



Stage à AOIP

L'Association des ouvriers en instruments de précision

Présentation

« AOIP étudie, fabrique et commercialise depuis plus de 100 ans des instruments de mesure et des systèmes de contrôle moteur. » (source : <https://www.aoip.fr/>)

Dans le secteur où je travaillais, le hardware, on utilisait une interface Excel pour faire des recherches sur leur base de données.

Mon travail dans cette entreprise était de réaliser une mise à jour de cet outil pour leur faciliter la tâche ainsi que de leur permettre de gagner du temps.

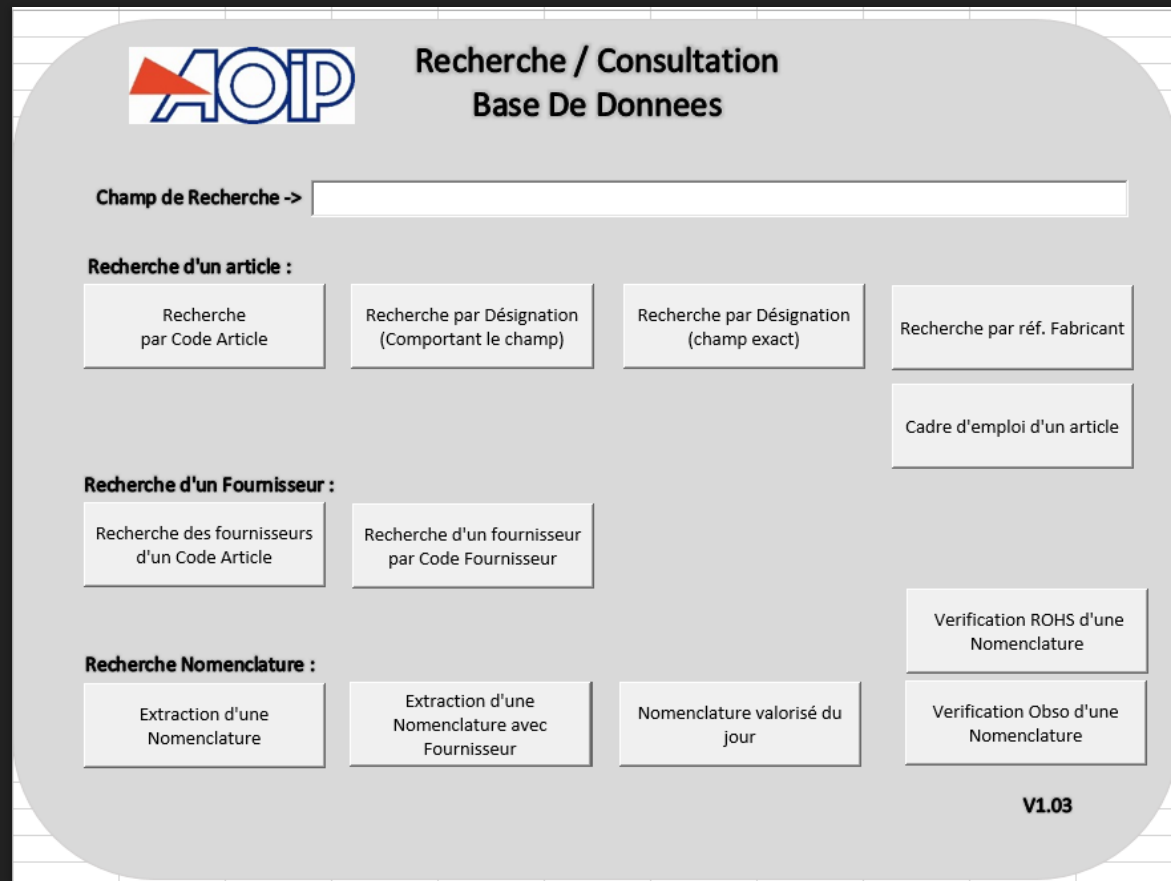
Tout mon codage a été en langage **VBA** et **MySQL**.

Nous pouvons résumer la mise à jour en 4 tâches différentes.

Remarque : Après relecture de ce rapport et avoir acquis de l'expérience, je me rends compte que la nomination des variables que j'ai créées ne correspondent pas à leur fonctionnalité. C'était une grosse erreur de ma part.

L'interface

Ancienne Version



AOiP Recherche / Consultation
Base De Donnees

Champ de Recherche ->

Recherche d'un article :

- Recherche par Code Article
- Recherche par Désignation (Comportant le champ)
- Recherche par Désignation (champ exact)
- Recherche par réf. Fabricant
- Cadre d'emploi d'un article

Recherche d'un Fournisseur :

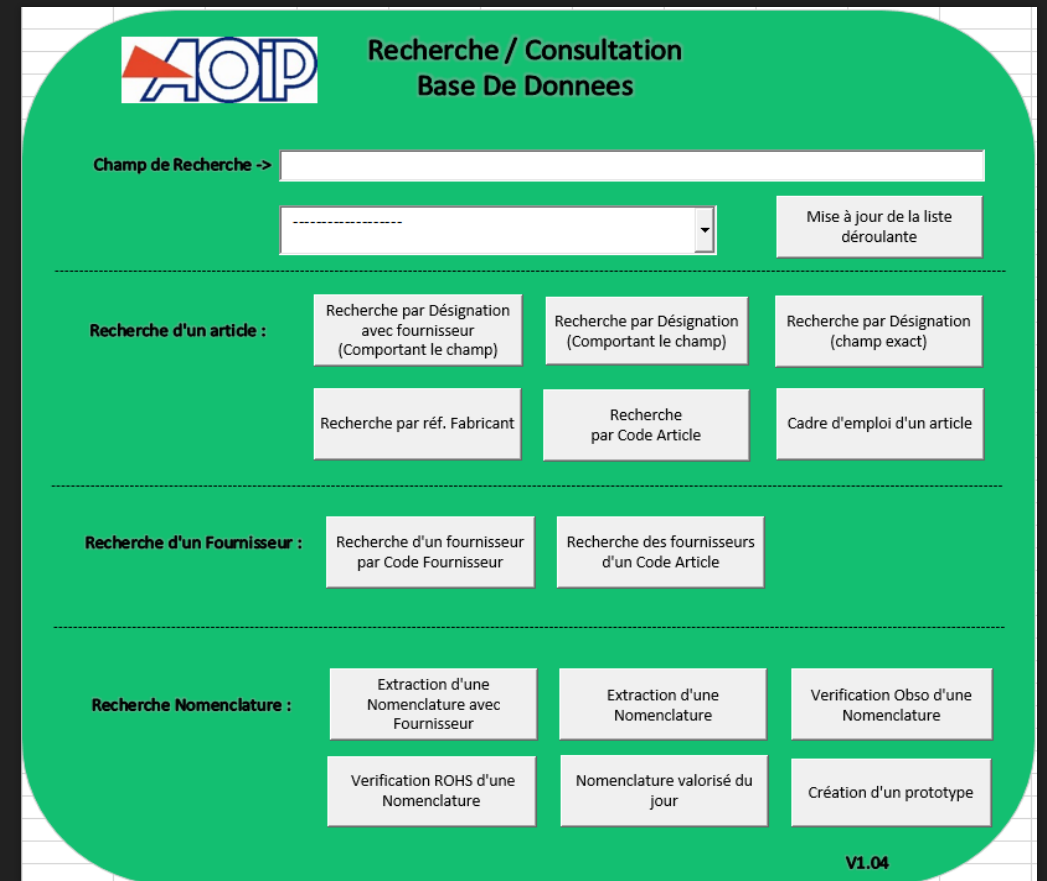
- Recherche des fournisseurs d'un Code Article
- Recherche d'un fournisseur par Code Fournisseur

Recherche Nomenclature :

- Extraction d'une Nomenclature
- Extraction d'une Nomenclature avec Fournisseur
- Nomenclature valorisé du jour
- Verification ROHS d'une Nomenclature
- Verification Obso d'une Nomenclature

V1.03

Nouvelle Version



AOiP Recherche / Consultation
Base De Donnees

Champ de Recherche ->

Mise à jour de la liste déroulante

Recherche d'un article :

- Recherche par Désignation avec fournisseur (Comportant le champ)
- Recherche par Désignation (Comportant le champ)
- Recherche par Désignation (champ exact)
- Recherche par réf. Fabricant
- Recherche par Code Article
- Cadre d'emploi d'un article

Recherche d'un Fournisseur :

- Recherche d'un fournisseur par Code Fournisseur
- Recherche des fournisseurs d'un Code Article

Recherche Nomenclature :

- Extraction d'une Nomenclature avec Fournisseur
- Extraction d'une Nomenclature
- Verification Obso d'une Nomenclature
- Verification ROHS d'une Nomenclature
- Nomenclature valorisé du jour
- Création d'un prototype

V1.04

Tâche numéro 1

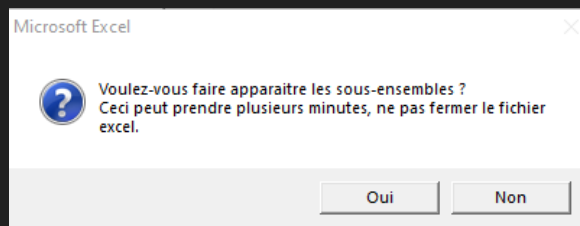
Recherche Nomenclature :			Verification ROHS d'une Nomenclature
Extraction d'une Nomenclature	Extraction d'une Nomenclature avec Fournisseur	Nomenclature valorisé du jour	Verification Obso d'une Nomenclature

- Je devais m'attaquer à tous ces boutons sauf à « **Nomenclature** valorisé du jour ». Chacun des quatre boutons utilisaient la même **macro**, la seule différence est qu'ils n'utilisent pas la même requête SQL. Expliquer ma démarche pour un seul de ces boutons revient donc à expliquer ma démarche pour les autres. Ainsi, nous allons seulement nous concentrer sur le bouton « Extraction d'une Nomenclature ».
- Ces boutons servent à afficher la nomenclature du composant électronique inscrit dans le champ de recherche. Cependant, un composant électronique est lui-même constitué de plusieurs composants, et ces mêmes composants peuvent aussi avoir une nomenclature et ainsi de suite. Cela est très contraignant pour les collaborateurs car ils doivent toujours retourner en arrière et réinscrire le code article du composant qui possède la sous-nomenclature dans le champs de recherche.
- Le but de ma tâche est d'afficher toutes les sous-nomenclatures de la nomenclature principale. Chaque sous-nomenclature se trouvera dans des feuilles différentes du classeur, des liens hypertexte seront à disposition dans la nomenclature mère pour accéder aux nomenclatures filles. Et pour finir, un lien hypertexte qui va de la fille vers la mère, c'est-à-dire un bouton « retour ».
- Remarque :
 - Macro : semblable à une fonction en programmation, seulement elle est attachée directement à un fichier Excel.
 - Nomenclature : Dans le cas présent c'est un ensemble de composant électronique

Tâche numéro 1 : Constat état actuel

- Voyons voir comment le bouton fonctionne si nous inscrivons dans le champs de recherche « CA 41244-002D » (c'est un code d'un composant électronique).

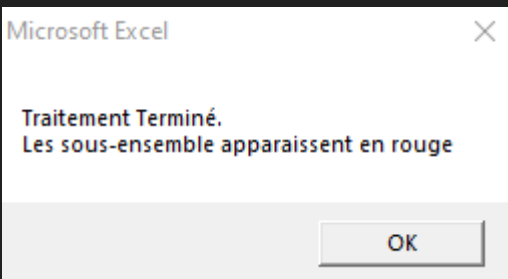
- Premièrement il nous affiche cela :



- Si nous cliquons sur « non » :

Nomenclature	Code Article	Quantité	Désignation	Repère
CA 41244-002D	ER 44137-000	1	RES 0805 0 ohm J 1/8W	C300
	FCA41244-XXXD	1	FICTIF COMMUNS CARTE CA 41244-XXXD	
	H5605458LF	1	RES 0805 100U F 1/8W	R315
	NON MONTE	0	COMPOSANT NON MONTE	R316

- Sinon :



->

Nomenclature	Code Article	Quantité	Désignation	Repère
CA 41244-002D	ER 44137-000	1	RES 0805 0 ohm J 1/8W	C300
	FCA41244-XXXD	1	FICTIF COMMUNS CARTE CA 41244-XXXD	
	H5605458LF	1	RES 0805 100U F 1/8W	R315
	NON MONTE	0	COMPOSANT NON MONTE	R316

Tâche numéro 1

```
' transformation en lien hypertexte
xlWs.Hyperlinks.Add xlWs.Range("B" & icol), Address="", _
SubAddress:="" & xlWs.Range("B" & icol).Value & "'!A1", _
TextToDisplay:=xlWs.Range("B" & icol).Value
' création des différentes feuilles de calcul
xlWb.Sheets.Add(after:=xlWb.Sheets(xlWb.Sheets.Count)).Name = xlWs.Range("B" & icol).Value
```

- J'ai commencé par transformer les code Articles, possédant un sous ensemble en lien-hypertexte. J'ai rajouté un bout de code juste après que le sous ensemble se transforme en rouge. Et pour chaque transformation en lien hypertexte, je crée une feuille qui possède le même nom que le code Article où le lien hypertexte la vise. `xlWs` est la feuille numéro 1 du classeur et `xlWs.Range(xy).Value` permet de cibler la valeur de la cellule `xy` dans la feuille 1 du classeur. `SubAddress` définit la cellule de la feuille que va cibler le lien, ici c'est la cellule `A1` de la feuille qui a le même nom que le code article qu'on vient de transformé en lien.

- J'ai rajouté 3 autres petites lignes de codes :

- Le tableau `rep()` contient tous les codes articles qui possèdent une nomenclature, il permet de ne pas répéter les sous-ensembles dans le classeur.

```
rep(k) = xlWs.Range("B" & icol).Value
```

- La variable `k` est égale au nombre de feuille totale dans le classeur, on lui ajoute 1 à chaque feuille créer. Elle permettra de stopper le code car la boucle principale qui affiche les sous ensembles est : `while f < k Or f = 1`, en effet `f` est la numérotation de la feuille, donc si nous avons pas parcourut toutes les feuilles ou alors que nous parcourons la toute première feuille, nous entrons dans la boucle.

```
k = k + 1 'k = au nombre de feuille donc on ajoute une feuille pour chaque sous nomenclature
```

- La variable `c` est égale au nombre de sous ensemble dans la nomenclature sur la quelle on est entrain de travailler, on la réinitialise à 0 quand on change de nomenclature mère.

```
c = c + 1 'variable qui va définir la taille du tableau des sous-nomenclatures
```


Tâche numéro 1

```
Dim l()  
ReDim l(1 To c)  
Dim q As Integer  
Dim m As Variant  
  
For Each cel In xlWs.Cells.Range("B2", xlWs.Cells.Range("B2").End(xlDown))  
    If cel.Interior.Color = RGB(255, 64, 64) Then  
        q = q + 1  
        l(q) = cel  
    End If  
Next  
  
q = 0  
  
'Affichage des nomenclatures dans les autres feuilles  
For Each m In 1
```

- En second temps Je fais une condition $\text{if } c > 0$, autrement dit si la nomenclature possède au moins un sous-ensemble on fait ce qui est inscrit ci-dessous sinon nous passons à la feuille suivante : $f = f + 1$
- je crée un tableau $l()$, ça longueur va de 1 à c .
- A l'aide d'une boucle qui parcourt la colonne B de la nomenclature, je crée une condition : si la couleur de fond est rouge j'ajoute au tableau, à l'indice q , le nom du code article.
- Ensuite on répète le programme principale pour chaque terme dans le tableau $l()$.
 - m va être égale au nom de la sous-nomenclature, et permettra donc de viser la feuille en question car la feuille possède déjà le nom de la sous-nomenclature.

Tâche numéro 1

- Enfin, il nous reste le bouton « retour ». Pour cela j'ai ajouté cette partie de code avant de faire les sous-ensembles

```
If f = 1 Then
    retour = "Principale Nomenclature"
Else
    retour = xlWb.Worksheets(f).Name
End If
```

- `xlWb` est une variable de type objet, c'est qui définit le classeur dans ce code, et `worksheet(f)` définit la feuille numéro `f`. Ainsi `xlWb.worksheet(f).Name` définit le nom de la feuille numéro `f` du classeur `xlWb`, c'est ce qu'on affecte à la variable `retour`.
- Et à la toute fin d'un traitement de sous ensemble j'ai rajouté ceci :

```
'lien à la page précédente
xlWb.Sheets(m).Hyperlinks.Add xlWb.Sheets(m).Range("A1"), Address:="", _
SubAddress:="" & retour & "!A1", _
TextToDisplay:=xlWb.Sheets(m).Range("A1").Value
```


Tâche numéro 1 : Résultat final

- Voyons voir ce que cela donne avec le même exemple précédent mais avec la mise à jour. Passons directement à « oui » pour afficher les sous-nomenclatures car sinon c'est le même résultat qu'avec l'exemple précédent. Nous pouvons y voir Le bouton « nomenclature » qui est le bouton retour. Et les deux dernières feuilles qui sont des sous-sous-nomenclatures.

[illegible]

	A	B	C	
1	<u>Nomenclature</u>	Code Article	Quantité	
2		DE 01684-030	1	RONDE
3		DE 05435-030	1	RONDE
4		DE 40097-001	0,008	GAINE T
5		DE 41244-000C2	1	C.IMP O
6		ER 40829-001	12	COSSE F
7		ER 40829-002	3	COSSE F
8		ER 40966-000	29	POINT D
9		ER 40990-000	3	COSSE F
10		ER 41924-049	1	QUARTZ
11		ER 41924-100	1	QUARTZ
12		ER 42074-101	2	CAPE 10
13		ER 42074-471	1	CAPE 47
14		ER 42082-106	11	CAPT B
15		ER 42084-472	2	CAPC 12
16		ER 42096-106	1	CAPE O
17		ER 42103-105	7	CAPC 12
18		ER 42107-331	4	CAPC 12
19		ER 42109-102	6	CAPC 08
20		ER 42109-152	1	CAPC 08
21		ER 42109-330	2	CAPC 08
22		ER 42110-103	2	CAPC 08
23		ER 42110-103	3	CAPC 08
24		ER 42114-104	2	C.PPS M
25		ER 42114-333	1	C.PPS M
26		ER 43055-000	1	DIO SCH
27		ER 43077-012	1	DIO PRO
28		ER 43101-000	2	DIO MIC
29		ER 43505-002	1	TRA NP
30		ER 43552-000	2	TRA NM
31		ER 43556-000	1	TRA NM
32		ER 43559-000	1	TRA PF

	A	B	C	D	
58	FCA41244-XXXX	ER 44151-015	1	RESEAU DIVISEUR SOT23 1K/25K	
59		ER 46012-680	1	IND 910MA .16R K 68U	
60		ER 46030-000	1	IND 5.4A 0.042R K 100µH	
61		ER 47161-000	1	IC RS232 LT1180C SOL18	
62		ER 47175-000	4	IC PHOTOCOUPLEUR TRANS SIMPLE TLP124	
63		ER 47198-000	1	REF 2.5V 0.2% 100PPM SOT23	
64		ER 47213-030	1	IC REG TENSION MIC5235-3.0YM5 SOT23-5	
65		ER 47213-033	1	IC REG TENSION MIC5235-3.3YM5 SOT23-5	
66		ER 47213-050	2	IC REG TENSION MIC5205-5.0YM5 SOT23-5	
67		ER 47216-000	3	PHOTOCOUPLEUR LOG HCPL0201	
68		ER 47218-006	1	RESET TEMPORISE SOT23-3 2.63V	
69		ER 47225-000	1	IC CONTROLEUR CHARGE RAPIDE BQ2003	
70		ER 47227-005	1	CONVERT DC/DC 1W 5V/5V	
71		ER 47229-030	1	IC REG TENSION SOT23-5 .1A-3.0V	
72		ER 47232-000	1	IC PHOTO MOS 1T 350V AQY210S SOP4	
73		ER 47233-000	1	IC CONVERT A/D 24 BITS SOP20 CS5532	
74		ER 47242-000	1	IC AMPLI DIF SO8 LT1168	
75		ER 47260-000	1	IC AMPLI23 AD8628	
76		ER 47610-000	1	IC LOG NAND 2ENT HC00	
77		ER 47618-000	1	IC COMPAR CMOS X2 TLC3702C SO8	
78		ER 47682-013	1	IC LOG BASCULE D 2 4013B	
79		ER 47682-093	1	IC LOG NAND TRIGGER SO14 4093B	
80		ER 47693-000	1	CONVERT DC/DC 12V/-12V ICL7660SCBAZ	
81		ER 47710-901	1	CONVERT DC/DC ELEVATEUR MAX771 SO8	
82		ER 47764-000	1	MEM EEPROM 32KX8 SO8 24LC256	
83		ER 47770-000	1	AMPLI OP CMOS RRIO LMC7101 SOT23-5	
84		ER 47774-000	1	IC MICRO M16C62 3V 128K+FLASH 100P65-A	
85	ER 47787-001	1	MICRO M16C62P		
86	ER 47881-000	1	AMPLI OP LTC2057HV SO8		
87	ER 48001-001	0,12	CON DORE MALE DROIT BERG 11.8MM 36C		
88	ER 48189-034	1	CON MALE HE10 DROIT VER.C.34C		
89	ER 48207-010	2	CON MALE HE10 COUDE SANS V 10C		
90	ER 48306-163	1	FUSIBLE RAPIDE 6.3X32 HPC 250V 16A		
91	ER 48306-000	1	SUPPORT FUSIBLE 6.3X32		
		Principale Nomenclature	FCA41244-XXXX	ER 47175-000	ER 47710-001

Tâche numéro 2

- Pour ma seconde mission, j'ai dû créer un bouton qui fonctionne similairement que le bouton « Recherche par désignation (comportant le champs) ». Ce bouton affiche les données de la table « ART » en fonction de la référence du code article, appelé « désignation ». Les données sont : « codeArticle », « ref fournisseur », « coutFabrication ».
- Ma mission, ici, consiste à afficher en plus la référence des fournisseurs, leur nom et leur code, à partir du « codeArticle », seulement pour une « désignation », il y n'a qu'un seul « codeArticle » et qu'un seul « coutFabrication » mais il peut y avoir, au minimum 0 noms, codes et références fournisseurs.

Tâche numéro 2

Ancienne Version

L'exemple, pris dans le champs de recherche est : « cac 00 »

1	CodeArticle	Designation1	CoutFabrication			
2	ER 60188	CACHE PROTECTION M12 035XXXX22003	0			
3	ER 60188-012	CACHE PROT.M12 NOIR 035122122003	0,286551724			
4	H5795169	CACHE TIROIR MP7000	0			
5	H5865021	CACHE BORNE OT160 A 800	8,77			
6						
7						

Nouvelle Version

1	CodeArticle	Designation1	CoutFabrication	CodeFournisseur	nomFournisseur	Ref Fournisseur
2	ER 60188	CACHE PROTECTION M12 035XXXX22003	0	No Data	No Data	No Data
3	ER 60188-012	CACHE PROT.M12 NOIR 035122122003	0,286551724	F00066	DIRECT	035 1221 22 003 Rohs
4	ER 60188-012	CACHE PROT.M12 NOIR 035122122003	0,286551724	F00153	IMPULSION	035 1221 22 003 Rohs/SKIFFY
5	ER 60188-012	CACHE PROT.M12 NOIR 035122122003	0,286551724	S4625	SKIFFY	035 1221 22 003 Rohs
6	H5795169	CACHE TIROIR MP7000	0	No Data	No Data	No Data
7	H5865021	CACHE BORNE OT160 A 800	8,77	F00121	ETN	1SCA 022 731 R8150 Rohs/ABB
8				S1037	ABB	1SCA 022 731 R8150 Rohs

Tâche numéro 2

- Je n'ai rien supprimé de l'ancien code, je lui ai simplement rajouté quelques lignes. Après avoir créer les données dans le classeur crée (ancienne version), mon code parcourt la colonne A jusqu'à arriver à une cellule vide.

```
For Each cel In xlWs.Cells.Range("A1", xlWs.Cells.Range("A1").End(xlDown))
```

- 3 Variables sont très importantes durant ce code : **o**, **a** et **x**
 - La variable **o** prend + 1 pour chaque itération. Elle définira sur quel code article on travaillera.
 - La variable **a** sera égale au nombre de codeArticle qui reste à terminer + 1.
 - La variable **x** sera égale au nombre de ligne qui composent la plus grande colonne.

Tâche numéro 2

- Premièrement nous vérifions si les données existent dans la table, si elles existent on les affichent à partir de la ligne o et de la colonne 4 (colonne D).
 - Sinon de la colonne D à G, à la ligne o, nous écrivons « No Data » en rouge.

```
If rs.EOF Then
    xlWs.Cells.Range("D" & CStr(o) & ":" & "G" & CStr(o)).Value = "No Data"
    xlWs.Cells.Range("D" & CStr(o) & ":" & "G" & CStr(o)).Font.Color = RGB(255, 0, 0)
Else
    xlWs.Cells(o, 4).CopyFromRecordset rs
End If
```

Tâche numéro 2

- Après avoir afficher les données, x prend donc le nombre de ligne de la colonne la plus longue, c'est-à-dire la colonne D ou E ou G.
- Si nous ne sommes pas au dernier « codeArticle » nous effectuons les instructions si dessous
 - a = au nombre de « codeArticle » qui reste + 1.
 - Nous copions une plage de cellule qui va du codeArticle suivant ($o + 1$) au dernier ($a + o$), puis nous la collons à partir de la ligne juste après x . Nous faisons tout cela entre les colonnes A et C.
 - Puis nous répétons les valeurs de la ligne o (entre la colonne A et C) jusqu'à x .

```
x = xlWs.Cells.Range("D1").End(xlDown).Row

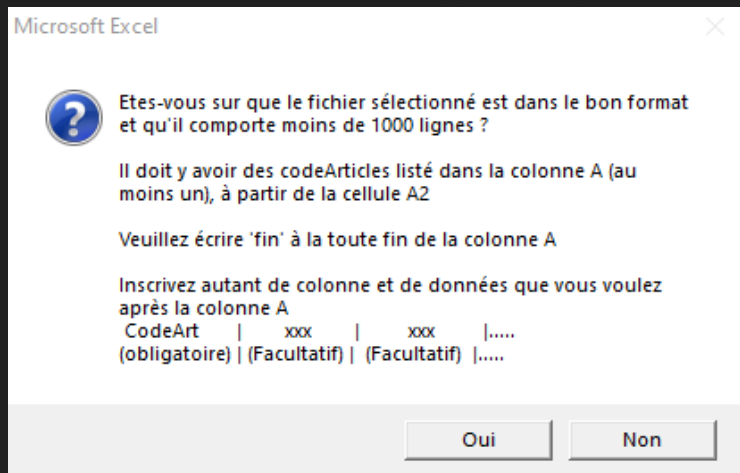
If xlWs.Cells.Range("A" & CStr(o + 1)).Value <> "" Then
    a = xlWs.Cells.Range("A" & CStr(o)).End(xlDown).Row + 1
    'copy colle les données dont on a pas utilisé jusqu'à la colonne la plus longue
    xlWs.Cells.Range("A" & CStr(o + 1) & ":C" & CStr(a + o)).Copy xlWs.Cells.Range("A" & CStr(x + 1) & ":C" & CStr(a + x))
    'répète les mêmes valeurs jusqu'à la colonne la plus lognue
    xlWs.Cells.Range("A" & CStr(o)).Copy xlWs.Cells.Range("A" & CStr(o) & ":A" & CStr(x))
    xlWs.Cells.Range("B" & CStr(o)).Copy xlWs.Cells.Range("B" & CStr(o) & ":B" & CStr(x))
    xlWs.Cells.Range("C" & CStr(o)).Copy xlWs.Cells.Range("C" & CStr(o) & ":C" & CStr(x))
End If
```

Tâche numéro 2

- Pour finir nous remplissons la plage de cellule entre o et x de couleur jaune si ce n'était pas le cas pour l'ancienne plage.
- Et nous finissons le code par une affectation $o = x$ car le prochain codeArticle s'est fait déplacé à $x + 1$, et comme à la prochaine itération o va prendre 1 alors o sera égale à $x + 1$ et sera donc à la bonne ligne.

Tâche numéro 3

- Le client m'a demandé d'ajouter un bouton « création de prototype ». Cela consisterait à ce que l'utilisateur implémente les données d'une nomenclature dans un fichier Excel à la main. Suite à cela, lorsqu'on appuierait sur le bouton, le programme nous proposerait de cibler un classeur Excel enregistré (ce serait la nomenclature faite à la main par le salarié)



- Puis le programme lui afficherait toutes les données nécessaire dans ce même classeur pour créer son prototype.

Tâche numéro 3

Le classeur Excel implémenté à la main

CodeArticle	CoutFabrication			
ER 60188	0			
ER 60188-012	2,5			
H5795169	8			
H5865021	8			
fin				

Transformation du classeur par le code

CodeArticle	CoutFabrication	Designation du code Article	Nom fournisseur	Ref Fournisseur
ER 60188		0 CACHE PROTECTION M12 035XXX22003	No Data	No Data
No Data	No Data	No Data	No Data	No Data
No Data	No Data	No Data	No Data	No Data
ER 60188-012	2,5	2,5 CACHE PROT.M12 NOIR 035122122003	DIRECT	035 1221 22 003 Rohs
			IMPULSION	035 1221 22 003 Rohs/SKIFFY
			SKIFFY	035 1221 22 003 Rohs
No Data	No Data	No Data	No Data	No Data
H5795169	8	8 CACHE TIROIR MP7000	No Data	No Data
H5865021	8	8 CACHE BORNE OT160 A 800	ABB	1SCA 022 731 R8150 Rohs
			ETN	1SCA 022 731 R8150 Rohs/ABB

Tâche numéro 3

- La démarche de cette tâche est presque similaire à celle du numéro 2.
- Seulement ici je n'ai pas répété les données, j'ai laissé des champs vide.

Tâche numéro 3

- Tout d'abord, il m'a été demandé à ce que le code doive fonctionner même si l'utilisateur laissait des champs vides, donc pour définir la fin du classeur, je leur ai imposé d'écrire « fin » à la fin de la **colonne A**.
- La boucle principale du code est une boucle **while**, tant que **condition = 0**, le programme continue. **Condition = 1** lorsque le codeArticle suivant = « fin ».

Tâche numéro 3

```
For h = 1 To c
  For g = 2 To fin - 1

    If Cells(g, h).Value = "" Then
      Cells(g, h).Value = "No Data"
      Cells(g, h).Font.Color = RGB(255, 0, 0)
    End If
  Next
Next
```

- Avant d'ajouter les données nécessaires pour l'utilisateur, je comble les cellules vides en y ajoutant « No Data » en rouge.
- Une première boucle qui va de 1 à c, c est égale au nombre de colonne que l'utilisateur a initié. La première boucle va donc parcourir les colonnes.
- Naturellement la deuxième boucle va parcourir les lignes jusqu'à la dernière ligne. En effet fin prend le numéro de la ligne de la cellule où il y a écrit « fin ».
 - Elle commence par la 2^e ligne car la première ligne correspond au titre des colonnes et non aux données.

Tâche numéro 3

- À la différence de la tâche numéro 2, les données qui sont initiées dans le classeur, et qui servent à la recherche dans la base de donnée, ne changent pas de ligne pendant les recherches des données mais après les recherches dans la base de donnée.
- En sachant que pour un code article il n'y a qu'une seule désignation du code article (ce sont les premières données qu'ont recherche dans la base), j'ai décidé que après les implémentation des données sur le fichier Excel, je relèverai les numéros de ligne de toutes les désignations et rattachée les données initiés par l'utilisateur par ces lignes.

Tâche numéro 3

```
'Relève les lignes des désignations du code article
Dim td()
ReDim td(1 To ca)
x = 0

For ligne = 2 To o

    If Cells(ligne, c + 1).Value <> "" Then
        x = x + 1
        td(x) = ligne
    End If

Next
```

- J'ai créé un tableau `td()` pour stocker les numéros de lignes, la taille du tableau est défini par le nombre de code article (`ca`).
- La boucle va de 2 à `o` (`o` = à la toute dernière ligne)

Tâche numéro 3

'Relève les données inscrit par l'utilisateur

x = c * ca

y = 0

Dim tu()

ReDim tu(1 To x)

For u = 2 To ca + 1

For v = 1 To c

y = y + 1

tu(y) = Cells(u, v).Value

Cells(u, v).Value = ""

Next

Next

- Le tableau `tu()` va stocker toutes les valeurs initiées par l'utilisateur, même les cellules qui étaient vides, car elles ont été remplacées par « No Data »

Tâche numéro 3

'Déplacement des données inscrit par l'utilisateur

y = 0

For d = 1 To ca

x = td(d)

For p = 1 To c

If tu(p + y) <> "No Data" Then

Cells(x, p).Value = tu(p + y)

Else

Cells(x, p).Value = tu(p + y)

Cells(x, p).Font.Color = RGB(255, 0, 0)

End If

Next

y = y + c

Next

- Maintenant nous déplaçons les données, grâce aux données implémentées dans les deux tableaux `td()` et `tu()`. Puis j'ai créé une condition de si la valeur qu'on utilise depuis de le tableau `tu()` est égale à « No Data », si ce n'est pas le cas, on le met en noir sinon en rouge.

Tâche numéro 3

'Derniere ligne du tableau

```
If Cells(Rows.Count, c + 1).End(xlUp).Row >= Cells(Rows.Count, c + 2).End(xlUp).Row Then  
    dernier = Cells(Rows.Count, c + 1).End(xlUp).Row  
Else  
    dernier = Cells(Rows.Count, c + 2).End(xlUp).Row  
End If
```

- Puis nous finissons par rendre le fichier un peu plus coloré.
- La variable **dernier** prend la dernière ligne du tableau.

Tâche numéro 3

- La variable **couleur** est soit égale à 1 ou 0, elle change à chaque itération, elle définira la couleur utilisée (jaune au bleu).
- **vale** et **vale2** définissent l'intervalle des lignes qui va être colorié. Si on arrive à la dernière ligne alors **vale2** est égale à **dernier + 1**.

'Mise en forme du fichier avec des couleurs
couleur = 1

For boucle = 1 To ca

'plage de cellule qui prend la couleur

If boucle <> ca Then

vale = td(boucle) '(ligne du début)

vale2 = td(boucle + 1) '(ligne de fin)

Else

vale = td(boucle) '(ligne du début)

vale2 = dernier + 1 '(ligne de fin)

End If

'Change de couleur

If couleur = 1 Then

For x = 1 To c + 3

For y = vale To vale2 - 1

Cells(y, x).Interior.Color = RGB(151, 185, 240)

Next

Next

couleur = 0

Else

For x = 1 To c + 3

For y = vale To vale2 - 1

Cells(y, x).Interior.Color = RGB(255, 216, 0)

Next

Next

couleur = 1

End If

Next

End Sub

Tâche numéro 4

A	B
Liste Resistance	Liste Condensateur
RES 0055	CAPA
RESIS 0055	COND
	CAPC
	CAPT
	CAPCC
	CAPCT

- Dans le fichier Excel qui comprend l'interface de la recherche de la base de donnée, j'y ai ajouté une seconde feuille.
- Dans cette deuxième feuille, les collaborateurs notent différentes **propriétés/noms** en fonction du nom de la colonne, donné par l'utilisateur. Par exemple la colonne A donne une « Liste Resistance ».
- Pour utiliser la **feuille 2** j'ai créé une liste déroulante dans la **feuille 1** qui prendra en compte le nom d'une colonne. Ainsi lorsque le salarié fera une recherche avec une colonne sélectionné, la requête du bouton sélectionné compilera autant de fois qu'il y ai de propriété dans la colonne. Par conséquent la requête donnera :
 - **barre de recherche + propriété1, barre de recherche + propriété2.**
- Enfin, il faut créer un bouton qui met à jour la liste déroulante.

Tâche numéro 4

'Colonne maximum

For k = 1 To 1000

If Worksheets("Feuil2").Cells(1, k).Value = "" Then

c = k - 1

k = 1000

End If

Next

'Ajout de donnée dans la liste déroulante

For i = 1 To c

Feuil1.ComboBox1.AddItem Worksheets("Feuil2").Cells(1, i).Value

Next

- La variable **c** prendra en compte le nombre de colonne au maximum.
- Puis on fait une boucle allant de **1** à **c**. Puis il ajoute à la liste déroulante les noms de toutes les colonnes de **1** à **c**.

Tâche numéro 4

Function Liste() As Integer

'nassim 28/06/2022

If Feuil1.ComboBox1.Value <> "-----" Then 'si la liste déroulante possède une valeur

'trouve la colonne en fonction de la valeur de la liste déroulante

For c = 1 To 1000

If Worksheets("Feuil2").Cells(1, c).Value = Feuil1.ComboBox1.Value Then

k = c

c = 1000

End If

Next

Liste = k 'la fonction retourne le numéro de la colonne qui est égale à la valeur

Else 'sinon la fonction retourne 0

Liste = 0

End If

End Function

- Si nous sélectionnons une colonne alors on parcourt la ligne 1 jusqu'à trouvé le même nom que nous avons sélectionné. Lorsque le programme le trouve, la fonction **Liste()** retourne le numéro de la colonne.
- Sinon la fonction retourne 0.

Tâche numéro 4

```
If Worksheets("Feuil2").Cells(i, col).Value <> "" Then
```

```
    If ActiveSheet.StringRecherche.Value = vbNullString Then  
        valueCodeArt = Worksheets("Feuil2").Cells(i, col).Value  
    Else
```

```
        v = Worksheets("Feuil2").Cells(i, col).Value  
        valueCodeArt = ActiveSheet.StringRecherche.Value & " " & v
```

- Ainsi J'ai modifié de la même manière toutes les requêtes. J'y ai ajouté tout d'abord une condition pour savoir si la fonction retourne 0, si c'est le cas le code ne change pas et fonctionne comme l'ancienne version.
- La variable `col` est égale à ce que retourne la fonction `Liste()`
- `ValueCodeArt` correspond à la variable dans la requête qui fait suite au « `Where...` ».
- Il y a une condition pour savoir s'il n'y a rien d'écrit dans la barre de recherche.
 - Si c'est le cas alors `ValueCodeArt` prend comme valeur seulement ce qu'il y a écrit dans la colonne en question à la ligne `i` (`i` est le paramètre de la boucle qui prend +1 à chaque itération à partir de 2, la boucle finit quand une cellule est vide)
 - mais si ce n'est pas le cas `ValueCodeArt` prend comme valeur ce qu'il y a écrit dans la barre de recherche et ce qu'il y a écrit dans la colonne en question à la ligne `i` (avec un espace entre les deux valeurs).