Université de METZ

Olivier HABERT

Septembre 2000

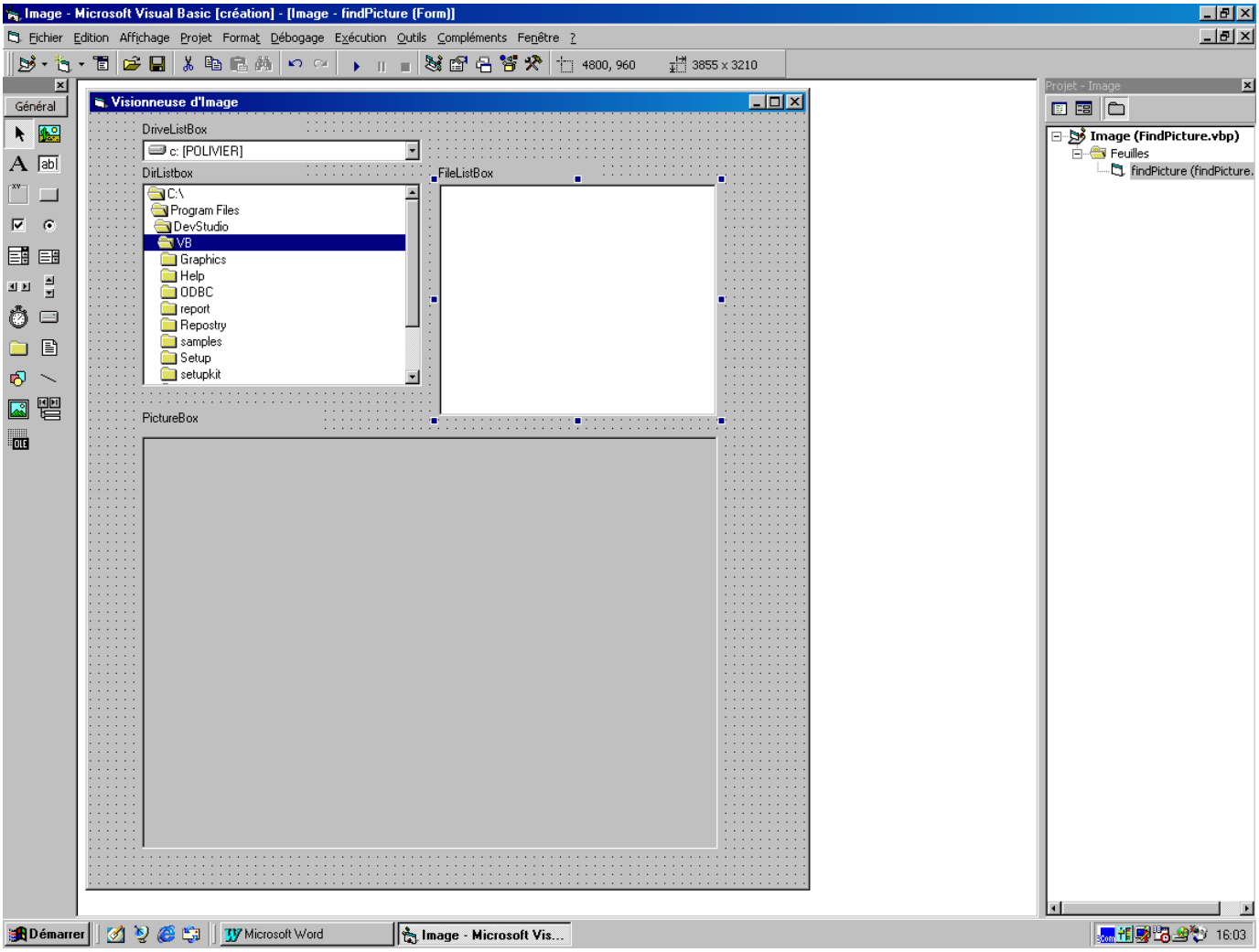
TD-TP Micro Informatique
Etude de Visual Basic

Notions à Assimiler

- Projet, fenêtre principale,
- Contrôles bouton, edit, label, picture
- Menus
- Création de boîte de dialogue
- Contrôles ouverture/enregistrement de fichier
- Barre de menu, barre de status, liste d'image, gestion d'onglets, contrôles spéciaux *(Toolbar, slider, barre de progression, …)*
- Instructions de Visual Basic *(if then, while, do, for, …)*
- Type de variable (integer, string, float, single…)
- Chaînes de caractère, Concaténation, Recherche d'Occurrence, *(mid, left, right,…)*
- Timers,
- Tableaux, Tableaux Dynamiques, Redimensionnement des Tableaux Dynamiques
- Base de données, lecture écriture de données dans une table, Création Dynamique de tables, de champs,…
- Contrôles spéciaux *(DBGRID, Calendriers, …)*

Exercice 1 :

On se propose de créer une visionneuse de fichiers images. Pour cela on va utiliser des contrôles comme le contrôle *DriveListBox*, *DirListBox*, *FileListBox*, pour la recherche des images sur le disque et le contrôle *PictureBox* pour visualiser l'image sélectionnée.



1) Créer un nouveau projet que vous nommerez **Image**
2) Placer les différents contrôles sur la frame principale
3) Configurer les différents contrôles *(mettre un filtre d'image dans le FileListBox,…)*
4) Ecrire le code adéquat réagissant au changement des différents contrôles *(DriveListBox met à jour DirListBox, DirListBox met à jour FileListBox et un Click sur le FileListBox Charge l'image sélectionnée)*
5) Tester le programme.

Exercice 2:

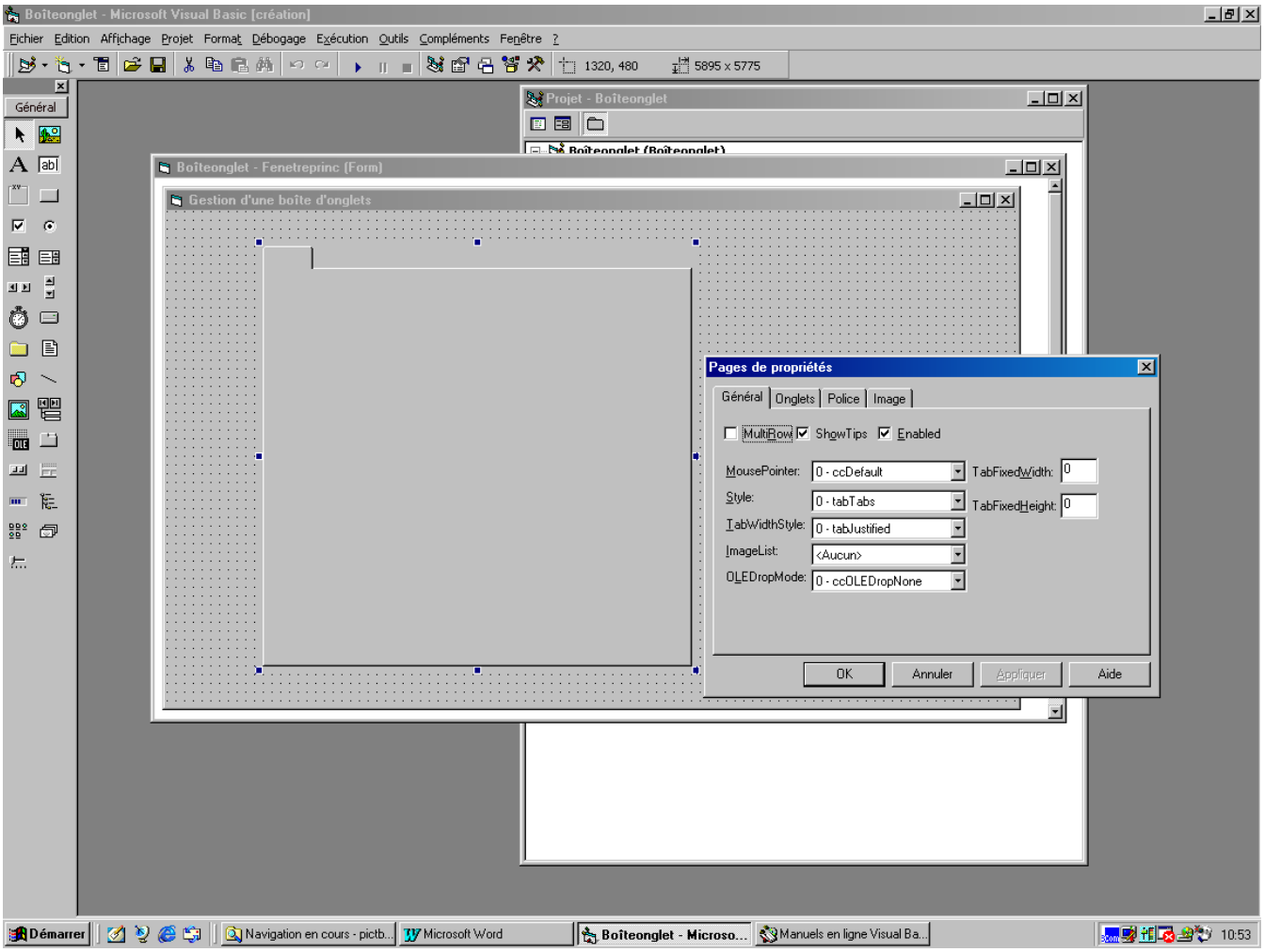
Gestion de Boîtes d'onglets

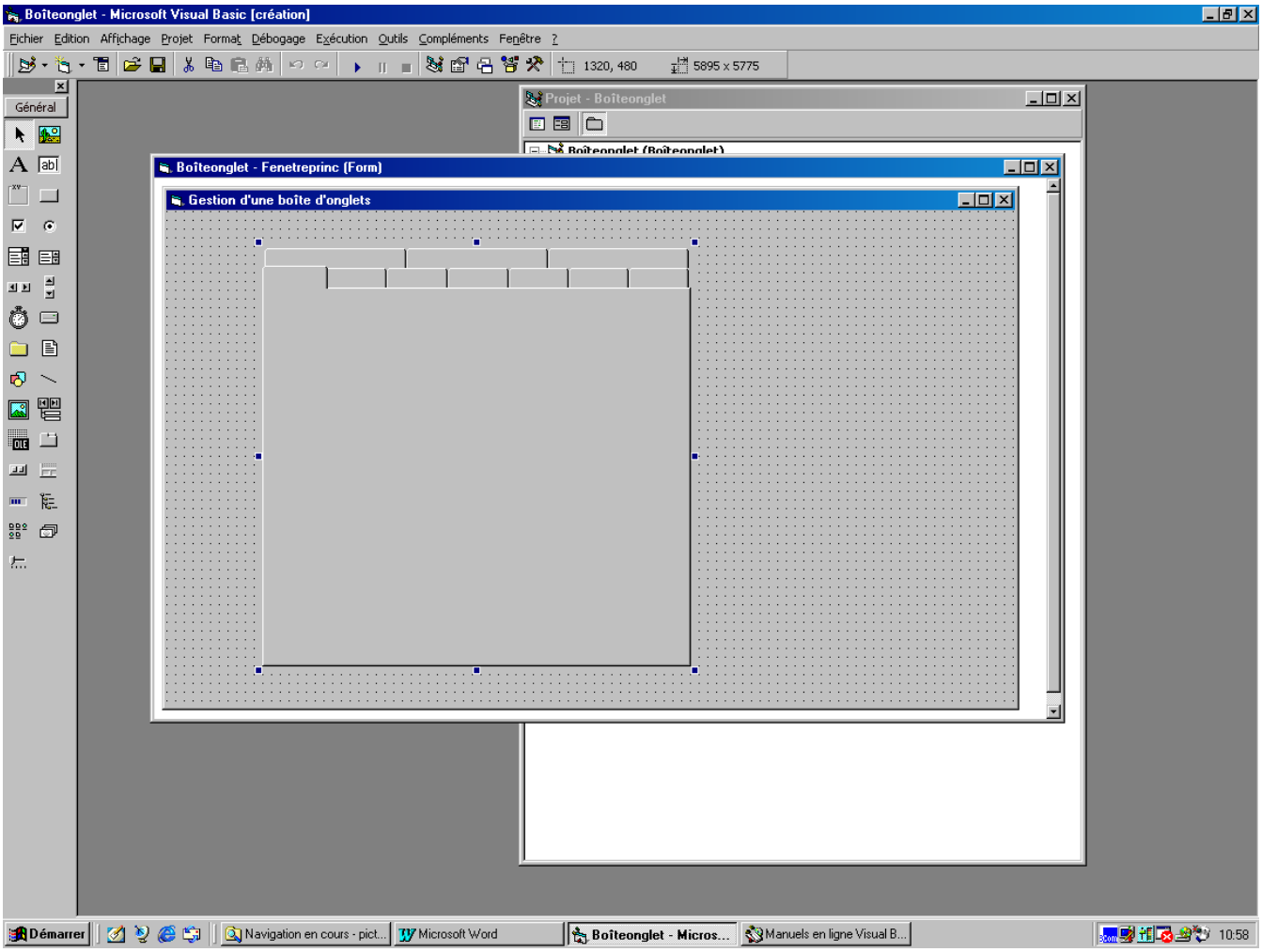
Une boîte d'onglets permet d'afficher un choix multiple de boîtes de dialogue dans une fenêtre en sélectionnant une étiquette *(onglet)*. Le contrôle *tabstrip* permet de gérer les boîtes d'onglets

1) Démarrer un nouveau projet.
- Nommer le projet "*boiteonglet* "
- Nommer la fenêtre principale "*fenetreprinc* "
- Modifier le titre de cette fenêtre en "**Gestion d'une boîte d'onglets** "

2) Installer le contrôle *tabstrip* dans la boîte à outils de Visual Basic *(Bouton de droite sur la boîte à outil et menu composant)*. Le composant à installer est : **Microsoft Windows Common Controls**

3) Parmi les nouveaux composants se trouve le contrôle *tabstrip*. Placer un de ces contrôles dans la zone client de la fenêtre principale.





4) Afficher la fenêtre de propriétés personnalisée du contrôle *(Bouton de droite et propriétés)*

5) Cocher le choix *MultiRow* du menu général et dans le menu *Onglets* insérer **10** onglets en cliquant **9** fois sur "*insérer un onglet* ". *(voir ce que vous devriez obtenir sur la deuxième figure ci-dessus)*

6) Exécuter le programme et cliquer sur les onglets du *tabstrip*. Rien de spectaculaire ne se passe ! C'est normal car c'est bien sur à vous de remplir les zones clients de chaque onglet en faisant apparaître les différents contrôles.
Pour cela pour chaque onglet, il faut afficher dans la zone client du *tabstrip* les différents contrôles qui l'habillent. Pour des raisons de simplicité dans la gestion de ces contrôles, il est commode d'afficher un contrôle *picturebox* pour chaque onglet qui contiendra les objets sur lesquelles l'utilisateur agira *(case à cocher, bouton, boîte d'édition, curseurs, …)*.
Voici la démarche à suivre :

- Placer un contrôle picturebox dans la fenêtre principale

- Nommer ce contrôle *PictTabStrip*

- Donner une valeur **0** à la propriété *index*. Ceci permettra de disposer d'un ensemble de contrôles de même nom *(PictTabStrip)* que l'on différenciera par leur index. *(PictTabStrip(0) , PictTabStrip(1), … PictTabStrip(9))*

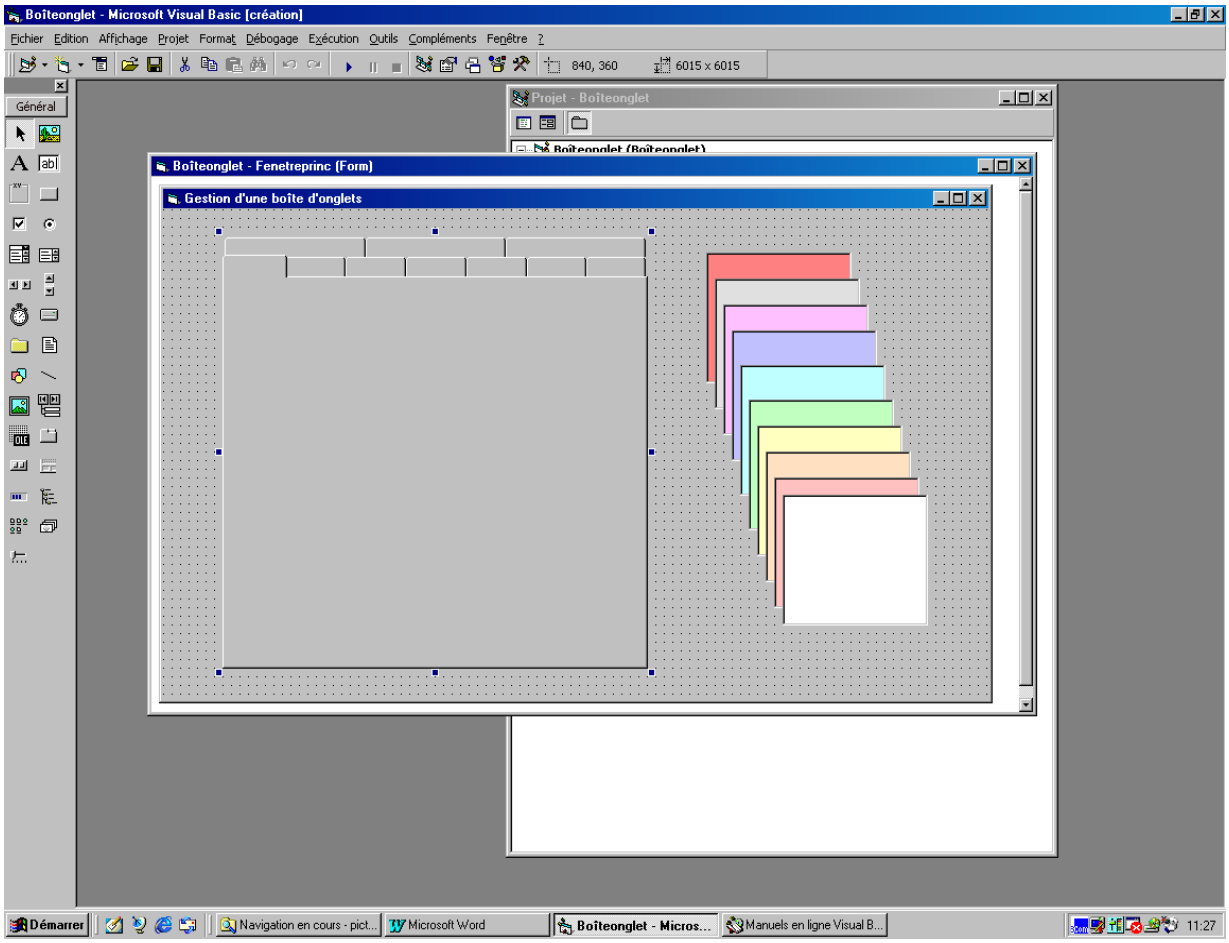
- Copier-coller *(Ctrl Inser/Maj Inser)* le contrôle *PictureBox* originel **9** fois pour obtenir au total **10** *PictureBox* à l'écran.

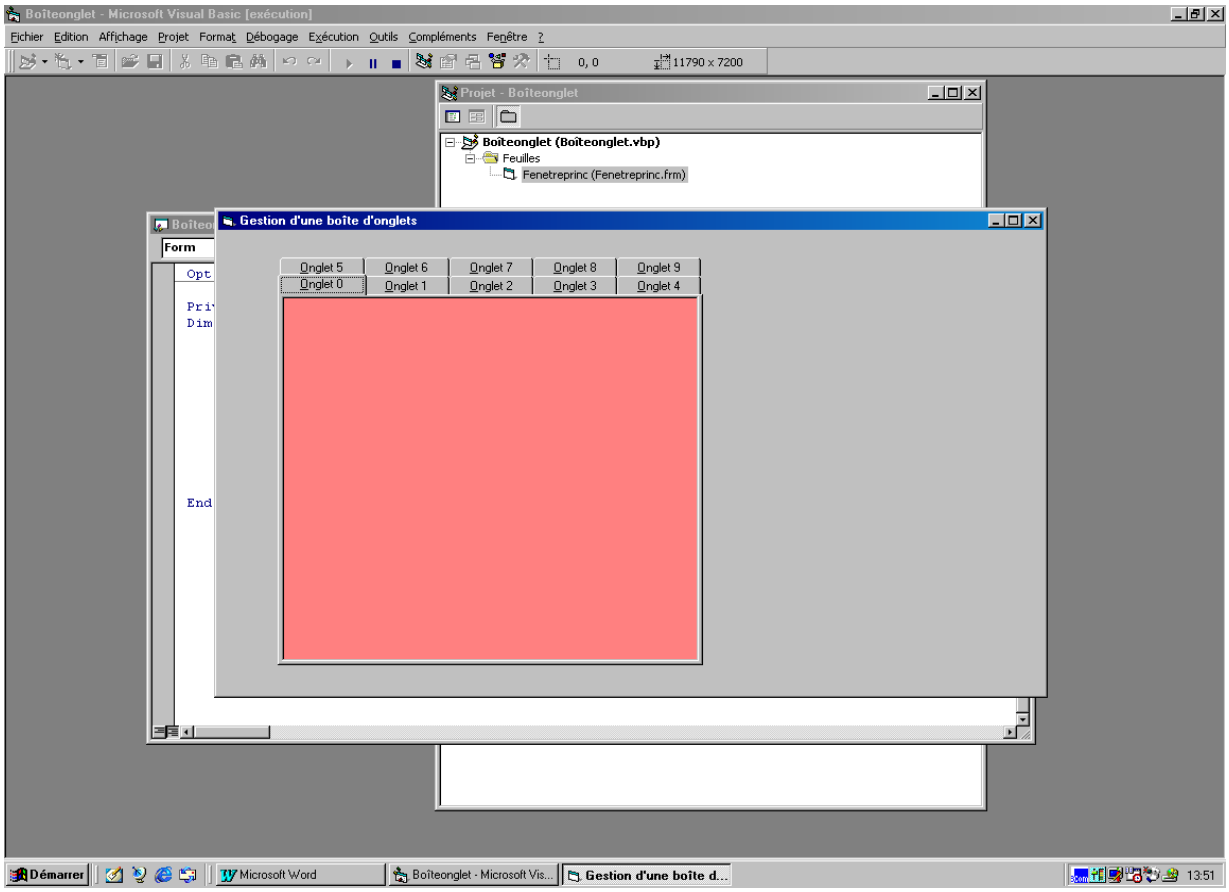
  !!! : Avant de faire un coller, s'assurer que la fenêtre principale est sélectionnée. Ainsi les *PictureBox*

seront des objets de la fenêtre principale. Si un autre objet est sélectionné *(un autre PictureBox par exemple)*, l'objet collé appartiendra à l'objet sélectionné et son affichage sera donc limité à la zone client de celui-ci.

7) Donner une couleur de fond différente a chaque contrôle *PictureBox*.

- Normalement la propriété index de chaque *PictureBox* a été automatiquement incrémenté et la première figure de la page suivante représente ce que vous devriez obtenir à l'écran.





8) Nous allons maintenant configurer en dynamique dans le programme la position et la taille de ces contrôles *PictureBox*

- Pour cela utilisons l'événement *Form_Load* de la fenêtre principale que l'on peut programmer en double cliquant sur celle-ci et écrivons les lignes de codes suivantes :

```
Private Sub Form_Load()
   Dim i As Integer
   ' Positionnement des différentes fenetres du tableau d'onglet
       For i = 0 To PictTabStrip.Count – 1
         TabStrip1.Tabs(i + 1).Caption = "&Onglet " & i
         With PictTabStrip(i)
                  .visible = False
                  .Move TabStrip1.ClientLeft, TabStrip1.ClientTop, TabStrip1.ClientWidth, TabStrip1.ClientHeight
         End With
       Next i
    PicttabStrip(0).ZOrder 0
    PictTabStrip(0).visible = True
End Sub
```

- Lancer le programme ! Normalement, la zone client du premier Onglet est recouverte par le contrôle *PictureBox* d'indice 0 *(Verifiez au moyen de la couleur)*

- Essayer de comprendre le programme en utilisant l'aide ***(F1)*** ou l'enseignant ***(He… M'sieur j'comprend pas ! ! !)***

9) Réagissons maintenant au clique d'un onglet en programmant la procédure d'événement *TabStrip1_Click().*

- Déclarer une variable globale dans le module de la fenêtre principale *(en haut de la page de code de la fenêtre principale)*

```
dim CurFrame as integer
```

- Taper le code suivant dans la procédure d'événement click du contrôle *TabStrip. (Double Cliquer sur le contrôle TabStrip)*

```
Private Sub TabStrip1_Click()
  Dim i As Integer
  ' Inutile de changer le cadre.
  ' Sinon, masque l'ancien cadre et affiche le nouveau.
  If TabStrip1.SelectedItem.Index = CurFrame Then Exit Sub
  For i = 1 To TabStrip1.Tabs.Count
    Picttabstrip(i - 1).Visible = False
  Next
  Picttabstrip(TabStrip1.SelectedItem.Index - 1).Visible = True
  ' Affecte la nouvelle valeur à CurFrame.
  CurFrame = TabStrip1.SelectedItem.Index
End Sub
```

- Lancer le programme et vérifier que chaque onglet fait apparaître un contrôle *PictureBox* différent dans la zone client du contrôle *TabStrip (Les couleurs doivent changer pour chaque onglet).*

- Comprendre le programme

10) Nous allons maintenant insérer des contrôles dans les différents *PictureBox* et vérifier qu'ils apparaissent bien avec le *PictureBox* conteneur quand l'onglet correspondant est sélectionné.

- Retrouver le *PictureBox* d'indice 3 et insérer à l'intérieur un contrôle *TextEdit* *(Pour cela sélectionner le PictureBox d'indice 3 et cliquer sur le TextBox de la boîte à outil, Dessiner le TextBox dans le PictureBox, celui-ci étant sélectionné).* Le Contrôle *TextBox* doit appartenir au contrôle *PictureBox* d'indice 3. Pour vérifier cette appartenance, déplacer le contrôle *TextBox* et vous devez remarquer que son déplacement est limité à la zone client du contrôle *PictureBox n°3.*
- Mettre la propriété *Text* du *TextBox* à Chaîne vide

- Insérer au début de la procédure *Form_Load()* la ligne suivante :
  - **Text1.Text = Me.Caption**

- Insérer à la fin de la procédure *Form_Load* les lignes suivantes :
  - **Text1.Left = 10 * Screen.TwipsPerPixelX**
  - **Text1.Width = Picttabstrip(3).Width - 20 * Screen.TwipsPerPixelX**

- Lancer le Programme et vérifier que le contrôle *TextBox* apparaît avec l'onglet correspondant avec une longueur légèrement inférieure à celle du *TabStrip*.

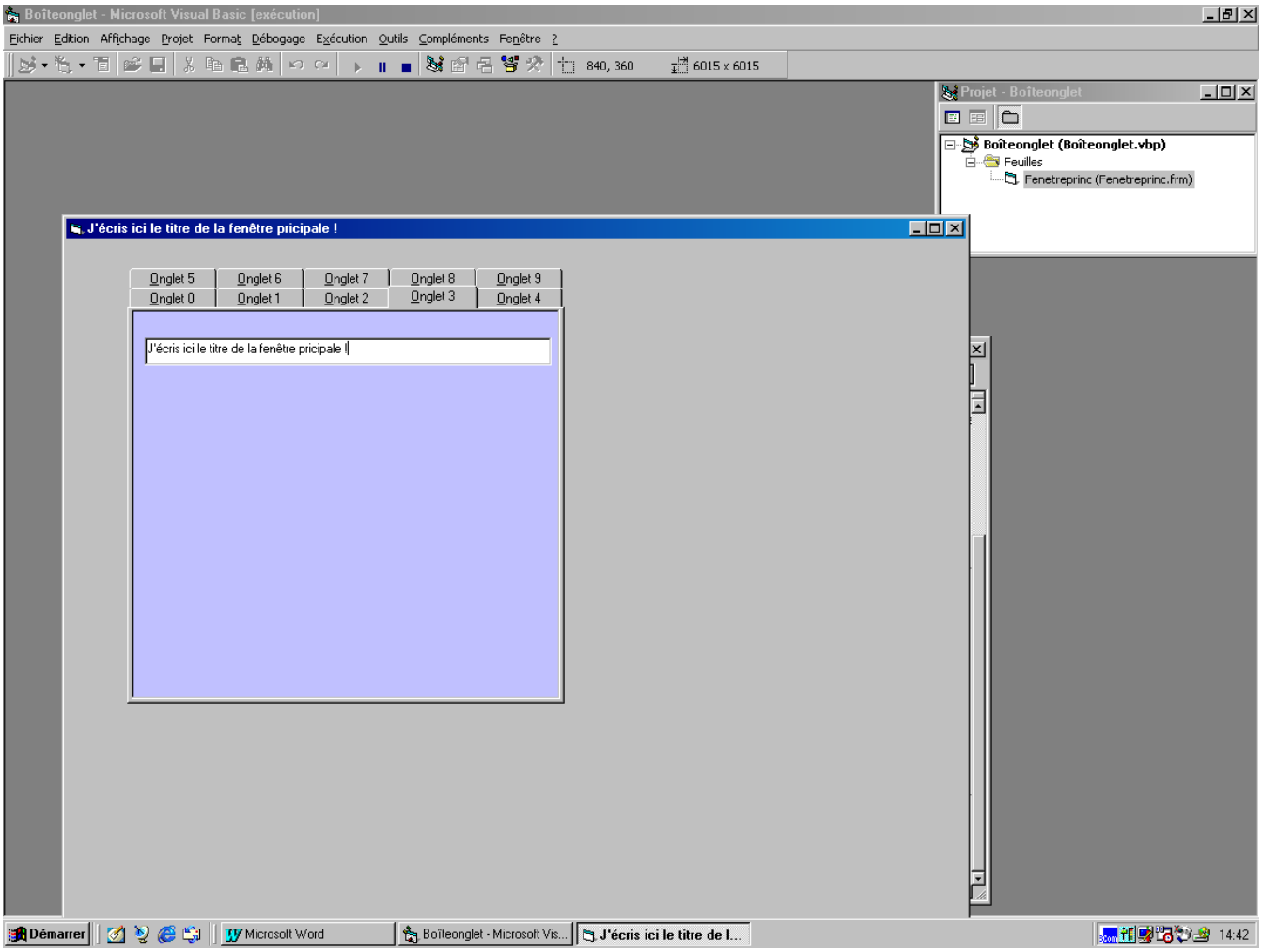
- Ecrire la procédure d'événement du *TextBox* suivante

```
Private Sub Text1_Change()
    Me.Caption = Text1.Text
End Sub
```

- Lancer le programme et modifier le contenu de la boîte d'édition de l'onglet 3.

- Insérer maintenant un bouton au niveau du *PictureBox* de l'onglet 0. Réagir à ce bouton en affichant la boîte de dialogue de la sélection des couleurs. Affecter la couleur choisie au fond de la fenêtre principale.

- Laisser votre imagination travailler pour mettre d'autres contrôles dans les autres *PictureBox*.

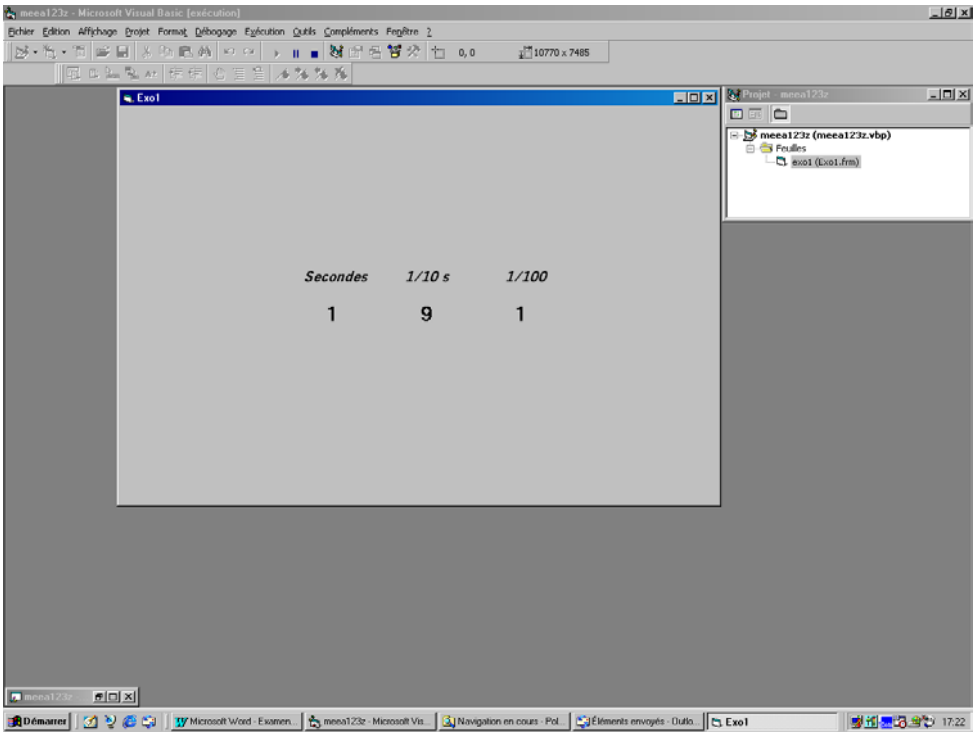


Exercice n°3 :

1) Le but de cet exercice est de créer un chronomètre en *seconde/dixième/centième*. Pour cela on utilise 3 contrôles *'label'* label3, label2, label1 affectant respectivement les secondes, les dixièmes de seconde et les centièmes de seconde. La base de temps s'effectue au moyen d'un contrôle ***timer*** réglé à **10ms**.
Ecrire un programme permettant d'afficher l'évolution de ce chronomètre à l'écran.

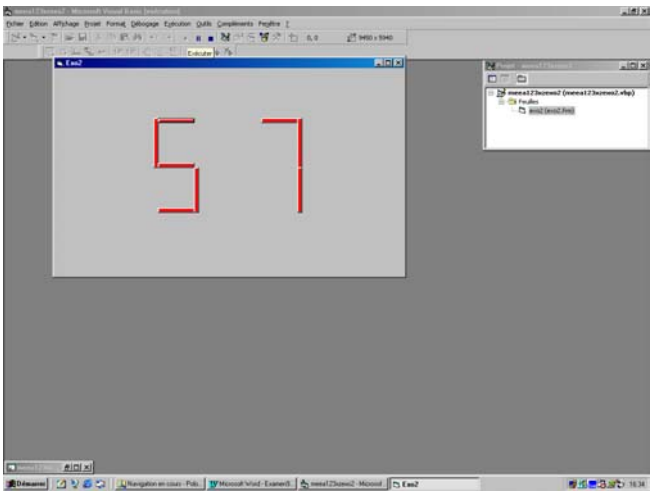
Conseils : Faire évoluer en testant en cascade les différents compteurs en partant bien entendu des centièmes de seconde.

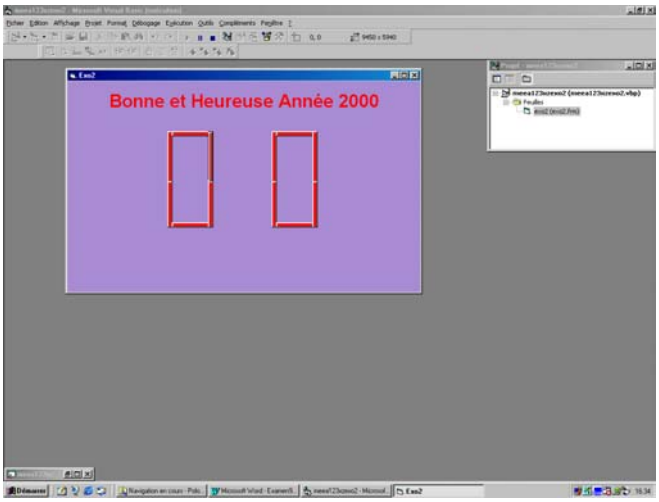
Ne pas s'inquiéter de la base de temps qui n'est pas respectée, c'est normal car Windows 98 ne peut pas générer pratiquement un **timer** de 1/10 de seconde *(En plus l'affichage consomme beaucoup de temps)*



Exercice n°4 :

1) On demande de réaliser un décompte de 99 à 00 qui symbolise les 100 dernières secondes avant la nouvelle année 2000. Quand le décompte arrive à 00, on rend visible un message **(contrôle label dans lequel on présente ses vœux à l'enseignant)**. Pour que l'effet soit plus percutant, on déclenche un feu d'artifice en faisant changer de manière cyclique et aléatoire la couleur de fond de la fenêtre principale.

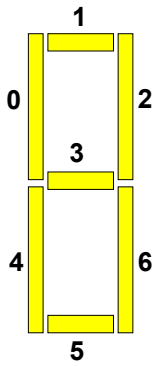




- Tout d'abord préparons le module d'affichage. On se propose d'afficher les nombres comme des afficheurs 7 segments dont les segments sont des contrôles boutons.
  Créer et placer sur la fenêtre principale 7 boutons de type ***COMMAND***. Pour cela créer un bouton que l'on appellera ***u*** comme unité. Ecrire le nombre ***0*** dans la propriété ***indice*** de ce bouton. Mettre la propriété ***style*** à ***graphique***, effacer le contenu de la propriété ***caption***, choisir une couleur de votre choix, mettre la propriété ***visible*** du bouton à ***false*** par défaut et modeler la forme du bouton pour qu'elle corresponde à la forme du bouton d'indice 0 *(voir la disposition des boutons sur la figure ci-dessous)*. Copier/coller ce bouton 6 fois. *(l'indice s'incrémente automatiquement)* et adapter à chaque fois la forme

du bouton au segment qu'il représente

Faire de même pour les dizaines en nommant 7 boutons *d(0), d(1),…..,d(6).*



- Ecrire une procédure ***afficheU*** dont le rôle sera de rendre visible ou pas chaque segment en fonction du nombre à afficher. Voici la structure que pourrait avoir cette procédure.

```
Sub afficheU(i As Integer)
   Select Case i
      Case 0:
         u(0).Visible = True
         u(1).Visible = True
         u(2).Visible = True
         u(3).Visible = False
         u(4).Visible = True
         u(5).Visible = True
         u(6).Visible = True
      Case 1:
            .
            .
      Case 9:
            .
   End Select
End Sub
```

Ceci est une solution simple mais facile, on peut faire autrement mais c'est plus compliqué, à vous de voir !

- Faire de même pour les dizaines en écrivant une procédure *afficheD* semblable à *afficheU*. Faire un copier/coller renommer la procédure remplacer *(menu édition/remplacer)* les ***u(*** par des ***d(*** .

- Placer un *timer (mettre la propriété **intervalle** à 10 pour accélérer le décomptage)* et un label dans la fenêtre principale contenant le message à afficher pour la nouvelle année.*(ce message est libre mais n'oublier pas qu'il s'adresse à l'enseignant et que c'est lui qui corrige !)*. La propriété ***visible*** du label doit être par défaut à ***invisible***

- Réagir à chaque <u>événement timer</u> en décomptant des variables globales de type ***integer*** *(unit et diz par exemple)* et en appelant les procédures ***afficheD*** et ***afficheU*** pour mettre à jour l'affichage.

- Quand le décomptage est à 00, rendre visible le label contenant le message et déclencher le feu d'artifice. Pour cela suivre les conseils suivants :

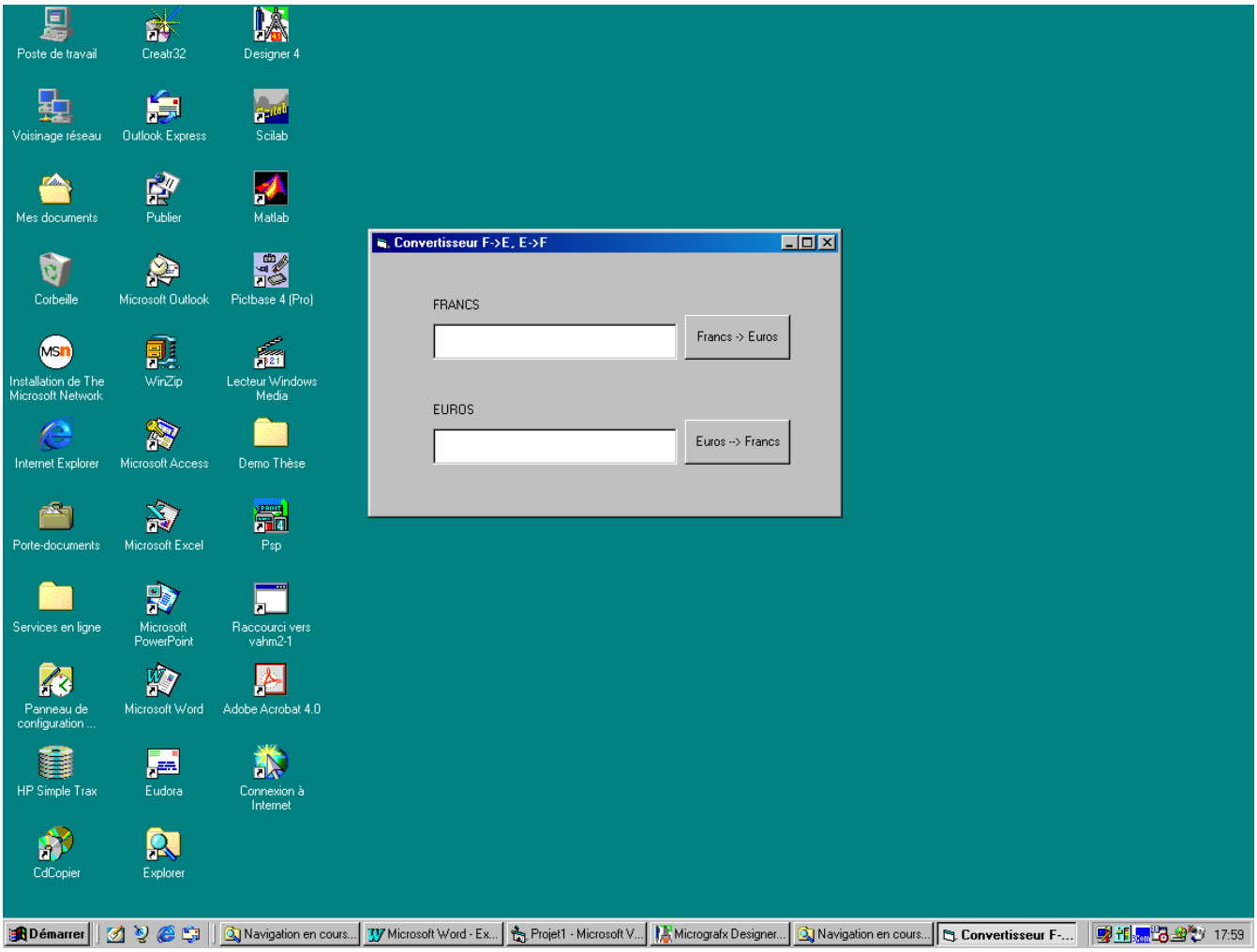
  - Pour n'utiliser qu'un seul *timer* pour le décomptage et le feu d'artifice, utiliser une variable booléenne globale précisant au niveau du *timer* si on décompte ou si on affiche le feu d'artifice. Cette variable sera initialisée à ***false*** dans la procédure ***Form_Load***.

- Pour changer la couleur du fond à chaque événement *timer*, on utilise les fonctions *RGB* et *Rnd (voir exemple ci-dessous et l'aide en ligne pour plus de précisions).*

```
Dim feuartifice as Boolean
Dim unit as Integer
Dim diz as integer

Private Sub Form_Load()
   unit = 9 ' initialisation de l'affichage
   diz = 9
   afficheU (unit)
   afficheD (diz)
   feuartifice = False
End Sub

Private Sub Timer1_Timer()
 If Not feuartifice Then
          ... à compléter
  Else
   Me.BackColor = RGB(255 * Rnd, 255 * Rnd, 255 * Rnd)
  End If
End Sub
```

Exercice n°5 :

1) Réaliser un convertisseur **Euro/Franc et Franc/Euro**. *(1 euro = 6.559 F)*



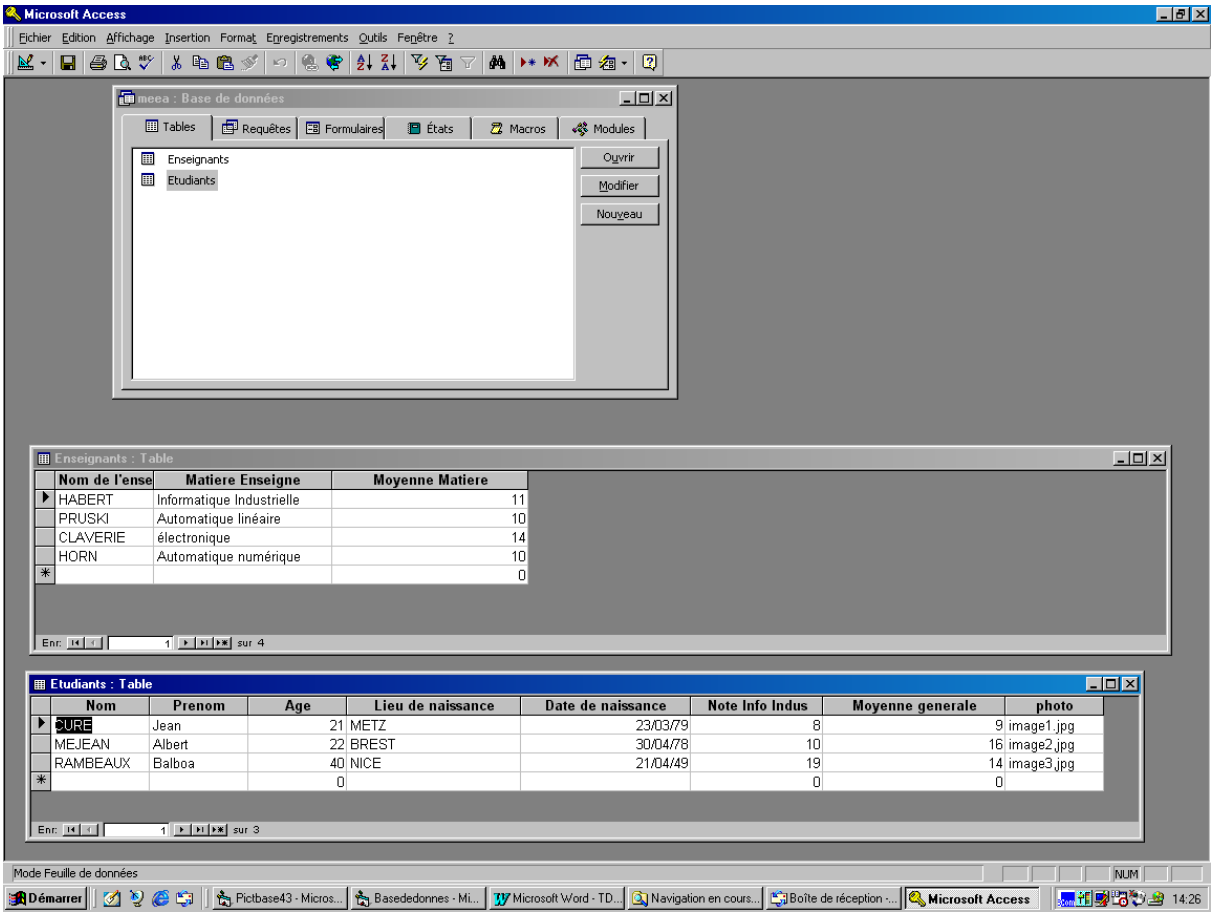
TD-TP Micro Informatique
Etude de Visual Basic

Accés aux bases de données sous Visual Basic

Sous Visual Basic, une base de données peut être vue comme un objet qui est structurée en ***tables*** elles-mêmes structurées en ***champs*** *(fields).* Par exemple dans la base de données que vous donnera l'enseignant, on dispose de deux tables nommées enseignants et étudiants.
La table étudiant est structurée en plusieurs champs nommés Nom, Prénom, Age,….. La table Enseignants est structurée également en plusieurs champ nommée Nom, Matière, ….
Donc chaque base de données est vue comme un objet **Database** qui contient un ensemble d'objets **Table** contenant elles-mêmes un ensemble d'objets **Champ**. Chaque ensemble de champs *(ligne)* est appelé un **Enregistrement**.
Chaque champ possède un type selon le format de l'information que l'on va stocker (texte, nombre, date, booléen, …).

1) Affichage de la structure d'une base de données

- Démarrer un nouveau projet.
- Nommer le projet "*Basededonnees* "
- Nommer la fenêtre principale "*AccesBD* ".
- Modifier le titre de cette fenêtre en **"Accés aux bases de Données"**.
- Inclure au projet la référence **Microsoft DAO …**
- Dessiner un contrôle *ListBox* dans la fenêtre principale.
- Mettre la base de données ***bd.mdb*** que vous a donné l'enseignant au niveau de la racine du disque **C**. Cette base a été créée sous ***Access*** et comprend deux tables dont la structure est donnée dans la copie d'écran ci-dessous



- Dans la procédure d'événement *Form_Load()* écrire le code suivant qui permet d'ouvrir la base de donnée et d'écrire le nom des tables non-système dans le contrôle *listbox*. La variable *db* de type *database* sera déclarée en tant que variable globale dans le code de la fenêtre principale. On prendra soin de fermer la base de données lorsque l'on quitte l'application. *(mettre db.close dans la procédure d'événement Unload)*

```
Dim db As Database  'Variable Globale
Private Sub Form_Load()
    Dim i As Integer
    List1.Clear
    Set db = OpenDatabase("c:\bd.mdb")
      For i = 0 To db.TableDefs.Count - 1
         If (db.TableDefs(i).Attributes And dbSystemObject) = 0 Then
            List1.AddItem db.TableDefs(i).Name
         End If
      Next i
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next
    db.Close
End Sub
```

- Comprendre le programme en s'appuyant sur l'aide en ligne.

2) Nous allons maintenant afficher la structure de chaque table dans un autre *ListBox*.

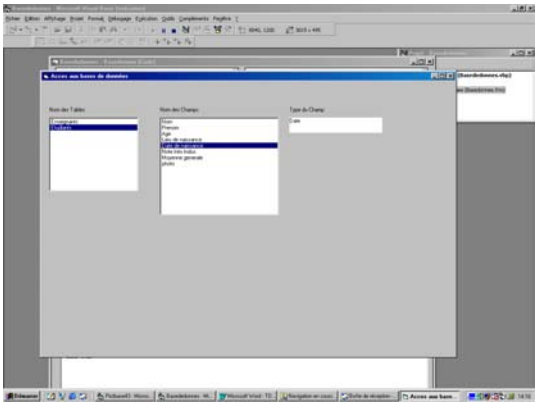
- La mise à jour de cette seconde *Listbox* s'effectuera lors d'un clique sur un nom de table de la première *ListBox*. Pour cela écrire le code suivant dans la procédure d'évènement Click du contrôle List1.

```
Private Sub List1_Click()
Dim i As Integer
    List2.Clear
    For i = 0 To db.TableDefs(List1.ListIndex).Fields.Count - 1
            List2.AddItem db.TableDefs(List1.ListIndex).Fields(i).Name
        Next i
End Sub
```

- Comprendre le programme !

3) Ajouter un contrôle label dans lequel vous écrirez de manière alphanumérique le type du champ sélectionné dans le contrôle *List2*.

- Pour cela on comparera la propriété ***type*** de chaque champ avec les nom de type prédéfinis *(dbtext, dbnumeric,dbdate,dbtime,… voir aide Visual basic)* en utilisant un ***Select Case*** comme dans l'exemple ci-dessus.



```
dbBigInt      Big Integer
dbBinary      Binary
dbBoolean     Boolean
dbByte        Byte
dbChar        Char
dbCurrency    Currency
dbDate        Date/Time
dbDecimal     Decimal
dbDouble      Double
dbFloat       Float
dbGUID        GUID
dbInteger     Integer
dbLong        Long
dbLongBinary                Long Binary
(OLE object)
dbMemo        Memo
dbNumeric     Numeric
dbSingle      Single
dbText        Text
dbTime        Time
dbTimeStamp                 Time Stamp
dbVarBinary   VarBinary
```

```
Select Case
db.TableDefs(List1.ListIndex).Fields(List2.ListIndex).T
ype
    Case dbText
      Label3.Caption = "Texte"
    Case dbDate
      Label3.Caption = "Date"
    Case dbTime
      Label3.Caption = "Heure"
          …. à compléter
  End Select
```

4) Nous allons maintenant voir comment lire et écrire dans un des champs d'une table.

- Tout d'abord la lecture : Pour cela il faut ouvrir un ensemble d'enregistrements. Dans un premier cas nous allons ouvrir tous les enregistrements de la table sélectionnée. La propriété de l'objet ***db*** permettant d'ouvrir un ensemble d'enregistrement est ***OpenRecordset***. L'argument principal est soit le nom d'une table si on veut ouvrir tous les enregistrements de la table soit une requête *SQL* que nous verrons plus tard. La variable permettant de mémoriser un ensemble d'enregistrements est le type ***RecordSet***
  Donc la syntaxe est la suivante :

```
dim rst as Recordset        ' à définir en variable Globale à coté de db

Set rst = db.OpenRecordset(List1.Text)      ' List1 contient le nom de la Table sélectionné dans le contrôle List1
```

- Quand l'ensemble d'enregistrement est ouvert, il faut maintenant parcourir cet ensemble *(par défaut, on pointe sur le premier enregistrement)* en utilisant les propriétés *MoveNext* et *MovePrevious* des objets *Fields*. Pour cela ajouter dans la fenêtre principale deux boutons dans lesquelles on placera en *caption* les symboles **<-** et **->**. *(voir figure ci-dessous)* . Définir chaque événement _click de chaque bouton de la manière suivante :

```
Private Sub Command1_Click()
On Error GoTo fin
   rst.MovePrevious
   Label5.Caption = rst.Fields(List2.ListIndex).Value
fin:
End Sub
```

```
Private Sub Command2_Click()
On Error GoTo fin
   rst.MoveNext
     Label5.Caption = rst.Fields(List2.ListIndex).Value
fin:
End Sub
```

A la suite de la fin du *Select Case (End Select)* de la procédure d'événement *List2_Click()* ajouter les lignes

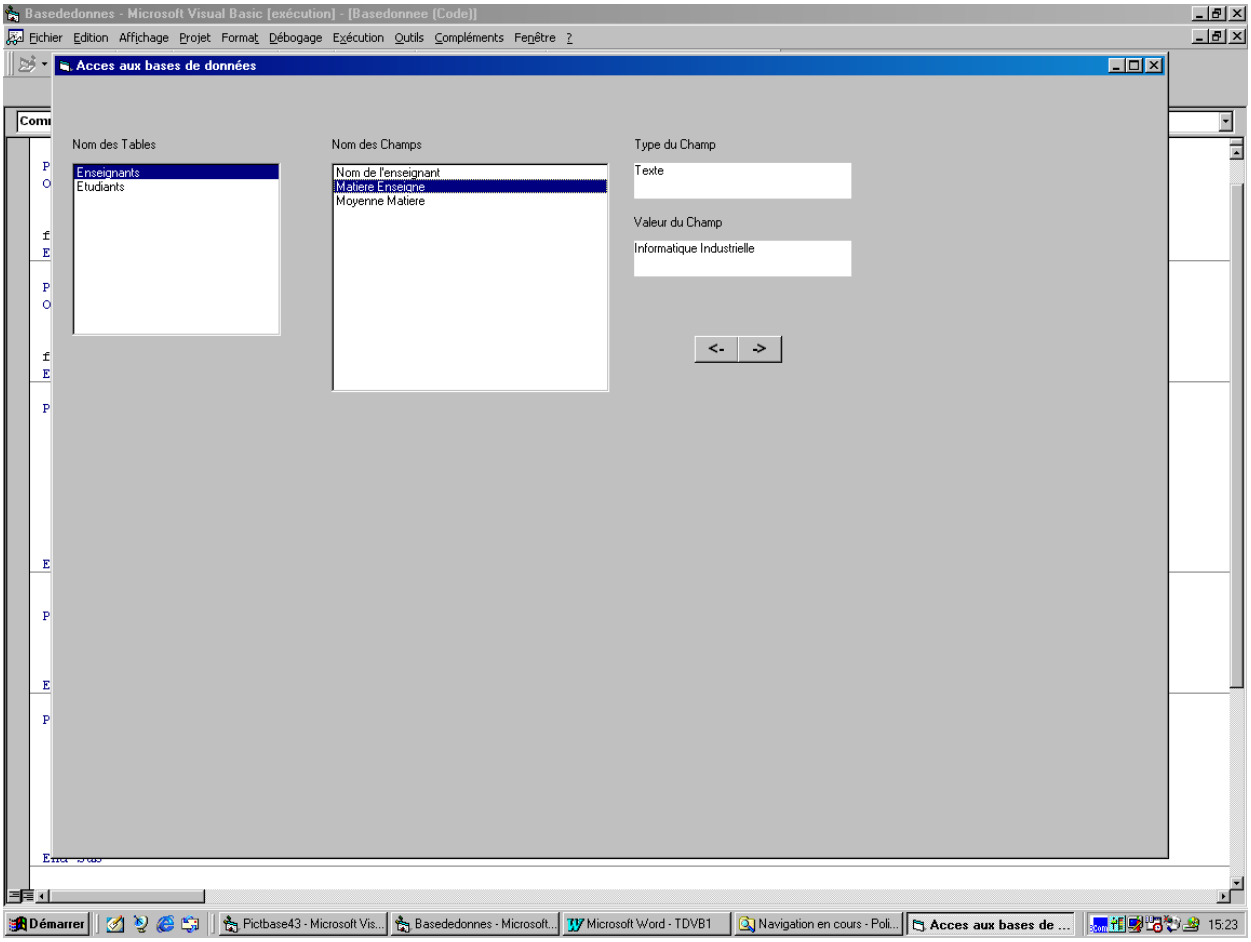
```
. .
End Select

Set rst = db.OpenRecordset(List1.Text)
Label5.Caption = rst.Fields(List2.ListIndex).Value
```

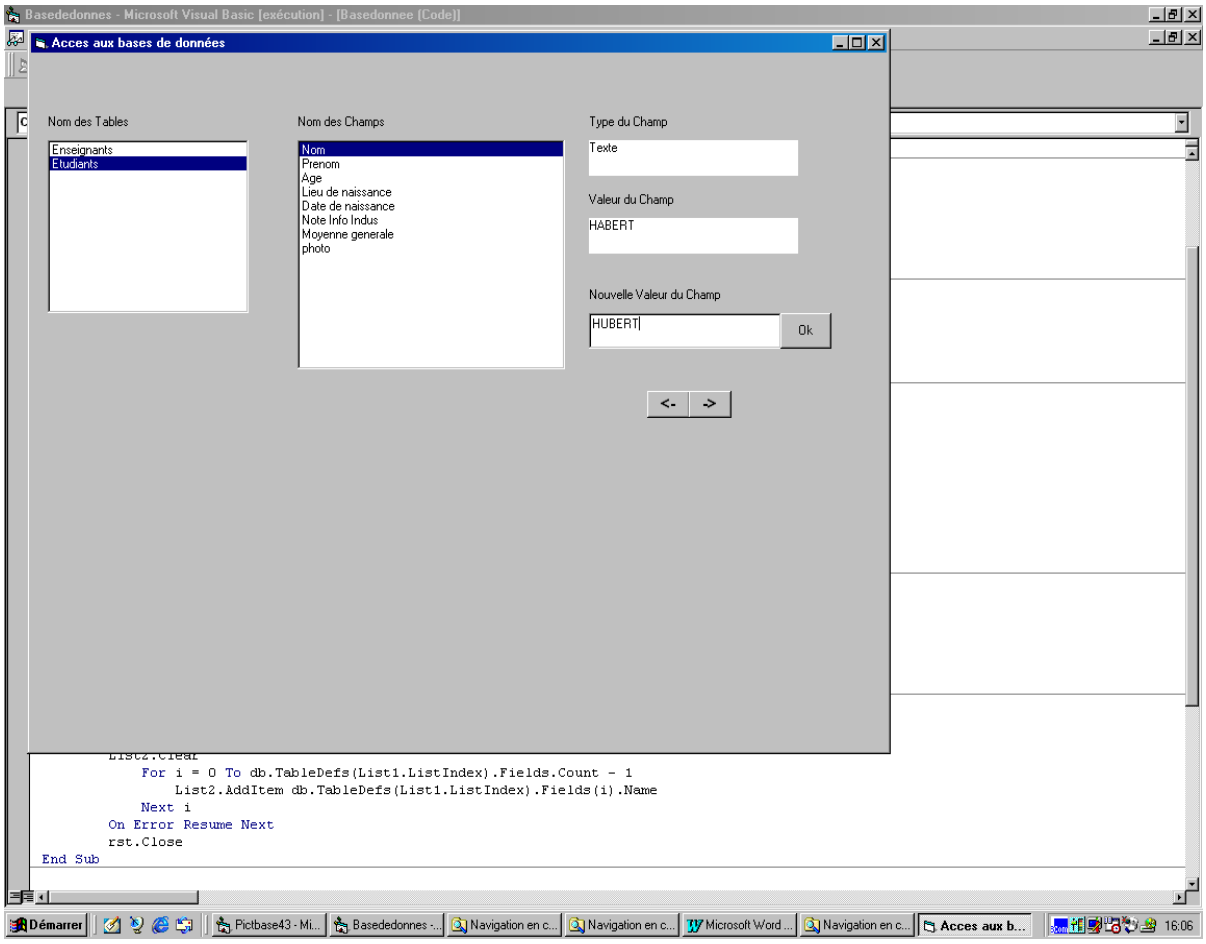
suivantes:

Lorsque l'on change de table *(procédure List1_Click())*, effacer l'ensemble d'enregistrements en ajoutant les lignes suivante:

```
On Error Resume Next
rst.Close
```

Les commandes *On Error …* permettent d'intercepter un erreur éventuelle et de ne pas provoquer l'arrêt brutal du programme. *(ceci nous évite par exemple de tester si **rst** est initialisé lorsque l'on ferme l'enregistrement ou d'éviter l'erreur lorsque l'on arrive à la fin de l'ensemble d'enregistrement. Pour les étudiants très intéressés demandé à l'enseignant une solution plus professionelle)*





- Ensuite l'écriture : Pour éditer un ensemble d'enregistrement, il faut utiliser la propriété ***Edit*** de l'objet ***Recordset***. On peut ensuite modifier les différentes valeurs des champs. Pour valider les modifications, il faut utiliser la propriété ***Update*** de l'objet ***Recordset***.
- Ajouter un contrôle TextBox et un Bouton Ok que vous placez à coté (voir figure ci-dessous). Ajouter le code suivant à la procédure de Click du nouveau Bouton :

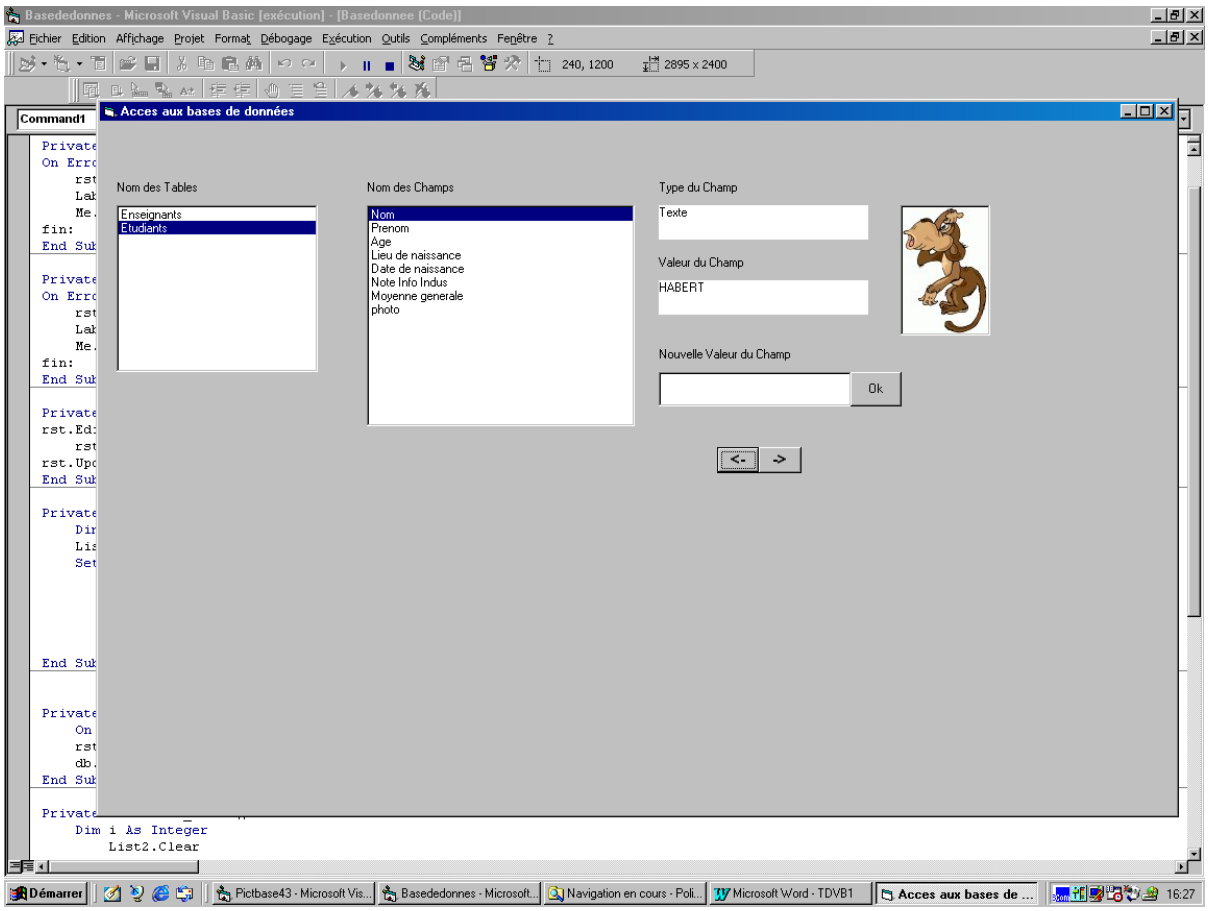
```
rst.Edit
   rst.Fields(List2.ListIndex).Value = Text1.Text
rst.Update
```

- Lancer le programme, positionner vous sur un champ, entrer des caractères dans le ***TexBox*** et valider par OK, passer au champ suivant et revenez sur le champ modifié. Normalement la nouvelle valeur a écrasé l'ancienne !

5) Ajouter un contrôle *PictureBox* et afficher dans ce contrôle la photo associée à chaque enregistrement de la table étudiant. *(image1.jpg, image2.jpg, ….. image11.jpg)*.

Demander les photos à l'enseignant ! Les photos seront mises dans le répertoire du projet Visual Basic et il est possible d'accéder au chemin par la propriété *path* de l'objet *App*.

Exemple : Picture1.Picture = LoadPicture(App.Path & "\" & rst.Fields("Photo").Value)


Basededonnes - Microsoft Visual Basic [exécution] - [Basedonnee (Code)]
Fichier Edition Affichage Projet Format Débogage Exécution Outils Compléments Fenêtre ?
Command1
Acces aux bases de données
Nom des Tables
Enseignants
Etudiants
Nom des Champs
Nom
Prenom
Age
Lieu de naissance
Date de naissance
Note Info Indus
Moyenne generale
photo
Type du Champ
Texte
Valeur du Champ
HABERT
Nouvelle Valeur du Champ
Ok
<-
->
Dim i As Integer
List2.Clear
Démarrer
Pictbase43 - Microsoft Vis...
Basededonnes - Microsoft...
Navigation en cours - Poli...
Microsoft Word - TDVB1
Acces aux bases de ...
16:27

TP-TD Micro Informatique

Création Dynamique de Tables, de Champs et d'Enregistrements

1) **Nous allons maintenant voir comment *créer* et *effacer* un enregistrement dans la table Etudiant de la base bd.mdb.**

Pour créer un enregistrement il faut utiliser les propriétés *Addnew* et *Update* d'un l'objet *Recordset*. Pour effacer un enregistrement il faut utiliser la propriété *Delete* d'un objet *Recordset*.

- Créer un bouton que vous nommerez ***NewEnr*** et ajouter le code suivant dans la procédure d'événement *Click* :

```
Private Sub NewEnr_Click()
   On Error GoTo fin
   rst.MoveLast
   rst.AddNew
   rst.Fields("Age").Value = 77
   rst.Fields("NOM").Value = "Truc"
   rst.Fields("Prenom").Value = "Machin"
   rst.Update
fin:
End Sub
```

- Exécuter le programme et appuyer sur le bouton *(dés que l'objet **rst** est initialisé, c'est à dire quand vous avez cliqué sur un nom de champs de la List2).*

- Vérifier qu'un nouvel étudiant de ***77 ans*** s'appelant ***Truc Machin*** existe maintenant dans la base.

- Créer un bouton que vous nommerez ***EffEnr*** et ajouter le code suivant dans la procédure d'événement *Click* :

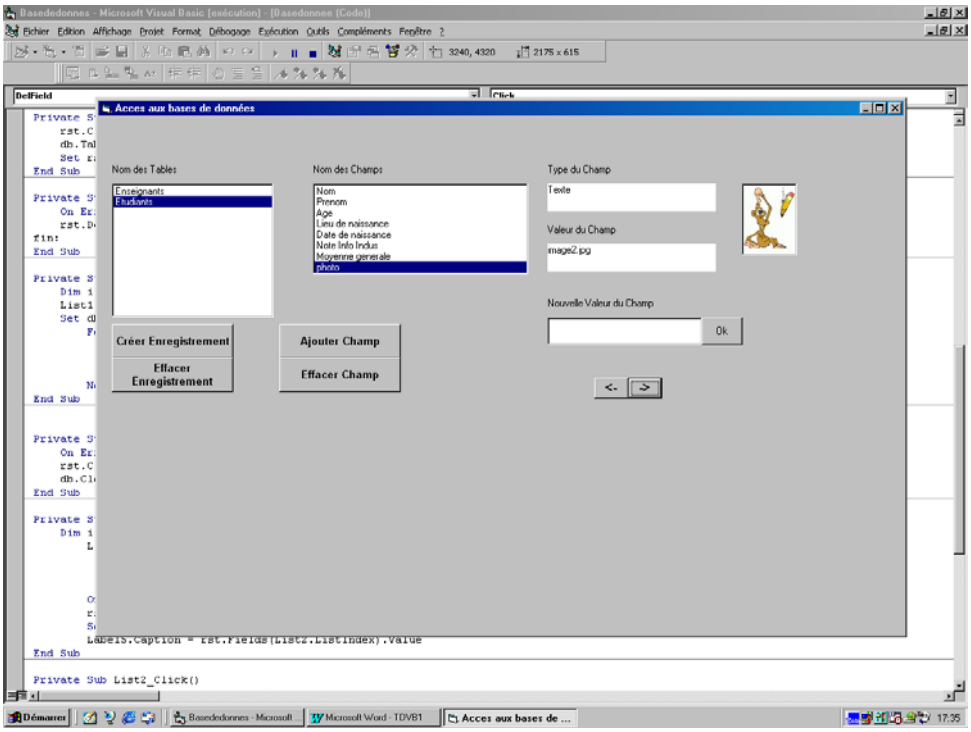
```
Private Sub EffEnr_Click()
   On Error GoTo fin
   rst.Delete
fin:
End Sub
```

- Se placer sur un enregistrement et appuyez sur ce bouton *(celui créé précédemment par exemple).*

- Vérifier que l'enregistrement en question a bien été effacé de la base.

2) **Nous allons maintenant voir comment créer et effacer un champ dans une table.**

- Insérer deux boutons que vous nommerez respectivement ***AddField*** et ***DelField***. L'insertion d'un champ s'effectue par la méthode ***Append*** de la collection de champs ***Fields*** de la table en cours.



- Entrer le code suivant pour le premier bouton :

```
Private Sub AddField_Click()
   Dim t As TableDef
   Dim f As Field

   rst.Close

   Set t = db.TableDefs(List1.Text)

   Set f = New Field
   f.Name = "Adresse"
   f.Type = dbText

   t.Fields.Append f
   Set t = Nothing
   Set f = Nothing

   Set rst = db.OpenRecordset("Select * from " & List1.Text)

End Sub
```

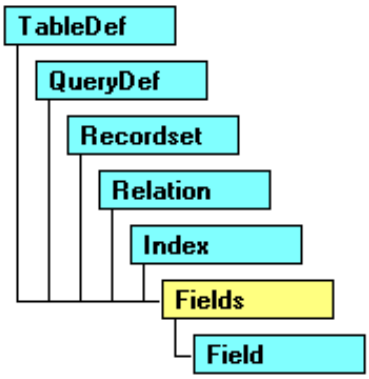
Explications:
Pour ajouter un champ, aucun ensemble d'enregistrements ne doit être en cours de sélection *(C'est logique car on touche maintenant à la structure interne de la table)*. Pour cela on ferme le **Recordset rst** avant de travailler. On l'ouvrira de nouveau à la fin de la procédure pour remettre le programme dans un état de fonctionnement normal. Ensuite on définit une variable **t** de type **TableDef** qui pointe vers la table en cours de la base. Un nouveau champ est crée de la même façon avec les instructions ***New*** et ***Set***. On affecte quelques propriétés de ce nouveau champ et on l'ajoute à la table en utilisant la fonction ***Append***.

- Vérifier que le nouveau champ a été créé après l'appui sur le bouton **AddField**.

- Ajouter le code suivant pour le second bouton :

```
Private Sub DelField_Click()
   rst.Close
   db.TableDefs(List1.Text).Fields.Delete List2.Text
   Set rst = db.OpenRecordset("Select * from " & List1.Text)
End Sub
```

- Vérifier que le champ en cours de sélection est supprimé lors de l'appui sur le bouton **DelField**.
- Essayer de comprendre les deux programmes précédents à l'aide de l'aide en ligne. Insister sur l'organisation orientée objet suivante d'une base de données.



3) Créons maintenant une nouvelle table ***joueurs*** possédant deux champs ***nom*** et ***score*** : Pour cela, pas besoin de créer de bouton, l'action va être programmée lors du chargement de la fenêtre principale *(dans l'événement Form_Load)*

- Recopier le code suivant dans l'événement *Form_Load* : Le code qui a changé par rapport à la procédure originelle est en italique.

```
Private Sub Form_Load()
   Dim i As Integer
   Dim TBL As TableDef
   Dim fld As Field

   Set db = OpenDatabase("c:\bd.mdb")
    On Error Resume Next
     ' on efface la table si elle existe
     db.TableDefs.Delete "joueurs"
     'on crée dynamiquement une nouvelle table
     Set TBL = New TableDef
     TBL.Name = "joueurs"
     'On ajoute deux champs à cette table
     Set fld = TBL.CreateField("name", dbText)
     TBL.Fields.Append fld
     Set fls = Nothing
     Set fld = TBL.CreateField("score", dbText)
     TBL.Fields.Append fld
     'On Ajoute la nouvelle table à la base de données
     db.TableDefs.Append TBL
     Set fld = Nothing
     Set TBL = Nothing
     db.TableDefs.Refresh

     List1.Clear
      For i = 0 To db.TableDefs.Count - 1
         If (db.TableDefs(i).Attributes And dbSystemObject) = 0 Then
            List1.AddItem db.TableDefs(i).Name
         End If
      Next i
End Sub
```

Normalement une table ***joueur*** a été créée avec deux champs ***name*** et ***score*** . Vérifier la bonne exécution du programme.

4) Définir un bouton qui crée un enregistrement dans la table sélectionnée dan le contrôle ***list1***. Vous pourrez ainsi créer des enregistrements dans la table joueurs et modifier les champs grâce au bouton **OK**.

5) En utilisant un bouton placé à coté de la boîte d'édition des nouvelles valeurs *(à coté du bouton OK) (ce bouton ne doit apparaître que lorsque le champ sélectionné est de type date)*, afficher un contrôle **calendar** *(insérer pour cela le composant **mscal.ocx nommé microsoft calendar control**)* dans une boîte de dialogue, choisir une date et l'insérer dans le champ **date** en cours de sélection.

6) Charger le composant **DBGRID.ocx** et essayer de lier la base de données bd.mdb à ce contrôle. Utiliser l'aide en ligne pour comprendre le fonctionnement de ce contrôle en lien avec une base de données.

Introduction aux Requêtes SQL (Structured Query Language)

Le langage **SQL** permet d'effectuer des recherches en utilisant des critères sur les champs et tables de la base de données. Les bases de données professionnelles sont en générales énormes *(de nombreuses tables liées entres-elles mémorisant plusieurs centaines de milliers, voir millions d'enregistrements)*. Lorsque l'on veut consulter une table pour écrire ou lire des données, il est impensable de charger toute la table en mémoire dans un objet *Recordset (d'autant plus que la table peut être une table réseaux distante)*. On utilise alors un langage spécifique que l'on envoi sous forme de *Scripts* à la base et qui permet de sélectionner certains ensembles d'enregistrement ou de champs dans une ou plusieurs tables. Le langage nommé SQL *(Structured Query Language)* permet notamment d'exprimer ces critères de sélection en associant des expressions logiques à des mots clés propres au langage.

La commande **Set rst = db.OpenRecordset("Etudiants")** met tous les enregistrements de la table Etudiants dans un objet ensemble d'enregistrements.
En SQL ceci est équivalent :

```
Set rst = db.OpenRecordset("Select * from Etudiants")
```

Voici d'autres exemples de requêtes que vous devez essayer lors de l'ouverture de la base :

```
Select * from Etudiants Where Age < 18
Select * from Etudiants WHERE (Age < 18) and Nom <> 'DUPONT'
```

Analyser leur signification et observer ce qui se passe dans le programme *(filtrage des enregistrements)*
Les sites suivants fournissent un didacticiel *(en anglais pour le premier et en français pour le second)* du langage SQL:

http://www.oragec.org/fr/trucs/sql_introduction.htm
http://wwwlsi.supelec.fr/www/yb/poly_bd/sql/tdm_sql.html

Consulter ce site et essayer les autres mots clés comme ***LIKE, DISTINCT, ORDER BY…***
Ecrire au moins 20 requêtes SQL utilisant les mot clés SELECT, WHERE, LIKE, DISTINCT, OR, AND, NOT, WHERE, MIN, COUNT, SUM

Programmation Active Server Page (ASP)

Olivier HABERT
Janvier 2001

Un didacticiel complet est disponible à l'adresse :
http://www.activeserverpages.com/learn/

HTML

Pour tout ce qui concerne l'apprentissage des rudiments de l'HTML, cherchez des didactitiels sur internet. Suivez le déroulement de ces didacticiels en générant des scripts HTML avec un éditeur comme bloc-notes et en les testant avec Netscape ou Internet explorer.
Pour la suite du TP, il faut au moins avoir assimilé les notions de base suivantes :

- Les différentes balises pour l'affichage
- Les différents niveaux d'affichage
- Les listes
- Et surtout la génération de tableaux

ASP

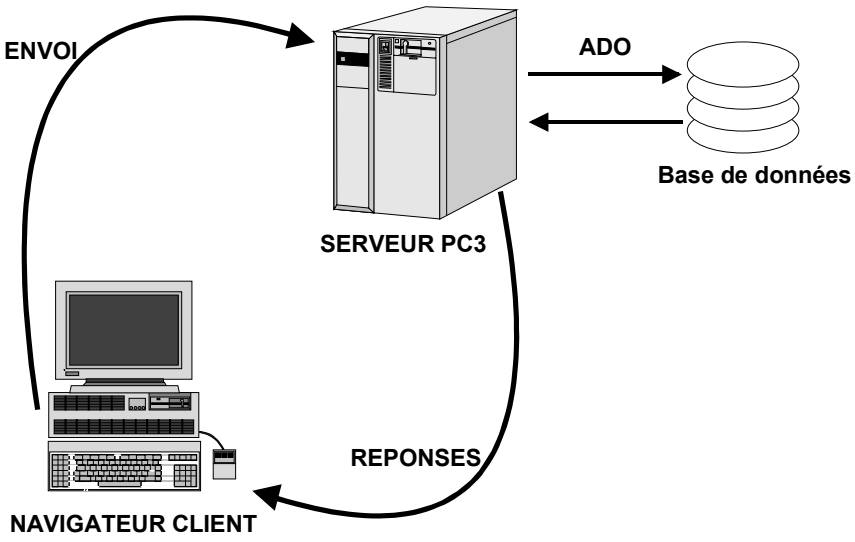
Les Actives Server Pages *(ASP)* sont un standard Microsoft permettant de générer dynamiquement des pages HTML. Une page ASP *(reconnaissable par l'extension .asp)* se caractérise par la présence des balises <% , qui indiquent au serveur HTTP qu'une commande VBScript ou JScript doit être interprétée. Voilà bien une caractéristique importante à saisir de ce langage de script puissant *(langage interprété)* exécuté du côté serveur et non du côté client *(les scripts en JavaScript ou les applets Java s'exécutent dans le navigateur d'un site)*.

L'ASP est donc un langage "interprété" : c'est-à-dire qu'il s'intègre dans une page HTML puis interprète le code source compris à l'aide de ces balises spéciales pour retranscrire le résultat au navigateur client.

Au contraire, un langage comme JAVA *(.exe, .class)* écrit des applets compilées sous une forme qui leur permet d'être exécutées ensuite comme des programmes, puis placées sur un serveur web.

L'ASP exprime toute sa puissance quand il s'agit de piocher dans les bases de données. En effet, ce langage sert d'intermédiaire entre le navigateur client et une base de données grâce à la technologie Active Data Object *(ADO)* qui fournit les élément nécessaires à la connexion au système de gestion de bases de données.

Schéma pour comprendre: le cheminement des informations via l'ASP



Qu'est ce qu'un fichier ASP ?

C'est un fichier d'extension .ASP qui permet de combiner du script html classique avec du script dérivé de langages évolués *(basic, java,…)* et permettant également l'utilisation de composants *(objets)* écris au moyen de n'importe quel langage.
Ces scripts sont exécutés coté serveur et permettent au client de recevoir un résultat sous forme de code html dynamiquement crée par ce serveur.
De cette façon, toute page ASP génère dynamiquement des pages html en fonction des paramètres envoyés par le client, des résultats coté serveurs *(calculs, recherche dans une base de données,…)*.

- Tout client pouvant lire le langage **html** peut se connecter à un serveur ASP

Organisation matérielle et logicielle d'une application ASP

- 1 serveur capable d'exécuter les scripts ASP *(PWS sous WIN98, Internet Information Server (IIS) sous Windows NT/2000).*
- 1 client capable de lire du script html.

Un serveur de pages actives comprend 5 objets standard pour l'utilisation global

- Un objet ***Request*** → pour obtenir de l'information à partir du client
- Un objet ***Response*** → pour envoyer de l'information vers le client
- Un objet ***Server*** → pour contrôler IIS
- Un objet ***Session*** → pour stocker l'information concernant les changements de paramétrage de la session utilisateur courante
- Un objet ***Application*** →pour partager les informations de niveau Application et les paramètres durant la durée de vie de l'application

Nous découvrirons ces objets au fur et à mesure de l'évolution du TP.
Tout d'abord, écrivons notre premier script pour " *sentir* " le but du TP !

Mon Premier SCRIPT

Suivre la démarche suivante

a) Prendre l'éditeur texte de votre choix *(bloc-notes par exemple)*
b) Recopier le script suivant :

```
<html>
<head>
<title> ASP Mon premier Script </title>
</head>
<body>
<% dim i
for i = 1 to 4 %>
     <H1> niveau <%=i%></H1>
<% next %>
</body>
```

c) Sauver le fichier sur le disque en l'appelant **firstasp.asp**
d) Charger le navigateur **netscape** ou **internet explorer**
e) Se connecter au serveur en utilisant l'adresse ftp://pc3/groupex *(x représente le numéro que l'enseignant vous a affecté)*
f) Prendre l'explorateur de Windows et glisser le fichier **firstasp.asp** dans la fenêtre du navigateur
g) Au moyen du navigateur, tester le fichier ASP en utilisant la connexion:

http://pc3/asp/groupex/firstasp.asp

Explication du programme ASP:

```
1   <html>
2   <head>
3   <title> ASP Mon premier Script </title>
4   </head>
5   <body>
6   <% dim i
7    for i = 1 to 4 %>
8    <H1> niveau <%=i%> </H1>
9    <% next %>
10  </body>
```

a) Les lignes 1 à 4 permettent de définir en **html** le texte qui apparaît dans le bandeau système de la fenêtre du navigateur
b) La ligne 5 débute grâce à la balise **<body>** le début du script **html**
c) La balise **<%** de la ligne 6 signifie que un script **VBSCRIPT** ou **JAVASCRIPT** va débuter
d) L'instruction **DIM i** déclare une variable i. En **VBSCRIPT**, les variables n'ont pas de type explicite tant qu'elles n'ont pas été utilisées *(type VARIANT)*
e) La ligne 7 débute une boucle en utilisant i. Cette ligne se termine par %> car le texte qui suit n'est plus du VBSCRIPT
f) La ligne 8 permet de générer 4 fois une ligne **html** en faisant varier un paramètre. Bien analyser cette ligne pour comprendre cette première approche du **VBSCRIPT**.
g) La ligne 9 se remet en **VBSCRIPT** pour compléter l'instruction de la ligne 7. **%>** termine cette ligne pour quitter le mode **VBSCRIPT**.
h) La ligne 10 termine le script **html**

En résume, le serveur a interprété ce script en exécutant 3 fois la ligne 7. Le client a reçu le code **html** suivant : *(vous pouvez le vérifier en affichant le code source au niveau du navigateur)*

```
<html>
<head>
<title> ASP Mon premier Script </title>
</head>
<body>
   <H1> niveau 1 </H1>
   <H1> niveau 2 </H1>
   <H1> niveau 3 </H1>
</body>
```

Maintenant il faut réfléchir …

Modifier le script précédent de manière à élever à chaque fois le niveau du texte (balises <H1>, <H2>, …..

Etude de l'objet Response

L'objet Response est utilisé pour envoyer de l'information vers le client.

a) La méthode Write **de cet objet permet d'écrire une chaîne vers le client.**

Exemple: le programme suivant est équivalent au programme étudié précédemment

```
<html>
<head>
<title> ASP Mon premier Script </title>
</head>
<body>
<% dim i
 for i = 1 to 4 %>
    <H1>niveau <%response.write(i)%> </H1>
<% next %>
</body>
```

b) La méthode **Redirect** force le navigateur a essayer de se connecter vers une adresse web

Exemple: Ajouter avant la balise </body> la ligne suivante:

<% Response.Redirect "http://www.microsoft.com" %>

c) La méthode **End** termine le script **ASP**

Exemple: Ajouter avant la ligne 9 la ligne suivante: **<%if i=2 then response.End%>**

D'autres méthodes existent, voir leur description sur le site suivant:

http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/iisref/vbob5sj8.htm

Etude de l'objet Request

Cet objet est utilisé pour obtenir des information à partir du client. Ces informations peuvent provenir de formulaires édités par l'utilisateur.
Dans le programme suivant, surligner le code en VBSCRIPT *(<% →début du VBSCRIPT, %> →fin du VBSCRIPT)*

```
<html><head>
<title>Réagir aux données d'un formulaire </title>
</head><body bgcolor="#808080">

<%
SELECT CASE lcase(request.form("proposer"))

        CASE "soumettre"
                %>
                <font face="arial" size = 20>
                <%
                response.write "Bienvenue en DESS "
                response.write request.form("namefirst") & " "
                response.write request.form("namelast")
                %>
                </font>
                <%
        CASE ELSE
        %>
          <Form method="post"><b>
          Pour vous inscrire en DESS, <br>entrer votre NOM et PRENOM:<p>
          NOM        <Input NAME="NameLast" size
="30"><br>
          PRENOM <Input NAME="NameFirst" size ="30"><br><br><br>
          <input type="submit" name="Proposer" value="Soumettre">
        </b>
        </form>
  <%END SELECT%>
        </body>
        </html>
```

a) Recopier le code et sauvez le fichier sous le nom *essaiform.asp*.

b) Se connecter au serveur en utilisant l'adresse ftp://pc3/groupex *(x représente le numéro que l'enseignant vous a affecté)*

c) Prendre l'explorateur de Windows et glisser le fichier *essaiform.asp* dans la fenêtre du navigateur

d) Au moyen du navigateur, tester le fichier ASP en utilisant la connexion:
http://pc3/groupex/essaiform.asp

e) Essayer le script, le comprendre en le modifiant prudemment.

f) Modifier le script en testant le nom et le prénom pour refuser d'inscrire l'étudiant Dupont Alain *(Faire apparaître le message "Suite à une première expérience désastreuse, les Dupont Alain ne sont plus acceptées en DESS ISI")*

Exercice

A l'aide des fonctions comme FormatDateTime(), Date() et Now()

(voir http://www.activeserverpages.com/learn/), écrire un script qui teste si on est le matin ou l'après-midi. En fonction de cela, écrire un commentaire du style:

Bonne matinée, nous sommes mardi 23 janvier 2001 ou *Bonne Après-Midi, nous sommes mardi 23 janvier 2001*

Bases de données avec ASP

Grâce à un script ASP, il est possible de se connecter à une base de données, rechercher de l'information et renvoyer cette information vers le client sous forme de page HTML. La base de données monagence.mdb servira de support pour l'étude des bases de données avec ASP. Cette base possède un table nommée client dont la structure est la suivante:

client : Table

Nom du champ	Type de données	Description
N°	NuméroAuto	numéro unique pour chaque enregistrement
prenom	Texte	prénom du client (50 cara max)
nom	Texte	nom du client (50 cara max)
age	Numérique	age du client
sexe	Texte	Préciser F ou M (1 cara max)
adresse	Texte	Adresse (150 cara max)
cheveux	Texte	descriptif des cheveux (50 cara max)
taille	Numérique	taille en cm
poids	Numérique	poids en Kg
photo	Texte	nom et extension du fichier photo
contact	Texte	adresse email

Affichons la valeur du champ *prenom* du premier enregistrement

a) Connexion à une base de données:

Il faut définir les 4 étapes suivantes

1) <%dim strconn →définition de la variable qui contient les information de connexion
2) strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source = C:\temp\monagence.mdb" →initialisation de la variable avec les information de connexion.
3) Set Conn = Server.CreateObject("ADODB.Connection") →création d'une connexion.
4) Conn.Open strconn →ouverture de la connexion définie en 2).

b) Création d'un recordset

1) Set RstAffiche = Server.CreateObject("ADODB.Recordset") → création d'un objet recordset.

c) Ouverture d'un recordset

1) RstAffiche.Open "Select * from client",Conn →Ouverture de tous les champs de la table client en utilisant la connexion Conn.
2) RstAffiche.MoveFirst →On se place sur le premier enregistrement s'il existe
3) If RstAffiche.eof = false then Response.Write RstAffiche("prenom").value → Teste si le recordset n'est pas vide et affichage du champs prenom.

d) Fermeture du recordset et de la connexion

1) RstAffiche.Close
2) Set RstAffiche = Nothing
3) Conn.Close
4) Set Conn = Nothing
5) %> →fermeture du script

e) Ecrire le programme et le tester

Affichons maintenant le prénom de tous les enregistrements

```
RstAffiche.MoveFirst
Do while not RstAffiche.eof
        Response.Write RstAffiche("prenom").Value%><br><%
      RstAffiche.MoveNext
Loop
```

f) Afficher tous les champs de tous les enregistrements

Affichons maintenant les résultats sous forme de tableau

Le tableau possède une ligne dans laquelle on trouvera le nom des différents champs. Les autres lignes contiendront le contenu des champs pour chaque enregistrement.

a) listons tous les noms de champs de la table

```
for i=0 to rstaffiche.fields.count-1
  Response.Write rstaffiche.fields(i).name & " "
Next
```

b) Utilisons ceci pour générer dynamiquement une la première ligne de tableau

```
<table BORDER = 1>
<tr>
<% for i=0 to rstaffiche.fields.count-1%>
<td>
        <% Response.Write rstaffiche.fields(i).name%>
</td>
<% next%>
</tr>
```

c) S'inspirer de b) pour afficher tous les enregistrements de la base de données

d) Afficher maintenant dans le tableau *(dans les cellules photo)* l'image associée à chaque enregistrement en utilisant la balise <img src = "http://pc3/imagesclients/….">

e) S'inspirer de II.6 pour envoyer la requête SQL par l'intermédiaire d'un formulaire. On essaiera plusieurs requêtes permettant de sélectionner des ensembles d'enregistrements selon des critères particuliers *(age compris entre 20 et 30 ans, afficher toutes les EVA, sélectionner en triant,…)*

EXERCICES

- Ecrire un script ASP permettant d'afficher un formulaire renseignant les différentes informations nécessaires à l'introduction d'un nouvel enregistrement *(Nom, Prenom, age, adresse,…)*.
Quand les différents champs sont renseignés, ajouter un enregistrement dans la table et affecter à chaque champ les valeurs du formulaire. *(revoir les méthodes addnew, uptdate,..).*

- Ecrire un formulaire permettant d'éditer des critères de recherche d'une personne dans la base. *(rechercher par exemple toutes les filles entre 18 et 28 ans qui sont blondes et dont le poids ne dépasse pas 50 Kg)*.
Pour cela il faut afficher des listes de choix, des radio boutons, des boutons, qui permettent de lier des critères de recherche avec des opérateurs logiques ou ensemblistes *(OR,AND,ALL,NOT, LIKE, DISTINCT,SORT BY,…)*.
L'utilisateur n'aura pas besoin de connaître SQL mais devra uniquement construire sa requête en effectuant des choix et en choisissant des opérateurs. En fonction de ces choix, le programme construit la requête, interroge la table et renvoie les résultats sou s forme de tableau à l'utilisateur.