Rapport de Stage

Sommaire

1. Semaine 1:

1.1 Développement de la Fonction SupDoublonsArray

La première semaine a été dédiée à la conception et à la mise en œuvre de la fonction `SupDoublonsArray`. Cette fonction a été développée pour éliminer les doublons d'un tableau multidimensionnel en se basant sur deux colonnes clés.

vba
```
Function SupDoublonsArray(Tbl, Optional colClé1, Optional colClé2)
    ' ... Logique pour supprimer les doublons ...
End Function
```

1.2 Préparation des Données

Suite à la création de la fonction, celle-ci a été appliquée à un tableau extrait de la feuille "ZSUIVI_EXEC" afin de supprimer les doublons de la première colonne. Le tableau sans doublons, `TabSansDoublons`, a été obtenu de la manière suivante :

vba
```
' Suppression des doublons du tableau TabGeneral
TabSansDoublons = TabGeneral
TabSansDoublons = SupDoublonsArray(TabSansDoublons, 1)
```

1.3 Traitement des Données et Construction du Tableau Résultat

Après avoir éliminé les doublons du tableau extrait de la feuille "ZSUIVI_EXEC", le focus s'est tourné vers la construction du tableau résultat, `TabResultat`. Celui-ci a été dimensionné pour accueillir les données consolidées et calculées à partir des colonnes spécifiques.

vba
```
' Définir les dimensions du tableau TabResultat
nbcol_TabResultat = 31
nblig_TabResultat = UBound(TabSansDoublons)
ReDim TabResultat(1 To nblig_TabResultat, 1 To nbcol_TabResultat)
```

Ensuite, une boucle a été utilisée pour remplir `TabResultat` en fonction des colonnes spécifiées et des conditions définies pour récupérer les valeurs les plus pertinentes.

vba
```
For i = 2 To nblig_TabResultat
    ' Assignation des valeurs initiales
    TabResultat(i, 1) = TabSansDoublons(i, 1)
    TabResultat(i, 12) = TabSansDoublons(i, 4)
    TabResultat(i, 23) = 0
    TabResultat(i, 25) = 0
```

```
    TabResultat(i, 27) = 0

    ' Boucle pour remplir TabResultat en fonction des colonnes
spécifiées
    For a = 2 To UBound(TabGeneral)
        If TabGeneral(a, 1) = TabSansDoublons(i, 1) Then
            ' Traitement des données pour récupérer les valeurs
appropriées
            ' ...
            ' Calcul des sommes
            If TabResultat(i, 1) = TabGeneral(a, 1) Then
                ' Calcul des sommes pour les colonnes spécifiées
                TabResultat(i, 20) = TabResultat(i, 20) +
TabGeneral(a, 13)
                TabResultat(i, 21) = TabResultat(i, 21) +
TabGeneral(a, 14)
                TabResultat(i, 22) = TabResultat(i, 22) +
TabGeneral(a, 17)
                TabResultat(i, 24) = TabResultat(i, 24) +
TabGeneral(a, 24)
                TabResultat(i, 26) = TabResultat(i, 26) +
TabGeneral(a, 30)
            End If
        End If
    Next a
Next i
```

Cette section de code traite chaque ligne du tableau résultat en utilisant les données du tableau général pour remplir les colonnes spécifiées et calculer les sommes.

1.4 Intégration des Données de l'Annuaire

Enfin, le tableau résultat a été enrichi en intégrant les données de l'annuaire. La feuille "Annuaire" a été sélectionnée, et un tableau `TabAnnuaire` a été créé pour stocker les informations pertinentes.

```
vba
' Ramener les 3 colonnes de l'annuaire, les mettre dans un tableau,
puis les mettre dans le tableau résultat
Sheets("Annuaire").Select
nblig_tabAnnuaire = Cells(Rows.Count, 6).End(xlUp).Row
ReDim TabAnnuaire(1 To nblig_tabAnnuaire, 1 To 4) ' 4 colonnes
```

```
' ...
' Boucle pour récupérer les données de la feuille Annuaire
' ...
' Remplir TabResultat avec TabAnnuaire
' …
```

2. Semaine 2:

2.1 **Intégration des données de la feuille "MAJ SIFAC"**

La première étape a été la sélection de la feuille "MAJ SIFAC" et la récupération des données nécessaires dans le tableau `TabControle`. Les doublons ont été éliminés à l'aide de la fonction `SupDoublonsArray`, et un nouveau tableau `TabControlef` a été créé pour stocker les données filtrées.

2.2 **Calcul des totaux et intégration au tableau existant**

Ensuite, des calculs ont été effectués sur `TabControle` pour obtenir des totaux en fonction de certaines conditions. Ces totaux ont été ajoutés au tableau principal, `TabResultat`, en utilisant les colonnes appropriées.

La démarche peut être résumée comme suit :

- **Sélection de la feuille "MAJ SIFAC" et récupération des données :**
  vba

```
' Sélection de la feuille MajSIFAC
Sheets("MAJ SIFAC").Select

' Calcul des lignes et colonnes du fichier
nbligTabControle = Cells(Rows.Count, 1).End(xlUp).Row

' Dimensionnement du tableau
ReDim TabControle(1 To nbligTabControle, 1 To 8) ' 8 colonnes
```

```
' Boucle pour récupérer les données de la feuille majsifac
For i = 2 To nbligTabControle
    TabControle(i, 1) = Cells(i, 1).Value ' Colonne 1
    TabControle(i, 2) = Cells(i, 2).Value ' Colonne 2
    ' ...
Next i
```

Suppression des doublons :

vba

```
' Supprimer les doublons de TabControle
TabControle = SupDoublonsArray(TabControle, 1, 2)
```

Calcul des totaux et intégration au tableau principal :

vba

```
' Redimensionnement du tableau filtré
ReDim TabControlef(1 To nbligTabControle, 1 To 5)

' Remplissage du tableau filtré avec les données spécifiques
For i = 2 To nbligTabControle
    TabControlef(i, 1) = Cells(i, 1).Value ' Colonne 1
    TabControlef(i, 2) = Cells(i, 2).Value ' Colonne 2
    TabControlef(i, 3) = Cells(i, 4).Value ' Colonne 4
    TabControlef(i, 4) = Cells(i, 5).Value ' Colonne 5
    TabControlef(i, 5) = Cells(i, 3).Value ' Colonne 3
Next i

' Calcul des totaux pour chaque doublon dans TabControle
For i = 2 To nblig_TabControlee
    For j = 2 To nbligTabControle
        If InStr(1, TabControlef(j, 5), "RG", vbTextCompare) = 0
Then
            If TabControle(i, 1) = TabControlef(j, 1) And
TabControle(i, 2) = TabControlef(j, 2) Then
                TabControle(i, 3) = TabControle(i, 3) +
TabControlef(j, 3)
                TabControle(i, 4) = TabControle(i, 4) +
TabControlef(j, 4)
            End If
        End If
    Next j
Next i
```

```
' Calcul des totaux dans TabResultat en fonction des doublons
For i = 2 To nblig_TabControlee
    For j = 2 To nblig_TabResultat
        If TabControle(i, 1) = TabResultat(j, 14) And TabControle(i,
2) = TabResultat(j, 15) Then
            TabControle(i, 5) = TabControle(i, 5) + TabResultat(j,
21)
            TabControle(i, 6) = TabControle(i, 6) + TabResultat(j,
26)
        End If
    Next j
Next i

' Calcul des écarts
For i = 2 To nblig_TabControlee
    For j = 2 To nblig_TabResultat
        If TabControle(i, 1) = TabResultat(j, 14) And TabControle(i,
2) = TabResultat(j, 15) Then
            TabControle(i, 7) = TabControle(i, 3) - TabControle(i,
6)
            TabControle(i, 8) = TabControle(i, 4) - TabControle(i,
5)
        End If
    Next j
Next i
```

Semaine 3:

3.1Calcul du Statut de la Commande

Le code effectue une évaluation minutieuse des données pour déterminer le statut approprié de chaque commande. Par exemple :

vb

```
If TabResultat(i, 20) + TabResultat(i, 21) = 0 And TabResultat(i,
26) = 0 Then
    TabResultat(i, 28) = "annulé"
ElseIf TabResultat(i, 20) + TabResultat(i, 21) <> 0 And
TabResultat(i, 26) = 0 Then
    TabResultat(i, 28) = "En cours"
```

```
ElseIf TabResultat(i, 20) + TabResultat(i, 21) <> 0 And
TabResultat(i, 26) <> 0 Then
    ' Gestion des cas où la commande est partielle ou soldée
    '...
End If
```

3.2 Application des Modifications au Fichier Principal

Une fois le statut de la commande déterminé, le code ouvre un fichier externe pour récupérer les mises à jour potentielles. Par exemple :

vba
```
Set ges = Workbooks.Open(filePath2)
Set feuilleSource = ges.Sheets(1)
```

Les données de ce fichier sont ensuite comparées(en fonction de quelques colonnes qui doivent être complétées par des gestionnaires) avec celles du fichier principal, et les changements sont appliqués en conséquence.

Le code parcourt les données des deux tableaux pour détecter les changements. Par exemple :

vba
```
For i = 2 To nblig_TabResultat
For j = 2 To nblig_TabNouveau
If TabResultat(i, 1) = TabNouveau(j, 1) Then
If TabResultat(i, 5) <> TabNouveau(i, 5) Then
TabResultat(i, 5) = TabNouveau(i, 5)
End If
If TabResultat(i, 6) <> TabNouveau(i, 6) Then
TabResultat(i, 6) = TabNouveau(i, 6)
End If
If TabResultat(i, 7) <> TabNouveau(i, 7) Then
TabResultat(i, 7) = TabNouveau(i, 7)
End If
If TabResultat(i, 8) <> TabNouveau(i, 8) Then
TabResultat(i, 8) = TabNouveau(i, 8)
End If
' Comparaison des valeurs pertinentes et mise à jour si nécessaire
            '...
        End If
    Next j
```

```
Next i
```

Semaine 3:

4.1Nettoyage et Mise en Forme des Données

Le code commence par effacer le contenu existant de la feuille "Resultat" et y écrit les nouvelles données provenant du tableau `TabResultat`. Les étapes incluent :

1. Effacement du contenu existant de la feuille "Resultat" :

vb
```
Sheets("Resultat").UsedRange.ClearContents
```

2. Écriture des nouvelles données sur la feuille "Resultat" :

vb
```
Sheets("Resultat").Range("A1").Resize(UBound(TabResultat),
UBound(TabResultat, 2)).Value = TabResultat
```

3. Détermination de la dernière ligne et de la dernière colonne avec des données :

vb
```
lastRow2 = Sheets("Resultat").Cells(Sheets("Resultat").Rows.Count,
"A").End(xlUp).Row
LastColumn = Sheets("Resultat").Cells(1,
Sheets("Resultat").Columns.Count).End(xlToLeft).Column
```

4. Référence à la dernière cellule de données :

vb
```
Set lastCell = Sheets("Resultat").Cells(lastRow2, LastColumn)
```

5. Application du style de tableau uniquement à la dernière ligne :

vb
```
Sheets("Resultat").ListObjects.Add(xlSrcRange,
lastCell.CurrentRegion, , xlYes).TableStyle = "TableStyleMedium14"
```

6. Ajout de sous-totaux aux colonnes spécifiques :

vb
```
Sheets("Resultat").Cells(lastRow2 + 1, 20).Formula = "=SUBTOTAL(9,
R2C:R[-1]C)"
' Répéter pour les autres colonnes...
```

Ces étapes assurent que les données sur la feuille "Resultat" sont nettoyées, formatées et prêtes à être utilisées.

4.2Création de Listes Déroulantes

Le code utilise la méthode `.Validation` pour chaque plage de cellules spécifiée, définissant des listes déroulantes avec des options prédéfinies.

vb
```
With Sheets("Resultat").Range("B2:B" & dernièreLigneee).Validation
    .Delete
    .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, _
         Formula1:="=Annuaire!$AC$2:$AC$8"
    '...
End With
' Répéter pour les autres colonnes...
```

Chaque liste déroulante est associée à une plage de cellules spécifique dans la feuille "Resultat", et les options disponibles sont définies en référençant les données d'une autre feuille, "Annuaire".

Semaine 5:

5.1 Création de fichier pour chaque gestionnaire, groupe et porteur

Création d'un tableau avec tous les gestionnaire, les porteurs et les porteur ou il est appliqué la fonction sans doublon .
vb
```
 For i = 2 To nblig_TabResultat
    TabGes(i, 1) = TabResultat(i, 2)
    TabPorteur(i, 1) = TabResultat(i, 3)

    TabGrp(i, 1) = TabResultat(i, 4)
```

```
Next i
' tab sans doublon de gesionnaire, Groupe, Porteur

   TabGes = SupDoublonsArray(TabGes, 1)
   TabGrp = SupDoublonsArray(TabGrp, 1)
   TabPorteur = SupDoublonsArray(TabPorteur, 1)
```

Création des fichiers pour chaque gestionnaire groupe et porteur avec toutes les données qui lui sont attribuées.

vb

```
For i = 2 To nblig_TabGes
    For A = 2 To nblig_TabResultat
 If TabResultat(A, 2) = TabGes(i, 1) Then
            nbligIncrement = nbligIncrement + 1
TabGesResult(nbligIncrement, 1) = TabResultat(A, 1)
TabGesResult(nbligIncrement, 2) = TabResultat(A, 2)
TabGesResult(nbligIncrement, 3) = TabResultat(A, 3)
etc

End If
    Next A
```

Ensuite mise en forme de la feuille et effacement du tableau pour accueillir les nouvelles données.

vb

```
    Set ges = Workbooks.Add
    nbligIncrement = 1

    ' Écrit sur la feuille 1 du nouveau classeur
   ges.Sheets(1).Range("A1").Resize(UBound(TabGesResult),
UBound(TabGesResult, 2)).Value = TabGesResult

  lastRow = Sheets(1).Cells(Sheets(1).Rows.Count, 1).End(xlUp).Row

' Trouver la dernière colonne avec des données dans la première
ligne
LastColumn = Sheets(1).Cells(1,
Sheets(1).Columns.Count).End(xlToLeft).Column
```

```
' Référence à la dernière cellule de données
Set lastCell = Sheets(1).Cells(lastRow, LastColumn)

' Appliquer le style de tableau uniquement à la dernière ligne
Sheets(1).ListObjects.Add(xlSrcRange, lastCell.CurrentRegion, ,
xlYes).TableStyle = "TableStyleMedium14"
'efface le tableau
    For y = LBound(TabGesResult, 1) To UBound(TabGesResult, 1)
        For Z = LBound(TabGesResult, 2) To UBound(TabGesResult, 2)
            TabGesResult(y, Z) = Empty ' Ou utilisez la valeur par
défaut appropriée
        Next Z
    Next y
nom = TabGes(i, 1)
' Enregistre le nouveau classeur
ges.Sheets("Feuil1").SaveAs filePath & nom & datee & ".xlsx"
 ges.Close
Next i
```

Exemple du fichier:

Numéro de flux	Gestionnaire	Groupe	Porteur	SIG	Matière	...
6	Gestionnaire2	Groupe4	Porteur6	...	...	...
7	Gestionnaire2	Groupe5	Porteur7	...	...	...
8	Gestionnaire2	Groupe5	Porteur8	...	...	...
9	Gestionnaire2	Groupe5	Porteur9	...	...	...
10	Gestionnaire2	Groupe6	Porteur10	...	...	...

Numéro de flux	Gestionnaire	Groupe	Porteur	SIG	Matière	...
1	Gestionnaire1	Groupe1	Porteur1	...	...	...
2	Gestionnaire1	Groupe1	Porteur2	...	...	...
3	Gestionnaire1	Groupe2	Porteur3	...	...	...
4	Gestionnaire1	Groupe2	Porteur4	...	...	...
5	Gestionnaire1	Groupe3	Porteur5	...	...	...