

Documentation de stage

➤ Formulaire de recherche « compte client »

The screenshot shows the 'Recherche Client' search form. At the top right is a yellow 'Se déconnecter' button. Below it, there are tabs for 'Recherche Client', 'Recherche Lien', 'Art83', and 'GDPR'. The main area is titled 'Recherche d'un client' and contains six input fields: 'Numéro client', 'Mail client', 'Identifiant annuaire (Tam_Id)', 'ID person internet (Pri_Id)', 'Identifiant A&M (Pri_Username)', and 'Numéro de contrat'. Below these fields are three buttons: 'Search Afer' (grey), 'Search Aviva' (grey), and 'Lancer la recherche' (yellow).

➤ Ce formulaire permet de rechercher un compte client en base de donnée à partir de multiples critères :

- Numéro de client
- Mail client
- Tam Id (identifiant annuaire ADLDS)
- Pri_id (identifiant Bdd)
- Identifiant client
- Numéro de contrat

➤ Le bouton Search Aviva appelle le back office du Dommage à partir d'un numéro de contrat abeille.

➤ Les boutons Search Afer interroge la base de données CRM à partir d'un numéro de contrat afer.

3

➤ Résultat de recherche « compte client »

The screenshot shows the search results table with the following columns: Prs Id, Nom Client, Prénom Client, Identifiant A&M, TAM_ID A&M, and PRI_ID A&M. There is one result row:

Prs Id	Nom Client	Prénom Client	Identifiant A&M	TAM_ID A&M	PRI_ID A&M
2 -9994415359	KLECZEWSKI	MIGA	tcreplet	6000000120	-9998278311 1

Below the table are navigation buttons: «» (left), 1 (current), «» (right), and a dropdown menu for '4 items per page'.

The screenshot shows the second page of search results table with the same columns. There is one result row:

Prs Id	Nom Client	Prénom Client	Identifiant A&M	TAM_ID A&M	PRI_ID A&M
2 -9999345784	AUBIN	PHILIPPE			

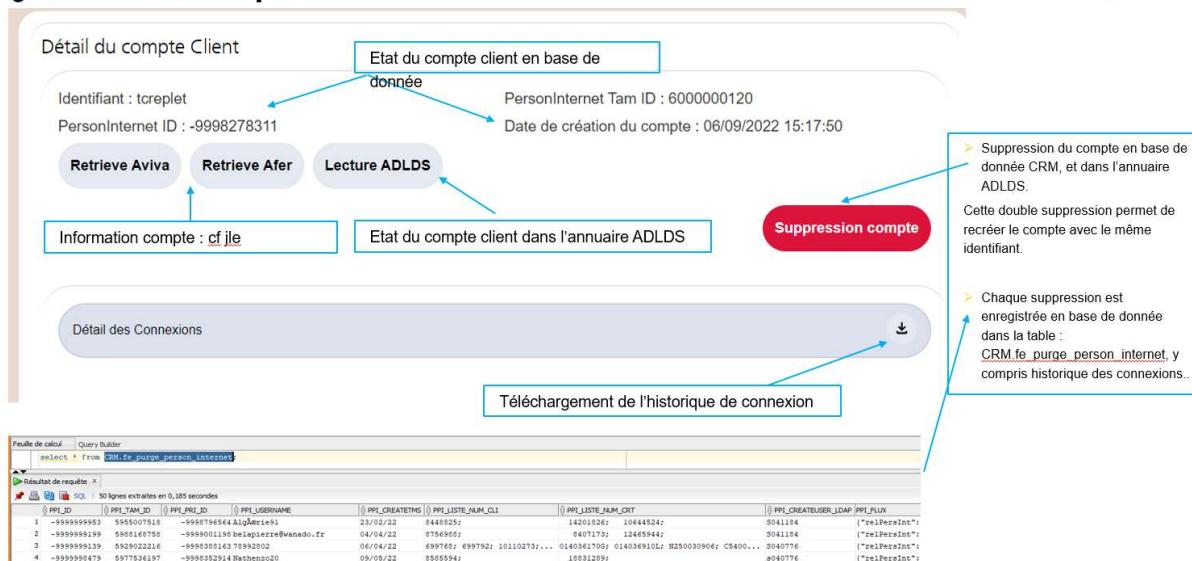
Below the table are navigation buttons: «» (left), 1 (current), «» (right), and a dropdown menu for '4 items per page'.

➤ 1 : si présence d'un pri_id, cliquer sur le numéro sous le pri_id pour accéder au détail du compte client.

➤ 2 : en cas d'absence du pri_id, cliquer sur le numéro en dessous du prs_id pour accéder au détail client / contrat.

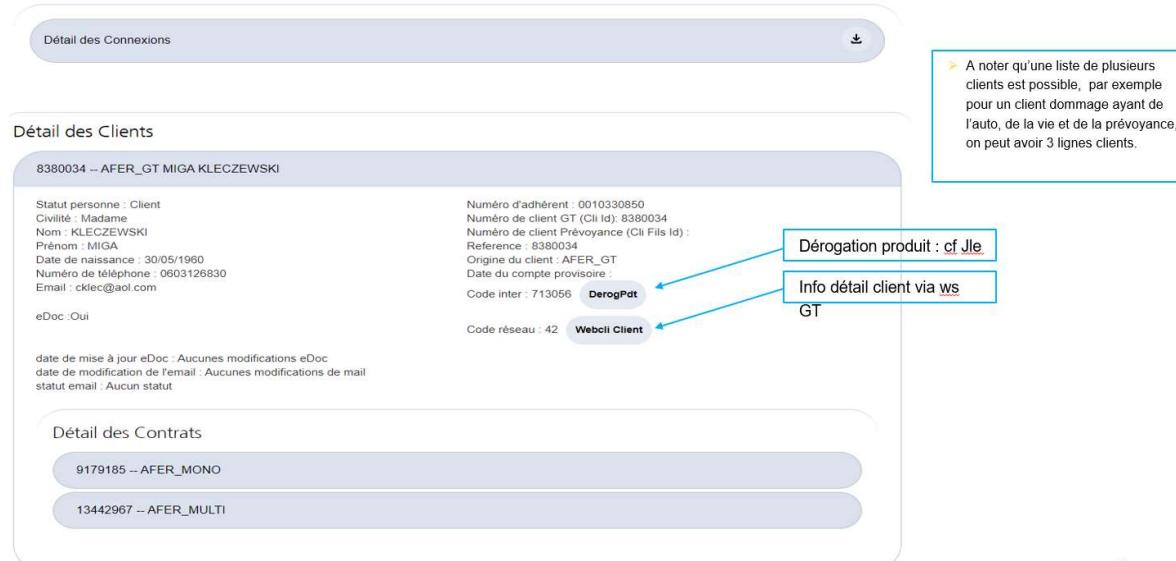
4

▷ Détail du compte client



5

▷ Détail client



6

7 Détail contrat

eDoc : Oui
Code réseau : 42 Webcli Client

date de mise à jour eDoc : Aucunes modifications eDoc
date de modification de l'email : Aucunes modifications de mail
statut email : Aucun statut

Détail des Contrats

9179185 -- AFER_MONO

Numéro de contrat : 9179185
Origine du contrat : AFER_MONO
Eligibilité contrat : DANS LE PERIMETRE, CONSULTATION COMPLETE
Status contrat : En cours
Date d'effet du contrat : 28/05/1993
Situation du contrat : contrat

Code Produit : N08
Date GDPR :
Statut GDPR (P.A.R.) :
Valeur de rachat : 142007
Date de sortie :
GT

Webcli_Contrat Tpoll_Mc Info détail contrat via ws GT

13442967 -- AFER_MULTI

Numéro de contrat : 13442967
Origine du contrat : AFER_MULTI
Eligibilité contrat :
Status contrat : Clôturé
Date d'effet du contrat : 11/04/2001
Situation du contrat : racheté

Code Produit : N09
Date GDPR :
Statut GDPR (P.A.R.) :
Valeur de rachat : 0
Date de sortie : 01/08/2006 00:00:00
GT

Webcli_Contrat Tpoll_Mc

7

8 Recherche lien signature démat

Bienvenue

Votre espace de recherche

Recherche Client Recherche Lien Art63 GDPR

Recherche d'un Lien

Numéro travail : 1094760245

Lancer la recherche

URL Client Afer:
<https://adherent.gie-afel.fr/hub/index.jsp?ssoTicket=9201ae6f-86e3-4daf-8699-c2be195524a9>

URL Client Aviva:
<https://www.aviva.fr/hub/?ssoTicket=9201ae6f-86e3-4daf-8699-c2be195524a9>

Résultat de la recherche

Permet de copier le lien

8

▷ Modification date d'affiliation Art. 83

Votre espace de recherche

Recherche Client Recherche Lien Art83 GDPR

Recherche d'un Lien

Numéro d'encaissement : E5419354

Lancer la recherche

Nom	Prénom	Affiliation
BOURRIN	YOHAN	01/12/2022
BOURRIN	MATTHIEU	01/12/2022

1 (current)

Changer la date d'affiliation

Valider la date

Résultat de la recherche

- A partir de la saisie d'un numéro d'encaissement, cette fonction permet de retrouver les affiliés du contrat, mais surtout de vérifier la date d'affiliation erronée.
- La page de résultat propose un calendrier permettant de modifier cette date et de la valider.
- On peut ensuite relancer la recherche pour vérifier la prise en compte de la nouvelle date.

8

9

▷ Recherche consentements client

Votre espace de recherche

Recherche Client Recherche Lien Art83 GDPR

GDPR

Clé client : FR-AFER:PPHAFER_9209632

Consentement Historique consentement Permission

```
{
  "personId": "FR-AFER:PPHAFER_9209632",
  "platform": "AFER",
  "sourceApplication": "AFER&MOI",
  "question": {
    "consentAviva": false,
    "questionAviva": "GROUPE",
    "consentThirdParty": false,
    "questionThirdParty": "GIE_TIERS"
  },
  "date": "2022-02-21",
  "proof": {
    "type": null,
    "location": null,
    "url": null
  },
  "isFromStock": false
}
```

```
{
  "personId": "FR-AFER:PPHAFER_9209632",
  "sourceApplication": "AFER&MOI",
  "date": "2022-02-21",
  "permissionsAviva": {
    "sms": false,
    "post": false,
    "email": false,
    "phone": false,
    "pushNotification": false
  },
  "permissionsThirdParty": {
    "sms": false,
    "post": false,
    "email": false,
    "phone": false,
    "pushNotification": false
  },
  "permissionsMarketingStudies": {
    "sms": false,
    "phone": false,
    "email": false
  }
}
```

- A partir de la saisie d'une clé gdpr, ces 3 boutons permettent de restituer l'état du consentement du client, avec les médias associés (permission). Le bouton historique permet d'obtenir l'historique des consentements.
- La clé gdpr est composée d'un préfixe qui fait référence au back office du client, puis on y ajoute le numéro de client.

Voici les différentes clés avec un exemple :

Portail :
FR-PORTAIL:TTCLICLI_51610718

Abellie :
FR-GT:PPH_9123490

Afer :
FR-AFER:PPHAFER_9281508

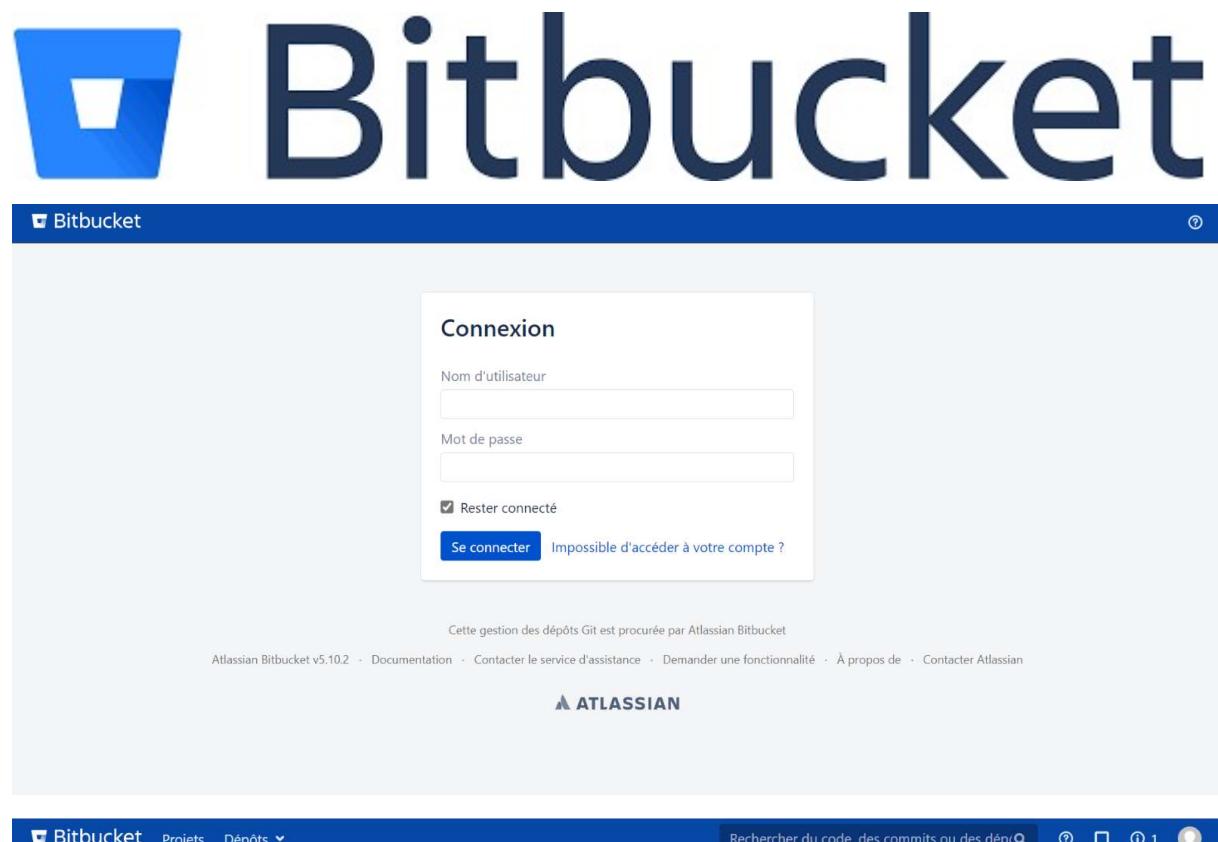
Eurofil :
FR-CIRARD:PEF01B0_000000001061126

Aviva Direct :
FR-MARK:MKBSIG_4298535002

8

10

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



A screenshot of the Bitbucket storage spaces page. The top navigation bar includes the Bitbucket logo, project and repository dropdowns, a search bar, and various icons. The main title is "Espaces de stockage" (Storage spaces). A sidebar on the left has a "Filtrer les répertoires par nom" (Filter repositories by name) input field and a list of repository names. The list includes: appdiag-back, appdiag-back-envs, appdiag-front, appged-back, client_service_back, client_service_back-envs, ged-documents-back, ged-documents-back-envs, hub-direct-back-envs, hub-direct-esign-front, hub-direct-esign-front-envs, and hub-dommage-back-envs. A "»" button is at the bottom of the list.

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



Welcome to Jenkins!

Username

Mot de passe

Sign in

Keep me signed in

CICD AVIVA

Rechercher ? KATIA BERTAUX SE DÉCONNECTER

Jenkins

Utilisateurs Historique des constructions Relations entre les builds Vérifier les empreintes numériques Bulk Builder Job Import Plugin Mes vues Open Blue Ocean Utilisation du disque

DEX CISO Security Scan APPROVED Zero Vulnerability **eclipse** JAKARTA EE Download DEX SHEET

Supporte JAVA 8/9 & JAKARTA EE
Portable et sans droits ADMIN
Serveurs et outils embarqués
Projets installés et IDE clef en main

Plateforme CICD [staging] d'Intégrations & Livraisons Continues CI/CD de la DSI Aviva France

DevSecOps by iT-LABS Aviva | : | : bernard.chatin@abeille-assurances.fr | : http://cicd-prod

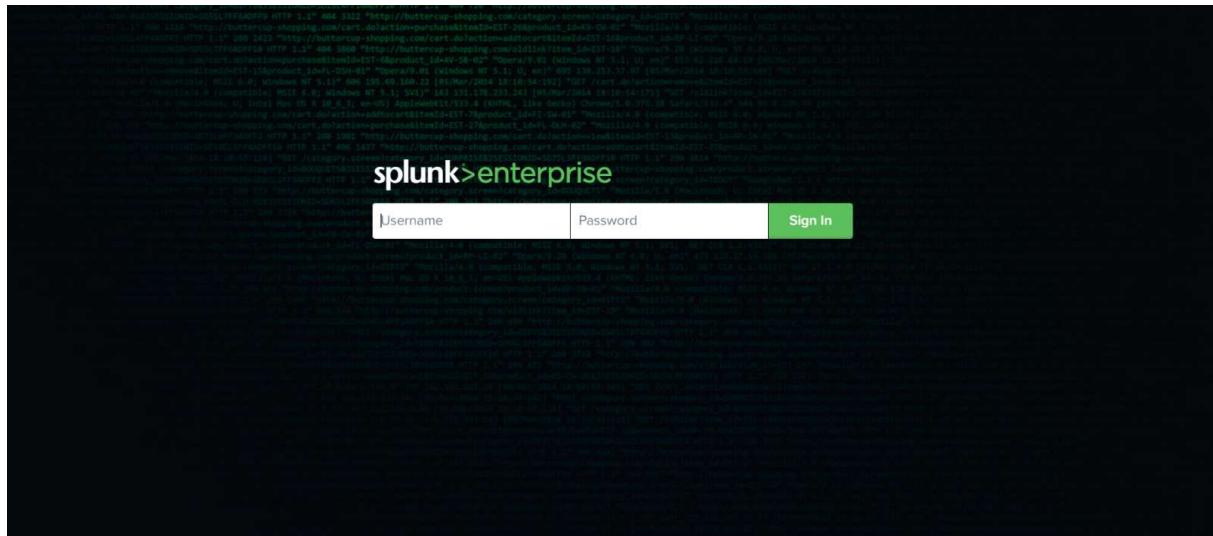
Splunk new URLs >> IN : http://intgsplunk | RC : http://rectsplunk | PX : http://pexpsplunk | EX : http://explsplunk

AVIVAFR-MA DIGFAC Tableau de bord Jenkins

Test Statistics Chart Test Trend Chart

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.



A screenshot of the Splunk Enterprise dashboard builder. The interface has a dark header bar with user information (KATIA BERTAUX, Messages, Paramètres, Activité, Aide, Rechercher) and a search bar. On the left, there's a sidebar titled "Apps" with a "Search & Reporting" icon. The main area is titled "Explorer Splunk Enterprise" and contains four cards: "Présentations des produits" (with a binoculars icon), "Manuel de recherche" (with a magnifying glass icon), "Manuel de Pivot" (with a document icon), and "Tableaux de bord et visualisations" (with a chart icon). Below these cards is a large, empty panel with a placeholder icon and the text "Choisissez un tableau de bord d'accueil".

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.

Jira Software

Bienvenue dans Jira Abeille Assurances

Nom d'utilisateur

Mot de passe

Retenir ma connexion sur cet ordinateur

Pas encore membre ? Pour demander un compte,
Veuillez contacter vos administrateurs Jira.

Connexion

You ne pouvez pas accéder à votre compte ?

System Dashboard

Introduction

Bienvenue sur Jira Abeille Assurances
Vous découvrez Jira ? Consultez le Guide utilisateur de Jira.

Qui me sont assignés

Aucun ticket ne vous est pour l'instant assigné. Passez une bonne journée !

Flux d'activité

Your Company JIRA

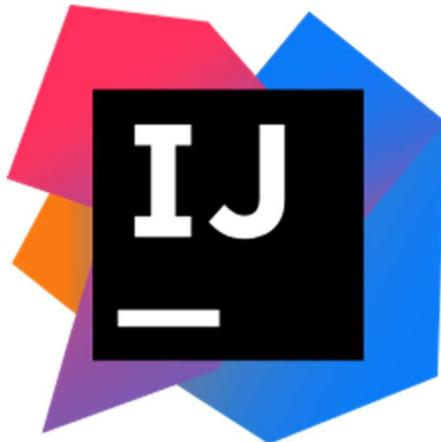
Aujourd'hui

[User icon] [File icon] a joint un fichier à JOP-49848 - Signature CSR pour l'environnement SAAS SMAX DEV par sectigo
• dev_esm_abeille-assurances_fr.cer
🕒 05/01/2023 11:24 Comment Vote Watch

[User icon] [File icon] a changé l'état à TECHNICAL VERIFICATION sur JOP-49840 - provocation de Failover des Clusters FW : vmpalofw000, ADC2FWO-SI1PL01, ADC2FWO-SI1CL01 pour test d'envois des logs des Fw Standby
🕒 05/01/2023 11:24 Comment Vote Watch

[User icon] [Question icon] a changé l'état à Traitement : Terminé sur IN-66831 - Incident sur mise à jour des coordonnées bancaires du contrat 13118344
🕒 Il y a quelques instants Comment Vote Watch

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 compléction 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 **boîte 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 débuggeur, 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).

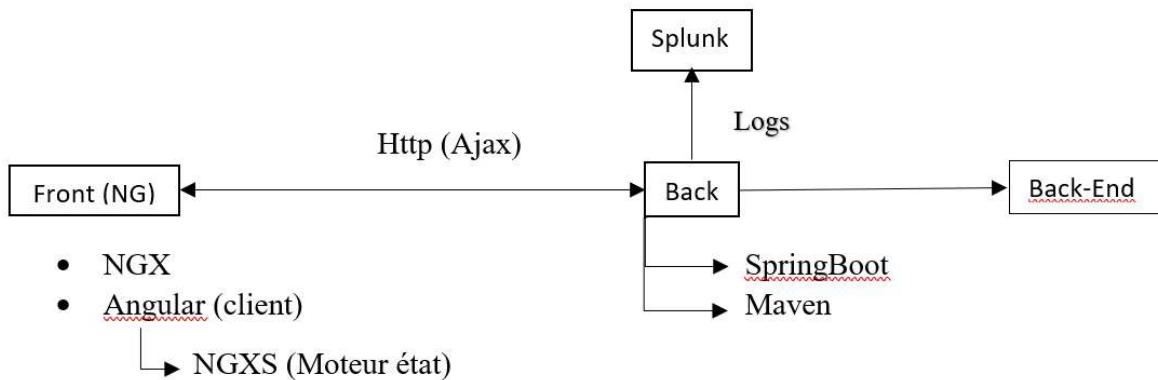
Un **Framework** est un ensemble de classes et de structures écrit dans un langage précis pour un langage précis et destiné à faciliter le développement dans ce langage.

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](#) ou [XHTML](#), [CSS](#), [JavaScript](#), [DOM](#), [XML](#), [XSLT](#), et surtout [l'objet 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](#) 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 peuvent être utilisés comme format afin d'emballer 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 Platform : 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

The screenshot shows the Abeille Assurances website's search interface. At the top, there is a navigation bar with links: Recherche Client, Recherche Lien, Art83, GDPR, and Référentiel produit (which is highlighted). Below the navigation bar, the page title is "Votre espace de recherche". There is a search input field labeled "Recherche d'un produit" and a text input field labeled "Numéro de série". At the bottom right of the search area is a yellow button labeled "Lancer la recherche".

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. »

The screenshot shows a MySQL Workbench interface with a results window titled "Résultat de requête". The SQL tab is selected, showing the query: "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';". Below the query, it says "Toutes les lignes extraites : 1 en 0,029 secondes". The results grid has columns: SÉRIE, PÉRIMÈTRE, CRÉATION_AUTO, SÉRIE_GROUPE_ELIGIBLE, and LIBELLÉ_SÉRIE. The data row is: 1 C90, 1 1, (null), ABEILLE EPARGNE PLURIELLE.

SÉRIE	PÉRIMÈTRE	CRÉATION_AUTO	SÉRIE_GROUPE_ELIGIBLE	LIBELLÉ_SÉRIE
1 C90	1 1	(null)		ABEILLE EPARGNE PLURIELLE

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

The screenshot shows a rendered table with three columns: Nom, Prénom, and Affiliation. The data rows are: BOURRIN, YOHAN, 01/12/2022 and BOURRIN, MATTHIEU, 01/12/2022.

Nom	Prénom	Affiliation
BOURRIN	YOHAN	01/12/2022
BOURRIN	MATTHIEU	01/12/2022

Partie 2.2 – Référentiel produit

Requête FE POLICYDICT :

```

select pod_l3_producttype Série,
key_longlabel Libellé_Série,

```

```

pod_l1_group Série_Groupe,
pod_pacte_fPacte,
pod_fis_pacte_fFis_Pacte,
pod_mono_fMono,
pod_multi_fMulti,
pod_ecplus_fEcplus_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. »

SÉRIE	LIBELLÉ_SÉRIE	SÉRIE_GROUPE	PACTE	FIS_PACTE	MONO	MULTI	ECPLUS_F	GSM	COLLECTIF
1 D12	ABEILLE RETRAITE PLURIELLE ENTREPRISE AFFILIE	1	1	(null)	(null)	(null)	1	1	

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

Nom	Prénom	Affiliation
BOURRIN	YOHAN	01/12/2022
BOURRIN	MATTHIEU	01/12/2022

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 »



Bienvenue

Votre espace de recherche

Recherche Client Recherche Lien Art83 GDPR Référentiel produit

Recherche d'un produit

Numéro de série

Export séries éligibles



Lancer la recherche

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

```
import {Component} from '@angular/core';
import {VnSpinnerComponent} from './_coreModule/components/vn-spinner/vn-spinner.component';
import build from '../build';

@Component({
  selector: 'app-module',
  template: '<router-outlet></router-outlet><ng-http-loader [entryComponent]="'vnSpinnerComponent'"></ng-http-loader>'
})
export class AppComponent {

  public vnSpinnerComponent = VnSpinnerComponent;

  constructor() {
    console.log(
      `Build Info:\n` +
      `Build Version: ${build.version}\n` +
      `Build Timestamp: ${build.timestamp}`
    );
  }
}
```

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

```

<!DOCTYPE html>
<html lang="fr-FR" data-resize-browser="">
<head>
    <title>Espace de signature - Abeille assurances</title>
    <base href="/">
    <link rel="stylesheet" media="all" href="./assets/css/bootstrap.min.css">
    <link rel="icon" type="image/ico" href="./assets/images/icons/abeille-assurances.ico"/>
    <link href="./assets/css/bundle.css" media="all" rel="stylesheet" />

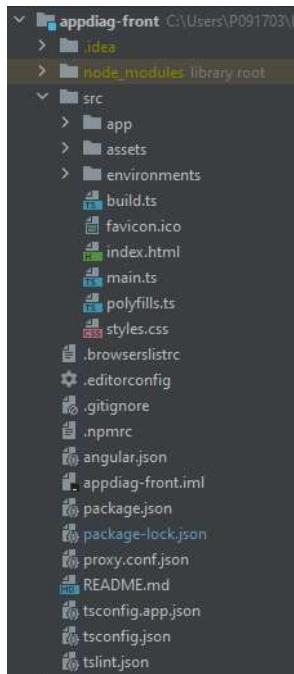
    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <meta name="apple-mobile-web-app-capable" content="yes"/>
    <meta name="msapplication-TileImage" content="./assets/images/favicon/aviva-windows-logo.png"/>
    <meta name="msapplication-TileColor" content="#ffd900"/>
    <meta name="title" content="Abeille-assurances"/>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
</head>
<body class="t3 noSubNav wholePageTabs js no-touch svg">
    <div>
        <div>
            <app-module></app-module>
        </div>
    </div>
</body>
<script src="http://code.jquery.com/jquery-2.1.1.js"></script>
</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.

```
"outputPath": "dist/appdiag-front",
```

appdiag-front.iml : Un fichier IML est un fichier de paramètres de module créé par IntelliJ IDEA, un environnement de développement intégré ([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](#) ou [.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.

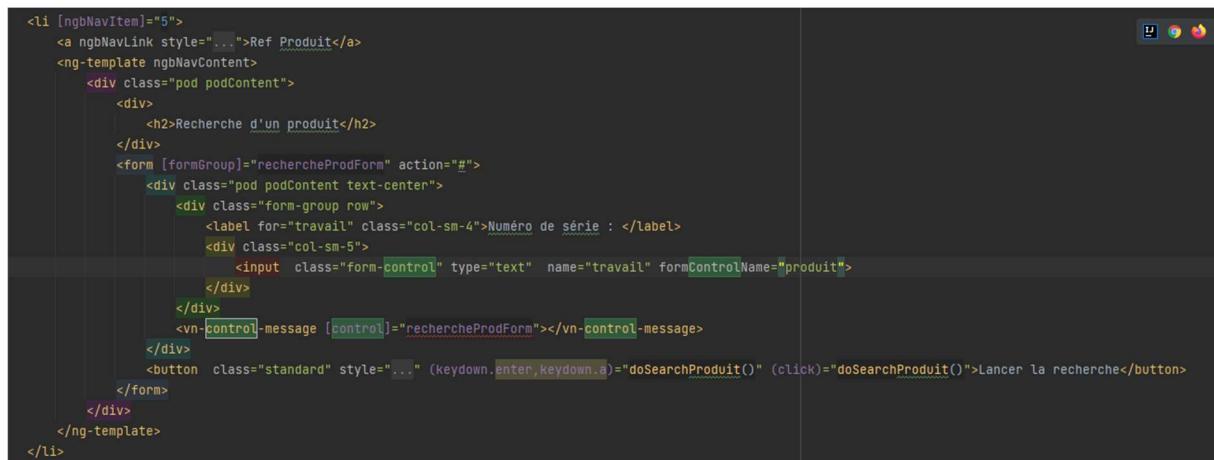
tsconfig.json : Configuration du compilateur TypeScript pour compiler l'application. Ce fichier est également utilisé par l'IDE afin de nous donner des informations utiles (mauvais type d'input par exemple).

tslint.json : TSLint est un outil d'analyse statique extensible qui vérifie le code [TypeScript](#) 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 :



```
<li [ngNavItem]="#5">
  <a ngbNavLink style="...">Ref Produit</a>
  <ng-template ngbNavContent>
    <div class="pod podContent">
      <div>
        <h2>Recherche d'un produit</h2>
      </div>
      <form [formGroup]="rechercheProdForm" action="#">
        <div class="pod podContent text-center">
          <div class="form-group row">
            <label for="travail" class="col-sm-4">Numéro de série : </label>
            <div class="col-sm-5">
              <input class="form-control" type="text" name="travail" formControlName="produit">
            </div>
          </div>
          <vn-control-message [control]="rechercheProdForm"></vn-control-message>
        </div>
        <button class="standard" style="..." (keydown.enter,keydown.KeyPress)="doSearchProduit()" (click)="doSearchProduit()">Lancer la recherche</button>
      </div>
    </ng-template>
  </li>
```

Dans le fichier .ts :

- Ajout d'un this pour les produits et création d'une nouvelle recherche de produit

```

        this.rechercheProdForm = this.fb.group( controlsConfig: {
            produit: ''
        });
    }

    rechercheForm: FormGroup;
    rechercheTForm: FormGroup;
    rechercheArtForm: FormGroup;
    rechercheGdprForm: FormGroup;
    modifyArtForm: FormGroup;
    rechercheProdForm: FormGroup;

```

- Ajout de proodata :

```

        artdata: any;
        proodata: any;

```

- Création de doSearchProduit

```

doSearchProduit() {
    if (this.doValidate(this.rechercheProdForm)) {
        const cri = this.rechercheProdForm.value;
        this.searchService.doSearchRefProduit(cri.produit).subscribe(next: (res) => {
            this.proodata = res.resultRef;
        });
    }
}

```

C:\Users\PO91711\IdeaProjects\appdiag-front\src\app\appDiagModule\services\app.diag.service.ts

Création de doSearchRefProduit pour faire fonctionner la méthode du dessus :

```

/*nouveau*/ doSearchRefProduit(référence): Observable<any> {
    const currentState = this.store.selectSnapshot(AppDiagAuthState);
    this.auth = currentState.token;
    const headers = new HttpHeaders( headers: {
        'Content-Type': 'application/json',
        'Authorization': this.auth
    });
    return this.http.post<any>( url: '/ma/appdiag/api/v1/RefProd/search/' + référence, body: null, options: {headers: headers});
}

```

Dans custom validator :

- Création de la function : isFillValue

```

function isFillValue(obj: any): boolean {
    return obj.value === undefined || obj.value === null || obj.value.length === 0;
}

```

- Création de la clause de validité onlyTenInFieldRequiredValidator :

```
static onlyTenInFieldRequiredValidator = (message) => {
  return (controlGroup) => {
    let controls = controlGroup.controls;
    if ( controls ) {
      let lengthFields = Object.keys(controls).filter(key=> !isEmptyValue(controls[key])).length;
      return (lengthFields >= 20) ? { 'vnOnlyTenInFieldRequiredValidator': { 'message': message } } : null;
    }
    return null;
  };
}
```

- 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 :

```
import javax.persistence.*;
import java.util.Objects;
7 usages
@Entity(name = "FE_SERIES_IC", schema = "CRM")
public class FeSeriesIC {

  5 usages
  private long iCNumSer;
  5 usages
  private String icNiveau;
  5 usages
  private String icAuto;
  5 usages
  private String icGroupe;

  @Id
  @Column(name = "IC_NUM_SER")
  public long getICNumSer(){return iCNumSer;}

  public void setICNumSer(long iCNumSer) {this.iCNumSer = iCNumSer;}

  @Basic
  @Column(name = "IC_NIVEAU")
  public String getIcNiveau() {return icNiveau; }

  public void setIcNiveau(String icNiveau) {this.icNiveau = icNiveau; }

  @Basic
  @Column(name = "IC_AUTO")
  public String getIcAuto() {return icAuto; }

  public void setIcAuto(String icAuto) {this.icAuto = icAuto;}
```

- Puis création d'un repository qui va prendre l'entité et va inclure la ou les requête SQL également :

```

@Repository
@Transactional
public interface FeProduitRepository extends CrudRepository<FeSeriesIC, Long> {

    @Query(value = "select ic_num_ser , ic_niveau , ic_auto , ic_groupe , key_longlabel " +
        "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'")
    List<FeSeriesIC> findAllByicNumSer(Long icNumSer);
}

```

- Mise en place d'un moyen de lancer un message qui spécifie qu'il faut écrire au moins 1 caractère :

|

Le champ doit être renseigné

- Mise en place d'un nombre maximal de caractère utilisés sur 20 :

produit

ro de série :

qqqqqqqqqqqqqqqqqqqqqqqqqqqq

- Finition des repository qui sont censés permettre au back d'utiliser les requêtes dont on a besoin sur la base de données.

```

1 usage
@Query(value = "select icNumSer , icNiveau , icAuto , icGroupe , key_longlabel " +
    "from fe_series_ic, fe_parameter " +
    "where key_class = 'POLICYDICT_OWNED' " +
    "and key_language = 'FR' " +
    "and icNumSer = key_shortlabel " +
    "and icNumSer = 'C90'")
List<FeSeriesIC> findAllByicNumSer(Long icNumSer);

1 usage
@Query(value = "select podL3Producttype , podFAM , podL1_group , pod_pacte_f , pod_fis_pacte_f , pod_mono_f , pod_multi_f , pod_ecplus_f , pod_crt_gsm , pod_crt_collectif " +
    "from fe_policydict, fe_parameter " +
    "where pod_l3_producttype = 'D12' " +
    "and key_class = 'POLICYDICT_OWNED' " +
    "and key_language = 'FR' " +
    "and pod_l3_producttype = key_shortlabel")
List<FeSeriesIC> findAllBypodID(Long podID);

```

- Création de l'interface du service permettant l'appel des requêtes :

```
4 usages 1 implementation  S083250 <supralol66_>
public interface SuiviActeService {
    1 usage 1 implementation  S083256 <supralol66_>
    List<FeSuiviActeIcv> recherche(Long id);
}
```

- Création du dit service :

```
import java.util.List;
@Service
public class RechercheProduitServiceImpl implements RechercheProduitService {

    2 usages
    @Autowired
    FeProduitRepository feProduitRepository;

    1 usage
    @Override
    public List<FeSeriesIC> findAllByiCNumSer(Long id) {
        List<FeSeriesIC> res = feProduitRepository.findAllByiCNumSer(id);
        return res;
    }
    @Override
    public List<FeSeriesIC> findAllByp0ID(Long id) {
        List<FeSeriesIC> res = feProduitRepository.findAllByp0ID(id);
        return res;
    }
}
```

- Création d'un modèle qui contient 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 :

```

public class FeProduitModel {

    2 usages
    private Long icNumServ;
    2 usages
    private String icNiveau;
    2 usages
    private String icAuto;
    2 usages
    private String icGroupe;
    2 usages
    private String pODL3Producttype;
    2 usages
    private String pODFAM;
    2 usages
    private String pODPacteF;
    2 usages
    private String pODFisPacteF;
    2 usages
    private String pODMonoF;
    2 usages
    private String pODMultiF;
    2 usages
    private String pODEcplusF;
    2 usages
    private String pODCrtCollectif;
    2 usages
    private String pODCrtGsm;

    public Long getIcNumServ() { return icNumServ; }

    public void setIcNumServ(Long icNumServ) { this.icNumServ = icNumServ; }

    public String geticNiveau() { return icNiveau; }

    public void seticNiveau(String icNiveau) { this.icNiveau = icNiveau; }

    public String getIcAuto() {return icAuto; }

    public void setIcAuto(String icAuto) { this.icAuto = icAuto; }
}

```

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

```

└ P091711
public DtoProduit() {
}

1 usage  └ P091711
public DtoProduit(FeSeriesIC feProduit) {
    ModelMapper modelMapper = new ModelMapper();
    SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    feProduitmodel.add(modelMapper.map(feProduit, FeProduitModel.class));

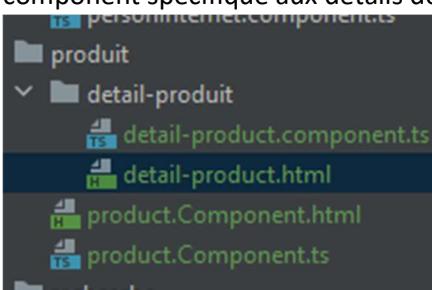
}

└ P091711
public List<FeProduitModel> getFeProduitmodel() {
    return feProduitmodel;
}

└ P091711
public void setFeProduitmodel(List<FeProduitModel> feProduitmodel) {
    this.feProduitmodel = feProduitmodel;
}

```

- 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 :

```
import {ActivatedRoute, Router} from '@angular/router';
import {AppDiagState, SaveSearch} from '../../../../../app.diag.state';
import {Store} from '@ngxs/store';
import {filter, first} from 'lodash';

@Component({
  selector: 'produit-component',
  templateUrl: './product.component.html'
})
export class ProductComponent extends VnAbstractComponent implements OnInit {

  icNumSer;
  data: any;

  constructor(private router: Router, private activatedRoute: ActivatedRoute, private store: Store) {
    super();
    if (this.router.getCurrentNavigation() && this.router.getCurrentNavigation()?.extras?.state?.icNumSer != null) {
      this.icNumSer = this.router.getCurrentNavigation().extras.state.icNumSer;
      this.store.dispatch(new SaveSearch({ data: {icNumSer: this.icNumSer}}));
    } else {
      this.icNumSer = this.store.selectSnapshot(AppDiagState).icNumSer;
    }
  }

  ngOnInit(): void {
    const currentState = this.store.selectSnapshot(AppDiagState);
    if (currentState) {
      this.data = first(filter(currentState.result.res, (e) => e.feProduit.icNumSer === this.icNumSer));
      //var datas = cloneDeep(currentState);
    }
  }
}
```

Produit.html :

```
<header-component>
  <div class="pod">
    <div class="podContent noBorder">
      <div class="pod podContent">
        <div>
          <h2>Détail du Client</h2>
        </div>
        <detail-produit-component [product]="data"></detail-produit-component>
      </div>
    </div>
  </div>
</header-component>
```

- On rajoute maintenant au fichier recherche.componenent.html le tableau permettant l'affichage des données trouvées lors de notre requête :

```

        </div>
        <button class="standard" style="..." type="submit" (keydown.enter,keydown.a)="doSearchProduit()"
            (click)="doSearchProduit()">Lancer la recherche
        </button>
    </form>
</div>
<div class="pod podContent">
    <ng-container *ngIf="proddata">
        <table class="tableRes">
            <thead>
                <tr>
                    <th scope="col">Série</th>
                    <th scope="col">Perimètre</th>
                    <th scope="col">Création</th>
                    <th scope="col">Groupe</th>
                    <th scope="col">Libellé</th>
                </tr>
            </thead>
            <tbody>
                <tr class="active-row"
                    *ngFor="let produit of proddata | slice: start: (page-1) * pageSize">
                    <td>{{produit.feProduitmodel.icNumSer}}</td>
                    <td>{{produit.feProduitmodel.icNiveau}}</td>
                    <td>{{produit.feProduitmodel.icAuto}}</td>
                    <td>{{produit.feProduitmodel.icGroupe}}</td>
                    <td>{{produit.feProduitmodel.icLibelle}}</td>
                </tr>
            </tbody>
        </table>
        <div class="d-flex justify-content-between p-2">
    
```

- Nous pouvons désormais accéder sur le site lors d'une recherche aux données du produit dont le numéro de série a été saisie :

Série	Perimètre	Création	Groupe	Libellé
	1	1	1	
	1	1	1	
	1	1	1	

« 1 (current) »

- Rajout d'un deuxième tableau permettant d'afficher une deuxième requête, utilisé au même moment que la première :

Série	Libellé	Groupe	Pacte	Fis_Pacte	Mono	Multi	Ecplus	GSM	Collectif

- Rajout d'une requête pour trouver le libelle dans la table FE_PARAMETER, avec le repository :

```

import com.aviva.appdiag.beans.entity.FeParameter;
import com.aviva.appdiag.beans.entity.FeSeriesIC;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import javax.transaction.Transactional;
import java.util.List;

2 usages
@Repository
@Transactional
public interface FeParameterRepository extends CrudRepository<FeSeriesIC, String> {

    1 usage
    @Query(value = "select keyLonglabel " +
        "from FeParameter " +
        "where keyClass = 'POLICYDICT OWNED' " +
        "and keyLanguage = 'FR' " +
        "and keyShortlabel = ?1")
    List<FeParameter> findAllByKeyShortLabel(String keyLonglabel);
}

```

Puis ajout d'une implémentation :

```

import com.aviva.appdiag.beans.entity.FeParameter;

import java.util.List;

3 usages 1 implementation
public interface ParameterService {
    1 usage 1 implementation
    List<FeParameter> findAllByKeyShortLabel(String criteria);
}

```

Puis d'un service :

```
import java.util.List;

@Service
public class ParameterServiceImpl implements ParameterService {
    1 usage
    @Autowired
    private FeParameterRepository feParameterRepository;

    1 usage
    @Override
    public List<FeParameter> findAllByKeyShortLabel (String id) {
        List<FeParameter> res = feParameterRepository.findAllByKeyShortLabel(id);
        return res;
    }
}
```

Création d'un nouveau Dto :

```
9 usages
public class DtoParameter {

    3 usages
    private FeProduitModel feProduitModel;

    public DtoParameter() {
    }

    1 usage
    public DtoParameter(FeParameter feParameter) {
        ModelMapper modelMapper = new ModelMapper();
        SimpleDateFormat dateFormat = new SimpleDateFormat(pattern: "dd/MM/yyyy");
        feProduitModel = modelMapper.map(feParameter, FeProduitModel.class);
    }

    public FeProduitModel getFeProduitModel() { return feProduitModel; }

    public void setFeProduitModel(FeProduitModel feProduitModel) { this.feProduitModel = feProduitModel; }
}
```

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

```
new "
@Operation(summary = "Permet de recuperer un produit")
@Parameter(name = "Authorization", description = "Username:password", required = true, allowEmptyValue = true, in = HEADER)
@PostMapping(value = @"/parameter", produces = MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
@ResponseBody
public ResponseEntity<Response> getParameterByCriteria(@RequestBody String parameterCriteria) {
    List<FeParameter> result = parameterService.findAllByKeyShortLabel(parameterCriteria );
    List<DtoParameter> resultParameter = new ArrayList<>();
    for(FeParameter singleResult : result){
        resultParameter.add(new DtoParameter(singleResult));
    }
    return new ResponseEntity<>(Response.addDtoResultParameter(resultParameter), HttpStatus.OK);
}
```

Malheureusement ce changement à entraîner 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 repository, 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 :

```

@GET
@MappingValue("produit", produces = MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
@ResponseBody
public ResponseEntity<Response> getProduitByCriteria(@RequestBody String produitCriteria) {
    List<FeSeriesIC> result = rechercheProduitService.findAllByICNumSer(produitCriteria);
    List<DtoProduit> resultProduit = new ArrayList<>();
    for(FeSeriesIC singleResult : result){
        List<FeParameter> resultParameter = parameterService.findAllByKeyShortLabel(produitCriteria);
        List<FePolicydict> resultPolicydict = policydictService.findAllByODID(produitCriteria);
        DtoProduit dtoProduit = new DtoProduit(singleResult);
        dtoProduit.getFeProduitModel().setIcLibelle(resultParameter.get(0).getKeyLonglabel());
        dtoProduit.getFeProduitModel().setPODL3Producttype(resultPolicydict.get(0).getpODL3Producttype());
        dtoProduit.getFeProduitModel().setPOD_L1_GROUP(resultPolicydict.get(1).getpODL1Group());
        dtoProduit.getFeProduitModel().setPODPacteF(resultPolicydict.get(2).getpODPacteF());/*
        dtoProduit.getFeProduitModel().setPODFisPacteF(resultPolicydict.get(3).getpODFisPacteF());
        dtoProduit.getFeProduitModel().setPODMonoF(resultPolicydict.get(4).getpODMonoF());
        dtoProduit.getFeProduitModel().setPODMultiF(resultPolicydict.get(5).getpODMultiF());
        dtoProduit.getFeProduitModel().setPODEcplusF(resultPolicydict.get(6).getpODEcplusF());
        dtoProduit.getFeProduitModel().setPODCrtCollectif(resultPolicydict.get(7).getpOCCrtCollectif());
        dtoProduit.getFeProduitModel().setPODCrtGsm(resultPolicydict.get(8).getpOCCrtGsm());*/
        resultProduit.add(dtoProduit);
    }
}

```

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 :

```

@ResponseEntity
public ResponseEntity<Response> getProduitByCriteria(@RequestBody String produitCriteria) {
    List<FeSeriesIC> result = rechercheProduitService.findAllByICNumSer(produitCriteria);
    List<DtoProduit> resultProduit = new ArrayList<>();
    List<FeParameter> resultParameter = parameterService.findAllByKeyShortLabel(produitCriteria);
    List<FePolicydict> resultPolicydict = policydictService.findAllByODID(produitCriteria);
    for(int essaie = 0 ; essaie < result.size(); essaie++){
        DtoProduit dtoProduit = new DtoProduit(result.get(essaie));
        dtoProduit.getFeProduitModel().setIcLibelle(resultParameter.get(0).getKeyLonglabel());
        dtoProduit.getFeProduitModel().setPODL3Producttype(resultPolicydict.get(0).getpODL3Producttype());
        dtoProduit.getFeProduitModel().setPOD_L1_GROUP(resultPolicydict.get(0).getpODL1Group());
        dtoProduit.getFeProduitModel().setPODPacteF(resultPolicydict.get(0).getpODPacteF());
        dtoProduit.getFeProduitModel().setPODFisPacteF(resultPolicydict.get(0).getpODFisPacteF());
        dtoProduit.getFeProduitModel().setPODMonoF(resultPolicydict.get(0).getpODMonoF());
        dtoProduit.getFeProduitModel().setPODMultiF(resultPolicydict.get(0).getpODMultiF());
        dtoProduit.getFeProduitModel().setPODEcplusF(resultPolicydict.get(0).getpODEcplusF());
        dtoProduit.getFeProduitModel().setPODCrtCollectif(resultPolicydict.get(0).getpOCCrtCollectif());
        dtoProduit.getFeProduitModel().setPODCrtGsm(resultPolicydict.get(0).getpOCCrtGsm());
        resultProduit.add(dtoProduit);
    }
    return new ResponseEntity<>(Response.addDtoResultProduit(resultProduit), HttpStatus.OK);
}

```

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

```

<tr>
    <th scope="col">Série </th>
    <th scope="col">Libellé </th>
    <th scope="col">Groupe </th>
    <th scope="col">Pacte </th>
    <th scope="col">Fis_Pacte </th>
    <th scope="col">Mono </th>
    <th scope="col">Multi </th>
    <th scope="col">Ecplus </th>
    <th scope="col">GSM </th>
    <th scope="col">Collectif </th>
</tr>
</thead>
<tbody>
    <tr class="active-row"
        *ngFor="let policydict of proddata">
        <td>{{policydict.feProduitModel.podl3Producttype}}</td>
        <td>{{policydict.feProduitModel.icLibelle}}</td>
        <td>{{policydict.feProduitModel.pod_L1_GROUP}}</td>
        <td>{{policydict.feProduitModel.podpacteF}}</td>
        <td>{{policydict.feProduitModel.podfisPacteF}}</td>
        <td>{{policydict.feProduitModel.podmonoF}}</td>
        <td>{{policydict.feProduitModel.podmultiF}}</td>
        <td>{{policydict.feProduitModel.podecplusF}}</td>
        <td>{{policydict.feProduitModel.podcrtCollectif}}</td>
        <td>{{policydict.feProduitModel.podcrtGsm}}</td>
    </tr>
</tbody>

```

- Ajout d'un bouton de téléchargement :

```

<input class="form-control" type="text" name="travail"
    formControlName="produit" minlength="1" maxlength="20">
</div>
<div class="col" style="text-align: right">
    <a type="button" style="vertical-align: center" class="button-download" [href]="downloadUrl" download="Export series IC.csv"> </a>
</div>
</div>
<div>
    <small>Le bouton "Rechercher" n'a pas d'effet sur ce bouton de téléchargement.</small>
</div>

```

- 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 :

```

public interface FeProduitDLRepository extends CrudRepository<FeSerieIC, Long>, FeProduitDLSpecification {
    @Query("select seri " +
        "from FeSeriesIC seri, FeParameter para " +
        "where para.keyClass = 'POLICYDICT_OWNED' " +
        "and para.keyLanguage = 'FR' " +
        "and seri.icNumSer = para.keyShortlabel " +
        "order by seri.icNumSer ")
    List<FeSeriesIC> findAllICNumSer();
}

```

Et une autre requête pour le libelle de FeParameter :

```
1 usage
@Query(value = "select para " +
        "from FeParameter para, FeSeriesIC ic " +
        "where para.keyClass = 'POLICYDICT_OWNED' " +
        "and para.keyLanguage = 'FR' " +
        "and para.keyShortlabel = ic " +
        "order by ic.icNumSer ")
List<FeParameter> findAllParameter();
```

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

```
new *
@Operation(summary = "Permet de recuperer un produit")
@Parameter(name = "Authorization", description = "Username:password", required = true, allowEmptyValue = true, in = HEADER)
@GetMapping(value = @"/allProduit", produces = MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
@ResponseBody
public ResponseEntity<Response> getProduit() {
    List<FeSeriesIC> result = produitDLService.findAllicNumSer();
    List<DtoProduit> resultProduit = new ArrayList<>();
    List<FeParameter> resultParameter = parameterService.findAllParameter();
    for(FeSeriesIC singleResult : result){
        DtoProduit dtoProduit = new DtoProduit(singleResult);
        dtoProduit.getFeProduitModel().setIcLibelle(resultParameter.get(0).getKeyLonglabel());
        resultProduit.add(dtoProduit);
    }
    return new ResponseEntity<>(Response.addDtoResultProduit(resultProduit), HttpStatus.OK);
}
```

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

Ensute on créé dans le front un nouveau service :

```
    return this.http.post<any>( url: '/ma/appdiag/api/v1/produit' , {reference}, {options: {headers: headers}} )
}
doSearchAllProduit(): Observable<any> {
    const currentState = this.store.selectSnapshot(AppDiagAuthState);
    this.auth = currentState.token;
    const headers = new HttpHeaders({ headers: {
        'Content-Type': 'application/json',
        'Authorization': this.auth
    }});
    return this.http.get<any>( url: '/ma/appdiag/api/v1/allProduit' , { options: {headers: headers}} );
}
```

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

```
doSearchAllProduit() {
    this.searchService.doSearchAllProduit().subscribe(next: (res) =>{
        this.allProduit = res.resultProduit;
        console.log(res);
        const currentState = this.store.selectSnapshot(AppDiagState);
        this.store.dispatch(new SaveSearch({ data: {resProduit: res}}));
        this.downloadFile();
    });
}
```

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

```
downloadFile() {
    let currentRow = [];
    const csvHeader = ['Serie', 'Perimetre', 'Creation Auto', 'Groupe', 'Libelle'];
    const formattedData = this.allProduit.map(row => {
        currentRow = [];
        currentRow.push(row.feProduitModel.icNumSer);
        currentRow.push(row.feProduitModel.icNiveau);
        currentRow.push(row.feProduitModel.icAuto);
        currentRow.push(row.feProduitModel.icGroupe);
        currentRow.push(row.feProduitModel.icLibelle);
        return currentRow;
    });
    let csvContent = 'data:text/csv;charset=utf-8,';
    formattedData.unshift(csvHeader);
    formattedData.forEach(rowArray => {
        const row = rowArray.join(';');
        csvContent += row + '\r\n';
    });
    this.downloadUrl = this.sanitizer.bypassSecurityTrustUrl(encodeURI(csvContent));
}
```

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



Serie	Perimetre	Creation Aut Groupe	Libelle
ADA	1		ASSURANCE DEPENDANCE
ADN	1		ASSURANCE DEPENDANCE
AED	0		ASSURANCE DEPENDANCE
AEM V	1		ASSURANCE DEPENDANCE
AE4 V	1		ASSURANCE DEPENDANCE
AE5 V	1		ASSURANCE DEPENDANCE
AGR	1		ASSURANCE DEPENDANCE
ANA	1		ASSURANCE DEPENDANCE
AND	1		ASSURANCE DEPENDANCE
APC C	1		ASSURANCE DEPENDANCE
APC V	1		ASSURANCE DEPENDANCE
APR	0		ASSURANCE DEPENDANCE
ARE	0		ASSURANCE DEPENDANCE
A02	1		ASSURANCE DEPENDANCE
B03	1		ASSURANCE DEPENDANCE
B04	1		ASSURANCE DEPENDANCE

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 crée, 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 :

```
2 usages
private Long dapId;
2 usages
private String dapCatRoot;
2 usages
private String dapCat;
2 usages
private String dapLibelle;
2 usages
private String dapRefP8;
2 usages
private String dapClasseP8;
2 usages
private String dapPublie;
2 usages
private Date dapPublicationD;
2 usages
private String dapCatLibelle;

@Id
@Column ( name= "DAP_ID")
public Long getdapId() { return dapId; }

public void setdapId(Long dapId) {this.dapId = dapId;}
```

Modèle :

```
public class FeDocAdherentP8Model {  
  
    2 usages  
    private String dapCatRoot;  
    2 usages  
    private String dapLibelle;  
    2 usages  
    private String dapRefP8;  
    2 usages  
    private String dapClasseP8;  
    2 usages  
    private String dapPublie;  
    2 usages  
    private Date dapPublicationD;  
  
    □ public String getdapCatRoot() {  
        return dapCatRoot;  
    }  
  
    □ public void setdapCatRoot(String dapCatRoot) {  
        this.dapCatRoot = dapCatRoot;  
    }  
  
    □ public String getdapLibelle() { return dapLibelle; }  
  
    □ public void setdapLibelle(String dapLibelle) { this.dapLibelle = dapLibelle; }  
}
```

Repository :

```
import com.aviva.appdiag.beans.entity.FeDocAdherentP8;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.CrudRepository;  
import org.springframework.stereotype.Repository;  
  
import javax.transaction.Transactional;  
import java.util.List;  
  
2 usages  
@Repository  
@Transactional  
public interface FeDocAdherentP8Repository extends CrudRepository<FeDocAdherentP8, String> {  
  
    1 usage  
    @Query(value ="select distinct (doc.dapCatRoot) " +  
          "from FeDocAdherentP8 doc ")  
    List<String> findAllCatRoot();  
}
```

```
import com.aviva.appdiag.beans.entity.FeDocAdherentP8;

import java.util.List;

3 usages 1 implementation
public interface RechercheAdherentCatService {
    1 usage 1 implementation
    List<String> findAllCatRoot();

}
```

Service :

Service implement :

```
import java.util.List;

@Service
public class RechercheAdherentCatServiceImpl implements RechercheAdherentCatService {
    1 usage
    @Autowired
    private FeDocAdherentP8Repository feDocAdherentP8Repository;

    1 usage
    @Override
    public List<String> findAllCatRoot() {
        List<String> res = feDocAdherentP8Repository.findAllCatRoot();
        return res;
    }
}
```

Dto :

```
ages
public class DtoCatRoot {

    3 usages
    private String feDocAdherentP8Model;

    public DtoCatRoot() {
    }

    1 usage
    public DtoCatRoot(String feDocAdherentP8) {
        this.feDocAdherentP8Model = feDocAdherentP8;
    }

    public String getFeDocAdherentP8Model() { return feDocAdherentP8Model; }

    public void setFeDocAdherentP8Model(String feDocAdherentP8Model) {
        this.feDocAdherentP8Model = feDocAdherentP8Model;
    }
}
```

Contrôleur :

```
@Operation(summary = "Permet de recuperer un produit")
@Parameter(name = "Authorization", description = "Username:password", required = true, allowEmptyValue = true, in = HEADER)
@GetMapping(value = @v"/adherent", produces = MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
@ResponseBody
public ResponseEntity<Response> getCatRoot() {
    List<String> result = rechercheAdherentCatService.findAllCatRoot();
    List<DtoCatRoot> resultCatRoot = new ArrayList<>();
    if(result != null){
        LOGGER.info(result.size());
    }
    else {
        LOGGER.info("Y a rien");
    }

    for (String singleResult : result) {
        DtoCatRoot dtoCatRoot = new DtoCatRoot(singleResult);
        resultCatRoot.add(dtoCatRoot);
    }
    return new ResponseEntity<>(Response.addDtoCatRoot(resultCatRoot), HttpStatus.OK);
}
```

- Et enfin dans le front on crée une nouvelle partie de la page principale :

```
</div>
<form [FormGroup]="rechercheAdP8Form" action="#">
    <div class="pod podContent text-center">
        <div class="form-group row">
            <label for="travail" class="col-sm-4">dap_cat_root : </label>
            <div class="col-sm-5">
                <select>
                    <option value="default">----</option>
                    <option *ngFor="let root of adhData" [value]="{{root.feDocAdherentP8Model.dapCatRoot}}"
                           {{root.feDocAdherentP8Model}}>
                    </option>
                </select>
            </div>
        </div>
    </div>
```

- 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 :



Welcome to Jenkins!

Sign in

Keep me signed in

	ABEILLE APPDIAG BACK DELIVER INTG	1 j 18 h - #88	S. O.	4 mn 26 s		Jenkins	
	ABEILLE APPDIAG BACK DELIVER PREX	1 j 18 h - #54	S. O.	2 mn 4 s		Jenkins	
	ABEILLE APPDIAG BACK DELIVER PROD	7 j 13 h - #9	S. O.	1 mn 25 s		Jenkins	
	ABEILLE APPDIAG BACK DELIVER RECT	1 j 18 h - #46	S. O.	1 mn 54 s		Jenkins	
	ABEILLE APPDIAG NG BUILD	1 j 18 h - #121	S. O.	2 mn 8 s		Jenkins	
	ABEILLE APPDIAG NG DELIVER INTG	1 j 18 h - #53	S. O.	17 s		Jenkins	
	ABEILLE APPDIAG NG DELIVER PREX	1 j 18 h - #26	S. O.	14 s		Jenkins	
	ABEILLE APPDIAG NG DELIVER PROD	7 j 14 h - #5	S. O.	14 s		Jenkins	
	ABEILLE APPDIAG NG DELIVER RECT	1 j 18 h - #35	S. O.	16 s		Jenkins	

Maven project ABEILLE | APPDIAG | BACK | DELIVER | INTG

INTEGRATION

Changements récents

	Utilisation du disque
Job	384 KB
All builds	384 KB
Locked builds	-
All workspaces	272 MB
Slave workspaces	136 MB
Non-slave workspaces	136 MB

Subprojects

Unresolved

sdlc_pipeline

Liens permanents

- Dernier build (#88), il y a 1 j 18 h
- Dernier build stable (#88), il y a 1 j 18 h
- Dernier build avec succès (#88), il y a 1 j 18 h
- Last completed build (#88), il y a 1 j 18 h

Maven project ABEILLE | APPDIAG | BACK | DELIVER | INTG

Ce build nécessite des paramètres :

PUBLIER_SERVICE
Livré le service dans le cluster Rancher

Branch
Choisissez la branche projet à construire.

Historique des builds

	find	tendance
#88	31 janv. 2023 15:53	384 KB
Atom feed des builds Atom feed des échecs		

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.