# Rapport d'activité

*Année : 2020 - 2021*


***Raphael Trabelsi***

*L3 MIAGE Sorbonne*

*Maître d'apprentissage :* ***Nicolas PERRIER***

*Tuteur enseignant :* ***Nicolas HERBAUT***

# Table des matières

# Remerciements

Raphael Trabelsi
42 rue Fessart
75019 Paris
Adresse électronique : raphael.trabelsi18@gmail.com

**Isotoner**
41-43 rue de Villiers, 92200 Neuilly-sur-Seine

A Paris le 14 juin 2021

Chers collègues,

Je tenais à vous remercier pour cette année que j'ai pu passer chez Isotoner.

Cette expérience a été très enrichissante tant personnellement, qu'en termes d'apprentissage. Elle m'a permis de développer des compétences dans des technologies que je n'avais jamais approchées jusqu'alors.

J'ai eu la chance d'avoir Timothée GIROUD et Nicolas PERRIER à mes côtés durant toute cette année. Merci d'avoir pris du temps pour moi et de m'avoir si bien intégré chez Isotoner. Les compétences et connaissances que vous m'avez apportées m'ont permis de consolider celles que j'avais acquises lors de mes études. Elles me seront très utiles pour ma future carrière professionnelle.

Malgré la crise sanitaire et les nombreuses contraintes que cela a engendré, j'ai eu l'occasion de rencontrer des personnes bienveillantes et investis. Je suis très heureux d'avoir pu travailler avec vous.

Je vous remercie également de m'avoir consacré du temps afin de m'expliquer le fonctionnement de l'entreprise et de m'avoir fait découvrir de nombreux métiers qui la compose. Cela m'a permis d'avoir une vision globale des métiers existants, et a conforté mon choix de projet professionnel.

Je vous prie d'agréer, Madame, Monsieur, l'expression de mes respectueuses salutations.

Raphael Trabelsi

# Introduction

J'ai choisi d'intégrer la licence 3 MIAGE proposé par l'université de la Sorbonne afin de pouvoir développer certaines compétences que j'ai eu l'occasion d'appréhender durant mon BTS Service Informatique aux Organisations. Celles qui me tenaient le plus à cœur ont été les matières autour de la programmation et de la gestion de projet.

Ainsi, après avoir postulé dans de nombreuses entreprises, j'ai finalement obtenu un entretien chez Isotoner. Ce dernier s'est bien déroulé, et on m'a indiqué que ce qui m'a différencié d'autres candidats potentiels, était un projet que j'ai eu l'occasion de réaliser durant le premier confinement. Il s'agissait d'un projet d'équipe entre amis, et dans lequel j'ai été nommé chef de projet afin de diriger les principales opérations concernant l'avancement de celui-ci, tout en contribuant au code. Ce projet a démontré ma motivation à travailler dans ce domaine et à savoir prendre des décisions pour les projets lorsque cela est nécessaire.

L'offre d'alternance concernait des développements dans différents langages, dont certains que je connaissais déjà (JavaScript, Java, SQL). Cependant, d'autres m'étais inconnus tels que le Swift qui concerne les développements sous iOS, et ou encore l'atelier de génie logiciel WinDev qui propose de nombreux outils afin de développer des applications. Ceci n'a pas été un problème et on m'a bien indiqué que j'aurais du temps afin de m'entraîner sur ces différentes technologies.

Plusieurs projets m'ont ainsi été confiés afin de me faire découvrir au fur et à mesure le fonctionnement de l'entreprise, les outils utilisés en interne et l'environnement de travail.

Je vais donc vous présenter durant ce rapport d'activité, l'élaboration d'un projet que j'ai eu à réaliser. Il s'agit d'un outil destiné aux commerciaux, permettant de visualiser l'état des stocks de l'entreprise.

Pour ce faire, je présenterai dans un premier temps Isotoner, ainsi que le service que j'ai intégré. Puis, je présenterai plus en détail les missions qui m'ont été confiées, et les étapes de réalisation de ma deuxième mission. Enfin, je terminerai sur les difficultés que j'ai pu rencontrer lors de l'élaboration de ma solution.

# Présentation de l'organisation

Isotoner est une entreprise fondée en 1910 et possède son siège social à Saint-Martin-Valmeroux (Auvergne). Elle est spécialisée dans l'industrie du textile, et est leader du marché français du chausson, gant et du parapluie. Constamment en recherche d'innovations, elle développe des technologies brevetées afin d'améliorer le confort de ses produits. Aujourd'hui, l'entreprise possède plus de 3 000 références en France et est présente dans plus de 6 500 points de ventes. (Isotoner - Wikipédia, 2020) (La maison Isotoner, s.d.)

## Présentation du service informatique

En France, Isotoner est divisé en 2 pôles qui assurent chacun des rôles complémentaires.

Le premier est à Saint-Martin-Valmeroux. Il s'agit du siège social de l'entreprise, mais également de son centre de distribution afin de couvrir l'ensemble de ses distributions en Europe. (La maison Isotoner, s.d.). C'est à cet endroit que se trouvent les traitements des commandes et la production. C'est également là-bas que se trouve l'essentiel de l'équipe de développement informatique.

*Siège de l'entreprise à Saint-Martin-Valmeroux (Cantal passion, s.d.)*

Le second pôle, celui dans lequel j'ai travaillé, se trouve à Levallois-Perret. Ici, l'entreprise s'occupe principalement de l'ensemble des activités administratives, commercial et marketing. Certains métiers liés à l'informatique s'y trouvent également, mais pas d'équipe de développement.

Vous pouvez consulter l'organigramme de l'entreprise, afin de comprendre où je me situe par rapport à ma hiérarchie. (Annexe 1 – Organigramme de l'entreprise)

J'ai donc été amené à travailler à distance avec l'équipe de Saint-Martin-Valmeroux. Nos échanges se faisaient essentiellement via l'application Telegram, et de nombreuses réunions vidéo ont été organisées afin de faire des points sur l'avancée du projet, ainsi que sur les éventuelles difficultés.

Le point faible qui a été relevé par mon maître d'apprentissage fut l'absence de développeurs web dans l'équipe jusqu'à maintenant.

Cette équipe de développement est composée de 9 personnes, et j'ai pu travailler avec 2 d'entre eux pour le projet qui m'a été confié.

- **Nicolas PERRIER** (Responsable d'exploitation informatique) : il a dirigé l'organisation du projet, tout en me laissant une certaine liberté dans la planification des tâches et sur certains choix importants du projet.
- **Rémi SERVEL** (Service informatique) : il s'est occupé de tout ce qui a touché au Web Service, notamment de la récupération et de la mise à jour des données.

# Présentation des missions réalisées

Au début de l'année, on m'a tout d'abord expliqué que mon objectif principal serait d'arriver au bout du développement d'une application iOS destiné aux commerciaux. Cette dernière a déjà été en partie réalisée durant plusieurs mois par des personnes qui ont quitté l'entreprise. Cependant, des fonctionnalités importantes manquent encore et ce sont celles-ci que je devrai développer en priorité.

Afin de travailler dans les meilleures conditions, 2 autres missions m'ont d'abord été confiées dans le but de m'aider à mieux m'intégrer dans l'organisation et de comprendre au fur et à mesure comment sont traitées et stockées les données chez Isotoner.

La première a d'abord consisté à développer un scanner de code bar qui enverrait les données scannées vers une base de données. Il m'a été donné un délai de 3 jours pour arriver au bout du développement afin qu'une équipe puisse réaliser rapidement un inventaire des stocks dans un entrepôt, à la suite de la crise sanitaire (ces données n'ayant pas été mises à jour depuis plusieurs mois, il était impératif d'avoir les délais les plus courts possibles). Cela m'a permis d'appréhender les gestions des différentes références par l'entreprise, et m'a familiarisé avec des termes que je rencontrerai tout au long de l'année.

Le deuxième projet qui m'a été confié s'est en revanche étendu sur plusieurs mois. Le but était de recréer un outil qui était devenu obsolète, suite aux nombreux changements qui ont été faits dans l'organisation des données en base. Ce dernier a pour objectif de lister les annonces de disponibilité pour chaque référence et celles qui seront réapprovisionnées ou non. C'est ce projet que j'ai choisi de détailler durant ce rapport.

## Contexte

Isotoner utilise en interne un site créé il y a plusieurs années afin de lister les différentes annonces de disponibilité. Cependant, celui-ci commence à être obsolète dû aux nombreuses évolutions qui ont eu lieu ces dernières années dans l'organisation des données pour chaque référence, ainsi que sur l'organisation des saisons de vente.

Cet outil récupérait l'ensemble des données depuis un fichier csv (type Excel) régulièrement mis à jour qui contenant l'ensemble des disponibilités produit. Ensuite, le site importait ces données dans une base afin de faciliter leur recherche. L'utilisateur pouvait ensuite se rendre sur l'outil afin d'afficher plus précisément les ruptures de stocks pour certaines références, certaines gammes etc…

Aujourd'hui, la base de données est régulièrement mise à jour directement sans passer par un fichier csv. De plus, certains champs présents auparavant ne sont plus d'actualité, ou bien ont été modifiés.

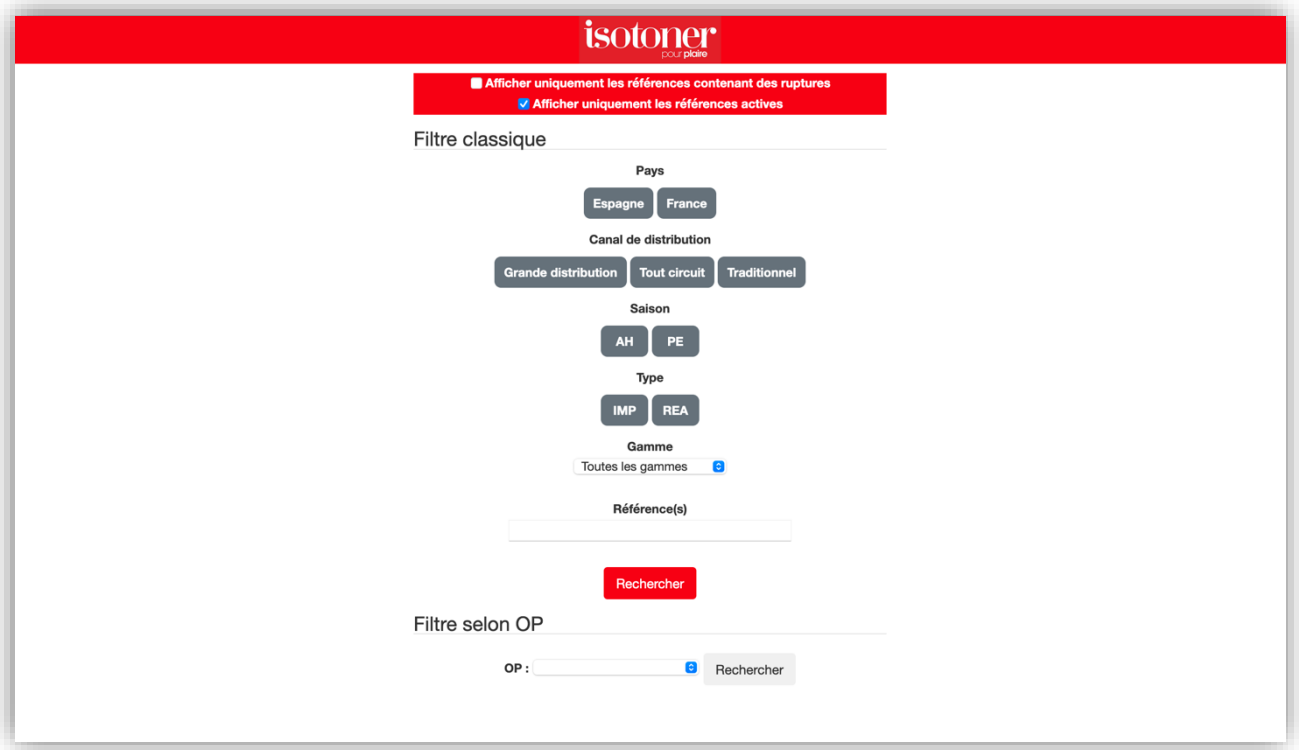
Puisque ces changements sont de grandes ampleurs, il a été décidé par l'entreprise de repartir sur de bonnes bases et de proposer un nouvel outil plus en adéquation avec les évolutions présentes et futures de ces données.

L'enjeu principal est donc de permettre aux commerciaux de pouvoir consulter facilement le statut des stocks, tout en proposant une interface agréable et intuitive.

Ainsi, l'ancien site me servira uniquement de repaire afin de ne pas désorienter les utilisateurs qui passeront sur la nouvelle version de l'outil.

## Aperçu de l'existant

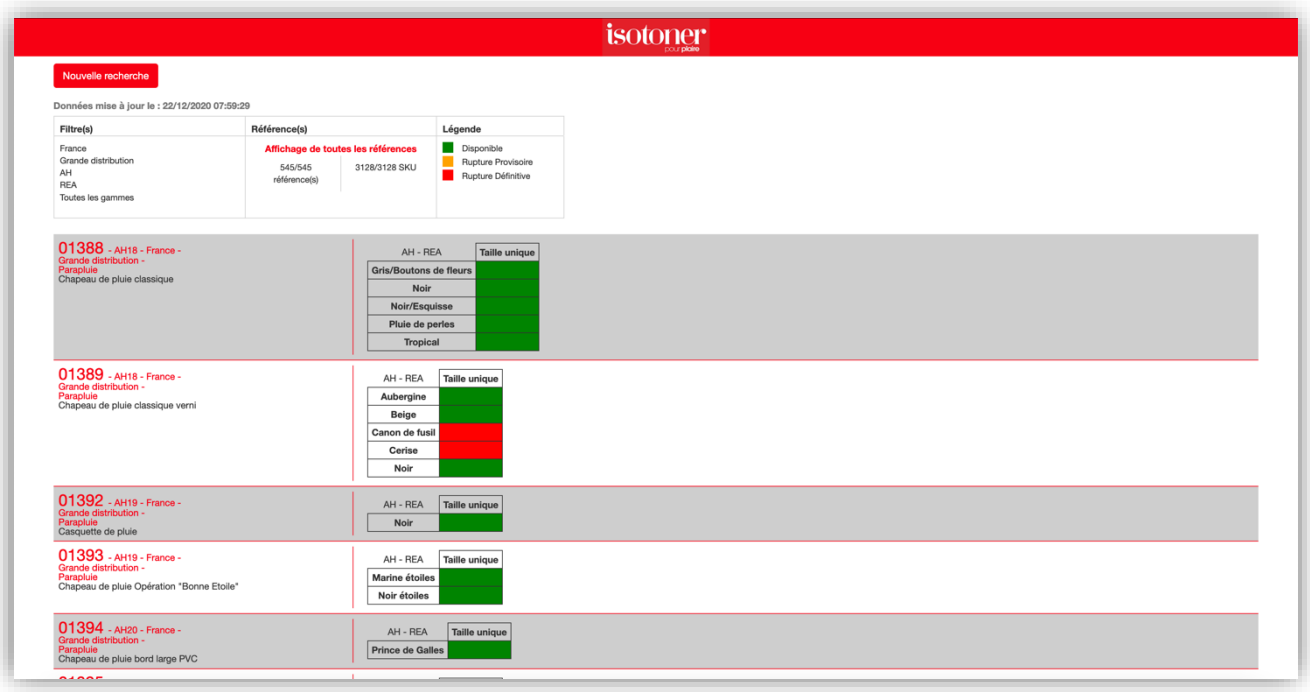
Le site créé il y a quelques années, a été réalisé en utilisant du php côté serveur. Celui-ci possède certaines fonctions de base qui devront être présentes sur le nouveau site.




*Ancienne page de sélection des filtres*

Sur cette page, les filtres par saisons devront être mis à jour en tenant compte des nouveaux besoins de l'entreprise. En effet, chaque produit contient maintenant la saison qui lui est associé et son année de lancement (AH20 ou PE21 par exemple). Ces saisons devront être au nombre de 4 sur la page d'accueil et seront mises à jour tous les 6 mois par un administrateur (à chaque changement de saison).

Les filtre par types « IMP » et « REA » et le filtre selon « OP » devront être retirés, car ils ne plus utilisés.




*Ancienne page de résultat*

Une fois la recherche effectuée, la page affiche un récapitulatif des filtres sélectionnés, et les résultats s'affichent les uns au-dessus des autres.

## Objectif de l'outil

L'outil développé a plusieurs objectifs. Tout d'abord, il doit pouvoir être utilisé et utilisable durant un grand nombre d'années en facilitant la mise à jour du code existant en cas de modifications importantes dans le futur. Il doit également refléter la nouvelle méthode de stockage des données en adaptant les filtres de recherches et en prenant en compte les nouveaux besoins des utilisateurs.

Après avoir fait plusieurs réunions et points quant aux attentes par rapport à l'outil, les principaux besoins exprimés ont été les suivants :

- Création d'une interface web
- Rechercher l'état des stocks des produits via des filtres
- Mise à disposition de nouveaux filtres de recherches (non-présents sur l'ancienne version)
- Affichage des résultats sur une page présentant un résumé des différents filtres appliqués
- Les résultats doivent présenter l'état du stock pour chaque variante des différents produit (sku)
- Retenir certains choix de filtres faits par les utilisateurs afin qu'ils soient automatiquement saisis les fois suivantes
- Possibilité de traduire le site en plusieurs langues
- Mise en place d'une page d'administration et permettre la modification des traductions et des saisons en cours

L'objectif était également de pouvoir faire évoluer le site de la façon la plus simple possible et sans repartir de zéro en cas de changements de l'architecture de stockage des données au sein de l'entreprise. Ce dernier devait donc être correctement documenté et commenté afin de pouvoir être modifié ultérieurement par des personnes n'ayant pas participé à son développement initial.

## Étude préalable

Avant de commencer tout travail de réflexion autour du projet, j'ai tout d'abord eu une réunion avec le DSI et les membres du SI qui participeront au projet avec moi. Celle-ci avait pour but de me présenter la façon dont sont gérées et stockées les différentes références que la marque propose. À la suite de cela, j'ai pris connaissance de l'existant et j'ai analysé son fonctionnement afin d'estimer la charge de travail et de réaliser un planning de réalisation.

Dans l'optique de repartir sur de bonnes bases, le site devait être plus réactif que son prédécesseur. J'avais donc comme limitation d'éviter au maximum d'utiliser du php, et au contraire, maximiser l'utilisation du JavaScript. Après avoir étudié plusieurs possibilités, mon choix s'est porté sur NodeJs. Son principal avantage était qu'il s'agissait d'un environnement de développement coté serveur basé sur du Javascript, ce qui signifie d'énormes gains de performances et de réduction des temps de réponse. Il s'agissait cependant qu'une technologie que je ne connaissais pas encore. Vis-à-vis de ses nombreux avantages, il a donc été décidé que le nouveau site serait développé grâce à NodeJs. Cela a bien sûr impliqué de prendre en compte le temps de formation dans mon planning prévisionnel (cf. Organisation).

Une fois ce choix fait, il restait encore une partie importante à laquelle je devais réfléchir, à savoir : comment accéder le plus efficacement possible aux données depuis le site ? Il n'était pas encore clair à ce moment si je devais créer une base de données plus efficace pour le site ou si je devrais faire autrement. J'ai donc été amené à rédiger un document afin de déterminer avec quel type de base de données il était plus judicieux de travailler (Annexe 2 - Choix de base de données SQL ou NoSQL)

Après avoir analysé ces 2 possibilités, j'ai réalisé un mcd (cf. Model de données) qui m'a permis de comprendre plus en détail à quoi correspondait exactement chaque champ pour les différents produits. Bien que ce MCD soit simpliste, cela m'a obligé à poser les bonnes questions afin de mieux cerner le contexte dans lequel j'allais travailler.

À partir d'ici, mon maître d'apprentissage m'a indiqué une nouvelle piste à étudier pour la récupération des données. Il s'agissait de l'utilisation d'un Web Service. Ne sachant pas ce dont il s'agissait et comment cela fonctionnait, j'ai mené quelques recherches. Il s'est avéré que le Web Service servirait de lien entre la base de données déjà existante et le nouvel outil que j'allais développer, ce qui était effectivement un meilleur choix que de faire migrer les données déjà existantes vers une nouvelle base de données.

Avec toutes ces informations et tous ces choix, j'ai donc commencé à réaliser un premier cahier des charges afin de cadrer le projet. Il me servirait également de support visuel dans le cadre de présentations ou de réunions. Ce dernier allait donc s'étoffer au fur et à mesure des différents retours et des nouveaux besoins qui ont pu survenir plus loin durant le projet.

Une fois les éléments essentiels du cahier des charges rédigés, celui-ci a été transmis à une autre personne qui s'est chargé de réaliser le Web Service. J'ai pendant ce temps réalisé les maquettes qui

allaient me permettre de développer le site en ayant une idée claire du design de l'outil. Elles m'ont également permis d'étoffer mon cahier des charges pour de futures présentations.

Une fois le Web Service et les maquettes terminées, le développement du site à proprement parler pouvait enfin débuter.

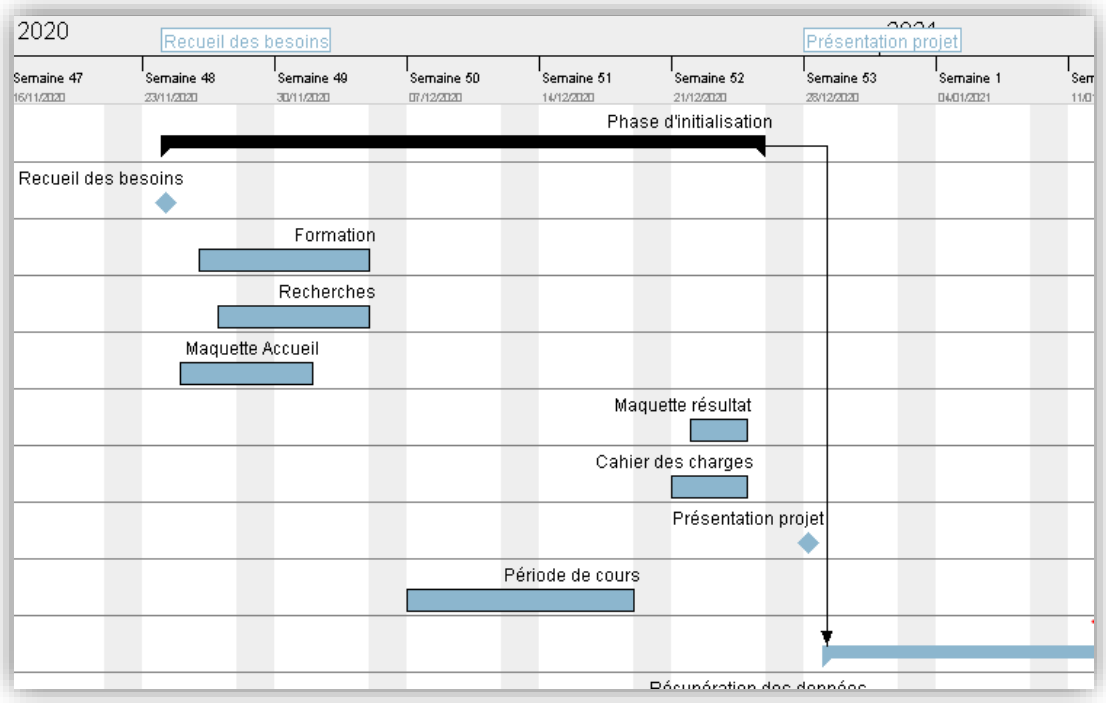
## Organisation

À chaque étape importante du projet, nous avons organisé des réunions pour récapituler l'avancement des tâches avec le DSI et mon maître d'apprentissage. Elles nous ont permis de constater bon déroulement du projet depuis la précédente réunion et d'anticiper les prochaines tâches à réaliser. Une fois le site lancé, nous récapitulions également les retours des utilisateurs, et nous planifiions éventuellement des changements en fonction de ces derniers.

Mes journées de travail se sont tout d'abord déroulées en présentiel en début d'année. Puis, lors du deuxième confinement, nous sommes passés à 3 jours de télétravail par semaine. J'ai donc dû m'organiser pour avoir constamment mon Mac avec moi, ainsi que mes notes prises en réunions.

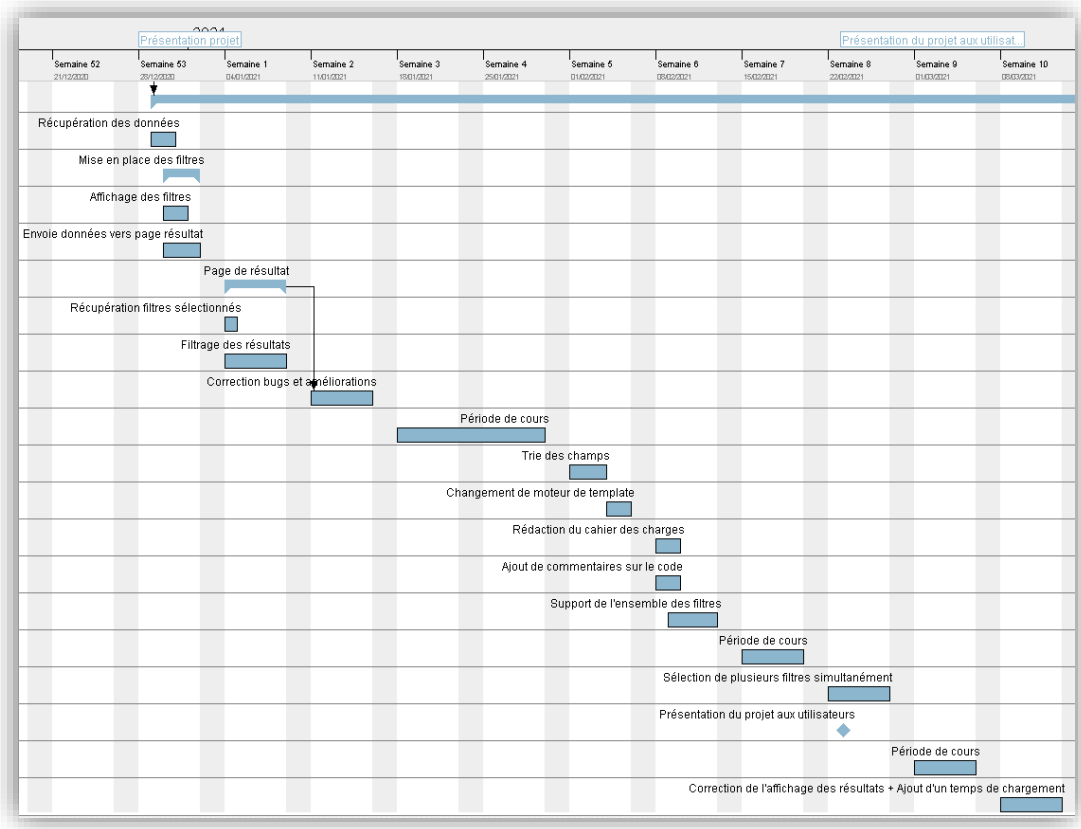
Chaque journée de travail devait se terminer sur l'envoi d'un bilan des tâches réalisées dans la journée à mon maître d'apprentissage afin d'améliorer le suivi du projet, et de détecter les moindres soucis ou écarts par rapport à celui-ci.

J'ai également réalisé un planning des tâches grâce au logiciel Gantt Project. Ce dernier a été réalisé au fur et à mesure des demandes. Il a également été ajusté à chaque réunion afin d'avoir une vision plus précise des tâches restantes, et de mieux pouvoir estimer le délai de chaque étape.

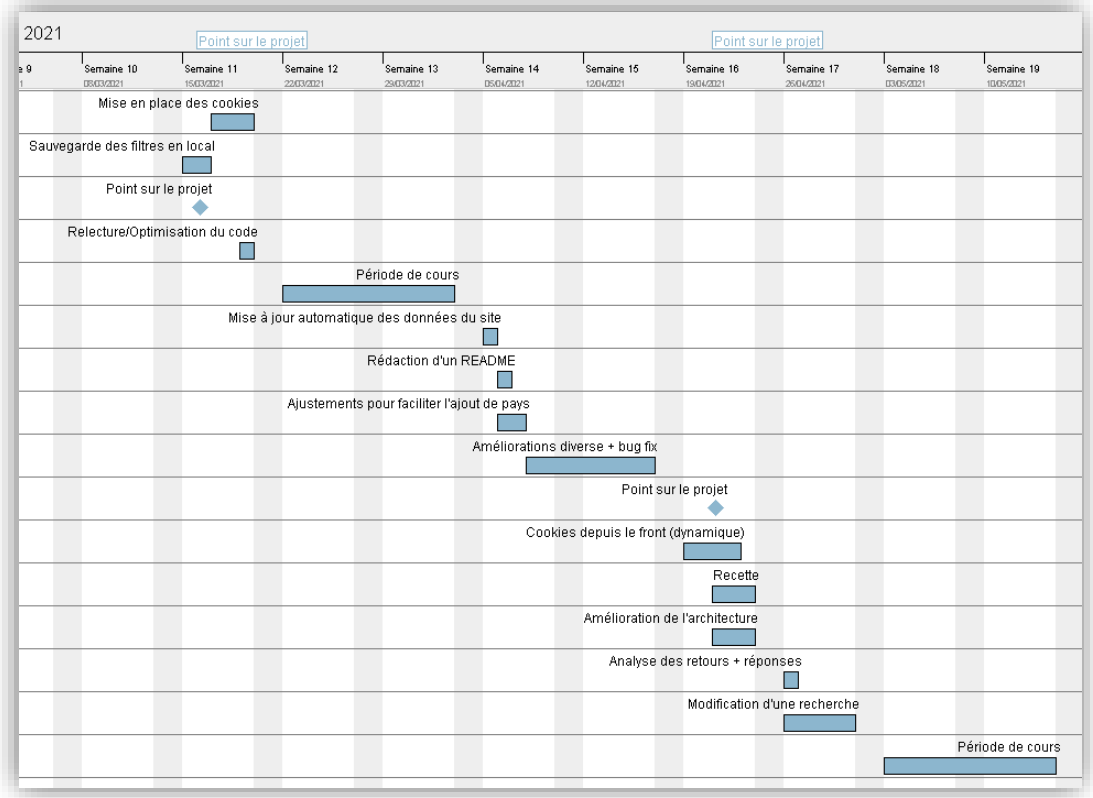



*Initialisation du projet*

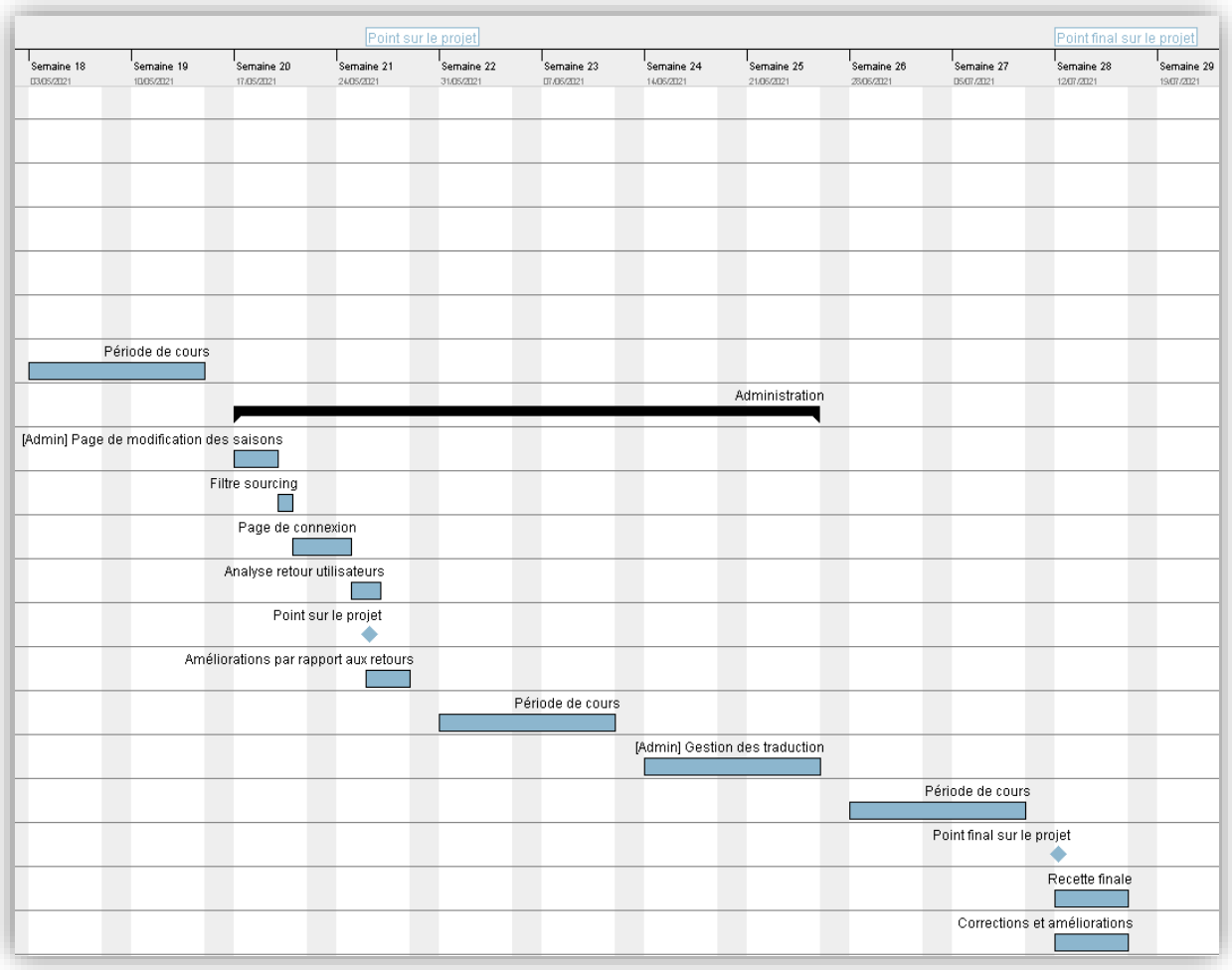
La première étape a été la phase d'initialisation. Celle-ci m'a permis de comprendre plus clairement quels seraient les enjeux et les attentes par rapport à ce projet.




*Début de la phase de conception*




*Milieu de la phase de conception*




*Fin de la phase de conception*

La phase de conception s'est principalement divisée en 3 étapes :

- La première a tout d'abord consisté à afficher correctement les filtres sur la page d'accueil, d'envoyer les filtres sélectionnés sur la page de résultat, et enfin de faire en sorte que les résultats s'affichent correctement.

- La deuxième étape a principalement consisté à améliorer le projet dans son ensemble, tant au niveau backend qu'au niveau du confort d'utilisation.

- Enfin, la dernière étape a été utile afin de faire des modifications en fonction des retours utilisateurs, et de créer la page d'administration.

# Réalisation

## Choix techniques

### Environnement de travail

L'un des projets sur lesquels je devais être amené à travailler consistait à continuer le développement d'une application iOS pour aider les commerciaux à passer des commandes de produits. Ainsi, malgré ma méconnaissance dans l'univers d'Apple, j'ai dû trouver mes marques avec le système d'exploitation du **Mac** qui m'a été confié.
En ce qui concerne l'IDE, pour l'application iOS je devrai travailler avec XCode, l'IDE d'Apple (obligatoire pour développer une application un environnement iOS). Pour le projet que j'ai choisi de présenter en revanche, je n'avais aucune contrainte particulière. J'ai donc choisi d'utiliser celui de **JetBrains** avec lequel j'étais déjà habitué à travailler lors de mes précédents projets web.

**Poste de travail**: MacBook Pro
**IDE**: PhpStorm

### Langages

Comme expliqué ci-dessus (cf. Étude préalable), le choix de l'utilisation de **NodeJs** a été le fruit d'une longue réflexion. Le but était d'obtenir un site fluide et de pouvoir traiter et trier un grand volume de données en un temps minimal. Php n'était donc pas optimal, car trop long pour exécuter certaines tâches. NodeJs en revanche, donne la possibilité de développer le backend du site en Javascript (qui est un langage beaucoup plus rapide en exécution). Pour l'affichage des pages, j'ai choisi d'utiliser **EJS** qui est très utilisé avec NodeJs pour les nombreuses possibilités qu'il offre. Son rôle est de générer des pages HTML, en séparant distinctement la partie frontend de la partie backend.

**Côté serveur**: Node.js
**Côté client**: HTML/CSS, Javascript
**Moteur de template** : EJS

### Outils

Afin de pouvoir développer plus efficacement, l'utilisation de certains composants externes est essentielle. En me renseignant sur NodeJs, j'ai pu noter qu'un framework en particulier, était souvent utilisée, il s'agit de **Express**. Ce dernier est utile pour simplifier de nombreuses fonctionnalités de NodeJs. La librairie **Bootstrap** a été utile pour simplifier le style des différents éléments, tels que les boutons ou les tableaux, et a également simplifié la « responsivité » (adaptation à plusieurs tailles d'écran). Lors de l'installation de NodeJs, **npm** est le repository qui est installé par défaut (gestionnaire de package). Celui-ci permet d'intégrer très facilement des éléments développés par d'autres personnes à son projet, ce qui peut s'avérer être un énorme gain de temps.

**Framework**: Express
**Library**: Bootstrap
**Repository**: Npm
**Gestion des versions** : Git, GitHub

### Accès aux données

**SGBD**: MySQL
**Web Service** : Architecture REST

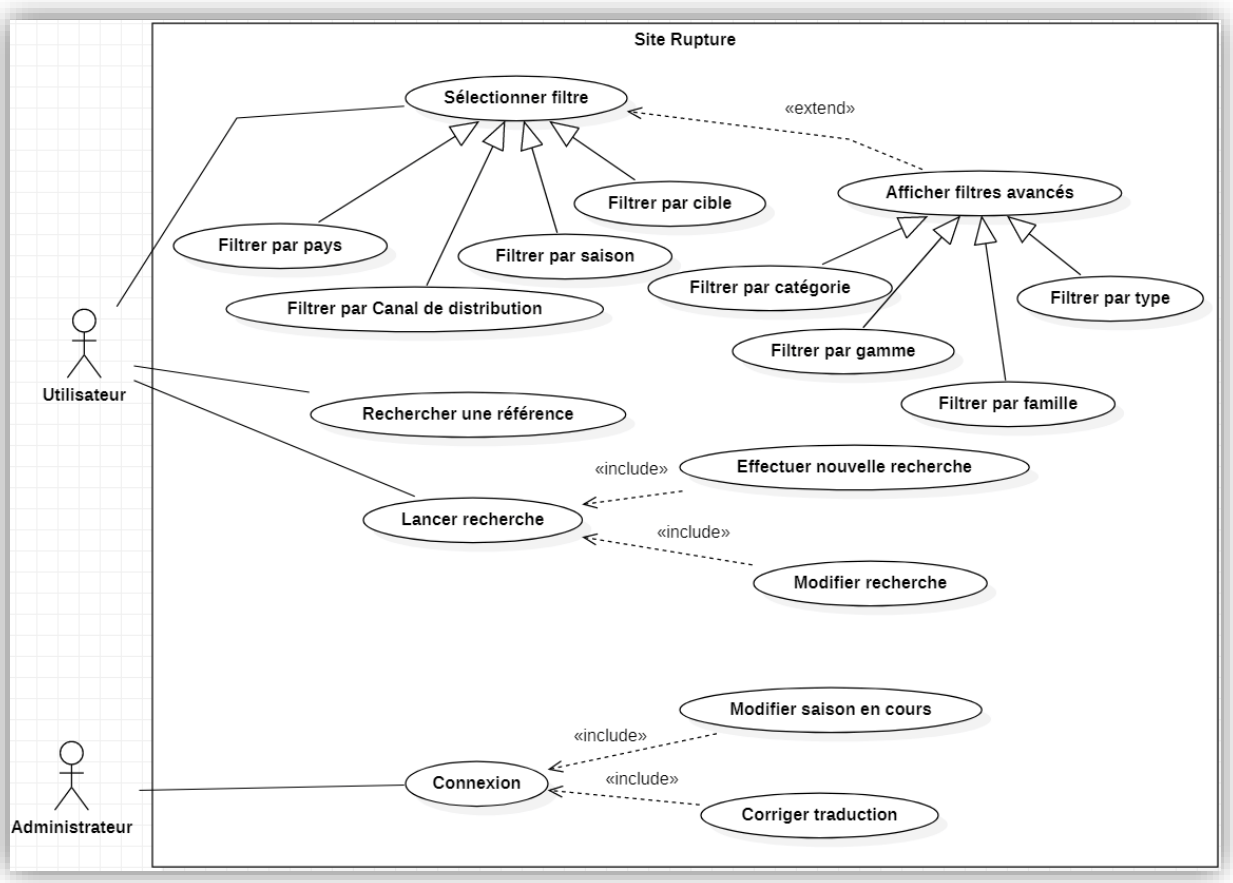
Avant d'en arriver à utiliser un Web Service, nous nous sommes interrogés sur la nécessité de recréer la base de données. J'ai donc réalisé un schéma **MCD** (Modèle Conceptuel de Données), ce qui m'a permis dans un premier temps de mieux comprendre l'utilité de chaque information stockée pour chaque produit (cf. Model de données). Nous avons finalement décidé d'accéder aux données de la base de données originale au travers d'un **Web Service**, afin de simplifier les requêtes et d'augmenter la sécurité (le mot de passe n'apparaît pas en clair dans le code du site pour la connexion à la base de données, contrairement à l'ancienne version du site).
Les données sont donc récupérées grâce à une requête http que l'on fait au Web Service. Ce dernier récupère l'ensemble des informations et des données nécessaires depuis la base de données, et les renvoie sous format **Json** au site. Le Web Service possède 3 méthodes :

- /Date_Maj ⇨ Récupère la date de la dernière mise à jour du Web Service
- /Filtre/{Données} ⇨ Récupère les filtres demandés (ex : « Distributions »)
- /Rupture/{Pays}/{Saison}/{Reference} ⇨ Récupère les résultats correspondants à la recherche

Voici un exemple de retour du Web Service après un appel pour obtenir l'état des stocks des produits de France pour la saison PE21 :


```
{ "Pays":"FRA",
  "Distribution":"Grande distribution",
  "Saison":"PE21",
  "Cat\u00e9gorie":"Chaussants",
  "Gamme":"Chausson",
  "Famille":"",
  "Type":"Mule",
  "Cible":"Femme",
  "R\u00e9f\u00e9rence":"95822",
  "D\u00e9signation":"Mule ergonomique",
  "CodeCouleur":"AA1",
  "Couleur":"Gris", "CodeTaille":"41",
  "Taille":"41",
  "Dispo":false,
  "DateDispo":"0000-00-00",
  "MAD":false
},
{ "Pays":"FRA",
  "Distribution":"Grande distribution",
  "Saison":"PE21", "Cat\u00e9gorie":"Chaussants",
  "Gamme":"Chausson",
  "Famille":"", "Type":"Mule",
  "Cible":"Homme",
  "R\u00e9f\u00e9rence":"96253",
  "D\u00e9signation":"Mule ergonomique velours",
  "CodeCouleur":"PDP",
  "Couleur":"Pied de poule",
  "CodeTaille":"40", "Taille":"40",
  "Dispo":false, "DateDispo":"0000-00-00",
  "MAD":false
}
…
```

## Cas d'utilisation




*Use case Diagramme*

## Fonctionnalités

L'ensemble des fonctionnalités que doit proposer le site sont les suivantes :

- Rechercher des produits via des filtres
  - Ajouter et retirer des filtres en sélectionnant/désélectionnant les différents boutons/menus déroulants.
  - Bouton de recherche qui envoie les données vers la page de résultats.
- Affichage des résultats de la recherche
  - Afficher la date de dernière mise à jour des données
  - Possibilité de relancer une nouvelle recherche en sélectionnant le bouton en question
  - Affichage d'un résumé des filtres sélectionnés sous forme de liste
  - Affichage des résultats sous forme de tableau regroupant les disponibilités pour chaque sku
- Page d'administration
  - Modifier simplement des saisons affichées sur la page d'accueil
  - Modifier les traductions du site
- Le site doit être responsif (adaptatif) sur tout type d'appareil

## Architecture

Le site a été réalisé grâce à la technologie NodeJs, en utilisant le Framework Express. Ce dernier suit un schéma MVC (Model – Vue – Controller), qui permet une plus grande lisibilité des différentes pages, et des processus qui vont conduire à l'affichage de ces pages.

Voici les éléments qui ont été créés pour ce projet :

- Le dossier **assets** contient les images affichées sur le site telles que le logo de l'entreprise.

- Le dossier **data** contient des fichiers au format Json permettant de sauvegarder les résultats de certains appels faits au Web Service :
  - ✓ **DateMaj.json** : La dernière date de mise à jour du site
  - ✓ **Filtres.json** : Les filtres affichés sur la page d'accueil

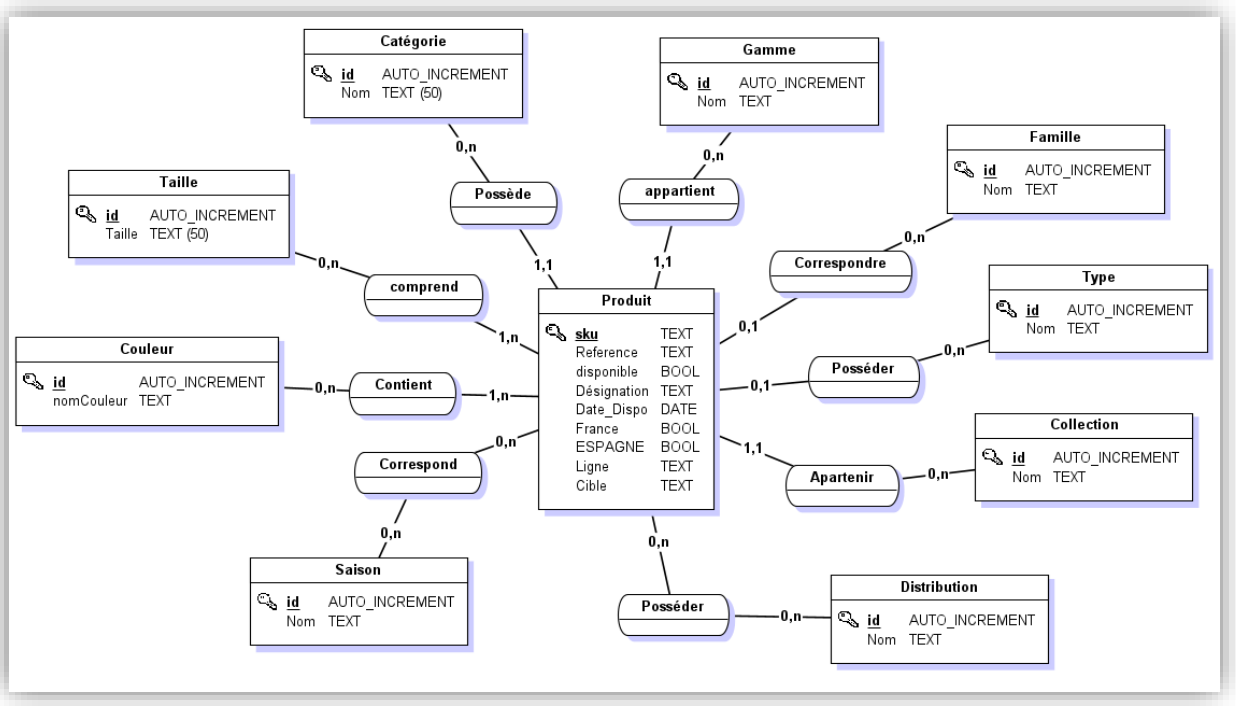
  La sauvegarde de ces éléments directement dans un fichier permet de limiter les requêtes au Web Service, et d'accélérer le chargement du site.

  - ✓ Ce dossier possède également un fichier **login.json** contenant l'identifiant et le mot de passe de l'administrateur.
  - ✓ **Saisons.json** : Les saisons affichées sur la page d'accueil

- **Node_module** est un dossier généré par NodeJs. Il contient des éléments nécessaires au bon fonctionnement de l'application.
- Le dossier **route** contient 3 fichiers : **admin.js**, **index.js** et **resultat.js**. Chacun de ces fichiers contient différentes routes qui vont mener à certaines parties du site. C'est également à l'intérieur de ces fichiers que sont effectués les traitements nécessaires avant l'affichage des pages (récupération des informations, tris et filtrage des donnés etc…).
- Le dossier **views** contient l'ensemble des vues du site. Un sous-dossier sépare celles accessibles par les utilisateurs lambda de celles accessibles uniquement par l'administrateur.
- Le fichier **package.json** regroupe les noms et versions de l'ensemble des packages à installer avec le projet pour son bon fonctionnement. Il nous permet également de demander à npm d'installer tous ces packages en ne réalisant qu'une seule commande.
- Le fichier **server.js** est la porte d'entrée de l'application. Il contient les chemins permettant d'accéder aux fichiers du dossier route. Il contient également les actions devant se dérouler de manière cyclique, à savoir la vérification des données et leur mise à jour.

## Model de données

Lorsque nous étions durant la phase de réflexion, nous hésitions à recréer une nouvelle base de données qui modéliserait les informations nécessaires pour le bon fonctionnement du site. Notre choix s'est finalement porté sur la récupération directement dans la base source, en communiquant avec elle à travers un Web Service. Il reste tout de même intéressant d'observer le schéma de base (MCD – Model Conceptuel de Données) réalisé. La conception de ce dernier m'a poussé à poser de nombreuses questions sur le fonctionnement des références produits.




*MCD*

Chaque table a son rôle spécifique que je vais détailler :

*Produit* : Représente les produits stockés dans la base. Il contient d'autres informations en provenance des autres tables, via des clés étrangères.

*Saison* : Permet de stocker les différentes saisons auxquelles un produit appartient

*Couleur* : Permet de spécifier la/les couleur(s) éventuelles de chaque produit

*Taille* : Permet de spécifier la/les taille(s) éventuelles de chaque produit

*Catégorie* : Permet de spécifier la catégorie de chaque produit

*Gamme* : Permet de spécifier la gamme de chaque produit

*Famille* : Permet de spécifier la famille éventuelle de chaque produit

*Type* : Permet de spécifier le type éventuel de chaque produit

*Collection* : Permet de spécifier la collection (saison de lancement) de chaque produit

*Distribution* : Permet de spécifier la/les distributions (points de vente) de chaque produit

## Documentation des différentes fonctionnalités

### Accueil / Sélection des filtres

| | |
|---|---|
| **Route dans index.js** | router.get('/:valid?', function(req, res) ) |
| **Paramètres et cookies** | Optionnel<br>- valid : Peut contenir un message d'erreur (par exemple, lorsqu'un champ obligatoire n'est pas sélectionné)<br>Cookies<br>- FiltresChecked : Contient les champs sélectionnés<br>- openMenu : Contient la position du menu des filtres avancés<br>- onlyRupture : Contient le statut (coché ou non) de la case « Afficher uniquement les références contenant des ruptures »<br>- onlyRefActives : Contient le statut (coché ou non) de la case « Afficher uniquement les références actives » |
| **Vue** | /views/index.ejs |
| **Rôles** | - Afficher les filtres disponibles<br>- Permettre à l'utilisateur de sélectionner les filtres désirés<br>- Rechercher des produits en tenant compte des filtres sélectionnés |
| **Éléments affichés** | - Header contenant le logo d'Isotoner<br>- Champs :<br>- Checkbox (x2) : « Afficher uniquement les références contenant des ruptures » et « Afficher uniquement les références actives »<br>- Buttons (x13) : Permet de sélectionner le filtre désiré (1 seul par catégorie maximum. Ex : On ne peut sélectionner qu'un seul pays, qu'un seul canal de distribution etc…)<br>- Select (x4) : Permet de sélectionner le filtre désiré<br>- Input : Permet d'entrer une référence (optionnel)<br>- Button : Lance la recherche |
| **Exceptions** | Une erreur peut être levée dans les cas suivants :<br>- Erreurs affichées en console.log (détectée depuis cette route):<br>- Si le serveur n'a pas réussi à récupérer les données depuis le Web Service.<br>- Erreurs affichées directement sur la page (détectée depuis la route '/resultat' ou depuis le front lors de la vérification des champs dans la fonction *validation()* présente dans le *footer.ejs*) :<br>- Si on n'a pas sélectionné de Pays<br>- Si on n'a pas sélectionné de saison<br>- Si on a sélectionné plusieurs pays (au lieu d'un seul)<br>- Si on a sélectionné plusieurs saisons (au lieu d'une seule)<br>- Si on n'a pas sélectionné au moins un pays et une saison |

### Description du code / de l'algorithme

Vue (index.ejs) :

- Affiche une div contenant un message d'erreur si la variable *messageErreur* existe.
- Affiche chacun des boutons et des listes déroulantes en remplissant leur contenu avec les variables reçues depuis le serveur (sauf pour France et Espagne, puisque ces valeurs ne changent pas).
- Contient des checkbox cachés en bas de la page. Avant l'envoi du formulaire, celles-ci seront cochées automatiquement. Ceci permet d'envoyer les données du formulaire en post vers le serveur.

Controller (routes/index.js – '/') :

- Récupère toutes les données du fichier *data.json* et les enregistre dans des variables.
- Trie des champs Catégorie, Gamme, Famille et Type dans l'ordre croissant.
- Stock la valeur des cookie *openMenu*, *onlyRupture*, *onlyRefActives* et *FiltresChecked* dans une variable.
- Renvoie toutes ces informations à la vue *index.ejs*.

#### Affichage des résultats

| | |
|---|---|
| **Route dans resultat.js** | router.post('/resultat', function(req, res) ) |
| **Paramètre** | Aucun |
| **Vue** | /views/resultat.ejs |
| **Rôles** | - Afficher les résultats correspondants aux critères de recherches entrés par l'utilisateur sur la page précédente |
| **Éléments affichés** | - Header contenant le logo d'Isotoner<br>- Button : Effectuer une nouvelle recherche<br>- Table : Affiche les filtres sélectionnés, le nombre de références et de SKU affichés, et une légende expliquant couleurs affichées dans les tables de résultats.<br>- p : Affiche la date et l'heure de dernière mise à jour des données.<br>- Table : Affiche les résultats sous forme de table les uns après les autres. Chaque table contient horizontalement les tailles et verticalement les couleurs des produits. Les cases sont remplies en fonction de l'état du produit (en stock ou non). |
| **Exceptions** | Une erreur peut être levée dans les cas suivants :<br>- Erreurs affichées en console.log (détectée depuis cette route):<br>- Si le serveur n'a pas réussi à récupérer les données depuis le Web Service (date et heure ou résultats de la recherche).<br>- Erreurs affichées directement sur la page d'accueil ('/') mais qui sont détectées ici :<br>- Si on n'a pas sélectionné de Pays<br>- Si on a sélectionné les 2 pays (au lieu d'un seul)<br>- Si on n'a pas sélectionné au moins un pays et une saison<br>- Si on a sélectionné plusieurs saisons (au lieu d'une seule) |

### Description du code / de l'algorithme

Vue (resultat.ejs) :

- Affiche une table qui résume les filtres sélectionnés, le nombre de références et de SKU affichés, et une légende expliquant les couleurs affichées dans les tables de résultats.
- Affiche la date et l'heure de dernière mise à jour.
- Boucle sur tous les résultats afin d'afficher les informations du produit en cours.

Controller (routes/resultat.js – '/resultat') :

- Récupère la date et l'heure de la dernière mise à jour depuis le Web Service. Pour cela, on fait appel à http.get('<url du Web Service pour trouver le résultat>').
- Vérifie les champs sélectionnés par l'utilisateur.
- Récupère le pays et la saison sélectionné.
- Recherche les produits correspondants à la recherche de l'utilisateur, en effectuant une requête au Web Service. Pour cela, on fait appel à http.get('<url du Web Service pour trouver le résultat>').
- Filtre les résultats obtenus via la recherche, en appliquant les filtres sélectionnés par les utilisateurs.
- Reformate le résultat reliant chaque produit à sa référence de façon à faciliter les traitements.
- Met l'ensemble des résultats dans un dictionnaire (Clé : Référence ; Valeur : les informations du produit).
- Rajoute un attribut *allTailles* pour chaque objet (un objet par référence) afin de stocker toutes les tailles dans l'ordre croissant associé à chaque référence.
- Rajoute un attribut *couleursTailles* pour chaque objet (un objet par référence) afin de stocker dans un dictionnaire l'ensemble des couleurs associées aux tailles correspondantes.
- Renvoie les résultats à la vue *resultat.ejs*.
- Affiche les différentes erreurs dans la console si nécessaire.
- Redirige vers la page d'accueil avec un message d'erreur personnalisé si un champ a mal été rempli par l'utilisateur.

## Page d'administration

| | |
|---|---|
| **Routes dans admin.js** | router.use('/dashboard', function(req, res) ) // Menu admin<br>router.post('/saison, function(req, res) ) // Changement de saison<br>router.use('/saison, function(req, res) ) // Affichage des saisons<br>router.use('/disconnect, function(req, res) ) // Déconnexion<br>router.post('/', function(req, res) ) // Vérification id/mot de passe<br>router.use('/', function(req, res) ) // Connexion |
| **Cookie** | Optionnel<br>- session : Si l'administrateur est connecté, il récupère le token généré par l'application afin de rester connecté pendant la journée. |
| **Vues** | /admin/dashboard.ejs // Menu admin<br>/admin/saison.ejs // Affichage et changement des saisons<br>/admin/connexion.ejs // formulaire de connexion |
| **Rôles** | /dashboard :<br>- Afficher un menu afin de sélectionner les paramètres du site à changer<br>- Afficher un récapitulatif comprenant les saisons en cours et la dernière mise à jour du Web Service<br>/saison :<br>- Afficher les saisons en cours<br>- Permettre la modification des saisons en cours<br>/disconnect :<br>- Déconnecte l'administrateur et réinitialise sa session<br>/ :<br>- Afficher un formulaire de connexion<br>- Vérifier que les informations entrées sont correctes |
| **Éléments affichés** | Différent en fonction de la page affichée |
| **Exceptions** | Une erreur peut être levée dans les cas suivants :<br>- Connexion avec un login ou un mot de passe erroné |

### Description du code / de l'algorithme

- Vues

  connexion.ejs :

  - Affiche le formulaire de connexion.

  dashboard.ejs :

  - Affiche un bouton de déconnexion.
  - Affiche un menu permettant de choisir ce que l'administrateur souhaite modifier sur le site.
  - Affichage d'un résumé des saisons actuelles et de la dernière mise à jour du Web Service.

saison.ejs :

- Affiche un bouton permettant de revenir au menu principal.
- Affiche les saisons actuelles.
- Affiche un aperçu des saisons telles qu'elles apparaissent sur l'accueil, ainsi que des boutons pour passer aux saisons suivantes ou précédentes.
- Affichage d'un éventuel message de confirmation en cas de changement.

- Controller (routes/admin.js)

/dashboard :

- Vérifie si l'utilisateur est bien connecté, si ce n'est pas le cas, on redirige vers la page de connexion.
- Si l'utilisateur est connecté, récupère les saisons et la date de mise à jour et les renvoie à la vue du tableau de bord.

/saison (post) :

- Vérifie si l'utilisateur est bien connecté, si ce n'est pas le cas, on redirige vers la page de connexion.
- Si l'utilisateur est connecté, réécrit les saisons actuelles sur le fichier *saisons.json*.

/saison (use) :

- Vérifie si l'utilisateur est bien connecté, si ce n'est pas le cas, on redirige vers la page de connexion.
- Si l'utilisateur est connecté, récupère les saisons actuelles depuis le fichier *saisons.json*.
- Renvoie éventuellement une variable indiquant à la vue d'afficher un message de succès en cas de changement de saisons.

/disconnect (use) :

- Réinitialise le cookie session de l'utilisateur et le redirige vers la page de connexion.

/ (post) :

- Vérifie si l'utilisateur est bien connecté, si ce n'est pas le cas, on redirige vers la page de connexion.
- Si l'utilisateur est connecté, on le redirige vers le tableau de bord.
- Si l'utilisateur n'a toujours pas été redirigé, c'est qu'il tente de se connecter. On vérifie donc si son identifiant et son mot de passe correspondent avec les informations que l'on a sur le fichier *login.json*.
- Si c'est le cas, on lui ajoute le cookie *session* avec le numéro de session du jour.
- Sinon, on le redirige vers la page de connexion.

/ (use) :

- Vérifie si l'utilisateur est bien connecté, si ce n'est pas le cas, on redirige vers la page de connexion.
- Si l'utilisateur est connecté, on le redirige vers le tableau de bord.
- Sinon on lui affiche la vue du formulaire de connexion.

## Synchronisation des données

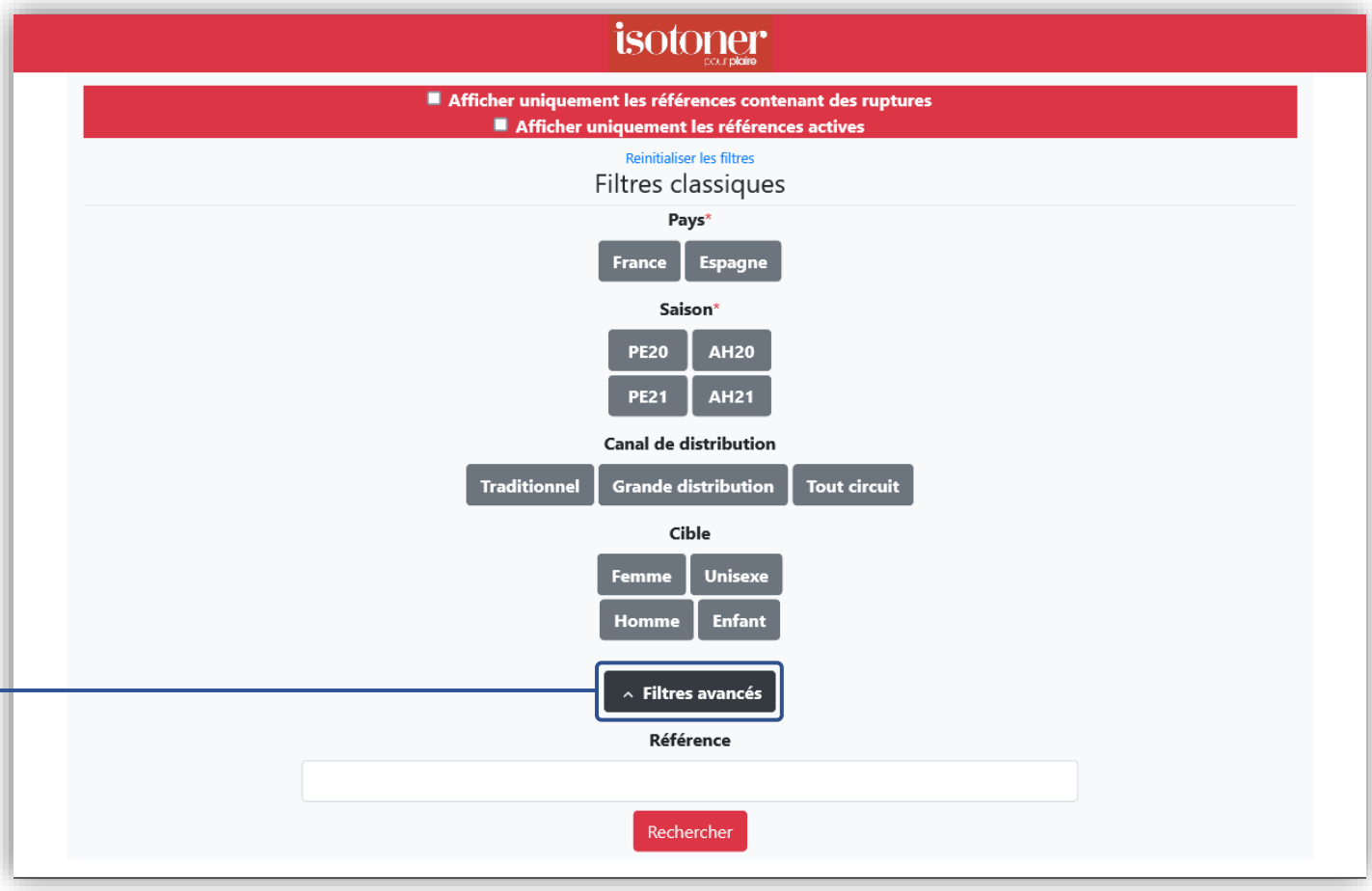
Controller (server.js) :

- Fonction asynchrone qui va boucler à l'infini sur une suite d'actions :
  - Attente d'une durée de 600 000ms (10 minutes).
  - Récupération de la date de mise à jour du Web Service.
  - Comparaison avec la date du jour.
  - Si les dates sont différentes et que la mise à jour est terminée, on récupère toutes les données des filtres à afficher sur la page d'accueil.
  - Récupère les champs de chaque bouton depuis le Web Service. Pour cela, on fait appel à http.get('<url du eb Service pour trouver le résultat>').
  - Récupère le résultat que l'on va mettre au format JSON (afin de pouvoir traiter ce résultat).
  - Vérifie si les fichiers *filtres.json* et *saisons.json* existent, sinon on les crée.
  - Génère un token valable pour la journée afin de permettre aux administrateurs de rester connecté.
  - Écriture des données dans un fichier *data.json*.

## Pages du site

Les maquettes ont été réalisées en HTML et en utilisant la librairie Bootstrap. Elles sont basées sur l'ancienne version du site, tout en modernisant l'interface.

### Page de sélection des filtres




*Nouvelle page de sélection des filtres*

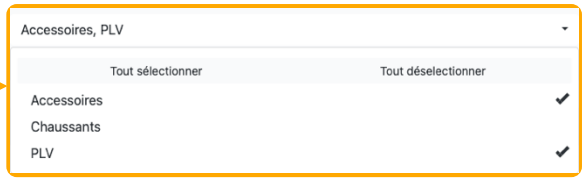
**Il s'agit de la page d'accueil. Elle permet à l'utilisateur de sélectionner un ou plusieurs filtre(s) afin d'effectuer sa recherche**

De nouveaux filtres ont été ajoutés par rapport à l'ancienne version :

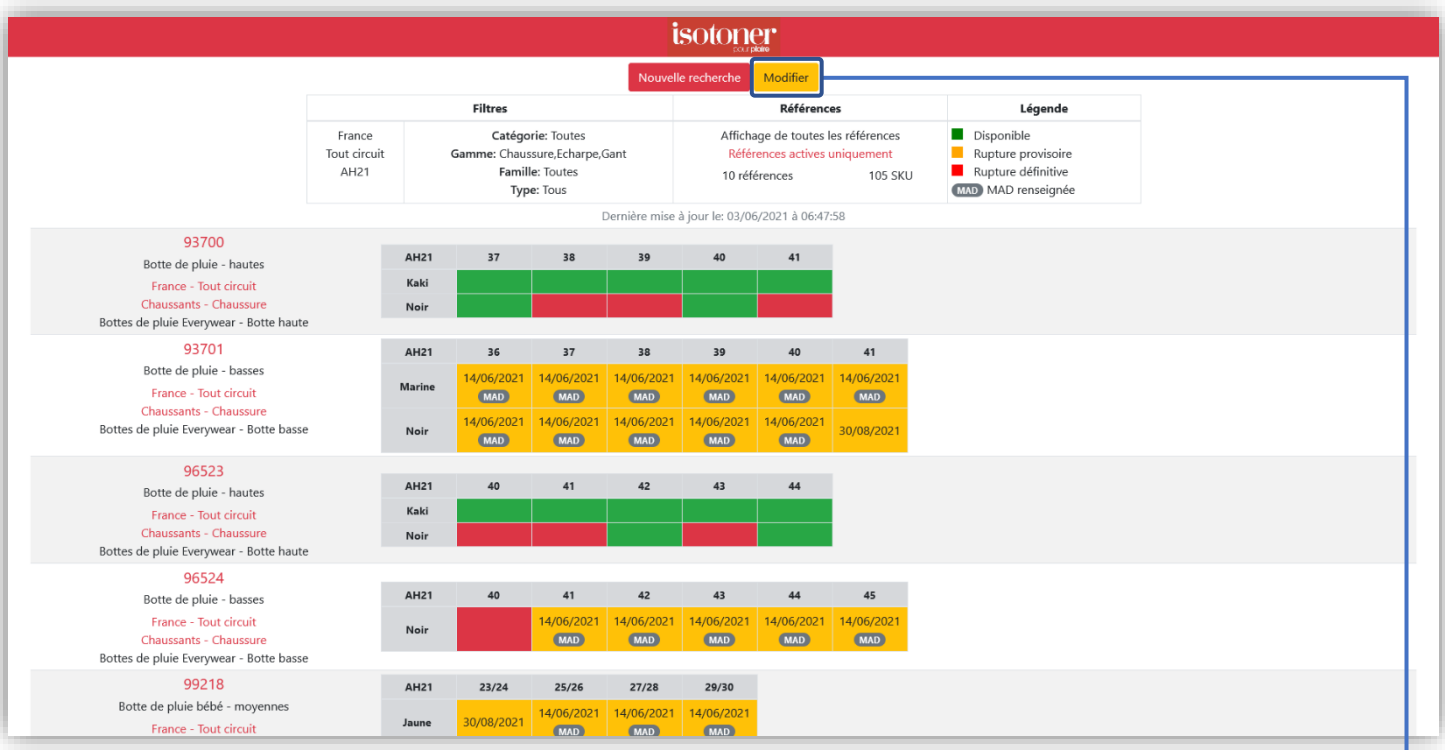
- Saison : Mise à jour de l'existant ⇨ Affiche les 4 saisons en cours
- Cibles (Femme, Homme, Unisexe, Enfant)
- Afficher uniquement les références contenant des ruptures


Bouton permettant d'afficher des filtres avancés.

Chacun d'entre eux possède un menu déroulant permettant de sélectionner plusieurs filtres en même temps, comme dans l'exemple ci-dessous :



## Page de résultats




*Affichage des résultats*

**Il s'agit de la page de la page permettant de consulter les résultats d'une recherche**

L'ensemble des produits correspondants aux critères sélectionné sur la page précédente influent sur les résultats affichés sur cette page.

Le bouton « Modifier » permet de retourner à la page d'accueil en présélectionnant les filtres saisis lors de la recherche.

Si l'utilisateur quitte le site après une recherche et retourne dessus, le comportement par défaut sera celui d'une modification de sa recherche. Autrement dit, les champs seront présélectionnés sur les champs saisies lors de sa dernière recherche. L'utilisateur a toujours la possibilité de les désélectionner en cliquant sur le bouton « Réinitialiser les filtres » de la page principale.

## Difficultés rencontrées

Cette année en entreprise m'a permis de découvrir toutes les étapes de réalisation d'un projet et toutes les difficultés qui l'accompagnent. Que ça soit par la découverte d'un nouvel environnement de travail, ou de soucis plus spécifiques liés aux projets, de nombreux obstacles ont dû être surmontés.

Tout d'abord la toute première complication que j'ai rencontrée, a été de m'habituer à un système d'exploitation que ne connaissait pas auparavant. En effet, l'utilisation de l'environnement Mac OS, avec tous les outils et les fonctionnalités qui l'accompagnent, était une première pour moi. Cela m'a beaucoup ralenti lors des premiers développements. C'est après plusieurs semaines de travail sur ce système d'exploitation et d'utilisation du matériel qui m'a été fourni, que j'ai fini par me sentir à l'aise et à pouvoir enfin avancer correctement.

Mise à part l'environnement, c'est la méthode de travail elle-même que j'ai eu du mal à assimiler au départ. En effet, mon maître d'apprentissage étant dans le Cantal, les seules interactions que nous avions, se faisaient par appel vidéo ou par message. Finalement, la mise en place d'une bonne communication m'a été extrêmement bénéfique. En effet, il a été décidé que je devrai communiquer quotidiennement l'avancement du projet, mes éventuelles difficultés, ainsi que mes demandes. De plus, les points réguliers que nous avons faits en vidéos m'ont également permis de mieux comprendre les enjeux du projet, et de mieux cerner les besoins en termes de conception.

Sur les projets en eux-mêmes, j'ai également rencontré quelques difficultés. Tout d'abord, l'utilisation de nouvelles technologies que je n'avais jamais approchées auparavant telles que NodeJs, ou la découverte de nouveaux concepts telles que les Web Services. Nous avons également envisagé d'autres possibilités, comme l'utilisation de flutter, mais celles-ci n'ont pas été retenu, car la montée en compétences était trop importante, et le projet ne devait pas avoir trop de retard.

J'ai également eu l'occasion durant ce projet, d'avoir des retours utilisateurs sur le travail effectué. Cependant, ces retours ne sont intervenus que très tardivement dans le développement de l'application. Ainsi, une fois cette dernière presque achevée, j'ai été amené à réaliser des modifications sur le code, qui n'étaient pas prévues dans le planning de base. De plus, j'ai pu expérimenter pour la première fois, les échanges entre développeur et utilisateur. J'ai notamment appris à éviter d'évoquer les aspects techniques des points relevés par les utilisateurs. Ces derniers souhaitent généralement la modification de certains aspects ou de certains comportements, sans se soucier des changements majeurs que cela peut entraîner dans le code du site.

J'ai également beaucoup de difficultés sur un point algorithmique précis du projet. En effet, les résultats devaient être affichés sous forme de tableau, comme celui-ci :

| AH20 | 6 | 6.5 | 7 | 7.5 | 8 |
|---|---|---|---|---|---|
| Bleu royal | | | | | |
| Bleu.canard | | | | | |
| Bordeaux | | | | | |
| Camel | | | N/A | | |
| Cognac | 04/10/2021 | 04/10/2021 | 04/10/2021 | 04/10/2021 | 04/10/2021 |
| Gris | N/A | N/A | | | N/A |
| Marine | 04/10/2021 | | 04/10/2021 | 04/10/2021 | 04/10/2021 |
| Marron | | | 04/10/2021 | 04/10/2021 | |
| Mastic | | | | | |
| Noir | 30/08/2021 | 04/10/2021 | 04/10/2021 | 04/10/2021 | 30/08/2021 |
| Orange | | | | | |
| Parme | | | | | |
| Poudré | | | | | |
| Rouge | | | 04/10/2021 | 04/10/2021 | 04/10/2021 |
| Sapin | 04/10/2021 | | | | |
| Seigle | | | | | |
| Vert | N/A | N/A | | N/A | N/A |
| Violet | | | | N/A | |

*Tableau présentant les ruptures pour un produit*

Or, l'algorithme pour générer les résultats de cette manière m'a posé quelques difficultés. En effet, le Web Service renvoie l'ensemble des produits dans un tableau contenant du Json (Annexe 3 - Retour des produits par le Web Service).

Le souci est que dans un tableau comme celui ci-dessus, certains produits ne sont pas renvoyés par le Web Service. Dans ce cas, il faut les afficher en rouge avec la mention « N/A ». C'était principalement la source de ma difficulté.

Dans une première version de l'algorithme, durant laquelle je n'avais pas noté cette nuance, j'affichais simplement l'ensemble des cases les unes après les autres. Mais bien évidemment, cela ne fonctionnait pas pour les références ayant des résultats en moins. Lorsque j'ai remarqué cela, je souhaitais tester de nombreuses autres possibilités. Cependant, je me suis heurté à un autre souci. Le moteur de template (permet d'afficher plus facilement des informations en provenance du back vers le front, en faisant une séparation nette entre les deux) que j'utilisais n'était pas adapté à ce que je souhaitais faire, et manquais de documentation. J'ai donc décidé de changer le moteur de template de l'ensemble du projet, afin de pouvoir réaliser de nombreux essais d'algorithmes. J'ai choisi d'utiliser EJS (à la place de Nunjucks) qui est très utilisé avec NodeJs pour les nombreuses possibilités qu'il offre. J'ai donc passé un petit moment afin d'appréhender les fonctionnalités de ce moteur.

J'ai finalement trouvé une solution après plusieurs jours de tests et de recherches. Voici globalement son fonctionnement :

Je reformate dans un premier temps les résultats, afin de séparer la référence du reste des informations de chaque produit. Puis, je stock ceci dans un dictionnaire contenant la référence en clé,

et un tableau d'objets contenant les produits appartenant à cette référence. Je rajoute ensuite 2 attributs à ce tableau d'objets : un qui contiendra toutes les tailles uniques dans l'ordre, et un autre contenant un dictionnaire associant les couleurs et les tailles disponibles pour chacune d'entre-elles. On vérifie ensuite pour chaque couleur, si elle ne possède pas de taille, on met la taille à -1 afin de la différencier des autres. (Annexe 4 - Algorithme côté serveur pour la récupération des couleurs et des tailles de chaque produit).

Vient ensuite l'affichage du tableau de résultat sur la vue qui m'a pris quelques jours également. Celui-ci a été beaucoup plus simple à réaliser depuis que j'ai maximisé le nombre de traitements effectués côté serveur, étant donné les limites du moteur de template (Annexe 5 - Algorithme côté front pour l'affichage de chaque tableau de résultats).

# Conclusion et bilan

Isotoner est une entreprise centenaire qui a acquis depuis toutes ces années une expertise incontestable dans le domaine des gants, chaussons, parapluies etc…

Elle est aujourd'hui implantée dans de nombreux pays, et tend à s'étendre de plus en plus. Elle continue toujours à commercialiser ses produits phares, tout en innovant constamment sur les technologies utilisées dans ces derniers. Cette expansion croissante a des répercussions sur le nombre de produits proposé par la marque, mais également sur le volume de données que l'entreprise doit stocker. Ainsi, lorsqu'une modernisation de certaines infrastructures devient essentielle, certains composants logiciels doivent également être mis à jour.

C'est ainsi que lors de cette année d'alternance, j'ai eu l'occasion de côtoyer de nombreuses technologies dont je n'avais jamais eu l'occasion d'essayer. Elles m'ont permis de recréer un outil qui ne pouvait plus être maintenu, du fait des récentes évolutions dans le stockage des données des produits proposés par l'entreprise. Il était donc primordial que cet outil web, très utilisé par les commerciaux, les forces de vente et divers autres services, soit à jour le plus rapidement possible.

L'organisation et la planification d'un tel projet étaient une première pour moi. En effet, j'ai dû pour la première fois me retrouver face à des retours utilisateurs. Ces derniers pouvaient influer sur les futures fonctionnalités du site, et modifier par la même occasion, le planning prévisionnel du projet.

Les diverses difficultés auxquelles j'ai dû faire face cette année chez Isotoner, qu'elles soient algorithmiques ou bien liées à un nouvel environnement de travail, ne m'ont pas découragées. Elles m'ont au contraire, poussé à rester en éveil constant sur les nombreux aspects du développement informatique.

Les connaissances acquises durant cette année universitaire m'ont également été d'une grande utilité dans le bon déroulement du projet. Ces enseignements m'ont en effet aidé à gérer efficacement le temps qui m'étais accordé pour ce projet, à bien organiser mon code afin que celui-ci soit le plus « propre » possible, et que toutes les parties puissent être relu dans quelques années par une autre personne qui aurait pour mission de faire évoluer le site par exemple.

Je suis globalement fière de ce qui a été accompli pour ce projet, j'ai pu découvrir de nombreux concepts qui me seront très utiles durant mon avenir en tant que développeur. J'ai fait la connaissance de nombreuses personnes aux métiers extrêmement variés qui m'ont permis de m'intégrer au mieux chez Isotoner. J'ai également beaucoup échangé avec mon maître d'apprentissage qui a su me partager son expérience de la meilleure façon, afin de me faire réussir et à m'aider pour que je puisse donner le meilleur de moi-même dans ce projet, et dans tous ceux que j'aurai à l'avenir.

# Références

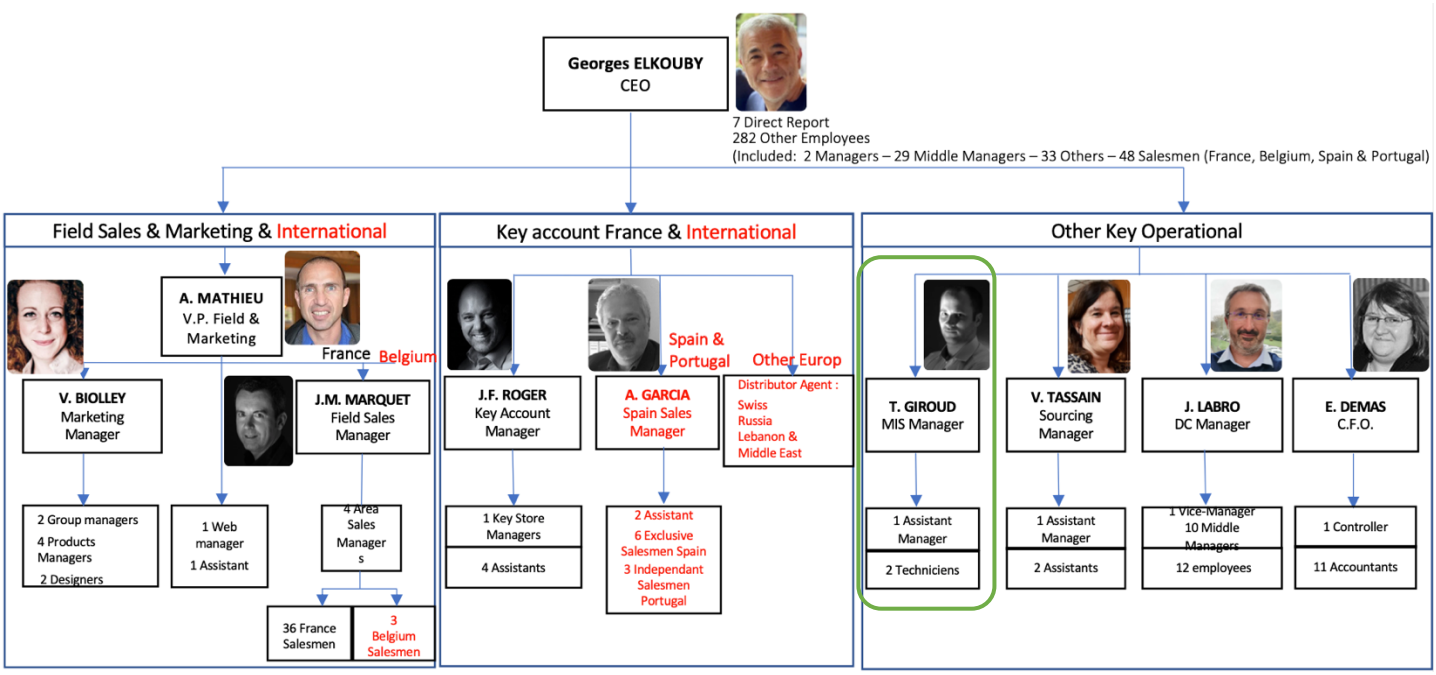
(s.d.). Récupéré sur Cantal passion: https://www.cantalpassion.com/personnages/5654-chanut-pere-et-fils-de-la-ganterie-a-isotoner

*Isotoner - Wikipédia*. (2020, 12 08). Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Isotoner

*La maison Isotoner*. (s.d.). Récupéré sur Isotoner: https://www.isotoner.fr/pages/la-maison-isotoner

# Annexes

*Annexe 1 – Organigramme de l'entreprise*




*Le service dans lequel je me trouve à Levallois.*

*Précision : L'équipe de développement informatique se trouve dans le Cantal, et ne sont pas représenté sur cet organigramme.*

*Annexe 2 - Choix de base de données SQL ou NoSQL*

## Choix base de données SQL ou NoSQL

| | SQL | NoSQL |
|---|---|---|
| **Requêtes complexes**[1] | Adaptée | Peu adaptée |
| **Stocker un nombre important de données**[2] | Adaptée | Pas adaptée |
| **Type de stockage de données**[3] | Structurées | Semi-structurées, non-structurées |
| **Evolution de la base de données**[4] | Peu adaptée | Adaptée |

## Choix des critères par rapport au projet

**Requêtes complexes** : Rajouter des filtres dans la requête

**Stocker un nombre important de données** : Un grand nombre de références disponibles dans la basede données

**Type de stockage de données** : La façon dont les articles sont définis dans la base de données

**Evolution de la base de données** : Pouvoir changer plus tard la façon de stocker les données

## Connexion et requêtes

SQL : Pour exécuter des requêtes sur une db MySQL il faut auparavant installer le package mysql surle serveur Node (*npm*[5] *install mysql*). Celui-ci permet la connexion, ainsi que l'exécution des requêtessur la base de données.[6]

NoSQL : Pour exécuter des requêtes sur une bd NoSQL (ici MongoDB), il faut auparavant installer lepackage mongodb sur le serveur Node (*npm*[7] *install mysql*). Celui-ci permet la connexion, ainsi que
l'exécution des requêtes sur la base de données[8]. Il est possible d'installer Mongoose afin desimplifier certaines opérations en définissant des schémas en amont[9].

## Héberger du Node

Il y a de nombreuses façons d'héberger un projet NodeJS (Hébergeur, VPS)[10]
Il est également possible d'avoir recours à un serveur d'applications web.[11]

[1] https://waytolearnx.com/2019/03/difference-entre-sql-et-nosql.html
[2] https://www.developpez.com/actu/124654/Quel-est-selon-vous-le-meilleur-SGBD-SQL-ou-NoSQL-un- developpeur-pense-que-vous-devez-opter-pour-SQL/
[3] https://www.cartelis.com/blog/choix-base-de-donnees/
[4] https://www.cartelis.com/blog/choix-base-de-donnees/
[5] https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/
[6] https://practicalprogramming.fr/nodejs-mysql/
[7] https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/
[8] https://practicalprogramming.fr/nodejs-mongodb/
[9] https://practicalprogramming.fr/nodejs-mongodb/
[10] https://oncletom.io/node.js/chapter-06/#hosting
[11] https://oncletom.io/node.js/chapter-06/#application-manager

*Annexe 3 - Retour des produits par le Web Service*


```
{ "Pays":"FRA",
  "Distribution":"Grande distribution",
  "Saison":"PE21",
  "Cat\u00e9gorie":"Chaussants",
  "Gamme":"Chausson",
  "Famille":"",
  "Type":"Mule",
  "Cible":"Femme",
  "R\u00e9f\u00e9rence":"95822",
  "D\u00e9signation":"Mule ergonomique",
  "CodeCouleur":"AA1",
  "Couleur":"Gris", "CodeTaille":"41",
  "Taille":"41",
  "Dispo":false,
  "DateDispo":"0000-00-00",
  "MAD":false
},
{ "Pays":"FRA",
  "Distribution":"Grande distribution",
  "Saison":"PE21", "Cat\u00e9gorie":"Chaussants",
  "Gamme":"Chausson",
  "Famille":"", "Type":"Mule",
  "Cible":"Homme",
  "R\u00e9f\u00e9rence":"96253",
  "D\u00e9signation":"Mule ergonomique velours",
  "CodeCouleur":"PDP",
  "Couleur":"Pied de poule",
  "CodeTaille":"40", "Taille":"40",
  "Dispo":false, "DateDispo":"0000-00-00",
  "MAD":false
}
…
```


*Annexe 4 - Algorithme côté serveur pour la récupération des couleurs et des tailles de chaque produit*

{ JavaScript }

```
// On reformate les résultats afin de faciliter le traitement (pour grouper
les produits par couleur et par taille dans la vue)
// Le nouveau format permettra d'avoir :   Référence : [ {"Pays": "FRA",
"Distribution": "Traditionnel" etc...}, {"Pays": "FRA", "Distribution":
"Traditionnel" etc...}]
let group = resultat.reduce((r, a) => {
    r[a.Référence] = [...r[a.Référence] || [], a];
    return r;
}, {});
// console.log(group2) // Afficher le résultat reformaté

// On met tous les résultat dans un dictionnaire. La clé est la référence,
et la valeur contient toutes les informations sur le produit
let dic = new Map()
for (const [key, values] of Object.entries(group)) {
    dic.set(key, values)
}
// console.log(dic.get("67102")) // ex: Récupère toutes informations pour
le produit ayant pour référence '99637'
// console.log(dic) // ex: Affiche tous les produits dans la console
```

```
// On rajoute un attribut allTailles qui contiendra toutes les tailles de
chaque référence dans l'ordre croissant
dic.forEach((values, keys) => {
    let toutesTailles = [];
    for (let i = 0; i < values.length; i++) {
        toutesTailles.push(values[i].Taille);
    }
    toutesTailles = [...new Set(toutesTailles)] // On retire les doublons
    toutesTailles.sort(); // On trie les tailles
    values.allTailles = toutesTailles // Cet attributs contient toutes les
tailles pour une référence
    dic.set(keys, values) // On réassigne chaque valeur à sa référence
})

// On rajoute un attribut couleursTailles qui contiendra un dictionnaire
possédant toutes les couleurs auquel on aura associé toutes les couleurs
dic.forEach((values) => {
    let couleurTailles = new Map(); // On crée un dictionnaire qui va
contenir les couleurs en clé et leur tailles associées en valeur
    let tailles = []; // On initialise un tableau de tailles. Il contiendra
l'ensemble des tailles associés à une couleur
    for (let i = 0; i < values.length; i++) { // On boucle sur tous les
produits d'une même référence
        if (i > 0 && values[i - 1].Couleur !== values[i].Couleur) {
            tailles = [] // On remet le tableau à vide quand on change de
couleur
        }
        tailles.push(values[i].Taille) // On rajoute la taille au tableau
        couleurTailles.set(values[i].Couleur, tailles) // On rajoute au
dictionnaire la couleur + le tableau de tailles associé
    }
    // console.log(couleurTailles) // ex: On peut afficher toutes les
tailles associées à chaque couleur


        couleurTailles.forEach((valuesTailles) => {

        // On va mettre des -1 aux emplacements où il manque des tailles
afin d'afficher des cases vides sur la vue
        if (valuesTailles.length !== values.allTailles.length) { // Si une
couleur contient le nombre de tailles maximal, cela ne sert à rien d'entrer
dans la boucle
            for (let i = 0; i < values.allTailles.length; i++) { // On
boucle sur l'ensemble des tailles
                if (valuesTailles[i] !== values.allTailles[i]) { // Si la
taille courante ne possède pas une couleur à un index i, on rajoute un -1 à
la place
                    valuesTailles.splice(i, 0, -1)
                }
            }
        }
    })
    values.couleursTailles = couleurTailles; // On rajoute l'ensemble des
couleurs et leur tailles correspondante dans un attribut
})
```

*Annexe 5 - Algorithme côté front pour l'affichage de chaque tableau de résultats*

{ Html + ejs }

```
<tbody>
<% var nb = 0 %> <%# On initialise une variable qui va servir à afficher
toutes les cases pour chacun des produits correspondant à la référence en
cours %>
<% entry.couleursTailles.forEach(function(tailles, couleur) { %> <%# On
boucle sur le dictionnaire couleursTailles contenu dans chaque référence %>
<tr>
    <th class="table-secondary align-middle" scope="row" width=100
style="font-size:0.9em"><%= couleur %> </th> <%# On affiche la couleur
en début de ligne %>
    <% for (var i = 0; i < tailles.length; i++) {%>
    <%if (tailles[i] === -1) {%> <%# Note : Ici on travail avec i car nb
concerne l'ensemble des produits alors que i ne correspond qu'à une seule
couleur %>
    <td class="bg-danger text-white align-middle">N/A</td> <%# Si la taille
en question est à -1, c'est qu'il n'y a pas cette taille pour la couleur
sélectionnée. On affiche donc une case rouge au lieu de vide %>
    <% } else {%>
        <% if (entry[nb].Dispo) {%> <%# Note : Ici on travail avec nb car
on boucle sur l'ensemble des produits de la référence %>
        <% if (entry[nb].DateDispo === "0000-00-00") {%> <%# S'il n'y a pas
de date de dispo, on l'affiche en vert %>
        <td class="bg-success"><%# entry[nb].Taille entry[nb].Couleur
%></td> <%# Retirer le commentaire pour pouvoir vérifier que tout est bien
placé %>
        <% } else {%>
            <td class="bg-warning align-middle"><%# entry[nb].Taille
entry[nb].Couleur %> <%# Retirer le commentaire pour pouvoir vérifier que
tout est bien placé %>
                <%= entry[nb].DateDispo %> <%# Sinon on l'affiche en jaune
%>
                <% if (entry[nb].MAD) {%> <%# Si MAD est à true, on affiche
un petit badge pour l'indiquer %>
                <span class="badge badge-pill badge-secondary">MAD</span>
                <% } %>
            </td>
        <% } %>
        <% } else{ %>
            <td class="bg-danger"><%# entry[nb].Taille entry[nb].Couleur
%></td> <%# Si le produit n'est pas dispo, on affiche la case en rouge %>
        <% } %>                          <%# ^ Retirer le commentaire pour
pouvoir vérifier que tout est bien placé %>
        <% nb = nb + 1 %> <%# On augmente nb de 1 uniquement si nous sommes
sur une case colorée %>
    <% } %>
    <% } %>
</tr>
<% }) %>
</tbody>
```