l'école d'ingénierie informatique

RAPPORT D'ACTIVITE DEVOPS

SEMWEEE

JANVIER-FEVRIER 2022

Denis Jongmanee

EPSI LILLE 2021/2022

DEUXIEME ANNEE B2

Remerciements

Je remercie d'abord **Rilah Mario Mihary**, responsable du développement qui m'a fait confiance dans les divers tâches qu'il m'a pu confier durant ce stage et ainsi qu'**Anthony Contat** de m'avoir accueilli dans son entreprise où j'ai pu acquérir une expérience professionnelle significative dans le domaine informatique.

Je remercie aussi l'équipe de développement de m'avoir aidé notamment **Andy Mihaja Andriamanday** qui a pris du temps à me répondre concernant mes sur interrogations sur le Back-end.

Je remercie enfin l'**EPSI**, qui m'a permis de réaliser ce stage qui s'inscrit dans mon parcours professionnel.

Table des matières

Introduction

Ce document est le rapport d'activité du stage de deuxième à l'EPSI de Lille afin d'obtenir un Bachelor DevOps.

Anthony CONTAT, **gérant de l'entreprise SEMWEE** m'a contacté à la suite de ma candidature sur une appel d'offre en ligne. J'ai été recruté grâce à mes compétences en accord avec son projet, que j'ai pu acquérir l'année dernière lors de mon premier stage.

Le stage s'est déroulé en télétravail en tant que stagiaire développeur et s'est effectué du 03 Janvier 2022 au 18 Février 2022 soit 7 semaines.

J'ai alors rejoint le pôle de développement qui travaille sur un outil proposé par SEMWEE. Les tâches qui m'ont été confiées concernent le Front-end ou/et Back-end de l'application.

Ce rapport d'activité se présente en plusieurs parties : La présentation de l'entreprise d'accueil, celle des technologies utilisées et celle des activités durant ces 7 semaines.

1. Présentation de l’entreprise d’accueil et des missions

1.1. Semwee

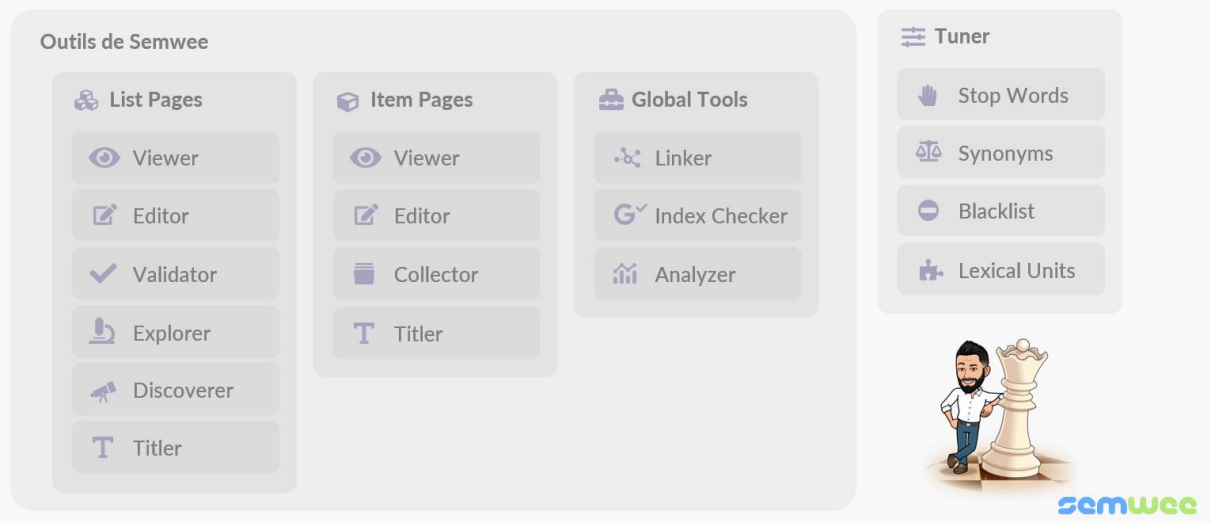
Semwee est actuellement en première phase de son projet : le développement de son application

Il s’agit est d’une suite d’outils à disposition des webmarketers afin que ces derniers améliorent les performances de sites catalogues en termes d'acquisition (SEO), de conversion (CRO), et d'expérience utilisateur (UX). Du côté du back-office Semwee apporte un lot important d'amélioration ergonomiques, et du côté du front-office, l'application permet d'adapter la structure du site web aux besoins des prospects. Semwee se veut être une solution aussi bien accessible aux marketeurs débutants qu'aux marketeurs confirmés, voire spécialisés, comme les chefs de projets web, ou les référenceurs.

Semwee recrute dans plusieurs domaines : développeur, webdesigner, chargé de commerce, etc…

La solution propose des outils sous la forme d’une application web qui utilise les Frameworks de Javascript.

Liste d’outils :



L’un des buts de l’application est de donner la possibilité à l’utilisateur de créer des projets où il peut charger des catalogues de produits pour générer des pages web type catalogue de produits.

1.2. Ma mission

Ma mission consistait à remplir diverses tâches. J'ai principalement travaillé sur la création et modification de projet et en termes d'outil : le Validator. J'ai participé à l'évolution du Front-end et du Back-end. De plus, j'ai pu donner mon avis sur le niveau du code source et celles des solutions présentes, pour ensuite proposer des améliorations avec l'accord de mon responsable.

Les technologies utilisées sont les Frameworks de Javascript : Angular pour le côté-client (front) et Node pour le côté-serveur (back) avec une base de données NoSQL MongoDB.

2. Technologies utilisées durant le stage

2.1. Le stack MEAN

L'application web développée par Semwee utilise le stack MEAN. Il s'agit d'une pile de logiciels Javascript et forme un ensemble qui permet de réaliser totalement un projet Web.

Il y a d'abord MongoDB représentant la base de données. Il utilise le NoSQL, qui est différent des bases de données classiques SQL. Le NoSQL est utilisé pour le stockage de données en grand nombres et offre une plus grande liberté entre mes collections (équivalent aux entités). Il est possible de lier les collections mais elles n'auront pas de fortes contraintes. Par exemple, il n'y a pas de condition à la suppression d'un objet. La différence avec les bases de données SQL se trouve aussi sur les identifiants. Chaque objet possède un « ObjectId ». C'est une chaine à 12 caractères et un type spécifique de MongoDB. Il est possible d'avoir une interface pour manipuler les données avec MongoDBCompass.

Ensuite, il y a Express et Node qui forment le Back-end. Node est l'application côté-serveur qui contient l'api et permet d'accéder à la base de données pour faire le lien avec le front. Express est un Framework de Node et permet de faciliter la création des routes de l'API.

Enfin, pour le front il y a Angular le Framework côté-client. Il tourne avec l'AJAX. C'est-à-dire qu'il n'a pas besoin de rafraichir l'url pour changer de page. Cela permet d'avoir une plus grande rapidité de navigation sur l'application. Il fonctionne avec les notions de components. Les fichiers javascript sont codées en Typescript, un langage strict et typé de Javascript. Il introduit aussi la notion de classes. Chaque component possède un fichier HTML, CSS et Typescript.

2.2. Les balises Angular

Dans mon précédent stage, j'ai aussi utilisé Angular comme technologie de développement pour le Front-end mais toutes les pages du site ont été écrites avec des balises HTML simple. Dans ce second stage, j'ai donc découvert les différentes balises qu'Angular propose. Ces balises sont des solutions alternatives qui permettent d'écrire du code HTML plus facilement. Ils sont appelés « Angular Material ». J'ai surtout utilisé les balises <mat-table> qui permet de créer des tableaux par colonne et <mat-card> pour l'afficher des modales.

2.3. RXJS

RXJS est une librairie pour la programmation asynchrone qui utilisent les Observables et proposent plusieurs fonctions qui permettent la gestion des lignes de codes (mais manière asynchrone). Cette librairie peut par exemple gérer la progression d'une barre de chargement. Elle offre aussi la possibilité d'utiliser les « Subscriptions », des objets qui peuvent exécuter des fonctions selon son paramétrage. Par exemple, il possible de lancer des exécutions de fonction de façon périodique.

2.4. Nodemon

Nodemon est une librairie que l'on peut installer sur le Back-end et permet de lancer un serveur Node. La différence d'un serveur lancé par Nodemon est qu'il est relancé automatiquement quand un fichier Javascript est modifié.

2.5. Versioning

L'équipe de développement utilise Github pour gérer le versioning du code. Pour la gestion des branches, chaque développeur qui participe au développement possède la sienne. Cela permet d'éviter des conflits entre les collaborateurs. Ces conflits sont résolus au moment des fusions sur la branche dite stable.

Pour accéder aux sources codes, le projet utilise des connexions par SSH au lieu du HTTPS. Cela permet d'ajouter des contributeurs sans demander l'accord du créateur du repository. Il suffit donc de générer une clé SSH et une personne qui a les droits doit entrer la clé publique du nouveau collaborateur dans le repository.

GitGraph, l'extension de VS Code qui permet de gérer les branches sans passer par les lignes de commande, est aussi utilisé. Il est pratique pour avoir un visuel l'arborescence des commits et permet de changer de branche de manière plus aisée.

2.6. Outils de communication

Les communications se font en distance étant donné la localisation différée des membres de l'équipe. Les réunions se font donc sur Zoom, logiciel de réunion en ligne. Une réunion appelé Daily Meeting (DM) se fait chaque matin pour informer l'avancement de chaque équipe (Design UX et Développement), partager les nouvelles informations et aborder les points qui peuvent poser problème.

De plus, Skype est aussi utilisé pour la communication par écrit. Plusieurs conversations classées par catégorie sont disponibles. Nous avons par exemple celui du Front-end, Back-end, Design UX ou Général. Il était aussi possible de se contacter par message privé.

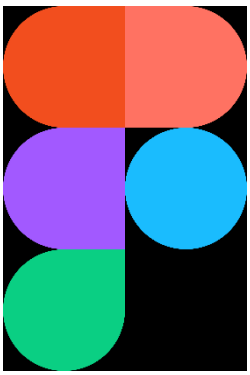
La conversation SMW Daily Reporting sert à lister les différentes tâches accomplis durant la journée et doit se faire chaque jour en fin de travail.

2.7. Figma

L’entreprise dispose de designers web qui ont en partie la charge de créer des maquettes du site sur Figma. Les développeurs ont accès à ces maquettes, ce qui leurs permettent de respecter le design voulu.

2.8. Trello

L’entreprise utilise Trello pour planifier les tâches de l’ensemble des équipes. Cet outil permet aussi de se concorder. Chaque tâche passe par plusieurs étapes : TO DO, DOING, Recette, Correction après Recette, PREPROD et enfin PROD (étapes pour les tâches de l’équipe de développement). Il y a aussi une catégorie qui permet de signaler les différents bugs rencontrés. Il est important de bien décrire sa tâche pour que celui qui se charge de la vérification (dans Recette) puisse bien comprendre ce qui a été fait.

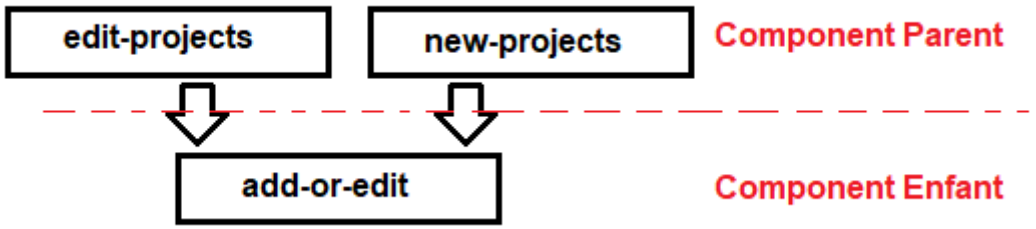


3. Mes activités

3.1. Création et édition d'un projet

Lorsque l'utilisateur se connecte sur le site avec son compte, il a la possibilité de créer ses projets pour générer les différentes pages de son E-commerce par des fichiers CSV.

Au niveau du code source, la création et édition d'un projet sont similaires, d'où d'utiliser un component commun, ce qui permet d'éviter des fichiers qui se ressemblent et qui se répètent. Angular permet de faciliter la communication entre les components : voici comment ses deux pages sont structurés :

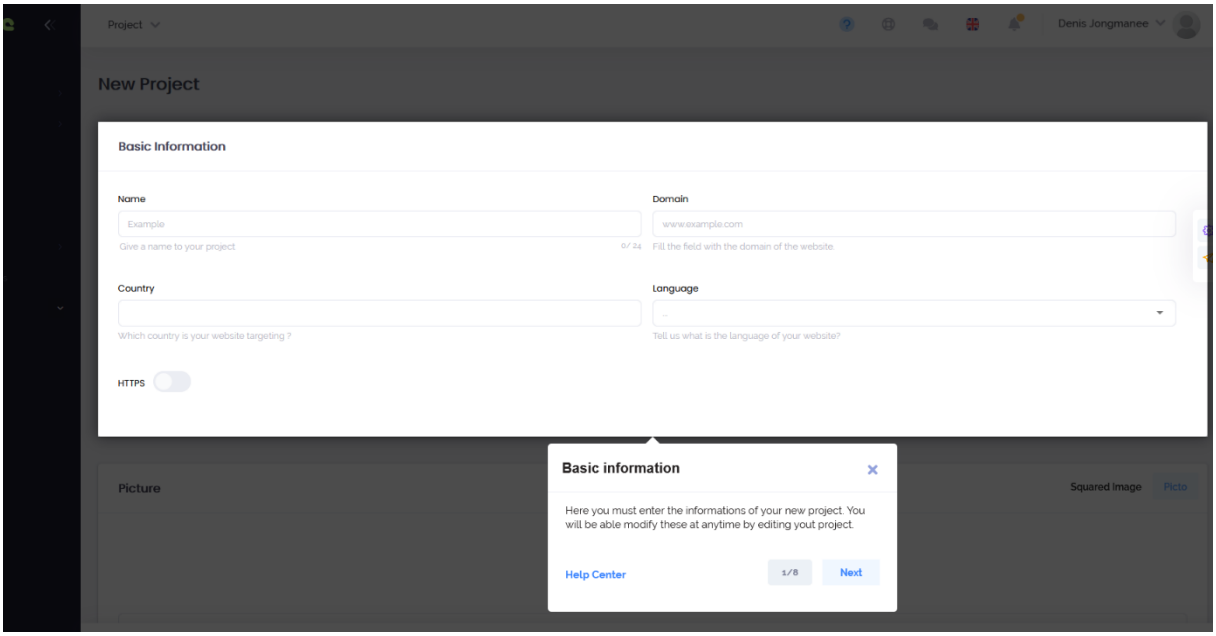


Le partage de données entre les components se fait en utilisant les « Inputs » (de parent à enfant) et « Output » (d'enfant à parent). De cette manière, On peut distinguer l'ajout et l'édition. Les fonctions et affichages communs se trouvent dans le component « Add-or-edit » et celles qui sont spécifiques à une fonctionnalité se trouvent dans le component parent.

3.1.1. New Project

New Project est la page qui permet de créer un projet. L'utilisateur est redirigé vers cette page automatiquement quand il souhaite accéder à la page « All Projects » (qui montre l'ensemble de ses projets) quand son compte n'a aucun projet. La page comporte un formulaire pour rentrer les informations, le pictogramme et le catalogue du projet.

3.1.1.1. L’assistant



La fonctionnalité d'assistant est liée au bouton "Interrogation" dans la top-bar.

Quand l'utilisateur clique dessus, certaines parties de la page où il se trouve deviennent cliquables et apportent de l'aide sur l'outil en question avec un système de bulles d'aide à travers lesquelles l'utilisateur peut naviguer d'une information à une autre.

Ma tâche consistait ici à traduire les boutons de l’assistant quand l’utilisateur change la langue du site.

L’application utilise un système de traduction de page où chaque texte possède des équivalents dans les autres langues. Ces textes sont contenues dans des fichiers JSON : fr.json pour la langue française par exemple. Le système de changement de langue a été implémenté et il suffit donc d’ajouter simplement la traduction dans le fichier JSON et de la faire appel dans les fichiers html. Pour cela il suffit d’utiliser un « pipe » Angular qui est une sorte de filtre.

```
{{Texte |translate }}
```

Un bouton permet de changer la langue et la traduction se fait automatiquement grâce au filtre pour chaque texte que comporte le site.

Le résultat pour les boutons de l’assistant :

```
<ngx-guided-tour
    skipText= "{{'AssistantButtons.HelpCenter'| translate}}"
    backText= "{{'AssistantButtons.Previous'| translate}}"
    nextText= "{{'AssistantButtons.Next'| translate}}"
    doneText= "{{'AssistantButtons.Done'| translate}}"
    progressIndicatorLocation= "top-of-tour-block">
</ngx-guided-tour>
```

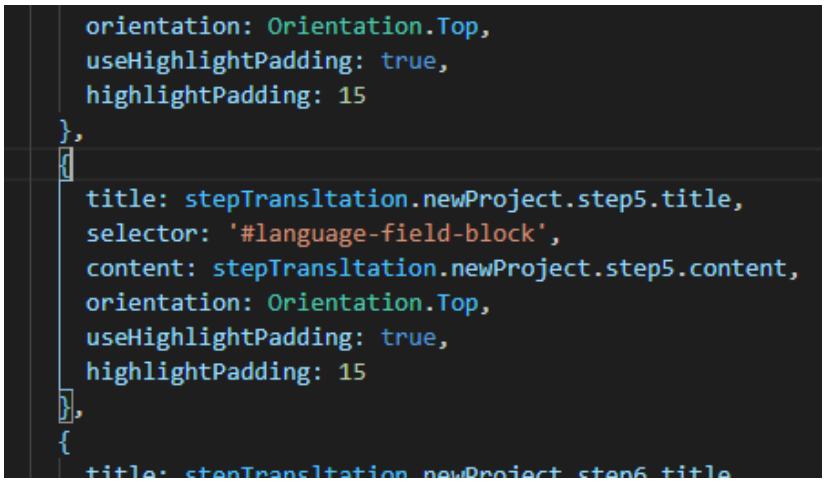
Pour arriver à ce résultat, j'ai dû d'abord regarder la documentation sur la librairie « ngx-guided-tour » pour savoir comment et où changer le texte des boutons.

Ensuite ma tâche consistait à rendre la page scrollable quand l'assistant est en exécution. En effet, quand l'assistant est dans l'une de ses étapes, il applique son propre CSS. De ce fait, il m'a fallu comprendre comment celui qui a développé la librairie ait décider d'agir sur la page avec sa librairie. Quand l'assistant est lancé, une surcouche visuelle est créée et il n'est plus possible de scroller (choix fait par le développeur de la librairie). Selon la documentation, il était possible d'exécuter du code au démarrage de l'assistant. Donc pour remédier à ce problème, j'ai utilisé le Javascript avec le DOM qui permet de modifier les éléments de style d'un block et d'y changer les propriétés. Ma logique consistait à réécrire le CSS que la librairie créait pour avoir le résultat voulu.

```
{
  title: stepTransltation.newProject.step1.title,
  selector: '#form-basicInfo-block',
  content: stepTransltation.newProject.step1.content,
  orientation: Orientation.Bottom,
  action: () => {

    //permet de corriger un probleme d'affichage du selector
    //probleme qui survient sur le navigateur Google Chrome
    setTimeout(() => {

      //un simple scroll suffit pour avoir la bonne affichage.
      //consiste à faire un scroll auto rapide et de revenir sur la posit

      if (!this.topPosition_Step1) {
        this.topPosition_Step1 = document.documentElement.scrollTop;
        document.getElementById('new-project-title').scrollIntoView()
        window.scroll(0, this.topPosition_Step1);
      }


    }, 500);

    //permet de scroller quand l'assistant est en cours
    const body = document.querySelector("body") as HTMLElement;
    body.style.overflow = "scroll";
  }
```

Cela montre qu'il est important de lire et comprendre l'ensemble de la documentation d'une librairie. Cela permet d'élargir au maximum son champ d'action.

On m'a demandé ensuite d'ajouter une nouvelle étape à l'assistant de New Project. L'assistant comportait huit étapes, je devais ajouter une neuvième. Les recherches et la compréhension de la librairie « ngx-guided-tour » précédemment acquises m'ont permis de remplir cette tâche aisément. Dans le fichier Typescript du component, j'ai ajouté un nouvel objet qui correspondait à l'étape dans la liste « step ».



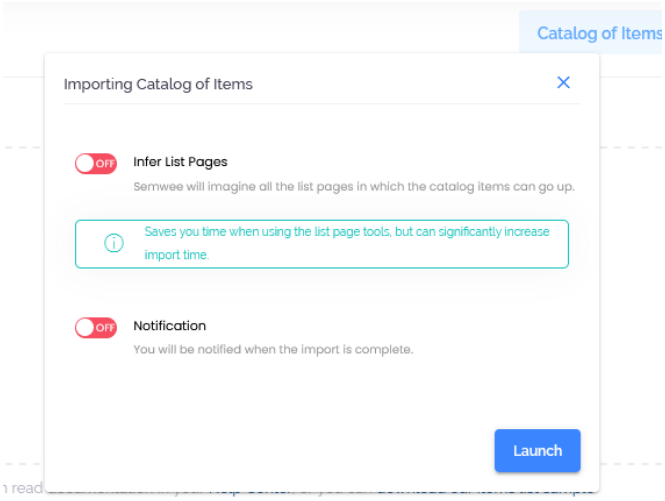
Une étape comporte différentes caractéristiques : « title » (titre du modale qui va apparaître au moment de l'affichage de l'étape pour afficher l'explication), « selector » (le bloc HTML qui va être mis en valeur), « content » (qui contient le texte du modale), orientation (spécifier où se trouve le modale par rapport au bloc) et « useHighlightPadding » et « highlightPadding » (qui permettent de d'ajouter un padding sur le bloc pour un meilleur rendu visuel). Les traductions des textes de l'assistant ne sont pas stockées dans un fichier JSON mais dans un service car les pipe ne peut être utilisé que dans les fichiers HTML.

3.1.1.2. Les étapes du chargement d'un catalogue dans un nouveau projet

L'utilisateur doit pouvoir charger un fichier csv qui représente son catalogue d'items sur l'application. C'est-à-dire que l'application doit lire le fichier et rentrer de manière correct les informations dans la base de données.

Le chargement d'un catalogue se fait à la création du projet mais peut se faire à l'édition. Cela implique la possibilité de créer un projet sans catalogue. Cependant quand il voudra accéder aux détails de son projet, l'interface va lui indiquer qu'il doit charger un catalogue pour accéder aux différents outils que propose Semwee.

Ma tâche consistait à ajouter une étape intermédiaire entre le choix du catalogue et son téléchargement. Il s'agit d'un modale dont son design a été pensé par l'équipe Design UX. Il doit demander à l'utilisateur l'avis sur deux informations : « Infer List Pages » et « Notifications ». Je n'ai cependant pas travaillé sur l'utilité de ces boutons. Ma tâche consistait seulement d'intégrer ce modal dans les étapes de chargement de catalogue.



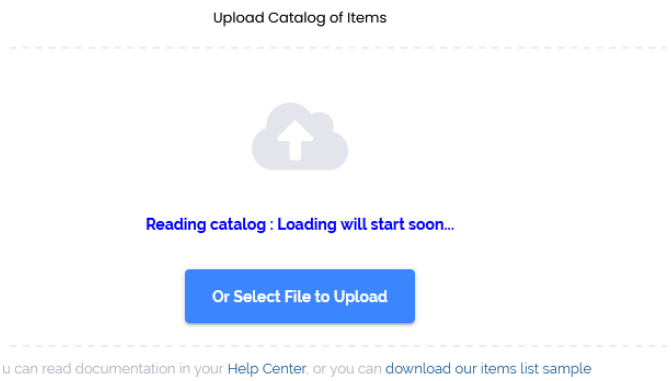
Pour ajouter ce modale, il m'a suffi de créer un nouveau component et de l'intégrer dans le fichier HTML de la page.

Angular facilite la création de component par ligne de commande :

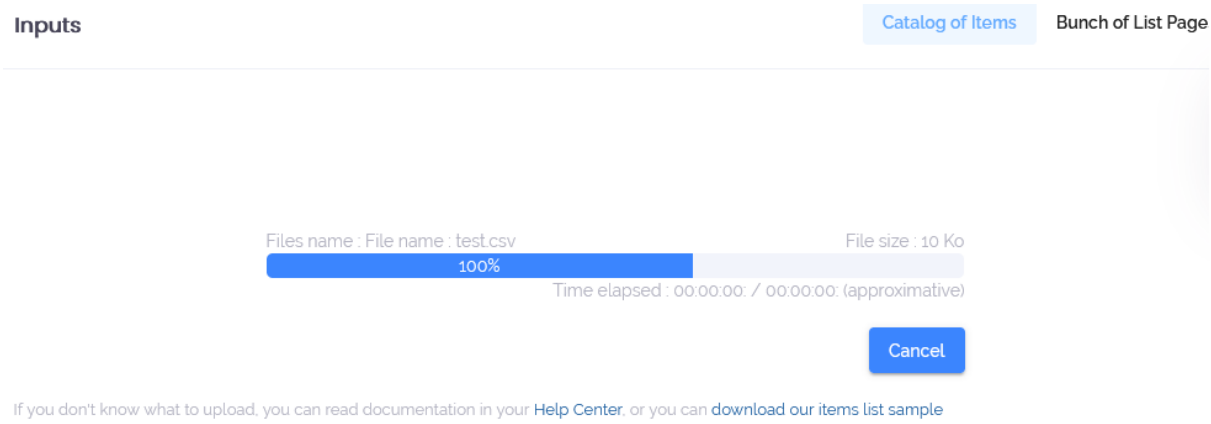
```
# Ng generate component /user-spaces/dashbord/components/projects/intermediate-modal
```

Cette commande va créer les différents et modifier les fichiers nécessaires à l'intégration d'un nouveau component automatiquement.

Après avoir intégré et présenté mon travail à mon responsable, j'ai remarqué au moment où l'on clique sur Launch, la barre de chargement ne s'affichait pas directement car la lecture du fichier pouvait prendre du temps selon sa taille. J'ai donc ajouté un message pour mettre l'utilisateur en attente.



Quand les données du fichier sont prêtes à être envoyé au Back-end, la barre de chargement apparaît. L'implémentation de la barre de progression a été pensé par le responsable Back-end. Je devais cependant changer les différents textes qui apparaissent autour de la barre de chargement comme suit :

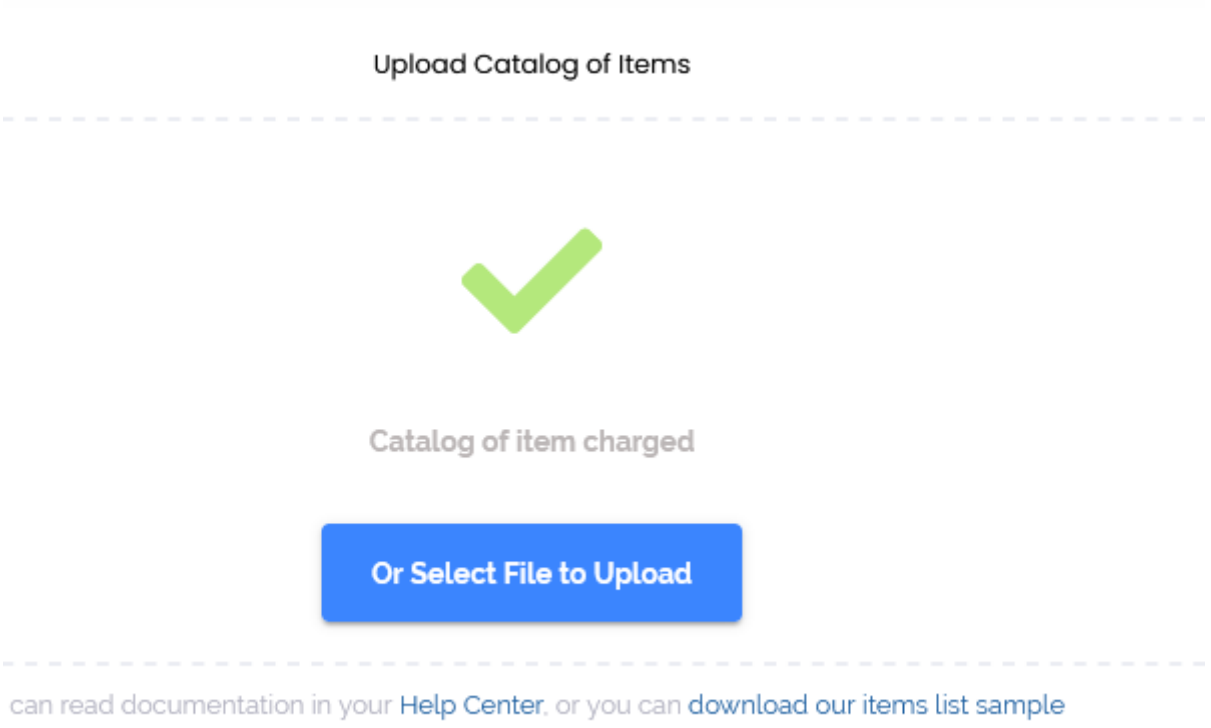


J'ai ajouté le nom et taille du fichier et le temps écoulé et estimé. Ces données proviennent du Back-end et sont envoyés dès le chargement des premières données du fichier.

J'ai aussi ajouté le bouton cancel qui permet d'annuler le téléchargement.

J'ai aussi ajouté une temporisation d'une seconde quand la barre de chargement atteint les 100%, ce qui permet d'avoir plus de clarté sur la transition d'affichage et pour que l'utilisateur peut bien comprendre que la barre de chargement s'est remplie et que le chargement est terminé.

Enfin, le système montre l'affichage finale suivant :



L'icône vert « check » n'est pas une image stockée dans le code source. Il s'agit de la balise <i> qui permet de récupérer les icones que Google met à disposition en

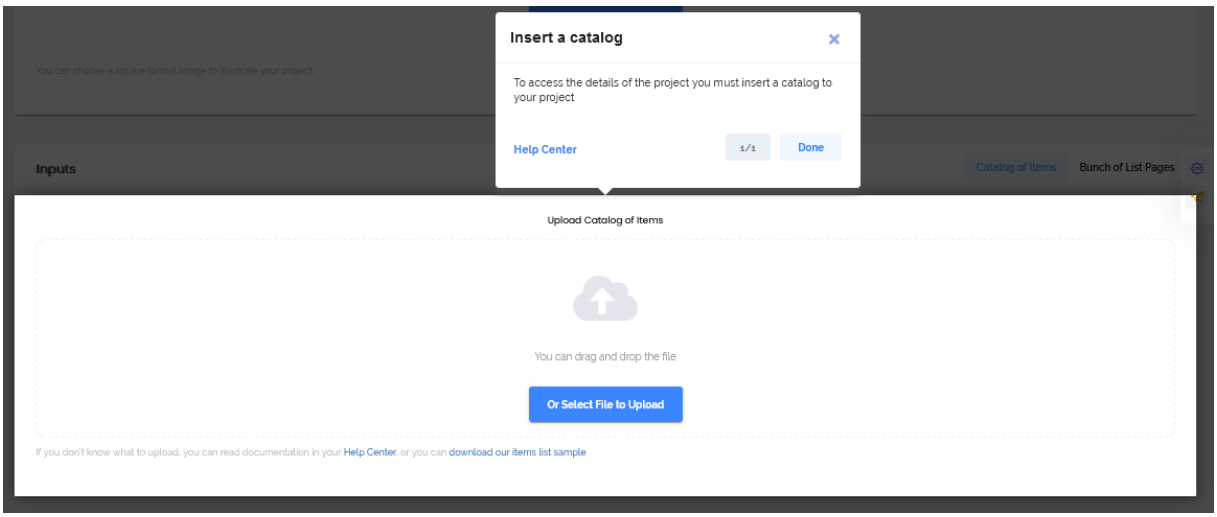
spécifiant par la classe. On a alors <i class= « fas fa-check »></i>. Il est ensuite possible d'ajouter du style CSS pour changer notamment la couleur.

3.1.2. Edit Project

3.1.2.1. D'une notification à une affichage assistant

L'utilisateur doit être averti quand son projet n'a pas de catalogue. Quand il va sur l'édition de projet, le système scrolle vers la partie input puis affiche une notification en rouge en bas à droite de l'écran pour informer l'utilisateur qu'il doit charger un catalogue sur son projet pour voir les détails.

Ma tâche consistait à changer cette notification par un affichage du même type que celui de l'assistant, donc en utilisant la librairie « ngx-guided-tour ». Voici le résultat :



Il fallait donc initialiser un « guided tour » et de le paramétrer pour avoir une seule étape sur « input catalog ».

```
//edit assistant
public editNoCatalogTour() {
  const language = this.cookieService.get('language');
  const stepTransltation = this.guidedTourAssistantTranslationService.getTourTranslation(language)
  this.guidedTourService.startTour({
    tourId: 'edittour',
    useOrb: false,
    steps: [
      {
        title: stepTransltation.editProject.stop0.title,
        selector: '#catalog-block .mat-card-content',
        content: stepTransltation.editProject.stop0.content,
        orientation: Orientation.Top,
        action: () => {
          //permet de scroller quand l'assistant est en cours
          const body = document.querySelector("body") as HTMLElement;
          body.style.overflow = "scroll";
        }
      }
    ],
  });
}
```

L'attribut « steps » ne comporte que d'une seule étape (un seul objet) et que l'utilisateur peut scroller la page (même fonctionnalité que dans la page « New Project »).

3.1.2.2. D'un modale à une page

L'édition d'un projet était accessible depuis un modale qui se trouve dans All Project. Ce choix a été fait pour gagner du temps sur le développement. Ma tâche consistait donc à transformer ce modale en une page pour que l'utilisateur puisse y accéder en passant par l'URL.

Pour pouvoir accéder à une page par une url, il faut créer une route en paramétrant le fichier de module.

```
      ).then((m) => m.AllProjectsModule),
  },
  {
    path: 'edit-projects/:projectId',
    loadChildren: () =>
      import(
        './dashbord/components/projects/edit-projects/edit-projects.module'
      ).then((m) => m.EditProjectsModule),
  },
  {
    path: 'new-project',
    loadChildren: () =>
      import(
```

Fichier dashbord.module.ts

L’identifiant du projet va se trouver dans l’url et il deviendra un paramètre que l’on pourra récupérer pour lancer les requêtes au Back-end. J’ai utilisé un fichier de module enfant (EditProjectsModule), ce qui permettra d’avoir un code plus extensible dans le cas où l’on souhaitera ajouter des fonctionnalités qui concernent l’édition de projet.

```
],
imports: [
  ProjectsModule,
  MatAutocompleteModule,
  RouterModule.forChild([
    { path: '', component: EditProjectsComponent }
  ])
],
```

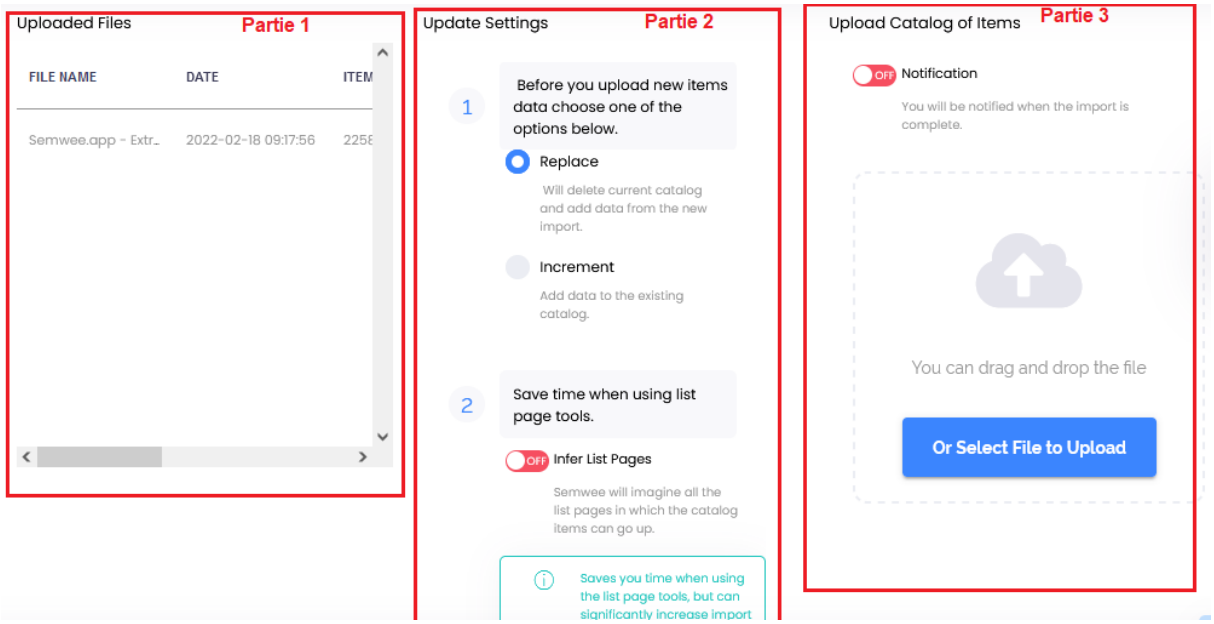
Fichier edit-projects.module.ts

Il fallait ensuite réécrire le component « Edit Projects » pour qu’ils puissent envoyer au component enfant « Add-or-edit » les informations dont il a besoin car les données concernant le projet ne sont plus envoyées à l’ouverture du modale. Il fallait donc récupérer l’identifiant qui se trouve dans l’url et faire les requêtes vers le Back-end.

La transformation du component « Edit Projects » de modale en page a permis de supprimer un bug d’affichage dans l’étape 1 d’un projet qui possède déjà un catalogue.

3.1.2.3. Les différentes étapes de mise à jour de catalogue

La partie input de catalogue sur la page d’édition de projet comporte trois parties : le tableau et les options de chargement et le choix du catalogue :



La partie 1 s'agit d'un tableau qui comporte l'historique des chargements de fichiers dans le projet.

L'une de mes tâches consistait à mettre à jour son affichage pour qu'il concorde à celui de la maquette. Cependant, il y avait un bug quand il était en modale. Le problème étant il récupérait le CSS du « LPNewValidator » et donc changeait de mise en forme si l'utilisateur était passé avant par cette page sans rafraîchir. C'est un problème d'Angular où les modales peuvent récupérer le CSS des autres components. C'est pourquoi, j'avais la consigne de changer le HTML du tableau pour qu'il soit exactement pareil à celui de « LPNewValidator ». Et ainsi, le CSS du tableau ne sera pas influencé car ils auront le même.

Quand le component « Edit Project » est passé de modale à une page. Il était possible de mettre le CSS que l'on souhaite. Le travail qui a été fait (Copier exactement le tableau de « LPNewValidator ») n'était pas inutile. En effet, les deux tableaux se ressemblaient. Une petite mise à jour du CSS a suffi pour correspondre le visuel à celui de la maquette (changement de couleur, réduction de la taille des cellules et suppression d'une bordure). De plus, j'avais déjà implémenté la fonction de changement de la taille des colonnes par la souris, une requête demandée à la suite par mon responsable.

La partie 2 et 3 permettent de mettre à jour le catalogue du projet. L'étape 2 permet de saisir le type de mise à jour que l'utilisateur souhaite faire. Il y en a deux : « Replace » et « Increment ». « Replace » va supprimer les items dans le catalogue pour les remplacer par ceux du nouveau. « Increment » va ajouter les items au catalogue. Il y a aussi deux types de « Increment » qui va gérer les items qui seront dupliqués : « Last Imported Item » (garder le dernier item importé) et « Most Recent Item » (garder le plus récent déterminé par une colonne Date).

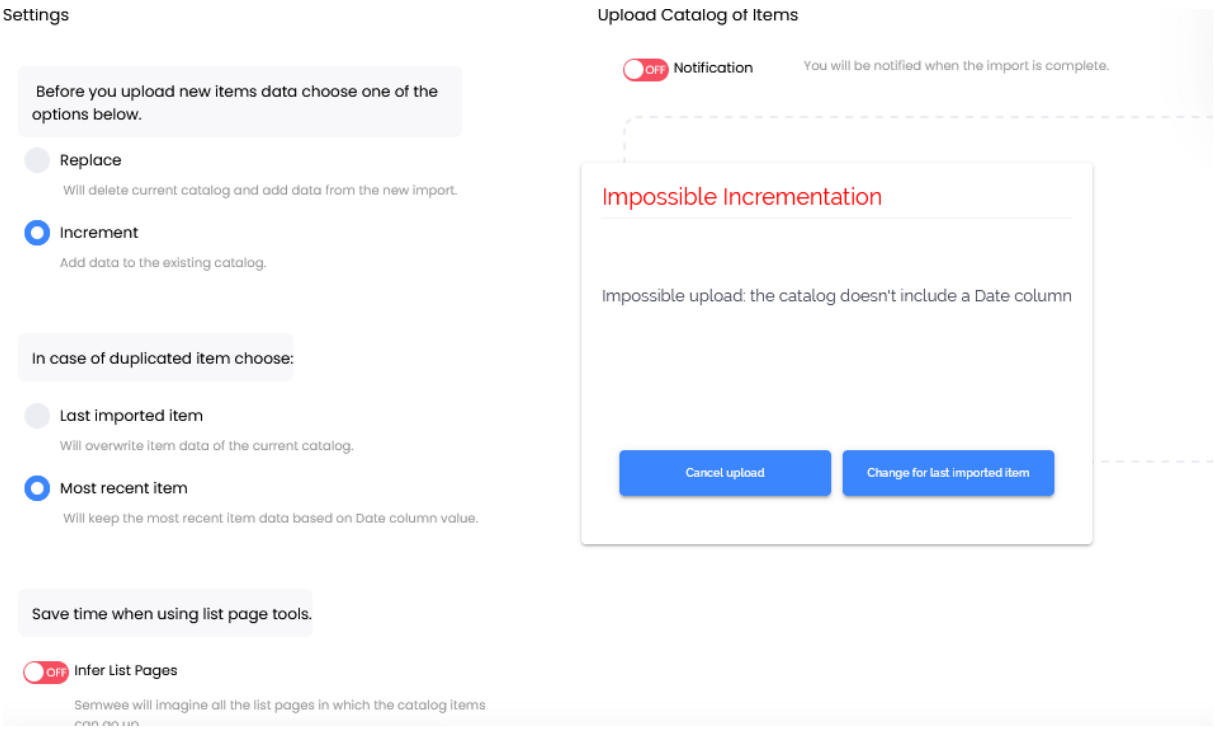
Ma tâche consistait changer le visuel statique en dynamique : Les options de duplication doivent apparaitre seulement pour « Increment ». Cela était très facile car

l'étape 2 utilisait un Angular matériel qui dynamiser les numéros d'étapes. En mettant une condition sur l'une des étapes, les numéros se mettaient à jour automatiquement.

Je devais ensuite implémenter le « Replace » et « Increment » pour que la mise à jour se fasse dans la base de données. Pour cela, j'ai dû changer la base de données en ajoutant un attribut à la collection des produits : j'ai ajouté « data_file_id », l'identifiant des détails du téléchargement. Cela a permis de différencier les produits déjà présents de ceux des nouveaux. Ainsi, on peut savoir quels produits doit être supprimer (selon le type de suppression de duplication voulu). Ensuite, cela a permis de régler un bug. Quand l'utilisateur charge un catalogue, les changements doivent se faire seulement quand l'utilisateur clique sur « Update ». J'ai donc ajouté une requête au moment où l'on quitte la page ou quand l'on charge un nouveau catalogue, les nouveaux items sont supprimés et cela permet d'avoir une base de données correcte.

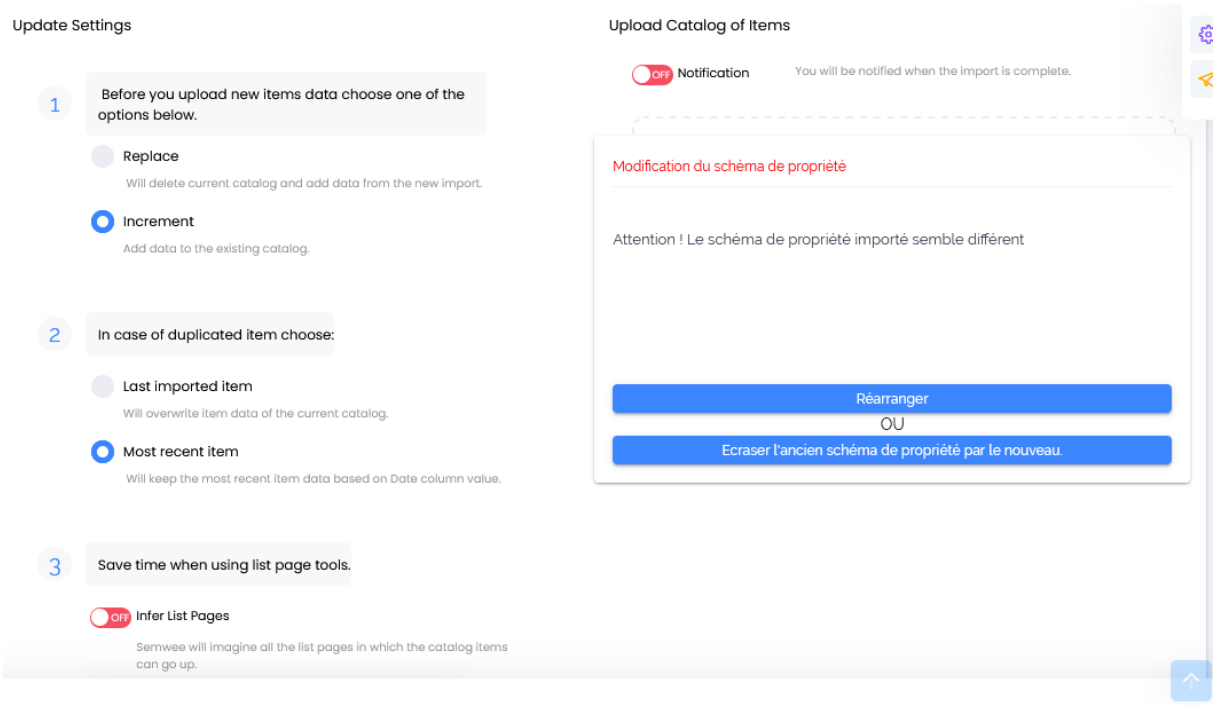
Enfin, la partie 3 est le bouton où l'on entre le fichier. Il est similaire à celui de « new Project ». Ce qui différencie, c'est qu'il n'y a pas la fenêtre intermédiaire qui demande le « InferListPage/Notification ».

En mode « Increment » « Most Recent Item », il doit vérifier si le nouveau catalogue contient une colonne Date. Si ce n'est pas le cas, il va demander à l'utilisateur d'annuler l'import ou changer le « Most Recent Item » pour « Last Imported Item ». Pour cela, j'ai ajouté une condition à l'entrée du back pour la détection de la colonne Date et changer la réponse pour prendre en compte le retour si la colonne Date est bien présente. S'il n'est pas présent, il va afficher un modale et agir selon le choix de l'utilisateur. Le modale est un component enfant. Il est similaire et fonctionne de la même manière que celui de la fenêtre intermédiaire « InferListPage/Notification » de « New Project ».



Il y a aussi un autre cas qui peut poser un problème dans l'algorithme de l'application s'il n'est pas traité. Il s'agit du schéma de propriétés qui change dans le cas où le nouveau catalogue a un schéma différent en « Increment ». En effet, il faut que chaque type de « facet » de chaque produit par catégorie soit le même.

Il fallait donc détecter si le schéma de propriétés est différent et si c'est le cas, demander à l'utilisateur s'il veut réarranger ou écraser l'ancien schéma de propriétés par le nouveau. « Réarranger » va ajouter les nouvelles facets à la suite des anciennes facets. Dans le cas du second, une nouvelle fenêtre s'ouvre pour que l'utilisateur puisse confirmer car cette action entraine la perte de données.



J'ai donc dû, à la suite de la vérification de la présence de la colonne Date en mode « Increment » « Most Recent Item », faire la détection du schéma de propriété. Les items sont alors entrés normalement mais quand l'utilisateur clique sur update, une requête supplémentaire vers l'api est lancée. Il va modifier les données avant de supprimer les items dupliqués.

J'ai rencontré plusieurs problèmes à ce moment-là. En effet, je devais comprendre comment fonctionne l'upload de catalogue. J'ai remarqué que l'application créer trois types de données : « products2 », « products » et « facet1 ». J'ai dû contacter le responsable du Back-end pour qu'il puisse m'expliquer comment fonctionne la gestion des produits dans la base de données. « products » et « facet1 » sont des données qui ne sont plus utilisés car ils s'agissaient de l'ancien modèle de la base de données et qu'avec product2, qui contenaient les produits et leurs facets en même temps, cela optimisait et facilitait la gestion des données. J'ai donc avec son accord, supprimer la création de ces données qui étaient désormais inutile.

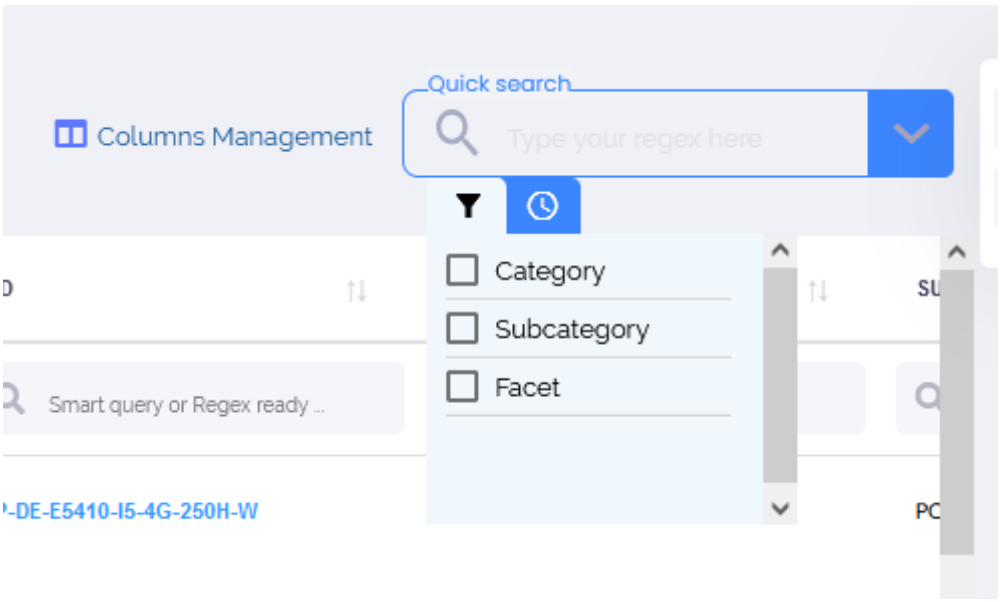
3.2. Quick Search dans LPNewValidator

L'outil LPValidator permet à l'utilisateur de valider en 3 étapes la liste des catalogues qu'il pourra générer. Actuellement, seulement les deux premières étapes ont été implémenter : récupérer la liste de produits d'un projet dans un tableau et créer la liste des catalogues selon les produits. Dans la chaque étape, l'utilisateur a la possibilité de décocher une ligne du tableau pour ne pas le prendre en compte dans pour le traitement. Chaque étape sont des tableaux qui affichent le résultat du traitement fait entre chaque étape.

Le component s'appelle « LPNewValidator » car il s'agit du nouvel outil va remplacer l'ancien.

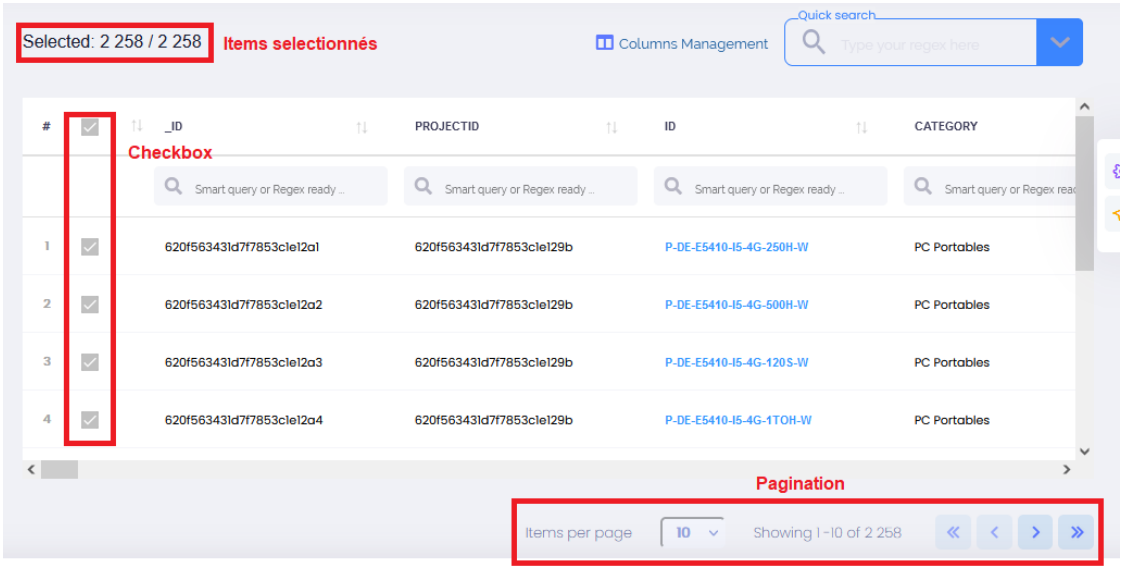
3.2.1. Visuel du Quick Search

LPNewValidator dispose d'un système de filtrage qui va permettre à l'utilisateur de trouver des produits selon plusieurs critères. La fonctionnalité était déjà intégrée, ma tâche consistait donc à modifier le design pour qu'il correspond à celui de la maquette en changeant le CSS.



3.2.2. Pagination avec le Quick Search

Voici le visuel de l'étape 1 :

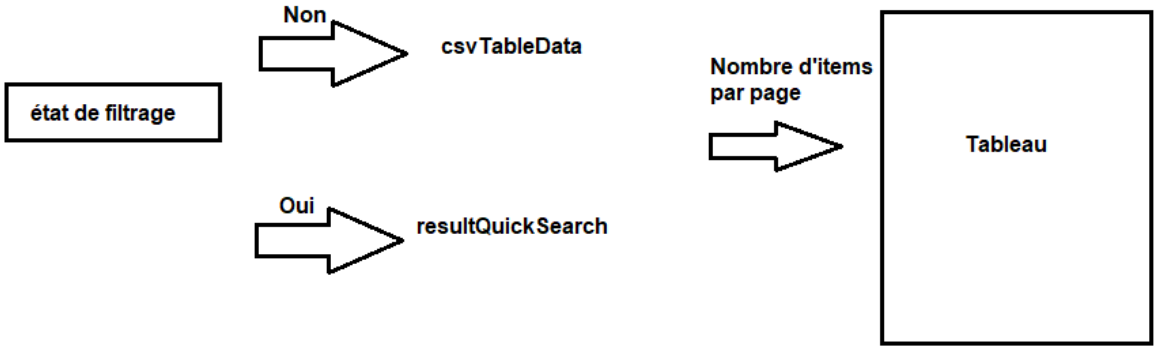


La pagination du tableau du LPNewValidator a déjà été implémenté : On peut avoir le nombre d'items par page, changement de pages… Cependant, ces fonctionnalités de la pagination ne fonctionnent pas avec les produits filtrés par le Quick Search. Par exemple l'ensemble des résultats est affiché (le nombre d'item par page n'est pas respecté).

La récupération des résultats se fait au Back-end et a été implémenté par un autre stagiaire. La recherche dans le Back-end a été optimisée mais le Front-End prend du temps à afficher les résultats.

Ma tâche consistait donc à intégrer les paginations avec les produits filtrés et d'optimiser l'affichage du résultat. J'ai constaté que l'affichage du Front-end prenait du temps car il devait afficher beaucoup de produits d'un coup. En effet, la fonction de filtrage affichait directement le résultat. Je me suis donc d'abord occupé de la pagination qui va afficher seulement 10 items (par défaut) par page.

Pour cela j'ai créé deux états : Affichage avec et sans filtrage. La gestion du choix de la liste de produits suit la logique du schéma suivant :



« csvTableDate » est la liste qui contient l'ensemble des items du projet et « resultQuickSearch » est la liste qui contient le résultat du filtrage.

De plus, il fallait aussi faire en sorte que les fonctionnalités du tableau fonctionnent avec le « Quick Search ». Chacun est une fonction dans le Typescript du component : les checkbox (Si l'une des deux listes change, le second doit aussi changer), les doubles flèches pour aller en début ou fin de page et l'affichage du nombre de résultats (doit prendre en compte liste actuel selon filtrage ou non).

3.3. Loading

Quand un traitement prend du temps et qu'il a besoin de faire attendre l'utilisateur, une surcouche apparait avec un message de Loading.

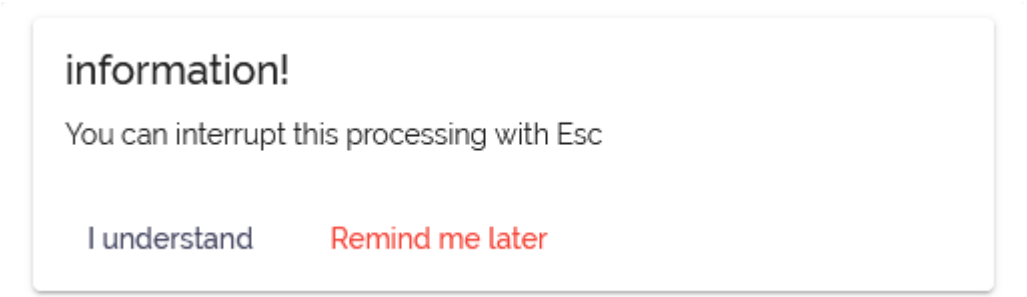
Pour gérer l'apparition du Loading, le projet utilise la librairie « ngx-spinner »

3.3.1. Fenêtre d’information pour le bouton Echap

L’utilisateur a la possibilité d’appuyer sur la touche Echap de son clavier pour annuler le Loading.

Après 10 secondes sur la page du Loading, un modale apparait en bas de l’écran pour informer que l’utilisateur a la possibilité d’appuyer sur Echap.

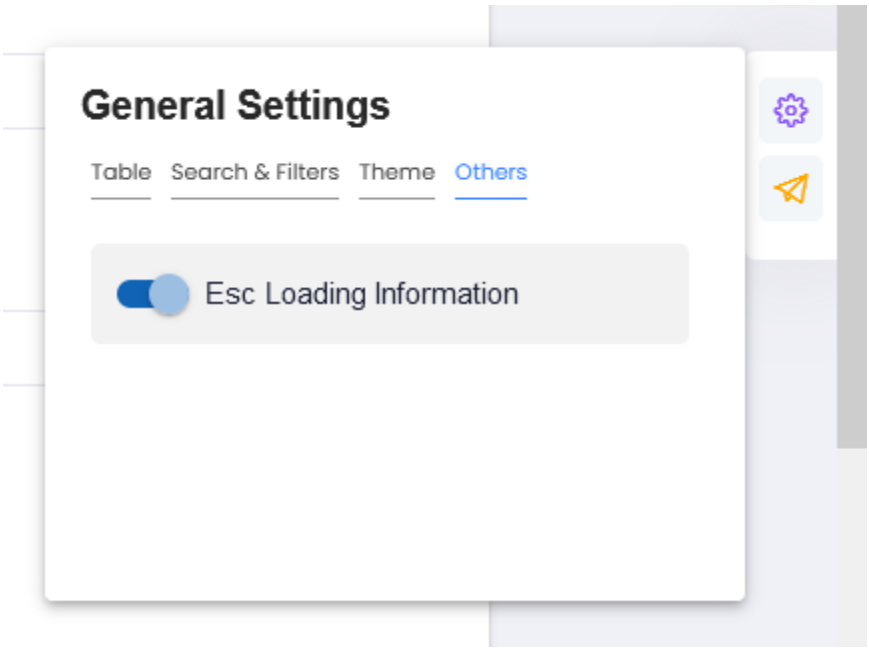
Ma tâche consistait à ajouter deux boutons sur ce modale pour que l’utilisateur ait la possibilité d’avoir deux choix : « me le rappeler plus tard » ou « j’ai compris ».



Quand l’utilisateur clique sur « j’ai compris », le modale d’information n’apparaitra plus pour informer l’utilisateur. Il s’agit d’un paramètre utilisateur qui est stocké dans la base de données. C’est d’un attribut de type Boolean et unique à l’utilisateur. Il est défini activé par default à la création du compte.

Quand l’utilisateur clique sur « j’ai compris », le paramètre est désactivé par une requête vers l’api.

3.3.2. Nouvelle configuration affichage utilisateur : « ESC Loading Information »



.

Un component présent sur toutes les pages comme le header ou la Side-bar se trouve sur la droite de l'écran. Il s'agit de la Tool-bar qui contient les paramètres de l'utilisateur.

Ma tâche consistait à ajouter un nouveau paramètre qui correspond à l'affichage ou non du modale d'information de la touche Echap quand l'utilisateur se trouve sur le Loading via un toggle. Il a donc la possibilité de modifier cette valeur directement depuis cette interface.

Quand l'utilisateur clique sur le toggle, une requête est lancée vers l'api pour changer la donnée dans la base et le cookie qui contient les paramètres de l'utilisateur se met à jour. Il est plus avantageux de stocker les informations dans les cookies car cela permet d'éviter de faire trop de requête vers le Back-end et éviter de ralentir l'application quand on souhaite juste accéder à la valeur de la donnée.

3.4. Autres

3.4.1. Optimisation du site : Les paramètres d'affichage de l'utilisateur

Il ne s'agit pas d'une tâche que l'on m'a confié. En effet, durant le stage, j'ai constaté un grand ralentissement de navigation du site sur certaine page où les performances du navigateur sont très demandées. Après plusieurs recherches, j'ai constaté que l'application utilisait des « ngDoCheck » dans plusieurs components

présents en même temps sur la page et que cela pouvait ralentir les applications Angular. « ngDoCheck » est une méthode d'Angular dans les components qui se lancent suite à divers évènements. Il existe d'autres méthodes du même type et sont appelés Hook cycle. De plus sur plusieurs components, il s'agissait de la même fonctionnalité : lire les paramètres utilisateurs dans l'url. A chaque interaction avec la page, l'application mettaient à jour les components et leur affichage. J'ai donc proposé à mon responsable de changer le système pour stocker l'ensemble des paramètres dans les cookies, système de stockage Front-end.

Mon idée était de, au lieu de vérifier à chaque interaction avec la page les paramètres utilisateurs, modifier l'affichage quand l'un des paramètres est modifié. Pour cela, j'ai utilisé les « subscriptions » de la librairie RXJS.

Les « subscriptions » permettent de relancer les fonctions quand ils sont rappelés. Au lieu de vérifier à chaque interaction avec la page pour voir s'il y a un changement, on change seulement quand il est nécessaire.

La lecture des paramètres de l'url se fait par « subscriptions ». Ainsi, lorsqu'on passe le nouveau paramètre par l'url, la suscription s'active pour mettre à jour les cookies et l'affichage selon les paramètres de l'utilisateur.

```
initQueryParamsStuff() {

  //detecte le changement de parametres
  this.queryParamsSubscription = this.route.queryParams.subscribe(params => {

    this.toggleChecked = JSON.parse(this.cookieService.get('headColor'));
    this.toggleCheckedAlternance = JSON.parse(this.cookieService.get('alterOption'));
    this.toggleCheckedGMK = JSON.parse(this.cookieService.get('gmkOption'));
    this.previousBool = this.toggleCheckedGMK;
    this.reboot();
    this.toggleCheckedSearchMasq = JSON.parse(this.cookieService.get('hideFilter'));
    this.toggleCheckedRegexFilter = JSON.parse(this.cookieService.get('regexFilter'));
    this.toggleCheckedCaseSensitive = JSON.parse(this.cookieService.get('caseSensitive'));
    this.toggleCheckedlightMode = JSON.parse(this.cookieService.get('lightMode'));
```

Par exemple, ici la fonction « initQueryParamsStuff() » n'est appelé qu'une fois dans la fonction d'initiation du component mais sera relancé quand l'application détecte un changement de paramètre dans l'url.

Pour mettre à jour les paramètres d'affichage, il suffisait de changer les paramètres des cookies et les paramètres de l'url pour déclencher cet « subscription ».

Il ne fallait oublier au moment où l'on quitte la page de supprimer la « subscription » :

```
ngOnDestroy(): void {
  console.log('hey');
  this.common.displayEscMessage = true;
  this.queryParamsSubscription.unsubscribe()
}
```

3.4.2. Correction des items du LPNewValidator.

J'ai découvert ce bug quand je testais la création de projet. Lorsque j'avais plusieurs projets qui contenaient un catalogue de plus de milliers de produits, l'accès au LPNewValidator faisait crash mon serveur Back-end à cause d'un manque de mémoire. J'ai alors soumis mon problème dans la conversation Back-end de Skype. En réponse, le responsable du Back-end m'a conseillé de lancer mon serveur Node avec une allocation de mémoire plus grande que celle par défaut.

Commande :

```
# node --max-old-space-size=4096 server.js
```

Cela a bien résolu mon problème mais quand j'ai pu voir le résultat du LPNewValidator, j'ai remarqué le nombre d'item anormalement grand. J'ai ensuite réinitialisé ma base de données.

J'ai constaté que le nombre de produits du LPNewValidator d'un projet correspondait à celui de l'ensemble des projets. Après avoir regardé et compris le code, j'ai compris que l'appellation des variables entre le back et le front était différent. J'ai alors fait part du problème avec mon responsable et j'ai pu ensuite corriger ce problème.

3.4.3. Installer des environnements de développement

Durant le sage, j'ai plusieurs fois, aider à installer les environnements de développements d'autres stagiaires. Cela se traduit par l'explication de l'utilisation du mode SSH pour récupérer le code sur Github ainsi de régler les problèmes d'installation des packages et de lancement des serveurs front et back.

La plupart des problèmes d'installation rencontrés étaient l'installation des packages à cause de conflits entre les versions. Par exemple, pour le front, le CLI Angular du projet est en version 11.X.X, la plupart des stagiaires avaient des version 13.X.X qui étaient incompatibles. Pour régler ce problème, il fallait passer les versions de 13.X.X à 12.X.X. Concernant le back, il s'agissait des versions de Node et de NPM. La solution envisagée était de donner ma version de back pour être certain de la compatibilité.

Il fallait aussi régler les fichiers de configuration des packages après l'installation des packages avec les commandes suivantes :

```
# npm audit fix
```

Puis

```
# npm audit fix --force
```

Conclusion

Durant ce stage, j'ai découvert un environnement de projet dans un environnement professionnel.

Par les différentes activités durant ce stage, j'ai pu renforcer mes compétences en algorithmie (par le Back-end) et en programmation web (par le Front-end) et ainsi approfondir mes connaissances en NoSQL.

J'ai pu alors réaliser plusieurs tâches variées qui m'a permis de consolider mes connaissances en Full-stack Javascript.

Ce stage m'a aussi permis d'acquérir de l'expérience sur Git et l'utilisation de Gitlab.

Annexes

Exemple de produits dans la collection Products2 de la base de données


```
_id: ObjectId("620f5beea7be089970a507e2")
projectId: ObjectId("620f5beea7be089970a507db")
data: Object
    Date: "2022-02-16T23:00:00.000Z"
    ID numÃ©rique: 19279
    ID: "PCK-DE-OPT-3010-DT-PNT-19-4G-500H-W"
    Category: "Ordinateurs de Bureau"
    Subcategory: "PC avec Ã©cran"
    Facet 1: "RAM CapacitÃ©"
    Facet 1 Value: "4Go RAM"
    Facet 2: "DDur capacitÃ© + type"
    Facet 2 Value: "500Go HDD"
    Facet 3: "CapacitÃ©"
    Facet 3 Value: "500Go"
    Facet 4: "Type de disque"
    Facet 4 Value: "HDD"
    Facet 5: "OS"
    Facet 5 Value: "Windows 10"
    Facet 6: "Marque"
    Facet 6 Value: "Dell"
    Facet 7: "Gamme"
    Facet 7 Value: "Optiplex"
    Facet 8: "ModÃ¨le"
    Facet 8 Value: 3010
    Facet 9: "Format"
    Facet 9 Value: "Desktop"
    Facet 10: "Type de processeur"
    Facet 10 Value: "Intel Pentium G Dual Core"
    Facet 11: "GÃ©nÃ©ration du processeur"
    Facet 12: "Processeur"
    Facet 12 Value: "Intel Pentium Dual Core G870"
    Facet 13: "FrÃ©quence du processeur"
    Facet 13 Value: "3,1 GHz"
    Facet 14: "MÃ©moire vive maximum"
    Facet 14 Value: "8Go max."
    Facet 15: "Type de mÃ©moire vive"
    Facet 15 Value: "DDR3"
    Facet 16: "Format du disque dur"
    Facet 16 Value: "3,5""
    Facet 17: "Interface disque dur"
    Facet 17 Value: "SATA"
    Facet 18: "Carte graphique"
    Facet 18 Value: "Intel HD Graphics 2500"
    Facet 19: "Marque de carte graphique"
    Facet 19 Value: "Intel HD Graphics"
    Facet 20: "Wifi"
    Facet 20 Value: "sans Wifi"
    Facet 21: "Bluetooth"
    Facet 21 Value: "sans Bluetooth"
    Facet 22: "Port(s) USB"
    Facet 22 Value: "8x USB 2"
    Facet 23: "Port(s) VidÃ©o"
    Facet 23 Value: "VGA / HDMI"
    Facet 24: "Port VGA"
    Facet 24 Value: "Avec VGA"
    Facet 25: "Port HDMI"
    Facet 25 Value: "Avec HDMI"
    Facet 26: "DisplayPort"
    Facet 27: "Autres prises"
    Facet 27 Value: "1x Ethernet (RJ45), 1x Jack Audio, 1x Jack Micro"
    Facet 28: "Lecteur CD/DVD"
    Facet 28 Value: "avec Lecteur CD DVD"
    Facet 29: "Webcam intÃ©grÃ©e"
    Facet 29 Value: "sans Webcam intÃ©grÃ©e"
    Facet 30: "Taille d'Ã©cran"
    Facet 30 Value: "19""
    Facet 31: "Accessoires livrÃ©s avec l'appareil"
    Facet 31 Value: "1x cÃ¢ble d'alimentation"
    Facet 32: "Dimensions"
    Facet 32 Value: "Hauteur 36 cm x Largeur 10 cm x Profondeur 41 cm"
    Facet 33: "Poids"
    Facet 33 Value: "7,5 Kg"
    Facet 34: "Garantie"
    Facet 34 Value: "Garantie 1 an piÃ¨ces et main d'oeuvre"
    Facet 35: "Tranche de prix"
    Facet 35 Value: "200â¬ - 300â¬"
    Facet 36: "Prix"
    Facet 36 Value: "Moins de 300â¬"
data_file_id: ObjectId("620f5beea7be089970a507dc")
Date: "2022-02-16T23:00:00.000Z"
ID: "PCK-DE-OPT-3010-DT-PNT-19-4G-500H-W"
```