

+



El Kibambo v. 2.1

Manuel officiel



EI CONCEPT
Tous droits réservés

Table de contenus

Table de contenus.....	1
Détails techniques.....	2
Vue d'ensemble	3
1. Les énumérations.....	4
1.1. KibamboErrorCode.....	4
1.2. KibamboColumnType.....	5
1.3. KibamboComparisonType.....	6
2. Les classes.....	6
2.1. KibamboException	6
2.2. KibamboColumn	7
a. Propriétés.....	8
b. Constructeurs.....	9
c. Méthodes.....	10
2.3. Kibambo	10
a. Constructeurs	12
b. Propriétés	14
AutoCompleteColumns	14
Columns	14
ComparisonType	14
RowsSatisfied et RowsUnsatisfied	14
SearchHandler et CancelSearchHandler	15
TitleField, TitleType, TitleOperator, TitleValue	15
c. Méthodes.....	15
Search()	15
CancelSearch().....	16
Initialize()	16
Mise en œuvre (C#).....	17
Mise en œuvre (VB.NET).....	28
Historique.....	37

Détails techniques

Créateur	Michael KYUNGU ILUNGA
Développé par	EI CONCEPT
Première version	Beta (v. 1.0) / 2021
Dernière version	2.1 (30 janvier 2022)
Dépôt	https://github.com/elconcept15/elkibambo/tree/main/winforms
Ecrit en	C#
Type	Bibliothèque de classes .NET
Framework .NET cible	> = 4.5
Langues prise en charge	Français et anglais

Vue d'ensemble

El Kibambo est une bibliothèque .NET permettant une implémentation rapide et facile des fonctionnalités de recherche sur des grilles de données dans les programmes Windows Forms.

Avec El Kibambo, la recherche peut se faire sur plusieurs colonnes de la grille de données (DataGridView) et la comparaison n'est pas limitée seulement à l'égalité.

El Kibambo est actuellement dans sa version 2.1, c'est la première version alpha.

Dans ce manuel, nous allons l'illustrer avec les langages de programmation C# et VB.NET.

El Kibambo prend en charge actuellement les opérateurs de comparaison suivants :

- ✚ == (égalité) ;
- ✚ != (différent) ;
- ✚ > (strictement supérieur) ;
- ✚ < (strictement inférieur) ;
- ✚ >= (supérieur ou égal) ;
- ✚ <= (inférieur ou égal) ;
- ✚ LIKE : une tentative d'El Kibambo d'utiliser la légendaire recherche par jeton comme l'opérateur SQL « LIKE ».

La bibliothèque El Kibambo possède comme classes :

- ✚ **Kibambo** : c'est la classe principale de la bibliothèque. C'est grâce à elle que la recherche est mise en œuvre.
- ✚ **KibamboColumn** : représente chaque colonne de la grille contenant les données.
- ✚ **KibamboException** : en cas d'erreur, El Kibambo déclenche des exceptions de ce type. Cette classe possède un attribut de type KibamboErrorCode nous permettant de mieux savoir le type d'erreur.

El Kibambo utilise les énumérations suivantes :

- ✚ **KibamboErrorCode** : chaque objet de type KibamboException possède l'attribut ErrorCode de type KibamboErrorCode nous donnant plus de précisions sur le type d'erreur.
- ✚ **KibamboColumnType** : chaque objet de type KibamboColumn possède une propriété de type KibamboColumnType indiquant le type de valeur de la colonne. Au fait, une colonne peut posséder des valeurs numériques ou des chaînes de caractères, et la façon dont la comparaison est faite pour les valeurs numériques et les chaînes de caractères n'est pas la même.

- ✚ **KibamboComparisonType** : permet de spécifier le type de comparaison (sensible ou non sensible à la casse, etc.). Chaque objet Kibambo possède un attribut `ComparisonType` permettant de spécifier le type de comparaison.

El Kibambo définit également un délégué, le délégué `KibamboDelegate`. [...] :

```
public delegate void KibamboDelegate(DataGridViewRow row);
```

```
Public Delegate Sub KibamboDelegate(row As DataGridViewRow)
```

1. Les énumérations

1.1. KibamboErrorCode

Vue d'ensemble :

```
public enum KibamboErrorCode
{
    ColumnNotRecognized = 0,
    ColumnsNotSpecified = 1,
    ComparisonTypeNotRecognized = 2,
    DgvOfDataMustBeDifferentThanDgvOfSearch = 3,
    MoreColumnsThanDgvData = 4,
    OperatorNotRecognized = 5,
    ValueMustBeNumeric = 6,
    YouMustSpecifyAtLeastOneOperator = 7
}
```

```
Public Enum KibamboErrorCode
    ColumnNotRecognized = 0
    ColumnsNotSpecified = 1
    ComparisonTypeNotRecognized = 2
    DgvOfDataMustBeDifferentThanDgvOfSearch = 3
    MoreColumnsThanDgvData = 4
    OperatorNotRecognized = 5
    ValueMustBeNumeric = 6
    YouMustSpecifyAtLeastOneOperator = 7
End Enum
```

Cette énumération possède comme valeurs :

- ✚ **ColumnNotRecognized** : avec El Kibambo, chaque colonne de la grille de données est passée sous la forme d'un objet de type `KibamboColumn` lors de la création de l'objet Kibambo. Pour être précis, on doit passer au constructeur de la classe Kibambo un tableau des objets `KibamboColumn`, représentant les colonnes de la grille contenant les données. Lors de la recherche, si telle colonne est utilisée, El Kibambo va chercher toutes les lignes dont telle colonne satisfait la recherche. Cette erreur est déclenchée au cas où El Kibambo n'arrive pas à retrouver une colonne. C'est le cas par exemple d'une colonne inexistante dans la grille contenant les données, ou supprimée/renommée (propriété `Name` d'un objet `DataGridViewColumn`) après instanciation de l'objet Kibambo.

- ✚ **ColumnsNotSpecified** : cette erreur est due au fait que nous n'avons pas spécifié des colonnes lors de l'instanciation de l'objet Kibambo et que la propriété `Kibambo.AutoCompleteColumns` vaut `false`.
- ✚ **ComparisonTypeNotRecognized** : ce type d'erreur est interne à la bibliothèque. Par défaut, la comparaison est de type non sensible à la casse et ignorant les signes diacritiques (`KibamboComparisonType.IgnoreDiacriticSignsAndCase`). Néanmoins, en cas d'erreur interne grave et qu'El Kibambo n'arrive pas à déterminer le type de comparaison (ce qui est vraiment très moins probable), cette erreur peut alors survenir.
- ✚ **DgvOfDataMustBeDifferentThanDgvOfSearch** : survient si nous passons un même `DataGridView` comme grille contenant les données et aussi comme grille de recherche lors de l'instanciation de l'objet Kibambo.
- ✚ **MoreColumnsThanDgvData** : survient si nous passons plus d'objets `KibamboColumn` à l'objet Kibambo que n'en contient réellement la grille contenant les données.
- ✚ **OperatorNotRecognized** : cette erreur survient dans le cas où, pour une raison ou pour une autre, El Kibambo n'arrive pas à reconnaître l'opérateur de comparaison lors de la recherche (ce qui est encore extrêmement très moins probable).
- ✚ **ValueMustBeNumeric** : cette erreur est due au fait que l'utilisateur saisit une valeur non numérique pour une colonne de type numérique.
- ✚ **YouMustSpecifyAtLeastOneOperator** : survient si nous passons un tableau vide pour les opérateurs lors de la construction d'un objet `KibamboColumn`.

1.2. KibamboColumnType

Vue d'ensemble :

```
public enum KibamboColumnType
{
    String = 0,
    Numeric = 1
}
```

```
Public Enum KibamboColumnType
    [String] = 0
    Numeric = 1
End Enum
```

Pour le moment, El Kibambo ne supporte que deux types de valeurs :

- ✚ **String** : pour les chaînes de caractères ;
- ✚ **Numeric** : pour les valeurs numériques.

Ainsi, El Kibambo peut être utilisée avec les autres types en utilisant `KibamboColumnType.String`.

Mais pourquoi ... ? Il faut savoir que la comparaison entre les numériques et les non numériques (chaînes de caractères) est diamétralement opposée. Si avec les numériques c'est l'ordre de

grandeur qui est utilisé, avec les chaînes de caractères ce n'est pas le cas. Ainsi, « "10" < "2" » donne vrai mais « 10 < 2 » donne faux !

1.3. KibamboComparisonType

Vue d'ensemble :

```
public enum KibamboComparisonType
{
    CaseSensitive = 0,
    IgnoreCase = 1,
    IgnoreDiacriticSigns = 2,
    IgnoreDiacriticSignsAndCase = 3
}
```

```
Public Enum KibamboComparisonType
    CaseSensitive = 0
    IgnoreCase = 1
    IgnoreDiacriticSigns = 2
    IgnoreDiacriticSignsAndCase = 3
End Enum
```

Cette énumération possède comme valeurs :

- ✚ **CaseSensitive** : pour une comparaison ordinaire des chaînes de caractères respectant la casse.
- ✚ **IgnoreCase** : pour une comparaison ordinaire des chaînes de caractères non sensible à la casse.
- ✚ **IgnoreDiacriticSigns** : pour une comparaison ignorant les signes diacritiques dont principalement les accents.
- ✚ **IgnoreDiacriticSignsAndCase** : fonctionne comme *IgnoreDiacriticSigns*, avec cet avantage que la comparaison est non sensible à la casse.

2. Les classes

2.1. KibamboException

Vue d'ensemble :

```
public class KibamboException : Exception
{
    public KibamboException(KibamboErrorCode errorCode);

    public KibamboErrorCode ErrorCode { get; }
    public string Message { get; }

    public override string ToString();
}
```

```
Public Class KibamboException Inherits Exception
    Public Sub New(errorCode As KibamboErrorCode)
```

```

Public ReadOnly Property ErrorCode As KibamboErrorCode
Public ReadOnly Property Message As String

Public Overrides Function ToString() As String
End Class

```

Héritant de la classe de base des exceptions Exception, cette classe définit deux propriétés :

- ✚ **Message** : accessible en lecture seule, elle fournit un message détaillant l'erreur. Cette propriété redéfinit celle définie dans la classe de base.

```
string KibamboException.Message { get; }
```

```
ReadOnly Property KibamboException.Message As String
```

- ✚ **ErrorCode** : contient le code d'erreur. Elle est de type KibamboErrorCode.

```
KibamboErrorCode KibamboException.ErrorCode { get; }
```

```
ReadOnly Property KibamboException.ErrorCode As KibamboErrorCode
```

La classe KibamboException redéfinit la méthode de base ToString() en ajoutant à la fin de la chaîne de caractères le message d'erreur :

```

public override string ToString()
{
    return base.ToString() + Environment.NewLine + "Kibambo Error Message : " + Message;
}

```

```

Public Overrides Function ToString() As String
    Return MyBase.ToString() + Environment.NewLine + "Kibambo Error Message : " + Message
End Function

```

2.2. KibamboColumn

La classe KibamboColumn permet de représenter chaque colonne de la grille contenant les données.

Vue d'ensemble :

```

public class KibamboColumn
{
    public static readonly string[] SupportedOperators;

    public KibamboColumn(string name, KibamboColumnType columnType);
    public KibamboColumn(string name, string header);
    public KibamboColumn(string name, string header, KibamboColumnType columnType);
    public KibamboColumn(string name, string header, KibamboColumnType columnType, string headerColumnType);
    public KibamboColumn(string name, string header, KibamboColumnType columnType, string headerColumnType, string[] operators);

    public KibamboColumnType ColumnType { get; set; }
}

```



```

    public string Header { get; set; }
    public string HeaderComponent { get; set; }
    public string Name { get; set; }
    public string[] Operators { get; set; }

    public override string ToString();
}

```

```

Public Class KibamboColumn
    Public Shared ReadOnly SupportedOperators As String()

    Public Sub New(name As String, columnType As KibamboColumnType)
    Public Sub New(name As String, header As String)
    Public Sub New(name As String, header As String, columnType As KibamboColumnType)
    Public Sub New(name As String, header As String, columnType As KibamboColumnType,
headerColumnType As String)
    Public Sub New(name As String, header As String, columnType As KibamboColumnType,
headerColumnType As String, operators() As String)

    Public Property ColumnType As KibamboColumnType
    Public Property Header As String
    Public Property HeaderComponent As String
    Public Property Name As String
    Public Property Operators As String()

    Public Overrides Function ToString() As String
End Class

```

a. Propriétés

KibamboColumn possède comme propriétés : Name, Header, ColumnType, HeaderComponent, Operators.

✚ **Name** : fait référence à la propriété DataGridViewColumn.Name de la colonne de la grille contenant les données que l'objet KibamboColumn représente.

```
string KibamboColumn.Name { get; set; }
```

```
Property KibamboColumn.Name As String
```

✚ **Header** : c'est l'intitulé de la colonne à afficher par El Kibambo.

```
string KibamboColumn.Header { get; set; }
```

```
Property KibamboColumn.Header As String
```

✚ **ColumnType** : représente le type de la colonne.

```
KibamboColumnType KibamboColumn.ColumnType { get; set; }
```

```
Property KibamboColumn.ColumnType As KibamboColumnType
```

- ✚ **HeaderColumnType** : le type de colonne à afficher par El Kibambo dans la grille de recherche.

```
string KibamboColumn.HeaderColumnType { get; set; }
```

```
Property KibamboColumn.HeaderColumnType As String
```

- ✚ **Operators** : représente un tableau des signes d'opérateurs de comparaison à utiliser pour la colonne. Pour rappel, El Kibambo ne supporte que huit opérateurs (==, !=, <, <=, >, >=, !=, LIKE). Vous devez spécifier que les opérateurs prisent en charge par El Kibambo. La présence d'un opérateur non pris en charge déclenche une `KibamboException` avec un code d'erreur valant `KibamboErrorCode.OperatorNotRecognized`. Le fait de spécifier un tableau vide déclenche également une `KibamboException` avec un code d'erreur valant `KibamboErrorCode.YouMustSpecifyAtLeastOneOperator`.

```
string[] KibamboColumn.Operators { get; set; }
```

```
Property KibamboColumn.Operators As String()
```

- ✚ **SupportedOperators** : constante contenant un tableau des opérateurs de comparaison prisent en charge par la bibliothèque El Kibambo.

```
string[] KibamboColumn.SupportedOperators
```

```
KibamboColumn.SupportedOperators As String()
```

b. Constructeurs

La classe `KibamboColumn` possède les constructeurs suivants :

```
KibamboColumn(string name, KibamboColumnType columnType)

KibamboColumn(string name, string header)

KibamboColumn(string name, string header, KibamboColumnType columnType)

KibamboColumn(string name, string header, KibamboColumnType columnType, string
headerColumnType)

KibamboColumn(string name, string header, KibamboColumnType columnType, string
headerColumnType, string[] operators)
```

```

KibamboColumn(name As String, columnType As KibamboColumnType)

KibamboColumn(name As String, header As String)

KibamboColumn(name As String, header As String, columnType As KibamboColuymnType)

KibamboColumn(name As String, header As String, columnType As KibamboColuymnType,
headerColumnType As String)

KibamboColumn(name As String, header As String, columnType As KibamboColuymnType,
headerColumnType As String, operators As String())

```

Détaillons chaque paramètre :

- ✚ name : valeur à affecter à la propriété KibamboColumn.Name ;
- ✚ header : valeur à affecter à la propriété KibamboColumn.Header ;
- ✚ columnType : valeur à affecter à la propriété KibamboColumn.ColumnType ;
- ✚ headerColumnType : valeur à affecter à la propriété KibamboColumn.HeaderColumnType ;
- ✚ operators : valeur à affecter à la propriété KibamboColumn.Operators.

En utilisant un constructeur ne prenant pas en paramètre un tableau des opérateurs de comparaison, tous les opérateurs de comparaison prisent en charge par El Kibambo seront automatiquement utilisés.

c. Méthodes

La classe KibamboColumn redéfinit juste une méthode : « ToString() ».

```

public override string ToString()
{
    return Header;
}

```

```

Public Overrides Function ToString() As String
    Return Header
End Function

```

2.3. Kibambo

La classe Kibambo constitue le cœur même de la bibliothèque El Kibambo. Alors, comment procède-t-on pour mettre en place la recherche avec El Kibambo ?

Au fait, il ne faut pas oublier qu'El Kibambo fonctionne actuellement qu'avec les grilles de données.

Lors de l'instanciation de l'objet Kibambo, nous devons passer au constructeur la grille contenant les données et celle permettant de mettre en œuvre la recherche.

Vue d'ensemble :

```

public class Kibambo
{
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, bool autoCompleteColumns =
false);
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns, bool
autoCompleteColumns = false);
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboDelegate
searchHandler, bool autoCompleteColumns = false);
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboDelegate
searchHandler, KibamboDelegate cancelSearchHandler, bool autoCompleteColumns = false);
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns,
KibamboDelegate searchHandler, bool autoCompleteColumns = false);
    public Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns,
KibamboDelegate searchHandler, KibamboDelegate cancelSearchHandler, bool autoCompleteColumns =
false);

    public bool AutoCompleteColumns { get; set; }
    public KibamboDelegate CancelSearchHandler { get; set; }
    public KibamboColumn[] Columns { get; set; }
    public KibamboComparisonType ComparisonType { get; set; }
    public List<DataGridViewRow> RowsSatisfied { get; }
    public List<DataGridViewRow> RowsUnsatisfied { get; }
    public KibamboDelegate SearchHandler { get; set; }
    public string TitleField { get; set; }
    public string TitleType { get; set; }
    public string TitleOperator { get; set; }
    public string TitleValue { get; set; }

    public void CancelSearch();
    public void Initialize();
    public void Search();
}

```

```

Public Class Kibambo
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, Optional
autoCompleteColumns As Boolean = False)
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, columns() As
KibamboColumn, Optional autoCompleteColumns As Boolean = False)
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, searchHandler As
KibamboDelegate, Optional autoCompleteColumns As Boolean = False)
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, searchHandler As
KibamboDelegate, cancelSearchHandler As KibamboDelegate, Optional autoCompleteColumns As
Boolean = False)
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, columns() As
KibamboColumn, searchHandler As KibamboDelegate, Optional autoCompleteColumns As Boolean =
False)
    Public Sub New(dgvData As DataGridView, dgvSearch As DataGridView, columns() As
KibamboColumn, searchHandler As KibamboDelegate, cancelSearchHandler As KibamboDelegate,
Optional autoCompleteColumns As Boolean = False)

    Public Property AutoCompleteColumns As Boolean
    Public Property CancelSearchHandler As KibamboDelegate
    Public Property Columns As KibamboColumn()
    Public Property ComparisonType As KibamboComparisonType
    Public ReadOnly Property RowsSatisfied As List(Of DataGridViewRow)
    Public ReadOnly Property RowsUnsatisfied As List(Of DataGridViewRow)
    Public Property SearchHandler As KibamboDelegate
    Public Property TitleHead As String
    Public Property TitleType As String
    Public Property TitleOperator As String

```

```

    Public Property TitleValue As String

    Public Sub CancelSearch()
    Public Sub Initialize()
    Public Sub Search()
End Class

```

a. Constructeurs

La classe Kibambo possède les constructeurs suivants :

```

Kibambo(DataGridView dgvData, DataGridView dgvSearch, [bool autoCompleteColumns = false])

Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns, [bool
autoCompleteColumns = false])

Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboDelegate searchHandler,
[bool autoCompleteColumns = false])

Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns,
KibamboDelegate searchHandler, [bool autoCompleteColumns = false])

Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboDelegate searchHandler,
KibamboDelegate cancelSearchHandler, [bool autoCompleteColumns = false])

Kibambo(DataGridView dgvData, DataGridView dgvSearch, KibamboColumn[] columns,
KibamboDelegate searchHandler, KibamboDelegate cancelSearchHandler, [bool
autoCompleteColumns = false])

```

```

Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, [autoCompleteColumns As
Boolean = False])

Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, columns As KibamboColumn(),
[autoCompleteColumns As Boolean = False])

Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, searchHandler As
KibamboDelegate, [autoCompleteColumns As Boolean = False])


Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, columns As KibamboColumn(),
searchHandler As KibamboDelegate, [autoCompleteColumns As Boolean = False])

Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, searchHandler As
KibamboDelegate, cancelSearchHandler As KibamboDelegate, [autoCompleteColumns As Boolean
= False])

Kibambo(dgvData As DataGridView, dgvSearch As DataGridView, columns As KibamboColumn(),
searchHandler As KibamboDelegate, cancelSearchHandler As KibamboDelegate,
[autoCompleteColumns As Boolean = False])

```

Détaillons chaque paramètre :

 dgvData : le DataGridView contenant les données ;

- ✚ dgvSearch : le DataGridView qui va permettre de mettre en place les fonctionnalités de recherche ;
- ✚ autoCompleteColumns : indique si nous voulons à ce qu'El Kibambo complète automatiquement ou non les colonnes non spécifiées ;
- ✚ columns : tableau d'objets KibamboColumn faisant référence à chaque colonne de la grille contenant les données ;
- ✚ searchHandler : délégué de type KibamboDelegate à appeler lors de la recherche, c'est-à-dire lors de l'appel de la méthode Kibambo.Search().
- ✚ cancelSearchHandler : délégué de type KibamboDelegate appelé par El Kibambo lorsqu'on veut annuler une recherche faite, c'est-à-dire lors de l'appel à la méthode Kibambo.CancelSearch().

Remarquons la présence d'un nouveau type : le délégué KibamboDelegate. Sa déclaration dans les méandres d'El Kibambo est la suivante :

```
delegate void ElKibambo.KibamboDelegate(System.Windows.Forms.DataGridViewRow row)
```

```
Delegate Sub ElKibambo.KibamboDelegate(row As System.Windows.Forms.DataGridViewRow)
```

Tout simplement une fonction qui ne renvoie rien et qui reçoit comme seul paramètre un objet de type DataGridViewRow représentant bien sûr une ligne de la grille contenant les données.

Mais pourquoi ? Rendez-vous lors de l'étude des méthodes Kibambo.Search() et Kibambo.CancelSearch() !

Il est à noter que :

- ✚ L'utilisation des constructeurs ne prenant pas en paramètre un tableau des objets KibamboColumn revient à laisser à la classe Kibambo d'utiliser les paramètres par défaut de chaque colonne de la grille contenant les données. Notez aussi qu'El Kibambo considèrera que la colonne est de type KibamboColumnType.String. Ceci, c'est si et seulement si la propriété Kibambo.AutoCompleteColumns vaut true. Néanmoins, si Kibambo.AutoCompleteColumns vaut false et que nous ne spécifions pas des colonnes, El Kibambo déclenche une erreur de type KibamboException avec comme code d'erreur KibamboErrorCode.ColumnsNotSpecified.
- ✚ Si la valeur de dgvData et celle de dgvSearch est la même, c'est-à-dire que nous passons au constructeur une même grille de données, et pour les données et pour la recherche, El Kibambo déclenche une erreur de type KibamboException avec comme code d'erreur KibamboErrorCode.DgvOfDataMustBeDifferentThanDgvOfSearch.
- ✚ Lors de l'initialisation (Kibambo.Initialize()), El Kibambo procède à la configuration de la grille de recherche. Par exemple, les propriétés suivantes sont désactivées (valeur passée à false) : AllowDrop, AllowUserToAddRows, AllowUserToDeleteRows, AllowUserToOrderColumns, AllowUserToResizeRows, RowHeadersVisible.

b. Propriétés

AutoCompleteColumns

```
bool Kibambo.AutoCompleteColumns { get; set; }
```

```
Property Kibambo.AutoCompleteColumns As Boolean
```

Cette propriété indique à El Kibambo de compléter automatiquement ou non les colonnes non spécifiées.

Ainsi, vous pouvez ne pas spécifier de colonnes et laisser la tâche à El Kibambo de les générer automatiquement sur base des colonnes de la grille contenant les données. Vous pouvez également ne spécifier qu'une partie de colonnes (si nécessaire, par exemple si certaines colonnes nécessitent certains opérateurs de comparaison et d'autres non), et de laisser à El Kibambo de compléter les colonnes non spécifiées.

Columns

```
KibamboColumn[] Kibambo.Columns { get; set; }
```

```
Property Kibambo.Columns As KibamboColumn()
```

La propriété Columns permet d'accéder en lecture comme en écriture aux objets KibamboColumn représentant chaque colonne de la grille contenant les données.

Grace à cette propriété, vous pourrez ne spécifier que certaines lignes de la grille contenant les données si vous voulez par exemple effectuer la recherche que sur certaines colonnes et d'autres non (bien sûr, la propriété Kibambo.AutoCompleteColumns doit valoir false dans ce cas).

ComparisonType

```
KibamboComparisonType Kibambo.ComparisonType { get; set; }
```

```
Property Kibambo.ComparisonType As KibamboComparisonType
```

Permet de spécifier le type de comparaison à utiliser lors de la recherche. La valeur par défaut de cette propriété est « KibamboComparisonType.IgnoreDiacriticSignsAndCase ».

RowsSatisfied et RowsUnsatisfied

```
List<DataGridViewRow> Kibambo.RowsSatisfied { get; }  
List<DataGridViewRow> Kibambo.RowsUnsatisfied { get; }
```

```
ReadOnly Property Kibambo.RowsSatisfied As List (Of DataGridView)  
ReadOnly Property Kibambo.RowsUnsatisfied As List (Of DataGridView)
```

Après une recherche grâce à la méthode `Kibambo.Search()`, la propriété `Kibambo.RowsSatisfied` contiendra toutes les lignes ayant satisfait aux critères de recherche et `Kibambo.RowsUnsatisfied` toutes les lignes n'ayant pas satisfait aux critères de recherche.

SearchHandler et CancelSearchHandler

```
KibamboDelegate Kibambo.SearchHandler { get; set; }
KibamboDelegate Kibambo.CancelSearchHandler { get; set; }
```

```
Property Kibambo.SearchHandler As KibamboDelegate
Property Kibambo.CancelSearchHandler As KibamboDelegate
```

Ces deux propriétés sont de type `KibamboDelegate` et représentent les fonctions appelées respectivement (si spécifiées) lors de l'appel aux méthodes `Kibambo.Search()` et `Kibambo.CancelSearch()`.

TitleField, TitleType, TitleOperator, TitleValue

Ces propriétés représentent les quatre en-têtes de la grille de données qui permettra de mettre en œuvre la recherche. C'est respectivement : « champ », « type », « opérateur », et « valeur ». Elles permettent juste de modifier les valeurs par défaut au besoin.

```
string Kibambo.TitleField { get ; set; }
string Kibambo.TitleType { get ; set; }
string Kibambo.TitleOperator { get ; set; }
string Kibambo.TitleValue { get ; set; }
```

```
Property Kibambo.TitleField As String
Property Kibambo.TitleType As String
Property Kibambo.TitleOperator As String
Property Kibambo.TitleValue As String
```

c. Méthodes

Search()

```
void Kibambo.Search()
```

```
Sub Kibambo.Search()
```

Cette méthode est appelée pour effectuer la recherche avec El Kibambo. Lors de la recherche, toutes les lignes ayant satisfait à la condition sont ajoutées dans la collection accessible grâce à la propriété `Kibambo.RowsSatisfied`. Celles n'ayant pas satisfait sont mis dans une autre collection accessible grâce à la propriété `Kibambo.RowsUnsatisfied`.

Lors de la recherche, si le délégué de recherche, `Kibambo.SearchHandler` a été spécifié, chaque ligne n'ayant pas satisfait aux critères de recherche sera passée à la fonction de traitement.

CancelSearch()

```
void Kibambo.CancelSearch()
```

```
Sub Kibambo.CancelSearch()
```

A l'appel de cette fonction, si la propriété `Kibambo.CancelSearchHandler` ne vaut pas null (c'est-à-dire qu'une fonction de rappel a été spécifiée) et que la collection `Kibambo.RowsUnsatisfied` n'est pas vide, les lignes n'ayant pas satisfait aux critères de recherche sont passées tour à tour au délégué `Kibambo.CancelSearchHandler`.

Il est ainsi raisonnable d'inverser ce qui a été fait dans le délégué de recherche. Par exemple, si lors de la recherche nous avons caché les lignes n'ayant pas satisfait aux critères de recherche, il paraît logique que dans le délégué `Kibambo.CancelSearchHandler`, il faut les réafficher.

Il est à noter que cette méthode vide les collections `Kibambo.RowsSatisfied` et `Kibambo.RowsUnsatisfied`.

Initialize()

```
void Kibambo.Initialize()
```

```
Sub Kibambo.Initialize()
```

Cette méthode est appelée pour initialiser/réinitialiser El Kibambo. Cette méthode est à appeler juste après avoir construit l'objet Kibambo ou après modification de ses propriétés.

🔗 Vous pouvez retrouver des exemples d'utilisation sur la playlist d'El Kibambo sur la chaîne Youtube d'El CONCEPT.

<https://youtube.com/playlist?list=PL7i4WkgFSFMIT75YwyJVNdON6Tzu8upT>

🔗 Lien de téléchargement de la bibliothèque :

<https://github.com/elconcept15/elkibambo/blob/main/winforms/elkibambo.dll>

🔗 En cas de besoin, [...] :

✉ Email : elconcept15@gmail.com

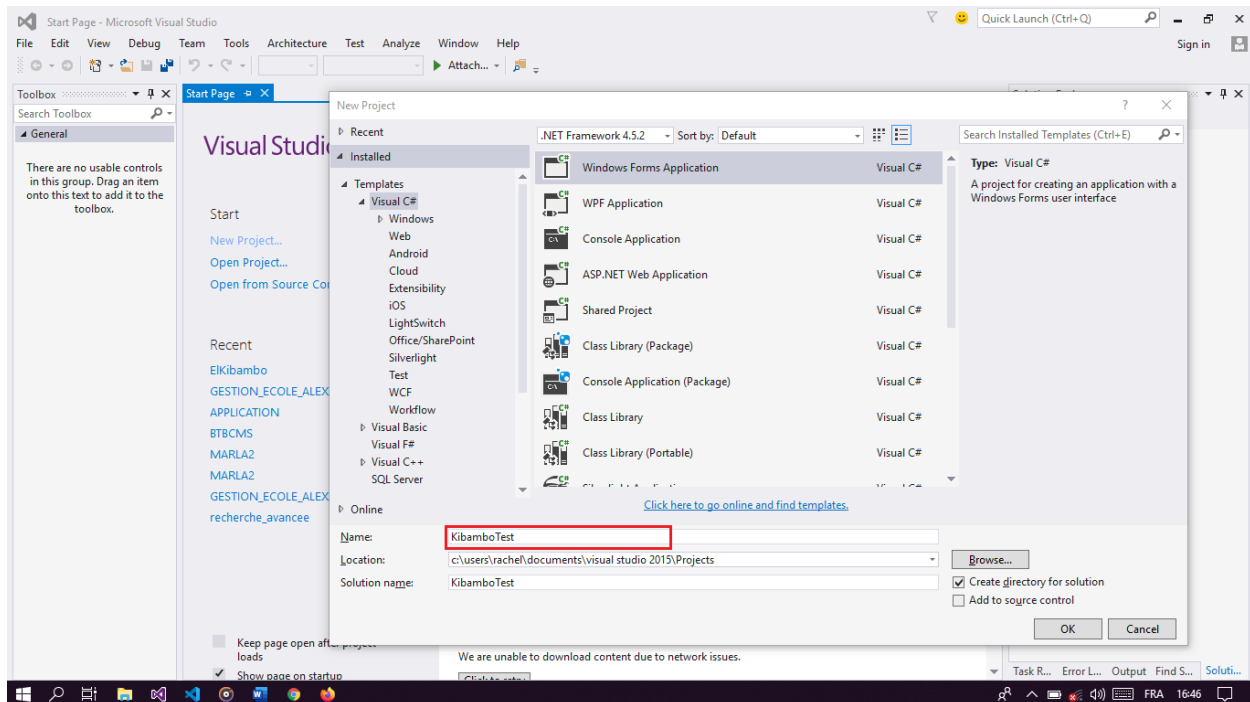
☎ Contact : +243 82 34 23 495 / +243 85 57 99 756

📘 Page Facebook : <https://web.facebook.com/elconcept15>

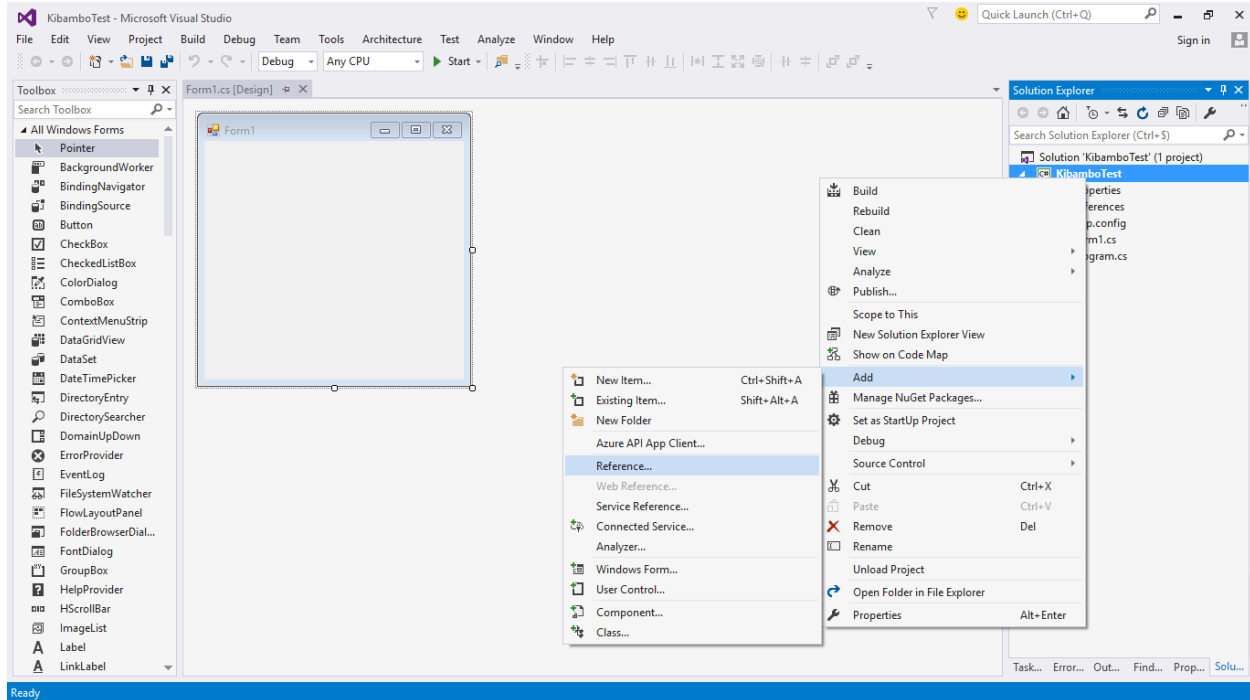
Mise en œuvre (C#)

Dans cette section, nous allons créer un programme C# Windows Forms illustrant l'implémentation de la bibliothèque El Kibambo. Nous allons utiliser comme IDE « Microsoft Visual Studio Enterprise 2015 ».

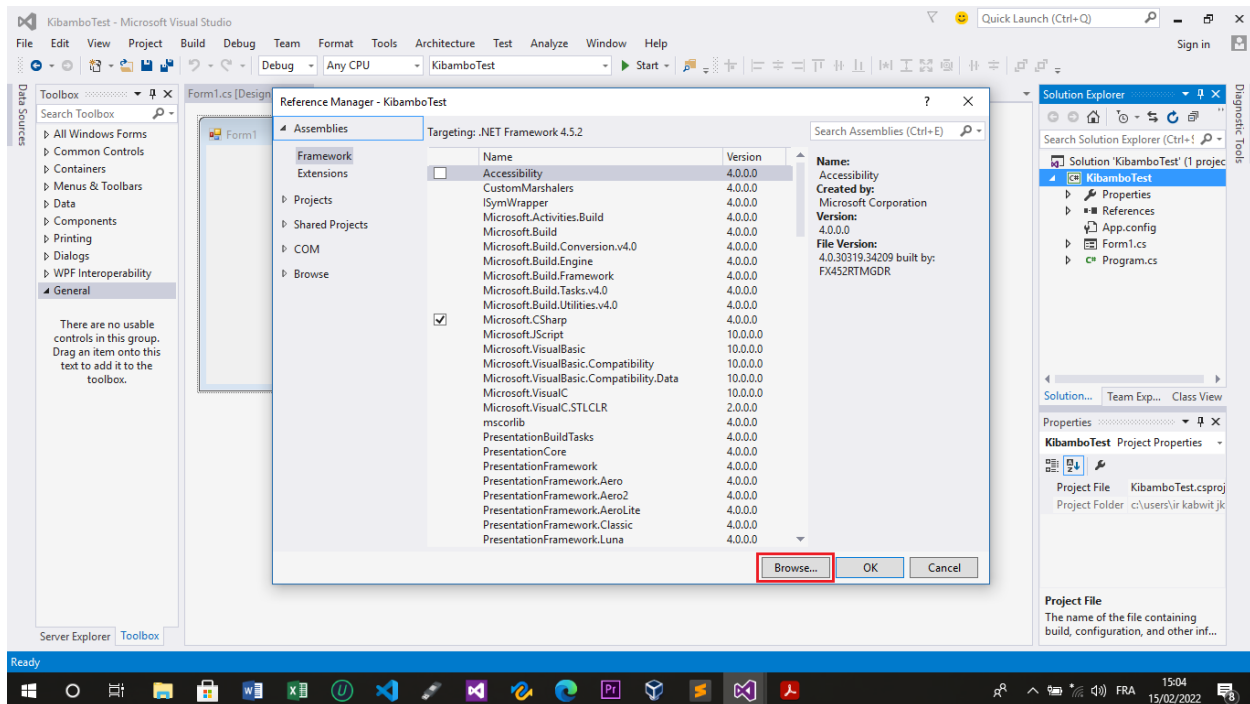
Créons un nouveau projet C# Windows Forms que nous nommons « KibamboTest ».



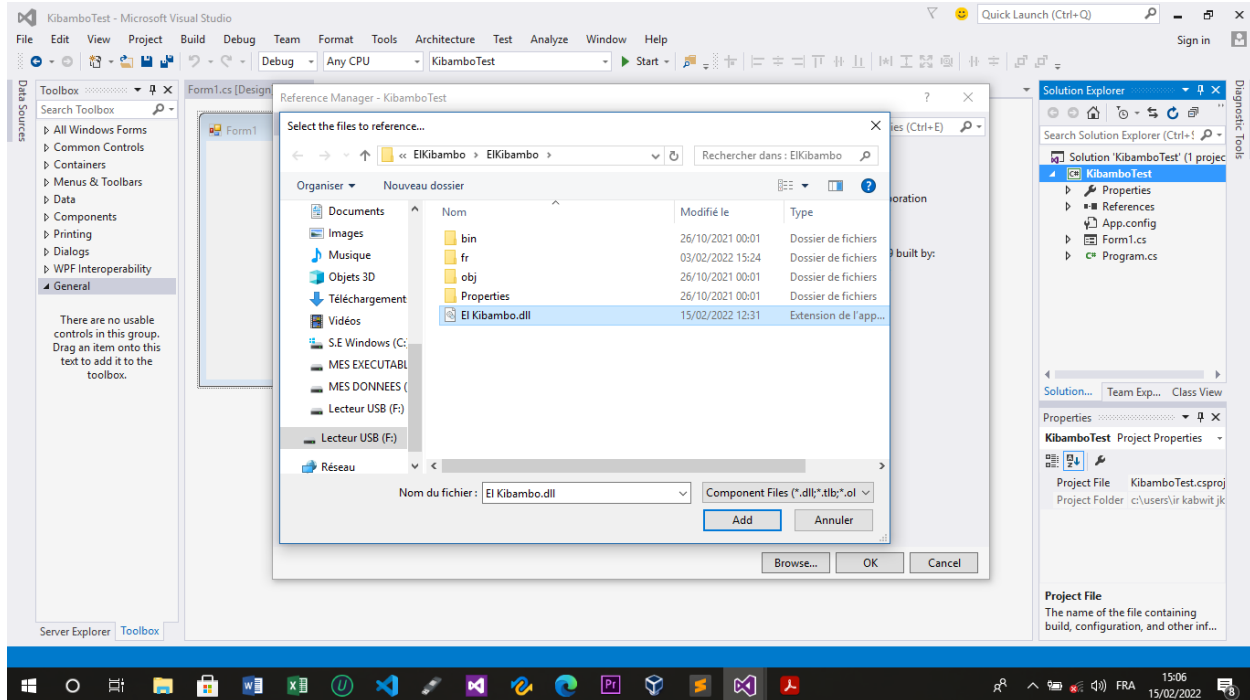
Ajoutons maintenant la référence à la bibliothèque El Kibambo. Pour cela, faites un clic droit sur le nom du projet, choisissez l'option « Ajouter », ensuite « Référence ... ».



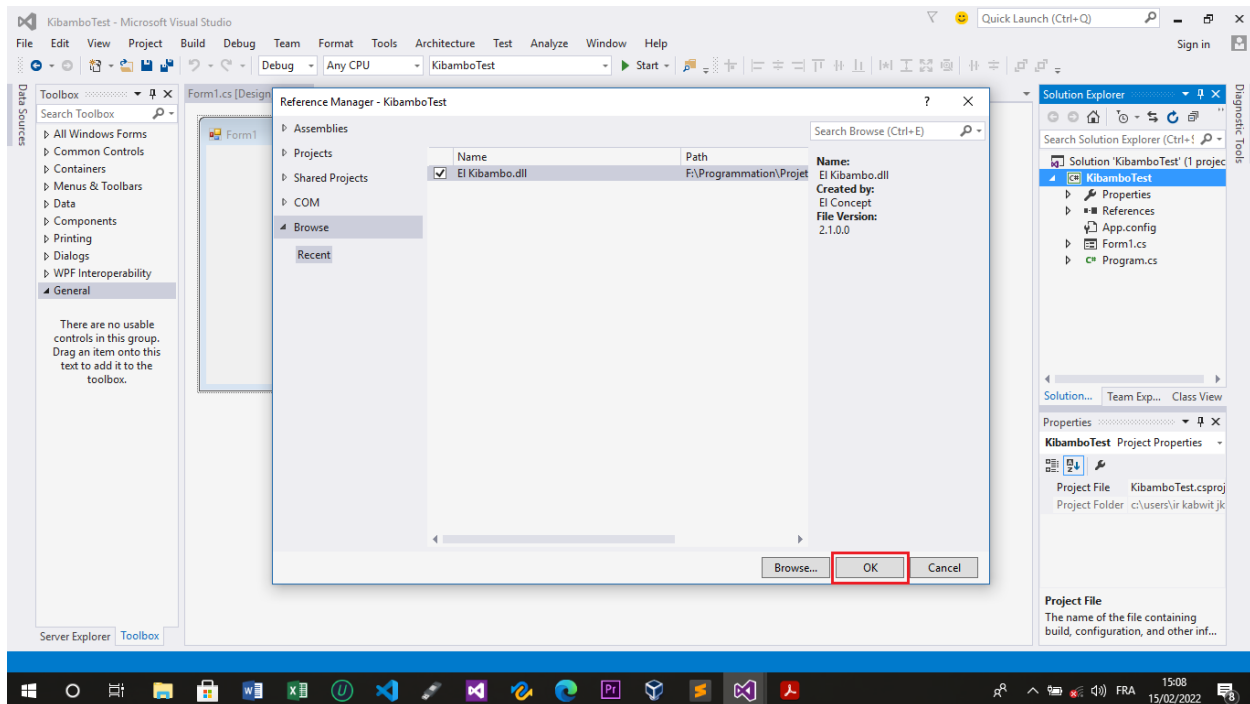
En cliquant sur l'option « Référence ... », Visual Studio vous affichera une interface semblable à :



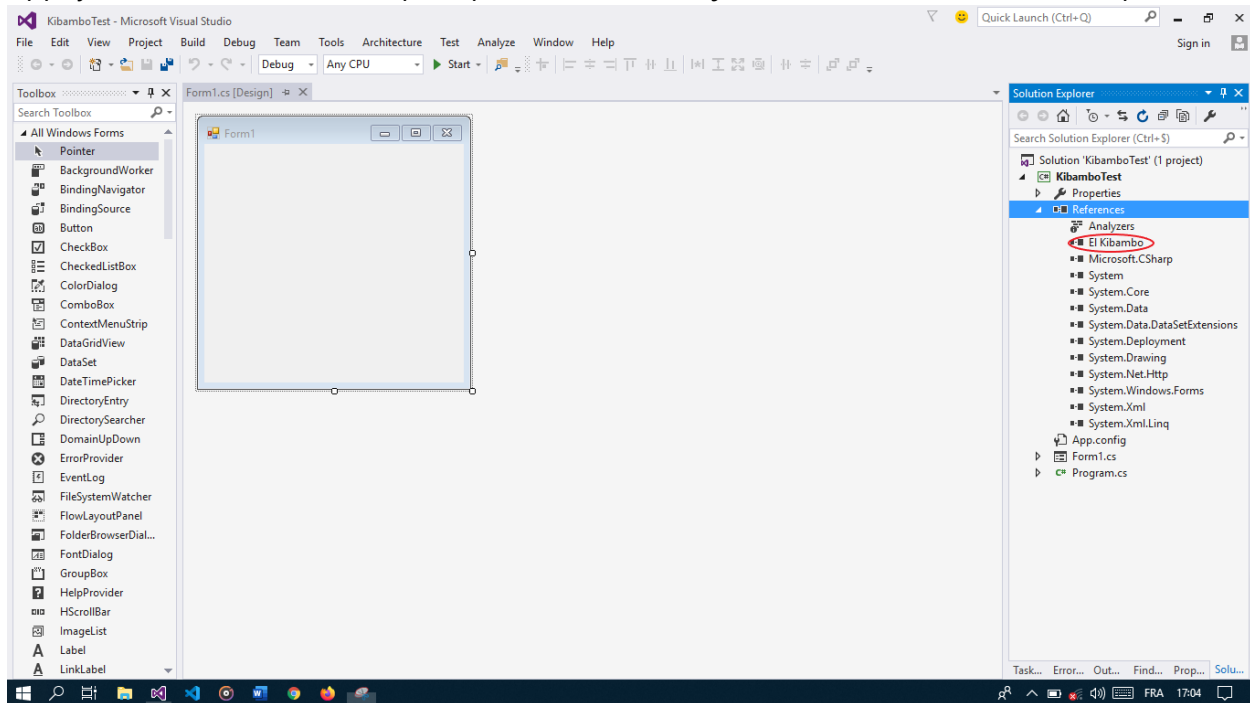
Appuyer sur le bouton « Parcourir ... », cela va vous afficher l'explorateur de fichiers afin de pouvoir choisir le fichier .dll de la bibliothèque.



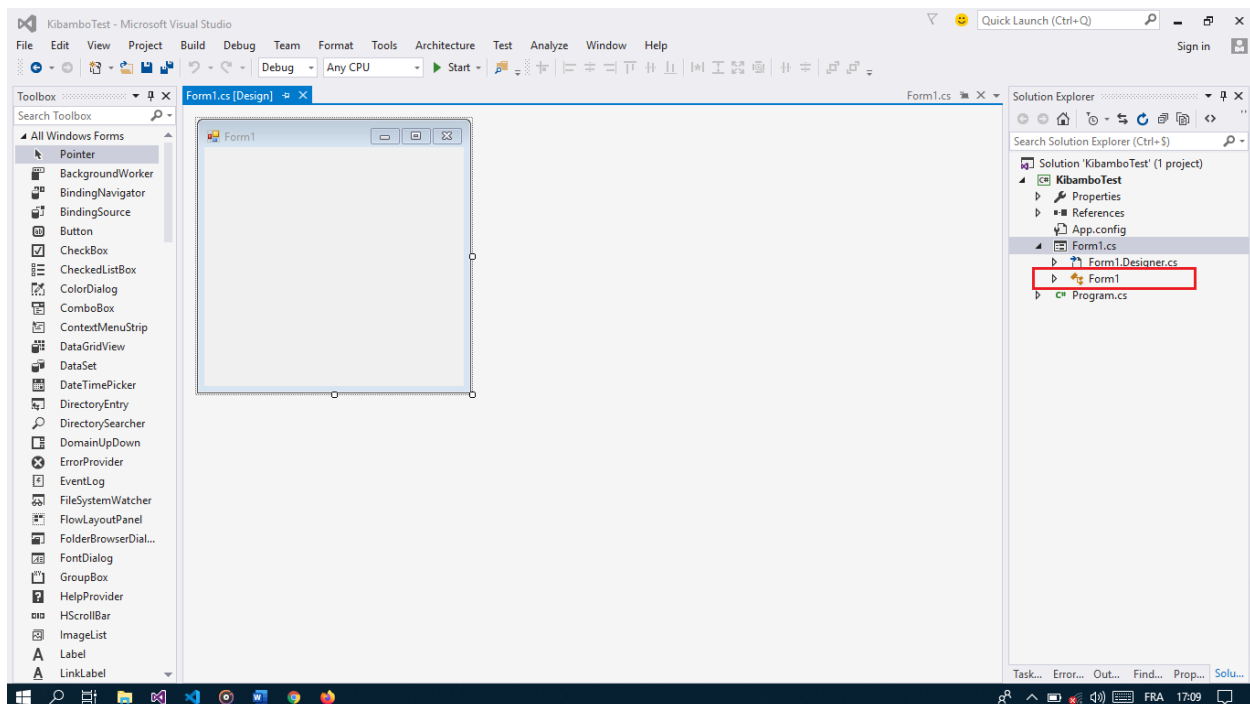
Appuyer sur le bouton « Ajouter », vous aurez une interface semblable à :



Appuyez sur le bouton « OK » pour pouvoir finaliser l'ajout de référence à la bibliothèque.



Maintenant, ouvrez le fichier « Form1.cs » :



Remplacez le code du fichier « Form1.cs » par le suivant :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```

using System.Drawing;
using System.Windows.Forms;
using ElKibambo;

namespace KibamboTest
{
    public partial class Form1 : Form
    {
        DataGridView dgvPersons;
        DataGridView dgvSearch;

        GroupBox gbSearch;

        Button btnShowAll;
        Button btnSearch;

        Kibambo kibambo;

        public Form1()
        {
            InitializeComponent();
            InitializeControls();
        }

        private void InitializeControls()
        {
            dgvPersons = new DataGridView();
            dgvSearch = new DataGridView();

            gbSearch = new GroupBox();

            btnShowAll = new Button();
            btnSearch = new Button();

            ((ISupportInitialize)(dgvPersons)).BeginInit();
            ((ISupportInitialize)(dgvSearch)).BeginInit();

            gbSearch.SuspendLayout();
            SuspendLayout();

            Controls.Clear();

            //
            // dgvcs1
            //
            var dgvcs1 = new DataGridViewCellStyle()
            {
                Alignment = DataGridViewContentAlignment.MiddleCenter,
                BackColor = SystemColors.Control,
                Font = new Font("Trebuchet MS", 9F, FontStyle.Bold, GraphicsUnit.Point, 0),
                ForeColor = SystemColors.WindowText,
                SelectionBackColor = SystemColors.Highlight,
                SelectionForeColor = SystemColors.HighlightText,
                WrapMode = DataGridViewTriState.True
            };

            //
            // dgvcs2
            //
            var dgvcs2 = new DataGridViewCellStyle()
            {

```

```

        Font = new Font("Segoe UI", 9F, FontStyle.Regular, GraphicsUnit.Point, 0)
    };

    //
    // dgvPersons
    //
    dgvPersons.AllowUserToAddRows = false;
    dgvPersons.AllowUserToDeleteRows = false;
    dgvPersons.AllowUserToResizeColumns = false;
    dgvPersons.AllowUserToResizeRows = false;
    dgvPersons.Anchor = ((AnchorStyles.Top | AnchorStyles.Bottom) | AnchorStyles.Left)
| AnchorStyles.Right;
    dgvPersons.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dgvPersons.BackgroundColor = Color.White;
    dgvPersons.BorderStyle = BorderStyle.Fixed3D;
    dgvPersons.ColumnHeadersDefaultCellStyle = dgvcs1;
    dgvPersons.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize;
    dgvPersons.EditMode = DataGridViewEditMode.EditProgrammatically;
    dgvPersons.Location = new Point(12, 12);
    dgvPersons.MultiSelect = false;
    dgvPersons.RowHeadersVisible = false;
    dgvPersons.RowsDefaultCellStyle = dgvcs2;
    dgvPersons.ScrollBars = ScrollBars.Vertical;
    dgvPersons.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dgvPersons.Size = new Size(470, 358);
    dgvPersons.TabIndex = 0;

    //
    // dgvSearch
    //
    dgvSearch.Anchor = (AnchorStyles.Top | AnchorStyles.Left) | AnchorStyles.Right;
    dgvSearch.BackgroundColor = Color.White;
    dgvSearch.BorderStyle = BorderStyle.Fixed3D;
    dgvSearch.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize;
    dgvSearch.Location = new Point(10, 19);
    dgvSearch.Size = new Size(268, 111);
    dgvSearch.TabIndex = 1;

    //
    // btnShowAll
    //
    btnShowAll.Anchor = AnchorStyles.Top | AnchorStyles.Right;
    btnShowAll.Location = new Point(106, 136);
    btnShowAll.Size = new Size(83, 23);
    btnShowAll.TabIndex = 3;
    btnShowAll.Text = "Tous afficher";
    btnShowAll.UseVisualStyleBackColor = true;
    btnShowAll.Click += btnShowAll_Click;

    //
    // btnSearch
    //
    btnSearch.Anchor = AnchorStyles.Top | AnchorStyles.Right;
    btnSearch.Location = new Point(195, 136);
    btnSearch.Size = new Size(83, 23);
    btnSearch.TabIndex = 2;
    btnSearch.Text = "Chercher";
    btnSearch.UseVisualStyleBackColor = true;
    btnSearch.Click += btnSearch_Click;

```

```

    //
    // gbSearch
    //
    gbSearch.Anchor = AnchorStyles.Top | AnchorStyles.Right;
    gbSearch.Controls.Add(btnShowAll);
    gbSearch.Controls.Add(btnSearch);
    gbSearch.Controls.Add(dgvSearch);
    gbSearch.Font = new Font("Segoe UI", 8.25F, FontStyle.Regular, GraphicsUnit.Point,
0);

    gbSearch.Location = new Point(490, 12);
    gbSearch.Size = new Size(286, 170);
    gbSearch.TabIndex = 2;
    gbSearch.TabStop = false;
    gbSearch.Text = "Recherche";

    //
    // Form1
    //
    AutoScaleDimensions = new.SizeF(6F, 13F);
    AutoScaleMode = AutoScaleMode.Font;
    BackColor = Color.White;
    ClientSize = new Size(789, 385);
    Controls.Add(gbSearch);
    Controls.Add(dgvPersons);
    Text = "El Kibambo - Illustration";
    Load += new EventHandler(Form1_Load);
    StartPosition = FormStartPosition.CenterScreen;

    ((ISupportInitialize)(dgvPersons)).EndInit();
    ((ISupportInitialize)(dgvSearch)).EndInit();

    gbSearch.ResumeLayout(false);
    ResumeLayout(false);
}

private void InitializeDgvPersonnes()
{
    // Ajout des colonnes
    dgvPersons.Columns.Add("registration_number", "Matricule");
    dgvPersons.Columns.Add("first_name", "Prénom");
    dgvPersons.Columns.Add("last_name", "Nom");
    dgvPersons.Columns.Add("age", "Age");

    // Objets "Person" à afficher dans la grille de données
    var persons = new List<Person>
    {
        new Person("0001", "Michael", "Kyungu Ilunga", 21),
        new Person("0002", "Emmanuel", "Ilunga Ndalamba", 25),
        new Person("0003", "Jonathan", "Mulimbi Somwe", 22),
        new Person("0004", "Gloria", "Ngombe Kibambo", 20),
        new Person("0005", "Achille", "Mutombo Mubakilay", 22),
        new Person("0006", "Marc", "Kyalika Musomena", 23),
        new Person("0007", "Martin", "Manama Kabeya", 26),
        new Person("0008", "Ornellah", "Masengo Mutoni", 20),
        new Person("0009", "Elie", "Ebukeya Tshombe", 22),
        new Person("0010", "Françoise", "Ebukeya Lukonde", 15),
        new Person("0011", "Hansvané", "Kashala Kahilu", 23),
        new Person("0012", "Benoit", "Kamona Bunake", 23),
        new Person("0013", "Samuel", "Ngandu Mwepu", 21),
        new Person("0014", "Jules", "Malemo Miyayo", 23),
    }
}

```



```

        new Person("0015", "Rachel", "Wany Mukanire", 23)
    };

    // Ajout des données dans la grille de données
    foreach (Person person in persons)
    {
        dgvPersons.Rows.Add(
            person.RegistrationNumber,
            person.FirstName,
            person.LastName,
            person.Age
        );
    }
}

private void InitializeKibambo()
{
    kibambo = new Kibambo(dgvPersons, dgvSearch, true)
    {
        Columns = new KibamboColumn[]
        {
            new KibamboColumn("registration_number", "Matricule")
            {
                Operators = new[] { "==", "!=" }
            },

            new KibamboColumn("age", "Age", KibamboColumnType.Numeric, "Entier", new
string[] { "==", "<", ">", "<=", ">=", "!=" })
        },

        SearchHandler = delegate (DataGridViewRow row)
        {
            // On choisit de rendre invisible les lignes n'ayant pas satisfait aux
critères de recherche.
            // Mai vous pouvez néanmoins faire autre chose, par exemple changer la
couleur, etc.
            row.Visible = false;
        },

        CancelSearchHandler = delegate (DataGridViewRow row)
        {
            // Comme nous avons masqué les lignes n'ayant pas satisfait aux critères
de recherche lors de la recherche, il semble logique que nous puissions les ré-affichées
lorsqu'on voudra annuler la recherche faite.
            // Nous devons donc mettre la ligne à son état initial avant la recherche.
            row.Visible = true;
        }
    };

    kibambo.Initialize();
}

private void Form1_Load(object sender, EventArgs e)
{
    InitializeDgvPersonnes();
    InitializeKibambo();
}

private void btnSearch_Click(object sender, EventArgs e)
{
    try

```

```

    {
        kibambo.Search();
    }

    catch (KibamboException ex)
    {
        switch (ex.ErrorCode)
        {
            case KibamboErrorCode.ValueMustBeNumeric:
                MessageBox.Show("Vous devez saisir une valeur de type numérique pour la colonne de type numérique !", "Erreur de saisie", MessageBoxButtons.OK, MessageBoxIcon.Error);
                break;

            default:
                MessageBox.Show("Une erreur est survenue lors de la recherche. Le message d'erreur est " + ex.Message + " !", "Erreur lors de la recherche", MessageBoxButtons.OK, MessageBoxIcon.Error);
                break;
        }
    }
}

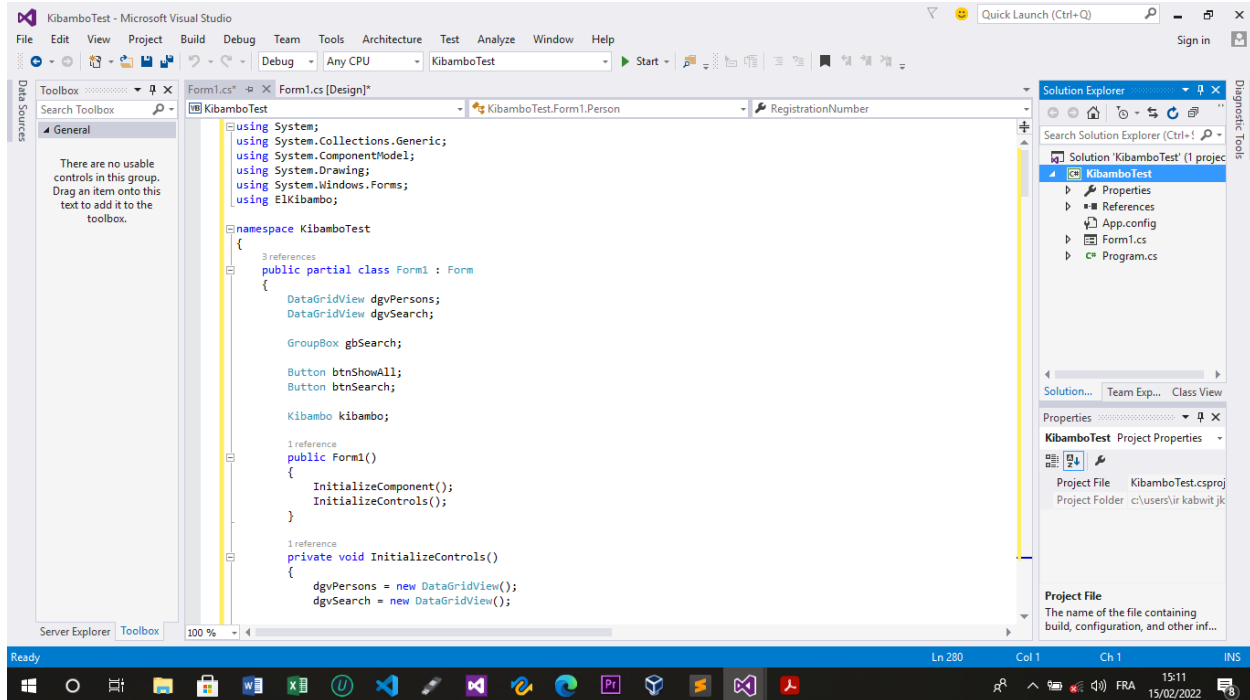
private void btnShowAll_Click(object sender, EventArgs e)
{
    kibambo.CancelSearch();
}

public class Person
{
    public string RegistrationNumber { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Age { get; set; }

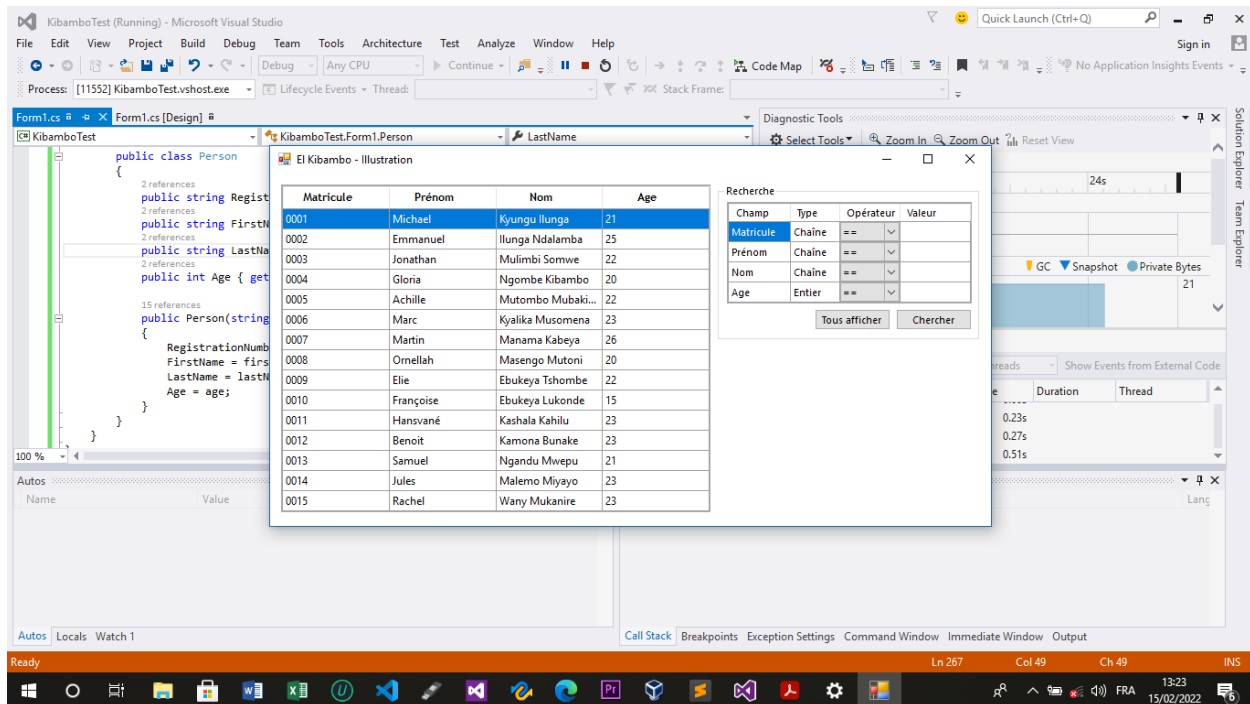
    public Person(string registrationNumber, string firstName, string lastName, int age)
    {
        RegistrationNumber = registrationNumber;
        FirstName = firstName;
        LastName = lastName;
        Age = age;
    }
}
}

```

Exécutez l'application en appuyant sur le bouton « Démarrer » ou grâce à la touche de fonction F5 :



Résultat :



On pouvait utiliser une syntaxe concise grâce aux expressions lambda lors de l'instanciation de l'objet Kibambo :

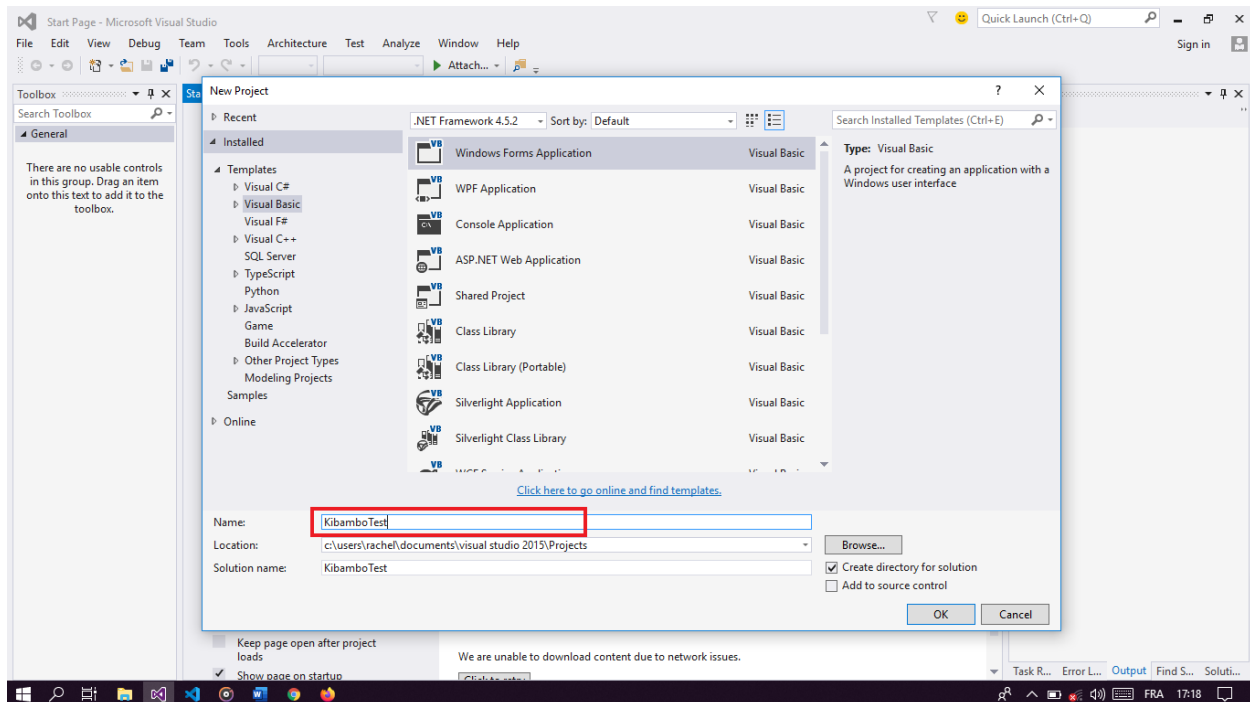
```
kibambo = new Kibambo(dgvPersons, dgvSearch, true)
{
    Columns = new KibamboColumn[]
```

```
{
    new KibamboColumn("registration_number", "Matricule")
    {
        Operators = new[] { "=", "!=" }
    },
    new KibamboColumn("age", "Age", KibamboColumnType.Numeric, "Entier", new string[] {
"==", "<", ">", "<=", ">=", "!=" })
    },
    SearchHandler = (row) => row.Visible = false,
    CancelSearchHandler = (row) => row.Visible = true
};
```

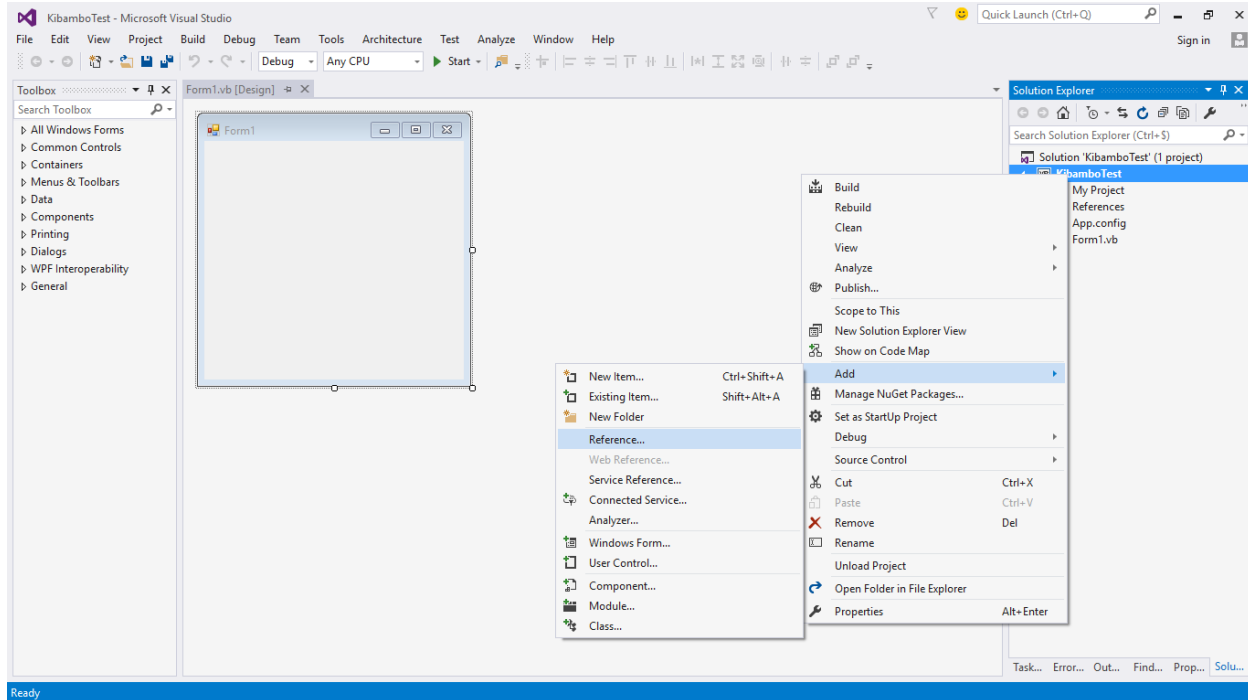
Mise en œuvre (VB.NET)

Dans cette section, nous allons créer un programme VB.NET Windows Forms illustrant l'implémentation de la bibliothèque El Kibambo. Nous allons utiliser comme IDE « Microsoft Visual Studio Enterprise 2015 ».

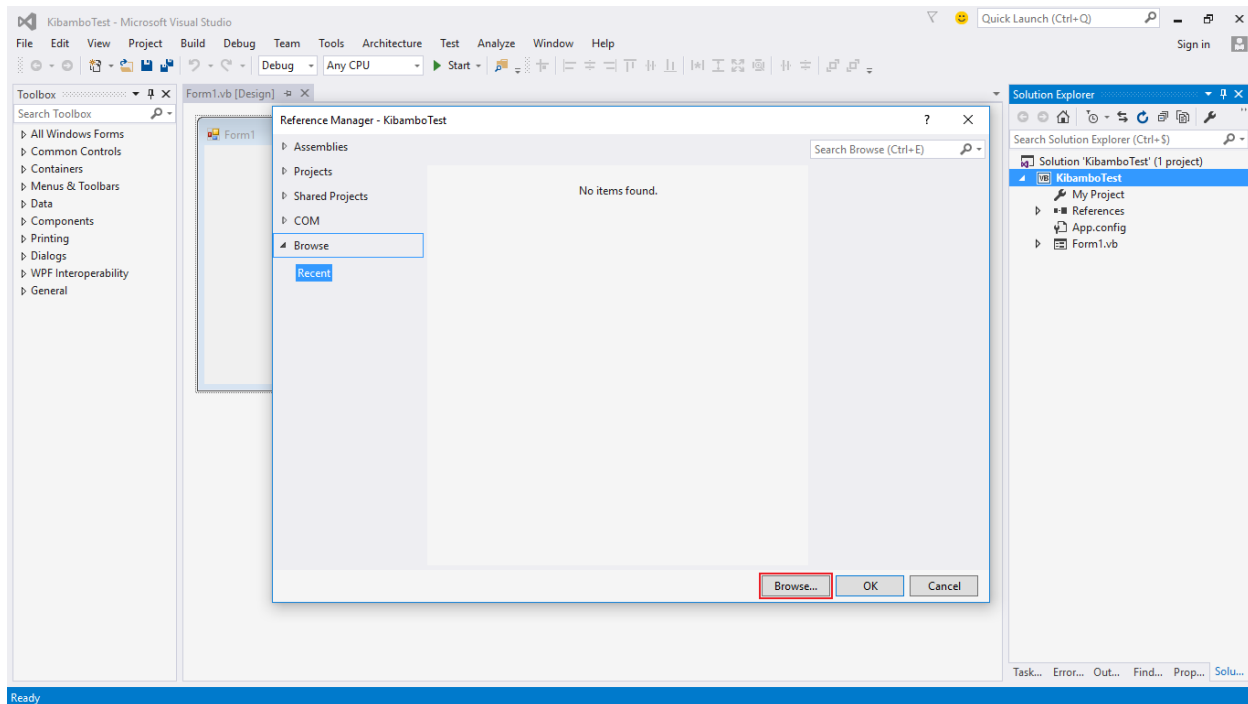
Créons un nouveau projet VB.NET Windows Forms que nous nommons « KibamboTest ».



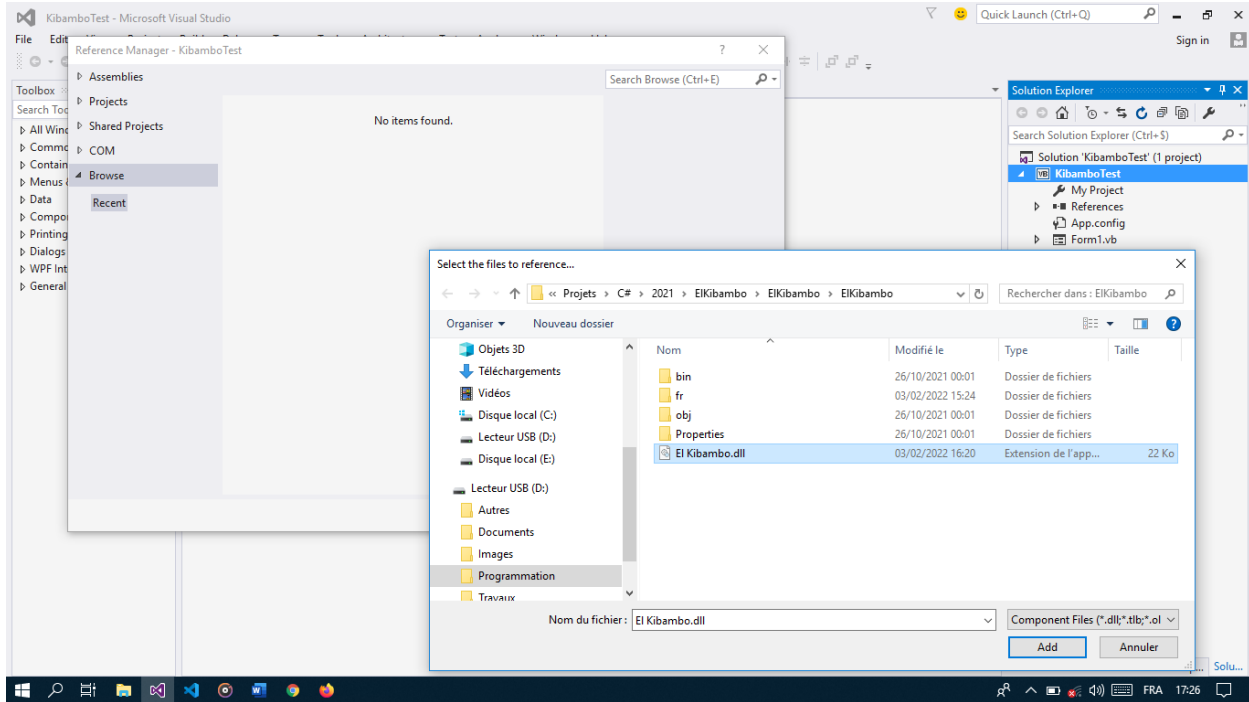
Ajoutons maintenant la référence à la bibliothèque El Kibambo. Pour cela, faites un clic droit sur le nom du projet, choisissez l'option « Ajouter », ensuite « Référence ... ».



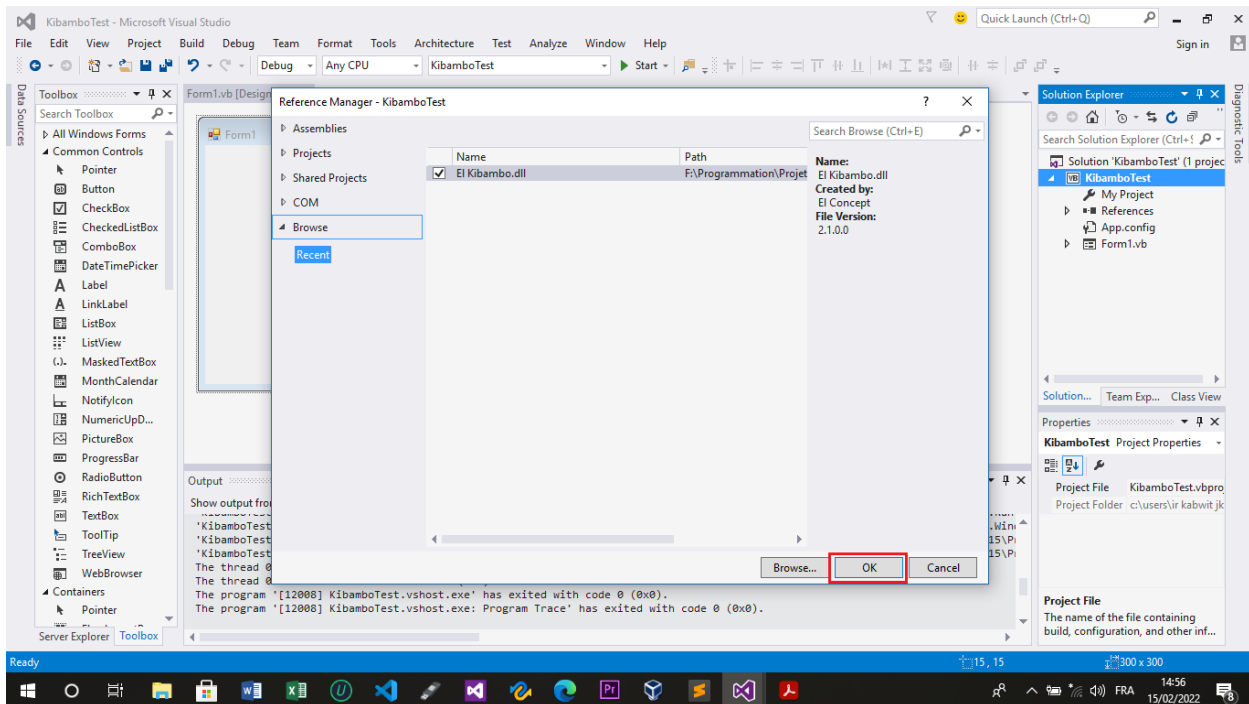
En cliquant sur l'option « Référence ... », Visual Studio vous affichera une interface semblable à :



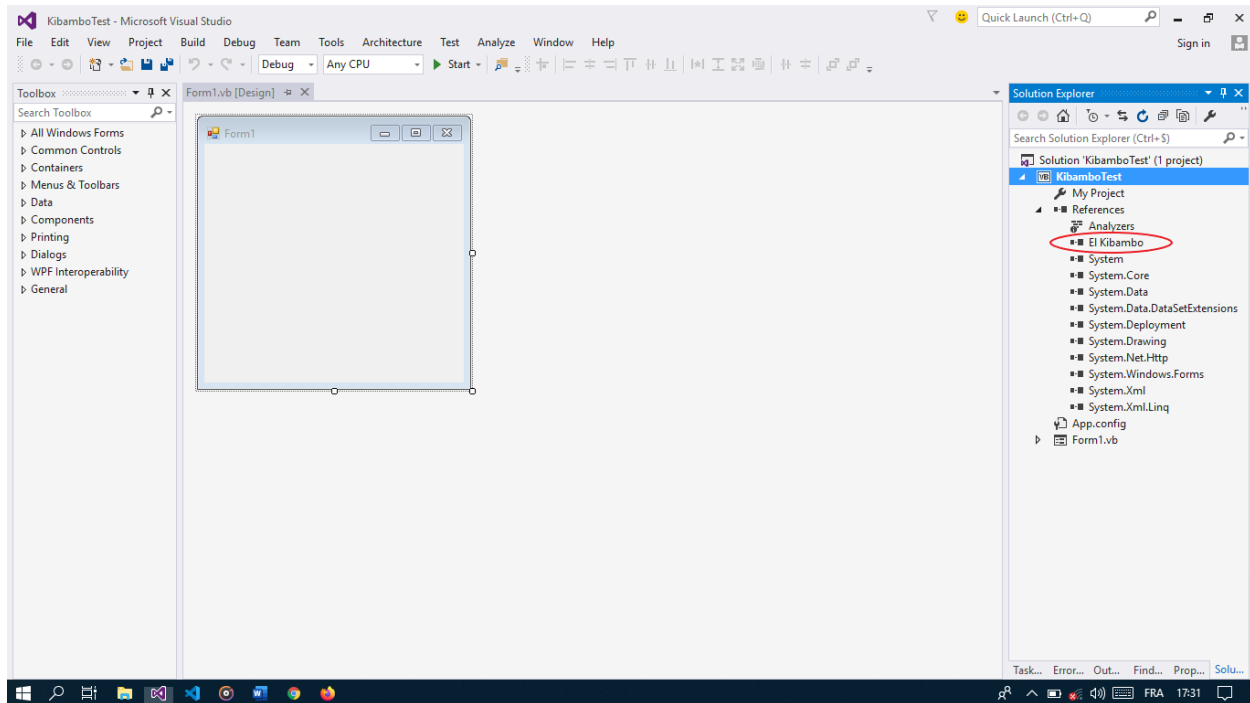
Appuyer sur le bouton « Parcourir ... », cela va vous afficher l'explorateur de fichiers afin de pouvoir choisir le fichier .dll de la bibliothèque.



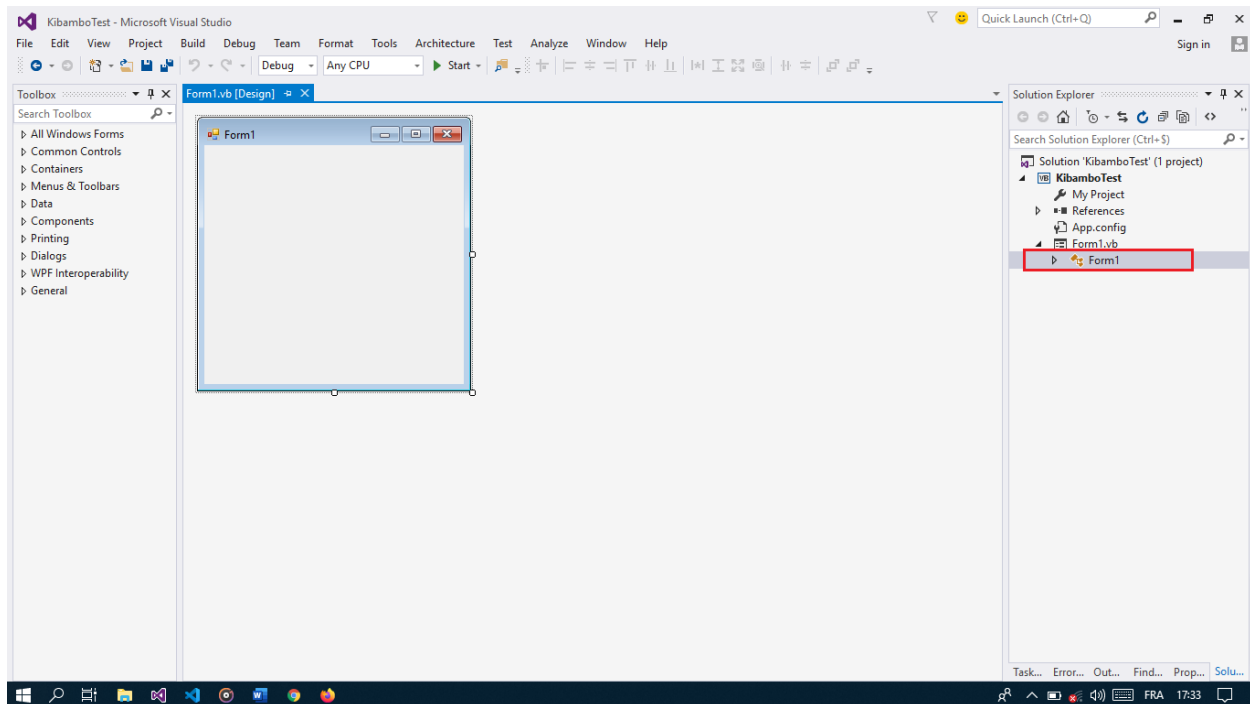
Appuyer sur le bouton « Ajouter », vous aurez l'interface suivante :



Appuyez sur le bouton « OK » pour pouvoir finaliser l'ajout de référence à la bibliothèque.



Maintenant, ouvrez le fichier « Form1.vb » :



Remplacez le code du fichier « Form1.vb » par le suivant :

```
Imports ElKibambo
Imports System.ComponentModel

Public Class Form1
    Dim dgvPersons As DataGridView
```



```

Dim dgvSearch As DataGridView

Dim gbSearch As GroupBox

Dim btnShowAll As Button
Dim btnSearch As Button

Dim kibambo As Kibambo

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    InitializeControls()
    InitializeDgvData()
    LoadData()
    InitializeKibambo()
End Sub

Private Sub InitializeControls()
    Dim dgvcs1 As New DataGridViewCellStyle
    Dim dgvcs2 As New DataGridViewCellStyle

    dgvPersons = New DataGridView()
    dgvSearch = New DataGridView()

    gbSearch = New GroupBox()

    btnShowAll = New Button()
    btnSearch = New Button()

    CType(dgvPersons, ISupportInitialize).BeginInit()
    CType(dgvSearch, ISupportInitialize).BeginInit()

    gbSearch.SuspendLayout()
    SuspendLayout()

    Controls.Clear()

    '
    ' dgvcs1
    '
    dgvcs1.Alignment = DataGridViewContentAlignment.MiddleCenter
    dgvcs1.BackColor = SystemColors.Control
    dgvcs1.Font = New Font("Trebuchet MS", 9.0!, FontStyle.Bold, GraphicsUnit.Point,
CType(0, Byte))
    dgvcs1.ForeColor = SystemColors.WindowText
    dgvcs1.SelectionBackColor = SystemColors.Highlight
    dgvcs1.SelectionForeColor = SystemColors.HighlightText
    dgvcs1.WrapMode = DataGridViewTriState.True

    '
    ' dgvcs2
    '
    dgvcs2.Font = New Font("Segoe UI", 9.0!, FontStyle.Regular, GraphicsUnit.Point,
CType(0, Byte))

    '
    ' dgvPersons
    '
    dgvPersons.AllowUserToAddRows = False
    dgvPersons.AllowUserToDeleteRows = False
    dgvPersons.AllowUserToResizeColumns = False
    dgvPersons.AllowUserToResizeRows = False

```

```

dgvPersons.Anchor = CType((((AnchorStyles.Top Or AnchorStyles.Bottom) _
    Or AnchorStyles.Left) _
    Or AnchorStyles.Right), AnchorStyles)
dgvPersons.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill
dgvPersons.BackgroundColor = Color.White
dgvPersons.BorderStyle = BorderStyle.Fixed3D
dgvPersons.ColumnHeadersDefaultCellStyle = dgvcs1
dgvPersons.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize
dgvPersons.EditMode = DataGridViewEditMode.EditProgrammatically
dgvPersons.Location = New Point(12, 12)
dgvPersons.MultiSelect = False
dgvPersons.RowHeadersVisible = False
dgvPersons.RowsDefaultCellStyle = dgvcs2
dgvPersons.ScrollBars = ScrollBars.Vertical
dgvPersons.SelectionMode = DataGridViewSelectionMode.FullRowSelect
dgvPersons.Size = New Size(433, 336)
dgvPersons.TabIndex = 0

'
' dgvSearch
'

dgvSearch.Anchor = CType((AnchorStyles.Top Or AnchorStyles.Left) _
    Or AnchorStyles.Right), AnchorStyles)
dgvSearch.BackgroundColor = Color.White
dgvSearch.BorderStyle = BorderStyle.Fixed3D
dgvSearch.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize
dgvSearch.Location = New Point(10, 19)
dgvSearch.Size = New Size(268, 111)
dgvSearch.TabIndex = 1

'
' btnShowAll
'

btnShowAll.Anchor = CType((AnchorStyles.Top Or AnchorStyles.Right), AnchorStyles)
btnShowAll.Location = New Point(106, 136)
btnShowAll.Size = New Size(83, 23)
btnShowAll.TabIndex = 3
btnShowAll.Text = "Tous afficher"
btnShowAll.UseVisualStyleBackColor = True
AddHandler btnShowAll.Click, AddressOf btnShowAll_Click

'
' btnSearch
'

btnSearch.Anchor = CType((AnchorStyles.Top Or AnchorStyles.Right), AnchorStyles)
btnSearch.Location = New Point(195, 136)
btnSearch.Size = New Size(83, 23)
btnSearch.TabIndex = 2
btnSearch.Text = "Chercher"
btnSearch.UseVisualStyleBackColor = True
AddHandler btnSearch.Click, AddressOf btnSearch_Click

'
' gbSearch
'

gbSearch.Anchor = CType((AnchorStyles.Top Or AnchorStyles.Right), AnchorStyles)
gbSearch.Controls.Add(btnShowAll)
gbSearch.Controls.Add(btnSearch)
gbSearch.Controls.Add(dgvSearch)

```

```

        gbSearch.Font = New Font("Segoe UI", 8.25!, FontStyle.Regular, GraphicsUnit.Point,
CType(0, Byte))
        gbSearch.Location = New Point(451, 12)
        gbSearch.Size = New Size(286, 170)
        gbSearch.TabIndex = 2
        gbSearch.TabStop = False
        gbSearch.Text = "Recherche"

        '
        ' Form1
        '

        AutoScaleDimensions = New.SizeF(6.0!, 13.0!)
        AutoScaleMode = AutoScaleMode.Font
        BackColor = Color.White
        ClientSize = New Size(750, 360)
        Controls.Add(gbSearch)
        Controls.Add(dgvPersons)
        Text = "El Kibambo - Illustration"
        StartPosition = FormStartPosition.CenterScreen
        CType(dgvPersons, ISupportInitialize).EndInit()
        CType(dgvSearch, ISupportInitialize).EndInit()
        gbSearch.ResumeLayout(False)
        ResumeLayout(False)

End Sub

Private Sub InitializeDgvData()
    dgvPersons.Columns.Add("registration_number", "Matricule")
    dgvPersons.Columns.Add("first_name", "Prénom")
    dgvPersons.Columns.Add("last_name", "Nom")
    dgvPersons.Columns.Add("age", "Age")
End Sub

Private Sub LoadData()
    dgvPersons.Rows.Add("0001", "Michael", "Kyungu Ilunga", 21)
    dgvPersons.Rows.Add("0002", "Emmanuel", "Ilunga Ndalamba", 25)
    dgvPersons.Rows.Add("0003", "Jonathan", "Mulimbi Somwe", 22)
    dgvPersons.Rows.Add("0004", "Gloria", "Ngombe Kibambo", 20)
    dgvPersons.Rows.Add("0005", "Achille", "Mutombo Mubakilay", 22)
    dgvPersons.Rows.Add("0006", "Marc", "Kyalika Musomena", 23)
    dgvPersons.Rows.Add("0007", "Martin", "Manama Kabeya", 26)
    dgvPersons.Rows.Add("0008", "Ornellah", "Masengo Mutoni", 20)
    dgvPersons.Rows.Add("0009", "Elie", "Ebukeya Tshombe", 22)
    dgvPersons.Rows.Add("0010", "Françoise", "Ebukeya Lukonde", 15)
    dgvPersons.Rows.Add("0011", "Hansvané", "Kashala Kahilu", 23)
    dgvPersons.Rows.Add("0012", "Benoit", "Kamona Bunake", 23)
    dgvPersons.Rows.Add("0013", "Samuel", "Ngandu Mwepu", 21)
    dgvPersons.Rows.Add("0014", "Jules", "Malemo Miyayo", 23)
    dgvPersons.Rows.Add("0015", "Rachel", "Wany Mukanire", 23)
End Sub

Private Sub Search(row As DataGridViewRow)
    ' On choisit de rendre invisible les lignes n'ayant pas satisfait aux critères de
recherche.
    ' Mais vous pouvez également faire autre chose, par exemple changer la couleur, etc.
    row.Visible = False
End Sub

Private Sub ShowAll(row As DataGridViewRow)
    ' Comme les lignes ont été cachées lors de la recherche, il paraît logique que l'on
puisse les réafficher lorsqu'on décide d'annuler une recherche faite.

```

```

        ' Il faut donc remettre la ligne à son état initial avant la recherche.
        row.Visible = True
    End Sub

    Private Sub InitializeKibambo()
        Dim kcm As New KibamboColumn("registration_number", "Matricule")
        kcm.Operators = {"==", "!="}

        Dim kca As New KibamboColumn("age", "Age", KibamboColumnType.Numeric, "Entier", {"==",
"<", ">", "<=", ">=", "!="})

        kibambo = New Kibambo(
            dgvPersons,
            dgvSearch,
            New KibamboColumn() {
                kcm,
                kca
            },
            AddressOf Search,
            AddressOf ShowAll,
            True
        )

        kibambo.Initialize()
    End Sub

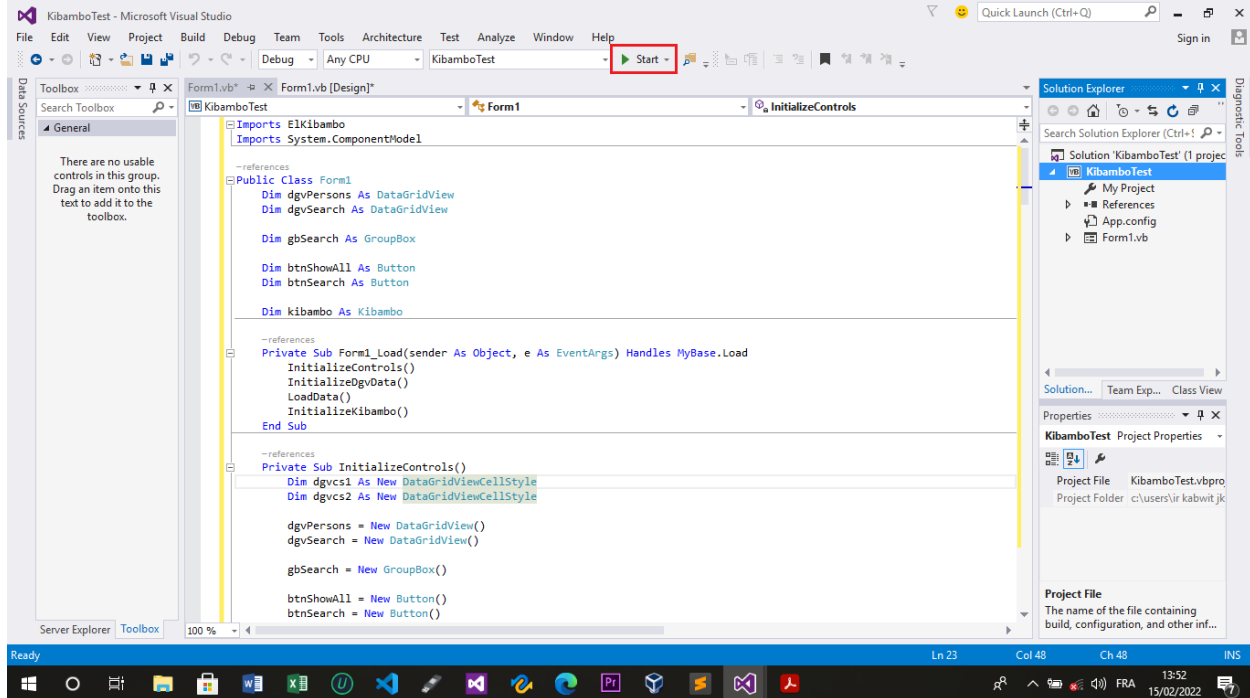
    Private Sub btnSearch_Click(sender As Object, e As EventArgs)
        Try
            kibambo.Search()
        Catch ex As KibamboException
            Select Case ex.ErrorCode
                Case KibamboErrorCode.ValueMustBeNumeric
                    MessageBox.Show("Vous devez saisir une valeur de type numérique pour la
colonne de type numérique !", "Erreur de saisie", MessageBoxButtons.OK, MessageBoxIcon.Error)

                Case Else
                    MessageBox.Show("Une erreur est survenue lors de la recherche. Le message
d'erreur est " + ex.Message + " !", "Erreur de saisie", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            End Select
        End Try
    End Sub

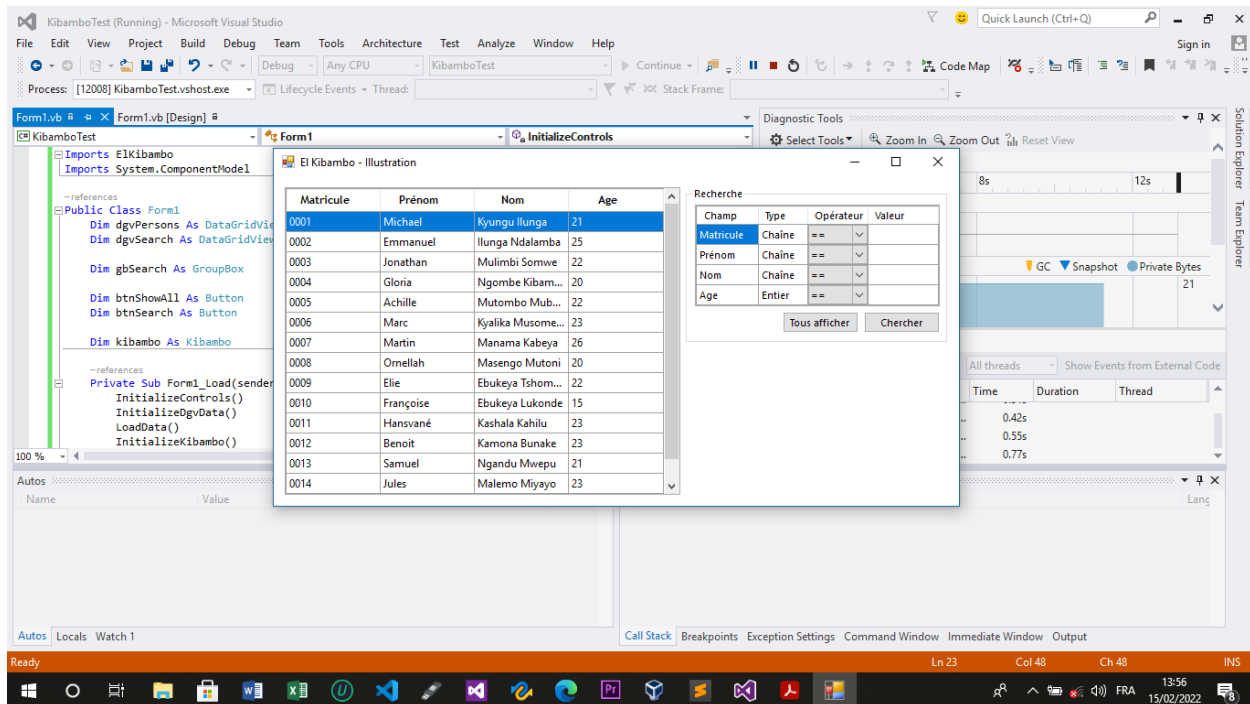
    Private Sub btnShowAll_Click(sender As Object, e As EventArgs)
        kibambo.CancelSearch()
    End Sub
End Class

```

Exécutez l'application en appuyant sur le bouton « Démarrer » ou grâce à la touche « F5 » :



Résultat :



Historique

La bibliothèque El Kibambo est née d'un constat fait par Michael KYUNGU ILUNGA, alors étudiant en deuxième graduat de l'Institut Supérieur Pédagogique de Lubumbashi. Son constat était la façon dont étaient implémentées les fonctionnalités de recherche dans les programmes Windows Forms. Il a constaté la façon dont ses collègues et étudiants de promotion (supérieure également) arrivait à implémenter les fonctionnalités de recherche, et leur façon de procéder ne donnait pas à l'utilisateur final la possibilité d'effectuer des recherches multicolones sur des grilles de données dans les programmes .NET Windows Forms avec le langage de programmation VB.NET, mais aussi C#.

Au fait, imaginons une grille de données contenant les données sur des personnes, par exemple « matricule », « nom », « prénom ». A l'ISP/L'SHI, les fonctionnalités de recherche ne permettaient qu'une recherche sur la colonne servant d'identifiant, comme la colonne « matricule » dans l'exemple ci-dessus. Pour d'autres, vous avez la possibilité d'effectuer des recherches sur n'importe quelle colonne, néanmoins sur une et une seule colonne. Ainsi dans l'exemple ci-dessus, c'est à vous de choisir, voulez-vous rechercher par « matricule » ? par « nom » ? ou par « prénom » ? Mais jamais sur plus d'une colonne à la fois, et les tests de recherche étaient limités qu'à une égalité pure et simple.

Pour Michael KYUNGU ILUNGA, l'idée était celle de permettre à l'utilisateur final d'effectuer des recherches multicolones, et dont le type de comparaison n'est pas limité qu'à l'égalité.

Lors d'un travail pratique ayant comme but la création d'un programme complet de gestion, travail pratique du cours de VB.NET alors dispensé par l'assistant Deogratias KITENGE KALUME, Michael KYUNGU ILUNGA décida de passer à l'action mais le résultat ne fut pas assez élogieux.

Il décida alors de faire appel à deux de ses collègues pour pouvoir relever le défi ensemble : Benoît KAMONA BUNAKE et Samuel NGANDU MWEPU. Pendant deux jours, Michael KYUNGU ILUNGA et Benoît KAMONA BUNAKE (Samuel NGANDU MWEPU n'ayant pas répondu favorablement) travaillèrent sur un projet alors nommé « Recherche avancée ».

Il s'agissait d'un programme Windows Forms écrit en C# qui permettrait une recherche multicolones et avec plusieurs tests de comparaison sur des données provenant des bases de données relationnelles (avec le système de gestion de bases de données MySQL). Au démarrage, il vous demande de choisir une table parmi celles d'une base de données, base de données ayant été spécifiée en dur dans le code. Vous choisissez une table, il vous affiche une grille contenant toutes les données de la table choisie et une autre grille vous permettant d'effectuer les opérations de recherche. Pour pouvoir effectuer la recherche, le programme générerait une requête SQL permettant de parfaire la recherche. Dans cette requête SQL, une simple clause « WHERE » était spécifié pour les colonnes dans lesquels l'utilisateur a saisi une

valeur mais aussi l'opérateur de comparaison choisi. Cette forte dépendance envers SQL était la plus grande difficulté.

C'est ainsi, Michael KYUNGU ILUNGA, en se basant sur ce programme, travailla sur un projet de bibliothèque .NET qu'on pourrait utiliser sans recourir obligatoirement à SQL, projet qu'il nomma alors « El Search ».

Les premières versions ne sont jamais sorties en version alpha, c'était juste des versions beta limitées à des utilisations personnelles de Michael KYUNGU ILUNGA, et d'un de ses proches camarades de promotion, Hans KASHALA KAHILU. Néanmoins, la version 2.1 est la première version alpha de la bibliothèque El Kibambo.

La bibliothèque a été annoncée officiellement le 5 janvier 2021 sous le nom « El Search », la bibliothèque a subi plusieurs phases de test et a été renommée « El Kibambo » mi 2021.

Le mot « Kibambo » vient de *Gloria NGOMBE KIBAMBO*, une collègue de premier plan à Michael KYUNGU ILUNGA. Pourquoi ce choix de nom ? Eh bien, pour mille et une raison [...] =)