Groupe 12 : 03/01/2008

Baverel Romain
Lacôte Hubert
Moine Julien
Stuber Guillaume

Rapport de Projet : BD40

Sujet 13 :
Gestion d'une CV thèque

Automne 2008 Enseignant : Christian Fischer

Sommaire

Introduction

Dans ce rapport, nous allons présenter notre projet de bd40. Ce projet a consisté à réaliser un logiciel de gestion d'une CV Thèque permettant de classer des CV ainsi que des offres d'emploi. Nous détaillerons les parties importantes de la conception de ce projet.

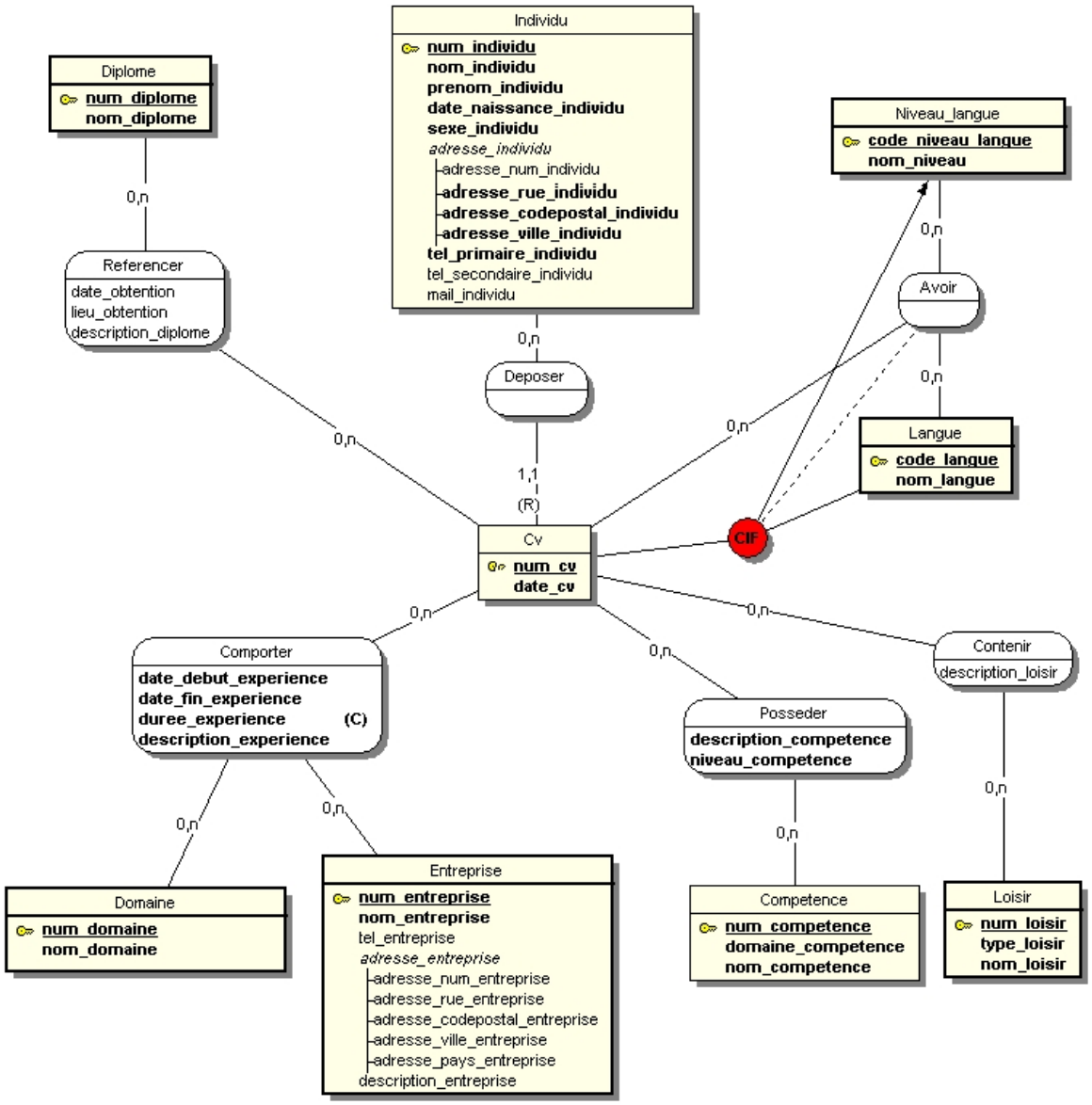
Nous avons choisi ce sujet car c'est un sujet qui nous semblait concret et parce qu'il se rapprochait de la réalité, le tout, en mettant en application ce que nous avons appris ce semestre.

Nous avons conçu cette CV Thèque de la sorte qu'une personne ait accès à toutes les données et puisse rentrer des CV ainsi que des offres d'emploi. Dans la réalité, notre logiciel pourrait être utilisé de cette manière : un employé d'une agence intérim recevra des CVs et des offres d'emploi qu'il saisira. Il pourra consulter, modifier, supprimer et rechercher suivant des critères précis dans la base de données.

I/ Etudes préalables

1) MEA

Le MEA de notre CV Thèque est divisé en deux parties, une première partie concerne le dépôt de CV :

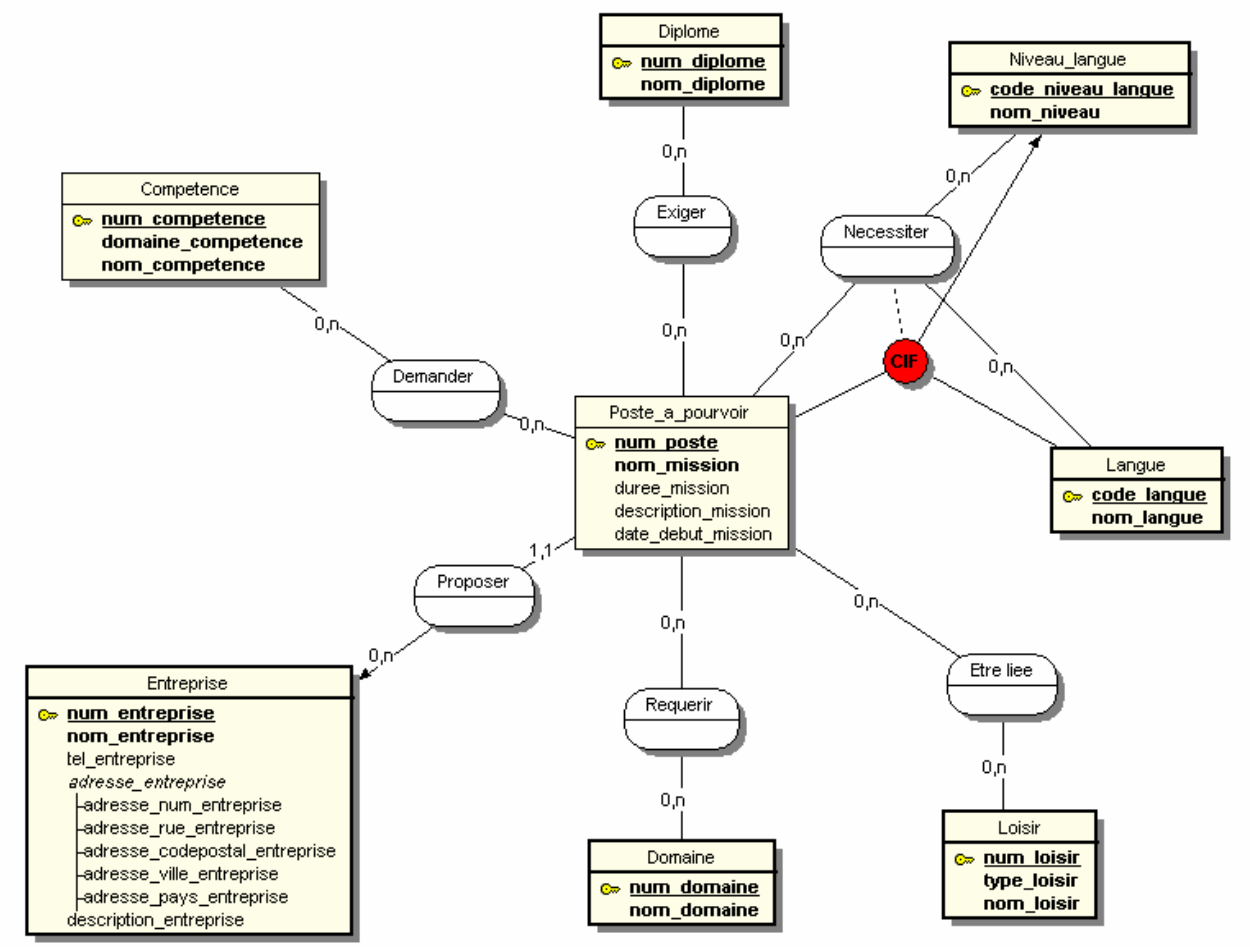


On peut voir sur ce modèle que le centre de notre modèle est l'entité CV, elle-même dépendant de l'existence d'une entité Individu, un CV ne pouvant exister sans celui-ci.

Nous avons dressé ce modèle avec comme objectif d'avoir plusieurs tables génériques; par tables génériques on peut entendre des tables contenants des informations qui n'ont pas attrait directement à l'individu : par exemple l'entité Diplôme deviendra une table complétée avec toute sorte de diplôme existant, alors que les informations personnelles concernant les diplômes de l'Individu concerné (date et lieu d'obtention, description) seront dans l'association « Référencer ». Ainsi sont construites les autres entités, à savoir Domaine, Entreprise, Compétence, Loisir, Langue, Niveau langue (qui est une entité car nous normalisons les niveaux de langue). Et de la sorte nous pouvons réutiliser ces tables « génériques » pour la deuxième partie de notre MEA, la partie traitant des offres d'emplois.

Enfin la Contrainte d'Intégrité Fonctionnelle (CIF sur le modèle) nous permet d'éviter d'avoir la clé primaire code_niveau_langue en clé primaire de la table qui se créera à partir de l'association Nécessiter lors du passage au Modèle Logique des Données. Ainsi on ne pourra pas avoir dans le même CV deux fois la même langue avec des niveaux différents.

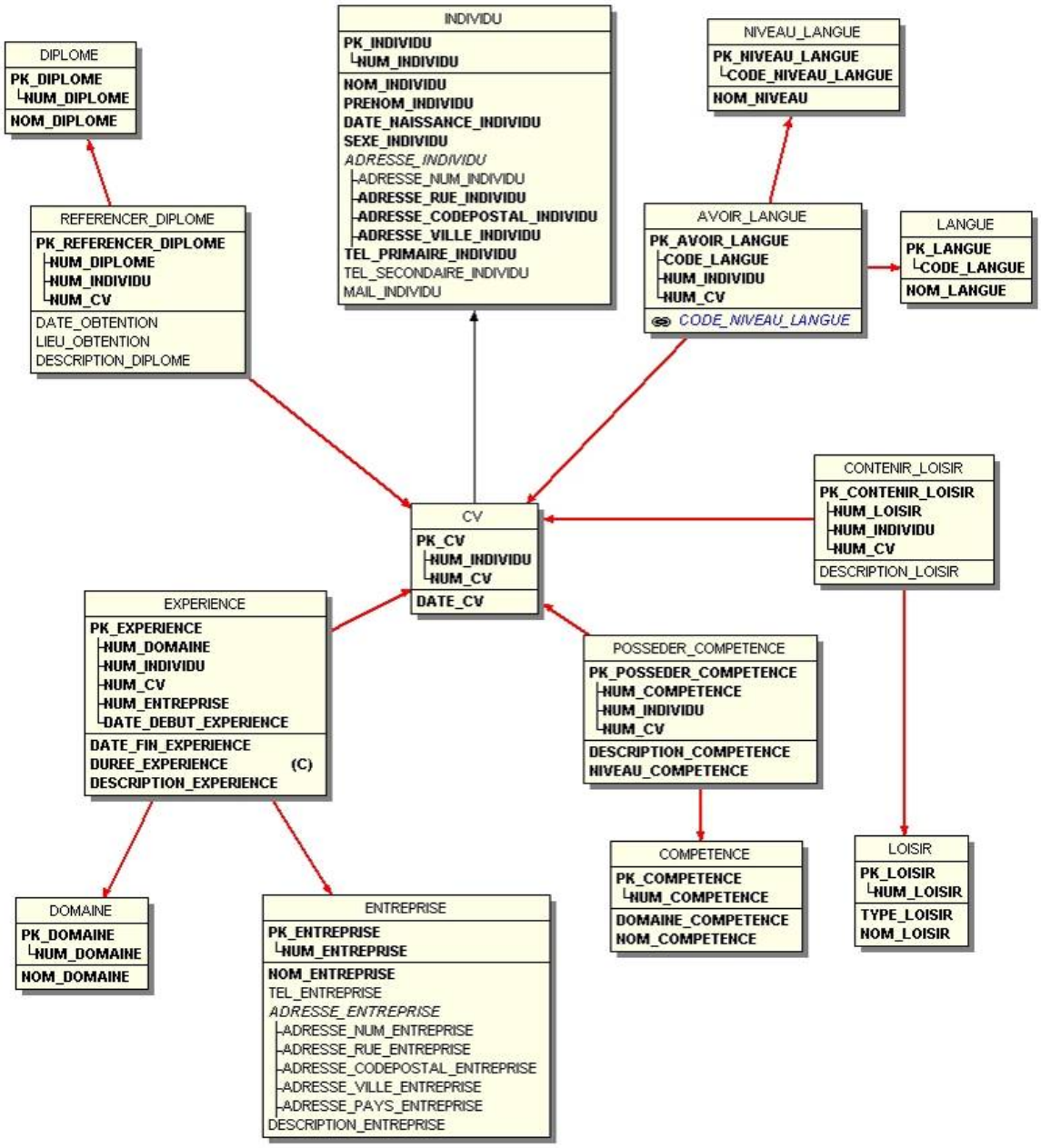
Voici le Modèle Entité Associations concernant la partie offre d'emplois.

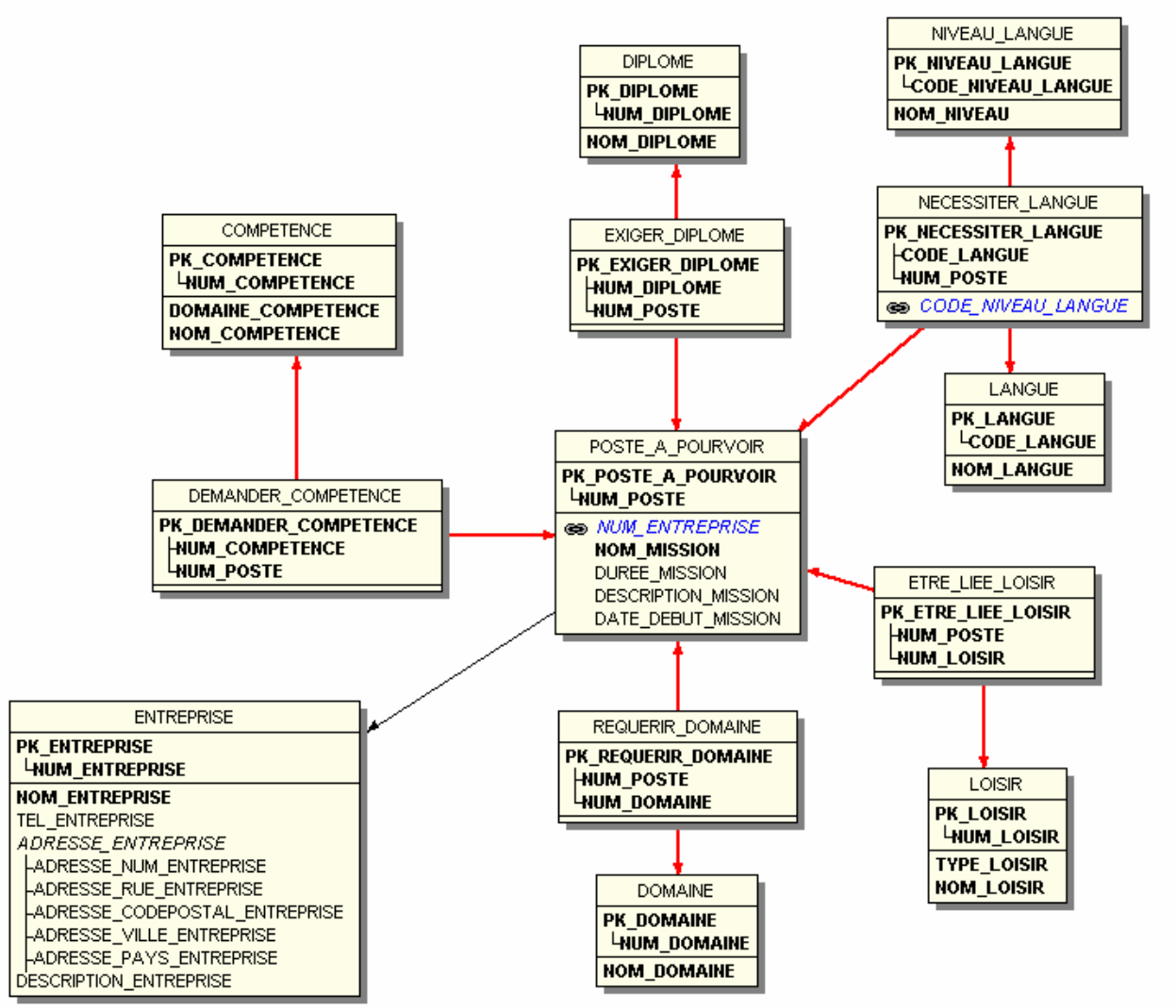


Ce MEA est construit sensiblement de la même façon que le premier : un élément central (ici, l'entité Poste_a_pourvoir) entouré des différentes entités « génériques », celles-ci étant bien évidemment les mêmes que sur le premier modèle. Il n'y a ici rien dans les relations ; en effet, les offres d'emploi se basent uniquement sur des données génériques.

2) MLD

Voici les Modèles Logiques des Données découlant de ces deux Modèles Entités Associations :




DIPLOME
PK_DIPLOME
NUM_DIPLOME
NOM_DIPLOME
NIVEAU_LANGUE
PK_NIVEAU_LANGUE
CODE_NIVEAU_LANGUE
NOM_NIVEAU
COMPETENCE
PK_COMPETENCE
NUM_COMPETENCE
DOMAINE_COMPETENCE
NOM_COMPETENCE
EXIGER_DIPLOME
PK_EXIGER_DIPLOME
NUM_DIPLOME
NUM_POSTE
NECESSITER_LANGUE
PK_NECESSITER_LANGUE
CODE_LANGUE
NUM_POSTE
CODE_NIVEAU_LANGUE
LANGUE
PK_LANGUE
CODE_LANGUE
NOM_LANGUE
DEMANDER_COMPETENCE
PK_DEMANDER_COMPETENCE
NUM_COMPETENCE
NUM_POSTE
POSTE_A_POURVOIR
PK_POSTE_A_POURVOIR
NUM_POSTE
NUM_ENTREPRISE
NOM_MISSION
DUREE_MISSION
DESCRIPTION_MISSION
DATE_DEBUT_MISSION
ETRE_LIEE_LOISIR
PK_ETRE_LIEE_LOISIR
NUM_POSTE
NUM_LOISIR
ENTREPRISE
PK_ENTREPRISE
NUM_ENTREPRISE
NOM_ENTREPRISE
TEL_ENTREPRISE
ADRESSE_ENTREPRISE
ADRESSE_NUM_ENTREPRISE
ADRESSE_RUE_ENTREPRISE
ADRESSE_CODEPOSTAL_ENTREPRISE
ADRESSE_VILLE_ENTREPRISE
ADRESSE_PAYS_ENTREPRISE
DESCRIPTION_ENTREPRISE
REQUERIR_DOMAINE
PK_REQUERIR_DOMAINE
NUM_POSTE
NUM_DOMAINE
LOISIR
PK_LOISIR
NUM_LOISIR
TYPE_LOISIR
NOM_LOISIR
DOMAINE
PK_DOMAINE
NUM_DOMAINE
NOM_DOMAINE

3) Dictionnaire des données

On peut retrouver les différents attributs des modèles dans le dictionnaire des données ci-dessous. Le type (E = Elémentaire, C = Composé, Ca = Calculé) et la nature (N = Numérique, AN = Alphanumérique, D = Date) des données sont indiqués.

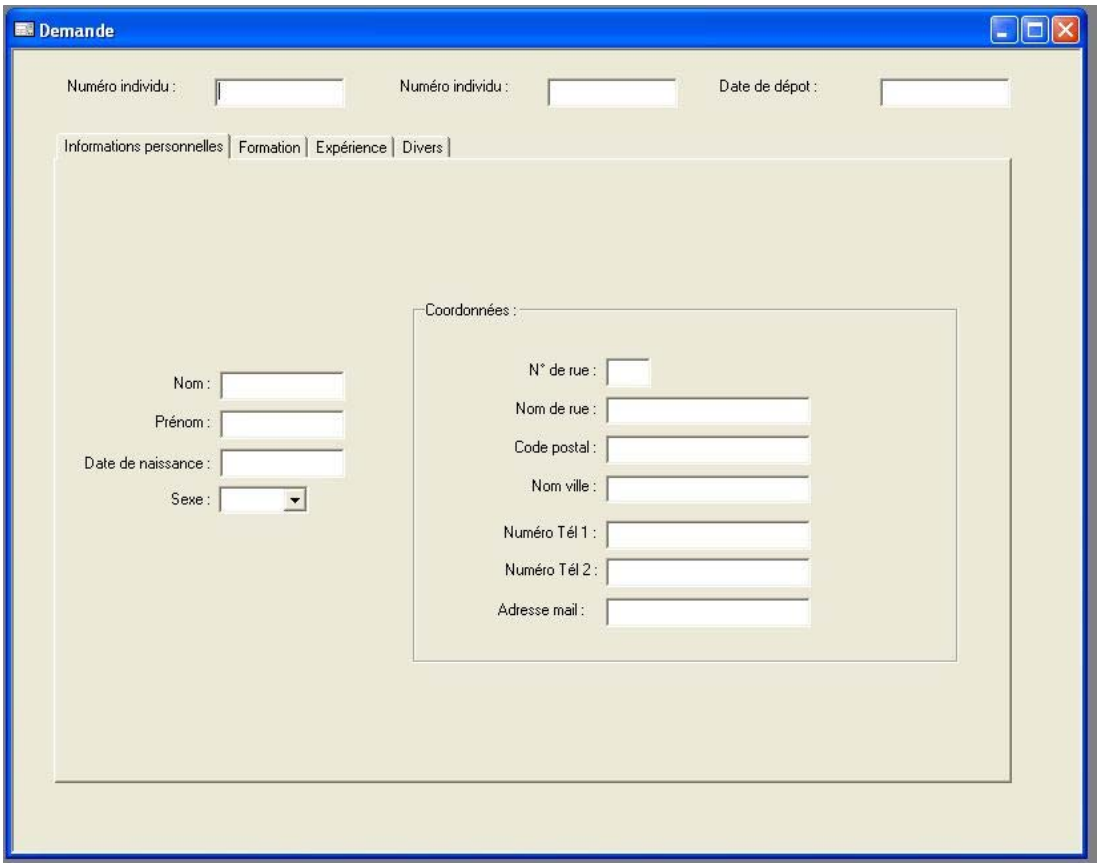
Nom	Type	Nature	Identifiant
Individu			
num_individu	E	N	oui
nom_individu	E	AN	
prenom_individu	E	AN	
date_naissance_individu	E	D	
sexe_individu	E	AN	
adresse_individu	C	AN	
adresse_num_individu	E	N	
adresse_rue_individu	E	AN	
adresse_codepostal_individu	E	AN	
adresse_ville_individu	E	AN	
tel_primaire_individu	E	AN	
tel_secondaire_individu	E	AN	
mail_individu	E	AN	
CV			
num_cv	E	N	oui
date_cv	E	D	
Diplôme			
num_diplome	E	N	oui
nom_diplome	E	AN	
date_obtention	E	D	
lieu_obtention	E	AN	
description_diplome	E	AN	
Langue			
code_langue	E	N	oui
nom_langue	E	AN	
Niveau langue			
code_niveau_langue	E	N	oui
nom_niveau_langue	E	AN	
Loisir			
num_loisir	E	N	oui
type_loisir	E	AN	
nom_loisir	E	AN	
description_loisir	E	AN	
Compétence			
num_competence	E	AN	oui
domaine_competence	E	AN	
nom_competence	E	AN	
description_competence	E	AN	
niveau_competence	E	AN	

Entreprise			
num_entreprise	E	N	oui
nom_entreprise	E	AN	
tel_entreprise	E	AN	
adresse_entreprise	C	AN	
adresse_num_entreprise	E	N	
adresse_rue_entreprise	E	AN	
adresse_codepostal_entreprise	E	AN	
adresse_ville_entreprise	E	AN	
adresse_pays	E	AN	
description_entreprise	E	AN	
Domaine			
num_domaine	E	N	oui
nom_domaine	E	AN	
Comporter expérience			
date_debut_expérience	E	D	
date_fin_expérience	E	D	
duree_expérience	Ca	N	
description_expérience	E	AN	
Poste à pourvoir			
num_poste	E	N	oui
nom_mission	E	AN	
duree_mission	E	N	
desription_mission	E	AN	
date_debut_mission	E	D	

4) Maquette

Afin de pouvoir avoir un aperçu de notre projet, nous avons d'abord réalisé une maquette sous Win'Design. Cela nous a permis de visualiser l'aspect des formulaires à mettre en œuvre et ainsi en faciliter la conception.

Voilà ci-dessous quelques images de notre maquette représentant différents formulaires :

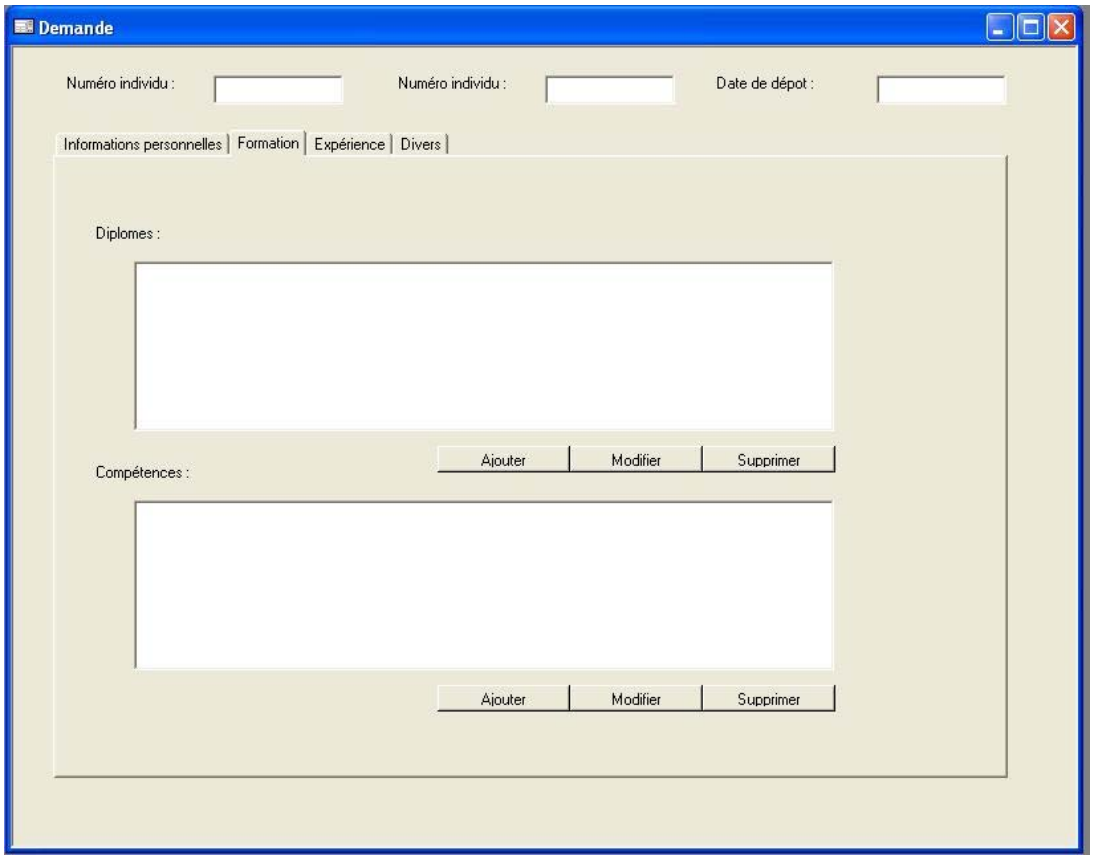


Nous voyons ci-dessus le choix adopté pour l'affichage d'un CV. Nous verrons dans une autre partie que le même formulaire est finalement utilisé que ce soit pour la consultation de CV, pour sa modification ou encore pour sa création. Ainsi chaque information du CV est présentée suivant divers onglets types.

Le premier onglet, « Informations personnelles », présente tout simplement toutes les informations propre à l'individu déposant son CV. On peut y retrouver son nom, prénom, date de naissance, sexe ainsi que ses coordonnées (adresse, téléphone, adresse électronique).

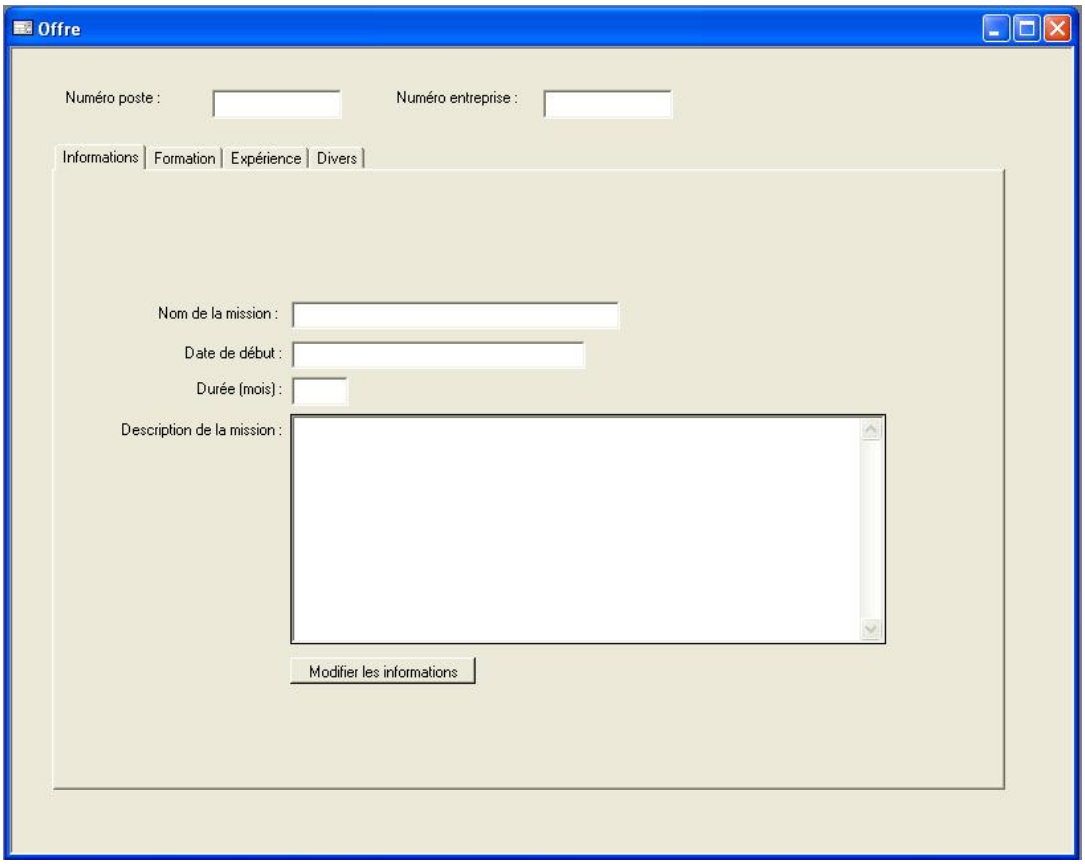
L'onglet « Formation », en image ci-dessous, contient les différents diplômes ainsi que les compétences du CV. On peut y observer différents boutons : « Ajouter » qui nous permet simplement d'ajouter une entrée à la section concerné, « Modifier » qui sert à en modifier une et enfin « Supprimer » qui permet la suppression d'une à plusieurs entrées sélectionnées.

L'onglet suivant est l'onglet « Expérience » qui contient les expériences personnelles en entreprise de l'individu concerné par le CV.



La maquette suivante (ci-dessous) est celle qui concerne le formulaire ouvert lorsqu'on veut consulter/modifier/supprimer une offre. Il s'agit là d'un formulaire de sélection parmi plusieurs autres, tous destinés à faciliter la sélection d'informations particulières, par la mise à disposition de moyens de recherche (par nom, intitulé,…).

Pour finir voici la maquette montrant comment sont affichées les offres d'emploi. On retrouve des onglets regroupant les différentes informations classées par catégories comme pour l'affichage d'un CV.



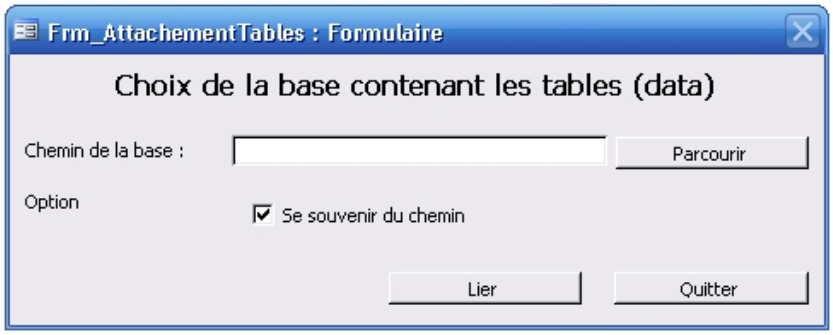
II/ Interface Access

1) Informations préalables

a) Installation

b) Mise à jour des tables

Lors d'un changement de place de la base CVTHEQUE_DATA (contenant donc les données) ou tout simplement de notre CVTHEQUE_APPLI, ou lors de l'installation de la base sur un nouveau poste, il se peut que les tables de CVTHEQUE_APPLI ne soient plus liées. Pour régler cela notre base de données ouvre à chaque nouveau lancement un formulaire de rattachement des tables. Grâce à la table LIAISON_DATA dans laquelle se trouve le dernier chemin de la base CVTHEQUE_DATA connu, si celle-ci est trouvée à l'emplacement indiqué, le formulaire ne s'affiche pas à l'écran. Sinon il faut rattacher les tables :

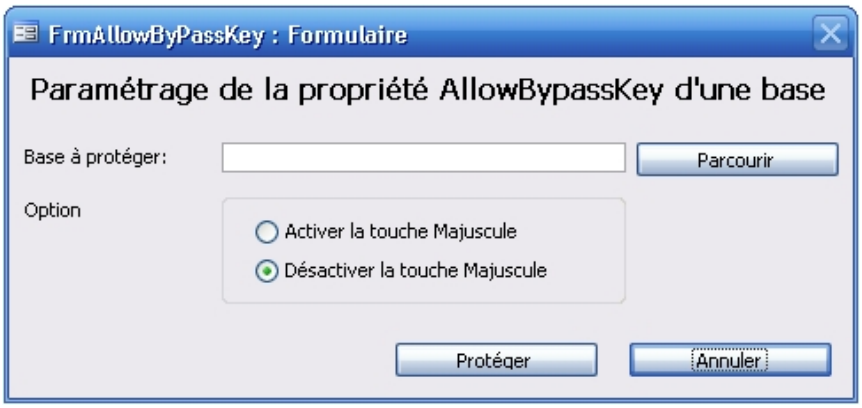


La case à cocher permet d'enregistrer le nouveau chemin dans la table LIAISON_DATA.

c) Sécurité

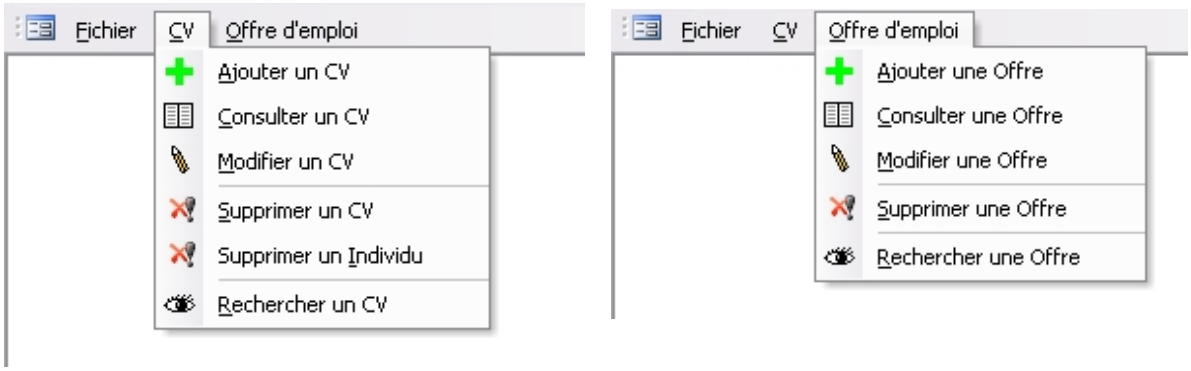
L'utilisateur de l'application n'aura que faire de pouvoir accéder directement aux tables, aux requêtes ou encore au code derrière les formulaires. Et tout simplement pour une question de sécurité, il va falloir bloquer cet accès. Ainsi, Microsoft Access propose effectivement une sécurité, mais celle-ci est facilement franchissable, la touche Shift enfoncée au début de l'application suffisant.

C'est pourquoi la base BaseSecu existe : elle permet grâce à un formulaire d'enlever cette possibilité d'outrepasser la sécurité avec shift, ou de la permettre. Ce formulaire se présente comme suit :

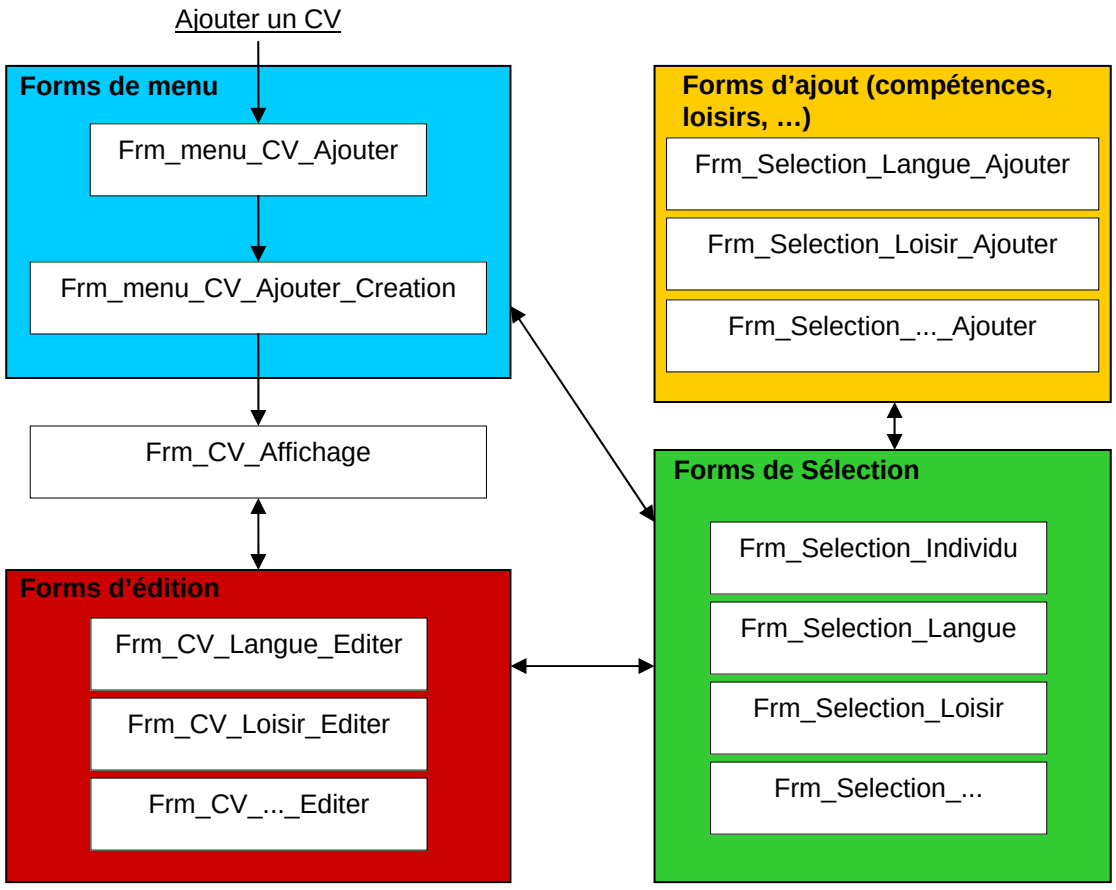


d) Organisation des formulaires

Commençons par la source : la barre de menu.



« Fichier » ne contient que l'entrée Quitter. Nous ne nous attarderons donc pas dessus. Chaque entrée des deux autres menus ouvre différents formulaires. Nous allons présenter ici ces différents formulaires. Nous verrons ici la partie CV, celle concernant la partie offre étant similaire. Le schéma ci-dessous est le processus de parcours des formulaires à partir de l'entrée « Ajouter un CV ».



Quelques explications s'imposent. Tout d'abord, une fois l'entrée sélectionnée, un formulaire du type Frm_Menu s'ouvre, ici le formulaire Frm_Menu_CV_Ajouter, qui demande le numéro d'individu. Nous pouvons récupérer ce numéro grâce à l'ouverture d'un formulaire de sélection via le bouton point d'interrogation  . On retrouve dans l'application à plusieurs reprise ce bouton qui à chaque fois ouvrira un formulaire de sélection. Celui-ci permet de rechercher l'objet de la sélection ainsi que d'en crée un nouveau dans le cas où il n'existerait pas encore. Pour ce faire, le formulaire de sélection appelle un Frm_Selection_..._Ajouter. Nous avons ici un deuxième Frm_Menu qui sert à donner à l'utilisateur le choix entre créer un nouveau CV vide ou bien à partir d'un ancien CV. Une fois le choix effectué, nous arrivons sur le Frm_CV_Affichage. Celui-ci est le même pour l'ajout d'un nouveau CV ainsi que pour la consultation ou la modification d'un CV. Il permet comme son nom l'indique à afficher le CV sous-forme d'onglet. Nous le détaillerons plus tard dans ce rapport. Parallèlement, Frm_Offre_Affichage est le formulaire d'affichage pour les offres, commun à l'ajout, la modification et la consultation de celle-ci.

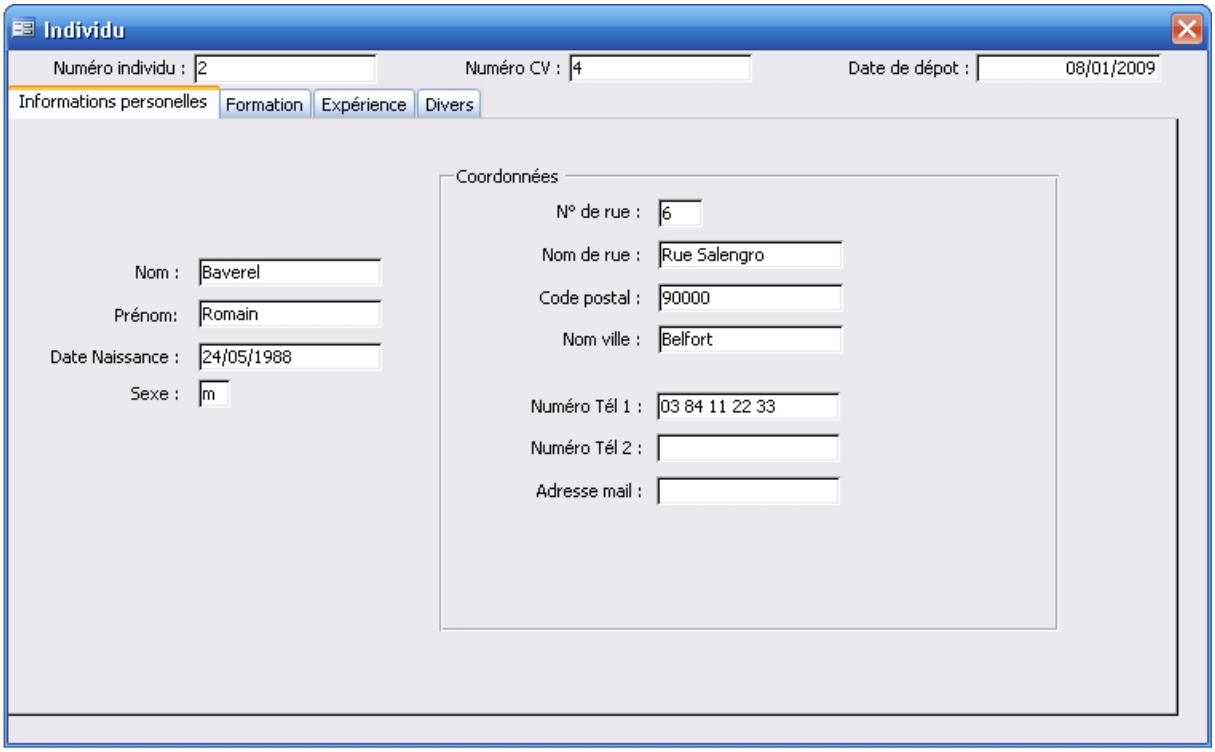
Il y a possibilité d'ajouter ou de modifier différentes informations au CV dans ce formulaire d'affichage. Ces options appellent les Frm_CV_..._Editer qui eux même pourront appeler des Frm_Selection. Ces derniers sont communs aux CV et aux offres. Les formulaires d'ajout de compétences, de langues… sont également communs aux deux parties.

Enfin les deux entrées de « suppression » n'ouvre qu'un formulaire de menu (avec formulaire de sélection associé) alors que les entrées de recherche ouvrent directement des formulaires de sélection suivis par le formulaire d'affichage du CV ou de l'offre.

2) Choix sur l'interface et codes associés

a) Formulaires d'affichage

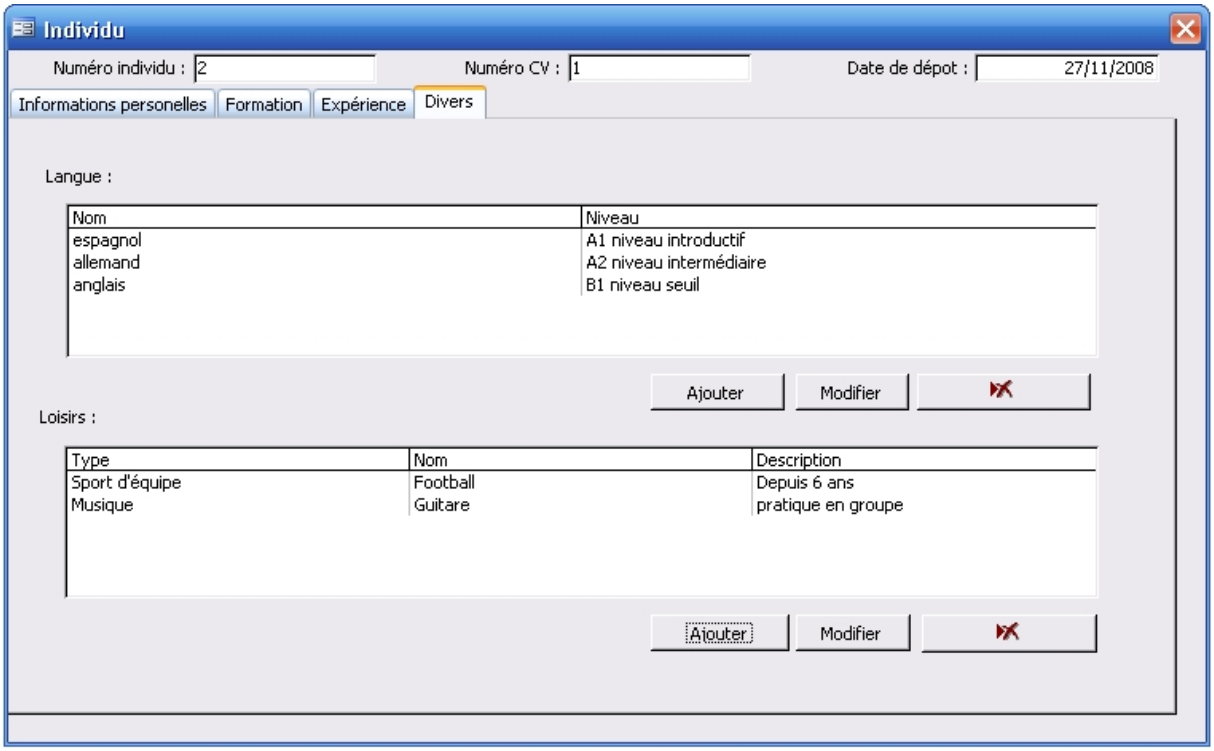
Les formulaires d'affichage de CVs ou d'offres sont les formulaires centraux de notre application. C'est par leur intermédiaire que l'on va pouvoir consulter, ajouter ou modifier respectivement des CVs ou des offres. Il se présente comme suit : (nous ne traiterons que de la partie CV et non de la partie offre ; en effet les deux étant construits sensiblement sur la même base, il aurait été futile de les traiter séparément)



Ce premier onglet n'est pas très compliqué, il ne s'agit que de l'exploitation de la source du formulaire, à savoir une simple jointure sur les tables CV et INDIVIDU. Le numéro d'individu et celui de CV sont tous deux récupérés grâce au filtre passé par le formulaire appelant précédent (un formulaire de menu). Ce filtre sera de la forme :

```
[INDIVIDU]![NUM_INDIVIDU] = … AND [CV]![NUM_CV]= …
```

Nous pouvons à présent passer sur d'autres onglets de ce formulaire. Un exemple suffira car les onglets « Formations », « Expérience » et « Divers » sont construits sur le même principe. L'image ci-dessous correspond à l'onglet « Divers ». On y retrouve les langues ainsi que les loisirs du CV. Nous avons choisi pour afficher ces différentes informations de le faire dans des listbox, qui nous semblaient l'outil le plus adéquat à un nombre variable de données.



On peut voir trois boutons sous chaque listbox : ajouter, modifier et une icône pour supprimer. Les boutons ajouter et modifier ouvrent tous les deux le même formulaire d'édition. Ces trois boutons ne sont pas présents lorsqu'il s'agit de consulter le CV. Ceci grâce aux arguments différents par rapport à l'ajout :

Code visual basic

```
DoCmd.OpenForm "Frm_CV_Affichage", acNormal, , "[INDIVIDU]![NUM_INDIVIDU] = " &
Me.txtNumIndividu & " AND [CV]![NUM_CV] = " & Me.txtNumCV, acFormPropertySettings,
acDialog, "#NO_AJOUT#NO_MODIF#NO_SUPPR"
```

Nous avons déjà vu la particularité du filter. Ce qui est intéressant ici est le dernier argument en vert. Cet argument deviendra l'option OpenArgs de Frm_CV_Affichage. A l'ouverture de celui-ci, le module CacherBoutons est appelé :

Code visual basic

```
Public Sub cacherBoutonsSelonArgumentsOuverture(ByVal la_form As Form)
    Dim masquerBoutonsAjout As Boolean
    Dim masquerBoutonsModif As Boolean
    Dim masquerBoutonsSuppr As Boolean
    Dim ctl As Control
    Dim typeCtl As String

 'Nous vérifions ici quels arguments on été transmis au formulaire concerné grâce à la fonction InStr.  Grâce à des booléens, on mémorise s'il faut masquer les boutons concernés

    If InStr(1, la_form.OpenArgs, "#NO_AJOUT") > 0 Then
        masquerBoutonsAjout = True
    End If
    If InStr(1, la_form.OpenArgs, "#NO_MODIF") > 0 Then
        masquerBoutonsModif = True
    End If
    If InStr(1, la_form.OpenArgs, "#NO_SUPPR") > 0 Then
        masquerBoutonsSuppr = True
    End If


'Enfin on parcourt chaque contrôles du formulaire et s'il s'agit d'un bouton à masquer on met sa propriété Visible à false

    For Each ctl In la_form.Controls
        typeCtl = Left(ctl.Name, 8)
        If masquerBoutonsAjout And typeCtl = "btnAjout" Then
            ctl.Visible = False
        End If
        If masquerBoutonsModif And typeCtl = "btnModif" Then
            ctl.Visible = False
        End If
        If masquerBoutonsSuppr And typeCtl = "btnSuppr" Then
            ctl.Visible = False
        End If
    Next ctl
End Sub
```

La source de chaque listbox est réactualisée à chaque changement (ajout, modification, suppression). On utilise pour cela la fonction RefreshQuery du formulaire. Celle-ci va parcourir chaque listbox du formulaire d'affichage et remettre la même requête sql en rowsource de celle-ci est surtout lui appliquer la méthode « Requery ». Un Exemple sur une des listes :

Code visual basic

```
'mise à jour de la liste Diplôme
    Dim sql As String

    sql = "SELECT DIPLOME.NUM_DIPLOME, DIPLOME.NOM_DIPLOME as Nom,
REFERENCER_DIPLOME.DATE_OBTENTION as [Date d'obtention],
REFERENCER_DIPLOME.LIEU_OBTENTION as [Lieu d'obtention],
REFERENCER_DIPLOME.DESCRIPTION_DIPLOME as [Description] "
```

```
   sql = sql & "FROM DIPLOME INNER JOIN REFERENCER_DIPLOME ON
DIPLOME.NUM_DIPLOME=REFERENCER_DIPLOME.NUM_DIPLOME "

   sql = sql & "WHERE NUM_CV = " & numeroDuCV & "AND NUM_INDIVIDU = " &
numeroDeLIndividu & " "

   sql = sql & "ORDER BY REFERENCER_DIPLOME.DATE_OBTENTION DESC;"

   Me.lstFormation.RowSource = sql
   Me.lstFormation.Requery
```

Avant de passer au formulaire suivant, attardons-nous sur la façon dont sont supprimées les entrées de nos listbox avec les boutons de suppression. La particularité étant que nous pouvons supprimer plusieurs entrées en en sélectionnant plusieurs à la fois :

Code visual basic

```
Private Sub btnSupprimerCompetences_Click()
   supprimerElement "compétence", Me.lstCompetence, "supprimerCompetence", 1
End Sub
```

Dans un premier temps on appelle la fonction générique « supprimerElement » qui prend en argument, le nom de l'information concernée, la listBox concernée ainsi que la fonction de suppression associée. Le dernier chiffre est le nombre de clés de la fonction cité dans l'argument précédent. Nous comprendrons mieux le rôle de ce dernier argument en regardant la suite :

Code visual basic

```
Private Sub supprimerElement(ByVal nomElement As String, ByVal liste As ListBox, ByVal
nomFonction As String, ByVal nbCles As Integer)

   Dim nbSelected As Integer

'On initialise tout d'abord le nombre d'éléments sélectionnés.
   nbSelected = liste.ItemsSelected.Count

'S'il est différent de 0 alors on demande confirmation de la suppression
   If nbSelected > 0 Then
      If MsgBox("Voulez-vous supprimer " & nbSelected & " " & nomElement & "(s) de ce CV ?",
vbQuestion & vbYesNo, "Confirmation.") = vbYes Then

         Dim numIndividu As Integer
         Dim numCV As Integer
         Dim numCle1 As Integer
         Dim numCle2 As Integer
         Dim numCle3 As Variant
         Dim numSelected As Integer
         Dim i As Integer

         numIndividu = Me.txtNumIndividu
         numCV = Me.txtNumCV
```

```
'On boucle alors sur chaque élément sélectionné et on appelle pour chaque entrées la
fonction de suppression passée en argument grâce à la fonction CallByName. Celle-ci ne
fera qu'exécuter la requête de suppression avec les éléments récupérés par cette fonction.
        For i = 1 To nbSelected
            numSelected = liste.ItemsSelected.Item(i - 1)
            numCle1 = liste.Column(0, numSelected)
            If (nbCles = 3) Then
                numCle2 = liste.Column(1, numSelected)
                numCle3 = liste.Column(2, numSelected)
                CallByName Me, nomFonction, VbMethod, numIndividu, numCV, numCle1, numCle2,
numCle3
            Else
                CallByName Me, nomFonction, VbMethod, numIndividu, numCV, numCle1
            End If
        Next i

'Puis bien sûr comme il y a eu un changement on appelle la fonction RefreshQuery de notre
formulaire.
        Me.RefreshQuery
    End If
  End If
End Sub
```

Nous avons précédemment dit que les boutons « ajouter » et « modifier » ouvraient des formulaires d'édition. Grâce à ceux-ci nous pouvons donc manipuler des informations via notamment les formulaires de sélections. Voyons donc comment ces derniers fonctionnent.

b) Formulaires de sélection

Avant d'attaquer le formulaire en soi, nous devons regarder comment celui-ci est ouvert. Le formulaire d'édition va l'appeler grâce à la fonction ouvertureFormDeSelection du module du même nom :

Code visual basic

```
Private Sub btnSelectionLoisir_Click()
   ouvertureFormDeSelection Me, "Frm_Selection_Loisir", "", "selectionLoisir"
End Sub
```

Les arguments de cette fonction sont les suivants : le formulaire actif (Me), le formulaire de sélection correspondant, un filtre si nécessaire et enfin la fonction de récupération des informations (ici c'est la fonction selectionLoisir qui a pour rôle de récupérer dans le formulaire de sélection de loisir la ligne de loisir sélectionnée).

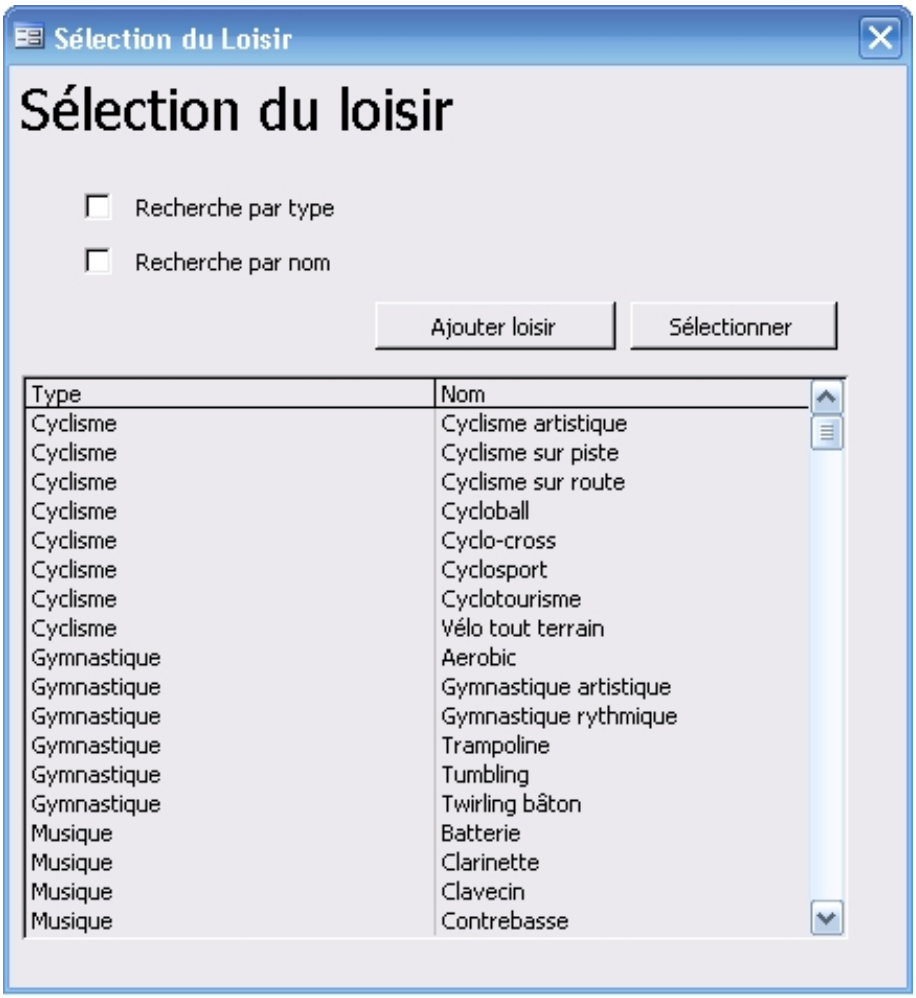
Voici la fonction dite :

Code visual basic

```
Public Sub ouvertureFormDeSelection(ByVal formAppelant As Form, ByVal nomForm As String,
ByVal filtre As String, ByVal fonctionSelection As String)
   Dim formOuvert As Form

'On ouvre le formulaire de selection passé en paramètre avec le filtre également passé en
paramètre
   DoCmd.OpenForm nomForm, acNormal, , , acFormPropertySettings, acWindowNormal, filtre
   Set formOuvert = Forms.Item(nomForm)

'Cette fonction se met en « pause » tant que la variable globale « opened » du formulaire de
sélection ouvert est à true.
   Do While formOuvert.opened
      DoEvents
   Loop
   formOuvert.SetFocus

'Une fois cette variable à false, on peut appeler la fonction de récupération passée en
paramètre grâce à la fonction CallByName, puis fermer le formulaire de sélection.
   If CurrentProject.AllForms.Item(nomForm).IsLoaded Then
      CallByName formAppelant, fonctionSelection, VbMethod, formOuvert
      DoCmd.Close acForm, nomForm, acSaveNo
   End If
End Sub
```

La fonction de récupération ne fait qu'accéder directement aux champs de la listBox pour mettre à jour la textBox du formulaire d'édition concerné. La fonction ouvertureFormDeSelection a pour avantage de pouvoir s'utiliser avec tous les formulaires d'édition au moment de leur appel aux formulaires de sélections.

Passons désormais dans le vif du sujet, le formulaire de sélection. Ci-dessous le formulaire de sélection du loisir :



Ici nous n'allons pas rentrer dans le détail des boutons : le premier « Ajouter loisir » ouvre donc un formulaire d'ajout alors que le deuxième permet de « sélectionner » l'entrée surlignée pour le formulaire d'édition appelant.

Nous allons plutôt nous attarder sur le rafraichissement de la listBox et sur la recherche. En effet nous avons au dessus de la listBox des cases à cocher qui font apparaître des textBox une fois cochés. Celles-ci permettent de rechercher suivant différents critères. Ici, en cochant les deux cases, on peut faire des recherches parmi les loisirs selon le type de loisir ou encore par nom. Nous retrouvons pour cela la fonction RefreshQuery qui présente des similitudes avec celles du formulaire d'affichage. A nouveau à chaque changement, la listBox est rafraîchi en appelant cette fonction.

Code visual basic

```
Private Sub chkRechType_Click()
   Me.txtRechType.Visible = Not Me.txtRechType.Visible
   RefreshQuery
End Sub
```

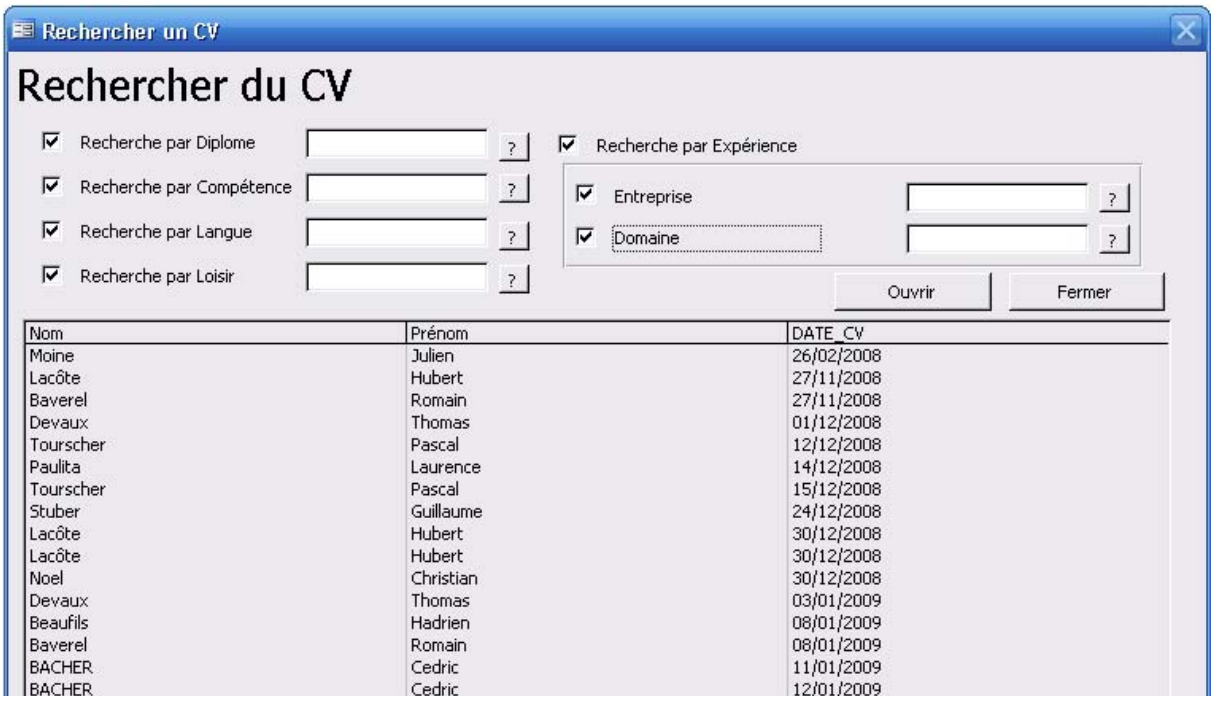
Ici, on rafraîchi la listBox quand on clique sur la case à cocher du type (chkRechType) et on fait apparaître la textBox associé.

Et voilà la fonction RefreshQuery associé au formulaire de sélection du loisir :

Code visual basic

```
Public Sub RefreshQuery()
   Dim sql As String
   Dim i As Long

'On construit dans un premier temps le début de notre requête avec un critère pour le where qui paraît inutile mais sert tout simplement à ne pas provoquer d'erreur s'il n'y a pas d'autres critères
   sql = "SELECT NUM_LOISIR, TYPE_LOISIR AS Type, NOM_LOISIR as Nom FROM LOISIR WHERE NUM_LOISIR <> 0 "

' Ensuite on complète cette requête avec les différents critères, suivant les cases cochés et le contenu des textBox
   If Me.chkRechType Then
      sql = sql & " AND TYPE_LOISIR like '*" & Me.txtRechType & "*' "
   End If
   If Me.chkRechNom Then
      sql = sql & " AND NOM_LOISIR like '*" & Me.txtRechNom & "*' "
   End If

'Enfin on clôt notre requête avec un tri. Cette requête est mise dans la source de notre listBox. Enfin on la rafraîchi grâce à la méthode requery.
   sql = sql & " ORDER BY TYPE_LOISIR, NOM_LOISIR ;"
   Me.lstResultat.RowSource = sql
   Me.lstResultat.Requery
End Sub
```

Enfin pour finir sur les formulaires de sélection, nous allons en voir un particulier : le formulaire de recherche de CV .



Nous avons déjà beaucoup plus de critères de recherche. Ici, la différence est surtout sur la requête. En effet, celle-ci ne s'applique désormais plus sur une seul table, mais sur plusieurs. Voici le début de la fonction RefreshQuery :

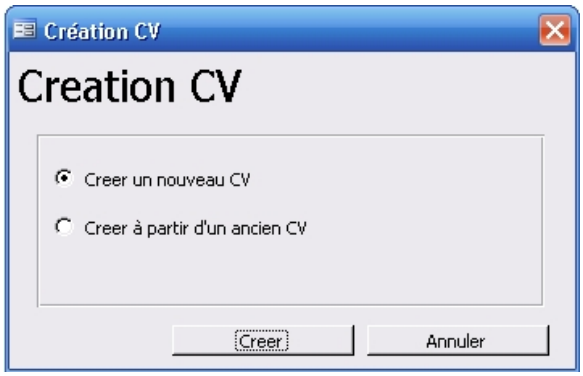
Code visual basic

```
Private Sub RefreshQuery()
   Dim sql As String
   Dim i As Long

'Comme tout à l'heure on commence par mettre le début de la requête
   sql = "SELECT  CV.NUM_INDIVIDU,CV.NUM_CV,INDIVIDU.NOM_INDIVIDU as Nom,
INDIVIDU.PRENOM_INDIVIDU as Prénom, CV.DATE_CV FROM CV INNER JOIN INDIVIDU ON
CV.NUM_INDIVIDU = INDIVIDU.NUM_INDIVIDU WHERE NUM_CV <> 0 "

'Puis suivant les critères, on vérifie dans les tables annexes grâce à la fonction EXISTS
   If chkRechDiplome Then
      If Me.txtRechDiplome <> "" Then
         sql = sql & "AND EXISTS ( SELECT  * FROM REFERENCER_DIPLOME " & _
                  "WHERE REFERENCER_DIPLOME.NUM_DIPLOME = " & Me.txtRechDiplome &
                  "AND REFERENCER_DIPLOME.NUM_CV = CV.NUM_CV AND
REFERENCER_DIPLOME.NUM_INDIVIDU = CV.NUM_INDIVIDU)"
      End If

   End If


   If chkRechCompetence Then
      If Me.txtRechCompetence <> "" Then
         sql = sql & "AND EXISTS " & _
              "( SELECT  * FROM POSSEDER_COMPETENCE " & _
              "WHERE POSSEDER_COMPETENCE.NUM_COMPETENCE = " &
Me.txtRechCompetence & " " & _
              "AND POSSEDER_COMPETENCE.NUM_CV = CV.NUM_CV " & _
              "AND POSSEDER_COMPETENCE.NUM_INDIVIDU = CV.NUM_INDIVIDU) "
   End If

…..
```

c) Formulaire de Menu

1. Formulaire d'ajout

Depuis la barre de menu, on peut ajouter un CV. Pour cela, il faut que la personne existe, on la recherche d'abord, puis on arrive au formulaire de choix suivant :



Il est en effet possible de choisir entre « Créer un nouveau CV » afin de partir d'un CV vide (bien entendu, les informations de la personne sont fixes). Dans ce cas, nous créons juste un CV associé à la personne sélectionnée précédemment et nous mettons la date d'aujourd'hui.

Code visual basic

```
Private Sub btnCreerCV_Click()
    Dim sqlWhere As String
    Dim max As Integer

'Récupération du numéro d'individu
    sqlWhere = "[NUM_INDIVIDU] = " & Me.OpenArgs

'Si le numéro du dernier CV n'existe pas NUM_CV vaudra 1 car c'est son premier CV
    If IsNull(DMax("[NUM_CV]", "CV", sqlWhere)) Then
        max = 1
    Else

'On récupère le numéro du dernier CV
        max = DMax("[NUM_CV]", "CV", sqlWhere)
        max = max + 1
    End If

'Requête d'insertion d'un CV
    DoCmd.RunSQL ("INSERT INTO CV(NUM_INDIVIDU,NUM_CV,DATE_CV) VALUES('" &
Me.OpenArgs & "','" & max & "','" & Date & "')")
```

Dans le cas où l'on veut mettre à jour un CV, il est possible de partir d'un ancien CV. Pour ce faire, l'utilisateur sélectionne « Créer à partir d'un ancien CV » et choisit le CV qu'il veut dupliquer. Nous faisons alors une requête afin de recopier les enregistrements dans le nouveau CV et ce pour chacune des informations (diplôme, compétence, expérience, langue et loisir).

Voici un exemple concernant la duplication des diplômes :

Code visual basic

```
INSERT INTO REFERENCER_DIPLOME ( NUM_DIPLOME, NUM_INDIVIDU, NUM_CV,
DATE_OBTENTION, LIEU_OBTENTION, DESCRIPTION_DIPLOME )
SELECT REFERENCER_DIPLOME.NUM_DIPLOME, " & Me.OpenArgs & ", " & max & ",
REFERENCER_DIPLOME.DATE_OBTENTION, REFERENCER_DIPLOME.LIEU_OBTENTION,
REFERENCER_DIPLOME.DESCRIPTION_DIPLOME
FROM REFERENCER_DIPLOME
WHERE (((REFERENCER_DIPLOME.NUM_CV)=" & Me.txtNumCV & ") AND
((REFERENCER_DIPLOME.NUM_INDIVIDU)=" & Me.OpenArgs & "));
```

Nous insérons un enregistrement en reprenant un ancien. Nous forçons juste le numéro de CV à une nouvelle valeur (max).

2. Formulaires de suppression

Nous donnons la possibilité à l'utilisateur de supprimer les informations concernant une personne car cet individu a le droit de l'exiger. Nous supprimons seulement le CV qui sera sélectionné. Les enregistrements qui sont liés à ce numéro de CV seront automatiquement supprimés (dans les tables REFERENCER_DIPLOME, EXPERIENCE…).

Code visual basic

```
Private Sub btnSupprimer_Click()
   If Me.txtNumCV <> "" And Me.txtNumIndividu <> "" Then
      DoCmd.RunSQL ("DELETE * FROM CV WHERE NUM_INDIVIDU=" &
Me.txtNumIndividu.Value & " AND NUM_CV=" & Me.txtNumCV.Value & ";")
      DoCmd.Close
   End If
End Sub
```

Afin de supprimer un individu, il faut d'abord supprimer tous ses CVs puis supprimer ses informations personnelles. En effet, un individu peut exister sans posséder de CV (cf. MLD). Les deux requêtes utilisées sont les suivantes :

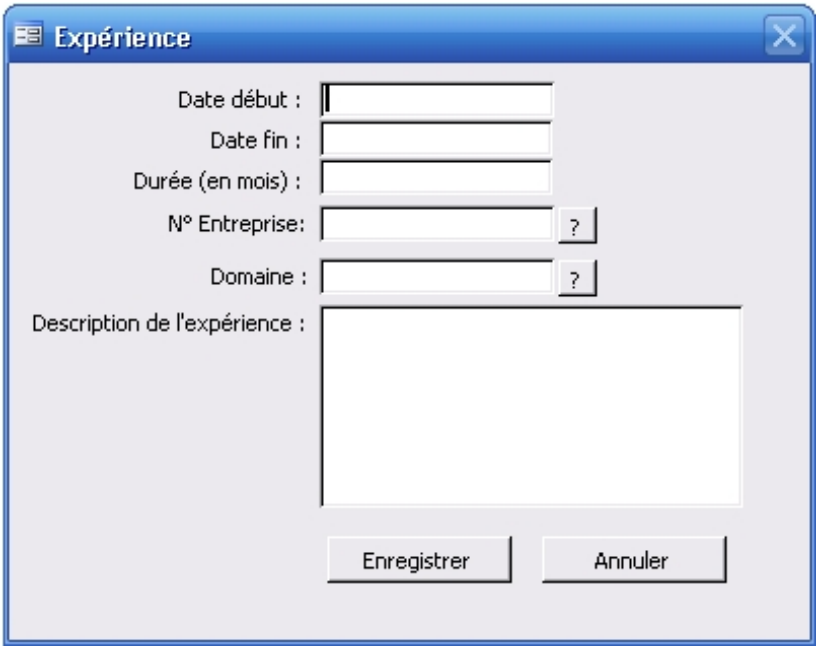
Requête SQL

```
"DELETE * FROM CV WHERE NUM_INDIVIDU=" & Me.txtNumIndividu.Value & ";"
"DELETE * FROM INDIVIDU WHERE NUM_INDIVIDU=" & Me.txtNumIndividu.Value & ";"
```

d) Autres formulaires

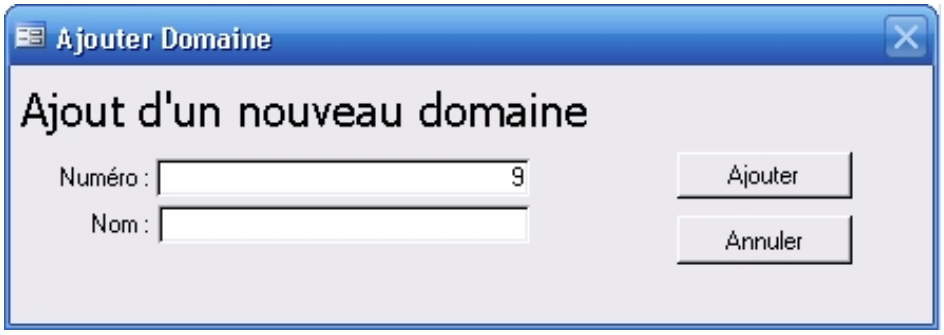
Nous allons mettre ici des exemples de type de formulaires que nous n'avons pas encore vu, à savoir les formulaires d'éditions ou d'ajout :

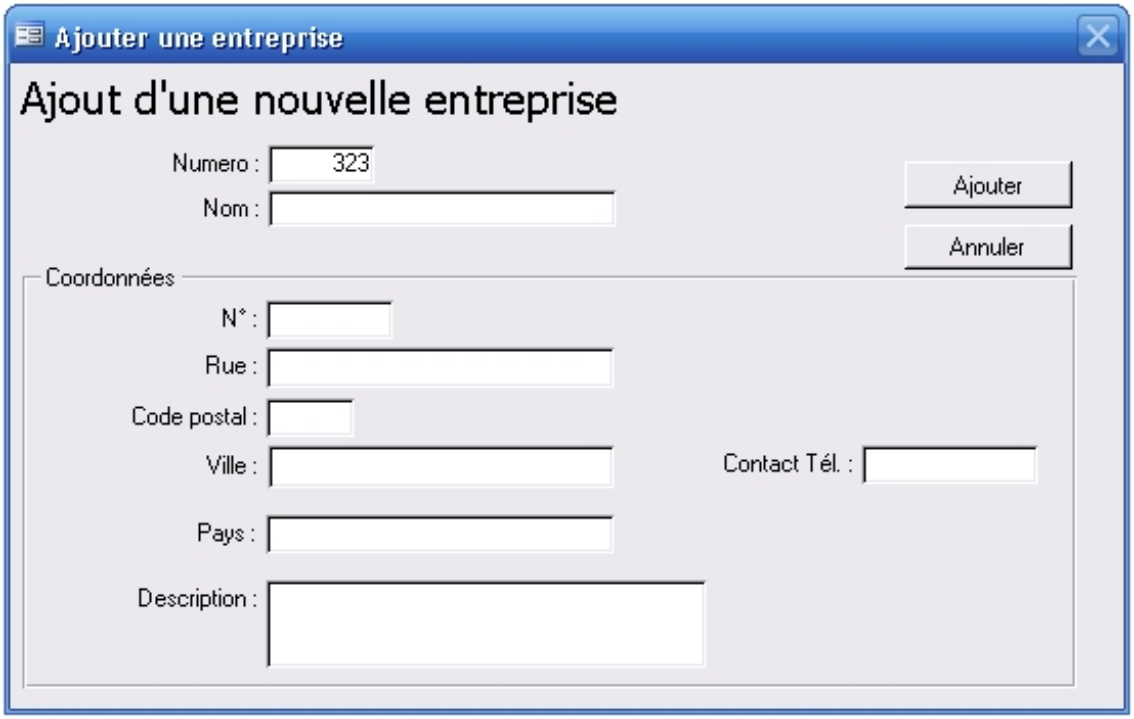
Ci-dessous les formulaires d'édition (ajout/modification) de diplômes et d'expérience du CV :



A savoir que dans ce dernier formulaire les 3 premiers champs sont « complémentaires » : la date de début est obligatoire puis si l'on met la durée ou la date de fin, le calcul du troisième champ resté vide est effectué suivant la valeur des deux premiers.

Ci-dessous les formulaires d'ajout d'un nouveau domaine ou d'une nouvelle entreprise :





Les numéros de ces ajouts sont mis automatiquement.

Conclusion

En conclusion, ce projet nous a tout d'abord permis de découvrir le système de gestion de base de données Microsoft Access. Nous avons pu découvrir comment concevoir une application permettant de gérer une base de données dans l'objectif de simuler un système proche de la réalité.

Ensuite, nous avons pu consolider nos bases acquises durant ce semestre en BD40 dans la construction de requêtes SQL complexes ainsi que la programmation en Visual Basic.

Enfin, ce travail nous a permis de progresser dans le travail de groupe. En effet, avec une bonne répartition du travail nous avons pu développer chacun de notre coté des parties de la CV Thèque.

Pour finir, nous pouvons aborder le sujet des améliorations possibles : nous pourrions rendre notre application utilisable par n'importe qui voulant ajouter son CV dans la base, consulter les offres, en ajouter… Dans ce cas là, il faudrait faire une procédure d'identification (login et mot de passe) pour les différents acteurs, faire également à ce moment là une partie administrateur avec plus de possibilités. Puis nous pourrions renforcer la relation entre CV et offres : en effet, on pourrait déposer une offre et les CVs respectant le profil recherché au mieux seraient triés par ordre de pertinence. Enfin, pour le moment nos informations sur les profils ne sont qu'une addition de critères (l'individu doit avoir telle compétence **ET** telle autre, etc…). Nous pourrions rajouter une option pour gérer le **OU**. Ainsi on pourrait également faire des recherches sur telle domaine **OU** bien sur un autre.