



End of Study Thesis

www.supinfo.com

Copyright SUPINFO. All rights reserved

Last name : BOISVILLIERS

First Name : Gaël

Campus Booster ID : 40611

Software standardization managing a pineapple's culture

Acknowledgments

First, I want to thank Mr. Patric Fournier, who accepted me as an intern at CIRAD on the Oumpapa project which gave me an opportunity to know a completely unknown area to me: the pineapple cultivation.

I also wish to thank the Basin Plat Staff and the other interns for various aid and advice and the warm welcome.

My last words are devoted to my Classmates who have maintained a collegial work atmosphere during the internship despite the distance.

Resume

My job in the Oumpapa project focuses on the transcript of a VB application in VB.NET. The application, which must be transcribed, is a model of a pineapple farm that allows the management of its culture. The first application being developed with procedural's methods, it was necessary to redesign the application with an object-oriented approach to move forward. This thesis outlines the points that have enabled me to lead this project during my end of study internship. In the first part, we will present the different aspects of data processing used in this area through the presentation of the company.

In a second step, we will present the field of my project.

Finally, we will look at the useful working methods that allow the work of a programmer to be brought again by another one.

Table Of Contents

1. INTRODUCTION.....	7
2. PRESENTATION OF THE COMPAGNY	8
2.1. COMPAGNY	8
2.2. IMPORTANT FIGURES	8
2.3. FUNCTIONS, STRUCTURE & ORGANIZATION.....	9
2.3.1. <i>Compagny functions</i>	9
2.3.2. <i>Flow chart</i>	10
2.3.3. <i>Team</i>	11
2.4. INFORMATION SYSTEM & TECHNICAL ENVIRONMENT	11
3. ENUNCIATION OF THE ISSUES.....	12
4. DESCRIBE THE CONTEXT	13
5. EXPLAINING THE CHOICE OF THE SUBJECT	16
6. PRESENTATION OF THE USUAL METHODS USED TO FACE THE PROBLEM	17
6.1. WORKING METHOD TO SOLVE THE PROBLEM.....	17
6.2. DIFFERENCE BETWEEN VB6 & VB.NET	19
6.3. OBJECT APPROACH	22
7. EXPLANATION OF THE APPROACHES CHOSEN TO RESOLVE THE PROBLEM	24
7.1. INTRODUCTION TO DOTNET	24
7.2. VB.NET.....	27
7.3. DATASET	29
7.4. DATAREADER	40
7.5. THE STREAMS	42
7.6. IMPRESSION STATEMENTS	44
8. DEMONSTRATION OF THE APPROACH'S ORIGINALITY	49
8.1. PROGRAMMING RULE.....	49
8.2. CODE OPTIMIZATION.....	52
8.3. POSITIONING	56

9. CONDITIONS OF APPLICATION OF THE SUGGESTED APPROACH	62
10. REFLECTING ON THE MANAGEMENT OF THE THESIS	63
11. CONCLUSION.....	64
12. READING REFERENCE	65

1. Introduction

Nowadays, data processing are present in many fields. Agriculture is one of the areas in which, you won't expect to meet it. However the Agricultural Research (CIRAD here) has developed an application that can assist the farmer in the management of the pineapple culture. This application allows him to optimize the management of his operations, whether in terms of planting, plants processing and even harvesting, its primary purpose is obviously to help the farmer to optimize his production of pineapples.

2. Presentation of the compagny

2.1. Compagny

My end of study's internship was made with CIRAD.

CIRAD for « Centre de coopération Internationale en Recherche Agronomique pour le Développement ». He works with french overseas and more then 40 south countries. The Reunion Island department is the largest outside the french territory. He intends to propose innovative solutions to problems that face the agriculture and the region to achieve their sustainable development, and in advancing the knowledge in various fields of science.

2.2. Important figures

CIRAD is a commercial government compagny in industry (**EPIC**). It employs 1,820 people, including 1,050 cadres. Its operating budget amounts to 200 million euros. It consists in three departments and 59 units.

An EPIC is, in France, a public company, incorporated in a public person form that's the work is to manage a public service activity.

The EPIC were created to respond to the need that can't be conducted properly by a private company that have to work with the competition, this concept is being challenged by liberal thinking that leads to their progressive privatization.

Some public institutions act jointly public administrative missions and public industrial and commercial missions.

The EPIC are mostly subject to private law although they enjoy, as a legal entity of public law, some public law privileges.

2.3. Functions, Structure & Organization

2.3.1. Company functions

In Reunion Island, CIRAD is structured, since 1 January 2007, by three department of research:

- [Production quality of the agricultural and tropical foods \(KAPPA Department\)](#)
- [Environmental risk, agriculture and integrated management resources \(REAGIR Department\)](#)
- [Plant protection \(3P Department\)](#)

Let's see more in detail the role of the department where I made my internship, which means the KAPPA department:

The KAPPA department has in prime objective to build technical routes of production, from the producer to the consumer, ensuring the traceability and quality of the product (health, nutritional and sensory...).

To this end, the mechanisms of processing and preservation of the product's quality are studied in the clusters, which are incorporated in their entirety. The implemented research actions are conducted in collaboration with local and regional partners.

Its work focuses on three areas:

Area 1: The optimal management of livestock production system. To help the industry to overcome them, while adapting to European standards, the KAPPA department works, as part of a first line of research, innovative tools to conduct bovine holdings.

Area 2: Fruit and horticultural integrated production. Works in this area aimed on a better understanding of the plant to the operation stage, in order to control the production quality.

Area 3: The development of the food quality product. This research is the development tool of characterization and improvement of the agricultural quality products whether an animal or vegetable origin.

2.3.2. Flow chart

Conseil scientifique	Conseil d'administration	Comité d'éthique
Président : Bernard Chevassus-au-Louis	Président : Patrice Debré	Président : Alain Ruellan
Direction générale		
Direction de la recherche et de la stratégie		
Départements de recherche		
Secrétariat général		
Direction de l'innovation et de la communication		
Direction des relations européennes et internationales		
Directions régionales		
France (Guadeloupe, Guyanne, Ile-de-France, Languedoc-Roussillon, Martinique, Nouvelle-Calédonie, Réunion)		
Etranger (Afrique, Amérique latine, Asie)		

2.3.3. Team

CIRAD activities are coordinated by a regional administration. It has 175 permanent employees, including 45 researchers and 7 directors, 123 technicians and employees. To these employees add twenty civil volunteer engineers to technical assistance (VCAT) and an equal number of doctoral students and trainees. If we add to these figures the staff received, it is now more than 260 peoples working in the premises of CIRAD.

2.4. Information system & Technical environment

As part of its mission, CIRAD develops equipments, processes and softwares. Most of these products are part of a tool range including benefits expertise and training. Professionals are responsible of the distribution and the promotion.

Treatment and processing materials of agricultural products, analysis devices of the food quality products, IPM tools...

- SCT
- H2SD
- RITA
- ...

Softwares

Pest identification softwares, plant species, biometric and static software, modeling software for plant growth, database...

- Plant modeling software
- Olympe : allow to create an operating typologie
- R : allow statistical analysis

3. Enunciation of the issues

How do you make an application modelling an unknown context?

How to transcribe a program in another langage?

Why will I use an objects technology?

How to approach an object problem?

Modularity (plug-in), reusability, scalability, using components librairies, how the object approach responds to these challenges?

4. Describe the context

The management of a pineapple farm can be modeled in the following manner:

See "A simplified modeling of a pineapple operating" below.

This simplified model still deserves a few comments.

As we can see on this model, the parcel and the bands are the central points of a farm.

According to the definition of IGN and National Education on geographic information, a cadastral parcel is a land's portion belonging to a single owner with some individuality because of the layout given to the property. Each parcel is composed of several bands which are composed of one or more ridges.



The TIF (Floral Induction Processing) is a treatment that allows an artificial flowering. From this TIF, it is possible to predict a harvest thanks to the treatment and weather data.

The user must also provide the technical route that he will apply to his parcel. The technical route is like an interventions data sheet that will be done on the parcel.

Discharge harvesting (plant material remaining after the pineapple harvesting) is also on the modeling because they are useful for the next planting.

The weather data must be part of the modeling because the records are essential to the forecast calculations dates of interventions.

The earlier application

After a familiarization with the previous version of the application, we noticed a few problems like:

- Procedure programming
- The user manipulate the database directly
- Redundancy code
- Imports the data temperatures
- ...

By definition, this type of programming using functions is called “procedure”.

These procedures may be like:

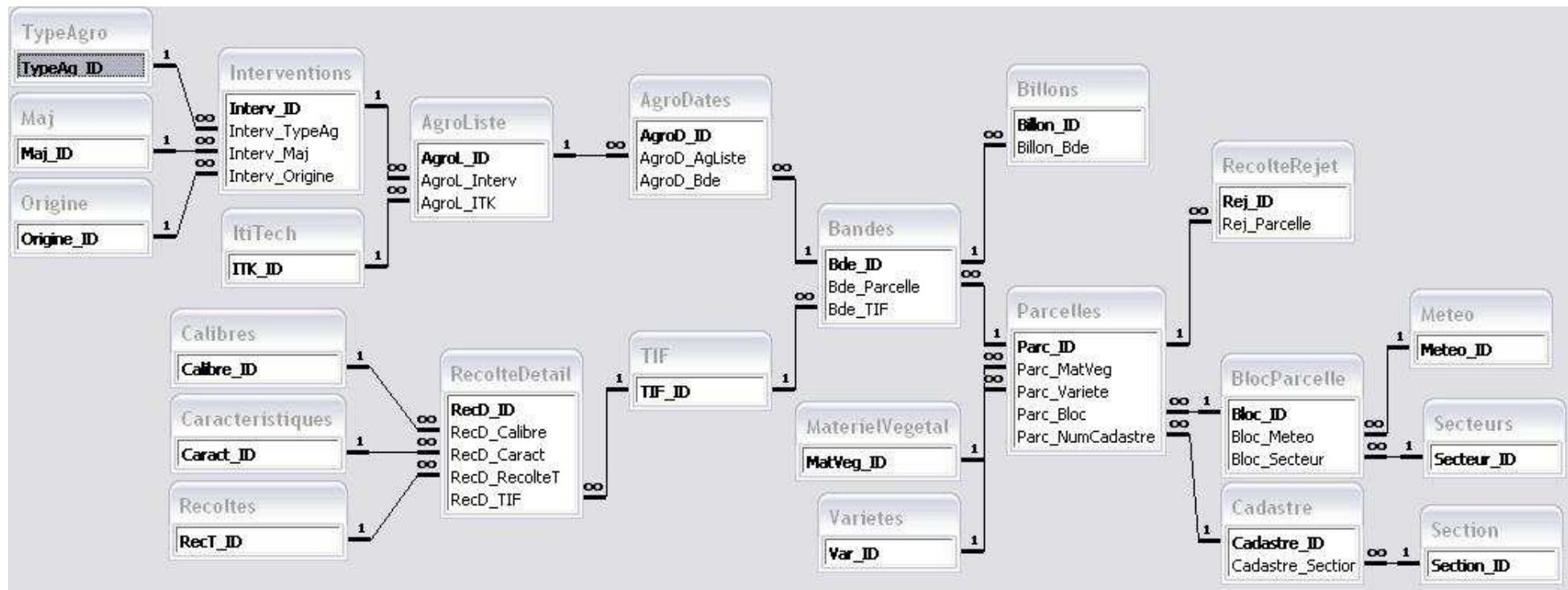
- Routine
- Method or function (sub or function in VB)

They contain a portion of code that performs a specific function. The program advantages are mostly:

- Code reuse at various locations
- An algorithm implementing most easily identifiable

We can see in the old version that each form has been added in accordance with the needs of the moment. We can also see the redundancy of the code or some variables.

My project was to transcribe the application in VB.NET while revising the object approach.



A simplified modeling of a pineapple operating

5. Explaining the choice of the subject

The internship subject seemed to be good challenge for a computer.

The project being a transcript of an application to the VB.NET, it seems appropriate to deal the development of the project based on this language.

The choice of language is a complicated challenge because it was unknown to me.

Also expected to return a full documentation of what has been achieved during my internship, and any researches intended for the project have been used to create this memory.

6. Presentation of the usual methods used to face the problem

6.1. Working method to solve the problem

In this case, my main problem was the fact that to not know the context, nor the language to transcribe, nor the final transcript one. I would have to establish a working method to complete this project.

The first thing to do in my opinion was to learn a lot about the subject that I have to model. Indeed, it is essential to have some knowledge about the subject that I have to deal with: like the vocabulary to use in order to facilitate communication.

So I decided to retrieve the documentation mainly in technical data form. I was also able to get some agricultural engineering lesson (documentation) with another intern.

The next step that must be carried out is the verification of specifications with the employer. In my case, the specification is a description of the current version of the software, so I decided to learn all the power of the software.

Having identified the context, we can now achieve a modeling database, which will be adapted to the new software. There are several methods of modeling (MERISE, UML, JACKSON, etc.).

- In this phase of modeling, it is also necessary to learn with the previous programmer the use of tables and fields he used. It is possible to obtain information by:
 - Oral: there can be data loss on each side
 - Written: the audited use a response plan that you made in order to obtain informations that you are interest in
- It is necessary to finalize the database in the presence of all parties so you don't have to change any time

Once the database is established, before beginning development, it must be better to cut the project into several modules to establish a working plan.

During the development phase, it is better to write functions in the Algorythmic form (or pseudo language) so it will be easier to transcribe whatever the final language is.

To this developing stage, there is the coding stage, which consist to translate the pseudo language (in our case the translation is done in VB.NET) and the testing stage (which can validate the functioning of the application).

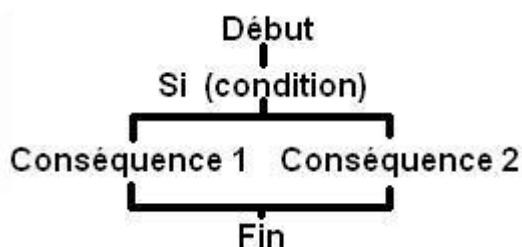
As mentioned earlier, the algorithm is a stage in the development. By definition, the algorithm is a sequence of actions and tests that define the behavior of the entities.

An algorithm can be defined by three scenarios:

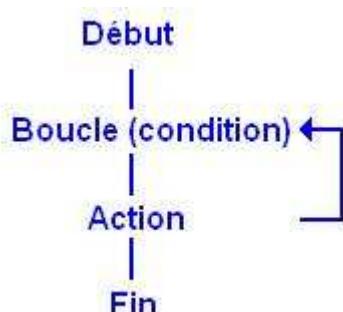
- ***Sequence***: performs lines of actions one by one

Start → Action 1 → Action 2 → Action 3 → End

- ***The choice***: execute an action depending on the condition

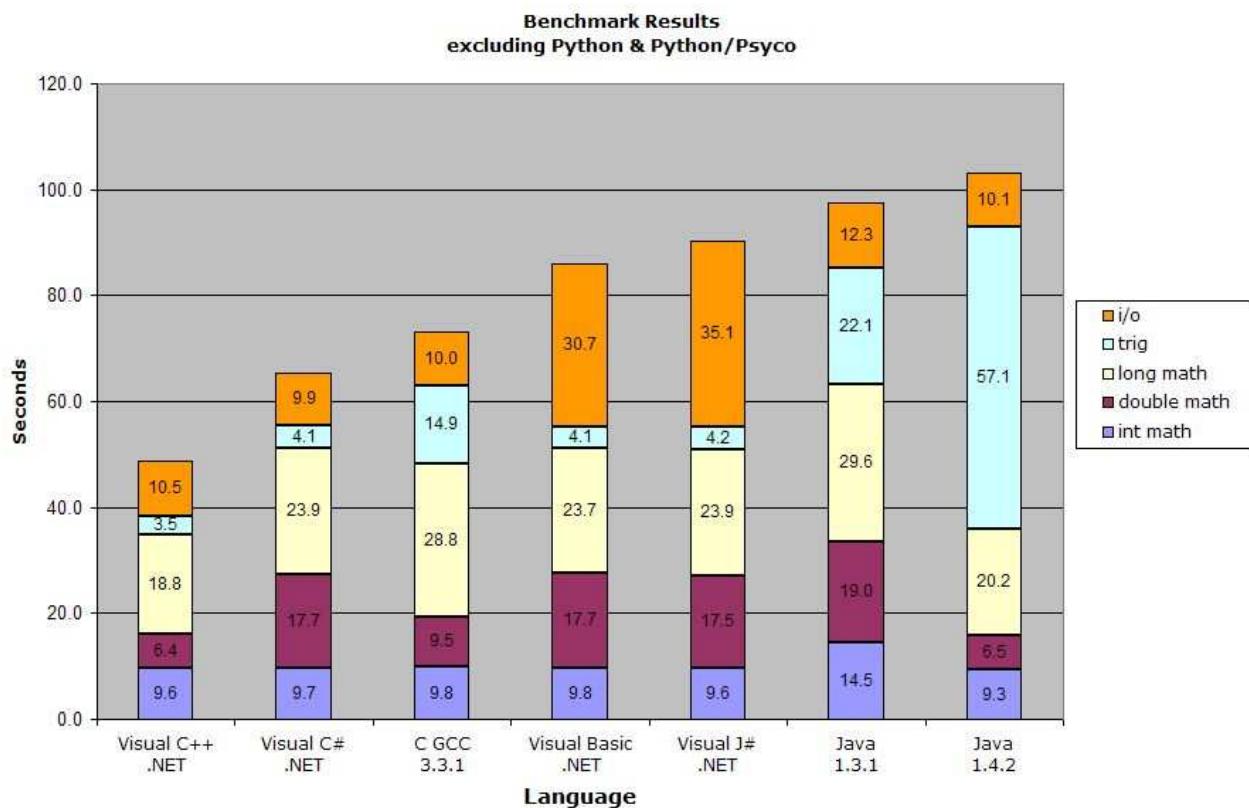


- ***Repetition***: performs a sequence of actions as many times as require the condition



6.2. Difference between VB6 & VB.NET

According to a study published by the site OsNews.com on the executing performance on Windows of several programming languages, we can see that the .NET languages are mostly well ranked.



Visual Basic.NET presents various important innovations according to the previous versions of Visual Basic. VB.NET supports inheritance, constructors, polymorphism, manufacturer overloading, structured exceptions, free models of thread, as well as many other features. There are new rapid features of development available like the XML designer, the server explorer and the web forms designer. With this release, VBScript offers a complete feature of Visual Basic.

The Framework .NET provides a variety of tools optimized for a rapid processing, but also to reduce the number of line code. We can take the example the search methods or sorting. VB.NET has also made some changes on the handling controls. Controls no longer possess default properties. Indeed, VB6 allowed to recover the value of a control with just his name, VB.NET requires defining the property.

VB.NET has made some changes:

- Use ADO.NET for data access
- Use GDI + for managing graphics (GDI for VB6)
 - no longer supports some graphical objects because of the new object approach
 - Its Clipboard supports many more formats
- The MultiThreading is now possible
- Manages polymorphism, surcharge, inheritance ... (fully supports the oriented object)
- The brackets are required to call a method, no more confusion with a variable (easy reading)
- The Return keyword is added to this version of VB (service can now return a variable)
- Changes of VB6 components (in another form)
 - groups of controls are now GroupBox or panels
 - text controls are now labels or TextBox
 - ...
- Doesn't support dynamic exchange of data
- Variables must be declared to be used
- The basis type for a variable is now Object (Variant for VB6)
- The concept of the collection is now widely used because it allows us to dynamically change the number of elements (Collection, ArrayList, ListBox, etc.)
- Integer goes into 32 bits, 64 bits in Long
- The use of Boolean can be made only by True or False (0 and -1 for VB6)
- Dim makes variables in the same type

Ex: Dim a, b, c as Integer

	VB6	VB.NET
A	Variant	Integer
B	Variant	Integer
C	Integer	Integer

- The String is no longer a fixed length
- The scope of a variable can now be attributed to a block

Typically, the framework .NET seems to slow down the speed of the code compared to vb6. However, in VB.NET, the reasoning is different. It is therefore necessary to optimize the use of tools offered by the Framework. The use of classes and methods makes it shorter to develop.

To conclude, we can say that for a transcript in oriented object, the application should be considered as a whole while optimizing the code using functions offered by the framework .NET.

6.3. Object approach

The challenge is to rethink the application to make it correspond to the object-oriented approach to optimize the use of VB.NET.

Programming with an object approach means we will mainly use classes for development. It is quite difficult to think object for a programmer who never done it, so I resumed a few rules of good programming from various sites and work on the object-oriented subject.

By definition, an object is an instance of a class. It includes:

- a name
- a behavior
- a state

The recorded advantages of the object-oriented are:

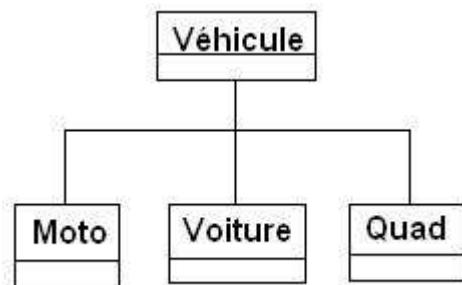
- reuse code
- the complexity reduction
- Encapsulation (implementation not shown)

ex: a driver doesn't need to know the process to start the engine of a car, he has just to turn the key.

The class is used to model objects from real life. This model corresponds to a compilation of all the information characterizing the object. Take a simple example:

- Object: a car
- Property: Number of wheels, Constructor, Color, Range ...
- Behavior: Engine Start, Engine Off ...

The OOP offers the possibility of creating abstract classes. These classes are used to define common characteristics between objects. Let us continue on the example of the car:



We can see in this example that car inherits characteristics from the abstract class Vehicle like Motorcycle and Quad (Wheel, Engine, and Chair). An abstract class can not be initiated, it serves only as a basic structure for the derived classes.

The presentation of the object-oriented being made, we must now how to define the objects of our project:

- We must identify all the objects
- Determine their behavior
- Determine the relationships they have with other objects

Once the object identification is made, it is possible to establish a suitable database, and even algorithms operation.

7. Explanation of the approaches chosen to resolve the problem

7.1. Introduction to dotnet

The subject of my end of study internship was to transcribe an application programmed in VB / Access on a Microsoft .NET platform. The chosen language for the transcription was the VB.NET. It was chosen to facilitate the updating. Visual Basic is a tool developed by Microsoft to easily develop applications running Microsoft Windows.

Visual Basic is a visual tool that allow us to create without programming concept a graphical user interface (**GUI**) by providing with the mouse graphic elements (buttons, images, text fields, menus, ...).

The advantage of this language is to be able to combine to the interface elements portions of code associated with events (mouse click, a key press ...). This is why, Visual Basic uses a small programming language derived from **BASIC** (Beginners All-Symbolic Instruction Code, or code instructions multi symbolic). The scripting language used by Visual Basic is rightly named **VBScript**, and it is a subset of Visual Basic. In addition, this language is used in many other applications as Microsoft Visual Basic (Access, Excel, Internet Explorer, etc.)

To create a soft, we just have to create a graphical interface with the library, then to program events related to interface elements.

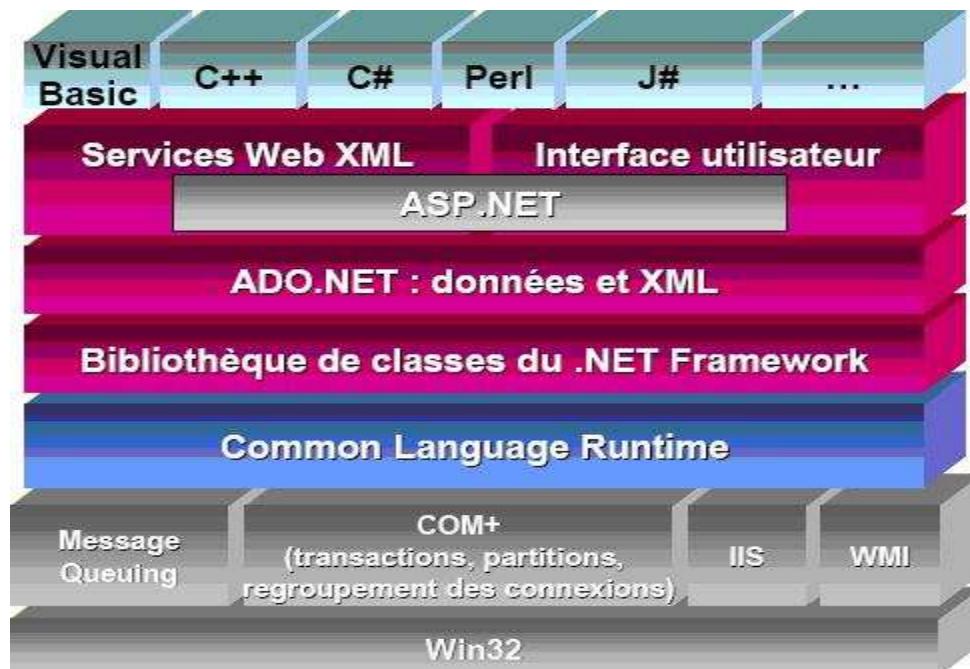
VB.NET is an update of VB language that uses object-oriented the .NET Framework.

The object language incorporates all the capabilities of procedural languages, the object management, the Windows graphical interface.

VB.NET uses the .NET Framework which uses the common language runtime (this managed environment provides security services or even memory management as "garbage collector") and especially ADO.NET which my internship is based.

The .NET Framework (class library) is the layer that links Windows to the VB application. The major drawback of the Framework is its slowness; however, it brings some advantages:

- It's install once
- Can be manipulated by several languages
- Provides a wide variety of tool

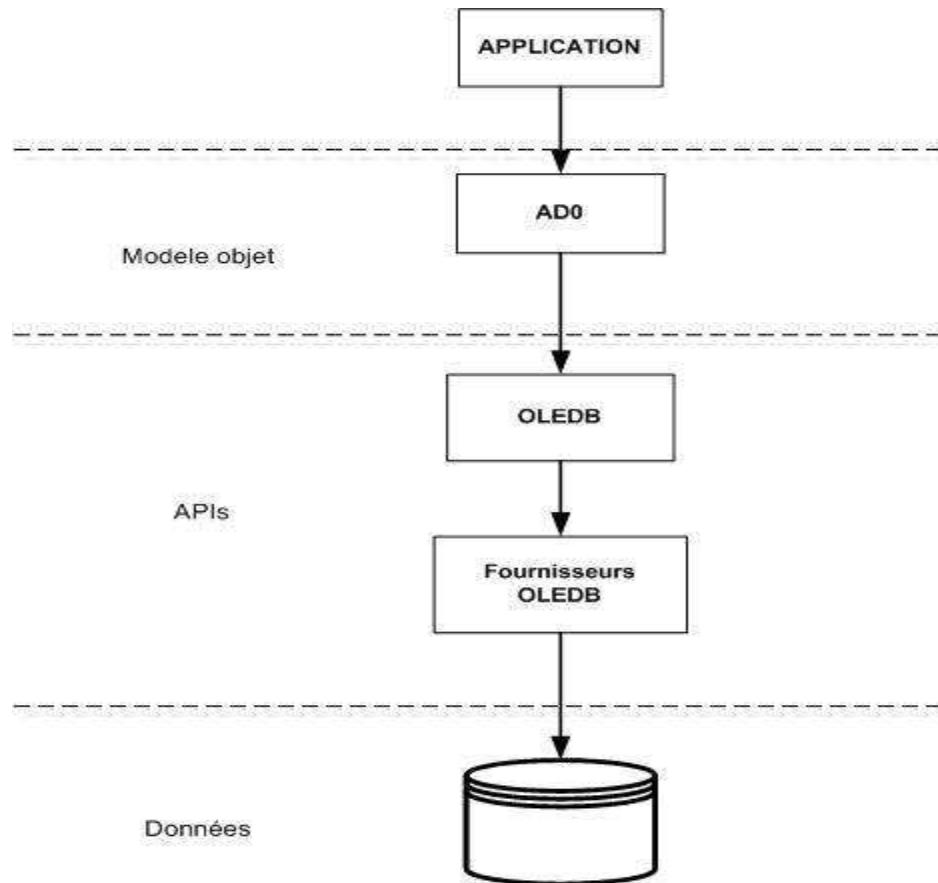


ADO.NET is an access method to the data source.

- **ADO** (Activex Object Database) is the layer access to the database
- The extention .NET means that the solution is managed and managed by the **CLR**
- ADO.NET has a unique language allowing him to manipulate all database

To access the database, it is necessary to load the specific drivers references to the database and to import namespaces. The database used in my project being **Access**, so I used the managed driver **OLE DB Provider**.

To access to the database, you must establish a connection to it. Each connection must have a **ConnectionString** which define the parameters of the database (path of the database, ID, password, etc.). The ConnectionString are different according to the database. You might ask how to write it. The site [Http://www.connectionstring.com](http://www.connectionstring.com) has a wide list of parameters connections.



We must access to a database according a precise algorithm:

1. Open the database: the Connection object has a method to perform this action (***Open***)
2. Manipulation of the database
 - a. Recover the result: ***SELECT***
 - DataReader (***Read Only***): connected mode, we use a ***Command*** object
 - DataSet: disconnected mode, we use a ***DataAdapter*** object

It has the structure of a local database
 - b. Direct Manipulation: ***UPDATE, INSERT, DELETE ...***
 - We use a Command object, queries are executed by the property ***ExecuteNonQuery***
3. Close the database (***Close***)

7.2. VB.NET

Even in object-oriented, it is necessary to split the problem into several parts. In the transcription, these parts are written in a function form.

In VB, functions can be written in two forms:

- "**Sub**" after treatment, the procedure don't return a direct value

```
Sub AfficherNom()
    monTextBox.Text = "Hello"
End Sub
```

- "**Function**" after treatment, the procedure *returns a value*.

This type of procedure needs to be typed or returns an object type

```
Function MonMessage() As String
    Return "Hello"
End Function
```

Passing parameters to a procedure can be done in two ways:

- By Value (**ByVal**): this method passes the contents of the variable as a parameter.
At the call of the procedure, the newly created variable use another memory space, plus the value passed by the calling method is retained.

```
Function AfficherMessage(ByVal msg As String) As String
    Return msg
End Function
```

- By reference (**ByRef**): this method passes the address (physical location in memory) of the variable as a parameter so the calling method wil use the modified variable.

```
Function AfficherMessage(ByRef msg As String) As String
    Return msg
End Function
```

All parameter passages which the method isn't specified will be made by value, but for a better clarity and understanding of the code, it's advise to specify the methods of passage.

It is also possible to declare a parameter as optional with the ***Optional*** keyword. This means that the argument isn't required to call the procedure but requires a variable default.

```
Function AfficherMessage(Optional ByVal msg As String = "Hello") As String
    Return msg
End Function
```

Finally VB can pass a list of parameters of the same type, therefore the number isn't known. In this case, we can use the option ***ParamArray***. The transition is always made in value mode and must be placed at the end of the statement.

```
Function Total(ByVal Depart As Integer, ByVal ParamArray Donnees() As Integer)
As Integer
    For Index As Integer = 0 To Donnees.Length - 1
        Depart += Donnees(Index)
    Next

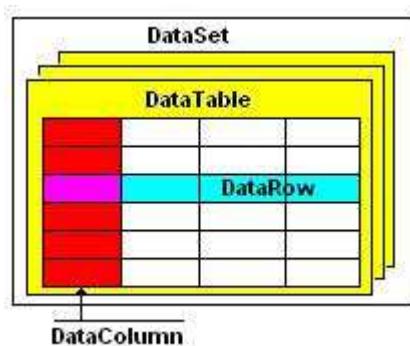
    Return Depart
End Function
```

7.3. DataSet

As mentioned previously, the main tool that I use to work on the records in the database is the DataSet.

The DataSet is a representation of the data in memory. The DataSet must be loaded from the database to work offline. After the update of the DataSet, it will be possible to update the database from this new data source.

The DataSet represents the database in memory. The DataTable could be compare as a table in memory. The DataTable (such as the DataView) is the data source for some controls (grid, listView, ComboBox, etc.). The DataTable and DataView have sort, filter and search methods. It would be interesting to master the manipulation of these objects.

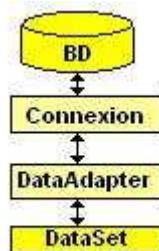


Filling DataGridView example:

```
Dim Matable As New DataTable
'Enregistre la liste dans une table virtuelle
Matable = Dtset.Tables("T_Secteurs")

monDataGridView.DataSource = Nothing
monDataGridView.DataMember = Nothing
monDataGridView.DataSource = Matable
```

To fill a DataSet, it is necessary to have:



```

'Paramétrage de la chaîne de connection
Connection.ConnectionString = My.Settings.maConnection
'ouverture de la connection
Connection.Open()
'définition de la requête (on sélectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'définition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
'remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()
  
```

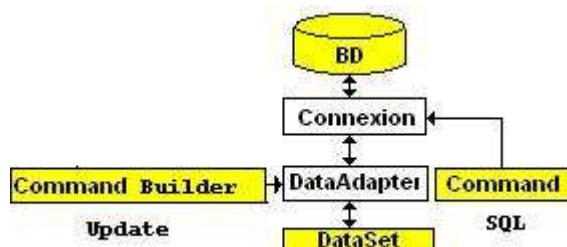
Based on these code and diagram, we can see that for fillinf a dataset, we must:

- Establish a connection to the database from an connection oblect
- A DataAdapter object (OleDbDataAdapter for Access database), which will fill the DataSet by using the Fill method

The syntax to get a data is quite simple:

```

'Utilisation du dataset - récupère le nom du secteur
DtSet.Tables("T_Secteurs").Rows(0).Item("Secteur_Nom")
  
```



```
Dim dv As New DataView(DtSet.Tables("T_Secteurs"))
dv.Sort = "Secteur_Nom"
Dim Existe As Integer = dv.Find(monSecteur.Secteur_Nom)
If Existe = -1 Then
    Dim NewLigne As DataRow

        With DtSet
            ' Création de la nouvelle ligne
            NewLigne = DtSet.Tables("T_Secteurs").NewRow
            'affectation des valeurs
            NewLigne("Secteur_Nom")= monSecteur.Secteur_Nom
            ' Ajout de la ligne à la table
            .Tables("T_Secteurs").Rows.Add(NewLigne)

            Connection.Open()
            ' Création CommandBuilder
            Dim CmdBuild As OleDbCommandBuilder
            CmdBuild = New OleDb.OleDbCommandBuilder(AdapTest)
            AdapTest.InsertCommand = CmdBuild.GetInsertCommand
            AdapTest.Update(DtSet, "T_Secteurs")
            Connection.Close()

        End With
Else
    MessageBox.Show("Ce secteur existe déjà!!!")
End If
```

We can see from that the CommandBuilder object is required to update the database.

This update is possible only if the data source correspond to tables in the database. We must think about dealing with errors exception.

To finish the presentation of this object, it is possible to create and fill it manually:

- Creation of the DataTable
- Creation of the columns
- Added line
- ...

We can see examples of simplified code to handle the database:

- **Reading a table**

```
'Librairies à utiliser
Imports System.Data
Imports System.Data.OleDb

Dim Connection As New OleDbConnection
Dim AdapTest As OleDb.OleDbDataAdapter
Dim DtSet As DataSet
Dim Sql As String

'Paramétrage de la chaîne de connection
Connection.ConnectionString = My.Settings.maConnection
'ouverture de la connection
Connection.Open()
'définition de la requête (on sélectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'définition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
'remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()

'Utilisation du dataset - récupère le nom du secteur
DtSet.Tables("T_Secteurs").Rows(0).Item("Secteur_Nom")
```

- Adding a row in the table

```

'Paramétrage de la chaîne de connection
Connection.ConnectionString = My.Settings.maConnection
'ouverture de la connection
Connection.Open()
'définition de la requête (on sélectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'définition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
'remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()

Dim dv As New DataView(DtSet.Tables("T_Secteurs"))
dv.Sort = "Secteur_Nom"
Dim Existe As Integer = dv.Find(monSecteur.Secteur_Nom)
If Existe = -1 Then
Dim NewLigne As DataRow

    With DtSet
        'Création de la nouvelle ligne
        NewLigne = DtSet.Tables("T_Secteurs").NewRow
        'affectation des valeurs
        NewLigne("Secteur_Nom")= monSecteur.Secteur_Nom
        'Ajout de la ligne à la table
        .Tables("T_Secteurs").Rows.Add(NewLigne)

        Connection.Open()
        'Création CommandBuilder
        '(génère automatiquement l'update entre le dataSet et la base de
        donnée
        Dim CmdBuild As OleDbCommandBuilder
        CmdBuild = New OleDb.OleDbCommandBuilder(AdapTest)
        AdapTest.InsertCommand = CmdBuild.GetInsertCommand
        AdapTest.Update(DtSet, "T_Secteurs")
        Connection.Close()

    End With
Else
    MessageBox.Show("Ce secteur existe déjà!!!")
End If

```

- **Editing a row in the table**

```
'Parametrag de la chaine de connection
Connection.ConnectionString = My.Settings.maConnection
'ouverture de la connection
Connection.Open()
'definition de la requete (on selectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'definition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
' remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()

'recuperer la clé de la ligne sélectionné
Dim CleUnik As String
CleUnik = monSecteur.Secteur_Num
'definition de notre table
Dim Matable As DataTable
Matable = DtSet.Tables("T_Secteurs")
'recherche la ligne a modifier dans notre table
Dim LaLigne As DataRow()
LaLigne = Matable.Select("Secteur_Num = " & CleUnik)
'affecte les modifications
LaLigne(0)("Secteur_Nom") = monSecteur.Secteur_Nom
'Modification dans la base

Connection.Open()
' Cr éation CommandBuilder
'(genere automatiquement l'update entre le dataSet et la base de donnée
Dim CmdBuild As OleDbCommandBuilder
CmdBuild = New OleDb.OleDbCommandBuilder(Adapter)
Adapter.UpdateCommand = CmdBuild.GetUpdateCommand()
Adapter.Update(DtSet, table)
Connection.Close()
```

- **Deleting an entry in the table**

```

'Paramétrage de la chaîne de connection
Connection.ConnectionString = My.Settings.maConnection
'ouverture de la connection
Connection.Open()
'définition de la requête (on sélectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'définition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
'remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()

'recuperer la clé de la ligne sélectionnée
Dim CleUnik As String
CleUnik = monSecteur.Secteur_Num

//recherche de la ligne à supprimer dans le dataset
Dim Ligne As DataRow()
Ligne = DtSet.Tables("T_Secteurs").Select("Secteur_Num = " & CleUnik)
'la supprime
Ligne(0).Delete()

////Mise à jour de la base
'ouverture de la connection
Connection.Open()
'Création CommandBuilder
'(qui génère automatiquement l'update dans la base de données)
Dim CmdBuild As OleDbCommandBuilder
CmdBuild = New OleDb.OleDbCommandBuilder(AdapTest)
'envoie de la commande de suppression au dataAdapter
AdapTest.DeleteCommand = CmdBuild.GetDeleteCommand()
'Mise à jour avec la base
AdapTest.Update(DtSet, "T_Secteurs")
'fermeture de la connection
Connection.Close()

```

It is also possible to fill a grid control (DataGridView in VB) without code line.

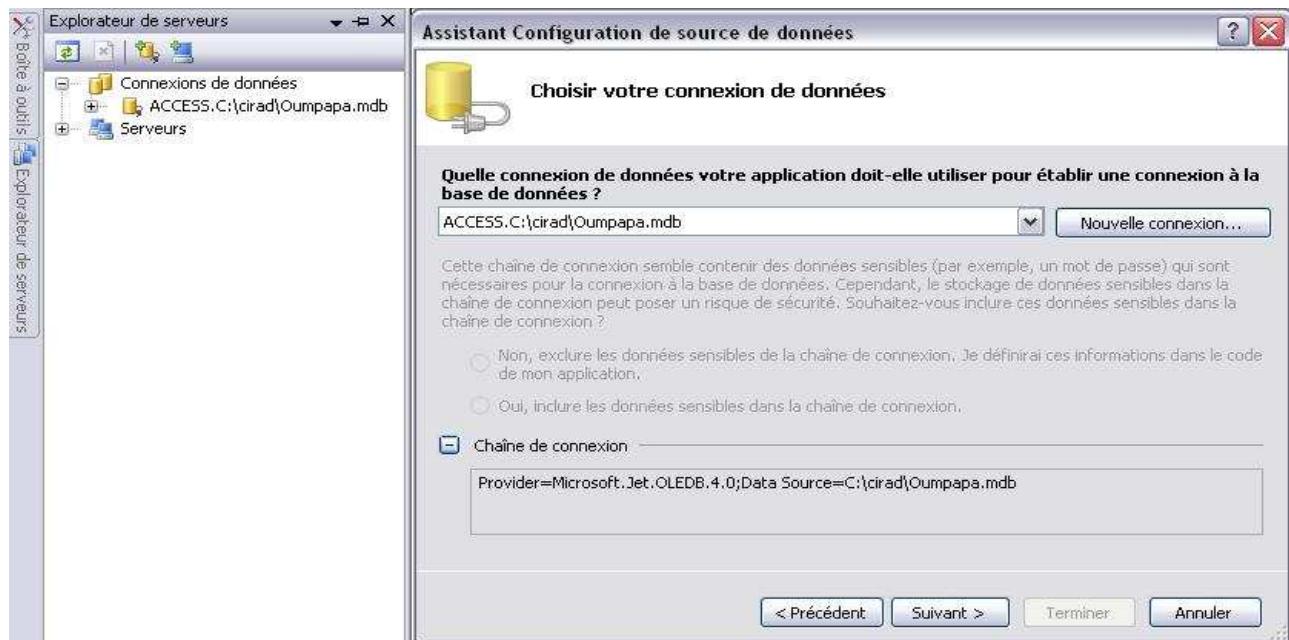
- The first thing is to create the data source to display (the example is a database table):

In the Visual Studio menubar, "Adding a new data source ..."

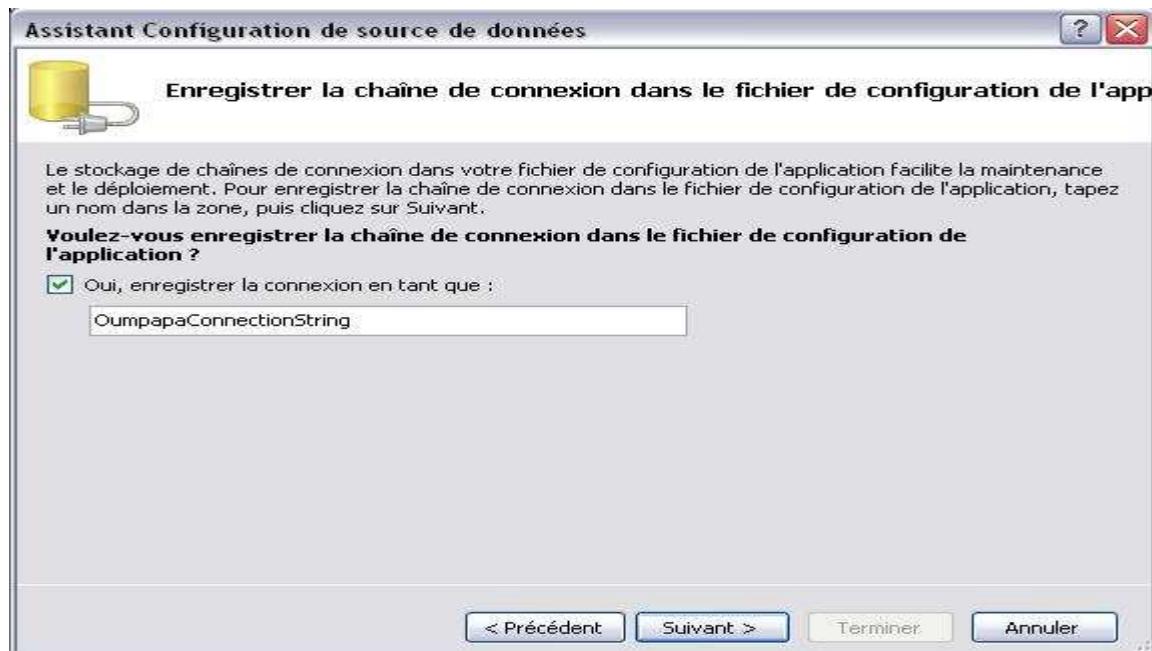




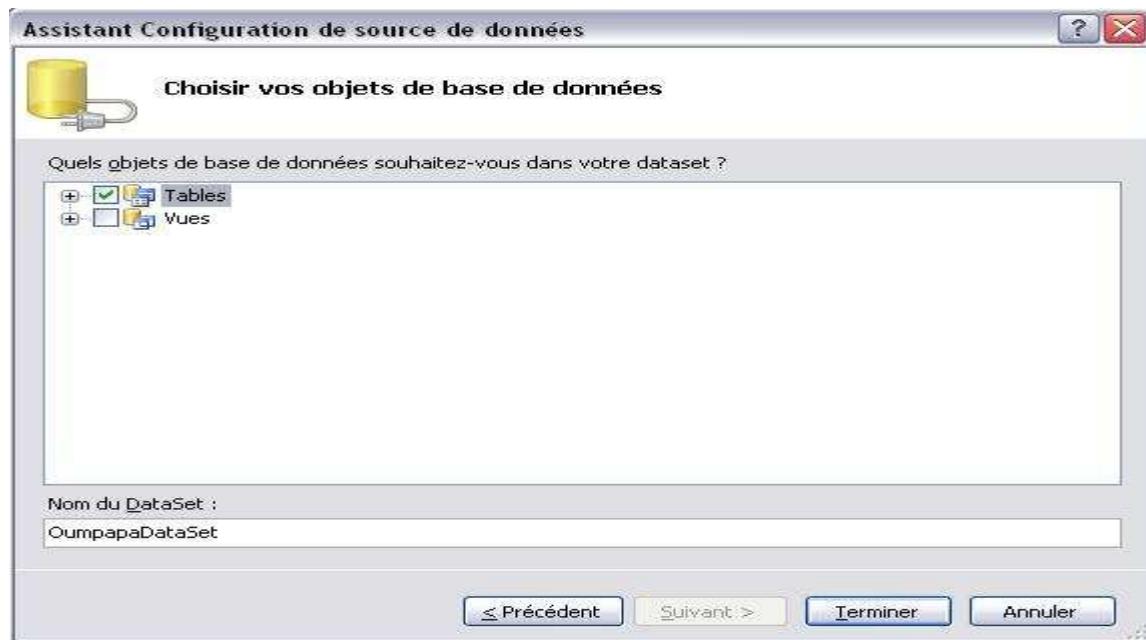
The database selection creates a data connection in the server browser. It is possible when you add the connection to define the login and password to access to the database.



We can see that a connection string to the database is created automatically.

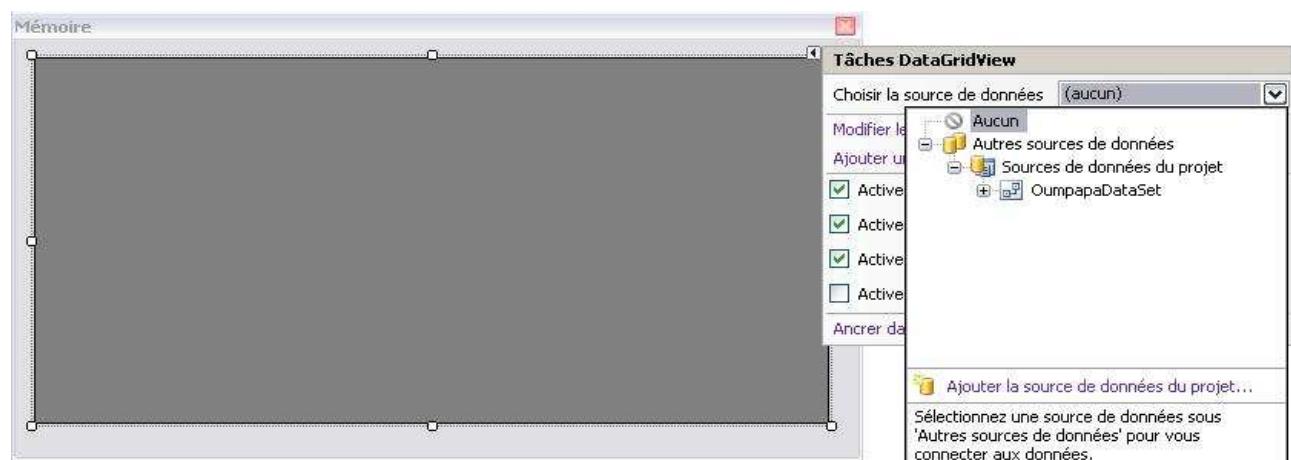


It only remains to select tables that take place in the DataSet.

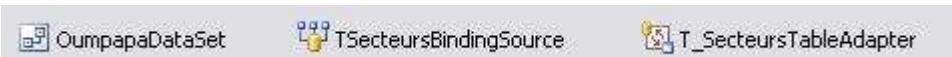


- Once the data source is in place, we can create the interface:
We add a DataGridView on the form (or any other control that uses the dataset as a data source).

Choose the table that will be displayed.

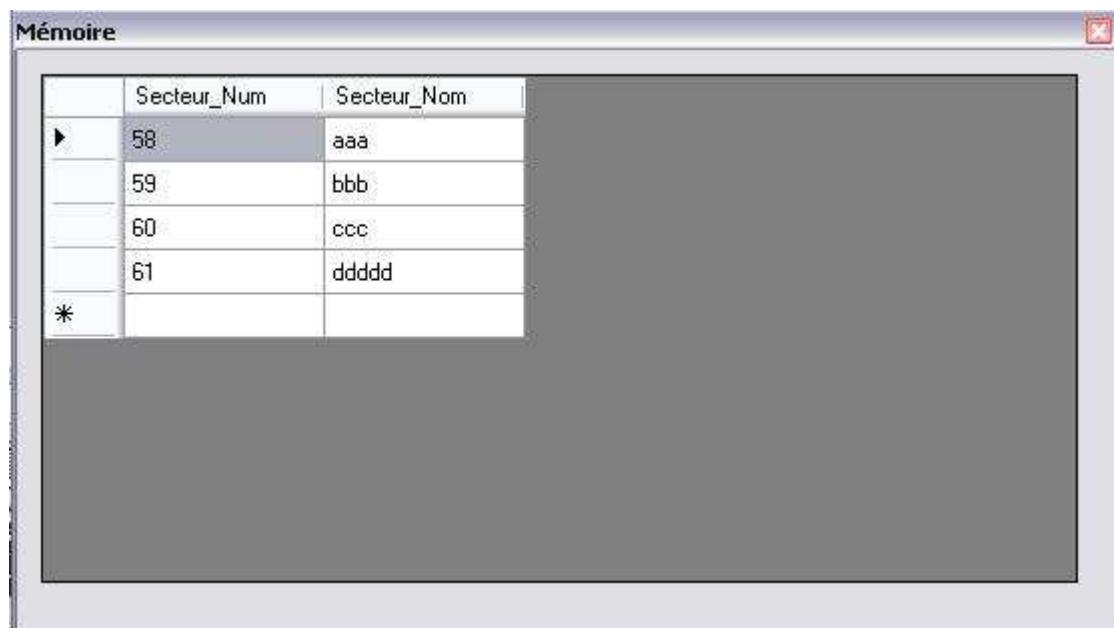


The structure of the selected table is set up in the data source control.



A DataSet, a BindingSource and a DataAdapter are added to the project.

Result of manipulation:



7.4. DataReader

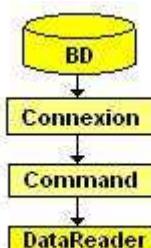
We shall now see the DataReader.

This object allows a quick reading of the Data Stream (*read-only*).

The access to the data source is made register by register. The modus operandi of the DataReader is a “*connected mode*” (monopolizing the connection to the data source), so we have to close the connection to the database to allow any connection to it. The DataReader is specific to the data source (*OleDbDataReader* for Access database).

Like the DataSet, loading a DataReader requires using the objects:

- **Connection:** possesses properties related to the connection to the database
 - *State*: connection status
 - Open / Close
 - ...
- **Command:** Command object is specific to the database (*OleDbCommand* for Access Database)



```

Dim MyConnexion As OleDbConnection = New OleDbConnection("Chaine de connexion")
Dim Mycommand As OleDbCommand = MyConnexion.CreateCommand()
Mycommand.CommandText = "SELECT * FROM MaTable"
MyConnexion.Open()
Dim myReader As OleDbDataReader = Mycommand.ExecuteReader()
Do While myReader.Read()
    ListBox1.Items.Add(myReader.GetString(0))
Loop
myReader.Close()
MyConnexion.Close()
  
```

The property ***Read*** of DataReader allows to read the data source component by component and returns a boolean to know if the reading is finished or not.

The Command object linked to the DataReader possesses an ***ExecuteScalar*** property which allows retrieving the result of a SQL query. It concerns the result of the instructions:

- ***COUNT***: count the number of rows returned
- ***AVG***: average
- ***MIN***: the minimum value
- ***MAX***: the maximum value
- ***SUM***: calculate the sum

```
Dim MyConnexion As OleDbConnection = New OleDbConnection("Chaine de connexion")
Dim Mycommand As OleDbCommand = MyConnexion.CreateCommand()
Mycommand.CommandText = "SELECT COUNT(*) FROM MaTable"
MyConnexion.Open()
Dim myReader As OleDbDataReader = Mycommand.ExecuteReader()

Dim iResultat As Integer = Mycommand.ExecuteScalar()

myReader.Close()
MyConnexion.Close()
```

The main properties of the object ***Command*** that were used in the project:

- ***ExecuteReader***: executes the query and returns a DataReader
- ***ExecuteScalar***: executes the query and returns the result of the calculation
- ***ExecuteNonQuery***: executes queries that do not return data (INSERT, DELETE, UPDATE)

Finally, error handling is made with the object ***OleDbException***:

```
Try
    ...
Catch ex As OleDbException
    ...
End Try
```

7.5. The Streams

The storage of the temperature data is on disk in "txt" files.

We need to know how to handle a *data stream*.

The *Stream* can be used for reading or writing in the file. Using the *Stream* means that the data is processed from beginning to end of the file.

Streaming follows a certain *algorithm*:

Writing in a text file:

1. *Instantiate* an object type *StreamWriter*: instantiation of an object can *create* or *edit* the file and then *open* it

```
'Crée ou écrase le fichier
Dim monSW As New StreamWriter("LeChemin\LeFichier.txt")

'Crée ou modifie le fichier
Dim monSW As New StreamWriter("LeChemin\LeFichier.txt", True)

'Crée ou écrase le fichier
Dim monSW As StreamWriter = File.CreateText("LeChemin\LeFichier.txt")

'Crée ou modifie le fichier
Dim monSW As StreamWriter = File.AppendText("LeChemin\LeFichier.txt")
```

2. *Write* in the file using methods *Write* or *WriteLine*

```
'Ecriture dans le fichier
monSW.Write("Hello")

'Ecriture dans le fichier puis passage à la ligne
monSW.WriteLine("Hello")
```

3. *Close* the file

```
'Fermeture du flux
monSW.Close()
```

- **Reading a text file:**

1. **Instantiate** an object type *StreamReader*

```
'L'instanciation ouvre le fichier
Dim monSR As New StreamReader("LeChemin\LeFichier.txt")

'L'instanciation ouvre le fichier
Dim monSR As StreamReader = File.OpenText("LeChemin\LeFichier.txt")
```

2. **Read** file

```
'Lire jusqu'à la fin du fichier - retourne un String
monSR.ReadToEnd()

'Lire jusqu'à la fin du fichier - retourne un String
Do Until monSR.Peek = -1
    monSR.ReadLine()
Loop
```

It is also possible to use the method Read () to read a number of bytes.

3. **Close** the file

```
'Fermer le fichier
monSR.Close()
```

7.6. Impression statements

This software is used to simplify the management of a pineapple culture, particularly through the use of a table, chart, listing ...

These results can be presented from the report printing.

A report printing can be compared to a "*template*" in a Web application.

In the previous version, these prints were made from the objects "*States*" of Access. These objects are directly linked to the database and the Windows forms, load the data for these statements were relatively simple.

What happens in VB.NET?

Visual Studio 2005 has a tool to create reports called "*Crystal Reports*".

By adding an element "*Crystal Report*" to the project (in the explorer solution: right click on the project => Add => Add a new element), an additional menu is added to the menu bar of visual studio, "Crystal Reports".

It's possible to use different types of data sources to load the state.



It's recommended to use the DataSet as a data source because it's the only thing to be independent of the database. The DataSet is part of the objects manipulated by the application .Net.

The DataSet can be picked up directly after a query on the database:

```

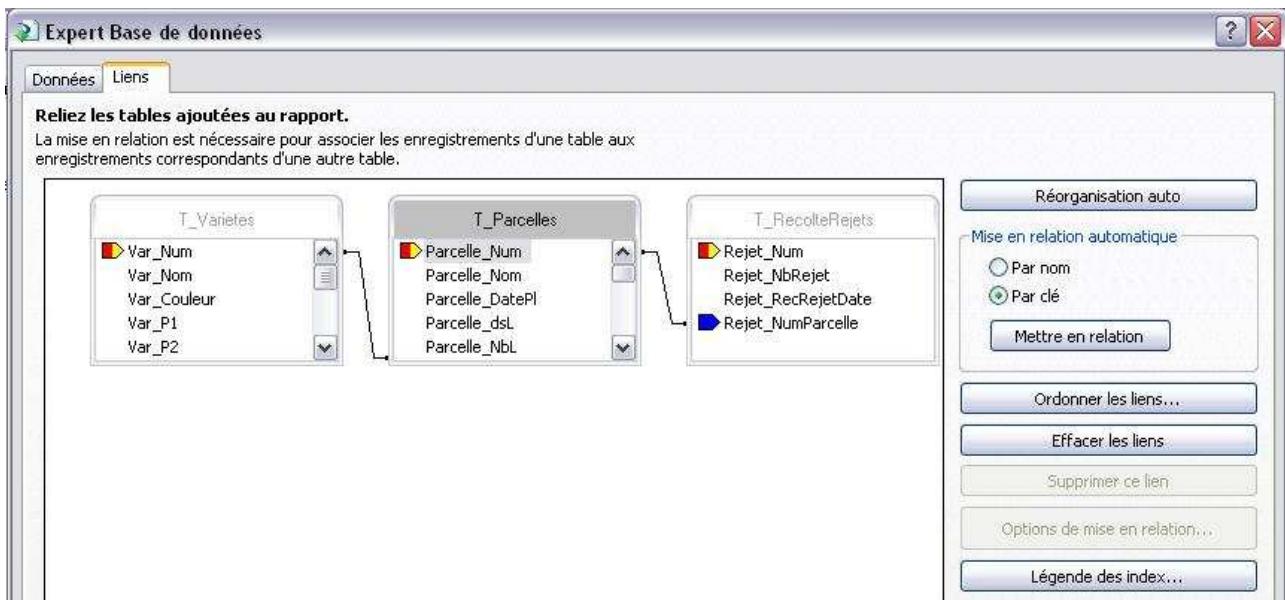
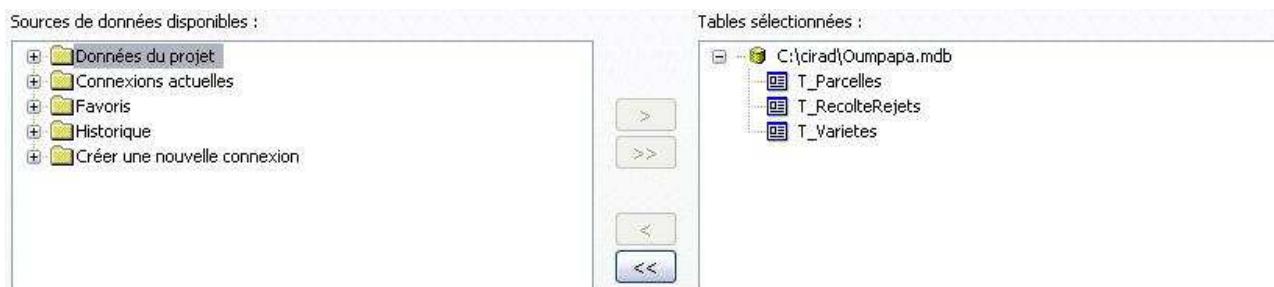
'Paramétrage de la chaîne de connexion
Connection.ConnectionString = My.Settings.maConnection
'Ouverture de la connexion
Connection.Open()
'Définition de la requête (on sélectionne tous les champs de la table)
Sql = "SELECT * FROM T_Secteurs"
'Définition du DataAdapter
AdapTest = New OleDbDataAdapter(Sql, Connection)
'Remplissage du dataset
DtSet = Nothing
DtSet = New DataSet()
AdapTest.Fill(DtSet, "T_Secteurs")
'Fermeture de la connection
Connection.Close()

```

With the GUI Visual Studio:

1. Create a data connection on the servers Explorer
2. Add the DataSet to the project
3. Drag tables on the DataSet designer

During the creation of the state, select the data source while checking the connections between tables.



In a WindowsForm, viewing a report printing is done through a "*CrystalReportViewer control*".

It would be best to add a Windows form to the project dedicated to report viewing.

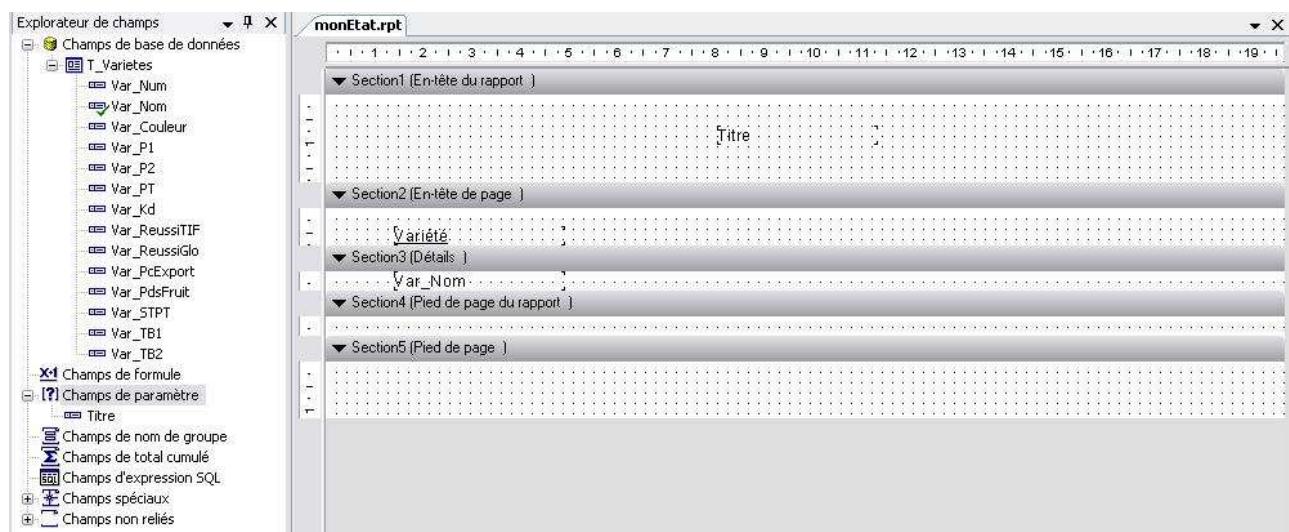
```
Dim monEtat As New EtatCree
monEtat.SetDataSource(monDataSet)
Dim maVariable As String
maVariable = "Bonjour"
monEtat.SetParameterValue("Titre", maVariable)
maFormeVisualisation.CRVViewerEtat.ReportSource = monEtat
maFormeVisualisation.Show()
```

In this example,

- **MonDataSet** : data source that will load the state (result of a query or DataSet)
- **SetDataSource** : method which get the data source
- **SetParameterValue** : method which get a parameter to the state (in this example the variable is called "title")
- **MaFormeVisualisation** : form which posses the control CrystalReportViewer
- **CRViewerEtat** : CrystalReportViewer control
- **ReportSource** - method that allows the state to spend viewer

N.B.: Passing parameter must be made before the data source.

The layout of Crystal Report is done entirely from the graphic designer of Visual Studio. It's possible to choose a style of report from the creating assistant report or start from a blank report.



To retrieve the fields that must be posted on the state, simply drag and drop the desired fields on the state.

Analysis of the state areas:

Header of the state: this on the front page of the state

Page header: present on every page of the state

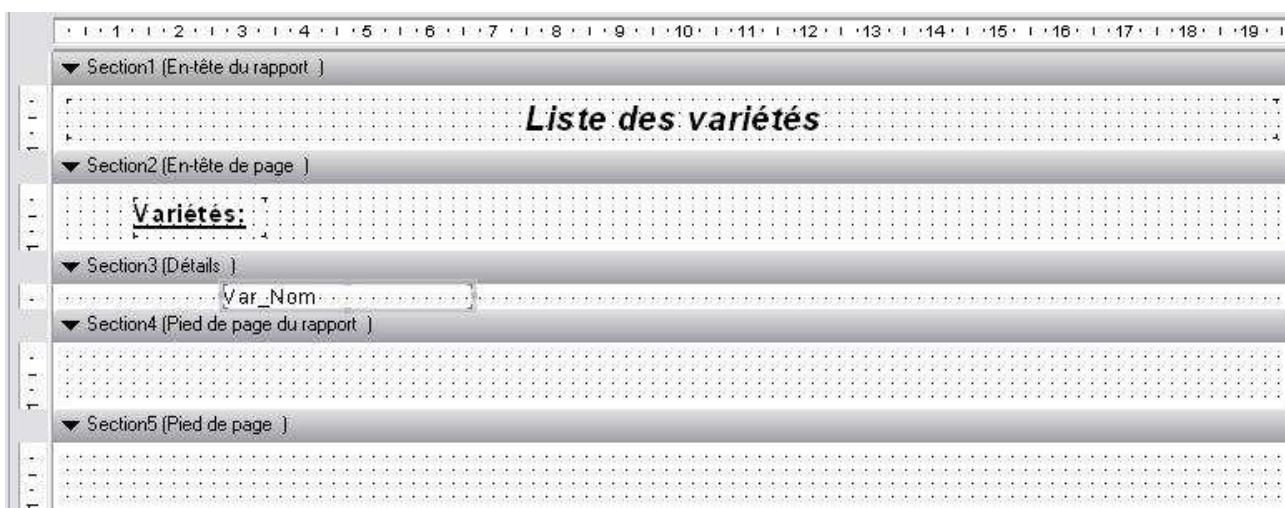
Details: results obtained with the application used in the data source

Footer State: present on the foot of the last page of the state

Footer: present on each footer of the state

Take the example of a query that displays entries in a table.

It wants to see a list of varieties listed in the database.



The report preview shows the following content:

Liste des variétés

Variétés:

- Cayenne
- MD 2
- RL41
- Queen Victoria
- Queen Mauritius
- Queen Tahiti
- Queen Mc Gregor

We can see that the result is what we done in the layout.

This result in raw form no longer requires a page layout. Crystal Report has obviously pokice formatting tools, text objects, line and framework to improve the presentation of the document.

It's possible to add additional areas in the state that will order the data according to certain criteria. These new zones are called "**Group**". The groups are created on the design part of the state from the menu or the mouse menu of Visual Studio.



[Imprimer le rapport](#)

Liste des variétés			
	<u>TIF</u>	<u>Global</u>	<u>Export</u>
Cayenne	0,95	0,72	0,70
MD 2	0,95	0,72	0,70
Queen Mauritius	0,00	0,00	0,00
Queen Mc Gregor	0,00	0,00	0,00
Queen Tahiti	0,00	0,00	0,00
Queen Victoria	0,95	0,72	0,70
RL41	0,95	0,72	0,70

To conclude this reporting presentation, we should know that the layout of a report printing must be carried out in accordance of the end user's need.

8. Demonstration of the approach's originality

The role that I play during my internship can be compared as an outside provider, I had to use a working method as best as a service provider can, therefore proposing a more professional service possible.

8.1. Programming Rule

The application is intended to be re taken by my training course's tutor, it is better to follow the policy of "readability of the code."

This policy follows three stages:

- Comments
- Nomenclature
- Clear code

Comments

- Comments were placed before the line of code (except for the variable declaration : commentary at the end of the line)

```
Dim MaVar As Integer = 0           ' Identifiant de la variété  
  
' Récupération de l'identifiant de la variété  
MaVar = RecupNumVariete("Victoria")
```

- We could make, at the beginning of each procedure, a description of the field:
 - Description of its use
 - Characteristic parameters
 - Return value or modified?
 - Example of use

- Comment loops seems appropriate especially in the case of nested loops
- The use of XML comment makes the code more understandable whatever the programmer's position in the code. The user gets the description of the procedure just by typing the name's function. This type of comment is accessible by typing three ratings ('').

```

    <summary>
    Fontion de Test
    </summary>
    <param name="Message"></param>
    <remarks></remarks>
Private Sub Test(ByVal Message As String)

End Sub


---


Private Sub Memoire_Load(ByVal sender As System.EventArgs)
    test(
        Test (Message As String)
        Fontion de Test
    End Class

```

Nomenclature

In addition to the comments, we can use variables' names to make the code more understandable.

- Use an explicit object name
- Avoid subjective names
- The concatenation of several words in the name's object is recommended
- In this case, we use capital letters to separate words
 - the first letter of each word is capitalized (Pascal Case)
 - the first letter of each word is capitalized except in the first word (Camel Case)
 - all the letters are capitalized (Upper Case)

Microsoft proposes a few rules when choosing names:

- Sub, Function: use the Pascal Case
- Variable:
 - using the Camel Case
 - the first letter defines the type of a variable or its scope
 - add calculation methods at the end of the name (Min, Max, Total)
 - Boolean variables must contain 'Is' in the name
 - use names with a single letter for the small loop index
- Parameter: use the Camel Case
- Constant: use the Upper Case
- Objet, Class:
 - use the Pascal Case
 - do not include the name of the class in the name of the property
 - interfaces name must begin with 'T'
 - using the Camel Case for private or protected variables of a class

- private variables must begin with '_'
- Table:
 - using the singular
 - do not incorporate the data type in the name of the column
- Other:
 - minimize abbreviations
 - insert a description of the returned value in the functions' name
 - avoid homonyms
 - avoid typographical signs
 - name must indicate the meaning rather than the method

Aeration code

Sometimes, it's difficult to read a code when it's too compact (often the case in lengthy routines).

Here are a few tips listed on the Web:

- avoid multiple instructions on a line
- separate the code by going to the next line for clarification
- well indent the code (especially in loops and If)

8.2. Code Optimization

Once the application is functional, we can move into a phase optimization by reducing the execution time of some functions or even the length of the code. According to the comparison made between VB6 and VB.NET, a VB6 application is faster, but can we make the program faster VB.NET?

Before anything else, we use object-oriented to avoid lengthy routines. VB.NET provides a lot of features like search or sorting methods, which will avoid nested loops (main cause of execution delay).

The selection of variables is part of the code optimization. Computers currently using 32-bit platforms, it is advised to use variables with a same width.

The fastest variable to the slower				
Integer	Integer	Long	Short	Byte
Real	Double		Single	Decimal

The integer treatment is the fastest, it is better to use float variables (in precision needs).

We should use the same type in passing or in reception. The use of a general variable in reception will slow the processing of the data.

Similarly, a passage of variable "by value" is faster than "by reference". Indeed, the move by value stores data directly in the stack.

Finally, the use of a variable is faster than an object property.

The same is true for when using tables, there are some tricks optimizations:

- Use the least possible dimension in a table
- Using an table of table is faster than a multi-dimensional table
- Search into a table is faster than in a collection
- Access to a variable is faster than that of an element of a table
- Affect the value of a table element to a variable if the use of this element is multiple
- Using a table is faster than a SelectCase If ... then
- BinarySearch is faster than the IndexOf of a table
- The use of generic methods (introduced by VB2005) gives the work on the table faster
- The use of a collection is better when the number of elements is unknown
- As well as for tables, generic collections are faster because they are typed

It's better to avoid Object variables, "none typed" and late links (the type of object is retrieved during the execution). Use a typed variable at the earliest possible to remove treatment during performance (analysis of the type of the object). All control made at the compiling leads less control during execution.

The handling of the files takes an important part in my internship project. It is advise to use directly the System.IO classes to perform these operations. Indeed, all methods of files manipulation use System.IO. So direct use removes some treatment steps.

The use of SelectCase in VB is faster than a long serie of If then, ElseIf.

We must put the most frequent treatment cases (case) at the beginning of the Select to reduce the lines of treatment.

There are also tips on programming operators:

- Divisions can be performed by "/" and "\."
 - o "\." is faster because it takes only the quotient of the division
 - o "/" also gets the rest or the fractional result
- $A += 1$ is faster than $A = A + 1$
- Use AndAlso and ElseOr instead of And and Or (the second expression is treated only if the first test is valid)
- Two consecutive If...then are faster than If ... And...then (the additional treatment won't be execute)
- Reduce the use of operations which demanding high processing time
 - o An addition is faster than a multiplication
 - o Log, Sinus, Cosinus are greedy operations
 - o If the calculated result of an expression is known in advance, it can be stored in a single variable and then used
- The typed conversions and DirectCast are faster than CType

Using ;

[With Object](#)

...

[With End](#) ;

is faster than an Objet.Propriété use because the evaluation of the object can be done only once.

Optimize a loop is also possible. All operations unmodified in a loop should be assigned to a variable outside the loop, the use of this variable avoid the recalculation the same transaction in the loop.

The use of a Try ... Catch ... Finally is much faster than the management of the On Error event.

Likewise, the use of the method "Exit For" avoids unnecessary processing loop.

In some cases, the use of thread can be better.

It is also possible to speed up the processing on String variables.

The use of the operator "&" for the string concatenation is faster than "+".

Using StringBuilder is faster than lots of concatenation.

There are also methods to speed up the display. There is an advantage to pre-load the window at startup of the application and make them invisible.

It is also better to change a variable and display it at the end instead of making changes and updates successively on visual objects. It is better to make a treatment on a table rather than on a file. Data processing in memory is faster.

According to the optimization case identified at various sites, we can perceive an optimization of speed on the big loops, data processing in memory and processing error handling.

8.3. Positioning

Finally, an application must be under a Windows Installer form to be distributed.

Visual Studio 2005 proposes two methods to deploy Windows applications.

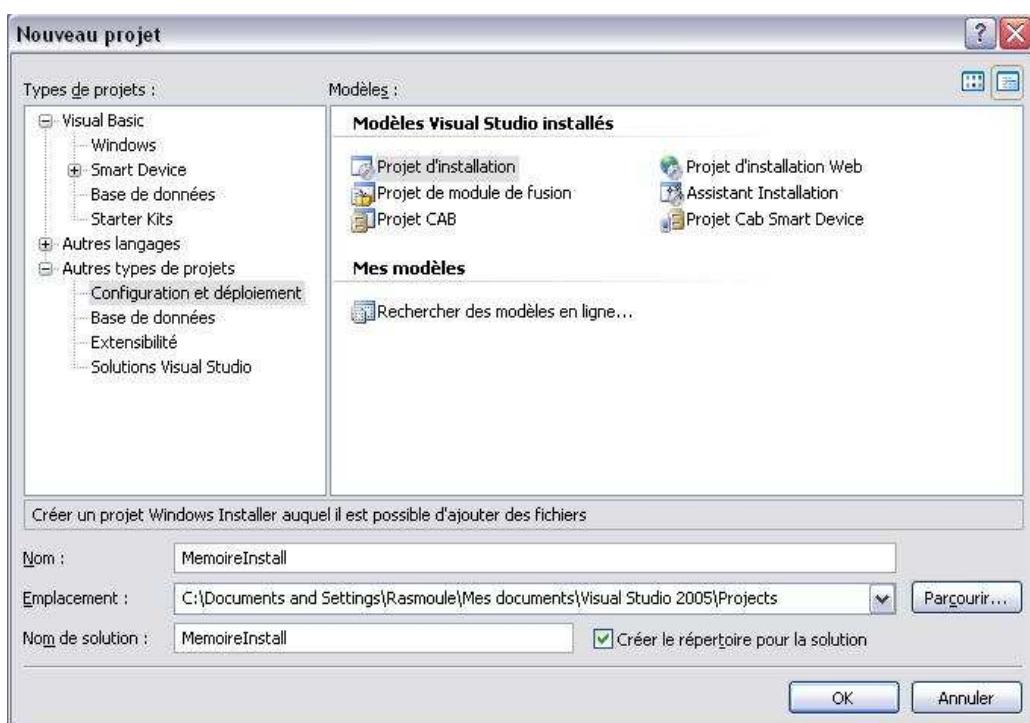
Windows Installer type deployment

This method creates a package of the application in an install file. ("EXE" and "MSI").

These setup files can be distributed in any traditional support.

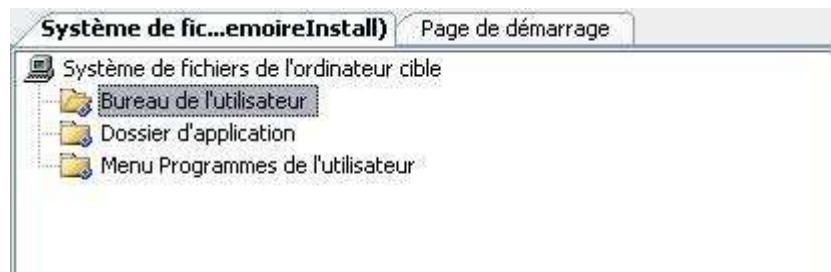
Above all, we must create a deployment project.

"File ➔ New project ➔ Configuration and deployment ➔ Installation project"



Once the project is created, Visual Studio displays the file system that will be used to install an application:

- **Desktop user:** any files in this directory will be copied to the user's desktop
- **Application directory:** all files in this directory will be copied into the install application directory
- **User programs menu:** any files in this directory will be copied into the start menu



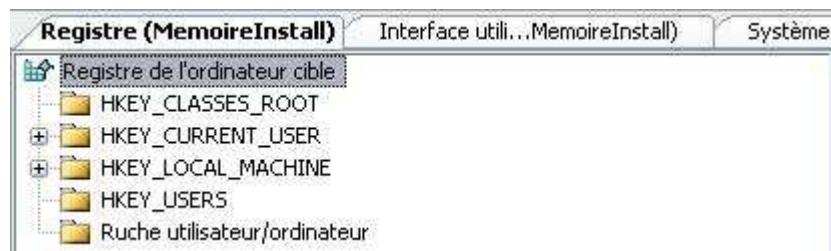
It is possible to add any specific Windows directories according to the organization that we want to adopt to install the software.

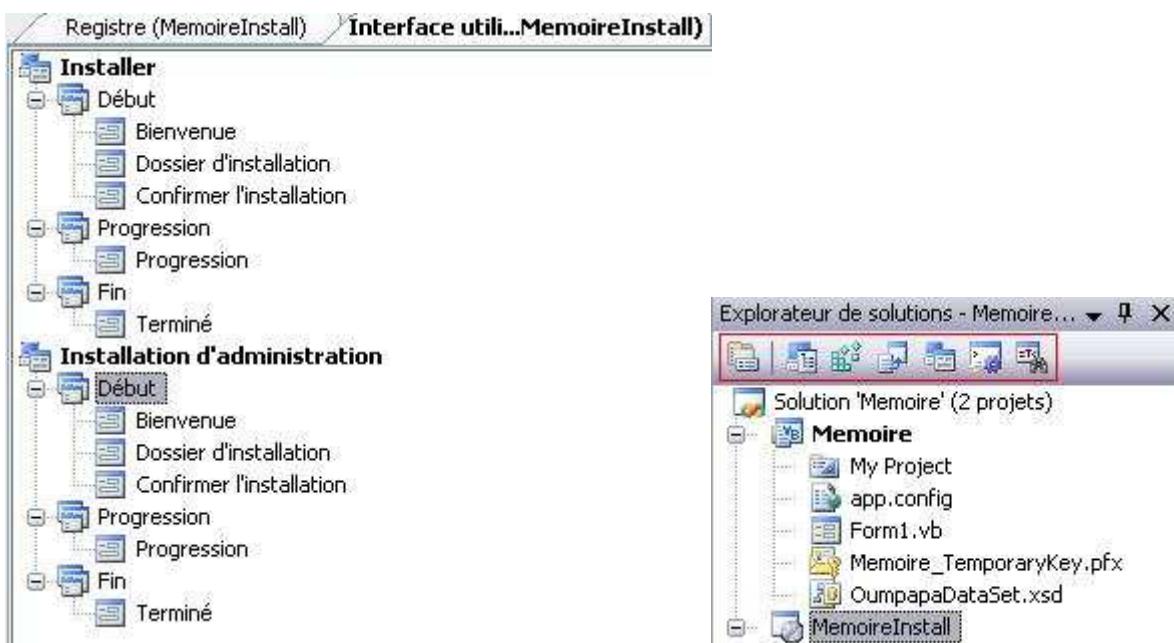


The deployment is completely editable from the registry database, the user interface installation, automatic directory creation, access type to the files...

The various features can be customized from specific windows.

These windows are available at the top of the built solution since the deployment project is selected (menu box in red on the image below).





It only remains to add the file which will be used by the end user to launch the application. This file is called "main output". To do this:

"Right click on the file system in which you wish to install ➔ Add ➔ Exit Project"



- **File Documentation:** Documentation files in XML format.
- **Output:** anything that will be generated by the Windows application
- **Local resources:** data non executable
- **Sources Files:** source files of the application (open source application)

Once the deployment project is created, we add it to the project that must be packaged.

“File → Add → Existing project → Choose the project deployment newly created”.

The customization terms of installation (setup interfaces, database registry, ownership of the files, etc.) must be defined by the developer.

The Windows Installer file will be generated in the Debug directory of the Project after a record and a generation of the project.

“Right click on the proposed deployment → Generate”

The publication ClickOnce

This method is based on the HTTP protocol, the user installs and runs the program from the location where it was published.

This deployment is mostly used in frequent change cases on the application (deployed applications are updated automatically).

The publication of the application can be made either on traditional media, either on a web page.

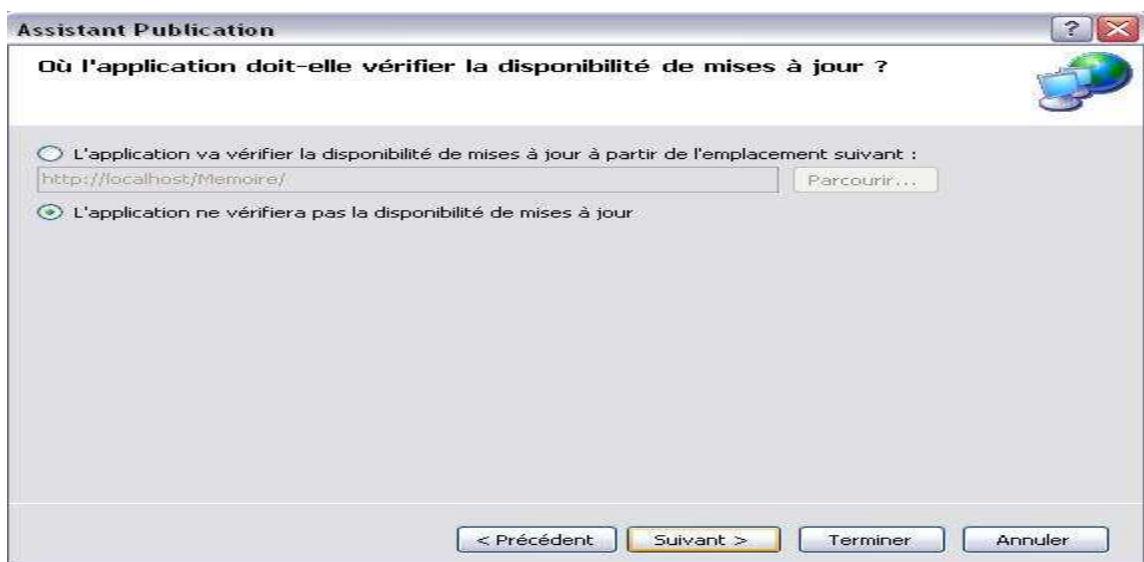
This method is as simple as the creation of Windows Installer thanks to the assistant of publication.

It can be accessed via:

“Right click on the project → Publish ... Or Visual Studio Generate Menu → Publish”



The window is fairly explicit; the selected location determines how the user will install the application (just like the next window).



We can see that the window proposes the possibility of updates.



At the end of the wizard, it creates generics files, as well as the installation file.

The solution presented is based on a publication on disk (which is use in my project). The Web publication requires some options that I could not experienced:

- Running IIS (Internet Information Service) on the web server
- Administrative privileges in IIS
- FrontPage extensions.

It is possible to configure parameters of publication through the project's properties and add some security settings.



9. Conditions of application of the suggested approach

My role during this stage is close to a service provider, so I had to document myself the most whatsoever on the culture of pineapple or the technology that I should use. It was therefore easier to obtain documentation for the rendering of the application and the thesis.

Through the lines of code for this application (calculations provided by CIRAD, algorithm calculations fairly complex), I understand the charge work provided by these research teams and the usef of the projects carried out in their walls.

So I decide to present in this report, the tools that I needed during the internship.

10. Reflecting on the management of the thesis

For my memory of Completion, as for the project, I tried to pass as much information, either on the Internet or around me.

Some concepts were discussed even though they could not be tested because it seemed important to mention here.

The validation period was in my opinion, a little too close to the start of the session because it could have taken another direction.

After this stage, I think I can say that the topic deserved further consideration whatsoever on the technical side or on the concept of the application.

11. Conclusion

After six months of internship at CIRAD, I was able to reach the goals that I have set myself. The system is functional; the layout of the GUI has to be done in conjunction with the end user. Working on a project with an unknown field and the information technology has been a real challenge for me.

The application has been tested only by the developer; one can only wait for the return of the test phase to establish the necessary changes.

I think it is possible for this application, to optimize the code in order to accelerate certain functions (data management temperature).

Finally, I believe it is possible to apply this program to another culture fruit.

12. Reading reference

Pineapple cultivation

Useful link to get an idea about a pineapple cultivation

Visual Basic .Net de Gilles Nicot – MicroApplication

IDE, Windows Forms, Web Forms... perfect to transcribe VB6 to VB.NET.

Microsoft msdn

101 examples for Visual Basic 2005

These examples are useful because they show a lot of use cases.

George Shepherd's Windows Forms FAQ

FAQ on Windows Forms.

Codes-SourceS

Forum that help programmers in lots of languages

Developpez.com

<http://dotnet.developpez.com/>

Article and FAQ on dotnet

<http://convertisseur.developpez.com/>

Useful converter for VB beginners

<http://www.developpez.net/forums/>

Help Forum for programmers

<http://sql.developpez.com/>

Articles and FAQ on SQL

<http://faqvbnet.developpez.com/>

FAQ on VB.NET

<http://dotnet.developpez.com/faq/dotnet/>

FAQ on dotnet