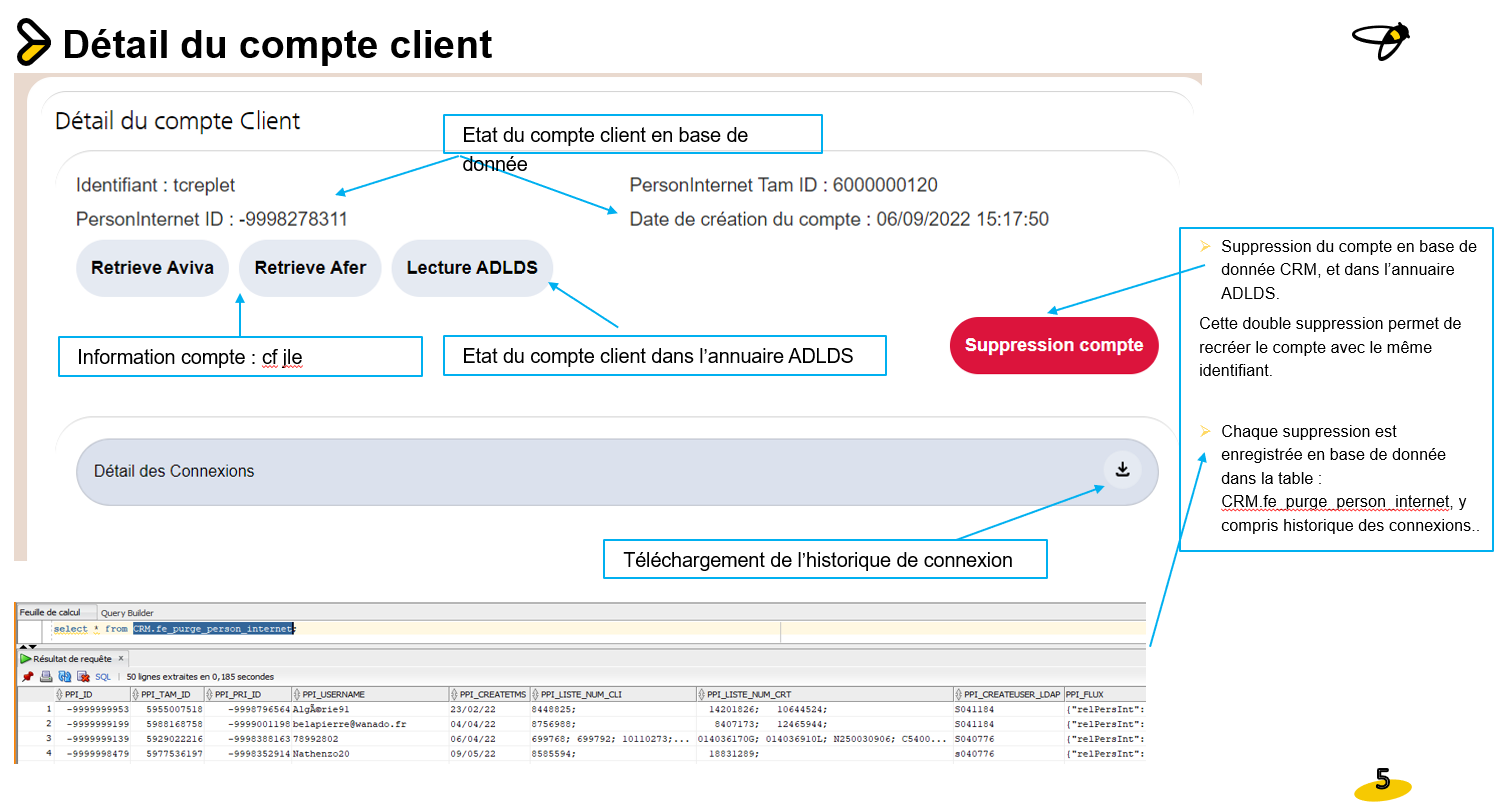
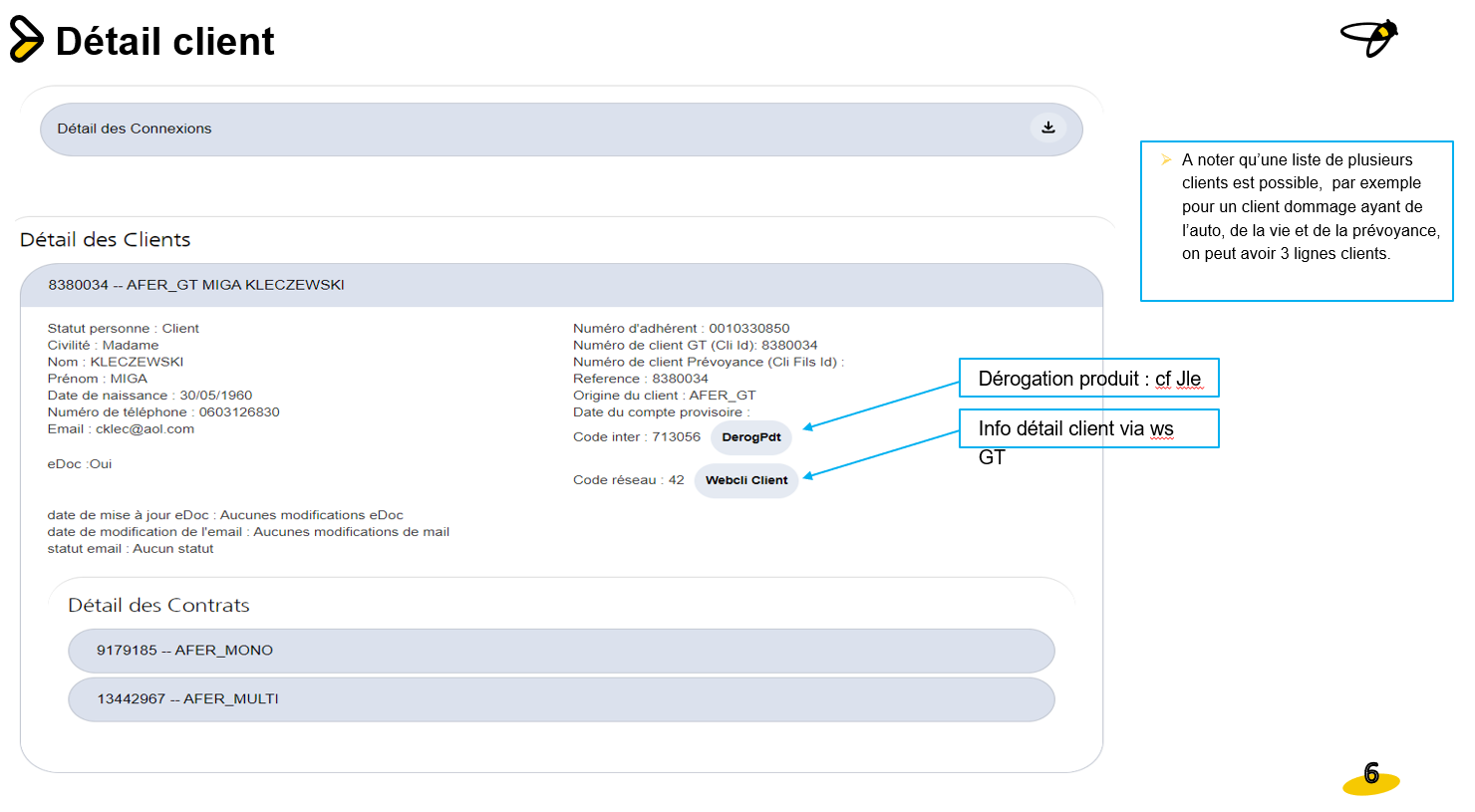
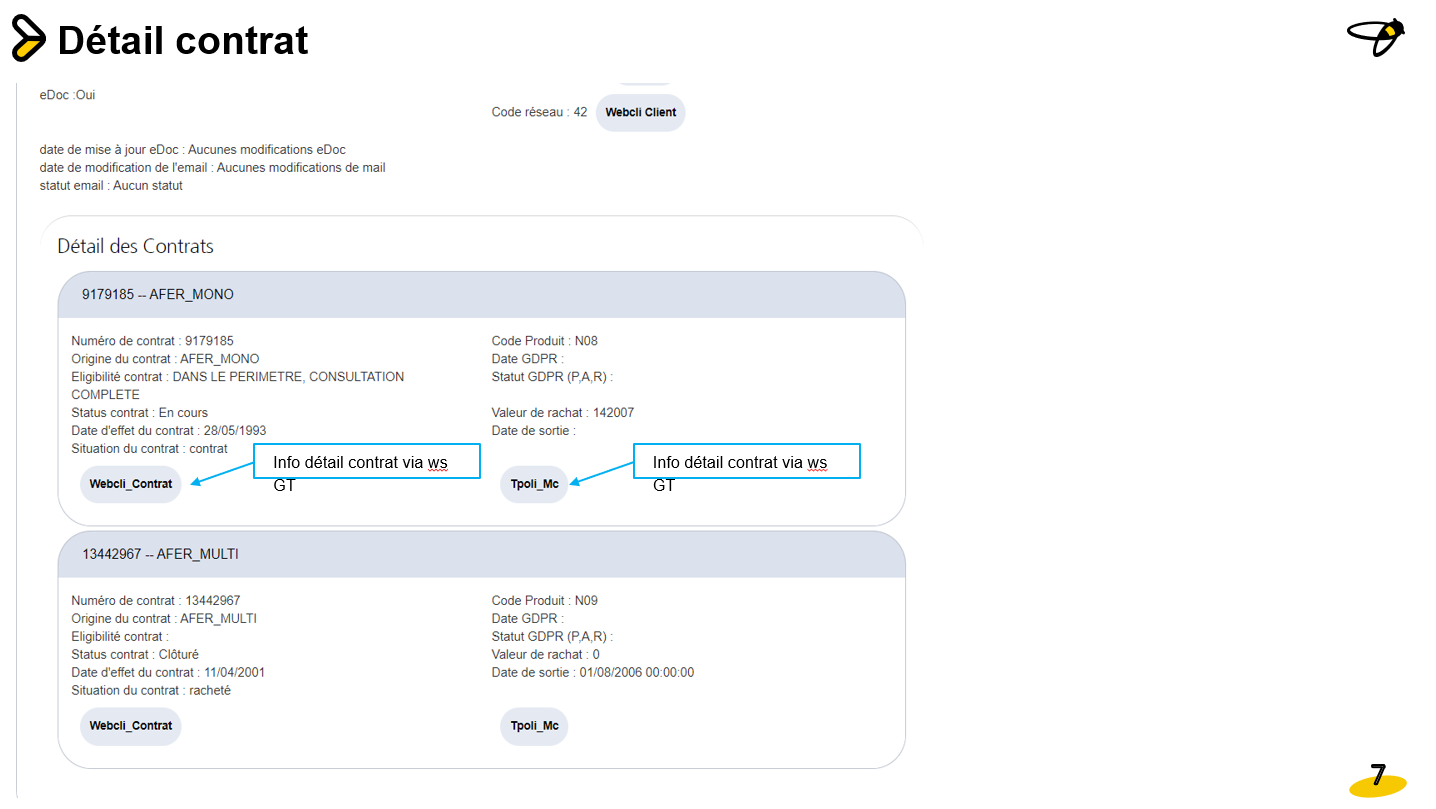
**Documentation de stage**

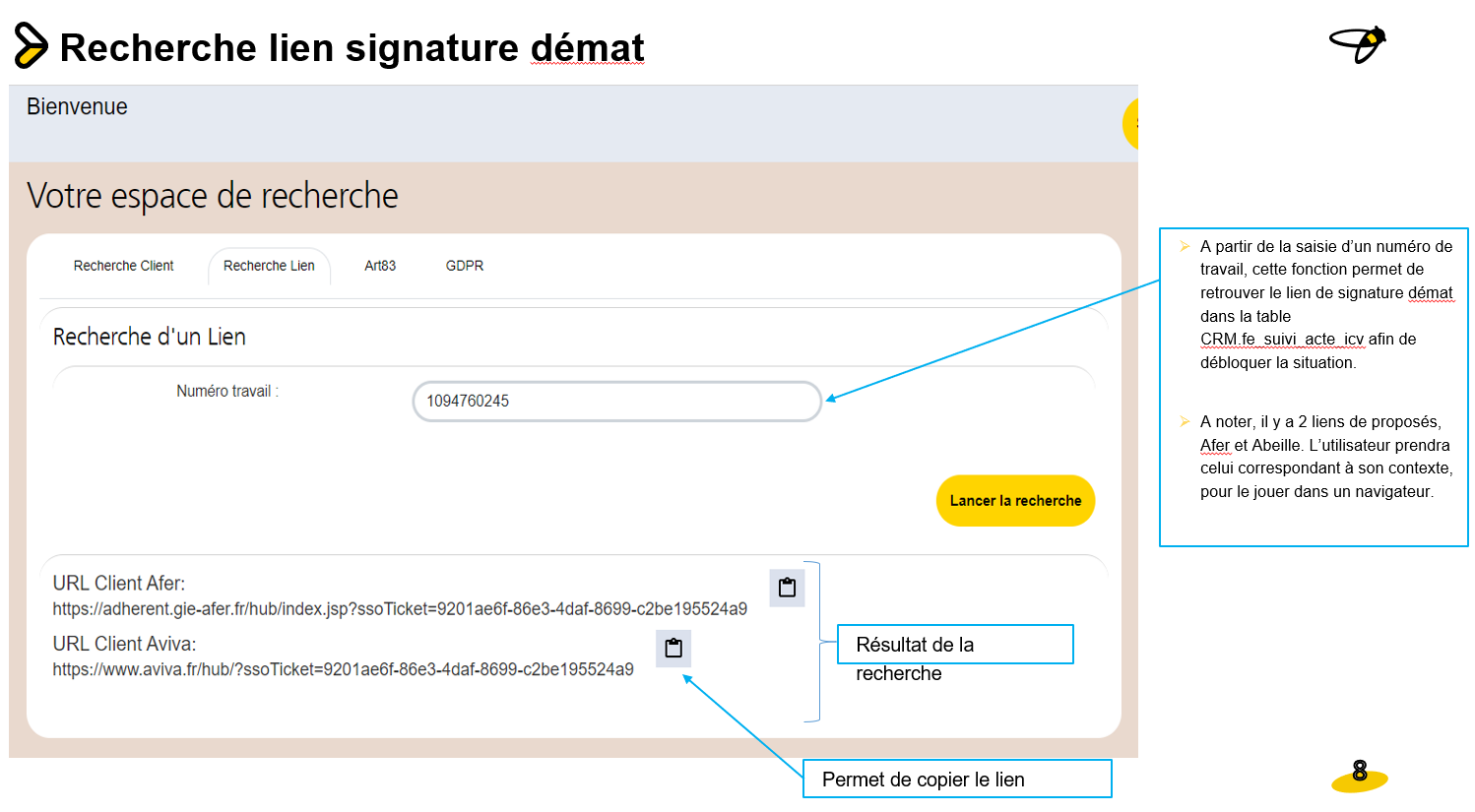


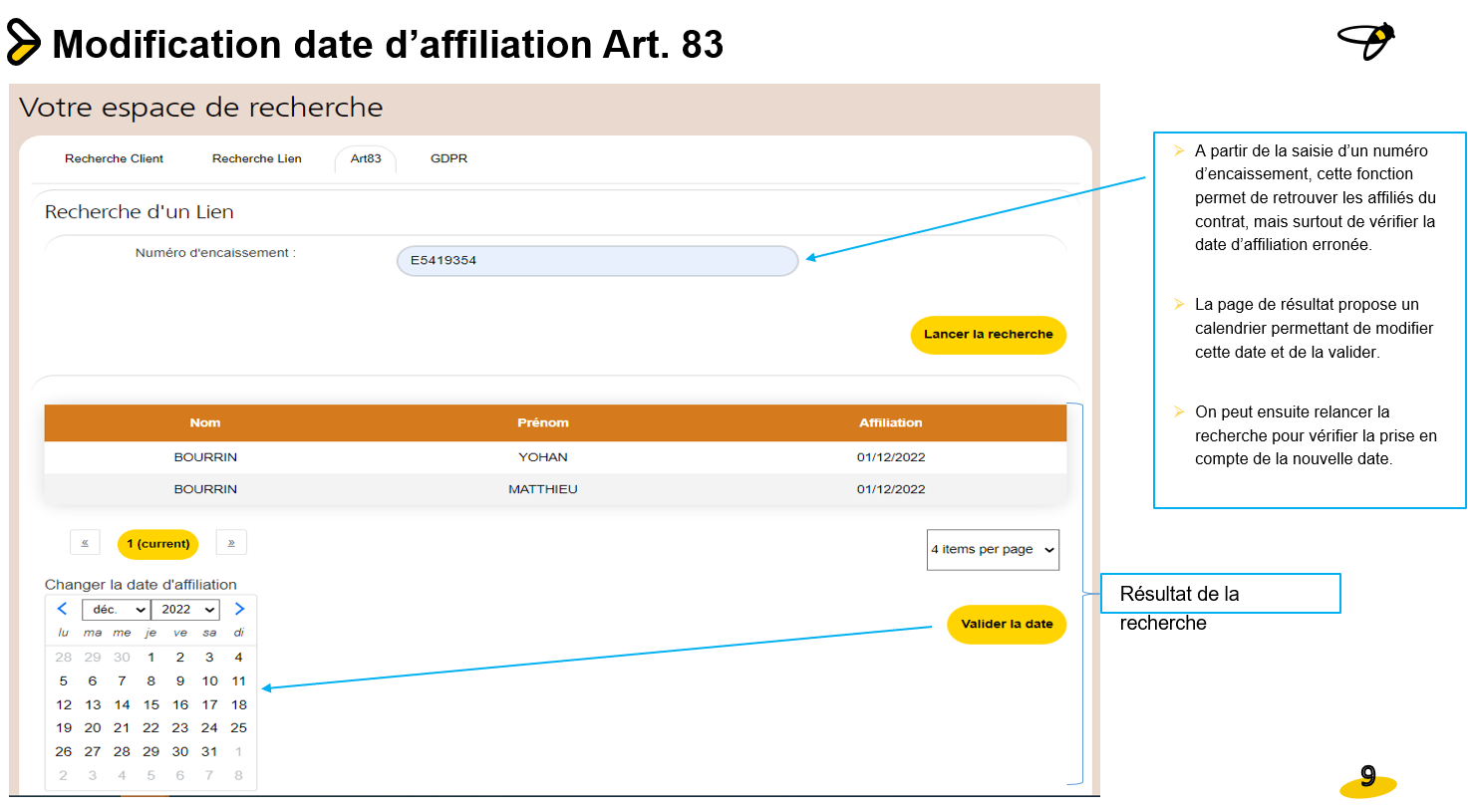


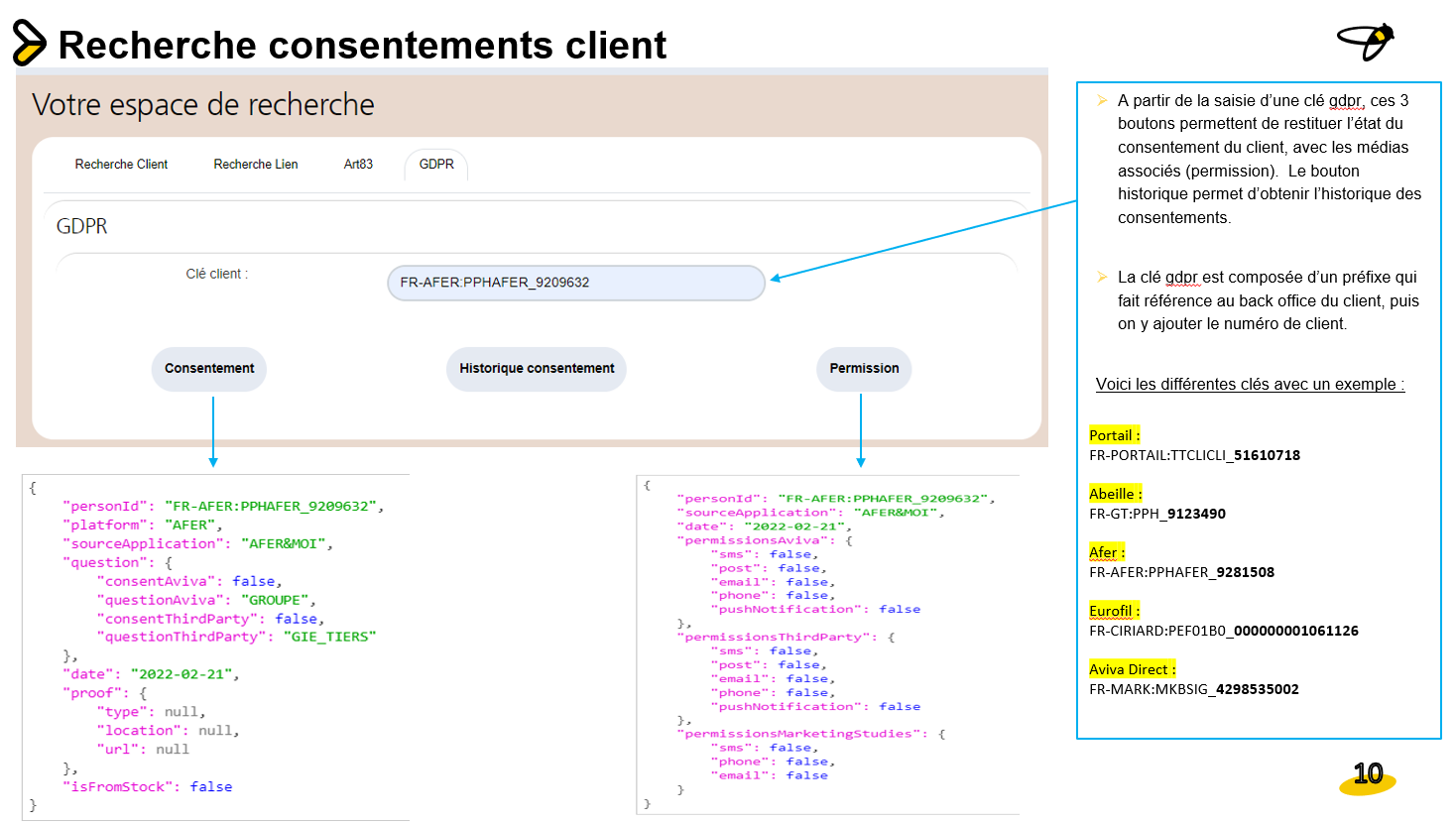








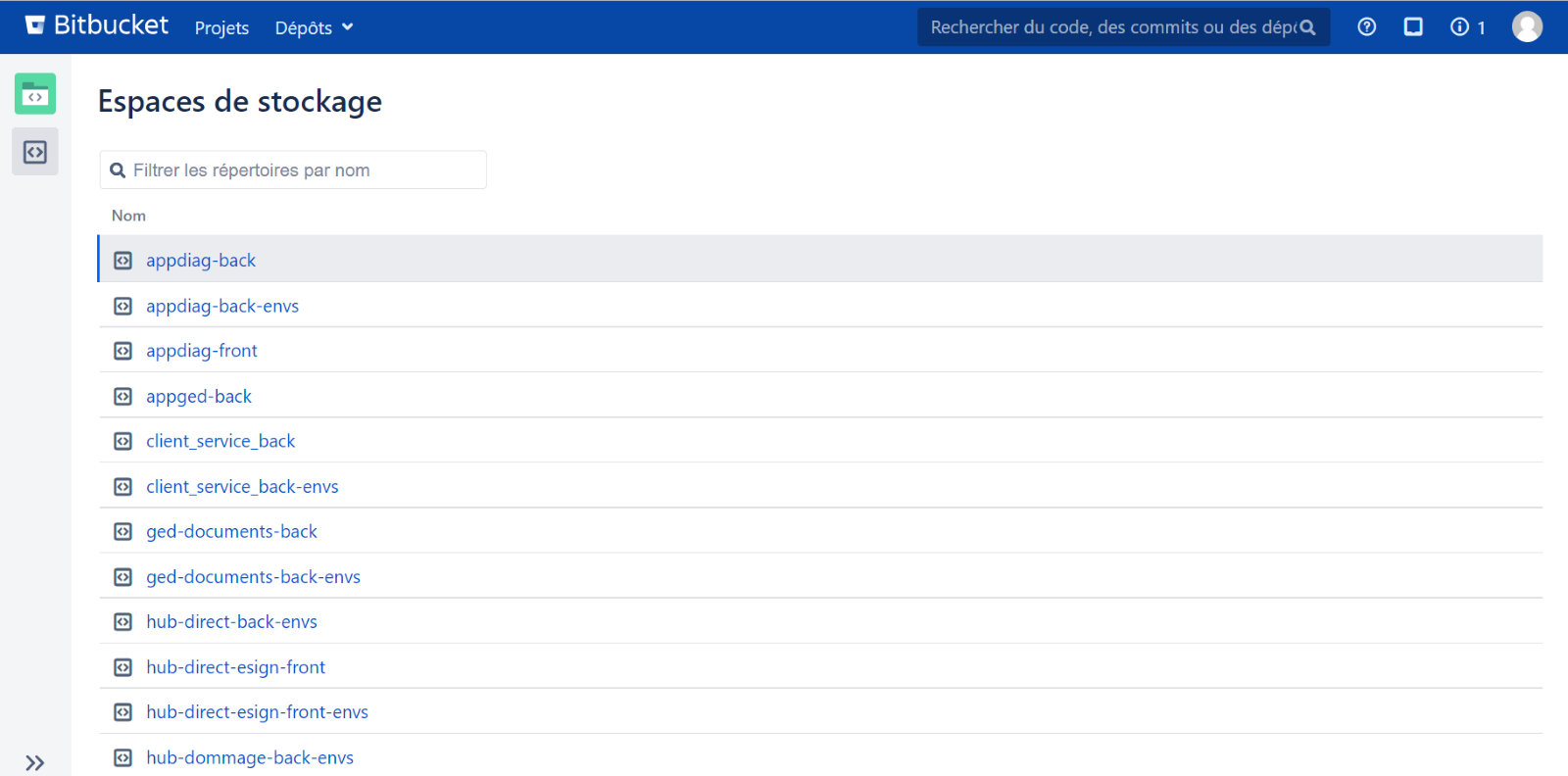




Lors de ce stage, nous aurons accès à certaines plateformes afin de nous diriger ou de nous aider dans nos missions (décrites juste après).





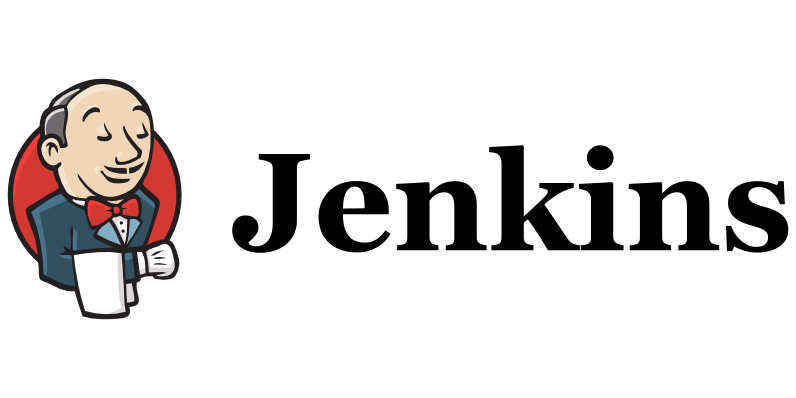


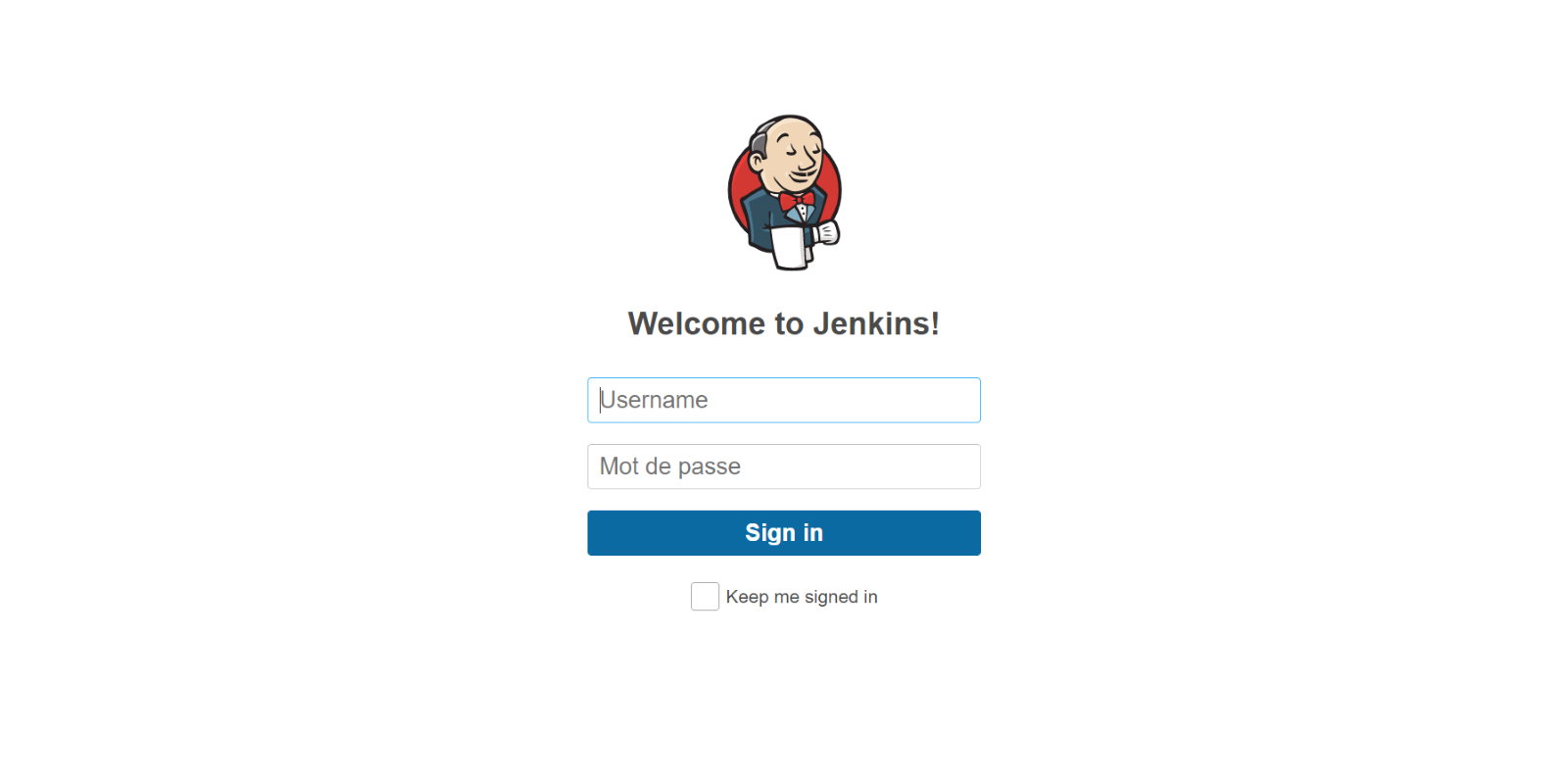
Bitbucket est un service web d'hébergement et de gestion de développement logiciel utilisant le logiciel de gestion de versions Git. Il s'agit d'un service freemium dont la version gratuite permet déjà de créer jusqu'à un nombre illimité de dépôts privés, accessibles par cinq utilisateurs au maximum.

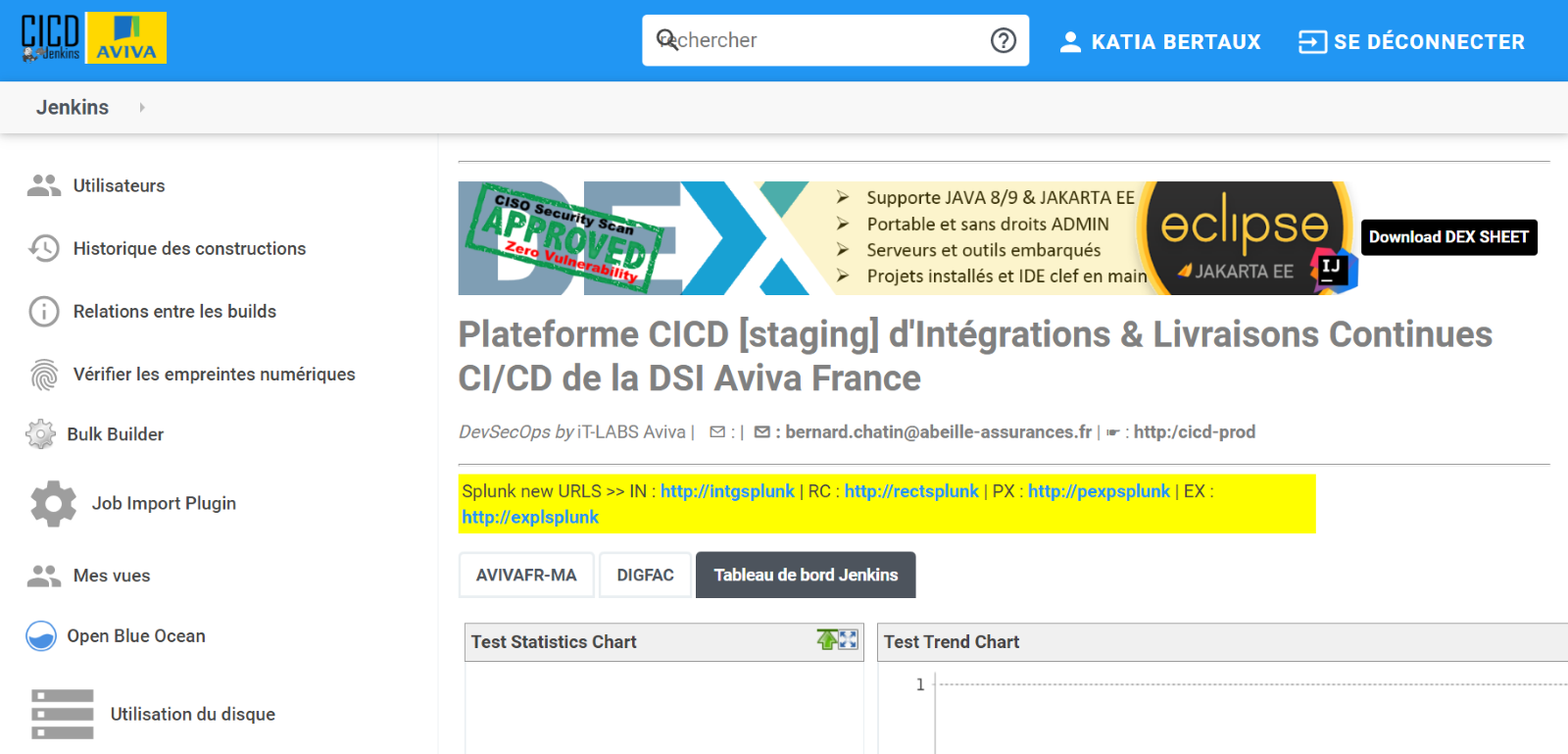
Il permet aux équipes de développeurs de disposer d’une plateforme commune pour planifier des projets, effectuer un travail collaboratif autour du code, mais aussi tester et déployer.

Bitbucket dispose des fonctionnalités suivantes :

* **Collaboration sur le code :** vous pouvez approuver la revue de code avec les pulls requests, créer une liste de contrôle des merges avec des approbateurs désignés, et mener des discussions directement dans le code source.
* **Livraison continue :** vous bénéficiez de la centralisation du développement, des tests et du déploiement avec la CI/CD intégrée ainsi que de la configuration sous forme de code et de boucles de feedback accélérées.
* **DevSecOps :** vous obtenez une visibilité en temps réel sur les problèmes de sécurité dans votre code et vos conteneurs. Vous pouvez identifier les corrections de vulnérabilité aux premiers stades du développement et surveiller les risques après déploiement.
* **Définition des rôles :** vous avez la possibilité de limiter l’accès à des utilisateurs et de contrôler leurs actions grâce aux autorisations de branche.



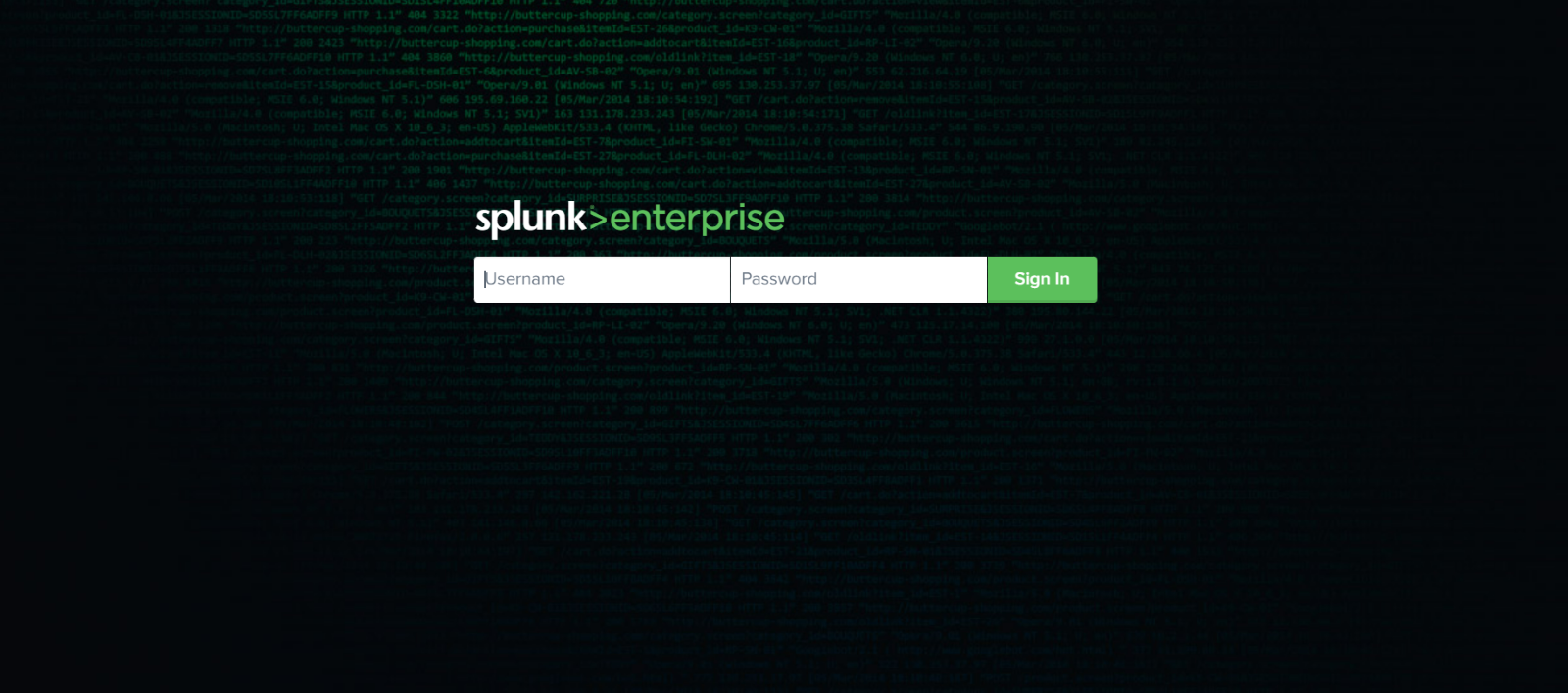


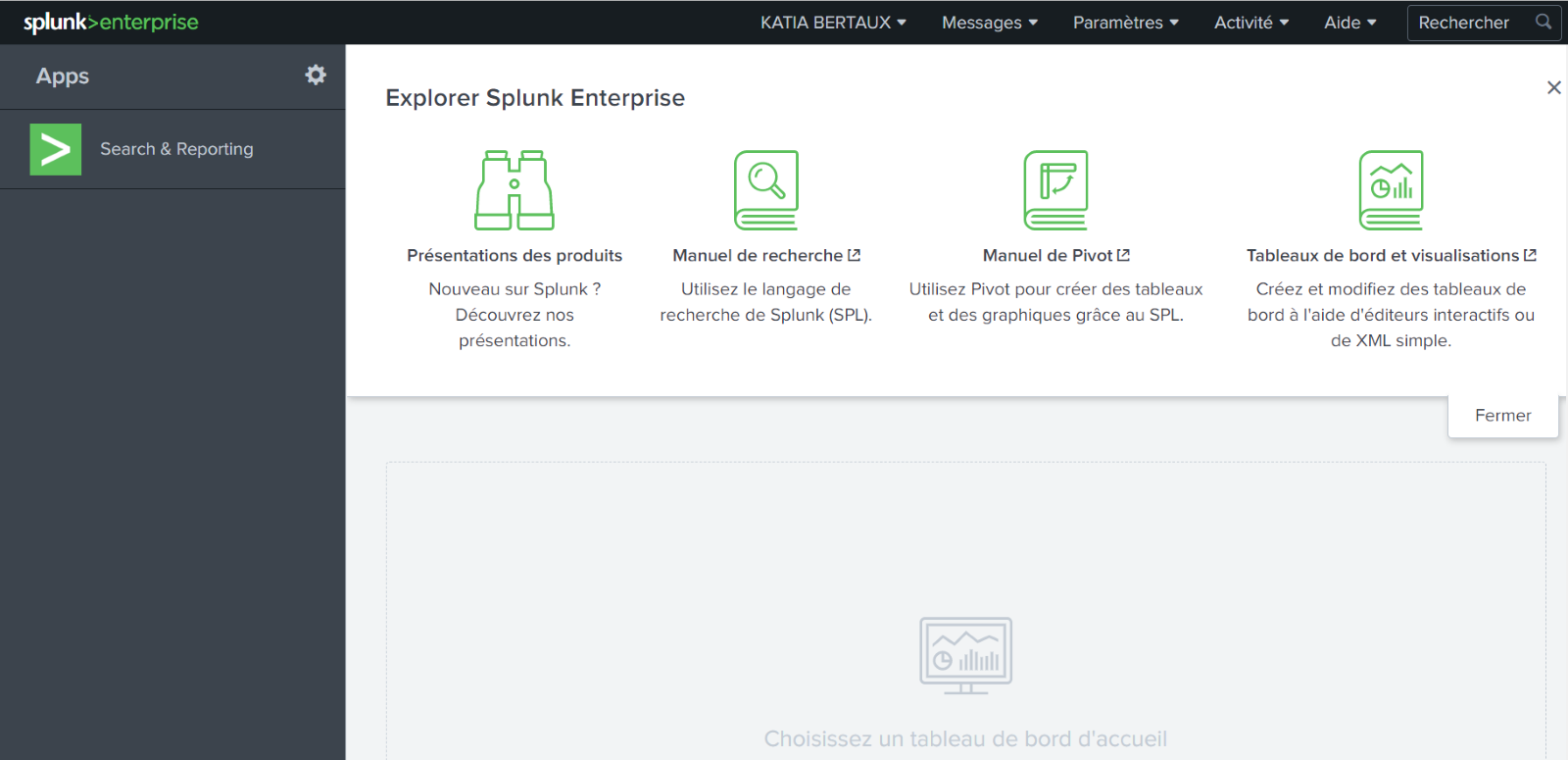


**Jenkins** est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue et la livraison continue.

Développé en Java. A chaque modification de code d'une application dans le gestionnaire de configuration, **Jenkins** se charge automatiquement de la recompiler, et de la tester.



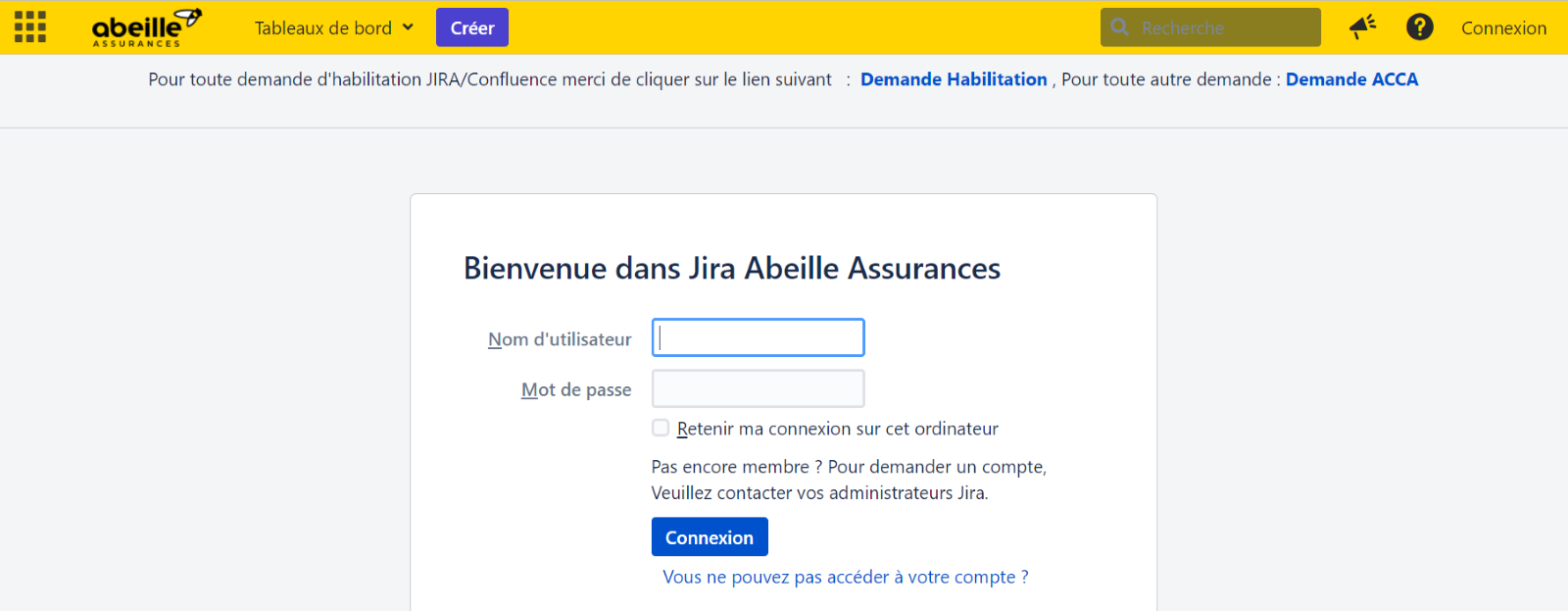


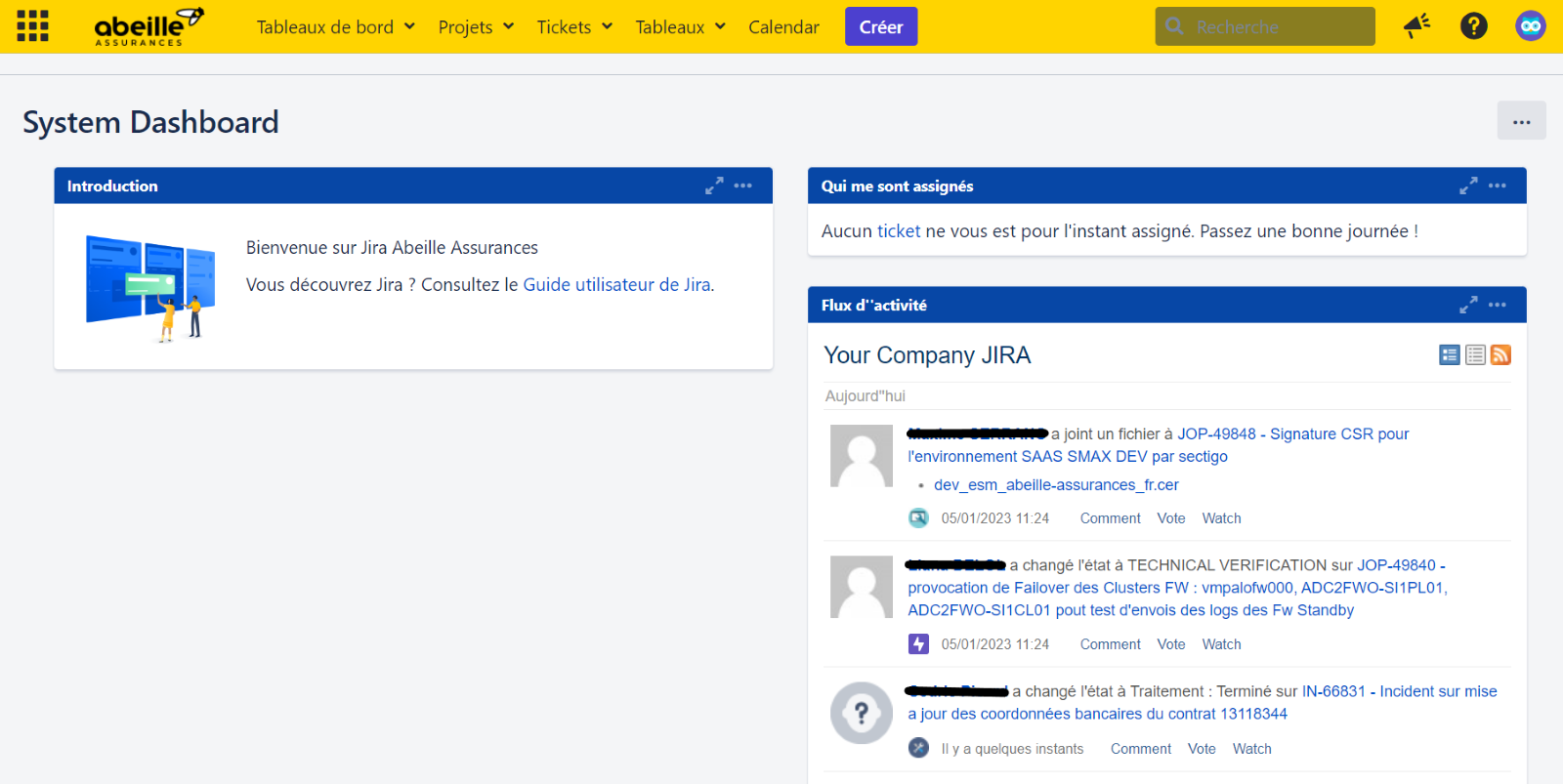


Splunk fait partie des plateformes logicielles largement utilisées pour**surveiller, rechercher, analyser et visualiser les données générées par les machines en temps réel**. Ce logiciel a la capacité de lire diverses informations, qu’elles soient non structurées, semi-structurées ou rarement structurées.

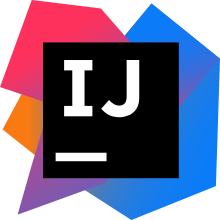
Il **capture, indexe et corrèle les data en temps réel dans un conteneur interrogeable**. Ensuite, cette plateforme produit des graphiques, des alertes, des tableaux de bord et des visualisations. Cette opération permet un**accès facile aux données dans toute l’organisation**. De ce fait, le diagnostic et la résolution de divers problèmes d’entreprise deviennent plus aisés.







À l'origine, **Jira** a été pensé comme un **outil** de suivi des bugs et des tickets. Mais aujourd'hui, **c'est** devenu un puissant **outil** de gestion du travail pour toutes sortes de cas d'usage, de la gestion des exigences et des cas de test au développement logiciel Agile.



**IntelliJ IDEA** est un *environnement de développement intégré (IDE)* pour les langages JVM (Java, Kotlin, Scala et Sensationnel) conçu pour maximiser la productivité des développeurs. Il effectue les tâches routinières et répétitives pour vous en fournissant une complétion de code intelligente, une analyse de code statique et des refactorisations, et vous permet de vous concentrer sur le bon côté du développement logiciel, ce qui en fait une expérience non seulement productive mais aussi agréable.

Un **environnement de développement intégré** (**IDE**) est une application logicielle qui aide les programmeurs à développer efficacement le code logiciel.

**Un Framework** est une **boite à outils pour un développeur web**. Frame signifie cadre et work se traduit par travail. Un Framework contient des composants autonomes qui permettent de faciliter le développement d'un site web ou d'une application.

**Quelle est la différence ?**

Un **IDE**est un notepad amélioré, un éditeur de texte et pleins d'outils pour faciliter des tâches comme la compilation ou le débogage. Un IDE n'est pas nécessairement associé à un langage, exemple Notepad+ qui peut être utilisé pour à peu près tous les langages.

Il s'agit donc d'un ensemble d'outils (comme l'éditeur de texte, un compilateur, le débuggueur, etc) destiné à vous "prendre par la main" pendant l'ensemble du processus de développement, du lancement du projet ("je veux créer un nouveau projet") à l'obtention du "produit fini" (mon projet fonctionne, je l'ai testé, je peux le diffuser).

Un **Framework**est un ensemble de classes et de structures écrit dans un langage précis pour un langage précis et destiné à faciliter le développement dans ce langage.

Il regroupe un ensemble d'implémentations de différentes fonctionnalités "clé en main", que vous pourrez — dans une certaine mesure — adapter à vos besoins, et que l'on pourrait comparer — en première approximation — à l'ensemble des pièces qui devraient composer une voiture.

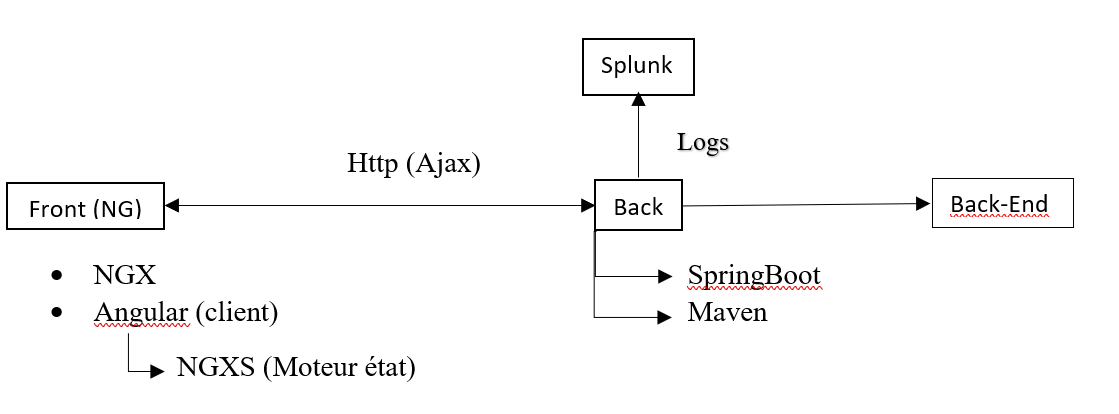
**AppDiag**

Plusieurs environnements : - INTG pour Intégration

- RECT pour Recette

- PEXP pour PreProd

- EXPL pour Prod



Le Ngx-Bootstrap est une bibliothèque très populaire pour utiliser des composants bootstrap dans des projets basés sur Angular. Il contient presque tous les composants de base de Bootstrap. Les composants ngx-bootstrap sont par conception modulaires, extensibles et adaptables.

NGXS est un modèle de gestion d'état + une bibliothèque pour Angular. Il agit comme une source unique de vérité pour l'état de votre application, fournissant des règles simples pour les mutations d'état prévisibles.

NGXS est calqué sur le modèle CQRS couramment implémenté dans des bibliothèques telles que Redux et NgRx, mais réduit le passe-partout en utilisant des fonctionnalités TypeScript modernes telles que des classes et des décorateurs.

**AJAX (*Asynchronous JavaScript + XML*)** n'est pas une technologie en soi, mais un terme désignant une « nouvelle » approche utilisant un ensemble de technologies existantes, dont : [HTML](https://developer.mozilla.org/fr/docs/Web/HTML) ou [XHTML](https://developer.mozilla.org/fr/docs/Glossary/XHTML), [CSS](https://developer.mozilla.org/fr/docs/Web/CSS), [JavaScript](https://developer.mozilla.org/fr/docs/Web/JavaScript), [DOM](https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model), [XML](https://developer.mozilla.org/fr/docs/Web/XML), [XSLT](https://developer.mozilla.org/fr/docs/Web/XSLT), et surtout [l'objet XMLHttpRequest](https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest). Lorsque ces technologies sont combinées dans le modèle AJAX, les applications web sont capables de réaliser des mises à jour rapides et incrémentales de l'interface utilisateur sans devoir recharger la page entière dans le navigateur. Les applications fonctionnent plus rapidement et sont plus réactives aux actions de l'utilisatrice ou de l'utilisateur.

Bien que le X de AJAX signifie XML, c'est le format [JSON](https://developer.mozilla.org/fr/docs/Glossary/JSON) qui est le plus souvent utilisé aujourd'hui à la place de XML, du fait de sa proximité avec JavaScript et de sa légèreté par rapport à XML. Autrement dit, tant JSON que XML peut être utilisés comme format afin d'empaqueter des données en AJAX.

**Extensible Markup Language (XML)** permet de définir et de stocker des données de manière à pouvoir les partager. XML prend en charge l'échange d'informations entre des systèmes informatiques tels que les sites web, les bases de données et les applications tierces. Les règles prédéfinies facilitent la transmission des données sous forme de fichiers XML sur n'importe quel réseau, car le destinataire peut utiliser ces règles pour lire les données avec précision et efficacité.

Apache **Maven** est un outil de gestion et d'automatisation de production des projets logiciels Java en général et Java EE en particulier. Il est utilisé pour automatiser l'intégration continue lors d'un développement de logiciel. Maven est géré par l'organisation Apache Software Foundation.

Avec la plate-forme **Splunk**, nous pouvons rechercher rapidement et facilement nos logs. Nous pouvons également apporter de légères modifications au code pour améliorer la lisibilité de nos logs dans l'ensemble de l'organisation.

**Spring Boot** est un framework qui facilite le développement d'applications fondées sur **Spring** en offrant des outils permettant d'obtenir une application packagée en JAR, totalement autonome

**Spring** est un framework open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests.

**Angular** est un framework Javascript qui permet de réaliser des applications WEB Cross Plateform : Web, mobile et desktop. Il est possible de développer sur **Angular** soit en Javascript natif, soit en Dart (un langage développé par Google), soit en TypeScript (un langage développé par Microsoft). C’est un framework pour clients, open source, basé sur TypeScript.

**TypeScript** est un langage fortement typé qui permet d'avoir un code plus structuré et plus facile à maintenir. Il se repose sur le langage JavaScript en ajoutant la possibilité de typer de manière statiques les variables et les fonctions. Le code peut être ensuite convertit en JavaScript en retirant les déclarations liées aux types.

On m’a donc donnée dans un premier temps une mission :

# **User référentiel produit**

### Contexte :

Nous sommes régulièrement sollicités pour fournir des informations diverses sur un produit.

L’idée de cette évolution est de **mettre à disposition une fiche produit complète.** Cela permettra également d’en vérifier le paramétrage.

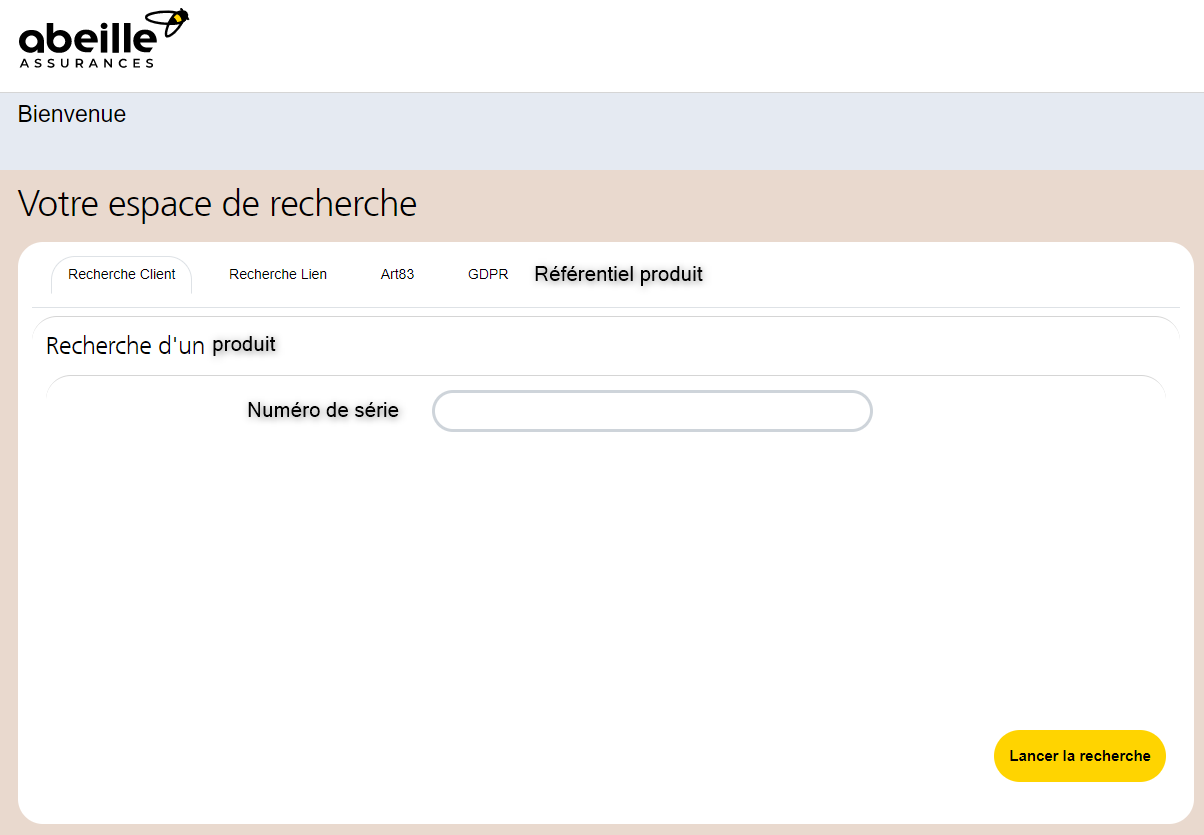
Pour ce faire, on va s’appuyer sur 3 tables : fe\_parameter, fe\_series\_ic et fe\_policydict

### Partie 1 - Interface de recherche :

Ce nouvel onglet sera nommé « Référentiel Produit »

Sur cette page de recherche, nous avons besoin d’une zone :

* Numéro de série



### Partie 2 - Résultat de la recherche :

Lorsqu’on clique sur le bouton « Lancer la recherche », il faut aller lire dans 3 tables et restituer certains champs qui sont spécifiés ci-après dans la requête.

Voici les requêtes à implémenter en fonction des infos recherchées.

#### *Partie 2.1 – Eligibilité de la Série*

**Requête FE\_SERIES\_IC :**

*select ic\_num\_ser Série, ic\_niveau Périmètre, ic\_auto Création\_auto, ic\_groupe Série\_Groupe\_Eligible, key\_longlabel Libellé\_Série*

*from crm.fe\_series\_ic, crm.fe\_parameter*

*where key\_class = 'POLICYDICT\_OWNED'*

*and key\_language = 'FR'*

*and ic\_num\_ser = key\_shortlabel*

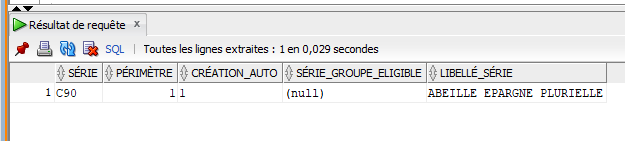
*and ic\_num\_ser = 'C90';*

La jointure vers fe\_parameter permet d’obtenir le libellé.

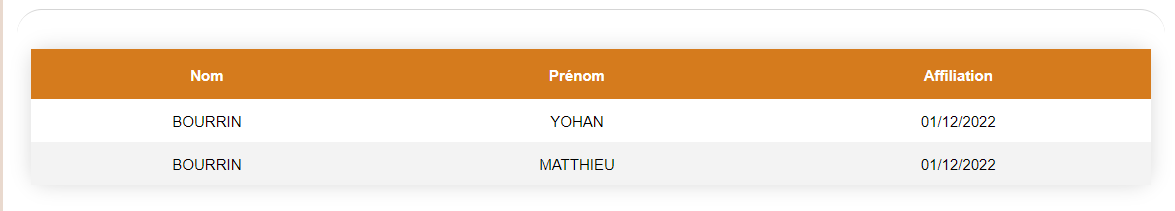
Il faut restituer les champs du select. En théorie, une ligne est attendue, **mais prévoir l’affichage de plusieurs lignes.**

**Si la requête ne remonte aucune ligne, il faut afficher le message suivant :**

**« Série non trouvée, ou non éligible. »**



Exemple pour le rendu en remplaçant par les champs du select :



#### *Partie 2.2 – Référentiel produit*

**Requête FE\_POLICYDICT :**

*select pod\_l3\_producttype Série,*

*key\_longlabel Libellé\_Série,*

*pod\_l1\_group Série\_Groupe,*

*pod\_pacte\_f Pacte,*

*pod\_fis\_pacte\_f Fis\_Pacte,*

*pod\_mono\_f Mono,*

*pod\_multi\_f Multi,*

*pod\_ecplus\_f Ecplus\_f,*

*pod\_crt\_gsm Gsm,*

*pod\_crt\_collectif Collectif*

*from crm.fe\_policydict, crm.fe\_parameter*

*where pod\_l3\_producttype = 'D12'*

*and key\_class = 'POLICYDICT\_OWNED'*

*and key\_language = 'FR'*

*and pod\_l3\_producttype = key\_shortlabel*

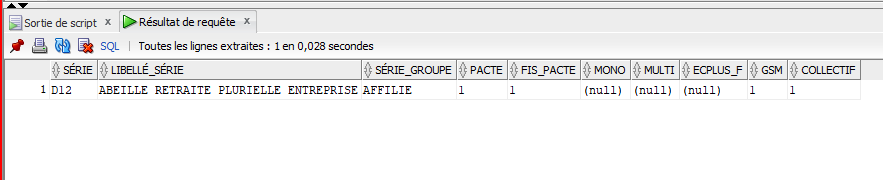
*;*

La jointure vers fe\_parameter permet d’obtenir le libellé.

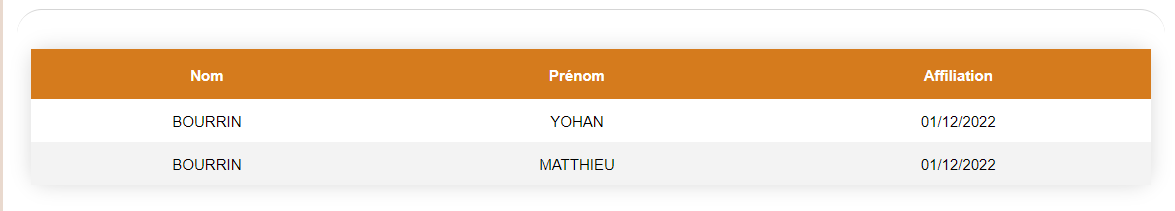
Il faut restituer les champs du select. En théorie, une ligne est attendue, mais prévoir **l’affichage de plusieurs lignes.** Exemple Série NPA.

**Si la requête ne remonte aucune ligne, il faut afficher le message suivant :**

**« Série non trouvée. »**



Exemple pour le rendu en remplaçant par les champs du select :



#### *Partie 2.3 : Export Séries éligibles*

Il nous est souvent demandé la liste des séries éligibles.

Nous allons proposer un bouton téléchargement de la table correspondante : fe\_series\_ic

**Requête export table fe\_series\_ic :**

*select ic\_num\_ser Série, ic\_niveau Périmètre, ic\_auto Création\_auto, ic\_groupe Série\_Groupe\_Eligible, key\_longlabel Libellé\_Série*

*from crm.fe\_series\_ic, crm.fe\_parameter*

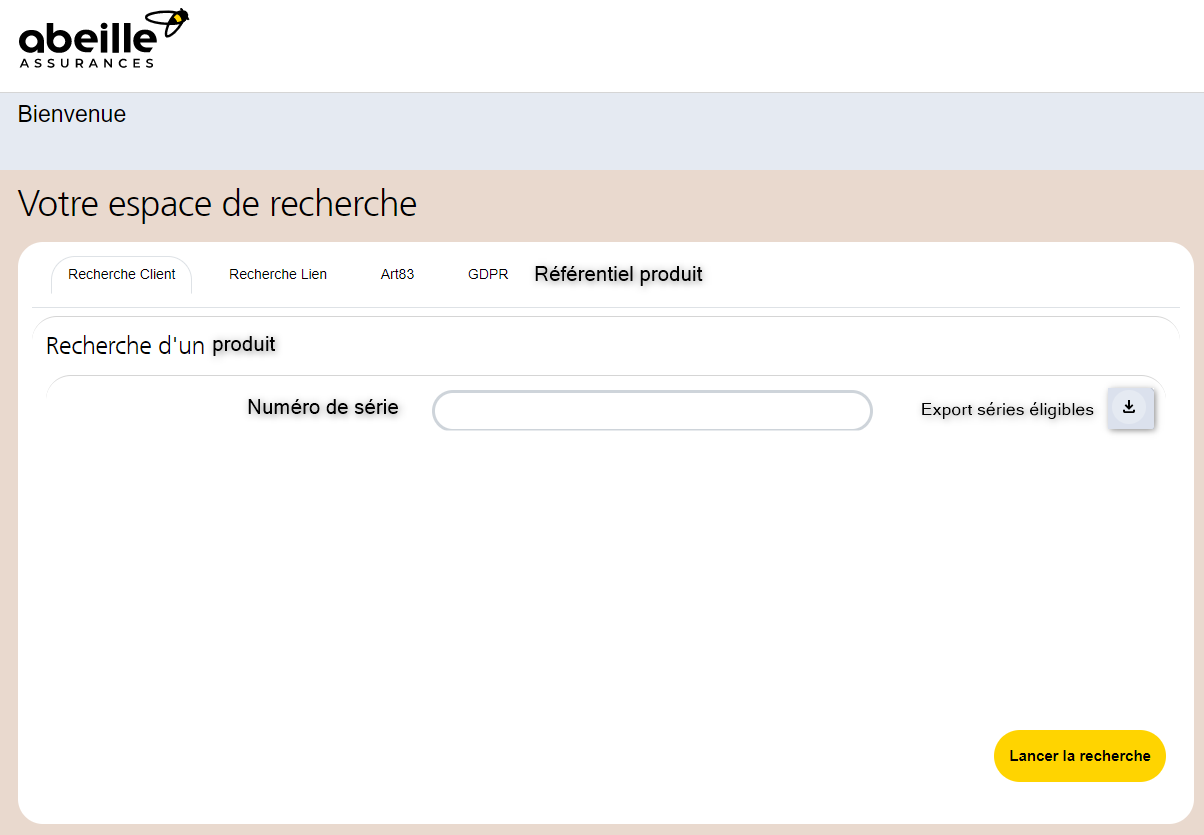
*where key\_class = 'POLICYDICT\_OWNED'*

*and key\_language = 'FR'*

*and ic\_num\_ser = key\_shortlabel*

*order by ic\_num\_ser ;*

Ajouter le bouton télécharger et le texte « Export séries éligibles »



**L’export doit être au format csv, séparateur ‘;’**

Nom du fichier : « Export series IC.csv »

Deuxième mission : on me demande de récupérer à partir d’un menu déroulant une catégorie et d’afficher tous les résultats en rapport avec cette catégorie dans la base de données.

**Angular** est un framework pour créer des Single Pages Application (SPA)

**Une Single Page Application (SPA)** est une application qui fonctionne dans un navigateur sans que l’utilisateur n’ait besoin de recharger la page.

**Approche Composant Angular**

Les applications Angular sont faites des composants (components).

Un composant est une combinaison entre un template HTML et une classe composant qui contrôle une portion de l’écran.

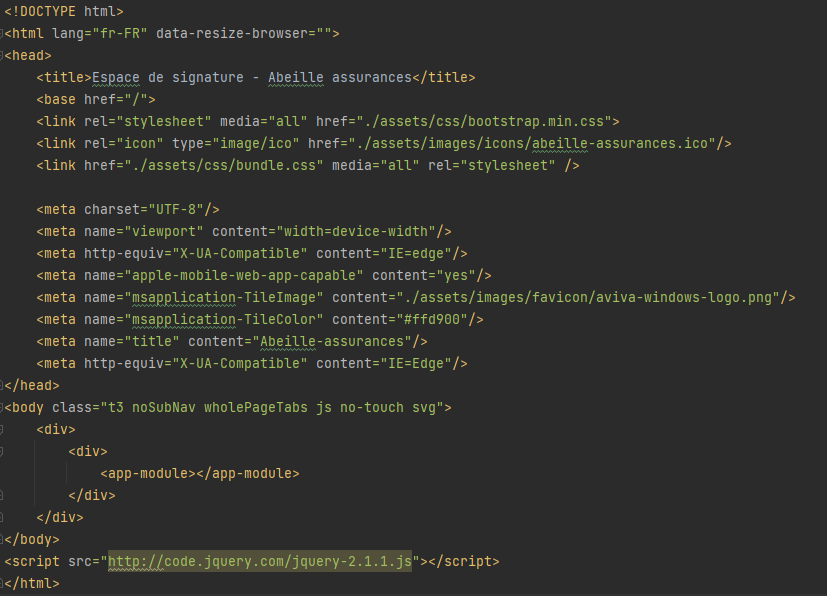
**app.component.ts**



Chaque composant contient un décorateur @Component qui prend un objet contenant des métadonnées. Elles contiennent l’emplacement du template HTML et du style pour le composant. Ici, il n’y a que le template.

La propriété selector permet d’indiquer à Angular d’afficher le composant à l’emplacement d’une balise html spécifique, ici <app-module> dans le fichier index.html.

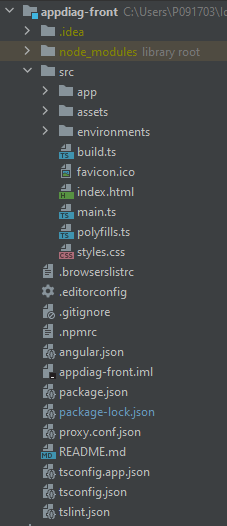
**index.html**



Nous retrouvons bien la balise <app-module> dans le fichier index.html qui indique où doit être placé le composant.

**Comprendre la structure**

**L’arborescence**



**Racine ./**

***.editorconfig :*** Configuration simple pour l’éditeur pour s’assurer que tous ceux qui utilisent le projet aient la même configuration de base. La plupart des éditeurs prennent en charge ce fichier.

***.gitignore :*** Fichier git pour assurer que les fichiers générés automatiquement ne sont pas pris en compte. Par exemple les dépendances ( /node\_modules), ainsi que le build du projet (/dist).

***.npmrc :*** Fichiers de configuration npm.

* npm obtient ses paramètres de configuration à partir de la ligne de commande, des variables d'environnement et des npmrcfichiers.
* La npm configcommande peut être utilisée pour mettre à jour et modifier le contenu des fichiers utilisateur et global npmrc.

***angular.json :*** Configuration pour Angular CLI. Vous pouvez par exemple configurer le nom et l’emplacement du folder où sera buildé le projet.



***appdiag-front.iml :*** Un fichier IML est un fichier de paramètres de module créé par IntelliJ IDEA, un environnement de développement intégré ( [**IDE**](https://techterms.com/definition/ide) ) utilisé pour développer des applications Java. Il stocke des informations sur un module de développement, qui est un composant Java, Plugin, Android ou Maven d'une application Java. Les fichiers IML stockent le type, les chemins, les dépendances et d'autres paramètres d'un module.

Les fichiers IML stockent les paramètres au niveau du module pour les projets de développement plus importants. IntelliJ IDEA utilise des fichiers [**.IPR**](https://fileinfo.com/extension/ipr) ou [**.IDEA**](https://fileinfo.com/extension/idea) (selon la version du projet) pour stocker les paramètres généraux du projet. Ces fichiers peuvent référencer plusieurs fichiers IML. Les fichiers IPR et IML utilisent tous deux le formatage XML.

Si vous supprimez accidentellement le fichier IML d'un module, vous pouvez souvent régénérer le fichier en ouvrant et en fermant votre IDE, en reconstruisant votre projet ou en réimportant votre projet.

***package.json :*** Configuration npm listant les paquets tiers que le projet utilise (les dépendances). On peut également ajouter nos propres scripts personnalisés ici.

* **NPM** (**Node** Package Manager) comme son nom l'indique est le "package manager" officiel de l'univers JavaScript (frontend / backend). Il est installé automatiquement lors de l'installation de **NodeJS**.

***package-lock.json :*** Arbre exacte des dépendances et de leurs propres dépendances, permettant de réinstaller exactement les mêmes versions dans votre équipe.

***proxy.conf.json :***Un proxy est un logiciel qui se situe entre votre application JavaScript/Angular effectuant la requête Ajax et votre API backend. C'est le choix à faire dans une appli classique.

***README.md :*** Documentation de base pour le projet, pré-remplie d’information de commande CLI. S’assurer de l’améliorer avec la documentation du projet afin que quiconque puisse builder l’application en suivant les instructions contenues dans le fichier !

***tsconfig.app.json :*** Extension de la configuration du compilateur TypeScript pour compiler l’application.

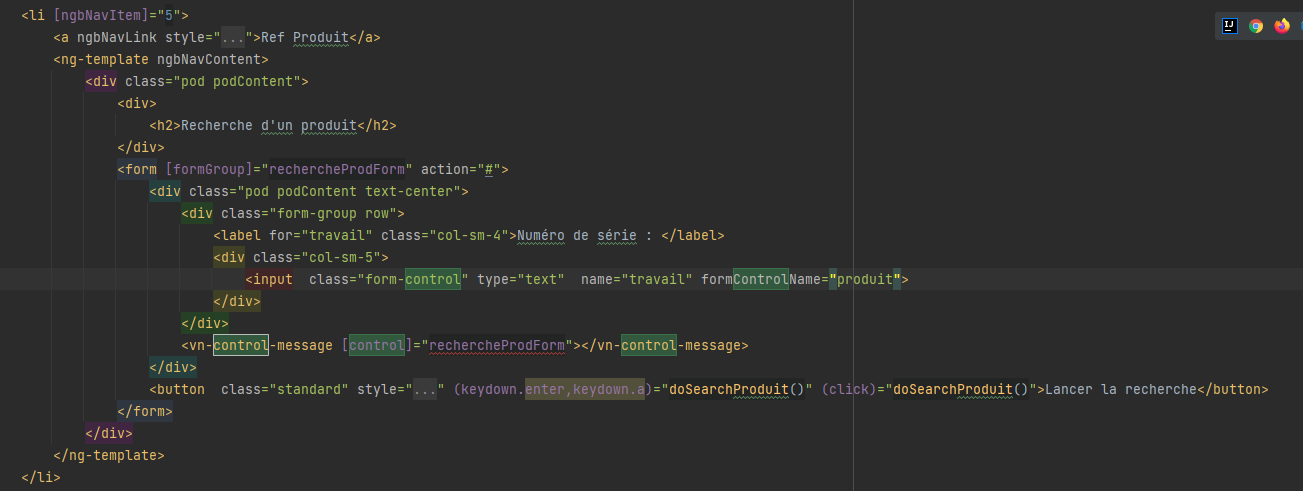
***tsconfig.json :*** Configuration du compilateur TypeScript pour compiler l’application. Ce fichier est également utilisé par l’IDE afin de nous donner des informations utiles (mauvais type d’input par exemple).

***tslint.json :*** TSLint est un outil d'analyse statique extensible qui vérifie le code [TypeScript](http://www.typescriptlang.org/) pour les erreurs de lisibilité, de maintenabilité et de fonctionnalité. Il est largement pris en charge par les éditeurs et les systèmes de construction modernes et peut être personnalisé avec vos propres règles, configurations et formateurs.

*Note du 05/01/23 :*

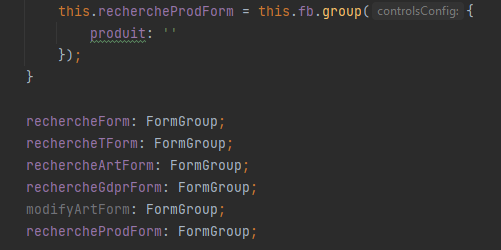
Changement dans : C:\Users\P091711\IdeaProjects\appdiag-front\src\app\appDiagModule\components\recherche/ recherche.component.html

Avec la création d’une nouvelle partie de la barre de navigation pour ajouter la référence produit ligne 243 :

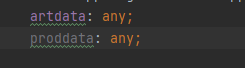


Dans le fichier .ts :

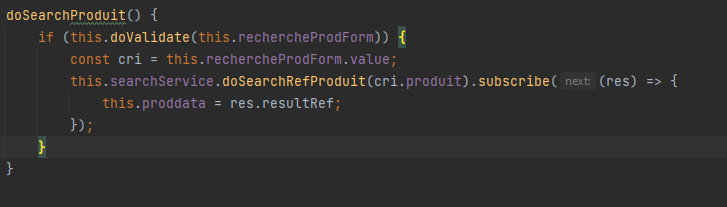
* Ajout d’un this pour les produits et création d’une nouvelle recherche de produit



* Ajout de proddata :



* Création de doSearchProduit

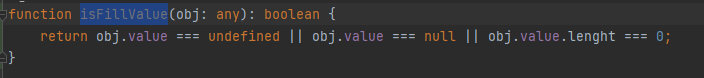
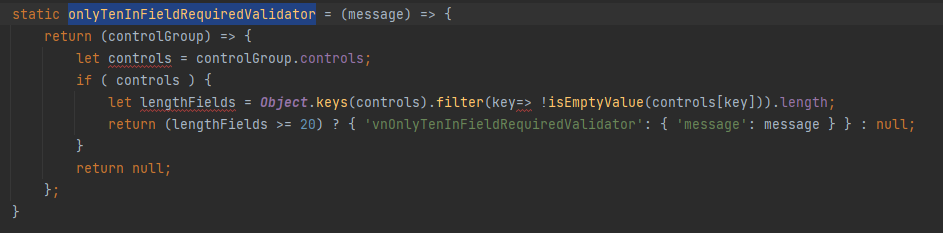
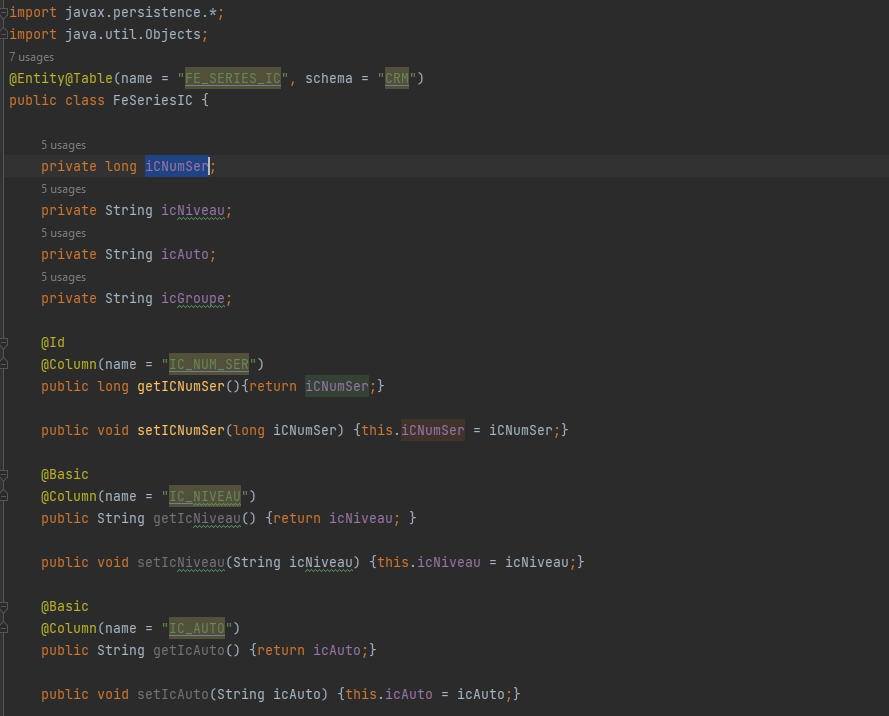
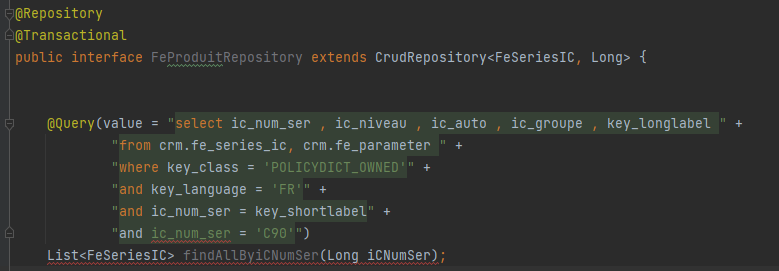
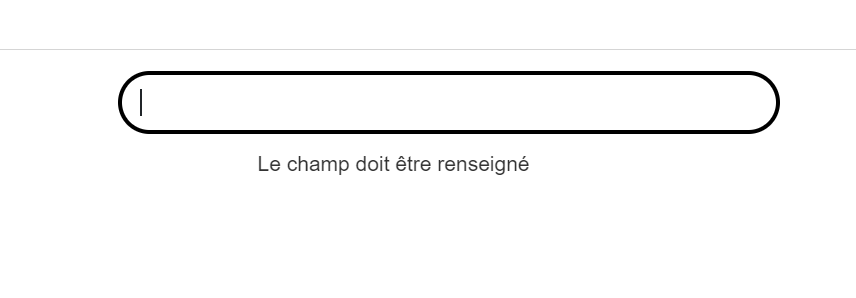
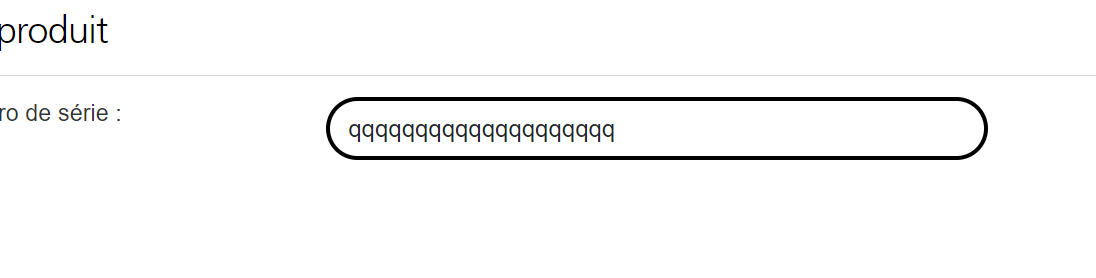
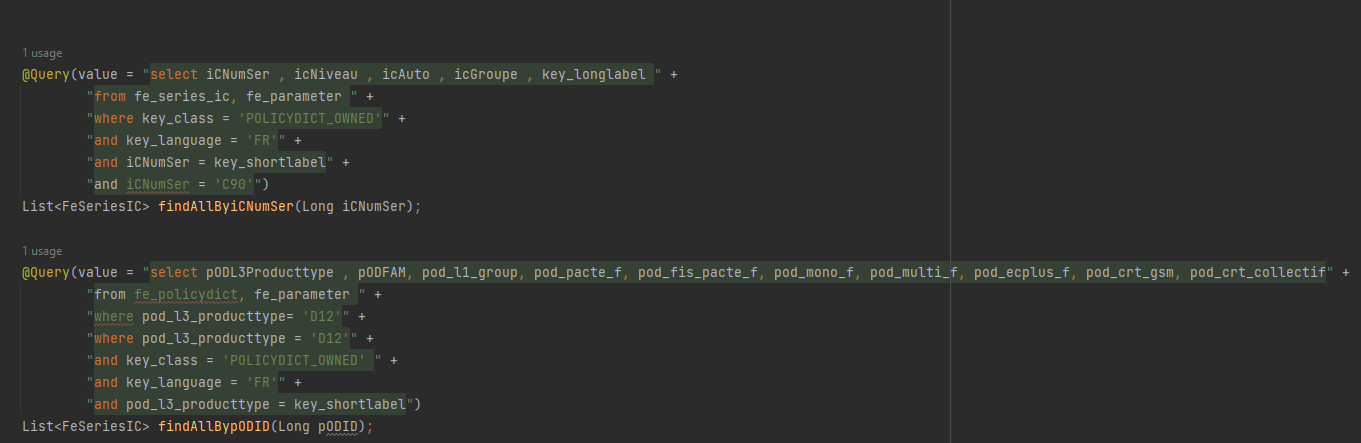
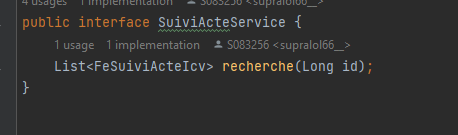


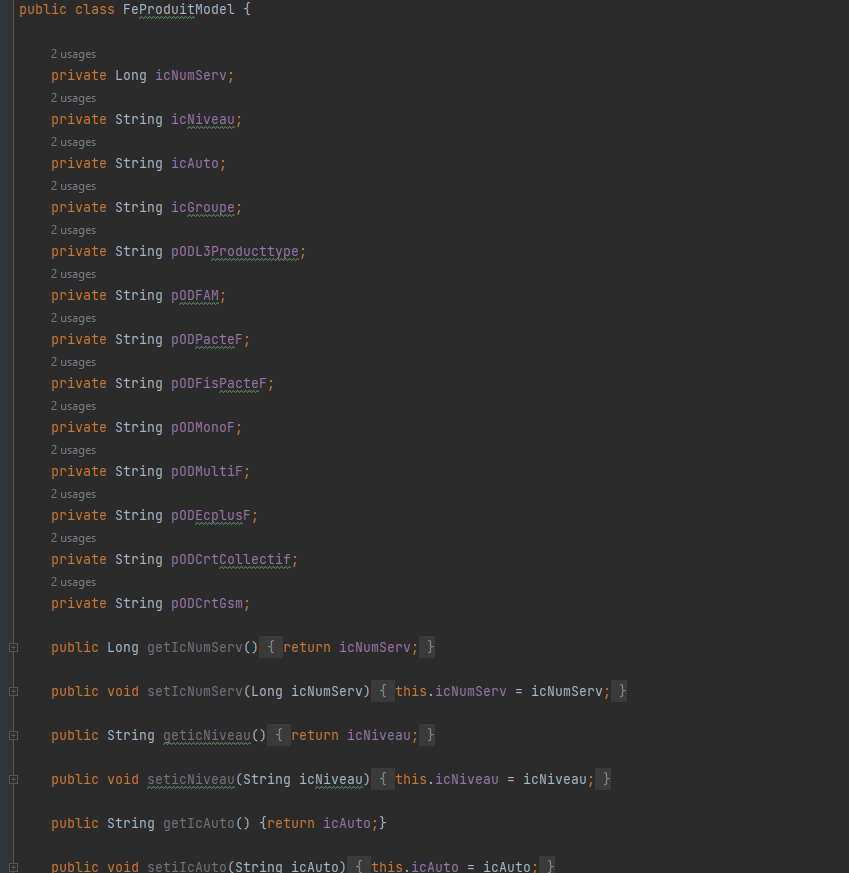
C:\Users\P091711\IdeaProjects\appdiag-front\src\app\appDiagModule\services\app.diag.service.ts

Création de doSearchRefProduit pour faire fonctionner la méthode du dessus :



Dans custom validator :

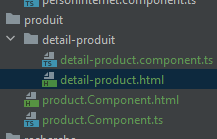
* Création de la function : isFillValue 
* Création de la clause de validité onlyTenInFieldRequiredValidator : 
* Ne pas oublier de faire le débogage des autres fonctions de l’appli pour mieux comprendre comment ça marche
* Création d’une entité dans le back, permettant de faire la connexion avec une table de la base de données : 
* Puis création d’un repository qui va prendre l’entité et va inclure la ou les requête SQL également : 
* Mise en place d’un moyen de lancer un message qui spécifie qu’il faut écrire au moins 1 caractère : 
* Mise en place d’un nombre maximal de caractère utilisés sur 20 : 
* Finition des repository qui sont censés permettre au back d’utiliser les requètes dont on a besoin sur la base de données. 
* Création de l’interface du service permettant l’appel des requetes : 
* Création du dit service : 
* Création d’un modèle qui continent à peu près la même chose qu’un entité sauf que l’entité ne va pas être affiché, elle ne sert qu’à contenir les éléments de la base de données et à être utilisées dans les différentes fonctions, alors que le modèle lui représente ce qui va être affiché ( il n’y a donc pas de lien avec la base de données), les entités comme les modèles possèdent une initialisation de variable ainsi qu’une création de getteur setteur pour chacun d’entre eux  :

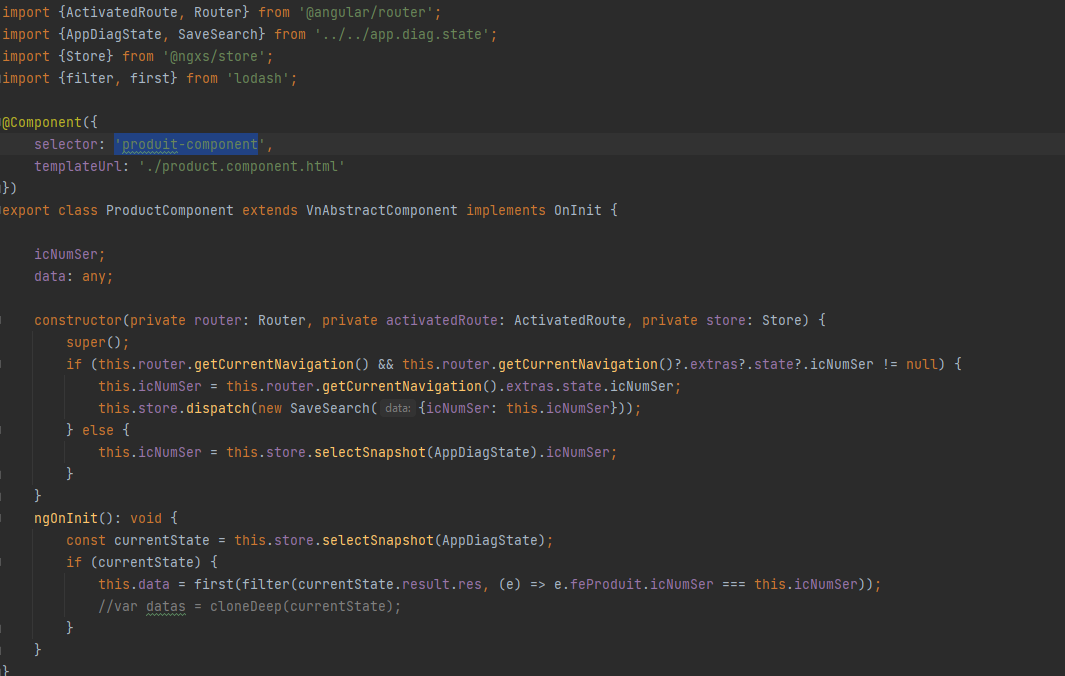


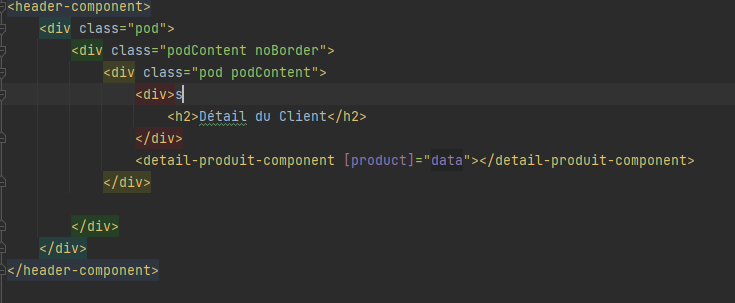
* Création de DTOProduit qui va permettre de passer les données dans le contrôleur

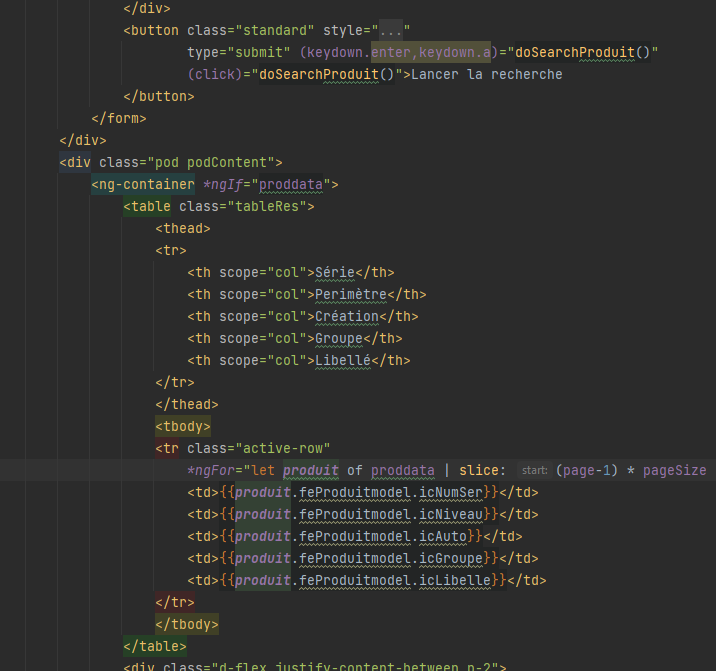
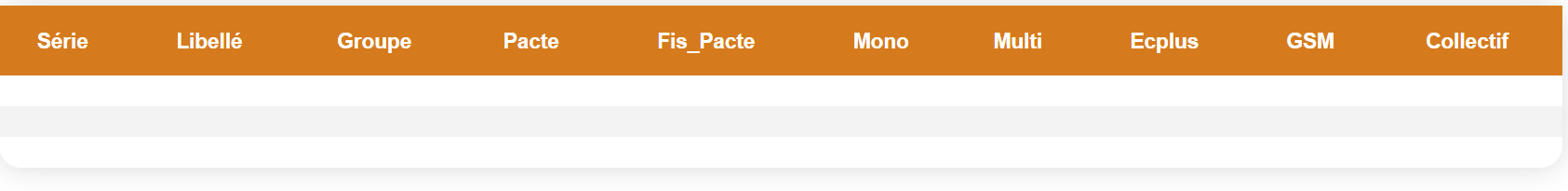


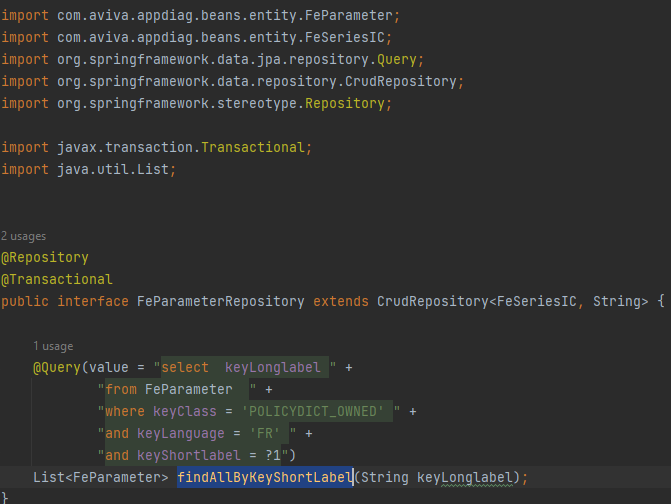
* Création de component produit : avec un component globale nommé produit et un component spécifique aux détails des produit :

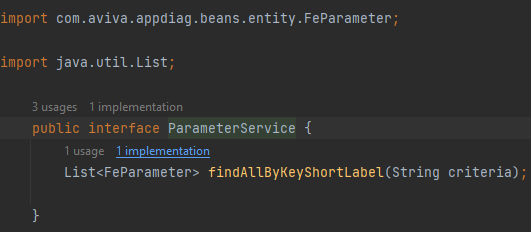


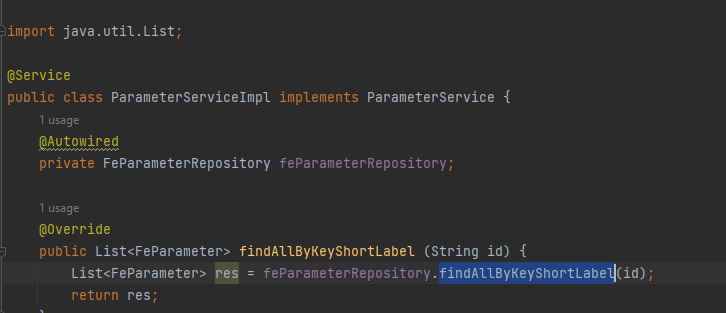
Avec pour chacun un fichier html et un fichier typescript. Produit.ts :

Produit.html : 

* On rajoute maintenant au fichier recherche.componenent.html le tableau permettant l’affichage des données trouvées lors de notre requête : 
* Nous pouvons désormais accéder sur le site lors d’une recherche aux données du produit dont le numéro de série a été saisie : 
* Rajout d’un deuxième tableau permettant d’afficher une deuxième requête, utilisé au même moment que la première : 
* Rajout d’une requête pour trouver le libelle dans la table FE\_PARAMETER, avec le repository :

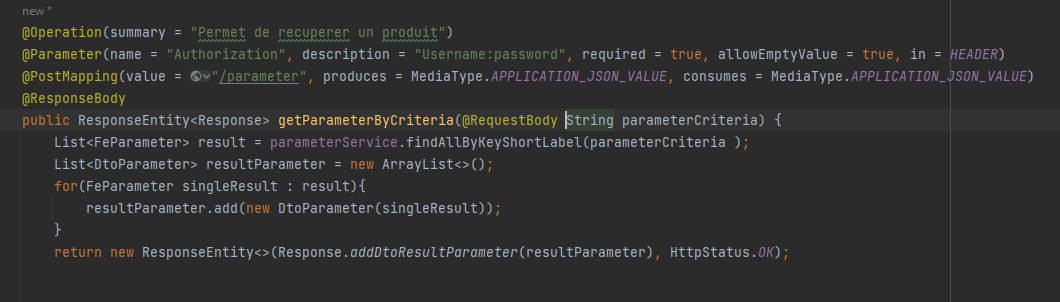


Puis ajout d’une implémentation : 

Puis d’un service : 

Création d’un nouveau Dto :



Et ajout d’une condition de recherche dans le recherche contrôleur : 

Malheureusement ce changement à entrainer une erreur : « Bad Request » qui empêche toujours l’affichage du libellé.

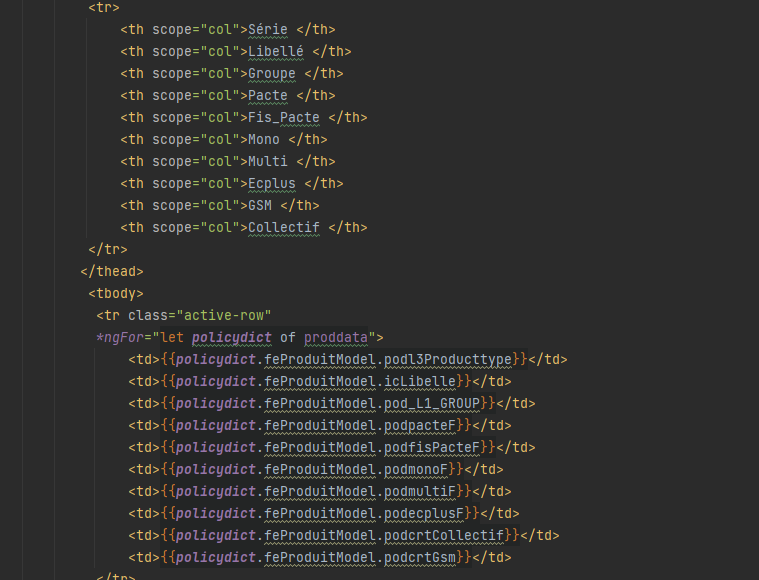
Pour l’affichage du 2nd tableau je pense devoir utiliser la même technique que celle précédemment utilisé pour le libellé, c’est-à-dire créer un nouveau répository, un nouveau service, une nouvelle implémentation, un nouveau DTO et une nouvelle section dans le recherche contrôleur.

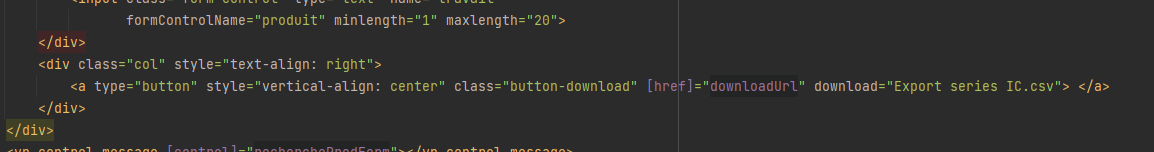
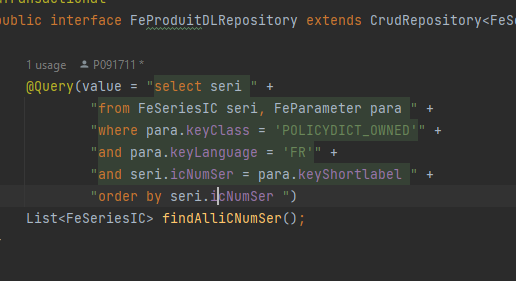
* Pour pouvoir envoyer les données de l’entité correspondant à Policydict et à Parameter, nous avons dû supprimer les deux Controller créé précédemment, et rajouter des conditions dans le for du premier : 

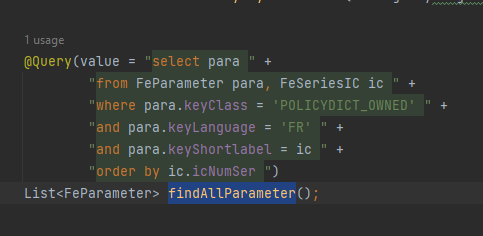
Petit problème néanmoins : si maintenant les valeurs sont bien transmises au front, celui-ci ne veut pas afficher les valeurs du deuxième tableau et le Controller.

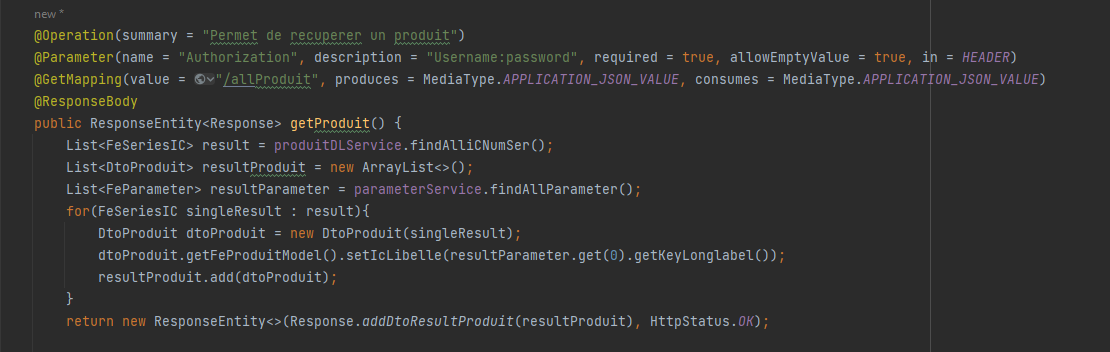
Problème réglé : 

De plus les noms utilisés pour l’affichage dans le front n’était pas les bons :

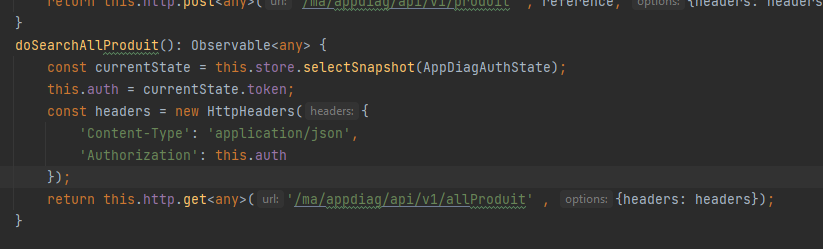


* Ajout d’un bouton de téléchargement : 
* Ce bouton récupère toutes les valeurs de la table FeSeriesIC et le libellé et les met dans un fichier .CSV. Pour cela on doit recréer une requête pour récupérer la table FeSerieIC complètement : 

Et une autre requête pour le libelle de FeParameter : 

Ensuite on ajoute les services et services Implémentation, puis on créé un nouveau Dto et un nouveau contrôler : 

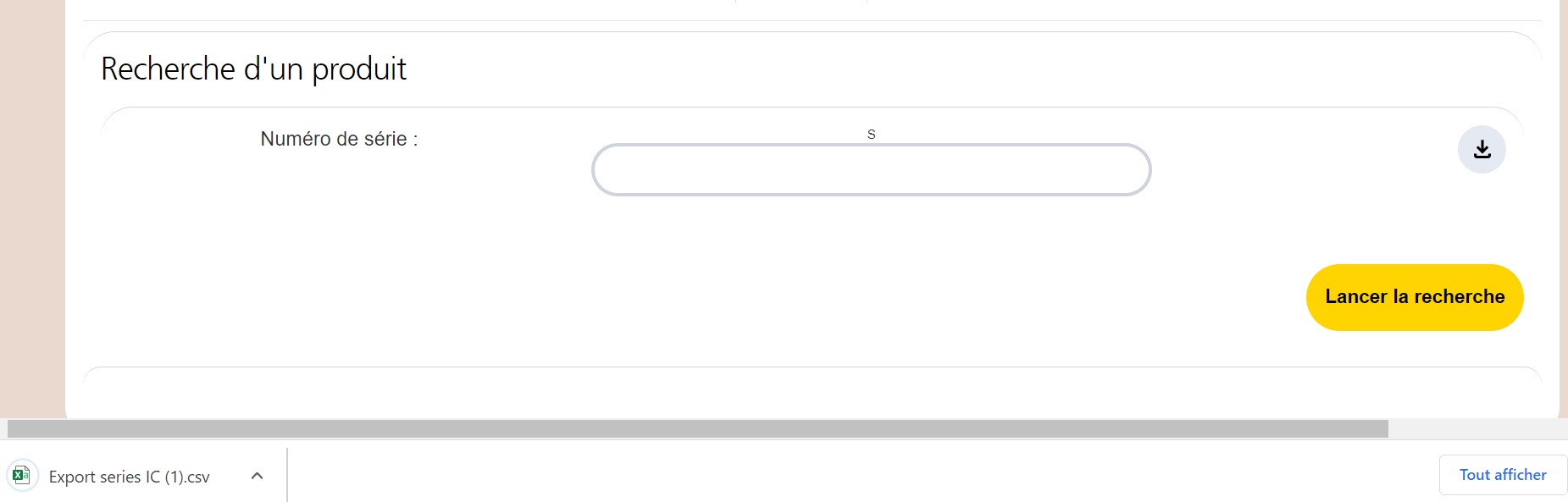
et on ne met pas de valeur à envoyer ce qui nous permet de transformer le PostMapping en Getmapping.

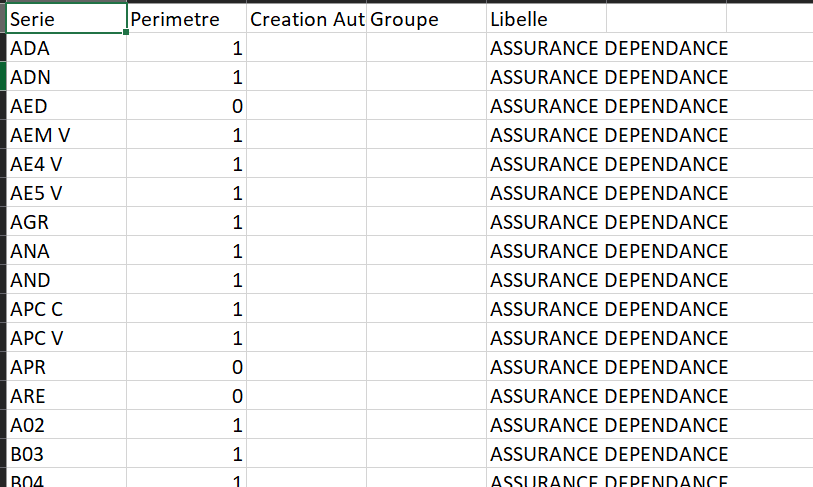
Ensuite on créé dans le front un nouveau service : 

Et dans recherche.component.ts ce qui va nous permettre de récupérer les valeurs : 

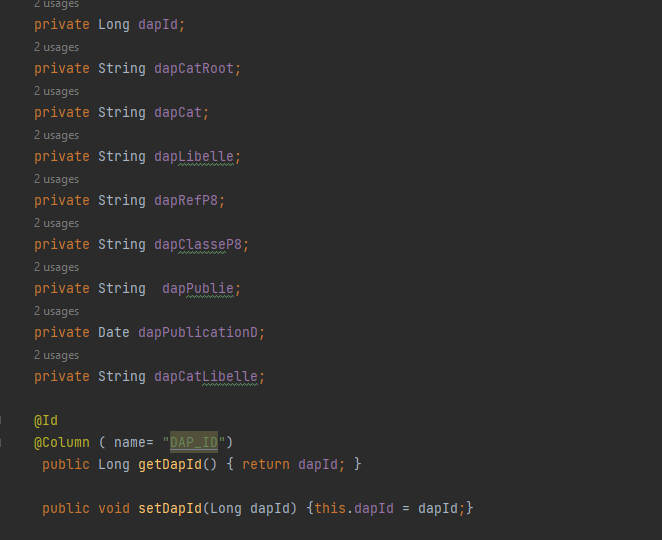
et ce qui va nou permettre de les télécharger en format CSV : 

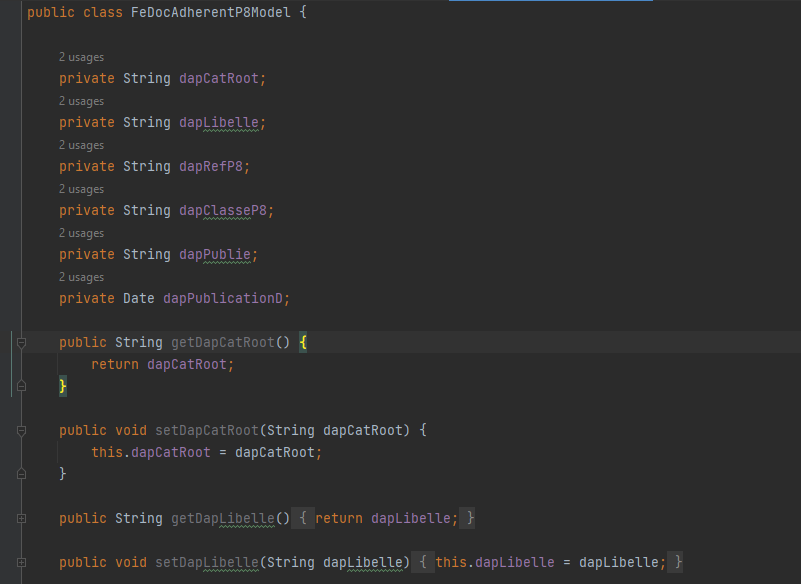
Et grâce à cela on peut télécharger le fichier contenant toute les références produites :

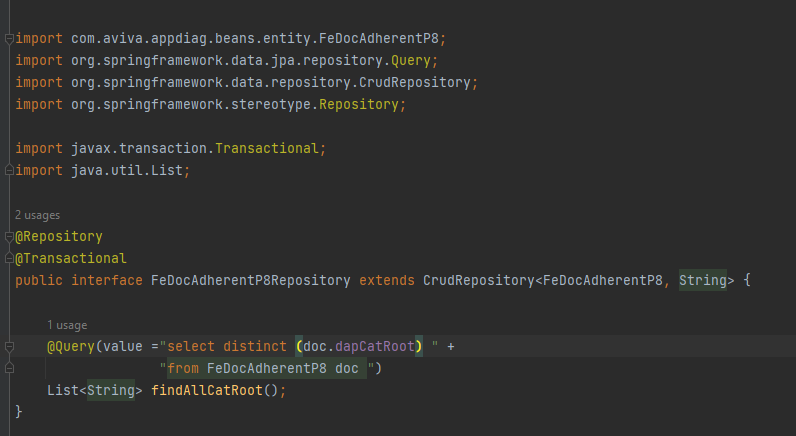


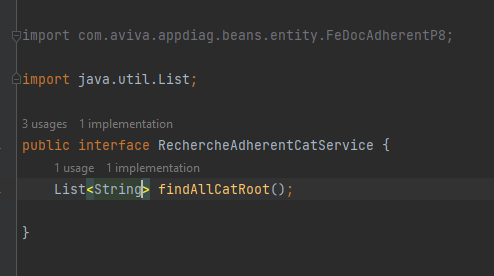


Nouvelle tâche : afficher dans un nouveau tableau quelque champs de la table fe\_doc\_adherent\_p8, à partir d’un menu déroulant qui va refléter les valeurs des catégories de cette table. :

* On créé, dans le back, une entité, un modèle, un repository (pour les requêtes), un service et un service implement, ainsi qu’un Dto et un nouveau contrôleur pour le menu déroulant : entité :

Modèle :

Repository : 

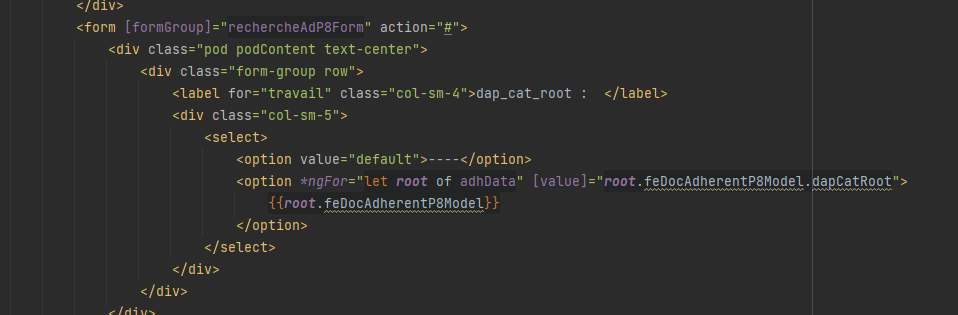
Service : 

Service implement : 

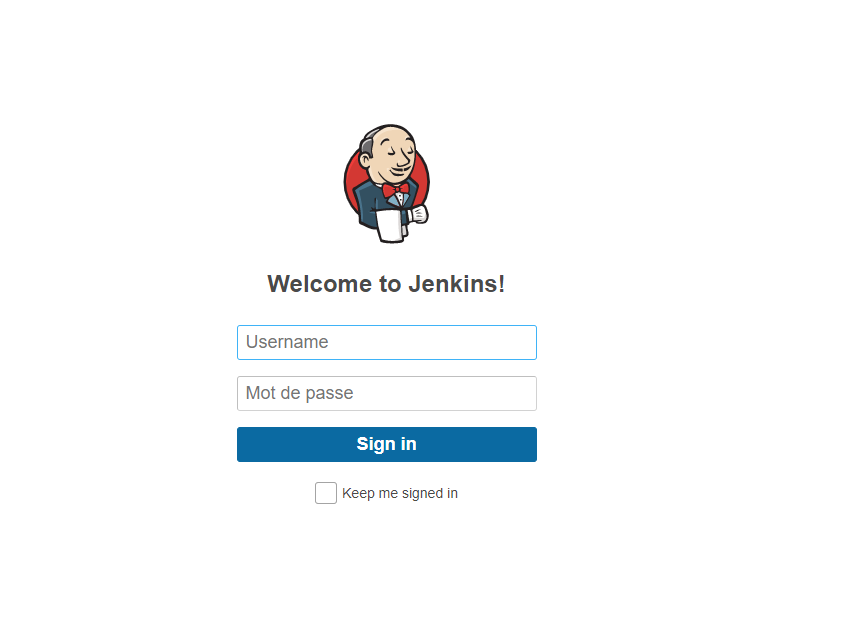
Dto : 

Controlleur : 

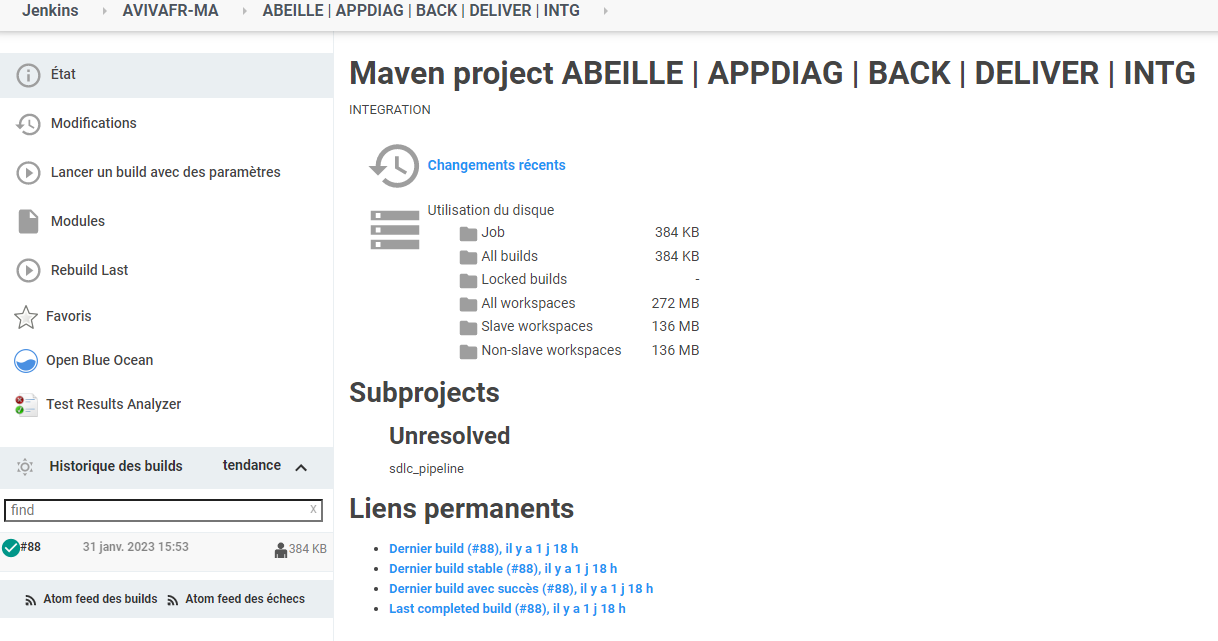
* Et enfin dans le front on créé une nouvelle partie de la page principale :



* Lorsque tout est fini on peut donc commit nos changements, puis on passe sur la branche master afin de commit également sur cette branche. Après cela on peut faire un cherry pick afin d’être sûr que ça fonctionne bien et de passer les données de la branche dev vers la branche master.
* Ensuite on utilise l’outil Jenkins qui va nous permettre de faire passer nos changements sur les différents environnements de l’application, Prex, Prod …, en back comme en front :









Une fois cela fait nous n’avons plus qu’à attendre les retours des personnes qui font les tests pour les éventuels changements ou amélioration à mettre en place.