



2024

Rapport de stage



Christopher Chambonnet

En stage chez AURIOR BESTT

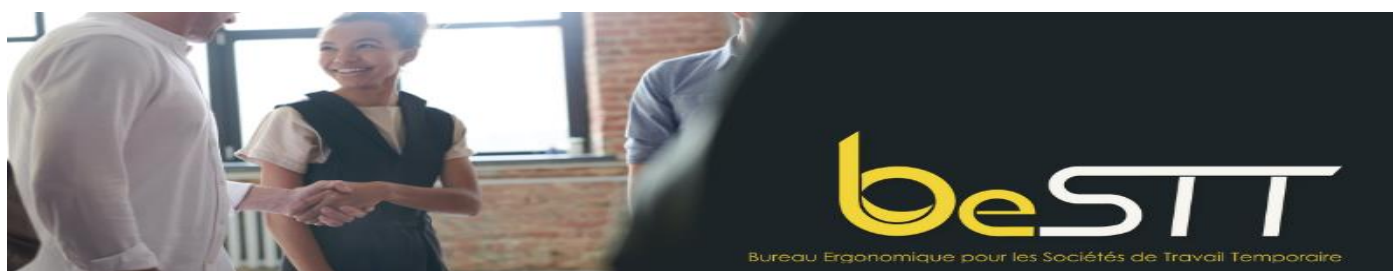


Session AFPA

2023 - 2024

Christopher Chambonnet

Développement Logiciel



Sommaire

<i>Remerciements</i>	<i>p.2</i>
<i>Présentation de la société</i>	<i>p.3</i>
<i>Description du logiciel BeSTT</i>	<i>p.4</i>
<i>Comment est développé BeSTT</i>	<i>p.5</i>

Projet : Application de prospection d'entreprises

- Présentation du projet	p.6
→ Tests de requêtes avec post-man.....	p.7 - 9
→ les structures	p.10 - 11
→ charger une liste au format Csv	p.12 - 13
→ Les variables	p.14
→ Les procédures	p.15 - 17
→ Les requêtes Html	p.18 - 20
→ Tableau associatif	p.21
→ Conditions	p.22 - 24
→ Table et liaison	p.25
-Autre : (hors-projet) :	
→ Exemple de requête classique	p.27 - 29

Remerciements :

*Avant toute chose, je tenais à remercier la société **Aurior** et l'ensemble de l'équipe pour m'avoir si bien accueilli et encadré durant ses 12 semaines.*

*Ce stage m'a permis d'apprendre **un nouveau langage** de programmation que j'apprécie et que je n'avais pas rencontré au sein de ma formation.*

*Je remercie tout particulièrement mon tuteur de stage **Eric Caillerez**, pour ses compétences et ses conseils lors des revues de code.*

*Il en est de même pour **M. LINOSSIER Raphael**, fondateur d'Aurior et co-fondateur de BeSTT, Que je remercie pour m'avoir accepté en stage.*

Ainsi que toute l'équipe pour leurs connaissances, leur pédagogie et l'attention qu'ils m'ont porté tout au long de ce stage afin de mener à bien mon projet.

*Je remercie le centre **AFPA d'Amiens** de m'avoir permis d'effectuer cette formation et de faire en sorte que chaque stagiaire puisse y accomplir celle de son choix dans les meilleures dispositions possibles.*

*Et enfin, je tenais aussi à remercier les formateurs du centre AFPA, **Mihaela Nobrega** pour sa forte implication, son écoute et son accompagnement au quotidien, ainsi qu'à **François-Régis Caumartin**, qui grâce à leur patience et leur pédagogie, ont réussi à nous transmettre leur passion ainsi que toutes les clés pour débiter notre métier de développeur de la meilleure manière possible.*

Présentation de la société AURIOR et de son logiciel

L'histoire de la société :

La société **Aurior** fût créée en Mars 2014 avec à sa tête **M. LINOSSIER Raphaël** qui en est le président, et se situe en région parisienne, dans le 11ème arrondissement. Son activité est « l'édition de logiciels applicatifs » (définition du code APE). Et dirige aussi la société d'intérim « **J4S** » à Paris.

Avant 2014, **M. LINOSSIER** recherchait vainement une réelle solution logiciel depuis près de 15 ans pour son agence de travail temporaire. Jusqu'au jour où il rencontra **Mr.Eric CAILLEREZ**, développeur dans le secteur de l'intérim depuis 25 ans.

De leur association est né le logiciel **BeSTT** (**B**ureau **E**rgonomique dédié aux **S**ociétés de **T**ravail **T**emporaire). Cette solution fût d'abord exclusivement développée pour la société J4S, puis une commercialisation du logiciel s'est mise en place à partir de 2014.

Depuis Août 2016, un établissement secondaire d'**Aurior** s'est établi à **Amiens**, avec à sa tête **Eric CAILLEREZ**, directeur général.



Bureau Ergonomique pour les Sociétés de Travail Temporaire

Description du logiciel BeSTT :

*Le logiciel **BeSTT** (**B**ureau **E**rgonomique pour les **S**ociétés de **T**ravail **T**emporaire) est un logiciel dédié aux agences intérim. Ses nombreuses fonctionnalités, pensées pour et par l'intérim et le recrutement, permettent de faciliter le quotidien des permanents dans toutes leurs tâches quotidiennes.*

Le logiciel BeSTT permet également une centralisation et une dématérialisation des données et informations. BeSTT gère les candidats, les intérimaires, les clients et la prospection.

Il est très simple d'utilisation pour la saisie des données des salariés et également des informations juridiques des clients grâce car le logiciel est en lien avec la base du greffe (Siret, raison sociale, Numéro de TVA).

***Il permet l'édition rapide** des contrats, des factures, des paies, des acomptes, des déclarations sociales et de la mutuelle, permettant un gain de temps quotidien considérable. BeSTT permet aussi de suivre toutes les statistiques internes et de les comparer aux années précédentes ou de calculer les marges précisément.*



Comment est développé BeSTT ?

BeSTT est développé en **WLangage**, qui est un langage de type « programmation procédurale » de 4ème génération, orienté objet. Il est inclus dans les outils de développement Windev (et également Webdev et Windev mobile).

Les caractéristiques essentielles d'un langage de 4ème génération sont un langage facile à apprendre, qui accélère le travail de programmation, dans lequel il y a un ensemble d'outils permettant de construire rapidement des menus ainsi que des formulaires électroniques et des rapports. Le Wlangage correspond en tout point à ces caractéristiques.



Windev, **Webdev** et **Windev mobile** sont des outils de développement édités par la société française PC SOFT, créée en 1984 et basée à Montpellier. Elle est spécialisée dans ce que l'on appelle les ateliers de génie logiciel (AGL).

Elle édite également **HyperFileSQL** (HFSQL), un moteur de base de données intégré aux outils Windev, Webdev et Windev Mobile.

Le projet : Logiciel de prospection d'entreprises

Présentation :

Détection de recruteurs :

Le but est d'interroger les données de « **la candidathèque** » (Api gérée en interne) pour lister des recruteurs potentiels et les fournir aux entreprises de travail temporaires clientes de **BeSTT** en fonction de leur géolocalisation et de leur secteur d'activité.

Mais avant tout ... Qu'est-ce qu'une API ?

Une API permet de rendre disponibles les données ou les fonctionnalités d'une application existante afin que d'autres applications les utilisent.

Utiliser une API permet donc d'utiliser un programme existant plutôt que de le redévelopper. C'est donc un grand gain de temps à la clé.

Premier problème ...

Cependant, « **la candidathèque** », regroupant les offres d'emploi de France-Travail et de notre ETT partenaire J4S, **manque de détails concernant les entreprises.**

C'est pourquoi je dois faire appel à une seconde Api (aussi gérée en interne) pour compléter les informations de la candidathèque transmises sur l'interface.

Cette seconde Api est « **Vérifie** ». Elle ne Regroupe que les entreprises mais les informations qui nous parviennent sont plus détaillées et fréquemment mis à jour.

Comment procéder ...

Je dois donc dans un premier temps créer une requête HTTP pour appeler l'Api Candidathèque et par le biais d'une clé, ne récupérer que les offres susceptibles de ne pas avoir été prospectés : celles de **France-Travail**.

Je dois ensuite aller chercher les informations manquantes par le biais d'une autre requête sur « **Vérifie** » en cohérence avec les résultats obtenus sur l'Api « **Candidathèque** ».

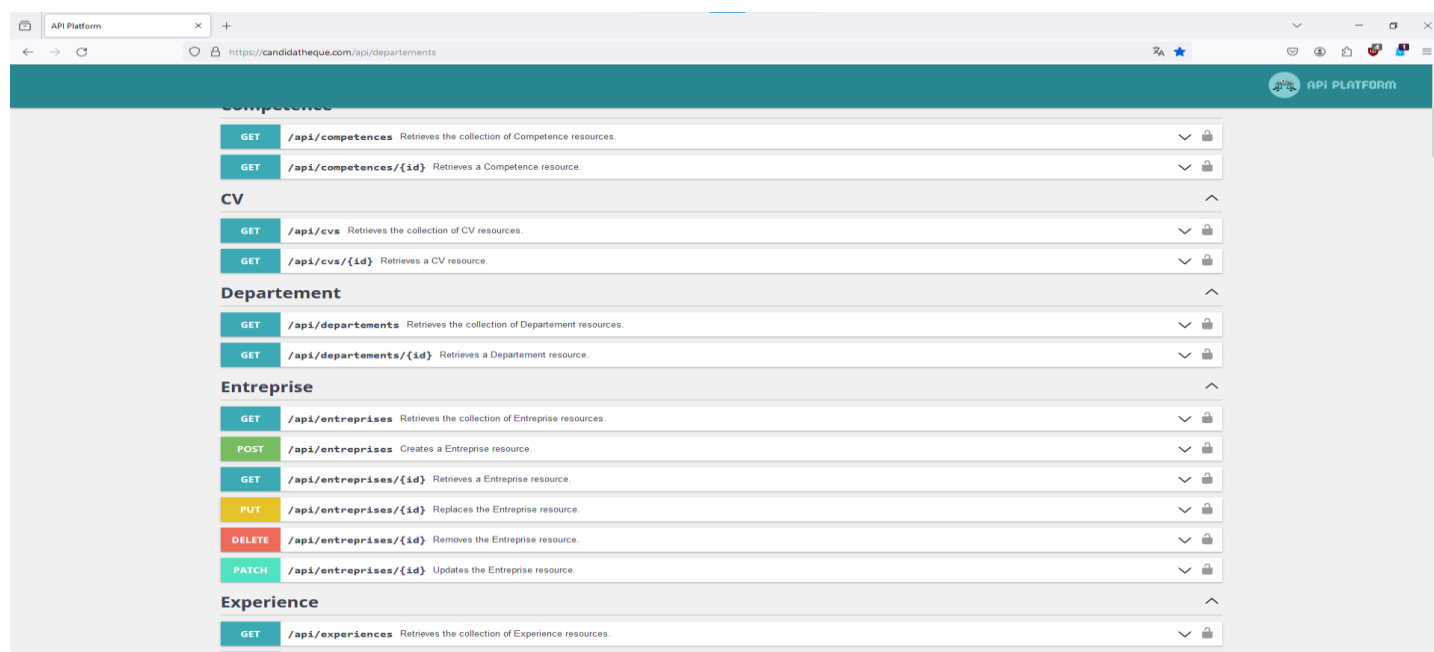
La recherche d'entreprises sur l'interface se fera soit en sélectionnant un département, soit en saisissant un secteur d'activité ou les deux.

Tests de requêtes avec Postman

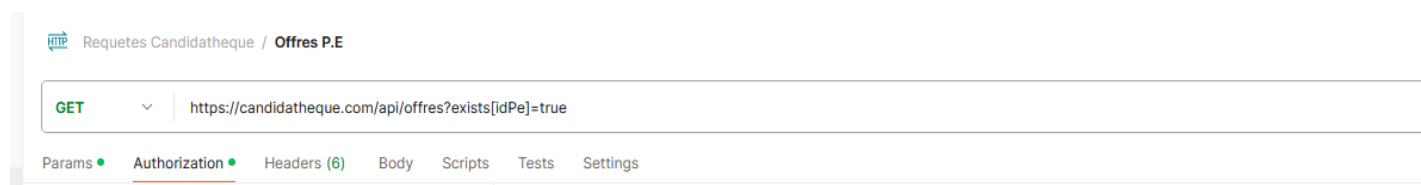
Tester une Api en amont nécessite une application dédiée. Ici j'utilise **Postman**. Il s'agit d'un outil de test et de développement d'API conçu pour envoyer des requêtes du côté client au serveur Web et recevoir une réponse du backend. Les informations reçues avec la réponse sont déterminées par les données envoyées avec la demande. Les variables et les environnements peuvent être utilisés pour stocker et réutiliser des données dans mes cas de test, telles que les URL, les paramètres, les en-têtes, les jetons, etc...

Mes collègues en charge de l'Api Candidathèque ont définis des filtres permettant l'accès rapide aux informations par le biais de clés dans l'Url.

Ces clés sont visualisables sur **Api-Platform** :



La première qui nous intéresse ici est, comme dit précédemment, celle permettant de ne récupérer que les offres de **France-Travail**. Que voici :



« [https://candidatheque.com/api/offres?exists\[idPe\]=true](https://candidatheque.com/api/offres?exists[idPe]=true) »

Si j'avais voulu voir toutes les offres, c'est la même requête mais sans « ?exists[idPe]=true »

Tests de requêtes avec Postman (suite)

La requête précédente rend ce type résultat (en Json) :

```
Body Cookies Headers (15) Test Results
Pretty Raw Preview Visualize JSON
1 [
2   {
3     "id": 84688,
4     "titre": "Technicien / Technicienne instrumentation (H/F)",
5     "datePublication": "2024-04-26T11:29:55+02:00",
6     "description": "Rattaché au Directeur Technique, vous prenez en charge la partie EIA (Electricité Instrumentation et Automatisme) des affaires qui vous sont cc
    projet jusqu'à sa réception par le client. En étroite collaboration avec le chef de projet, vous conseillez et assurez les suivis contractuels et technique
    auprès du client et des sous-traitants ; tout en coordonnant les activités entre les différents intervenants de bureau d'études, de conception, d'installat
    en service. Vous êtes garant des délais et des budgets qui vous sont alloués et respectez les exigences du système de Management Qualité.\n\nDans ce cadre,
    :\n-L'analyse des spécifications techniques client,\n-Le choix de l'instrumentation en fonction du Process, des contraintes mécaniques et des contraintes e
    des offres fournisseurs et l'achat de matériel,\n-La réalisation des plans d'ensembles et de détails (listes, plans de BJ, schéma de boucles, ...),\n-La ge
    ou du contrôle de commande,\n-Le dimensionnement des équipements (études d'armoires et de coffrets)\n-L'implantation et le suivi du montage des équipements
    câbles ) sur les installations.\n\nIssu d'une formation technique au minimum Bac+2, une première expérience en EIA est un plus.\nVous avez un niveau d'a
    utiliser les logiciels de dessin (Autocad ) ainsi que le Pack Office.\n\nVous êtes rigoureux(-se) et organisé(e). Votre énergie et votre aptitude naturelle
    permettront de vous intégrer rapidement au sein de notre structure en plein développement.\n\nCe poste s'exerce au sein du bureau d'études de notre site de
    dizaine de kilomètres d'Angers et comporte des déplacements ponctuels à l'étranger.\n\nDans le cadre de la diversité, nous étudions à compétences égales t
    de personnes en situation de handicap.",
7     "metier": {
8       "libelle": "Ingénieur / Ingénieure instrumentation en industrie"
9     },
10    "postules": [],
11  },
12  ]
```

Je récupère donc le nom de l'entreprise, le nom et numéro de département, le nom de ville, son code postal et le libelle de métier. Le reste ne me servira pas.

Un petit tour sur Vérifie via Postman :

GET entreprises	10	"libelleVoieEtablissement": "MANIHI COTE MONTAGNE TUAMOTU",
ETT	11	"codePostalEtablissement": "98770",
Sans ETT (+dep)	12	"libelleCommuneEtablissement": "MANIHI",
Sans ETT +dep +Activité(...)	13	"codeCommuneEtablissement": "98727",
	14	"dateDebut": "2009-05-27",
	15	"etatAdministratifEtablissement": "F",
GET Siret	16	"activitePrincipaleEtablissement": "32.12Z",
GET entreprises (PerPage)	17	"nomenclatureActivitePrincipaleEtablissement": "NAFRev2",
GET GET Bestt (avec siret)	18	"caractereEmployeurEtablissement": "N",
GET GET uniteLegales (par siret)	19	"uniteLegale": {
GET APE	20	"siren": "000325175",
GET APE Copy	21	"dateCreationUniteLegale": "2000-09-26",
GET etatAdministratifEtablissem...	22	"sexeUniteLegale": "M",
	23	"prenom1UniteLegale": "THIERRY",
	24	"prenom2UniteLegale": "",
	25	"prenomUsuelUniteLegale": "THIERRY",
	26	"pseudonymeUniteLegale": "",
	--	"dateCreationUniteLegale": "2000-09-26",

Nous ne nous en rendons pas compte sur cette image, mais les informations d'entreprises sont bien mieux fournies sur Vérifie (Logique, Vérifie est conçu pour ça).

Le petit souci, comme pour la candidathèque, c'est que certaines infos sont parfois ambiguës voir manquantes d'une entreprise à une autre.

Ce qui m'a parfois amené à demander à un collègue d'en modifier ou ajouter.

Par exemple, pour l'adresse dans la candidathèque, j'ai demandé à séparer le nom de département et son numéro pour me faciliter un peu le travail.

Par le nom :

[https://candidatheque.com/api/offres?exists\[idPe\]=true&departement.nom=Rhône\(69\)](https://candidatheque.com/api/offres?exists[idPe]=true&departement.nom=Rhône(69))

Par le code :

[https://candidatheque.com/api/offres?exists\[idPe\]=true&departement.code=69](https://candidatheque.com/api/offres?exists[idPe]=true&departement.code=69)

Exemple de requête sur Vérifie :

Exclure les entreprises de travail temporaire de la liste :

Par le code APE des ETT = 7820Z
activitePrincipaleEtablissement=78.20Z

L'exclure : !=78.20Z

Résultat :

<https://verifyie.fr/api/v1/etablisements?activitePrincipaleEtablissement!=78.20Z>

... Et rechercher par département, par exemple, la somme :

<https://verifyie.fr/api/v1/etablisements?activitePrincipaleEtablissement!=78.20Z&codePostalEtablissement=80>

Ajout de filtres :

Entreprises encore actives (A sinon F): etatAdministratifEtablissement=A

... Et qui emploient (O sinon N): caractereEmployeurEtablissement=O

Cela donne :

<https://verifyie.fr/api/v1/etablisements?activitePrincipaleEtablissement!=78.20Z&codePostalEtablissement=80&etatAdministratifEtablissement=A&caractereEmployeurEtablissement=O>

Pour résumer, J'ai extrait les offres P.E (France-travail) de la candidathèque et relevé les infos utiles pour une recherche plus approfondie dans Verifyie.

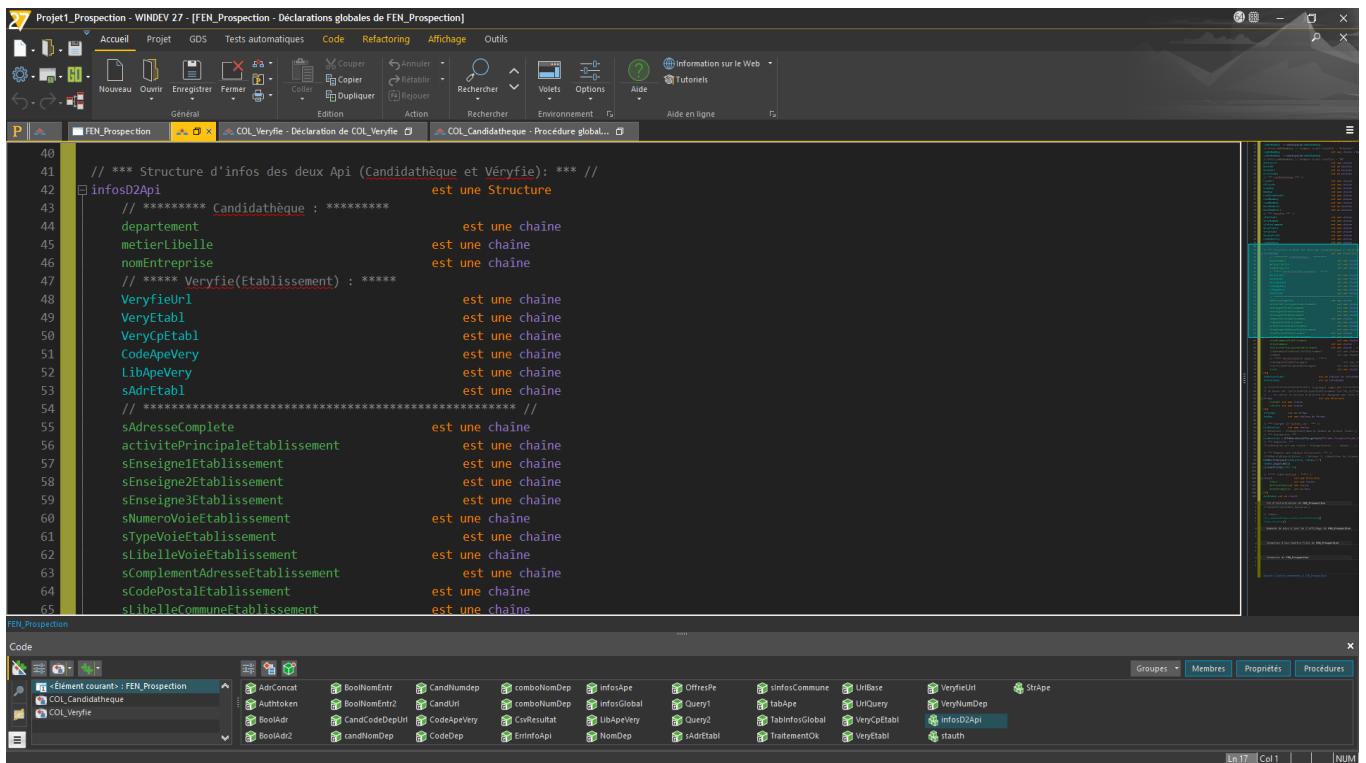
Ensuite, j'ai exclu les offres venant d'agences d'intérim par le biais de leur code APE qui est 7820Z (= 78.20Z pour la clé "activitePrincipaleEtablissement").

Passons maintenant à Windev ! :-)

Qu'est-ce qu'une Structure, et comment s'en sert-on ?

Une structure est un type de données personnalisé qui regroupe des variables qui peuvent être de types différents. C'est en quelque sorte un squelette type d'ensemble de variables (que l'on appelle des membres).

Une fois définie, pour se servir d'une structure, il faut déclarer une variable du type de la structure :



Ici, je déclare une structure qui s'appelle « infosD2Api ». (Comme ça c'est parlant)

Elle contient des variables de types chaîne de caractère « Ainsi » (par défaut) car toutes les langues et données dans le projet utilisent l'alphabet latin. Dans le cas contraire, il faudrait préciser « Unicode » après « ... est une chaîne ».

(On observe un préfixe 'S' devant son nom qui se met automatiquement à la déclaration, afin de se souvenir du type de déclaration).

Déclarons maintenant une variable de ce type afin de pouvoir utiliser la structure :

```
FIN
gstTabInfosGlobal      est un tableau de infosD2Api
gstInfosGlobal         est un infosD2Api
```

On remarque une nouvelle fois un préfixe devant le nom de la variable ainsi qu'une couleur différente... On retrouve le 'ST' qui signifie que cette variable est une structure et un 'g'. Il signifie tout simplement que cette variable est de type global.

Nous reviendrons sur ce type de variable peu après.

Structure (suite)

Un exemple rapide sur l'utilisation d'une structure mais ici ne contenant que deux attributs (membres) du secteur d'activité.

Pour avoir accès à une des variables de la structure, il suffira d'écrire :

« Nom de la variable de structure » . « Nom de la variable à manipuler »

Comme dans cet exemple ici à gauche :

```
infosGlobal.metierLibelle      = infos.metier
infosGlobal.nomEntreprise      = infos.nomEntreprise
```

Une structure doit être créée pour ensuite créer un tableau et une variable de type tableau pour pouvoir le parcourir comme le montre le script suivant :

```
// ***** Traitement combo APE ***** //
// Se baser sur "activitePrincipaleEtablissement"(ex:"81.21Z")de Véryfie
// ... et sortir le secteur d'activité en chargeant une liste d'APE au format CSV
StrApe                                est une Structure
├── CodeAPE est une chaîne
├── Libelle est une chaîne
└── FIN
infosApe                             est un StrApe
tabApe                               est une tableau de StrApe
```

J'ai pris la liberté d'enlever les préfixes « ST... » car je ne suis pas obligé de les laisser bien que ce soit un repère utile selon l'utilisation. (Je préfère ici me contenter que de noms de variables bien parlants et je vais peu les manipuler)

J'ai ensuite créé une variable définie comme tableau de cette structure : « tabApe »

Charger un fichier Csv et remplir un combo :

Enchaînons alors sur l'utilisation du petit tableau de structure vu précédemment pour remplir une liste déroulante (Un champ combo dans le jargon Windev) à partir un fichier Csv :

```
// *** Charger le fichier.csv : *** //
CsvResultat      est une chaîne
//<Résultat> = fChargeTexte(<Nom et chemin du fichier Texte> [, <Mode de chargement>])
// *** Entreprise: ***
CsvResultat = UTF8VersAnsi(fChargeTexte("D:\Mes Projets\Projet_Prospection\CodeAPELibelle.csv"))
// *** Domicile: ***
//CsvResultat est une chaîne = fChargeTexte("C:\Users\Chris\Documents\GitHub\Projet_stage_CDA\Sauv

// *** Remplir mon tableau (structure): *** //
//CSVVersTableau(<Chaîne> , <Tableau> [, <Séparateur de colonne>])
CSVVersTableau(CsvResultat, tabApe,";")
tabApe.Supprime(1)
ListeAffiche[COMBO_SA]
```

```
// ***** Token Refresh : ***** //
stauth      est une Structure
|   Token      est une chaîne
|   RefreshToken est une chaîne
|   DatePeremption est un Date
| FIN
| Authtoken est un stauth
```

Je déclare une variable « CsvResultat » dans laquelle un fichier.csv est chargé et converti en chaîne Ainsi.

J'utilise ensuite la fonction « CSVVersTableau » pour le convertir en tableau en utilisant « ; » comme séparateur de colonnes.

Je supprime la ligne inutile et affiche le contenu de ce tableau (qui est la liste APE) avec la fonction « ListeAffiche » dans le Combo (la liste déroulante) nommé « COMBO_SA ».

(Vous pouvez remarquer la structure en dessous contenant trois membres dont le Token d'authentification, une autre pour le rafraichir, et une dernière pour réinitialiser sa durée de vie à chaque rafraichissement.)

Cela donne :

The screenshot shows the 'Prospection' web application interface. At the top, the title 'Prospection' is displayed in blue. Below it, there is a checkbox labeled 'Exclure les offres des ETT' which is checked. The 'Département' field is set to '80 - Somme'. A note below it says '* Sélectionnez dans la liste ou saisissez le N°'. The 'Secteur d'activité' field is open, showing a list of activities. The selected option is 'Réparation de produits électroniques grand public'. The list includes various categories such as 'Activités des organisations professionnelles', 'Activités des syndicats de salariés', 'Activités des organisations religieuses', 'Activités des organisations politiques', 'Autres organisations fonctionnant par adhésion', 'Réparation d'ordinateurs et d'équipements', 'Réparation d'équipements de communication', 'Réparation de produits électroniques grand public', 'Réparation d'appareils électroménagers et de biens personnels', 'Réparation de chaussures et d'articles en cuir', 'Réparation de meubles et d'équipements du bâtiment', 'Réparation d'articles d'horlogerie et de bijouterie', 'Réparation d'autres biens personnels et domestiques', 'Blanchisserie-teinturerie de gros', 'Blanchisserie-teinturerie de détail', 'Coiffure', 'Soins de beauté', 'Services funéraires', 'Entretien corporel', 'Autres services personnels n.c.a.', 'Activités des ménages en tant qu'employeur', 'Activités indifférenciées des ménages en tant qu'employeur', and 'Activités indifférenciées des ménages en tant que chefs de famille'. On the left side, there are links for 'Objets 3D', 'élécharge', and 'idéos'.

Même procédé pour la liste (combo) des départements :

This screenshot shows the same 'Prospection' web application interface, but with the 'Département' dropdown menu open. The 'Secteur d'activité' field is now set to 'Réparation de produits électroniques grand public'. The 'Code APE' field is set to '9521Z'. A blue 'Rechercher' button is visible at the bottom of the form.

Les variables globales

Une variable globale, contrairement à une variable locale, peut être utilisée à n'importe quel endroit du traitement : Par exemple une variable que l'on déclare dans un bouton ne pourra être utilisée qu'à l'intérieur de ce même bouton. Ici, les variables déclarées étant globales, cela permettra de les utiliser à n'importe quel endroit du traitement.

Une variable globale est en quelque sorte une variable « passerelle » contenant des données pouvant être écrasées ou réinitialisées d'un traitement à un autre.

```
1  Déclarations globales de FEN_Prosection  Si Erreur : par programme Quand Exception : par programme
2  PROCÉDURE MaFenêtre()
3
4  UrlBase          est une chaîne
5  UrlQuery         est une chaîne
6  Query1          est une chaîne
7  Query2          est une chaîne
8  TraitementOk     est un booléen
9
10 // *** Chaîne liste combo département: *** //
11 // "Ardennes - 08"
12 comboNomDep      est une chaîne = ExtraitChaine(COMBO_Département..Valeur, rangDernier, " - ")
13 //infos(comboNomDep) // Exemple visuel résultat = "Ardennes"
14 comboNumDep      est une chaîne = ExtraitChaine(COMBO_Département..Valeur, rangPremier, " - ")
15 //infos(comboNumDep) // Exemple visuel résultat = "08"
16 //infos(comboNumDep) // Exemple visuel résultat = "08"
17 AdrConcat        est une chaîne = comboNomDep + " " + comboNumDep
18 BoolAdr          est un booléen
19 BoolAdr2         est un booléen
20 ErrInfoApi       est un booléen
21 // *** Candidature *** //
22 CandUrl          est une chaîne = "https://candidature.com/api/"
23 OffresPe         est une chaîne = "offres?exists[idPe]=true"
24 CodeDep          est une chaîne = "&departement.code="
25 NomDep           est une chaîne = "&departement.nom="
26 CandCodeDepUrl   est une chaîne = CandUrl + OffresPe + CodeDep + comboNumDep
27 CandNumDep       est une chaîne = CandUrl + OffresPe + CodeDep
28 candNomDep       est une chaîne = CandUrl + OffresPe + NomDep
29 BoolNomEntr      est un booléen
30 BoolNomEntr2     est un booléen
31 // *** Verify *** //
32 sAdrEtabl        est une chaîne
33 VeryNumDep       est une chaîne
34 sInfosCommune    est une chaîne
35 VerifyUrl        est une chaîne = "https://verify.fr/api/v1/"
36 VeryEtabl        est une chaîne = VerifyUrl + "etablisements"
37 VeryCpEtabl      est une chaîne = VerifyEtabl + "?codePostalEtablissement=" + comboNumDep
38 CodeApeVerify    est une chaîne
39 LibApeVerify     est une chaîne
```

Exemples de variables globales (prennent une couleur bleue).

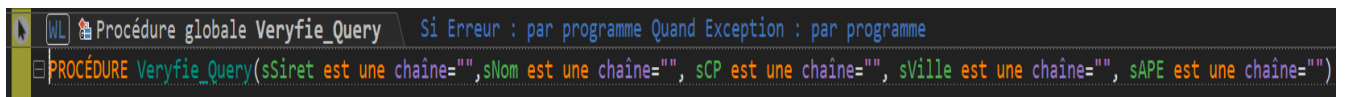
Comme on peut le voir ici, il est possible d'initialiser des valeurs et des fonctions à ces variables dès leur déclaration.

Qu'est-ce qu'une procédure globale et à quoi sert-elle ?

Une procédure globale est en fait une fonction (qui peut également être locale), qui nous servira à stocker du code que l'on pourra appeler à tout moment. Cela permet principalement d'éviter les redondances de codes.

Contrairement à une procédure locale, la procédure globale pourra être appelée de n'importe quelle fenêtre du logiciel.

Afin que cette procédure puisse être utilisée avec des paramètres simples et généraux, je l'ai décrite comme suit :

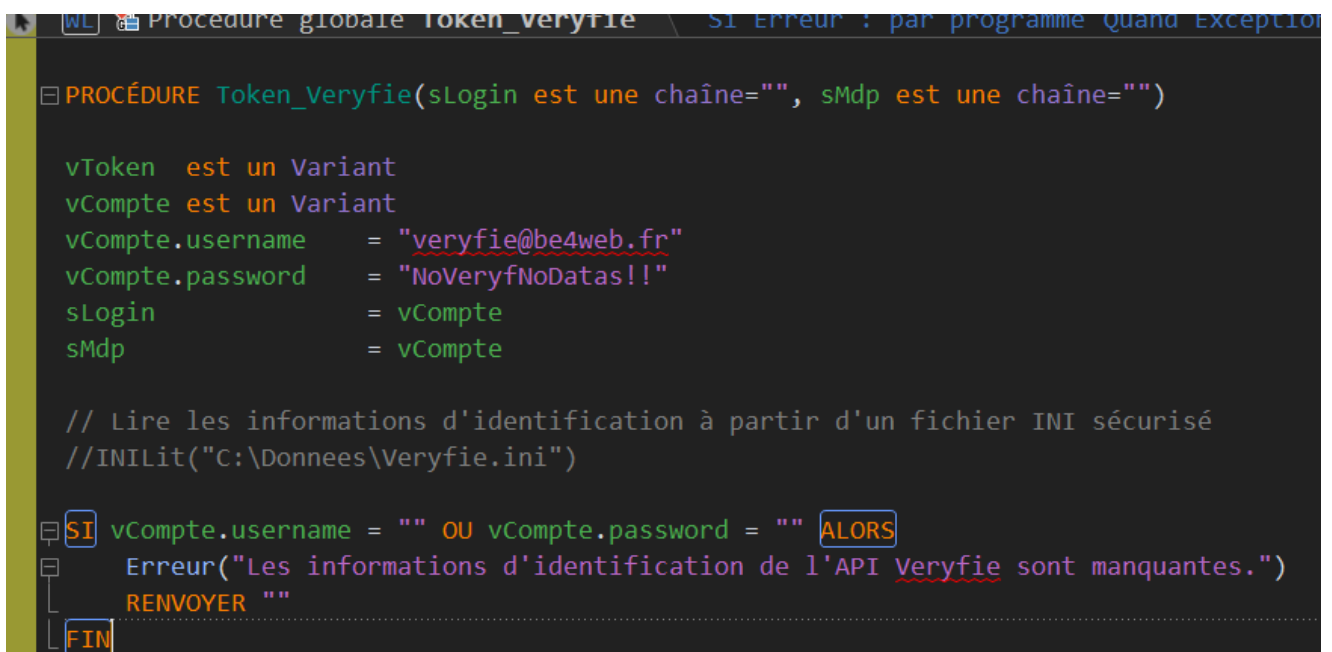


On peut donc voir que je n'appellerai que 5 paramètres (pour l'Api Verify qui compléteront les infos recueillis de l'Api Candidathèque) :

Le siret, le nom d'établissement, le code postal, la ville, le code Ape

(Par souci de performances, il est préférable de limiter le nombre de paramètres dans une procédure.)

Exemple de procédure globale avec l'authentification avec ici, les données de connexion :



Vous remarquerez que l'on a à faire à un nouveau type de variable : le variant

Le type variant :

Le type variant permet de stocker n'importe quelle valeur : booléen, numérique, chaînes de caractères, date, des sous-éléments (nommés ou indicés), structures, classes, tableaux, listes etc...

procédure globale et méthodes de connexion

Poursuivons avec les méthodes de connexion dans une procédure :

```
SI HTTPRequête("https://verify.fr/api/login check", "", "X-HTTP-Method-Override: POST", VariantVersJSON(vCompte), "application/json") ALORS
  sReponse est une chaîne=HTTPDonneRésultat(httpRésultat)
  vToken = JSONVersVariant(sReponse)

  SI vToken.token..Existe _ET_ vToken.token > "" ALORS
    gsTokenVerify = vToken.token
    Trace("Token Verify : " + RC + vToken.token)
  SINON
    gsTokenVerify = ""
  FIN
FIN
//Trace(gsTokenVerify.valeur)
RENOYER gsTokenVerify
```

Cette portion de code effectue une requête HTTP POST à l'API de Verify pour obtenir un jeton d'authentification. Si la réponse HTTP contient un jeton d'authentification valide, la variable globale gsTokenVerify est définie sur la valeur du jeton. Sinon, la variable globale gsTokenVerify est définie sur une chaîne vide.

La dernière ligne de code « RENOYER » renvoie la valeur de la variable globale gsTokenVerify à la fonction ou la procédure qui a appelé ce code.

En renvoyant cette valeur, ce code permet à la fonction ou la procédure appelante d'utiliser le jeton d'authentification pour effectuer d'autres requêtes http authentifiées auprès de l'Api.

```

[WL] Procédure globale Verifyie_Query / Si Erreur : par programme Quand Except
PROCÉDURE Verifyie_Query(ssiret est une chaîne="",sNom est une chaîne="", sCP est une chaîne="")
    sEnteteSup      est une chaîne
    sMethodeHTTP    est une chaîne
    //Définir le temps de réponse (délai d'expiration en ms):
    nTimeout        est un entier    = 10000
    vReponse        est un Variant
    sMime            est une chaîne   = "application/json"
    nPage            est un entier    = 30

    sMethodeHTTP     = "POST"
    sEnteteSup       = ""
    bRep est un booléen

    SI gsTokenVerifye>" " ALORS
        //Si la variable globale gsTokenVerifye contient une valeur.
        // Si c'est le cas, l'en-tête d'autorisation est ajouté à sEnteteSup.
        sEnteteSup += "Authorization: Bearer " + gsTokenVerifye + RC
    SINON
        vReponse.Erreur      = Vrai
        vReponse.ErreurMessage = "Pas de Token"
        RENVOYER vReponse
    FIN
    vReponse.Erreur      = Faux

    SI sMethodeHTTP>" " ALORS
        sEnteteSup += "X-HTTP-Method-Override: " + sMethodeHTTP
    FIN
    //httpParamètreMode Permet de paramétrer le mode de gestion des requêtes
    //HTTPParamètre(httpParamètreMode,1)
    //httpParamètreDésactiveCache En mode sécurisé (HTTPS), permet de gérer le cache
    //0 (Faux par défaut) : Permet de réactiver le cache de Internet Explorer.
    //1 (Vrai) : Permet de désactiver le cache de Internet Explorer.
    HTTPParamètre(httpParamètreDésactiveCache,1)
    HTTPTimeOut(nTimeout)

```

Cette procédure effectue une requête HTTP à l'API de Verifyie pour récupérer des données sur les établissements en fonction des paramètres d'entrée, et renvoie une réponse contenant les résultats de la recherche.

Explications :

La requête de recherche est construite en fonction des paramètres d'entrée. Si le paramètre sSiret est présent, la recherche est effectuée en fonction de celui-ci. Sinon, la recherche est effectuée en fonction des autres paramètres (sAPE, sNom, sCP, sVille).

```

sResultat      est une chaîne
sUrlBase       est une chaîne
sUrlResultat   est une chaîne
sUrlBase="https://verifyie.fr/api/v1/"
sQuery         est une chaîne

sQueryBase est une chaîne = "etablissements"

SI sSiret>" " ALORS
    sQuery+="/" + sSiret
SINON
    SI sAPE>" " ALORS
        sUnite est une chaîne = "activitePrincipaleEtablissement"
        // sQuery+=["&"]+sUnite+"="+sAPE
        sQuery+=["&"]+sUnite+"="+sAPE[[1 À 2]]+"."+sAPE[[3 À 5]]
    FIN
    SI sNom>" " ALORS
        sQuery+=["&"]+"denominationUniteLegale="+sNom
    FIN
    SI sCP>" " ALORS
        sQuery+=["&"]+"codePostalEtablissement="+sCP
    FIN
    SI sVille>" " ALORS
        sQuery+=["&"]+"libelleCommuneEtablissement="+sVille
    FIN
FIN

sUrlBase = sUrlBase + sQueryBase + "?" + sQuery
vresultat est un JSON
tabPerPage est un tableau de JSON
//bufResultat est un buffer
nMaxPerPage est un entier = 1000

```

Requêtes et pagination

```
BOUCLE
    sUrlResultat = sUrlBase + ["&"] + "etatAdministratifEtablissement=A" + ["&"] + ...
    "caractereEmployeurEtablissement=0" + ["&"] + "per_page=" + nMaxPerPage + ["&"] + "page=" + nPage
    bRep = HTTPRequete(sUrlResultat,"",sEnteteSup, "",sMime)
    SI PAS bRep ALORS
        vReponse.Erreur = Vrai
        vReponse.ErreurMessage = ErreurInfo(errMessage)
        RENVOYER vReponse
    FIN
    sResultat = HTTPDonneResultat(httpEntete)
    SI Contient(sResultat, " 200") ALORS
        sResultat = HTTPDonneResultat(httpResultat)
        vresultat = ChaîneVersJSON(sResultat)

        // Boucle pour ajouter chaque élément au tableau global
        POUR TOUT vElem DE vresultat
            TableauAjoute(tabPerPage, vElem)
        FIN
        SI vresultat..Occurrence < nMaxPerPage ALORS
            SORTIR
        SINON
            nPage = nPage + 1
        FIN
    SINON SI Contient(sResultat, " 500") _ET_ tabPerPage..Occurrence > 0 ALORS
        // fin de boucle pagination
        SORTIR
    SINON
        vReponse.Erreur = Vrai
        vReponse.ErreurMessage = HTTPDonneResultat(httpEntete)
        RENVOYER vReponse
    FIN
FIN
vReponse.Erreur = Faux
vReponse.Resultats = tabPerPage
//Sérialise(tabPerPage,bufResultat,psdJSON)
RENOYER vReponse
```

Une boucle est utilisée pour récupérer tous les résultats de la recherche en fonction de la pagination. La requête HTTP est effectuée à l'aide de la fonction HTTPRequete, et le résultat est stocké dans la variable sResultat.

Explications :

Si la requête HTTP échoue, une réponse d'erreur est renvoyée. Sinon, le résultat est converti en JSON à l'aide de la fonction ChaîneVersJSON, et chaque élément est ajouté à un tableau global.

Si le nombre d'éléments dans la réponse est inférieur à la limite maximale par page, la boucle est interrompue. Sinon, le numéro de page est incrémenté et la boucle se poursuit.

Si la réponse contient une erreur 500 et que le tableau global contient des éléments, la boucle est interrompue. Sinon, une réponse d'erreur est renvoyée.

La procédure renvoie la réponse de la requête, y compris les résultats de la recherche dans le tableau global.

Requêtes et récupération de données

```
PROCEDURE Boucle_Verifie()

// ***** Boucle_Verifie : ***** //
vReponseVerifie est un Variant = Verifie_Query()
BoolCp2          est un booléen

SI vReponseVerifie.Erreur.Existe _ET_ PAS vReponseVerifie.Erreur ALORS
POUR TOUT infos DE vReponseVerifie.Resultats
//SI infos <> "" ET infos <> Null ALORS

Cp      est une chaîne = infos.codePostalEtablissement
// S'il ne trouve pas le code postal établissement, alors il prend le Cp commune
SI infos.codePostalEtablissement = "" ALORS
Cp = infos.codeCommuneEtablissement
FIN
VerNumDep est une chaîne = FEN_Prospedition.VerNumDep
VerNumDep      = ExtraitChaîne(Cp,1,"/",DepuisFin)
VerNumDep      = SansEspace(VerNumDep)
//VerifieNomDep
comboNumDep est une chaîne = FEN_Prospedition.comboNumDep
// Si ma variable département(Ver) contient le numéro département sélectionné dans mon combo alors ...
SI VerNumDep = Contient(Cp, comboNumDep, SansCasse) ALORS
Query2 est une chaîne = FEN_Prospedition.VerCpEtabl
BoolCp2 = Vrai
FIN

sAdrEtabl      est une chaîne
sInfosCommune  est une chaîne
sCodePostalEtablissement est une chaîne
sActivitePrincipaleEtablissement est une chaîne // = FEN_Prospedition.infosGlobal.sActivitePrincipaleEtablissement
siren          est une chaîne // = FEN_Prospedition.infosGlobal.siren
// *** Etablissement : *** //
sInfosCommune = ...
infos.codeCommuneEtablissement + " " + infos.libelleCommuneEtablissement
sCodePostalEtablissement = infos.codePostalEtablissement
sAdrEtabl = ...
infos.numeroVoieEtablissement + " " + infos.typeVoieEtablissement + " " + infos.libelleVoieEtablissement + ", " + ...
infos.codeCommuneEtablissement + " " + infos.libelleCommuneEtablissement
sActivitePrincipaleEtablissement = infos.activitePrincipaleEtablissement //code APE //Trace ok
// *** Unite légale : *** //
siren = infos.uniteLegale.siren
//sMail = infos.uniteLegale.mail
//FIN
FIN
Trace("Infos_Verifie : " + RC + ...
"APE : " + FEN_Prospedition.infosGlobal.sActivitePrincipaleEtablissement + RC + ...
"ADR : " + FEN_Prospedition.sAdrEtabl + RC + ...
"SIREN : " + FEN_Prospedition.infosGlobal.siren)
// "MAIL : " + infosGlobal.sMail)

sAdrEtabl      est une chaîne = FEN_Prospedition.sAdrEtabl
sInfosCommune  est une chaîne = FEN_Prospedition.sInfosCommune
SI sAdrEtabl = "" _OU_ sAdrEtabl = Null ALORS
FEN_Prospedition.COL_adr = FEN_Prospedition.sInfosCommune
Trace("Adresse complète manquante. Adresse commune = " + sInfosCommune)
FIN
FIN
Trace("BoolCp2(Verifie) = " + BoolCp2)
RENVOYER BoolCp2
```

cette procédure est utilisée pour récupérer des données sur les établissements à partir de l'API Verifie, et pour filtrer les résultats en fonction du numéro de département sélectionné dans le combo (liste déroulante) de la fenêtre "FEN_Prospedition" (l'interface).

Requêtes et récupération de données (suite)

Explications du code précédant :

La procédure commence par la déclaration de deux variables : "vReponseVerify" est un variant qui contiendra la réponse de l'API Verify, et "BoolCp2" est un booléen qui sera utilisé plus tard dans le code.

La fonction "Verify_Query" est appelée pour récupérer les données sur les établissements à partir de l'API Verify. Les résultats sont stockés dans la variable "vReponseVerify".

Une boucle "POUR TOUT" est utilisée pour parcourir les résultats de la requête Verify. Chaque itération de la boucle traite un seul établissement.

Dans la boucle, la variable "Cp" est définie comme étant le code postal de l'établissement. Si le code postal de l'établissement n'est pas disponible, le code postal de la commune est utilisé à la place.

La variable "VeryNumDep" est définie comme étant le numéro de département du code postal de l'établissement et "comboNumDep" étant le numéro de département sélectionné dans le combo départements de la fenêtre principale "FEN_Prospection".

Si le numéro de département de l'établissement correspond au numéro de département sélectionné dans le composant "comboNumDep", alors la variable "BoolCp2" est définie comme étant vraie et la chaîne de requête "Query2" est définie comme étant la valeur de la chaîne "VeryCpEtabl" de la fenêtre "FEN_Prospection".

Les variables "sAdrEtabl", "sInfosCommune", "sCodePostalEtablissement" et "sActivitePrincipaleEtablissement" sont définies et remplies avec les informations de l'établissement en utilisant les données de la variable "infos".

(La fonction "Trace" est utilisée pour afficher les informations de l'établissement dans une fenêtre de débogage.)

Le tableau associatif

```
// ***** Boucle CandidatHèque : *****
Boucle_CandidatHèque()
// ***** Boucle_Verifie : *****
Boucle_Verifie()
// *****

// Si les codes postaux des deux Api correspondent au combo :
BoolCp1 est un booléen = Boucle_CandidatHèque() = Vrai
BoolCp2 est un booléen = Boucle_Verifie() = Vrai
SI BoolCp1 = Vrai ET BoolCp2 = Vrai ALORS
  // ... ALORS rafraichir le tableau avec ces valeurs
  TableAffiche(TABLE_Résultats, taCourantBandeau)
  //infosGlobal.metierLibelle      = infosGlobal.metierLibelle
  //infosGlobal.metierLibelle
  //infosGlobal.nomEntreprise      = infosGlobal.nomEntreprise
FIN

tabAssoD2api est un tableau associatif (ccSansAccent + ccSansCasse + ccSansEspace + ccSansPonctuationNiEspace + SansDoublon) de chaînes

tabAssoD2api["dep"]      = infosGlobal.departement
tabAssoD2api["metier"]   = infosGlobal.metierLibelle
tabAssoD2api["nomEntr"]  = infosGlobal.nomEntreprise
tabAssoD2api["adr"]      = ADRConcat
tabAssoD2api["ape"]      = infosGlobal.sActivitePrincipaleEtablissement
tabAssoD2api["siren"]    = infosGlobal.siren
//tabAssoD2api["mail"]   = infosGlobal.sMail
```

Ce code définit une procédure pour chacune des deux Api qui sont appelées l'une après l'autre. Ces procédures sont des boucles qui effectuent des requêtes sur les deux API pour récupérer des données.

Après avoir appelé les deux boucles, le code vérifie si les deux ont renvoyé Vrai. Si c'est le cas, cela signifie que les codes postaux des résultats de recherche des deux API correspondent au code postal sélectionné dans le combo « COMBO_Département ».

Le code appelle alors la fonction TableAffiche() pour afficher les résultats dans un tableau nommé « TABLE_Résultats ».

(Les « Tables » chez Windev, sont les tableaux physiques, cela peut porter à confusion.)

Le code définit aussi un tableau associatif nommé tabAssoD2api. Ce tableau est utilisé pour stocker des paires de valeurs, où chaque paire se compose d'une clé et d'une valeur. Les clés sont des chaînes de caractères qui identifient les valeurs stockées dans le tableau.

Ici, les clés sont "dep", "metier", "nomEntr", "adr", "ape" et "siren", et les valeurs correspondantes sont stockées à partir des données récupérées à partir des deux API.

Les conditions Booléennes

```
POUR TOUT infos DE tabAssoD2api
// *** Comparaison adresses : *** //
SI Contient(sAdrEtabl, comboNumDep, ccSansAccent + ccSansCasse + ccSansEspace + ccSansPonctuationNiEspace + ccSansEspaceIntérieur ) ET
Contient(sAdrEtabl, comboNumDep, ccSansAccent + ccSansCasse + ccSansEspace + ccSansPonctuationNiEspace + ccSansEspaceIntérieur ) ALORS
tabAssoD2api.Insère("adr", sAdrEtabl)
BoolAdr = True
SINON
SI sAdrEtabl = "" ET Contient(sInfosCommune, comboNumDep, SansCasse) ALORS
tabAssoD2api.Insère("adr", sInfosCommune)
BoolAdr2 = True
SINON
SI sInfosCommune = "" ALORS
COL_adr = RougeClair = "Donnée manquante de l'Api"
BoolAdr2 = False
ErrInfoApi = True
FIN
FIN
// *** Comparaison nom entreprise : *** //
// Si nomEntreprise existe dans Candidattheque et que la chaîne est contenue dans Enseigne de Vérifie alors ...
nomEntr est une chaîne

SI tabAssoD2api["nomEntr"]..Existe = Vrai ET Contient(infosGlobal.sEnseigne1Etablissement, infosGlobal.nomEntreprise, SansCasse) ALORS
nomEntr = infosGlobal.nomEntreprise
tabAssoD2api.Insère("nomEntr", infosGlobal.nomEntreprise)
BoolNomEntr = True
SINON
SI tabAssoD2api["nomEntr"]..Existe = Faux ALORS
nomEntr = infosGlobal.sEnseigne1Etablissement
BoolNomEntr2 = True
SINON
SI infosGlobal.sEnseigne1Etablissement = "" ALORS
nomEntr = infosGlobal.sEnseigne2Etablissement
BoolNomEntr2 = True
SINON
SI infosGlobal.sEnseigne2Etablissement = "" ALORS
nomEntr = infosGlobal.sEnseigne3Etablissement
BoolNomEntr2 = True
Trace(BoolNomEntr)
SINON
SI nomEntr = "" ALORS
COL_nomEntreprise = RougeClair = "Donnée manquante de l'Api"
BoolNomEntr2 = False
ErrInfoApi = True
FIN
FIN
// Si l'une ou l'autre adresse ET l'un ou l'autre nom d'entreprise à été trouvé ALORS ...
SI (BoolAdr = True OU BoolAdr2 = True) ET (BoolNomEntr = True OU BoolNomEntr2 = True) ALORS
TraitementOk = True
TableauAjoute(TabInfosGlobal, infosGlobal)
FIN
```

Ce code parcourt chaque élément "infos" dans le tableau associatif "tabAssoD2api". Pour chaque élément, il effectue deux comparaisons :

Comparaison des adresses et des noms d'entreprises.

Si l'une des variables BoolAdr ou BoolAdr2 est True et que l'une des variables BoolNomEntr ou BoolNomEntr2 est True, alors la variable TraitementOk est définie sur True.

Le contenu de la variable infosGlobal est alors ajouté au tableau TabInfosGlobal à l'aide de la fonction TableauAjoute().

La boucle POUR TOUT parcourt tous les éléments du tableau associatif tabAssoD2api, et pour chaque élément, la variable infos prend la valeur de cet élément.

Cette page de code se termine par ceci :

```
TableAffiche(TABLE_Résultats)
TableDésactiveFiltre(COL_departement)
TableActiveFiltre(COL_departement, filtreEgal, comboNumDep)
```

TableAffiche est une fonction qui affiche le contenu de la table TABLE_Résultats dans l'interface utilisateur.

TableDésactiveFiltre(COL_departement) est une fonction qui désactive le filtre sur la colonne COL_departement de la table TABLE_Résultats.

TableActiveFiltre est une fonction qui active le filtre sur la colonne COL_departement de la table TABLE_Résultats (tableau physique) en utilisant le critère de filtre filtreEgal (qui signifie que la colonne doit être égale à une valeur donnée) et en utilisant la valeur de comboNumDep (qui est une variable contenant le numéro de département sélectionné par l'utilisateur dans l'interface utilisateur) comme valeur de filtre.

Cela signifie que seules les lignes de la table TABLE_Résultats qui ont la même valeur que comboNumDep dans la colonne COL_departement seront affichées dans l'interface utilisateur.

Les conditions Booléennes (suite)

Code du bouton :

```
// *****  
// Je vais boucler les résultats de chaque Api et les associer dans le tableau "tabAssoD2api",  
// ... pour ensuite injecter mes données dans mon tableau physique :  
// *****  
  
// *** Traitement Combo département: (connexion candidathèque) *** //  
  
// Si lors du clic bouton, une ligne combo n'a été sélectionnée ...  
SI MoiMême = True ET (COMBO_Département..Valeur = COMBO_Département..ValeurInitiale)...  
    OU (COMBO_SA..Valeur = COMBO_SA..ValeurInitiale) ALORS  
        COMBO_Département = RougeClair = "Sélection combo manquante"  
        Trace("Sélection combo manquante")  
        RETOUR  
    FIN  
    Trace("ComboVal = " + COMBO_Département..Valeur + RC + "ComboValInit = " + COMBO_Département..ValeurInitiale)  
// Si clic sur bouton + Int coché et lignes combo sélectionnées :  
SI MoiMême = True ET INT_ExclureETT = True ...  
    ET (COMBO_Département..Valeur <> COMBO_Département..ValeurInitiale)...  
    ET (COMBO_SA..Valeur <> COMBO_SA..ValeurInitiale) ALORS  
        UrlBase = CandUrl + OffresPe + CodeDep + comboNumDep  
    SINON  
        // Si clic sur bouton (+ Int non-coché) et lignes combo sélectionnées :  
        SI MoiMême = True ET INT_ExclureETT = False ...  
            ET (COMBO_Département..Valeur <> COMBO_Département..ValeurInitiale)...  
            ET (COMBO_SA..Valeur <> COMBO_SA..ValeurInitiale) ALORS  
                UrlBase = CandUrl + "departement.code=" + comboNumDep  
            FIN  
        Trace("UrlCand = " + UrlBase)  
    FIN
```

Il s'agit d'un traitement qui se déclenche lorsque l'utilisateur clique sur un bouton. Le but du traitement est de récupérer des données depuis une API en fonction des valeurs sélectionnées dans deux combo : "COMBO_Département" et "COMBO_SA" (Secteur d'activité).

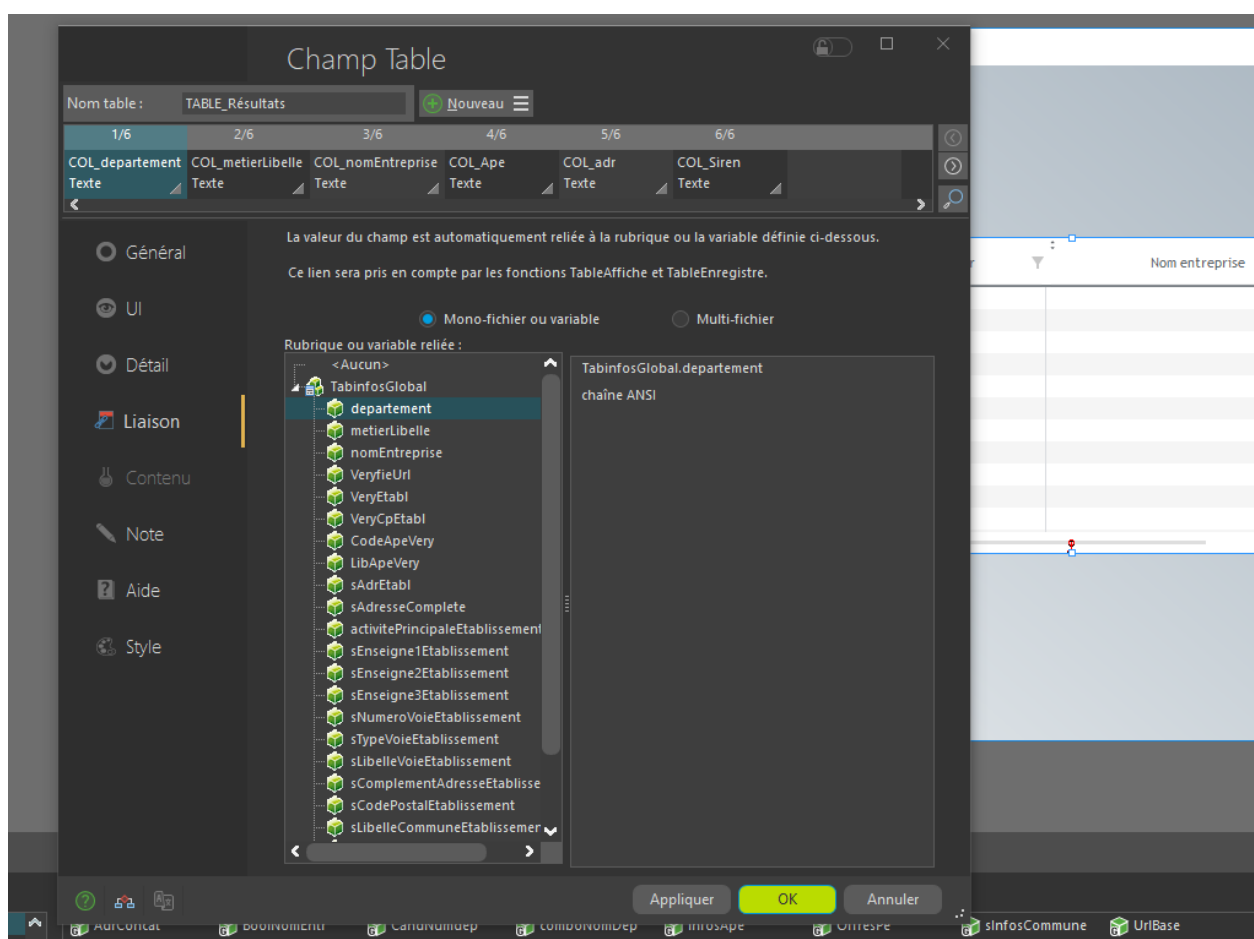
Le code commence par vérifier si aucune valeur n'a été sélectionnée dans les deux combos. Si c'est le cas, un message d'erreur est affiché et le traitement est interrompu à l'aide de la commande "RETOUR".

Si au moins une valeur a été sélectionnée, le code vérifie si l'option "INT_ExclureETT" est cochée. Cette option permet d'exclure les entreprises de travail temporaire (ETT) des résultats de la recherche.

Si l'option est cochée et que des valeurs ont été sélectionnées dans les deux combos, le code construit l'URL de la requête API en concaténant des chaînes de caractères et les valeurs sélectionnées dans ces combos.

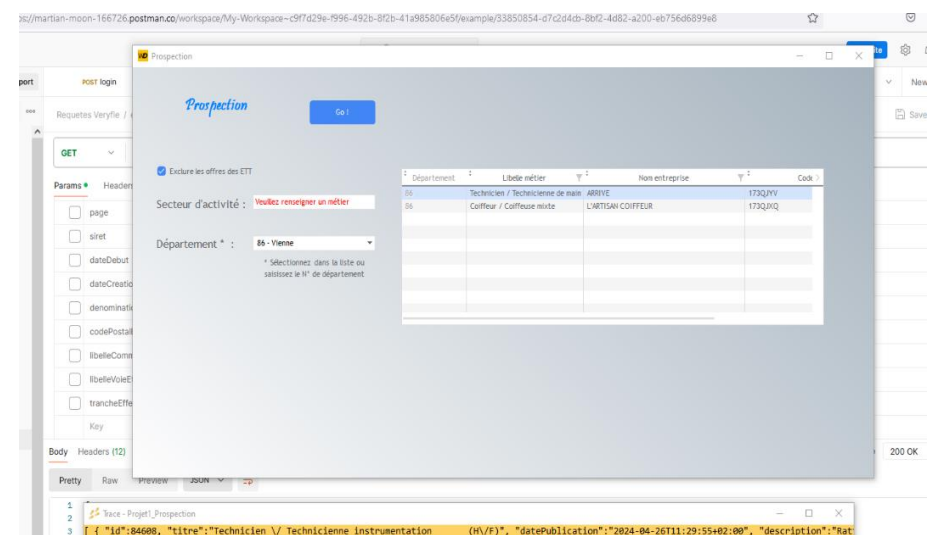
Si l'option n'est pas cochée mais que des valeurs ont été sélectionnées dans les deux combos boxes, le code construit une URL de requête API différente, qui n'exclut pas les ETT des résultats de la recherche.

Table et liaison



Après un clic droit sur la table, dans « description » / « contenu », il est possible de lier celle-ci par variables ou par programmation.

Idem pour chaque colonne de la table dans l'onglet « Liaison » ou l'on peut voir ici qu'elle sont liés à chaque variable correspondante.



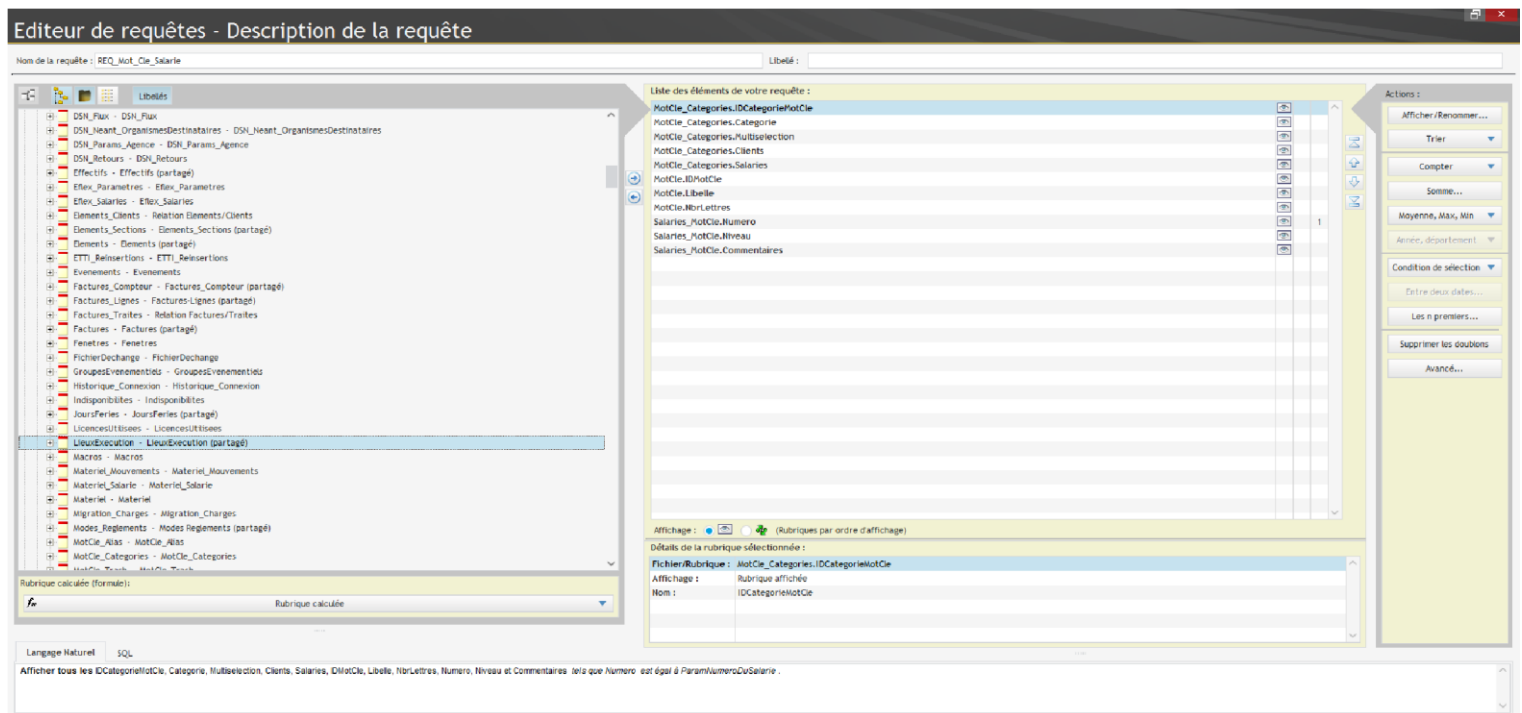
Exemple d'une requête classique avec Windev (hors-projet)

Les deux principales manières d'effectuer des requêtes avec Windev sont :

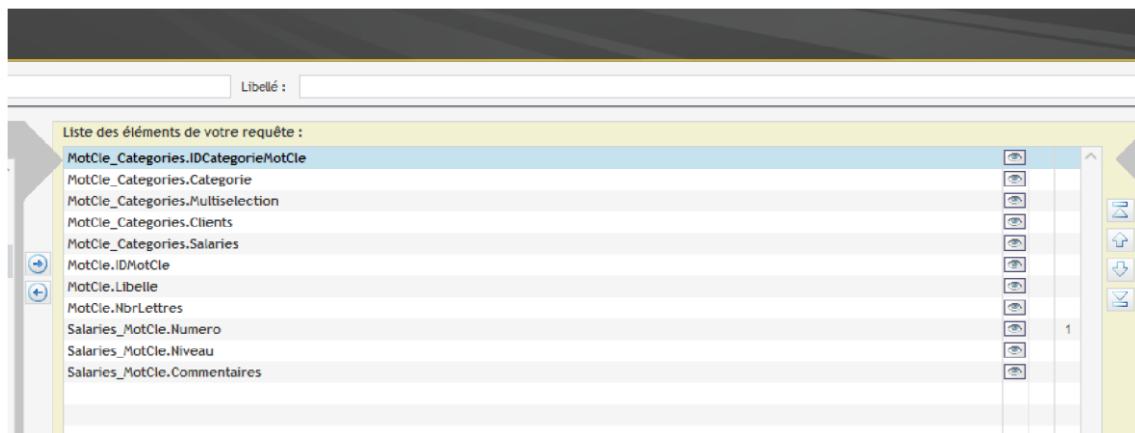
→ Les écrire en dur grâce à la fonction ' HExécuteRequêteSQL '

→ La créer et l'enregistrer avec l'éditeur de requête pour l'exécuter ensuite avec la fonction ' HexecuteRequête ' (c'est ce cas précis que nous allons voir).

En ouvrant l'éditeur de requête, vous pouvez bien évidemment effectuer tout type de requête, mais nous nous occuperons pour l'instant des requêtes de type ' Select '.



Avec l'éditeur de requête, vous sélectionner sur la gauche les tables (ou colonnes de tables qui vous intéressent, puis vous les placez dans la liste située sur la droite.



Sur la droite de la liste vous avez une colonne (où il y a le ' 1 ') qui vous permet de placer des conditions à votre requête.

Description d'une condition

Cette requête sélectionnera les enregistrements pour lesquels :

La rubrique

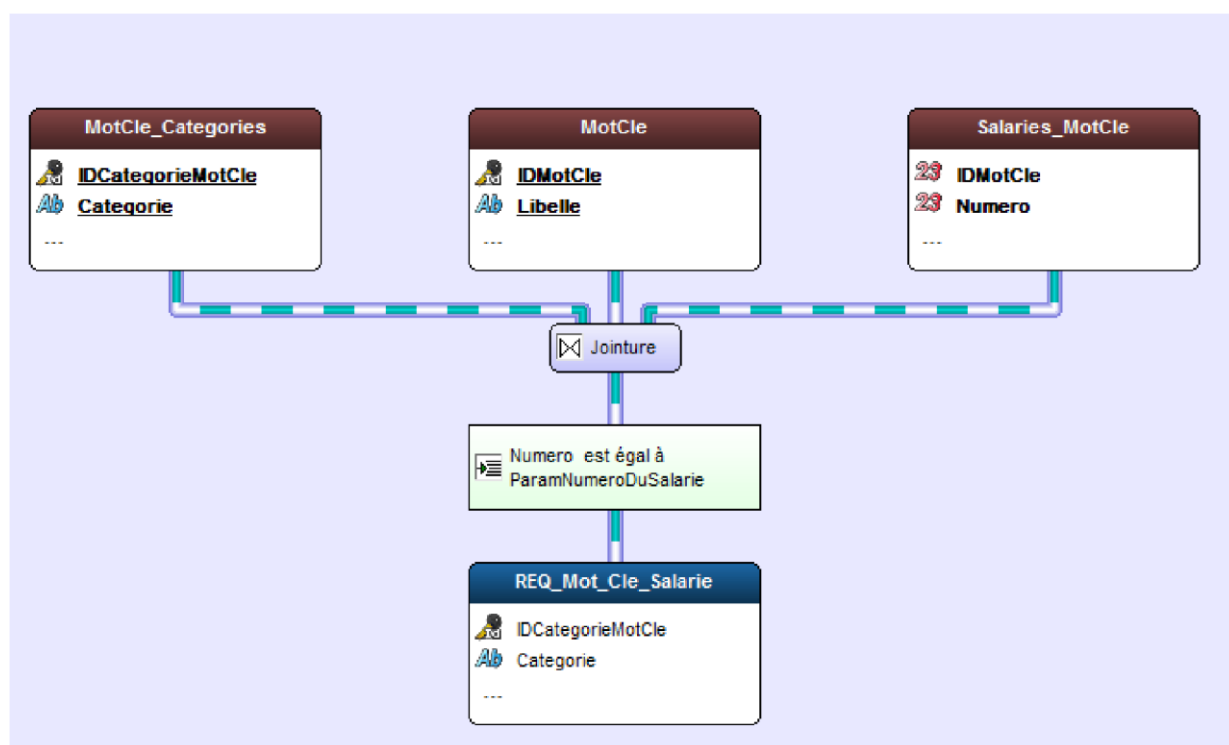
☐ la valeur
☒ au paramètre
☐ toutes les valeurs de (ALL)
☐ n'importe quelle valeur de (ANY)

Nom du paramètre :

Remarque : les guillemets ne sont pas nécessaires pour les chaînes.
Ils sont gérés automatiquement par l'éditeur.

Ici on remarque que la rubrique (qui est en fait la colonne de la table) sera égale à un paramètre dont le nom sera 'ParamNumeroDuSalarie'

Validez ensuite votre requête et obtenez cet écran récapitulatif :



On obtient un écran très clair de la requête :

- Les tables ou colonnes de table que l'on a choisi,
- La jointure (qui s'est faite automatiquement)
- Le paramètre d'exécution de la requête
- Le résultat de la requête.

Revenons à la requête qui nous concerne et observons la fin du code :

```
HExécuteRequête(REQ_AfficheTypeDocuments)

POUR TOUT REQ_AfficheTypeDocuments
    listeClasseur += ["," + REQ_AfficheTypeDocuments.CodeTypeDocument]
FIN

Export_Salarie(sIdDuSalarie, listeClasseur, stParametreOnglets)
```

J'exécute donc la requête 'REQ_AfficheTypeDocuments' sans aucun paramètre, ce qui revient à dire que je souhaite prendre tout le contenu de la table sans exception.

(L'équivalent en langage SQL serait donc 'SELECT * FROM TypeDocuments')

Une fois la requête exécutée, j'effectue une boucle de type 'POUR TOUT' qui me permettra de parcourir chaque ligne du résultat de requête.

Dans cette boucle, je mettrai donc dans ma variable 'listeClasseur', Chacun des 'codeTypeDocument' existants dans la table. Ils seront séparés par un ';' (ce qui me permettra de mieux les récupérer dans une prochaine boucle).

Enfin, j'appelle ma procédure dans laquelle je place ces deux variables et cette structure en paramètres.