL est grandement configurable avec la possibilité de créer ses propres types de données ou de créer des fonctions personnalisées.

2.2.3. Microsoft SQL Server

Microsoft SQL Server est un système de gestion de base de données propriétaire conçu par Microsoft. La première version est sortie en 1989. Néanmoins, de nouvelles versions sont publiées environ tous les deux ans (2017, 2016, 2014, 2012…). Microsoft SQL Server est indépendant du langage de programmation ainsi que de la plateforme utilisée (Windows, Linux) (Microsoft, 2018).

Figure 7 : Microsoft SQL Server

Source : https://worldvectorlogo.com/logo/microsoft-sql-server

De plus, selon la base de données des vulnérabilités du NIST[9], Microsoft SQL Server est la base de données la moins vulnérable des sept dernières années (https://www.microsoft.com/fr-fr/sql-server/sql-server-2017#footnote).

2.2.4. Synthèse des technologies de base de données

Les trois types de bases de données choisies correspondent au besoin du projet. La grande différence est que Microsoft SQL Server est un logiciel payant contrairement à ses concurrents, MariaDB et PostgreSQL qui sont gratuits.

De plus, ils sont simples à intégrer avec la majorité des langages de programmation utilisés dans le monde du développement. Ils sont également indépendants du système d'exploitation utilisé, ce qui est un point essentiel pour une entreprise de l'envergure de Novelis.

2.3. Scénario 2 : frameworks de développement

Dans ce chapitre, nous listons les frameworks de développement web les plus connus. Un framework est un ensemble de librairies visant à faciliter la tâche des développeurs. Son but est de créer une architecture spécifique à tous les projets ainsi que des fonctionnalités de base qui seront réutilisables autant de fois que nécessaire.

2.3.1. Django

Django est un framework de développement web open-source et écrit en Python. Il existe également d'autres frameworks Python comme Pyramid ou Flask. Cependant, Django nous permet d'automatiser de nombreuses tâches et dispose d'une communauté active ainsi que d'une documentation très complète (Django Project, 2018).

Figure 8 : Django Project

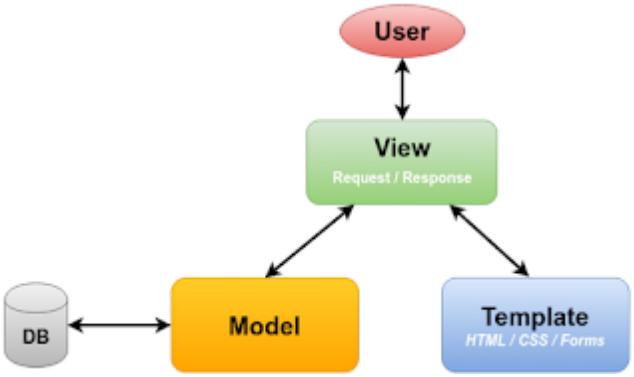
Source : https://www.djangoproject.com/community/logos/

En ce qui concerne le fonctionnement général de Django, celui-ci utilise une architecture MVT (Modèle – Vue - Template). Contrairement au très populaire MVC (Modèle – Vue – Contrôleur), l'architecture MVT utilise un template qui est un fichier spécial où l'on peut rajouter des conditions

[9] NIST : National Institute of Standards and Technology

ou des boucles. Le framework s'occupera ensuite de générer la vue HTML (Hypertext Markup Language) en fonction du template, puis affichera cette dernière à l'utilisateur final. Nous trouvons ci-dessous une représentation graphique de l'architecture MVT.

Figure 9 : Architecture MVT



Source : (Singh, 2016)

Django nous permet également de gagner du temps en automatisant certains processus tel que l'authentification d'un utilisateur. De plus, il est également capable de générer une interface d'administration de toutes les données en introspectant les différents modèles de notre projet. Django utilise un ORM (Object-relational mapping) pour la communication avec la base de données. Chaque modèle représentera une table de notre base de données. Cela facilite grandement les interactions avec la base de données pour les développeurs.

2.3.2. Ruby on Rails

Ruby on Rails est un framework web similaire à Django mise à part que le langage de programmation utilisé est le Ruby. Il est gratuit et open-source. De plus, il est possible de contribuer au code source du framework comme plus de 4'500 personnes à travers le monde. Ruby on Rails possède une certaine convention de programmation à respecter afin que tous les développeurs reprenant un projet puissent directement lire et comprendre le code (Ruby on Rails, s.d.).

Figure 10 : Ruby on Rails

Source : https://rubyonrails.org/

De nombreuses fonctionnalités sont disponibles dès l'installation de ce dernier, sans configuration requise (ORM, génération de modèles/contrôleurs/vues…).

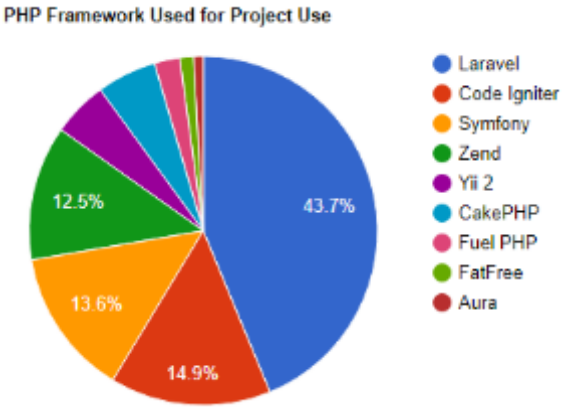
2.3.3. Laravel

Figure 11 : Laravel

Source : https://laravel.com/

Laravel fait partie des nombreux frameworks web utilisant PHP. Comme on peut le voir sur cette analyse de coderseye, Laravel reste le framework le plus utilisé dans les projets PHP actuellement (https://coderseye.com/best-php-frameworks-for-web-developers/).

Figure 12 : Liste des frameworks PHP



Source : https://coderseye.com/best-php-frameworks-for-web-developers/

Laravel est un framework utilisant l'architecture MVC, il a été conçu pour créer des applications plus rapidement. Il possède une très grande communauté d'utilisateurs ainsi qu'une documentation très complète. Pour ce qui est des fonctionnalités, Laravel fournit un ORM pour l'accès à la base de données ou encore un gestionnaire de paquets pour intégrer de nouvelles librairies à son projet plus facilement (Laravel, s.d.).

2.3.4. Express

Express est un framework web simple et rapide pour Node.js. En effet, Node.js est un langage web de programmation côté serveur utilisant le JavaScript. Il utilise le moteur d'exécution haute performance V8 de Google. En effet, Node.JS est un langage de programmation asynchrone, c'est-à-dire qu'il est non bloquant. Chaque requête effectuée à notre serveur s'exécutera sans avoir à attendre le résultat d'une autre opération (efficace pour les accès à la base de données qui prennent souvent du temps). Un système de « *callback* » a été conçu afin de pouvoir recevoir la réponse de la requête une fois celle-ci terminée (NodeJS, s.d.).

Figure 13 : NodeJS - Express

Source : (Sharma, 2017)

Express est donc une infrastructure web très simple, mais qui fournit de nombreuses fonctionnalités centrales sans forcément masquer celles de Node.js. Il apporte une couche supplémentaire à Node.js qui est de bas niveau (ExpressJS, 2017). Le principal avantage du NodeJS est que le langage utilisé pour la partie serveur est le même que pour le côté-client. Les développeurs n'ont qu'une seule technologie à connaître pour le développement de leur application.

De plus, Node.js possède un gestionnaire de paquets très complet comptant plus de 650'000 paquets. Il nous permet d'installer, d'utiliser ainsi que de publier de nouveaux paquets.

2.3.5. Synthèse des frameworks de développement

Concernant les frameworks de développement, les différentes fonctionnalités mentionnées correspondent très bien aux besoins du projet. Ils possèdent tous des architectures pouvant simplifier le travail d'un développeur.

De plus, étant donné la grandeur du projet, un ORM ainsi qu'un gestionnaire de paquets sont essentiels.

Pour ce qui est de l'intégration d'un de ces frameworks au sein de Novelis, NodeJS et PHP sont avantagés dû à leur renommée et à leur utilisation courante.

2.4. Scénario 2 : librairies graphiques

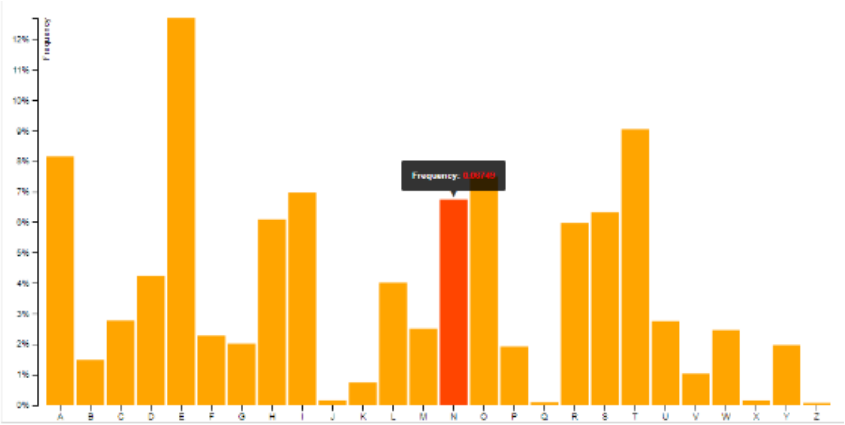
Cette partie de l'état de l'art consiste en l'analyse des librairies graphiques les plus populaires.

2.4.1. D3.js - Data-Driven Documents

D3.js est une librairie Javascript open-source permettant la création de graphiques sur la base de données. Ces différentes visualisations sont personnalisables à l'aide du HTML et du CSS[10]. D3.js utilise les capacités des navigateurs modernes en créant des animations interactives au format SVG[11] qui est un format de données utilisées pour créer des graphiques vectoriels (D3.js - Data-Driven Documents, 2017).

Il existe de nombreuses possibilités de visualisation telles que les « *box plots* », les « *scatterplot* » ou des plus connues comme le graphique en barres. Cependant, cette bibliothèque requiert plus de configurations que d'autres librairies dues à la complexité de certains graphiques.

Figure 14 : Graphique en barres avec info-bulle sur D3.js



Source : http://bl.ocks.org/Caged/6476579

Cette bibliothèque est assez flexible, en nous permettant de créer des graphiques adaptés à nos besoins. Elle est capable de gérer de grands jeux de données. Le code source complet est disponible sur GitHub.

[10] CSS : Cascading Style Sheets

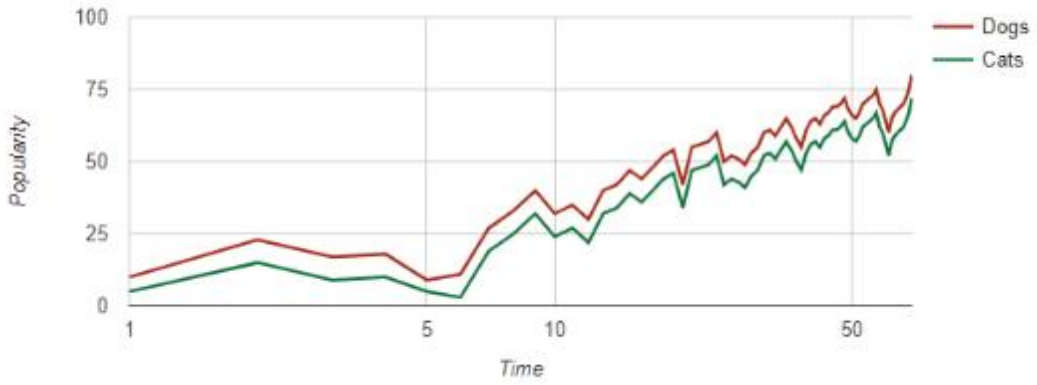
[11] SVG : Scalable Vector Graphics

2.4.2. Google Charts

Google Charts est un outil pour créer des graphiques de façon simple et rapide. Son usage est entièrement gratuit, cependant les données utilisées sont envoyées sur un serveur de l'entreprise. Il n'est pas possible d'importer les fichiers de la bibliothèque directement, l'utilisation des graphiques se fait via l'API[12] de Google Charts.

De nombreuses possibilités de graphiques sont disponibles selon nos besoins. De plus, il existe différentes options permettant de personnaliser nos graphiques. Google Charts est compatible avec les différents navigateurs modernes ainsi que les formats mobiles.

Figure 15 : Google Charts - Graphique linéaire



Source : https://google-developers.appspot.com/chart/interactive/docs/gallery/linechart

Il existe plusieurs façons de collecter des données pour ensuite les utiliser avec Google Charts. Les formats les plus utilisés sont le CSV et le JSON (Google Charts, 2017).

2.4.3. Chartist.js

Chartist est une librairie graphique open-source simple à configurer et surtout adaptative au support utilisé (smartphone, ordinateur portable…). Elle a été développée par une communauté d'utilisateurs ayant été, par le passé, déçue de certaines librairies. Les graphiques sont personnalisables, flexibles et compatibles avec des navigateurs comme Internet Explorer 9 pour les fonctionnalités de base (Chartist.js, s.d.).

[12] API : Application Programming Interface

Figure 16 : Chartist.js

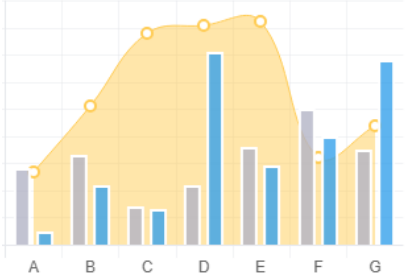
Source : https://gionkunz.github.io/chartist-js/

Le code source complet de Chartist est également disponible sur GitHub.

2.4.4. Chart.js

Chart.js est une librairie JavaScript conçue pour les développeurs souhaitant créer des graphiques, simplement et rapidement. Il est entièrement open-source et disponible sur GitHub. Il y a huit types de graphiques différents que l'on peut animer ou mixer selon nos besoins. Il utilise la technologie des « canvas » qui est compatible avec les navigateurs modernes à partir d'Internet Explorer 9. Les graphiques sont également adaptatifs à la taille de l'écran utilisé (Chart.js, s.d.).

Figure 17 : Graphique mixte sur Chart.js



Source : https://www.chartjs.org/

2.4.5. Synthèse des librairies graphiques

Nous pouvons constater que les librairies graphiques sont toutes plus ou moins adaptées à nos besoins. Cependant, la complexité des graphiques proposée par D3.js n'est pas forcément utile compte tenu de nos exigences.

De plus, Chart.js et Chartist.js sont compatibles avec de plus anciens navigateurs comme Internet Explorer qui est également utilisé à Novelis. Ils sont également adaptatifs à la taille de l'écran utilisé.

Google Charts est également un outil intéressant excepté que les données sont envoyées sur les serveurs de Google.

3. Choix

Cette partie de l'analyse va nous permettre de comparer les différents outils présentés durant l'état de l'art. Pour chaque catégorie d'outils, des critères d'évaluation avec une pondération spécifique ont été établis. Tous les outils seront ensuite comparés sur la base de ces critères à l'aide d'une matrice décisionnelle. Cette dernière nous permet de faire le choix final.

Pour chaque critère, une note de zéro à deux est attribuée, la liste suivante nous permet de comprendre l'échelle de valeurs utilisée dans notre matrice décisionnelle.

- 0 : le critère d'évaluation n'est pas atteint
- 1 : le critère d'évaluation est partiellement atteint
- 2 : le critère d'évaluation est entièrement atteint

3.1. Outils de récolte de données et de visualisation

3.1.1. Critères d'évaluation

- Récolte : l'outil propose une interface intuitive pour la récolte des données
- Visualisation : les graphiques disponibles sont pertinents
- Intégration : l'outil est facilement intégrable à Novelis

3.1.2. Matrice décisionnelle

	Récolte	Visualisation	Intégration	Total
Pondération	**2**	**1**	**2**	
Tableau	0	2	1	**4**
QlikView	0	2	1	**4**
PowerBI	0	2	1	**4**

Tableau 1 : Matrice décisionnelle des outils de récolte de données et de visualisation

3.1.3. Synthèse

À l'aide de la matrice décisionnelle, nous pouvons voir que tous les outils sont similaires d'un point de vue de récolte de données ainsi que de visualisation. Étant donné que ces outils « all-in-one » ne possèdent pas d'interface intuitive pour la récolte des données, notre propre solution est donc bénéfique. Elle sera intégrée par la suite à Novelis.

3.2. Technologies de base de données

3.2.1. Critères d'évaluation

- Gratuité : l'outil est gratuit
- Plateforme : l'outil est multiplateforme (Windows, Linux)
- Fonctionnalité : les fonctionnalités proposées sont utiles
- Intégration : l'outil est facilement intégrable à Novelis

3.2.2. Matrice décisionnelle

	Gratuité	Plateforme	Fonctionnalité	Intégration	Total
Pondération	**1**	**2**	**1**	**2**	
MariaDB	2	2	2	2	**12**
PostgreSQL	2	2	2	0	**8**
SQL Server	0	2	2	0	**6**

Tableau 2 : Matrice décisionnelle des technologies de base de données

3.2.3. Synthèse

À l'aide de ce comparatif, nous constatons que notre choix se porte sur MariaDB. En effet, Novelis utilise déjà cette technologie de base de données pour ses projets, ce qui renforce notre décision. Dans un autre contexte, nous nous serions peut-être dirigés sur PostgreSQL qui a fait ses preuves à travers le temps depuis sa création. De plus, c'est un système de gestion de base de données que nous n'avons jamais utilisé durant notre formation scolaire. SQL Server quant à lui, n'est disponible qu'en version d'évaluation pour une durée limité.

3.3. Frameworks de développement web

3.3.1. Critères d'évaluation

- Gratuité : l'outil est gratuit
- Fonctionnalité : les fonctionnalités proposées sont utiles
- Plateforme : l'outil est multiplateforme (Windows, Linux)
- Stabilité : l'outil a fait ses preuves à travers le temps
- Intégration : l'outil est facilement intégrable à Novelis

3.3.2. Matrice décisionnelle

	Gratuité	Fonctionnalité	Plateforme	Stabilité	Intégration	Total
Pondération	**1**	**1**	**2**	**2**	**2**	
Django	2	2	2	2	**1**	**14**
Ruby on Rails	2	2	2	2	**1**	**14**
Laravel	2	2	2	2	**2**	**16**
NodeJS/Express	2	2	2	2	**2**	**16**

Tableau 3 : Matrice décisionnelle des frameworks de développement web

3.3.3. Synthèse

Nous observons que les frameworks de développement web choisis sont tous similaires. Ruby on Rails et Django sont deux frameworks très intéressants, cependant leur intégration au sein de Novelis requiert plus de configurations que pour Laravel (PHP) et Express (NodeJS).

Il nous reste donc Laravel et Express qui sont similaires d'un point de vue global. Malgré tout, notre choix se porte sur NodeJS/Express pour son aspect asynchrone (non bloquant). De plus, le langage de programmation pour la partie serveur et celle client est le même, ce qui facilite la tâche pour un développeur devant reprendre le projet dans le futur.

3.4. Librairies graphiques

3.4.1. Critères d'évaluation

- Licence : l'outil est « open-source »
- Simplicité : l'outil est simple à mettre en place
- Graphique : les graphiques disponibles sont pertinents
- Compatibilité : l'outil est compatible avec des anciens navigateurs

3.4.2. Matrice décisionnelle

	Licence	Simplicité	Graphique	Compatibilité	Total
Pondération	**1**	**1**	**2**	**2**	
D3.js	2	1	1	1	**7**
Google Charts	0	1	2	0	**5**
Chartist.js	2	2	2	1	**10**
Chart.js	2	2	2	2	**12**

Tableau 4 : Matrice décisionnelle des librairies graphiques

3.4.3. Synthèse

Cette dernière matrice décisionnelle nous montre la similitude entre Chartist.js et Chart.js, qui sont deux librairies open-source très simples à utiliser et surtout légères. L'incompatibilité de Google Charts avec les plus anciens navigateurs le laisse un peu en retrait. De plus, l'envoi des données sur l'API de Google Charts nous pousse à nous diriger sur une autre librairie. D3.js est également une très bonne librairie. Cependant, la complexité des graphiques ainsi que l'interaction avec ces derniers ne sont pas nécessaires dans notre projet.

Pour finir, Chartist.js et Chart.js proposent une solution facile à implémenter. La grande différence est que les animations de Chartist.js ne sont pas supportées par des navigateurs comme Internet Explorer.

Nous choix se porte donc sur Chart.js.

3.5. Synthèse des choix technologiques

En définitive, notre choix final se porte donc sur la création de notre propre application en utilisant la base de données MariaDB, le framework de programmation web Express ainsi que la librairie graphique Chart.js. Nous passons maintenant au développement de notre solution.

4. Développement

Dans ce chapitre, les différents outils et technologies utilisés pour le développement de notre application sont listés. Par la suite, nous décrirons notre solution web en expliquant le schéma de la base de données, l'architecture de notre application et l'arborescence des dossiers. Cependant, nous n'entrerons pas dans les détails, car ceci n'est pas une documentation technique. Cette dernière se trouve sur le CD-ROM joint au rapport.

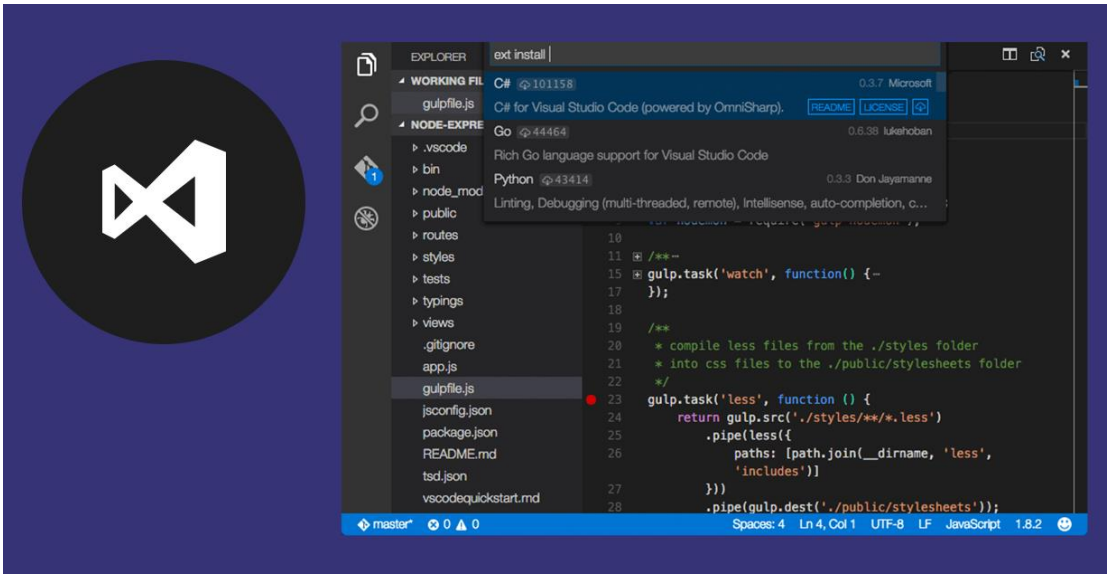
Étant donné que MariaDB et NodeJS (Express) ont déjà été mentionnées dans l'état de l'art, ces technologies ne seront pas répétées dans cette partie. Cependant, elles font partie intégrante de notre application. En ce qui concerne Chart.js, il n'a pas été nécessaire d'implémenter cette fonctionnalité, à la suite de la modification en cours de route de la ligne directrice de notre application. Ces changements sont expliqués dans le chapitre « Application ».

4.1. Outils de développement

4.1.1. Visual Studio Code

Visual Studio Code est un éditeur de code optimisé pour le développement et le débogage d'applications web. Ce logiciel est multiplateforme. De nombreuses extensions sont disponibles afin d'ajouter de nouveaux langages, thèmes ou autres services. Ces extensions sont exécutées dans des processus séparés pour ne pas ralentir l'éditeur (Visual Studio Code, 2018).

Figure 18 : Visual Studio Code



Source : https://code.visualstudio.com/

Cet outil gratuit et open-source offre de nombreuses fonctionnalités telles que l'autocomplétion qui permet d'éviter des erreurs de syntaxe en proposant des choix selon le type de variables ou les modules importés. De plus, il comprend une extension Git permettant de récupérer le code depuis un dépôt ou d'envoyer les modifications effectuées directement depuis l'éditeur.

4.1.2. Gitlab

Gitlab permet de gérer toutes les étapes du cycle de vie de développement d'un produit. Les équipes de développeurs peuvent gérer le code source du projet à l'aide du gestionnaire de versions Git. Le chef de projet quant à lui, peut s'assurer que toutes les « user-stories[13] » ont bien été implémentées. Cet outil est similaire à Github à l'exception que sa version gratuite nous autorise à créer des projets privés. L'accès au projet peut être donné à certaines personnes en fournissant leurs adresses e-mail (Gitlab, s.d.).

Figure 19 : GitLab

Source : https://about.gitlab.com/

Tout dépôt Git devrait contenir un fichier « gitignore » qui permet de spécifier au gestionnaire de versions les dossiers à ne pas prendre en compte. En effet, tout projet NodeJS contient un dossier nommé « node_modules » qui n'a pas besoin d'être dans le dépôt.

4.2. Technologies Backend[14]

4.2.1. NPM

NPM est le gestionnaire de paquets pour JavaScript. Un paquet est une archive contenant des fichiers nécessaires à l'installation d'un programme. NPM nous permet donc de rechercher des paquets selon nos besoins, de les télécharger puis de les installer. Il est également possible de publier ses propres paquets au besoin. NPM est automatiquement installé lors du téléchargement de NodeJS (npmjs, s.d.). L'intégralité des technologies utilisées dans ce travail a été installée à l'aide de ce gestionnaire.

[13] Fonctionnalité à implémenter dans un projet

[14] Backend : Couche d'interaction avec le serveur

4.2.2. Sequelize

Sequelize est un logiciel permettant de faire le lien entre les tables de notre base de données et les modèles d'objets. Chaque modèle correspond donc à une table de la base de données. Cela facilite grandement les interactions avec la base de données pour les développeurs. Outre la possibilité d'insérer ou de récupérer des données, cet outil offre d'autres fonctionnalités comme les transactions ou les associations entre les tables (Sequelizejs, s.d.).

4.2.3. Socket.io

Socket.io est une librairie Javascript permettant la communication en temps réel entre le client et le serveur. Il pourrait nous permettre de développer des applications telles qu'un chat ou pour la modification d'un document en ligne. L'avantage de cette librairie est que tous les utilisateurs peuvent interagir en même temps et voient les modifications apportées par les autres membres en temps réel (Socket.io, s.d.).

4.2.4. Bcrypt

Bcrypt est une librairie permettant de hasher les mots de passe. Il est basé sur l'algorithme de chiffrement « Blowfish ». Toutes les fonctions sont disponibles en synchrone ainsi qu'en asynchrone, cependant il est recommandé d'utiliser les méthodes asynchrones afin de conserver le principe non bloquant du NodeJS (Kelektiv, s.d.).

4.2.5. Multer

Multer est une librairie permettant de gérer les formulaires pour les importations de fichiers. En effet, à de multiples reprises, l'utilisateur va importer des fichiers sur notre site web. À l'aide de cette librairie, il est possible de récupérer les fichiers importés, de les stocker sur le serveur pour finalement en retirer le contenu nécessaire. Le type de fichiers supporté ne dépend pas de Multer, cependant la plateforme est configurée pour n'accepter que certains types de fichiers (.txt et .xlsx entre autres) afin d'éviter d'avoir des erreurs (expressjs, s.d.).

4.2.6. File System

File System est une librairie native à NodeJS, elle est installée par défaut sur chaque application NodeJS. Comme son nom l'indique, elle permet d'interagir avec des fichiers. Concernant la lecture de fichiers, il est préférable de l'utiliser pour les fichiers textes. De nombreuses librairies permettant de lire des CSV ou des Excel sont dérivées de cette dernière (Node.js, s.d.).

4.2.7. XLSX

Xlsx est une librairie permettant de lire le contenu d'un fichier Excel. À la suite de l'importation des fichiers sur le serveur grâce à Multer, le contenu du fichier Excel importé peut être récupéré pour ensuite stocker les résultats dans la base de données. Une fois le fichier traité par la librairie, ce dernier retourne un objet JSON avec tous les classeurs ainsi que les cellules du fichier (SheetJS, s.d.).

4.2.8. CSV-Writer

CSV Writer est une librairie utilisée pour générer des fichiers CSV. Celle-ci est très flexible, car il est possible de créer le CSV depuis un objet Javascript ou directement depuis un tableau. Il est important de spécifier les en-têtes du fichier ainsi que les valeurs liées à ces dernières (ryu1kn, s.d.).

4.2.9. Apache

Apache est un serveur web « open-source » créé en 1995. Il est considéré comme le serveur web le plus populaire depuis avril 1996. Il est utilisé dans ce projet afin de déployer la solution web au sein du réseau de Novelis. Plusieurs modules peuvent être intégrés à Apache pour gérer des services comme la redirection ou la réécriture d'URL (Apache, 2018).

En général, une application NodeJS dispose de son propre serveur, cependant Apache est utilisé pour rediriger l'URL du serveur de l'entreprise vers le serveur de l'application tournant en arrière-plan.

4.2.10. Forever

Forever offre la possibilité de lancer un programme en arrière-plan. En cas d'erreurs dans le processus, il garantit que l'application ne s'arrête pas. Il y a deux façons de lancer Forever : en utilisant la ligne de commande ou en l'intégrant directement dans son code source. De plus, Forever fournit un système d'historiques d'évènements (log) pour le débogage (foreverjs, s.d.).

4.3. Technologies Frontend[15]

4.3.1. Axure

Axure est un programme permettant de créer des « mock-up[16] » sans avoir à coder. Une version d'essai est disponible pendant 30 jours, cependant l'école fournit une licence afin de bénéficier de cette solution pour une durée illimitée. Cet outil très puissant permet de créer des vues adaptables au support ou de gérer des formulaires comme sur un vrai site internet (Axure, s.d.).

La création de ces prototypes nous a permis de gagner du temps lors du développement. En effet, ces derniers ont été présentés aux personnes en charge du projet à Novelis puis ont été validés afin d'éviter tous problèmes dans le futur.

4.3.2. Materialize CSS

Materialize CSS est un framework utilisé pour créer des interfaces utilisateurs modernes et ergonomiques. Ces dernières sont adaptatives au format du support utilisé (téléphone mobile, tablette ou ordinateur). L'utilisation de cette librairie permet un gain de temps considérable, car un style par défaut a été attribué à tous les composants (boutons, champs de texte…). Si l'on veut modifier ce style, il suffit de modifier le style global pour que celui-ci soit appliqué automatiquement à tous nos composants (Materialize CSS, s.d.).

De plus, une documentation avec des exemples de code est fournie afin d'aider les nouveaux utilisateurs à débuter avec cette librairie. Il est également possible de donner un feed-back sur Materialize ou de poser une question en cas d'incompréhension d'une fonctionnalité.

Ce framework est basé sur les principes du « Material Design », qui est un ensemble de règles proposées par Google afin de créer des interfaces utilisateurs ergonomiques et intuitives. Ce concept est également utilisé pour les applications de smartphone Android. Le but final de cette pratique est d'offrir à l'utilisateur le même type d'interfaces peu importe l'appareil utilisé.

4.3.3. Pug

Pug est un moteur de template permettant de générer du code HTML. Ce dernier permet au développeur de mettre des conditions ou des boucles directement dans leurs vues. Le code sera ensuite retranscrit en HTML pour être supporté par les navigateurs. Le système de balisage n'existe pas en Pug, il utilise l'indentation comme certains langages de programmation comme le Python.

[15] Frontend : Couche d'interaction avec l'utilisateur

[16] Mock-up : Prototype d'interfaces utilisateur

C'est-à-dire que si un élément est contenu dans un autre, il devra avoir une indentation supplémentaire que son élément parent (Pugjs, s.d.).

Figure 20 : Exemple de boucle et d'indentation en Pug

```
-
  var list = ["Uno", "Dos", "Tres",
          "Cuatro", "Cinco", "Seis"]
each item in list
  li= item
```

```
<li>Uno</li>
<li>Dos</li>
<li>Tres</li>
<li>Cuatro</li>
<li>Cinco</li>
<li>Seis</li>
```

Source : https://pugjs.org/language/code.html

Au lancement de ce projet, le nom de la librairie était Jade. Cependant, l'entreprise a été forcée de changer de nom, car Jade était une marque déposée. C'est pourquoi lorsque l'on recherche des informations sur Pug, il peut arriver que le nom Jade apparaisse.

4.3.4. JQuery

JQuery est un framework « open-source » et léger dérivé du Javascript. En effet, son installation ne requiert que 30 kilo-octets. En quelques lignes de code, il est possible de manipuler des objets HTML ou exécuter des requêtes Asynchronous JavaScript And XML (Ajax). De plus, il est supporté par tous les navigateurs récents (jQuery, 2018).

4.3.5. JQuery Validation Plugin[17]

JQuery Validation Plugin offre une validation pour les formulaires côté-client. C'est-à-dire, qu'il est possible de paramétrer les champs des formulaires en spécifiant à l'utilisateur de rentrer un nombre à virgules par exemple. De nombreuses autres fonctionnalités sont également disponibles comme la vérification d'emails, de numéro de téléphones et autres… En cas de besoin, l'API de JQuery Validation autorise à créer des validations personnalisées. Les messages d'erreurs apparaissent en anglais, mais sont disponibles dans 37 autres langues.

Cependant, le but de ce plugin n'est pas de remplacer la vérification côté-serveur. Il apporte uniquement une plus grande interaction avec l'utilisateur ainsi qu'une couche de sécurité supplémentaire (jQuery Validation Plugin, s.d.).

[17] Plugin : Module d'extension

4.3.6. DataTables

DataTables est une bibliothèque gratuite et « open-source » pour l'affichage de tables sur les sites internet. Cette librairie est basée sur le jQuery. Elle ajoute de nombreuses fonctionnalités aux tables comme la pagination, la recherche instantanée, le filtrage ou encore le tri des colonnes. La source des données peut provenir du client comme du serveur en utilisant les requêtes Ajax. Tout le monde peut contribuer à cette librairie en développant de nouveaux modules (DataTables, 2018).

Figure 21 : DataTables exemple

Show 10 entries Search:

Name	Position	Office	Age	Start date
Airi Satou	Accountant	Tokyo	33	2008/11/28
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12
Bradley Greer	Software Engineer	London	41	2012/10/13
Brenden Wagner	Software Engineer	San Francisco	28	2011/06/07
Brielle Williamson	Integration Specialist	New York	61	2012/12/02
Bruno Nash	Software Engineer	London	38	2011/05/03
Caesar Vance	Pre-Sales Support	New York	21	2011/12/12
Cara Stevens	Sales Assistant	New York	46	2011/12/06
Cedric Kelly	Senior Javascript Developer	Edinburgh	22	2012/03/29
Name	Position	Office	Age	Start date

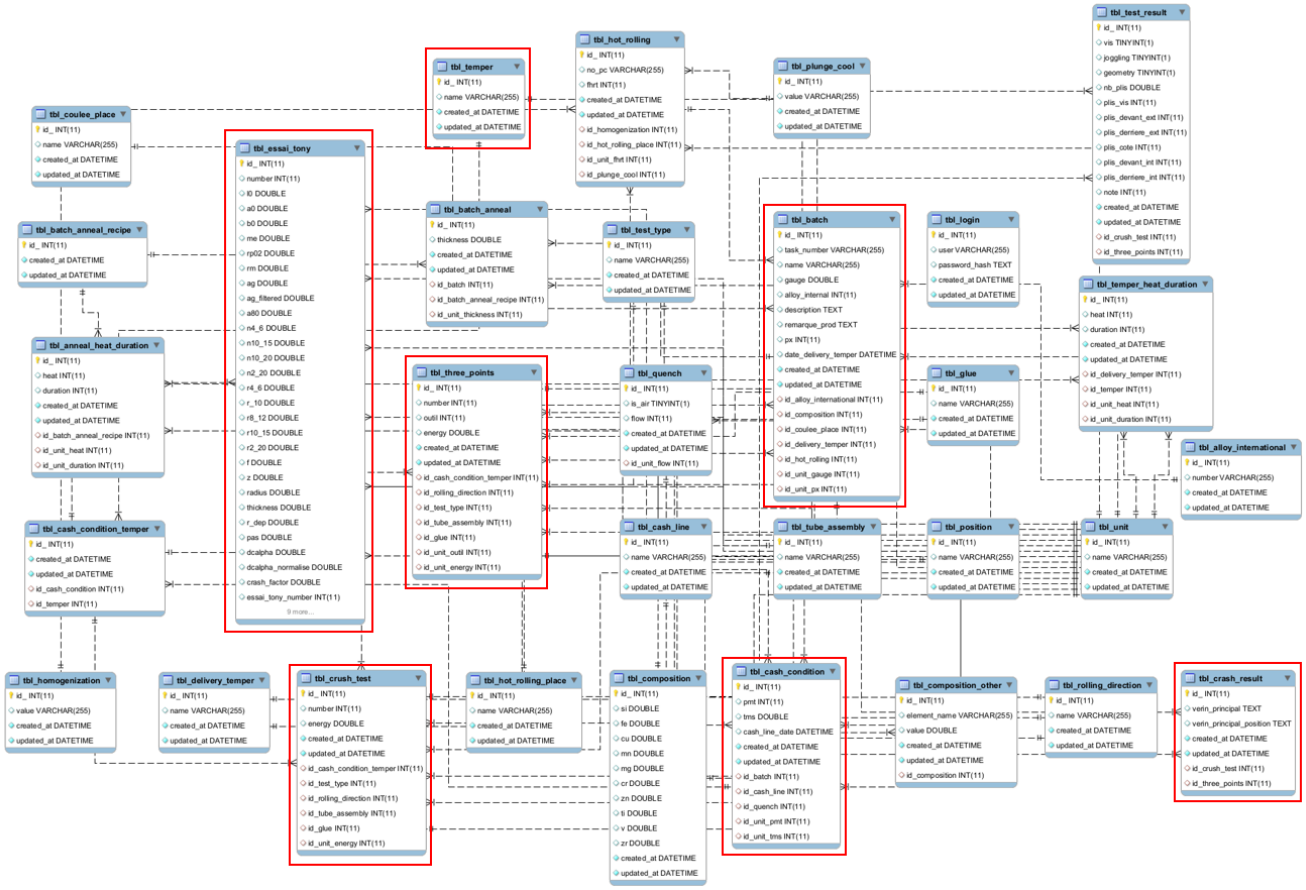
Showing 1 to 10 of 57 entries Previous 1 2 3 4 5 6 Next

Source : https://datatables.net/

4.4. Base de données

4.4.1. Schéma de la base de données

Figure 22 : Schéma de la base de données



4.4.2. Structure générale de la base de données

À la suite d'une discussion avec le personnel informatique de Novelis, il nous est demandé de respecter les conventions et bonnes pratiques de l'entreprise afin de garantir une certaine linéarité entre ce projet et leurs projets existants. Voici la liste des éléments requis dans la base de données tirée de la documentation technique de Novelis (Novelis Tosi, 2018, p. 7) :

- Utilisation du moteur de stockage INNODB pour les nouvelles tables
- Le nom des tables et des champs d'une table doit être en minuscule afin d'éviter les problèmes de « case-sensitive » entre Windows et Linux
- Le nom des tables doit commencer avec le préfixe « tbl » (ex : tblmachine).
- Le nom des tables et des champs peut utiliser le caractère « _ » pour séparer des mots (ex : tbl_machine).
- Dans une table, la clé primaire commence par le préfixe « id_ ».
- Les champs « created_at » et « updated_at » sont requis dans toutes les tables afin de garder un historique.

Pour notre part, nous avons décidé d'utiliser le caractère « _ » pour la séparation des mots afin de faciliter la lisibilité.

La base de données comprend 31 tables, cependant nous en présentons uniquement sept pour comprendre le but principal de ces tables et les liens entre ces dernières. Une grande partie des autres tables représentent des listes à choix ou encore des tables intermédiaires pour les relations.

4.4.3. tbl_batch

La table « batch » correspond au lot d'aluminium à sa sortie de l'usine, c'est l'élément central de la base de données, une grande partie des autres tables seront liées à cette dernière. Il comprend des éléments comme l'alliage international, l'alliage interne, l'épaisseur du lot ou la composition chimique.

4.4.4. tbl_cash_condition

Une fois notre lot d'aluminium à l'état F, c'est-à-dire « as-fabricated », des traitements supplémentaires chimiques ou thermiques sont appliqués pour arriver à l'état final (état de livraison au client). Ces traitements sont donc des « cash condition », leur but est de réchauffer le rouleau de manière continue puis de le refroidir avec de l'air ou avec de l'eau. Les principales mesures sauvegardées dans la base de données sont le type de refroidissement ainsi que des données mécaniques comme le « PMT » et le « TMS ». Pour chaque « batch », il est possible d'avoir plusieurs « cash condition ».

4.4.5. tbl_temper

La table « temper » correspond à des traitements thermiques effectués pour simuler le processus une fois la marchandise arrivée chez le client (ex : la peinture). Chaque table possède donc un champ nom, plusieurs durées (minutes ou heures) et plusieurs températures (°C). Ceci est la dernière étape avant les tests crashs.

4.4.6. tbl_three_points

La table « three_points » est le premier type de test effectué sur un échantillon d'aluminium. Chaque test effectué a donc un « temper », qui est lié à une « cash condition », qui est lié au « batch ». Les principaux éléments sauvegardés sont l'énergie absorbée, l'assemblage (vis ou colle), la direction (long, travers, diagonal) et le numéro d'échantillon.

4.4.7. tbl_crush_test

La table « crush_test » est identique à la table « three_points » excepté que le test effectué en laboratoire n'est pas le même.

4.4.8. tbl_essai_tony

La table « essai_tony » correspond à des essais mécaniques faits au laboratoire de production. Chaque entrée représente un échantillon d'aluminium avec un numéro, une direction, une épaisseur ainsi que des valeurs mécaniques (rm, rp02…). Comme pour les crashs « three_points » et « crush », ils sont directement liés à un « temper ».

4.4.9. tbl_crash_result

Cette dernière table représente les résultats des crashs « three_points » et « crush ». Les champs verin_principal et verin_principal_position composent la table. Les données sont récupérées d'un fichier texte d'environ 3'000 lignes. Une réflexion en amont a été nécessaire pour le stockage des données dans cette table. En effet, nous ne devions pas nous concentrer sur une ligne en particulier, mais uniquement sur l'ensemble des résultats. À la suite d'un test de performance, pour l'insertion et la récupération des données, les résultats suivants sont constatés :

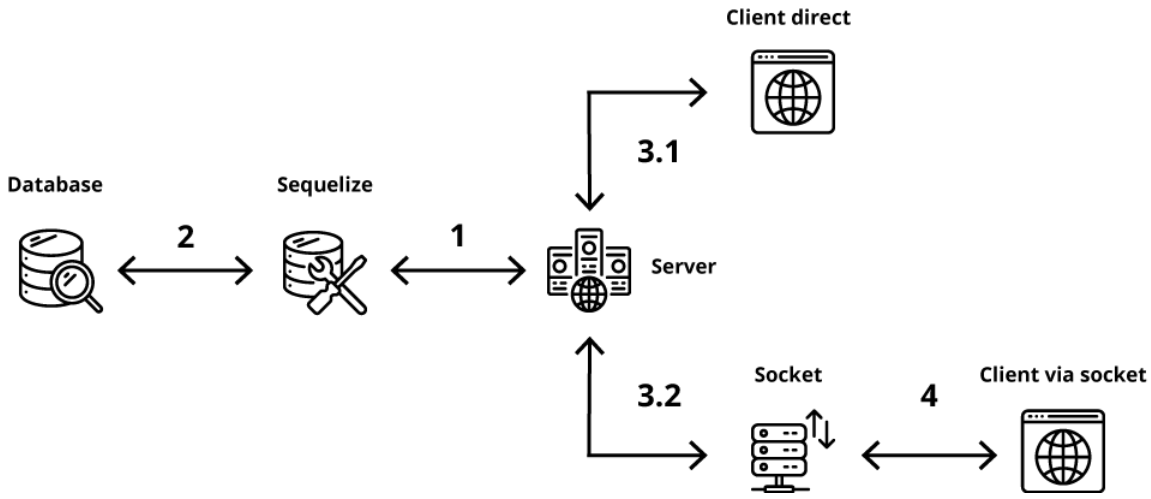
- Insertion et récupération ligne par ligne : ~2 secondes
- Insertion et récupération regroupée : ~ 100 millisecondes

C'est pourquoi nous avons décidé de regrouper toutes les informations dans un champ texte que nous séparons par des « ; ».

4.4.10. Architecture

Voici l'architecture globale de notre application.

Figure 23 : Architecture



Source : données de l'auteur

Cinq éléments composent l'architecture de notre application :

- Le serveur
- L'ORM Sequelize
- La base de données
- Le socket
- Le client

Le serveur est l'élément central de notre application. Pour toutes requêtes à la base de données, il ne communique jamais directement avec cette dernière. Il passe par un intermédiaire qui est l'ORM. Son but est de transmettre la requête provenant du serveur et ensuite de lui retourner la réponse de la base de données.

La communication client-serveur et serveur-client peut se faire de deux manières différentes :

Ils peuvent interagir directement à l'aide du protocole Hypertext Transfer Protocol (HTTP). Cependant, le serveur ne fait que répondre aux requêtes du client. Ce procédé est utilisé lorsqu'il faut rediriger le client sur une autre page ou lors d'un import de fichier.

Le deuxième type de communication requiert l'utilisation d'un intermédiaire, en l'occurrence un WebSocket. Le WebSocket permet une communication en temps réel synchrone entre le client et le serveur. Avant l'apparition de cette technologie, la seule façon d'interagir avec le serveur depuis le

Javascript client était l'utilisation de l'Ajax. Cependant, il n'était pas possible au serveur de communiquer directement avec le client sans avoir reçu, au préalable, une requête de celui-ci. L'utilisation du WebSocket permet au serveur d'envoyer directement les réponses au client puis de rafraîchir instantanément les informations affichées à l'utilisateur sans avoir besoin de réactualiser la page. Son utilisation est relativement simple, chaque partie est en mesure d'émettre des requêtes ainsi que d'être à l'écoute des messages entrants.

4.5. Arborescence du projet

Cette section comprend l'arborescence de notre solution web. Nous y expliquons la structure de nos dossiers ainsi que les différents fichiers. Voici un aperçu global des principaux dossiers du projet.

Figure 24 : Arborescence du projet

```
- __bin__
- __config__
- __db__
- __migrations__
- __models__
- __modules__
- __public__
  - __DataTables__
  - __css__
  - __js__
  - __res__
- __res__
- __routes__
- __socket__
- __views__
- [app.js](Novelis-Crash-Test/app.js)
- [node_modules](Novelis-Crash-Test/node_modules)
- [package-lock.json](Novelis-Crash-Test/package-lock.json)
- [package.json](Novelis-Crash-Test/package.json)
```

Source : données de l'auteur

Le dossier « bin » contient le script permettant de lancer notre application. C'est dans ce dossier que l'on configure notre serveur ainsi que le port qui est écouté par ce dernier (port 3000 en l'occurrence).

Le dossier « config » contient la configuration de l'ORM dans le fichier config.json. Le nom de la base de données, l'utilisateur, le mot de passe ainsi que le type de base de données sont requis pour que Sequelize fonctionne.

Figure 25 : Configuration du fichier config.json

```
"development": {
  "username": "aleks",
  "password": "",
  "database": "novelis",
  "host": "127.0.0.1",
  "dialect": "mysql",
  "timezone": "+02:00",
  "logging": false
```

Source : données de l'auteur

Le dossier « db » contient toutes les requêtes CRUD (Create, Read, Update, Delete) à la base de données. Étant donné la taille de cette dernière, nous avons décidé de créer un fichier pour chaque table afin de garder une certaine lisibilité dans notre code.

Le dossier « models » correspond à toutes les tables de la base de données.

Le dossier « modules » correspond à la couche de traitement de notre projet, c'est là que sont implémentées les fonctionnalités d'insertion, de modification, de lecture de fichiers ou d'export.

Le dossier « public » contient tous les fichiers Javascripts et CSS clients ainsi que les librairies frontend.

Le dossier « routes » représente les contrôleurs de notre application. Les différentes URL utilisées et les redirections y sont implémentées.

Le dossier « socket » contient toutes les requêtes bidirectionnelles (client-serveur et serveur-client).

Le dossier « views » représente la couche de présentation de notre application.

5. Application

Ce chapitre liste les étapes de développement, la coordination de notre travail avec Novelis ainsi que les fonctionnalités majeures de notre application avec les problèmes rencontrés lors de l'implémentation de ces dernières.

5.1. Étapes de développement

Le développement de notre application s'est fait en plusieurs étapes. La première consistait à insérer toutes les informations dans la base de données afin de regrouper toutes les données de façon centralisée. L'insertion est découpée en plusieurs parties pour améliorer l'interaction avec l'utilisateur. Au départ, l'utilisateur doit remplir des formulaires pour insérer les données de base (batch, cash condition, temper…). Cependant, l'insertion des résultats des crashs se fait à l'aide d'importation de fichiers pour automatiser le processus.

Pour les listes à choix comme l'alliage international ou les lieux de coulée, une interface administrateur avec un mot de passe a été créée afin qu'uniquement les personnes autorisées puissent ajouter de nouvelles données.

Une fois ces données insérées, la visualisation de ces dernières a constitué la deuxième étape de notre processus de développement.

La troisième et dernière étape était de permettre à l'utilisateur d'afficher une liste de tous les « batch » pour lesquels un traitement thermique a été effectué et donc des crashs-tests par la même occasion. Le but de cette liste est de pouvoir filtrer les résultats selon nos besoins puis d'exporter les données en format CSV afin de pouvoir comparer les résultats.

La possibilité d'afficher les résultats des crashs-tests avec des graphiques n'a pas été implémentée. En effet, il a été convenu avec les personnes en charge du projet que les fonctionnalités de filtrage et d'exportation étaient plus importantes actuellement. Les employés de Novelis possèdent déjà des macros Excel permettant d'afficher des graphiques selon les résultats entrés. Ils n'auront plus qu'à « copier-coller » les résultats exportés dans le fichier Excel pour la visualisation de graphiques.

5.2. Coordination interne

La réalisation de ce travail est effectuée en coordination avec les responsables du projet à Novelis. Une gestion de projets similaire à la gestion SCRUM est utilisée. Des rendez-vous toutes les semaines ont été fixés, lors desquelles nous discutions de la priorité des fonctionnalités à implémenter. Le travail accompli la semaine précédente y est également présenté. Lors des derniers jours, il a fallu faire des choix par rapport aux fonctionnalités restantes. En effet, étant donné la taille du projet, il

n'était pas possible d'implémenter toutes les fonctionnalités discutées initialement. C'est pourquoi la priorisation était d'autant plus importante à ce moment-là.

Les personnes en charge du projet profitent également de cette occasion pour proposer des modifications ou valider les fonctionnalités implémentées. L'avis des employés et donc futurs utilisateurs nous a permis également de voir si notre application est intuitive et compréhensible par les personnes du métier.

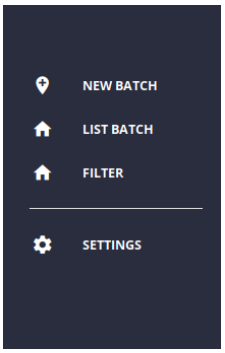
5.3. Fonctionnalités

Dans cette partie, nous allons décrire la liste des principales fonctionnalités de notre application.

5.3.1. Navigation

Un menu de navigation sur le côté a été implémenté afin de permettre à l'utilisateur de se rendre à l'endroit souhaité le plus facilement possible. L'utilisateur a la possibilité de créer un nouveau lot d'aluminium, d'afficher la liste de ces derniers, naviguer à la page de filtre ou modifier les listes à choix dans les paramètres (uniquement pour les administrateurs). La liste des « batch » n'est pas décrite, car elle représente simplement un tableau regroupant tous les lots avec la possibilité d'afficher les détails de ces derniers.

Figure 26 : Menu de navigation

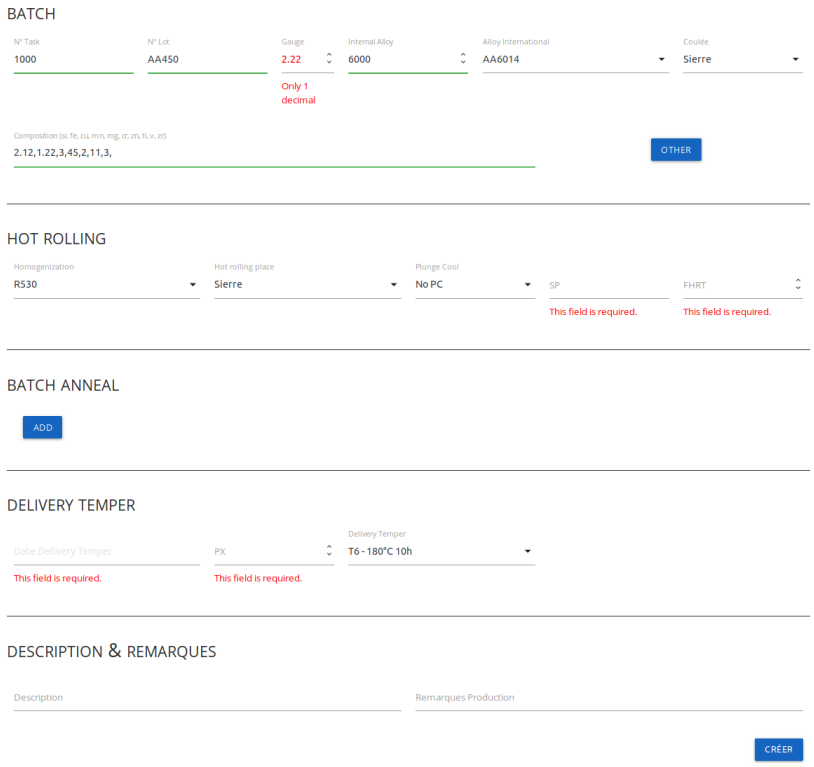


Source : données de l'auteur

5.3.2. Création d'un « batch »

Comme expliqué dans le chapitre de la base de données, le « batch » est l'élément central de notre plateforme, il est donc important d'offrir à l'utilisateur une façon simple et intuitive de créer un lot en s'assurant que ce dernier ne fasse pas d'erreurs lors de la création.

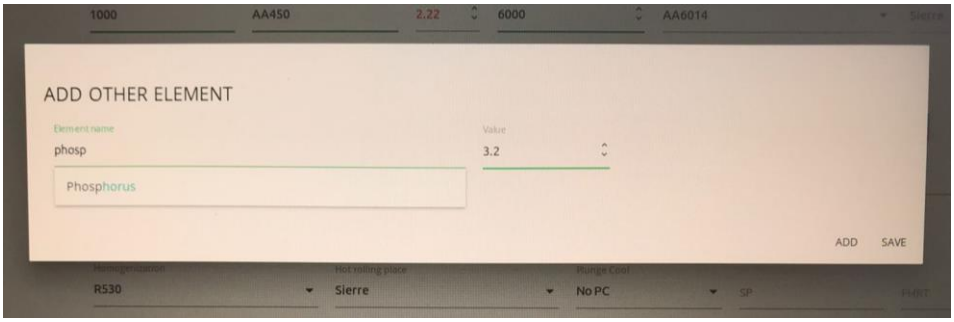
Figure 27 : Création d'un lot



Source : données de l'auteur

Ce formulaire est séparé en plusieurs étapes bien distinctes. Nous pouvons également apercevoir une gestion d'erreurs pour les champs obligatoires ainsi que le nombre de décimales maximales pour la « gauge » (épaisseur). En fournissant un message d'erreur à l'utilisateur, nous nous assurons que celui-ci comprenne la modification à apporter. En ce qui concerne la composition chimique, tout lot d'aluminium est composé de 10 éléments de base (silicium, fer, cuivre…). Cependant, il arrive que la production décide d'ajouter de nouveaux éléments à la composition. C'est pourquoi nous avons implémenté cette variante en proposant à l'utilisateur une autocomplétion provenant du tableau périodique des éléments.

Figure 28 : Autre composition chimique



Source : données de l'auteur

5.3.3. Détails d'un « batch »

Une fois la création du lot terminé, l'utilisateur peut afficher les détails de ce dernier. Toutes les informations sont reprises depuis la base de données.

Figure 29 : Détails d'un lot

BATCH DETAILS | CASH CONDITIONS

BATCH DETAILS

N° Task	N° Lot	Gauge	Alliage interne	Alliage international	Coulée	Plunge Cool	Sp	Fhrt	Hot Rolling Place	Homogenization
123	123	1.2mm	1000	AA6014	Sierre	No PC		1000°C	Sierre	R530

COMPOSITION

Silicium	Fer	Cuivre	Manganèse	Magnésium	Chrome	Zinc	Titane	Vanadium	Zirconium
1	2	3	4	5	6	7	8	9	10

DELIVERY TEMPER

Delivery Temper	Heat	Duration	Date
T6	180°C	10h	2018-07-18

BATCH ANNEAL

No batch anneal

DESCRIPTION & REMARQUES

desc | rem

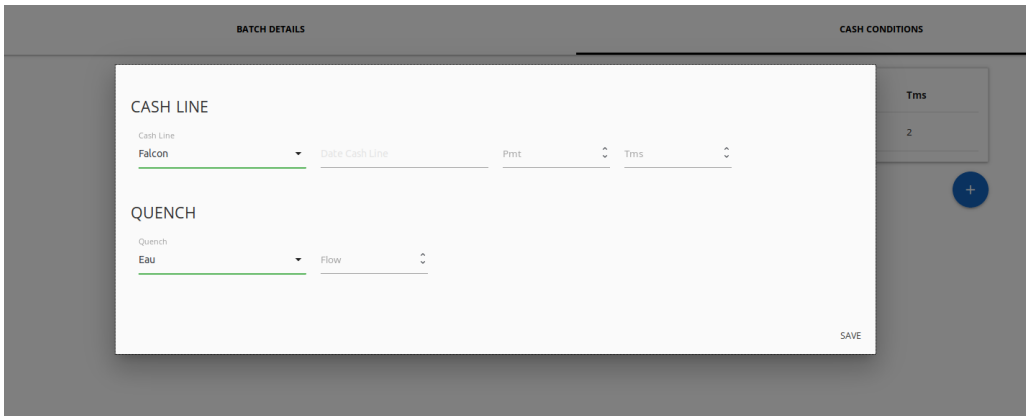
Source : données de l'auteur

Comme nous pouvons l'apercevoir, cette page est divisée en deux onglets : les détails du lot et les « cash conditions » (cf. point 4.4.4). Cette deuxième partie permet à l'utilisateur d'insérer des nouvelles cash conditions pour ce lot.

5.3.4. Création d'une « cash condition »

L'insertion de cet élément se fait à l'aide d'un formulaire. Si l'utilisateur choisit le type de refroidissement à eau, un champ pour le débit d'eau en m^3/h est dynamiquement créé. Afin d'éviter les erreurs de l'utilisateur, l'unité est insérée automatiquement dans la base de données.

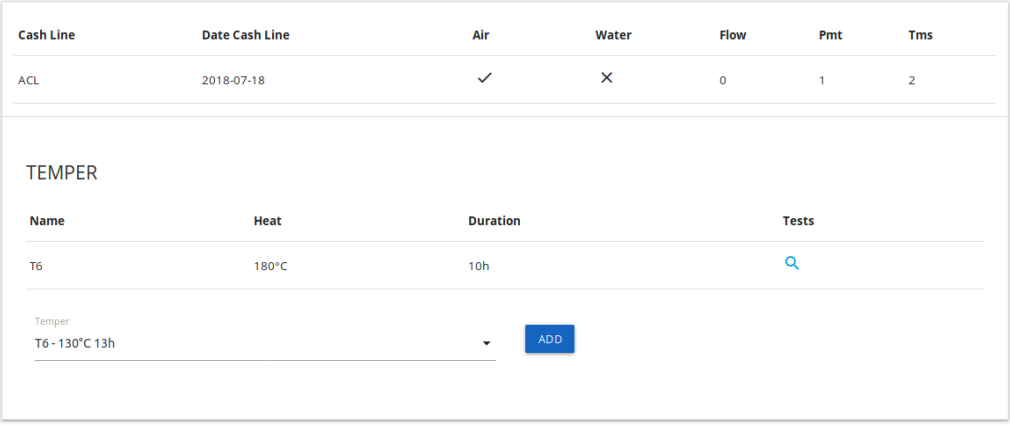
Figure 30 : Création d'une cash condition



Source : données de l'auteur

Une fois la condition insérée dans la base de données, l'utilisateur a la possibilité de la sélectionner pour afficher les « temper » liés à cette dernière.

Figure 31 : Détails d'une cash condition



Cash Line	Date Cash Line	Air	Water	Flow	Pmt	Tms
ACL	2018-07-18	✓	✕	0	1	2

TEMPER

Name	Heat	Duration	Tests
T6	180°C	10h	

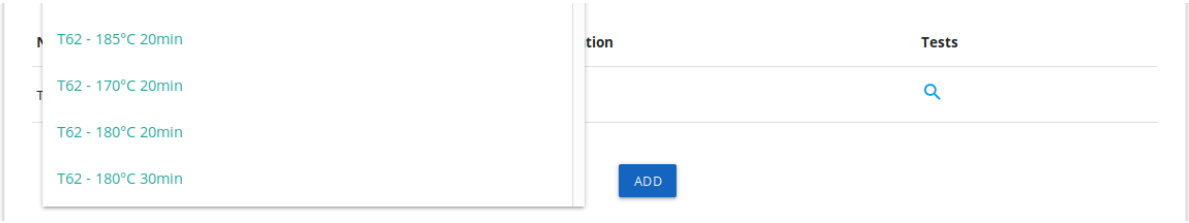
Temper
T6 - 130°C 13h
ADD

Source : données de l'auteur

5.3.5. Création d'un « temper »

Après sélection de la condition, l'utilisateur a la possibilité d'y ajouter des traitements thermiques qui sont suivis de « crashs-tests ». Il suffit simplement de sélectionner le « temper » qui nous convient dans la liste déroulante (il est également possible d'ajouter de nouveaux éléments dans cette liste dans la partie administration).

Figure 32 : Création d'un temper



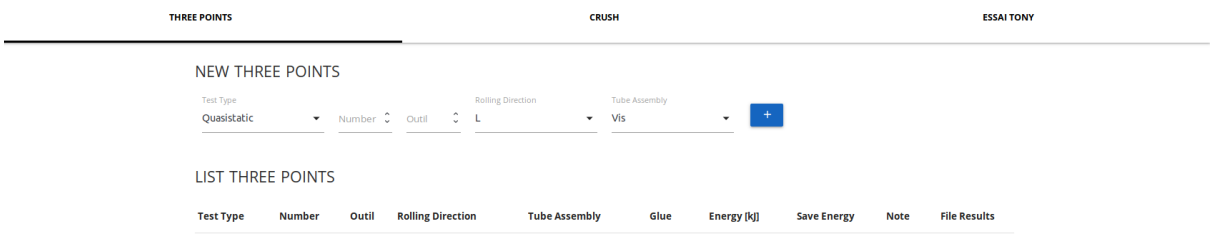
Source : données de l'auteur

Un système de vérification a été implémentée pour qu'un même « temper » ne puisse pas être ajouté plusieurs fois.

Nous allons maintenant parcourir les différents « crashs-tests » qui découlent de ces traitements thermiques. Pour arriver à la page correspondante, il suffit de cliquer sur la petite loupe située sous l'en-tête « Tests ».

5.3.6. Aperçu global des crashs-tests

Figure 33 : Vision globale des crashs-tests



Source : données de l'auteur

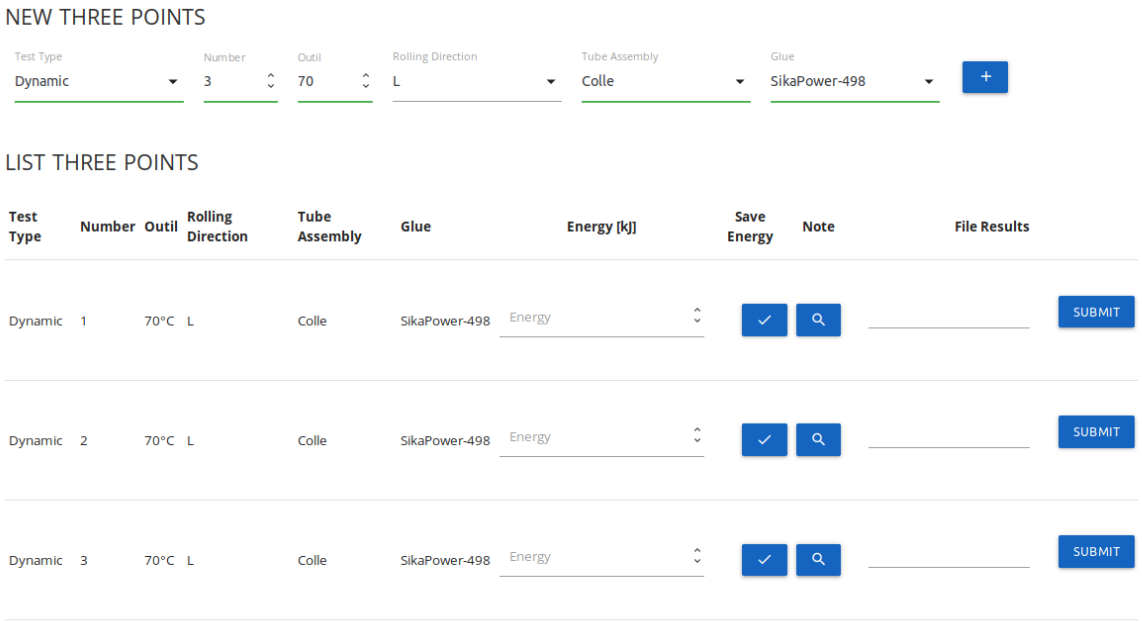
Nous pouvons apercevoir que cette page est divisée en trois parties distinctes : les tests « three points », « crushs » et les « essais Tony ». Pour passer d'une page à l'autre, il suffit de naviguer entre les onglets.

Pour la création de nouveaux tests, la partie « crush » n'est pas présentée, car celle-ci est similaire aux « three points ».

5.3.7. Création d'un test « three points »

L'insertion d'un nouveau test « three points » se fait par le biais d'un formulaire. Il faut sélectionner le type de test, entrer le nombre de tubes testés et sélectionner la direction ainsi que l'assemblage. Si l'assemblage sélectionné est la colle, un nouveau champ pour le nom de la colle est ajouté dynamiquement. Une fois le formulaire validé, les informations sont ajoutées dans la base de données puis affichées sur la page.

Figure 34 : Création de nouveaux tests three points



Source : données de l'auteur

Une entrée a été créée pour chaque tube testé. Cette étape nous a simplement permis de créer un nouveau test. Cependant, ce dernier n'a aucune utilité tant que les différents tests ne sont pas complétés en ajoutant l'énergie, les notes et les résultats. Nous ne présenterons pas tous ces éléments en détail, excepté pour les résultats des tests.

5.3.8. Création des résultats

Les résultats sont récupérés à l'aide d'un fichier généré par la machine effectuant le test dans l'usine du R&D de Novelis. Voici un aperçu de ce fichier.

Figure 35 : Résultats des tests

```
[0:6]   [2:1]
Vérin supérieur - Mesure force (externe)        DB verin principal\Position
kN
-0.228885       -94.2
-0.114443       -94.2
-0.221256       -94.2
0.907912        -12.1
0.831617        -12.4
0.93843 -12.5
0.831617        -12.8
0.93843 -13
0.816358        -13.2
0.900282        -13.5
```

Source : données de l'auteur

La taille du fichier varie selon les résultats, mais compte environ 3'000 lignes en moyenne. Les employés de Novelis n'ont donc plus qu'à importer le fichier de résultats pour le tube souhaité. Afin d'éviter les erreurs, ce formulaire d'importation n'accepte que les fichiers textes. Ce fichier est ensuite traité pour en récupérer les résultats de chaque ligne.

Figure 36 : Insertion des résultats

LIST THREE POINTS

Test Type	Number	Outil	Rolling Direction	Tube Assembly	Glue	Energy [kJ]	Save Energy	Note	File Results
Dynamic	1	70°C	L	Colle	SikaPower-498	Energy			Results already uploaded !
Dynamic	2	70°C	L	Colle	SikaPower-498	Energy			SUBMIT
Dynamic	3	70°C	L	Colle	SikaPower-498	Energy			SUBMIT

Source : données de l'auteur

Comme nous pouvons le constater sur l'image ci-dessus, les tubes pour lesquelles des résultats ont déjà été importés ont une annotation spécifique.

La principale difficulté de cette partie était la lecture du fichier et la récupération des résultats. Pour chaque ligne du fichier, il a fallu déterminer l'index de départ du premier numéro et l'index de

fin de ce dernier en vérifiant si le caractère est un chiffre, un « - » pour les chiffres négatifs ou un « . » pour les décimales et ainsi de suite pour le deuxième numéro. Comme vu dans le point 4.4.9, la totalité de ces résultats est ensuite regroupée en un seul et unique « String » ayant comme délimiteur des « ; » afin d'être stockée dans la base de données.

5.3.9. Création d'un « essai tony »

Le processus d'insertion de « l'essai tony » est similaire à celui du « three points ». Cependant, l'utilisateur doit saisir un numéro d'échantillon, une direction et une position. Il est possible d'ajouter d'autres essais à l'aide du bouton « + ». L'import des résultats se fait également directement dans le formulaire de création. Pour ces essais mécaniques, les résultats sont représentés par un fichier Excel également généré par la machine faisant le test.

Figure 37 : Création d'un essai tony

Source : données de l'auteur

L'illustration suivante présente un aperçu des résultats du fichier Excel.

Figure 38 : Résultats de l'essai tony

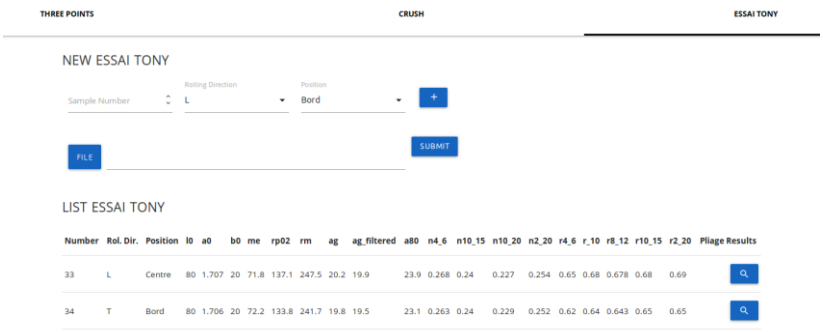
Date/heu	LOT	Direction	Etat	l0 L[lo 0] mm	a0 a[lo 0] mm	b0 b[lo 0] mm	me m[lo E] GPa	rp02 R[lo p0.2] MPa	rm R[lo m] MPa	ag A[lo g] %	ag_filtere A[lo g rép %	A80 A80 %	n4_6 n[lo 4-6/A	n10_15 n[lo 10-15	n10_20 n[lo 10-20	n2_20 n[lo 2-20/	r4_6 r[lo 4-6/A	r_10 r[lo 10%]	r8_12 r[lo 8-12/	r10_15 r[lo 10-15	r2_20 r[lo 2-20/	Id	z ValZ
########	1	TRAV		80	1.707	20.01	71.9	135.5	246.7	20	19.9	23.8	0.27	0.243	0.23	0.256	0.65	0.65	0.652	0.65	0.66	-	67.76
########	2	TRAV		80	1.713	20.01	71.3	132.6	241	20.1	19.8	23.5	0.265	0.242	0.23	0.253	0.62	0.64	0.636	0.64	0.65	-	66.53
########	3	TRAV		80	1.711	20.01	74	133.2	241.6	22.4	21.9	25.7	0.257	0.248	0.24	0.251	0.37	0.4	0.402	0.41	0.42	-	69.27
########	4	TRAV		80	1.719	20.01	71.9	139.7	250.3	20.1	19.9	24.1	0.264	0.239	0.227	0.251	0.69	0.71	0.709	0.71	0.72	-	64.79
########	5	TRAV		80	1.715	20.01	71.6	136.1	245.4	20.9	20.4	24.7	0.262	0.242	0.23	0.251	0.63	0.65	0.648	0.65	0.66	-	64.5
########	6	TRAV		80	1.72	20.01	73.3	137.7	246	22.1	21.7	25.5	0.251	0.246	0.239	0.247	0.38	0.4	0.403	0.41	0.41	-	61.36
########	33	TRAV		80	1.707	20	71.8	137.1	247.5	20.2	19.9	23.9	0.268	0.24	0.227	0.254	0.65	0.68	0.678	0.68	0.69	-	70.39
########	34	TRAV		80	1.706	20	72.2	133.8	241.7	19.8	19.5	23.1	0.263	0.24	0.229	0.252	0.62	0.64	0.643	0.65	0.65	-	62.77
########	35	TRAV		80	1.712	20	74.6	134.5	242.3	22.1	21.9	26.1	0.254	0.246	0.238	0.249	0.38	0.4	0.405	0.41	0.42	-	62.8
########	36	TRAV		80	1.724	20	72.1	142.9	252.1	20.3	19.9	24.5	0.258	0.235	0.223	0.246	0.7	0.72	0.718	0.72	0.72	-	62.67
########	37	TRAV		80	1.716	20	72.4	138.5	246.3	20.3	20.1	24.4	0.257	0.238	0.226	0.247	0.61	0.63	0.633	0.64	0.64	-	65.97
########	38	TRAV		80	1.722	20	74.4	138.5	245.3	23	22.5	27.2	0.249	0.243	0.236	0.244	0.39	0.42	0.418	0.42	0.43	-	64.39

Source : données de l'auteur

Les numéros d'essais sont utilisés pour retrouver la ligne à enregistrer dans le fichier Excel. En effet, si le numéro d'essai est le 33, toutes les informations de la ligne correspondante sont

récupérées. La lecture du fichier Excel est réalisée à l'aide de la librairie « xlsx ». Après extraction des résultats, ils seront stockés dans la base de données. Ces derniers sont ensuite affichés à l'utilisateur.

Figure 39 : Aperçu des résultats après extraction

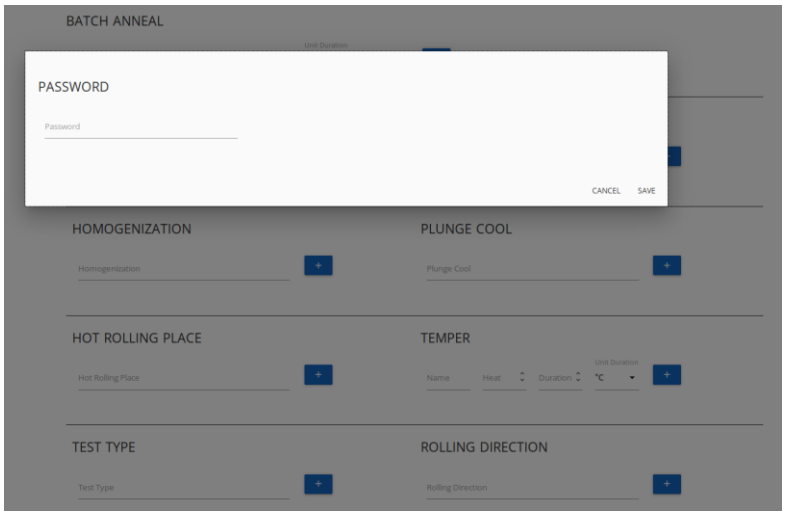


Source : données de l'auteur

5.3.10. Paramètres

Les administrateurs ont accès à la partie « Settings » pour insérer de nouvelles données dans la base de données, plus spécialement pour les différentes listes à choix utilisées dans l'application. Un mot de passe est requis pour accéder à cette page.

Figure 40 : Paramètres



Source : données de l'auteur

Une fois connectée, il suffit de rajouter les données souhaitées à la bonne section.

5.3.11. Filtrage des données

La dernière fonctionnalité majeure offre à l'utilisateur la possibilité de filtrer les données selon ses besoins.

Figure 41 : Filtrage des données

LIST BATCH

Search:

N° Task	N° Lot	Gauge (mm)	Alloy Internal	HO	Fhrt	Cash Line	Pmt (°C)	Tms (s)	Px	Temper	Energy 3 points	Energy crush	Note crush	Rp02	Rm	DC Alpha	F	Z	Crash Factor	Details
123	123	1.2mm	1000	R530	1000°C	ACL	1	2	1000	T6 - 180°C 10h	0	0	0	135.45	244.6	0	0	0	0	
1000	1000	1.2mm	1000	R530	1000°C	Tiger	100	200	100	T62 - 205°C 30min	1.4000000000000001	0	0	133.2	241.6	0	0	0	0	

Showing 1 to 2 of 2 entries

EXPORT

Source : données de l'auteur

Chaque ligne de ce tableau représente un traitement thermique d'un lot. Étant donné qu'un lot peut avoir plusieurs traitements thermiques, la colonne n° lot peut se répéter. À l'aide de la barre de recherche en haut à droite, il peut ensuite entrer des valeurs qui vont filtrer les résultats. Il faut s'imaginer que dans le futur le nombre de lignes présentes dans ce tableau ne fera qu'augmenter. C'est pourquoi la possibilité de diminuer la quantité de résultats était primordiale.

L'utilisateur a ensuite la possibilité d'exporter les données en sélectionnant la ou les ligne(s) correspondantes. L'exportation des données se fait ensuite en CSV. Deux fichiers seront créés, le premier représente toutes les valeurs de la base de données sans les résultats des « crashs-tests ». Le deuxième correspond uniquement aux résultats. Nous avons décidé de séparer ces éléments dans le but de faciliter la compréhension des fichiers générés.

L'export des données est une des fonctionnalités les plus complexes que nous avons dû implémenter. En effet, étant donné la masse de données stockées, il était important de ne pas se perdre dans les relations entre les différentes tables.

5.4. Déploiement

Une fois l'application terminée, il a fallu la déployer au sein de Novelis afin qu'elle puisse être utilisable par les employés de l'entreprise. Un descriptif complet de la façon de déployer une application NodeJS est expliqué dans la documentation technique. Le déploiement nous a posé plusieurs difficultés.

Premièrement, c'est la première fois que nous déployons une application NodeJS durant la totalité de notre cursus scolaire. Le premier challenge était donc de savoir comment configurer un serveur pour y déployer notre site web. Après plusieurs recherches et discussions avec M. Roger Schaer, nous avons finalement réussi à faire le déploiement sur notre ordinateur personnel. Ensuite, nous avons été contraints à adapter la configuration de notre ordinateur sur le serveur de l'entreprise. Quelques changements ont été opérés, pour finalement obtenir, un résultat satisfaisant.

La deuxième grosse difficulté de cette partie du travail était surtout liée à la communication avec les informaticiens de l'entreprise. En effet, le laboratoire de R&D n'est pas le même département que celui des informaticiens. Il a été nécessaire de les appeler à plusieurs reprises pour leur expliquer la situation actuelle du projet, car ils ne sont pas forcément au courant de tout ce qui se passe au R&D. Après plusieurs discussions et échanges de mail, nous avons finalement fixé un rendez-vous pour le déploiement. La grande difficulté était de ne faire aucune erreur sur le serveur de l'entreprise dont nous ne connaissions pas forcément la configuration exacte.

En somme, l'objectif est atteint puisque l'application est accessible uniquement depuis l'intranet de l'entreprise.

6. Améliorations

L'outil développé pour Novelis permet une gestion centralisée des données des différents lots d'aluminium produits à l'usine de Sierre ainsi que des crashs-tests effectués au laboratoire de recherche et développement. Toutefois, de nombreuses améliorations sont encore possibles. En effet, 360 heures de travail ne sont pas suffisantes pour faire une analyse de l'existant, le développement d'une application complète et une analyse des résultats. Les personnes en charge du projet auraient souhaité disposer encore d'autres fonctionnalités, cependant il n'était pas possible de les implémenter par un manque de temps.

Dans l'idéal, l'application devrait permettre aux utilisateurs d'insérer toutes les informations dans la base de données, de filtrer les données (état actuel) mais surtout de pouvoir afficher les résultats dans des graphiques pour comparer les lots entre eux directement sur la plateforme. De plus, la possibilité de joindre des images et des vidéos est également une fonctionnalité envisageable pour le futur. À l'heure actuelle, il n'est pas possible de modifier les données existantes ou de les supprimer, car cela ne faisait pas partie des priorités établies au début du projet.

7. Gestion de projets

7.1. Planification du travail

Une planification du travail a été effectuée au début du projet en réalisant la liste des principales tâches à effectuer ainsi que les délais pour accomplir ces dernières (cf. annexe). Dans la globalité, les échéances ont été majoritairement respectées. Voici la liste des tâches planifiées au début de notre rapport.

Description des tâches	Date d'échéance
Cahier des charges	15.05.2018
Compréhension du domaine et des besoins	25.05.2018
Analyse des données	
Analyse de l'existant au niveau de la récolte de données et de la visualisation	
État de l'art des différentes technologies de base de données	01.06.2018
Analyse des besoins en termes d'interfaces utilisateurs et visualisations	08.06.2018
Choix technologique pour la conception de la solution logicielle	08.06.2018
Conception d'un modèle pour la base de données	15.06.2018
Conception d'une solution logicielle adaptée à l'environnement de travail	13.07.2018
Récolte de nouvelles données, transformation des données existantes	13.07.2018
Réalisation d'un prototype d'interfaces utilisateurs pour la saisie/transformation/visualisation des données	13.07.2018
Correction des différents bugs et amélioration du projet	27.07.2018
Remise du travail final	30.07.2018
Rédaction du rapport final	En continu
Documentation technique	
Prise de rendez-vous avec les personnes de contact pour mon travail de Bachelor	

Tableau 5 : Liste des tâches planifiées

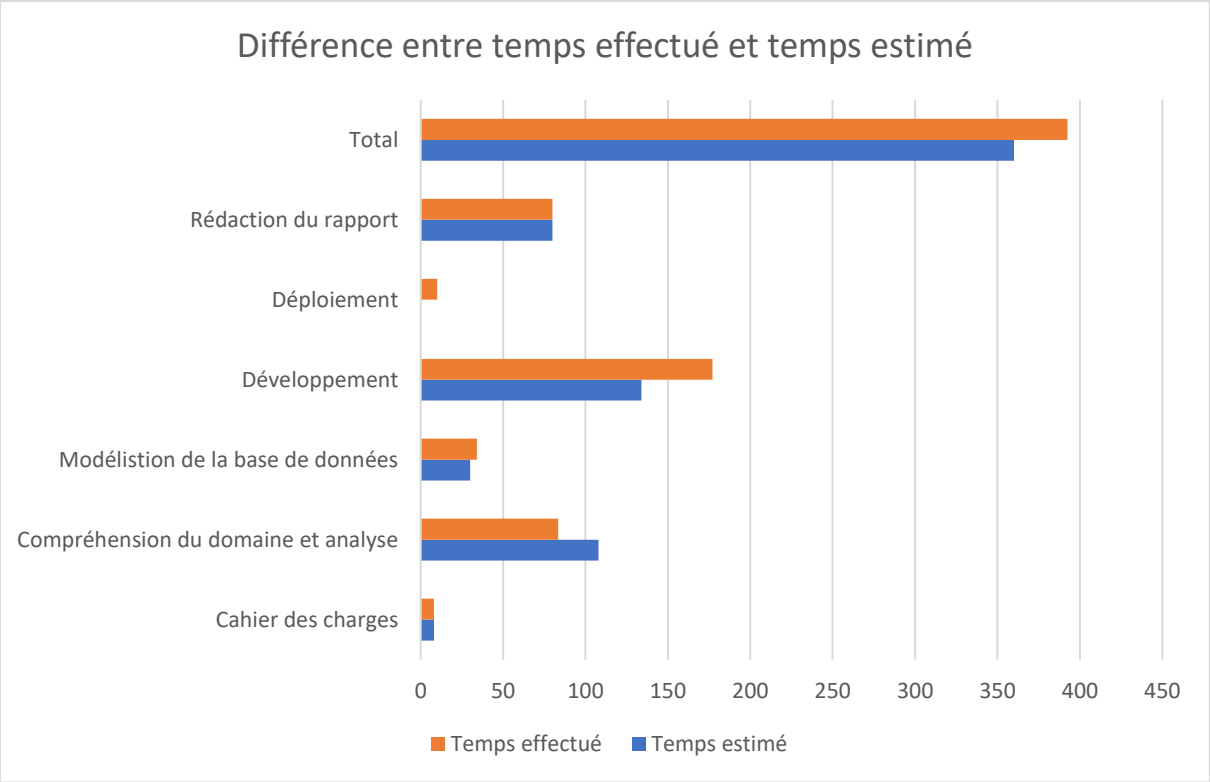
7.2. Gestion du travail

Nous avons travaillé avec une gestion de travail similaire à la méthodologie SCRUM sans respecter tous ses concepts comme le « daily scrum » ou autres. Des séances étaient organisées environ toutes les semaines que ce soit avec Mme Glassey Balet ou avec Mme Fumeaux de Novelis. Lors de ces réunions, nous discutions du travail accompli la semaine précédente et préparions les tâches à accomplir pour la semaine à venir.

7.3. Heures effectuées

Dans cette partie, la différence entre les heures effectuées et les heures planifiées au début de ce travail sont comparées. Comme nous pouvons nous en apercevoir sur le graphique ci-dessous, le nombre d'heures estimé était de 360 heures, cependant nous en avons effectué 392.5. La modélisation de la base de données ainsi que le développement nous ont pris le plus de temps. En effet, il est important de noter que la base de données permet le stockage de toutes les données de production du lot d'aluminium ainsi que des différents crashs-tests réalisés au laboratoire. De plus, le déploiement n'avait pas été prévu dans notre planification initiale.

Figure 42 : Graphique temps effectué / temps estimé



Source : données de l'auteur

8. Bilan

8.1. Connaissances acquises

La réalisation de ce travail constituait notre première immersion dans le monde du travail. En effet, le développement d'un produit de cette envergure représentait pour nous quelque chose de complètement nouveau. Une bonne gestion de projet et de temps a été primordiale pour mener à bien ce travail.

Ce travail m'a également permis d'apprendre à établir une planification au début du projet et surtout à m'y tenir tout au long d'un projet de trois mois. Nous avons appris à travailler avec des personnes provenant d'un métier complètement différent de celui d'informaticien. Il était important d'adapter sa communication et surtout de ne pas utiliser de termes trop techniques afin que tout le monde puisse comprendre le thème abordé.

Dans le cadre scolaire, nous n'avions jamais eu à réaliser des travaux avec une telle recherche. Pour chaque information trouvée, une vérification de la véracité des données était à faire. Cela nous sera très utile pour un futur travail comme le mémoire de Master.

D'un point de vue plus global, ce projet nous a également forcés à prendre des initiatives et surtout à trouver les réponses par nous-mêmes sans attendre qu'elles viennent à nous. C'est typiquement ce qui nous est arrivé pour le déploiement, nous avons dû prendre contact nous-mêmes avec les employés du secteur informatique en leur expliquant le projet et ce que nous attendions d'eux.

8.2. Difficultés rencontrées

Au début du travail, il était difficile d'établir un cahier des charges correct avec une planification précise. Ceci est dû au fait que la portée du projet et les processus métiers nous étaient inconnus.

Nous arrivons donc à notre deuxième grosse difficulté qui était la complexité du sujet et des processus métiers chez Novelis. En effet, n'ayant aucune connaissance préalable dans le domaine de la métallurgie, les premières réunions étaient plutôt perturbantes, car nous avions l'impression d'en ressortir encore plus perdus à cause du niveau de complexité. Un gros travail de compréhension a été réalisé afin d'assimiler la logique du métier avant de pouvoir se lancer dans des analyses ou recherches.

Une fois les analyses terminées et que nous sommes passés à la partie développement du projet, nous nous sommes rendu compte que les responsables du projet ne savaient pas exactement ce qu'ils voulaient en termes de fonctionnalités ou peut-être qu'ils avaient simplement de la peine à l'exprimer. Nous leur avons donc soumis quelques propositions en fonction des connaissances que nous avions sur leur future utilisation. Nous avons également dû modifier notre projet à plusieurs

reprises, car ce qui avait été dit la semaine d'avant n'était plus forcément le cas la semaine suivante selon les employés qui venaient voir l'avancement du projet.

Lors du développement, il a souvent fallu leur rappeler qu'il fallait bien prioriser les fonctionnalités, car il nous était impossible de développer tout ce dont ils avaient besoin. De plus, l'introduction d'un mot de passe dans les paramètres de l'application a uniquement été mentionnée à la fin du projet. Cela a engendré des modifications imprévues dans la base de données pour rajouter cette couche supplémentaire.

9. Conclusion

L'entreprise Novelis utilisait initialement une gestion des données pour les « crashs-tests » avec des fichiers et dossiers. En effet, les lots d'aluminium fabriqués à l'usine de Sierre sont soumis à des tests visant à caractériser un alliage et à améliorer la résistance et la déformation de ces derniers. Le problème majeur de cette méthode de gestion est la redondance des données entre les fichiers ainsi que la recherche d'informations qui peut s'avérer lente.

Ce travail de Bachelor nous a permis de mener une analyse complète sur un sujet spécifique. En effet, après compréhension du domaine, nous avons analysé les besoins de l'entreprise afin de pousser nos recherches le plus précisément possible. Après l'analyse des différents outils présents sur le marché, nous avons développé une application permettant aux employés de Novelis une gestion de leurs données pour le processus de fabrication de l'aluminium ainsi que pour les différents « crashs-tests » effectués au laboratoire de R&D. L'utilisateur a également la possibilité de filtrer les résultats selon certains critères spécifiques pour ensuite exporter le tout dans un fichier CSV.

De plus, ce projet nous a permis de développer la totalité d'une application de manière indépendante. Lors de notre cursus scolaire, nous avions plutôt l'habitude de travailler en groupe en nous séparant les rôles pour la partie Frontend et Backend. Nous avons également été amenés à collaborer avec une entreprise de renommée internationale. Ce travail très concret pourrait grandement leur faciliter le travail quotidien.

Notre solution est actuellement un prototype fonctionnel d'une base de données centralisée pour la gestion des données. Il permet également à l'entreprise d'avoir une solide base sur laquelle pourront s'appuyer les futurs potentiels développeurs afin de rajouter de nouvelles fonctionnalités. Dans l'idéal, l'application devrait permettre aux employés de Novelis d'afficher des graphiques pour les résultats des crashs et de comparer les différents tests entre eux.

10. Références

Apache. (2018). *Welcome! - The Apache HTTP Server Project*. Récupéré sur Apache HTTP Server Official Website: https://httpd.apache.org/

Axure. (s.d.). *Prototypes, Specifications, and Diagrams in One Tool | Axure Software*. Récupéré sur Axure Official Web Site: https://www.axure.com/

Chart.js. (s.d.). *Chart.js | Open source HTML Charts for your website*. Récupéré sur Chart.js Official Web site: https://www.chartjs.org/

Chartist.js. (s.d.). *Chartist - Simple responsive charts*. Récupéré sur Chartist.js Official Web site: https://gionkunz.github.io/chartist-js/

Cheiney, J.-P., Picouet, P., Saglio, J.-M., & Abdessalem, T. (2011, Novembre). *sgbd98_extrait02*. Récupéré sur Dbweb: http://dbweb.enst.fr/teaching/INF225/sgbd98_extrait02.pdf

D3.js - Data-Driven Documents. (2017). *D3.js - Data-Driven Documents*. Récupéré sur D3.js Official Web site: https://d3js.org/

DataTables. (2018). *DataTables | Table plug-in for jQuery*. Récupéré sur DataTables Official Web Site: https://datatables.net/

Django Project. (2018). *The Web framework for pefectionists with deadlines | Django*. Récupéré sur Django Project Official Web Site: https://www.djangoproject.com/

ExpressJS. (2017). *Express - Node.js web application framework*. Récupéré sur ExpressJS Official Web site: http://expressjs.com/

expressjs. (s.d.). *expressjs/multer: Nodej.js middleware for handling `multipart/form-data`*. Récupéré sur Multer Github Page: https://github.com/expressjs/multer

foreverjs. (s.d.). *foreverjs/forever: A simple CLI tool for ensuring that a given script runs continuously (i.e. forever)*. Récupéré sur Github foreverjs Official Website: https://github.com/foreverjs/forever

Gitlab. (s.d.). *Product | GitLab*. Récupéré sur GitLab Official Web Site: https://about.gitlab.com/product/

Google Charts. (2017, Février 23). *Charts | Google Developers*. Récupéré sur Google Charts | Developers: https://developers.google.com/chart/

jQuery. (2018). *jQuery*. Récupéré sur jQuery Official Web Site: https://jquery.com/

jQuery Validation Plugin. (s.d.). *jQuery Validation Plugin | Form validation with jQuery*. Récupéré sur jQuery Validation Official Web Site: https://jqueryvalidation.org/

Kelektiv. (s.d.). *kelektiv/node.bcrypt.js: bcrypt for NodeJS*. Récupéré sur Bcrypt Github Web Site: https://github.com/kelektiv/node.bcrypt.js

Laravel. (s.d.). *Introduction - Laravel - The PHP Framework For Web Artisans*. Récupéré sur Laravel Official Web site: https://laravel.com/docs/5.6

MariaDB Foundation. (2018). *About MariaDB - MariaDB.org*. Récupéré sur https://mariadb.org: https://mariadb.org/about/

Materialize CSS. (s.d.). *Documentation - Materialize*. Récupéré sur Materialize Official Web site: https://materializecss.com/

Microsoft - Power BI. (2018). *Qu'est-ce que Power BI | Microsoft Power BI*. Récupéré sur https://powerbi.microsoft.com: https://powerbi.microsoft.com/fr-fr/what-is-power-bi/

Microsoft - Power BI. (s.d.). *Types de visualisations dans Power BI - Power BI | Microsoft Docs*. Récupéré sur https://docs.microsoft.com: https://docs.microsoft.com/fr-fr/power-bi/power-bi-visualization-types-for-reports-and-q-and-a

Microsoft. (2018). *SQL Server 2017 sur Windows et Linux | Microsoft*. Récupéré sur https://www.microsoft.com: https://www.microsoft.com/fr-fr/sql-server/sql-server-2017

Node.js. (s.d.). *File System | Node.js v10.7.0 Documentation*. Récupéré sur Nodejs Official Web site: https://nodejs.org/api/fs.html

NodeJS. (s.d.). *Node.js*. Récupéré sur NodeJS Official Web Site: https://nodejs.org/fr/

Novelis. (2017, Mars 3). *Novelis-Corporate-Fact-Sheet-November-3-2017.pdf*. Récupéré sur novelis: http://2gjjon1sdeu33dnmvp1qwsdx.wpengine.netdna-cdn.com/wp-content/uploads/2017/10/Novelis-Corporate-Fact-Sheet-November-3-2017.pdf

Novelis Tosi, O. (2018, Avril 23). novelis_convention. Sierre, Valais, Suisse.

npmjs. (s.d.). *npm*. Récupéré sur npm Official Web Site: https://www.npmjs.com/

PostgreSQL. (2018). *PostgreSQL : About*. Récupéré sur https://www.postgresql.org: https://www.postgresql.org/about/

Pugjs. (s.d.). *Getting started - Pug*. Récupéré sur Pug Official Web Site: https://pugjs.org/api/getting-started.html

Qlik. (2018). *Une plateforme complète de Business Intelligence (BI)*. Récupéré sur www.qlik.com: https://www.qlik.com/fr-fr/products

Ruby on Rails. (s.d.). *Ruby on Rails | A web-application framework that includes everythng needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern*. Récupéré sur https://rubyonrails.org/: https://rubyonrails.org/

ryu1kn. (s.d.). *ryu1kn/csv-writer: Convert objects/arrays into a CSV string or write them into a CSV file*. Récupéré sur csv-writer Github page: https://github.com/ryu1kn/csv-writer

Sequelizejs. (s.d.). *Manual | Sequelize | The node.js ORM for PostgreSQL, MySQL, SQLite and MSSQL*. Récupéré sur Sequelize Official Web site: http://docs.sequelizejs.com/

Sharma, V. (2017, Octobre 29). *Setup Basic Server with Express Framework - Vanila Blog - Web & Mobile Development, UI/UX*. Récupéré sur Blog - Vanila: https://blog.vanila.io/setup-basic-server-with-express-framework-37b2ec749a6d

SheetJS. (s.d.). *SheetJS/js-xlsx: SheetJS Community Edition -- Spreadsheet Parser and Writer*. Récupéré sur js-xlsx Github page: https://github.com/SheetJS/js-xlsx

Singh, S. (2016, Juillet 27). *Web Framework Wars : Django vs Ruby on Rails - The Geeks Watch*. Récupéré sur The Geeks Watch - Blogspot: https://thegeekswatch.blogspot.com/2016/08/framework-wars-django-vs-ruby-on-rails.html

Socket.io. (s.d.). *Socket.iO*. Récupéré sur Socket.IO Official Web site: https://socket.io/

Tableau Software. (2017). *Logiciel de visualisation de données | Tableau Software*. Récupéré sur www.tableau.com: https://www.tableau.com/fr-fr/trial/data-visualization

Tableau Software. (2017). *Mission | Tableau Software*. Récupéré sur www.tableau.com: https://www.tableau.com/fr-fr/about/mission#power-empower

Visual Studio Code. (2018). *Visual Studio Code - Code Editing. Redefined*. Récupéré sur Visual Studio Code Official Web Site: https://code.visualstudio.com/

VSlink. (s.d.). *Novelis Switzerland*. Récupéré sur VSlink: https://www.vslink.ch/emploi-valais/novelis-switzerland.html

Annexe I : Cahier des charges

Figure 43 : Cahier des charges p.1

1. Introduction

Ce travail de Bachelor a pour but de réaliser un prototype fonctionnel de gestion des données liées aux "crash-tests" pour la société Novelis. Novelis est la plus grande entreprise de produits laminés en aluminium et l'un des plus importants recycleurs d'aluminium au monde. Son usine de Sierre est le leader de la production de tôles de carrosserie en alliage d'aluminium.

Les propriétés des alliages d'aluminium pouvant varier en fonction de leur composition et procédé de fabrication, il est important de les soumettre à des tests de qualification en laboratoire. Actuellement, toutes les données sont stockées dans différents fichiers Excel remplis au fur et à mesure par les différents employés du laboratoire de Ra&D du Technopôle. Différents graphiques sont générés à partir de ces fichiers permettant de visualiser les données « crash-tests » de façon plus claire.

2. Description de l'outil

Le but du prototype est de proposer une base de données centralisée permettant de visualiser ces données, de faire des recherches selon certains critères ainsi que d'ajouter de nouvelles informations depuis une interface utilisateur. Les employés n'auraient plus à manipuler deux voire trois fichiers Excel pour un même « crash-test », toutes les informations seraient directement regroupées à un seul endroit. Cela leur facilitera la tâche, mais permettra également de baisser le risque d'erreurs liées à la recopie des données d'un fichier à l'autre en automatisant certains processus.

3. Étapes de réalisations envisagées

- Compréhension du domaine et des besoins, par discussion avec les personnes intéressées
- Analyse de l'existant au niveau de la récolte de données et de la visualisation
- Analyse des données
- Analyse des besoins en termes d'interfaces utilisateurs et visualisations
- Choix technologique pour la conception de la solution logicielle
- Conception d'une solution logicielle qui convienne à l'environnement de travail
- Modélisation des données
- Récolte de nouvelles données, transformation des données existantes
- Réalisation d'un prototype d'interfaces utilisateurs pour la saisie/transformation/visualisation des données

1

Source : données de l'auteur

Figure 44 : Cahier des charges p.2

4. Planification

N°	Description des tâches	Commentaires	Date d'échéance
1	Cahier des charges		15.05.2018
2.1	Compréhension du domaine et des besoins	Tâches en parallèle	25.05.2018
2.2	Analyse des données		
2.3	Analyse de l'existant au niveau de la récolte de données et de la visualisation		
3	État de l'art des différentes technologies de base de données		01.06.2018
4	Analyse des besoins en termes d'interfaces utilisateurs et visualisations		08.06.2018
5	Choix technologique pour la conception de la solution logicielle	Analyse de l'environnement de travail de Novelis afin de choisir une technologie convenable à leur environnement de travail	08.06.2018
6	Conception d'un modèle pour la base de données		15.06.2018
7	Conception d'une solution logicielle adaptée à l'environnement de travail		13.07.2018
8	Récolte de nouvelles données, transformation des données existantes		13.07.2018
9	Réalisation d'un prototype d'interfaces utilisateurs pour la saisie/transformation/visualisation des données		13.07.2018
10	Correction des différents bugs et amélioration du projet		27.07.2018
11	Remise du travail final		30.07.2018

2

Source : données de l'auteur

Figure 45 : Cahier des charges p.3

11.1	Rédaction du rapport final	Tâches à réaliser tout au long de mon travail de Bachelor, durant chacune des étapes mentionnées ci-dessus.	En continu
11.2	Documentation technique		
11.3	Prise de rendez-vous avec les personnes de contact pour mon travail de Bachelor		

5. Répartition des heures

Semaine	Du / au	N° de la tâche	Total
18	30.04.2018 - 04.05.2018	0 (prise de rendez-vous)	3
19	07.05.2018 - 11.05.2018	Tâche 1	8
20	14.05.2018 - 18.05.2018	Tâche 2.1 / Tâche 2.2 / Tâche 2.3	27
21	21.05.2018 - 25.05.2018	Tâche 2.1 / Tâche 2.2 / Tâche 2.3	27
22	28.05.2018 - 01.06.2018	Tâche 3	27
23	04.06.2018 - 08.06.2018	Tâche 4 / Tâche 5	27
24	11.06.2018 - 15.06.2018	Tâche 6	27
25	18.06.2018 - 22.06.2018	Tâche 7 / Tâche 8 / Tâche 9	27
26	25.06.2018 - 29.06.2018	Tâche 7 / Tâche 8 / Tâche 9	27
27	02.07.2018 - 06.07.2018	Tâche 7 / Tâche 8 / Tâche 9	40
28	09.07.2018 - 13.07.2018	Tâche 7 / Tâche 8 / Tâche 9	40
29	16.07.2018 - 20.07.2018	Tâche 10	40
30	23.07.2018 - 27.07.2018	Tâche 10	40
31	30.07.2018	Tâche 11	0
Total			360

3

Source : données de l'auteur

11. Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Nicole Glassey Balet
- Fabian Cretton
- Maude Fumeaux

Sierre, le 30 juillet 2018

Aleksandar Lazic