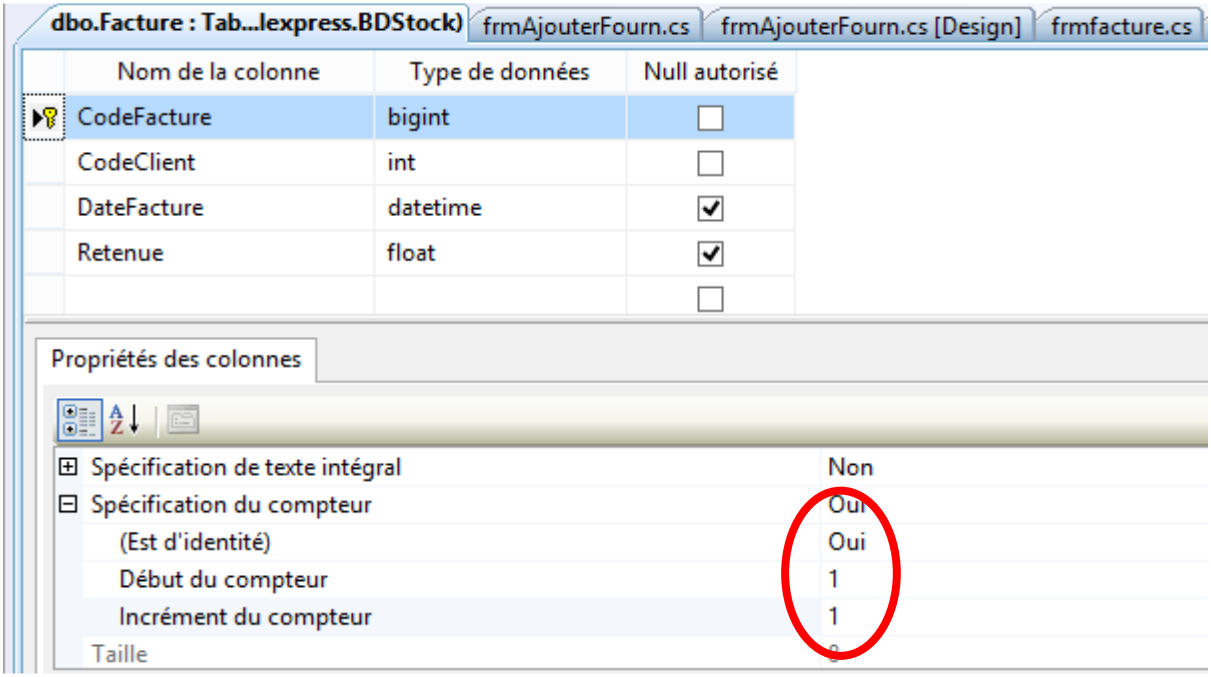
TP 7 : BD ET CRYSTAL REPORTS

OBJECTIFS :

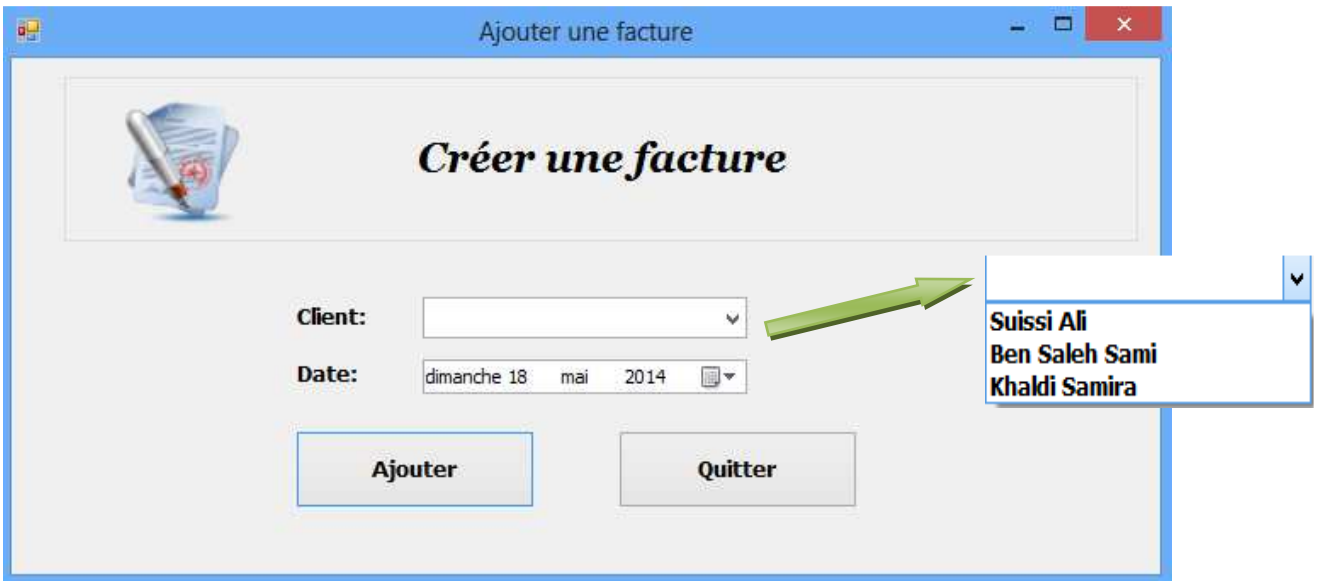
Apprendre d'autres manipulation sur la bases de données et l'utilisation de Crystal reports.

I- Gestion des factures:

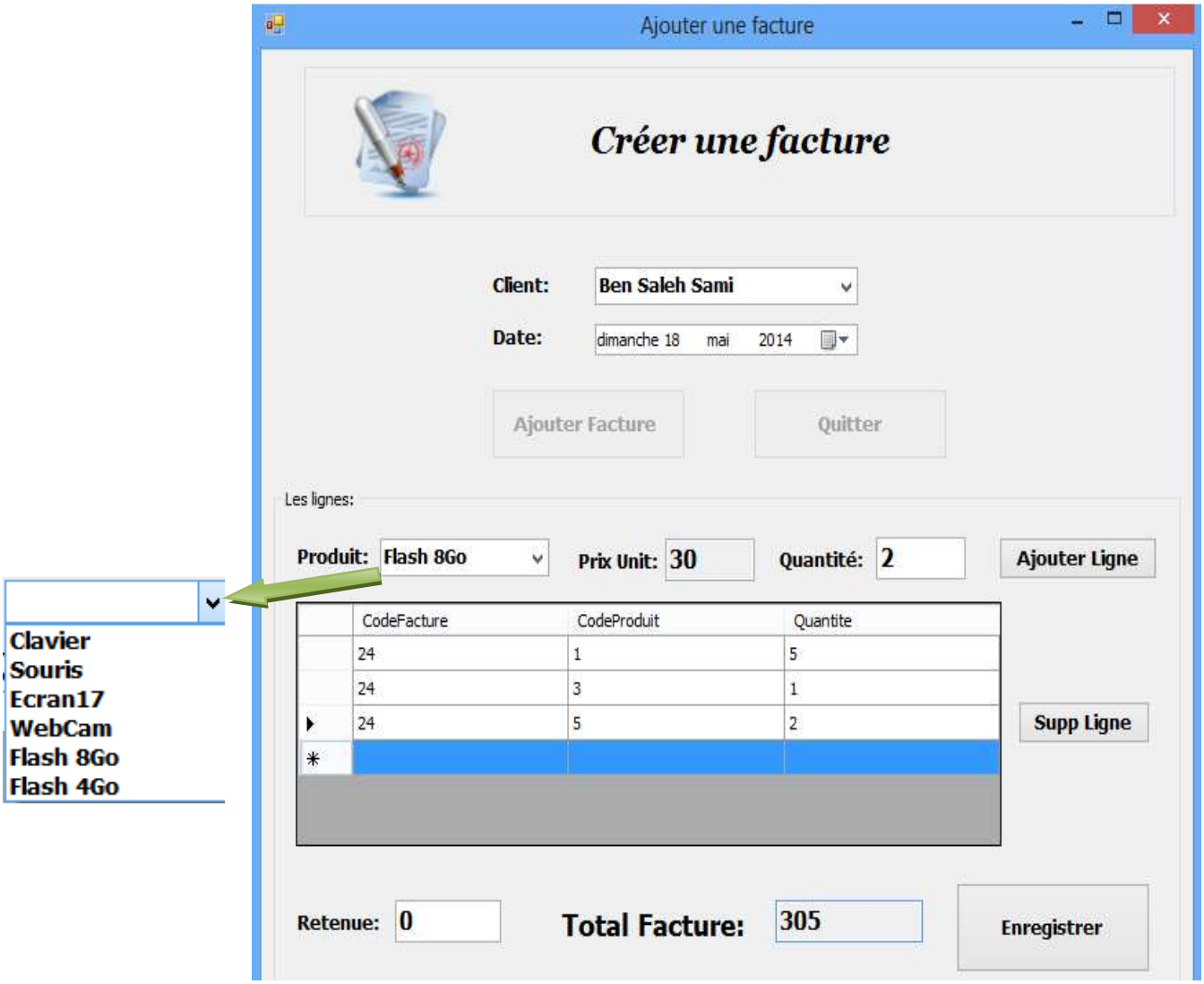
1. Dans la table Facture, changer le type de la clé primaire (codeFacture) de la table en une clé automatique:



2. Préparer l'interface suivante permettant de créer une nouvelle facture, la liste des clients doit être chargé à partir de la base comme indiqué ci-dessous:



3. Le bouton ***Ajouter Facture*** permet d'insérer la facture dans la table **Facture**, et d'afficher la partie du formulaire réserver à la création des lignes de la facture:



Changer les propriétés suivantes:

- Le groupeBox des lignes (grpLigne) , visible= false;
- Le textBox Prix Unit (txtPrix) et Total Facture (txtPrix), ReadOnly=true;
- Le dataGridView, AutoSizeColoumsMode = Fill;
- L'interface, size.Height=320;

Indication sur le code:

- *Déclaration globale:*

```
SqlConnection cnx;
SqlCommand cmd, cmd2, cmd3;
SqlDataAdapter adap, adap2,adap3;
SqlDataReader dread;
DataSet dset, dset2, dset3;
SqlCommandBuilder cb;
DataRow dtr;
int qt;
long codeFacture;
float t = 0;
```

- *Chargement de l'interface::*

```
cnx = new SqlConnection();
cnx.ConnectionString="Data Source=.\\sqlexpress;Initial
Catalog=BDStock;Integrated Security=True";
cnx.Open();
//commande pour charger la liste des clients
cmd=new SqlCommand();
cmd.CommandText="select CodeClient, cast(Nom as varchar(20)) + ' '
+  cast(prenom as varchar(20)) as NomPrenom  from Client";
cmd.Connection=cnx;
adap = new SqlDataAdapter(cmd);
dset = new DataSet();
adap.Fill(dset, "Client");
cmbClient.DataSource = dset.Tables[0];
cmbClient.DisplayMember = "NomPrenom";
cmbClient.ValueMember = "CodeClient";
//commande pour charger la liste des Produits
cmd3 = new SqlCommand();
cmd3.CommandText = "select CodeProduit, NomProduit ,QtEnStock from
Produit where QtEnStock >0;";
cmd3.Connection = cnx;
adap3 = new SqlDataAdapter(cmd3);
dset3 = new DataSet();
adap3.Fill(dset3, "Produit");
cmbProd.DataSource = dset3.Tables[0];
cmbProd.DisplayMember = "NomProduit";
cmbProd.ValueMember = "CodeProduit";
cmbProd.SelectedIndex = -1;
cmbClient.SelectedIndex = -1;
```

- *Le bouton Ajouter ligne:*

```
if (MessageBox.Show("Voulez vous vraiment ajouter cette facture
?", "Ajout d'une Facture", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
{
   try
   {
      //insertion de la facture dans la table facture
      cmd.CommandText = "insert into Facture(CodeClient,DateFacture)
       values (" + cmbClient.SelectedValue.ToString() + ",'" +
      dateTimePicker1.Value.Date.ToString() + "')";
       cmd.ExecuteNonQuery();

      //Afficher la deuxième partie de facture
      this.Height= 640;
      grbligne.Visible = true;

      //recupérer le code de la facture
      cmd.CommandText = "select max(codeFacture) from Facture;";
       codeFacture = (long)cmd.ExecuteScalar();

      //créer le dataset pour les lignes de la facture
      cmd2 = new SqlCommand();
      cmd2.CommandText = "select * from LigneFacture where
codeFacture=" + codeFacture.ToString();
      cmd2.Connection = cnx;
```

```
            adap2 = new SqlDataAdapter(cmd2);
            dset2 = new DataSet();
            //recupérer les clés de la table
            adap2.MissingSchemaAction = MissingSchemaAction.AddWithKey;
            adap2.Fill(dset2, "LigneFacture");
            dataGridView1.DataSource = dset2.Tables[0];
            btnAjoutFact.Enabled = false;
            btnQ1.Enabled = false;
        }
        catch (Exception ex)
        {
              MessageBox.Show(ex.Message);
        }
    }
```

- *L'évènement SelectionChangeCommitted de la liste des produits* ***cmbProd****:*

```
cmd.CommandText = "select CodeProduit, PrixUnitaire,QtEnStock  from
Produit where CodeProduit=" + cmbProd.SelectedValue.ToString();
cmd.Connection = cnx;
dread = cmd.ExecuteReader();
dread.Read();
txtprix.Text = dread[1].ToString();
txtQt.Text = dread[2].ToString();
qt = (int)dread[2];
dread.Close();
```

- *L'évènement Validating du textBox* ***txtQt****:*

```
if (int.Parse(txtQt.Text) > qt)
{
   MessageBox.Show("qantité en stock insuffisante pour la quantité
demandé");
   txtQt.Text = qt.ToString();
   txtQt.Focus();
}
```

- *Le bouton Ajouter ligne:*

```
try
{
    if (cmbProd.SelectedIndex != -1 && int.Parse(txtQt.Text) > 0)
    {
        dtr = dset2.Tables[0].NewRow();
        dtr[0] = codeFacture;
        dtr[1] = int.Parse(cmbProd.SelectedValue.ToString());
        dtr[2] = int.Parse(txtQt.Text);
        dset2.Tables[0].Rows.Add(dtr);
        //mettre à jours la quantité en stock du produit ajouté
        dset3.Tables[0].Rows[cmbProd.SelectedIndex][2] =
(int.Parse(dset3.Tables[0].Rows[cmbProd.SelectedIndex][2].ToString
())-int.Parse(txtQt.Text)).ToString();
        t += float.Parse(txtQt.Text) * float.Parse(txtprix.Text);
        txtTot.Text = t.ToString();
    }
}
catch (Exception ex)
{
     MessageBox.Show(ex.Message);
}
```

- *Le bouton Supp Ligne:*

```
try
{
    int i = dataGridView1.CurrentRow.Index;
    long code =
long.Parse(dataGridView1.CurrentRow.Cells[1].Value.ToString());
    int qt =
int.Parse(dataGridView1.CurrentRow.Cells[2].Value.ToString());
    float p;
    cmd = cnx.CreateCommand();
    //création d'une requete avec parametres
    cmd.CommandText = "select PrixUnitaire from Produit where
    CodeProduit=@code";
    //definition d'un nouveau parametre
    cmd.Parameters.Add("@code", SqlDbType.BigInt);
    //préparer la requete pour le serveur
    cmd.Prepare();
    cmd.Parameters["@code"].Value = code;
    p = float.Parse(cmd.ExecuteScalar().ToString());
    t -= p * qt;
    txtTot.Text = (t - float.Parse(txtRet.Text)).ToString();
    dset2.Tables[0].Rows[i].Delete();
    dset2.AcceptChanges();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

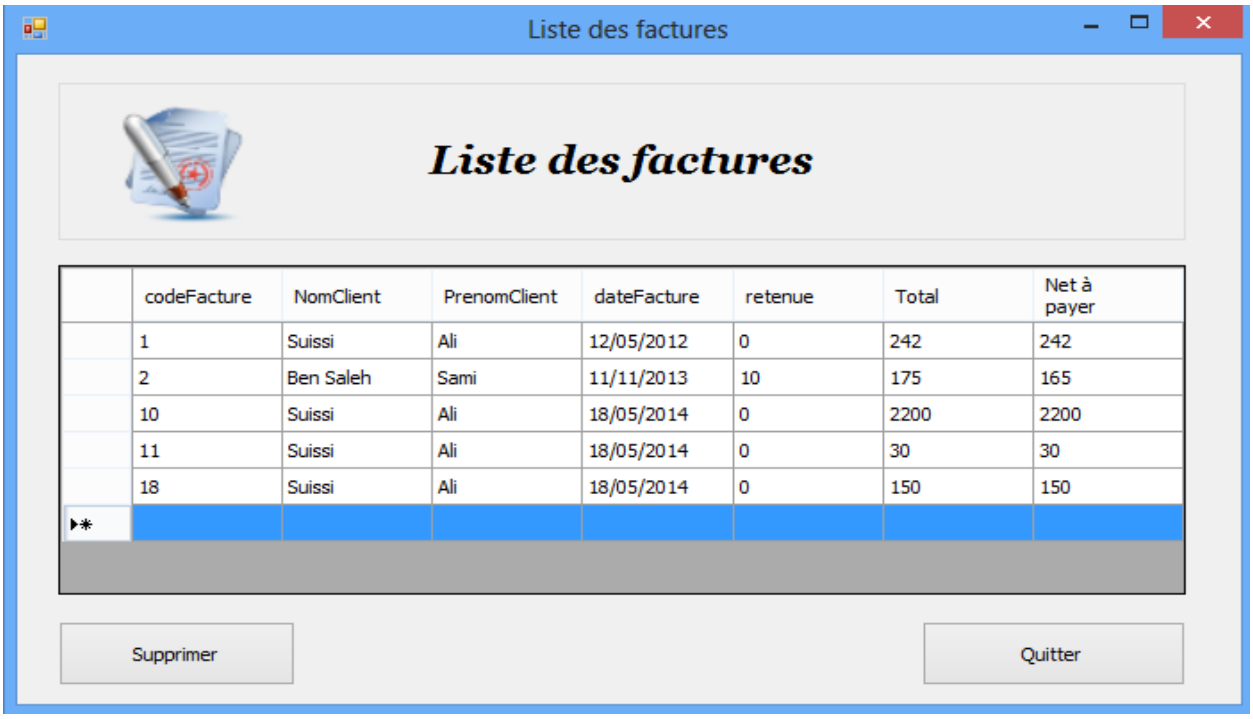
- *L'évènement Validating du textBox txtRet:*

```
//mettre à jour le total de la facture
txtTot.Text = (float.Parse(txtRet.Text) + t).ToString();
```

- *Le bouton Enregistrer:*

```
try
{
if (MessageBox.Show("Voulez vous vraiment enregistrer la facture
?", "Ajout d'une Facture", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
{
    //Enregistrer les de la factures dans la base de données
    cb = new SqlCommandBuilder(adap2);
    adap2.Update(dset2, "LigneFacture");
    cb = new SqlCommandBuilder(adap3);
    //mettre à jours la liste des produits (quantité en stock)
    adap3.Update(dset3, "Produit");
    //ajouter le retenue dans la table facture
    cmd.CommandText = "update Facture set retenue=" + txtRet.Text +
" where codeFacture=" + codeFacture.ToString() ;
    cmd.ExecuteNonQuery();
    MessageBox.Show("Facture enregistrée");
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

4. Créer l'interface suivante permettant de gérer la liste des factures:

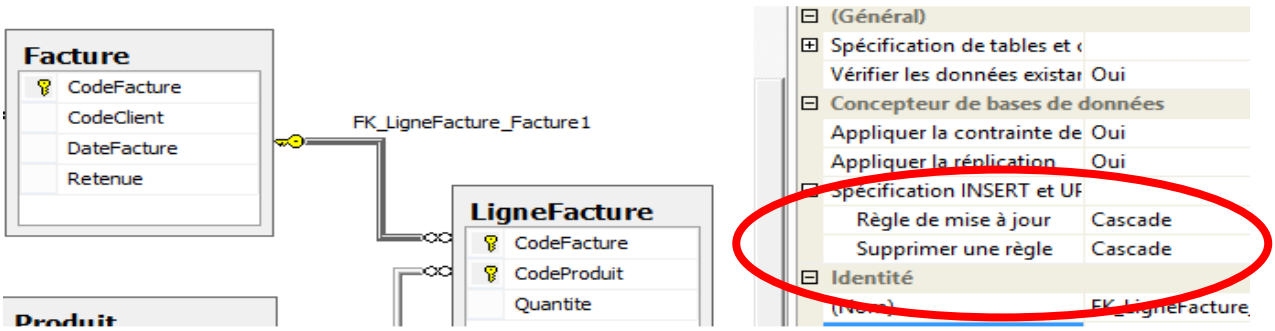


Requête SQL de charger de la liste des factures:

```
cmd = new SqlCommand();
cmd.CommandText = "select f.codeFacture,  CAST(c.Nom AS nvarchar) as
NomClient,   CAST(c.Prenom AS nvarchar) as PrenomClient,dateFacture,
retenue,SUM(p.PrixUnitaire*lf.Quantite) as Total
,SUM(p.PrixUnitaire*lf.Quantite)-retenue as 'Net à payer'
from Facture f, Client c, ligneFacture lf, produit p
where f.codeClient=c.codeClient
and lf.codeFacture=f.codeFacture
and lf.codeProduit=p.codeProduit
group by f.codeFacture,  CAST(c.Nom AS nvarchar),CAST(c.Prenom AS
nvarchar),dateFacture, retenue;";
```

Suppression d'une facture:

Il faut changer les spécifications des la relation entre la table LigneFacture et celle Facture:

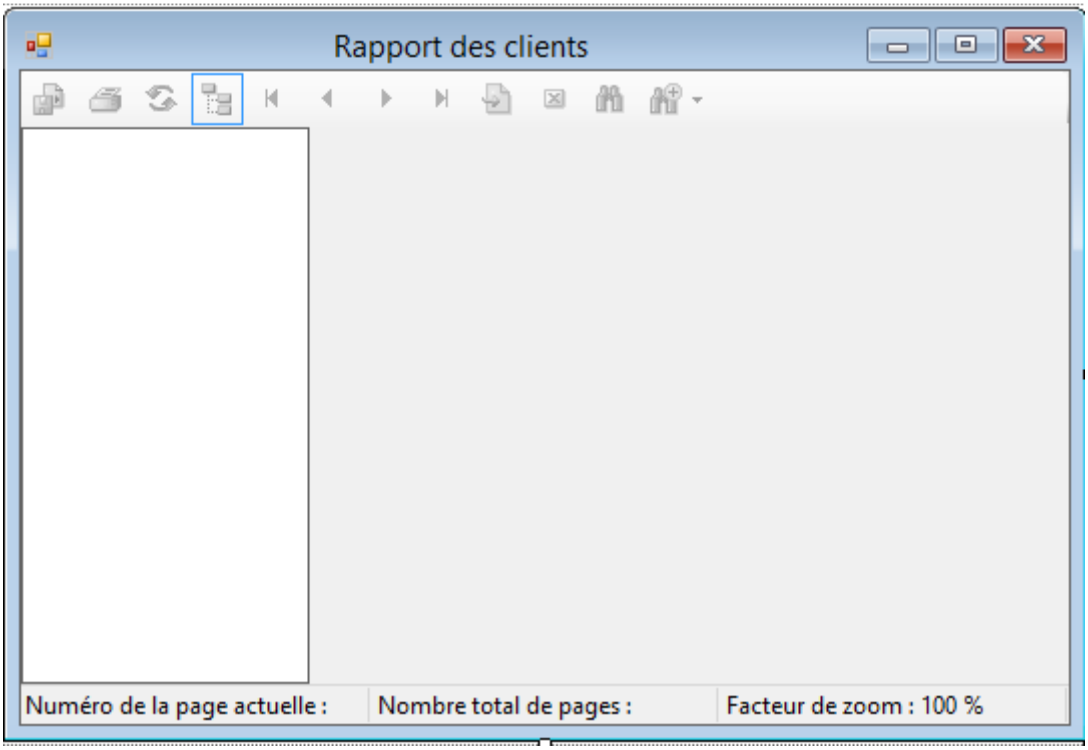



```
//Code de suppression:
int i = dataGridView1.CurrentRow.Index;
dset.Tables[0].Rows[i].Delete();
cb = new SqlCommandBuilder(adap);
adap.Update(dset, "Facture");
```

II- Utilisation de Crystal Reports:

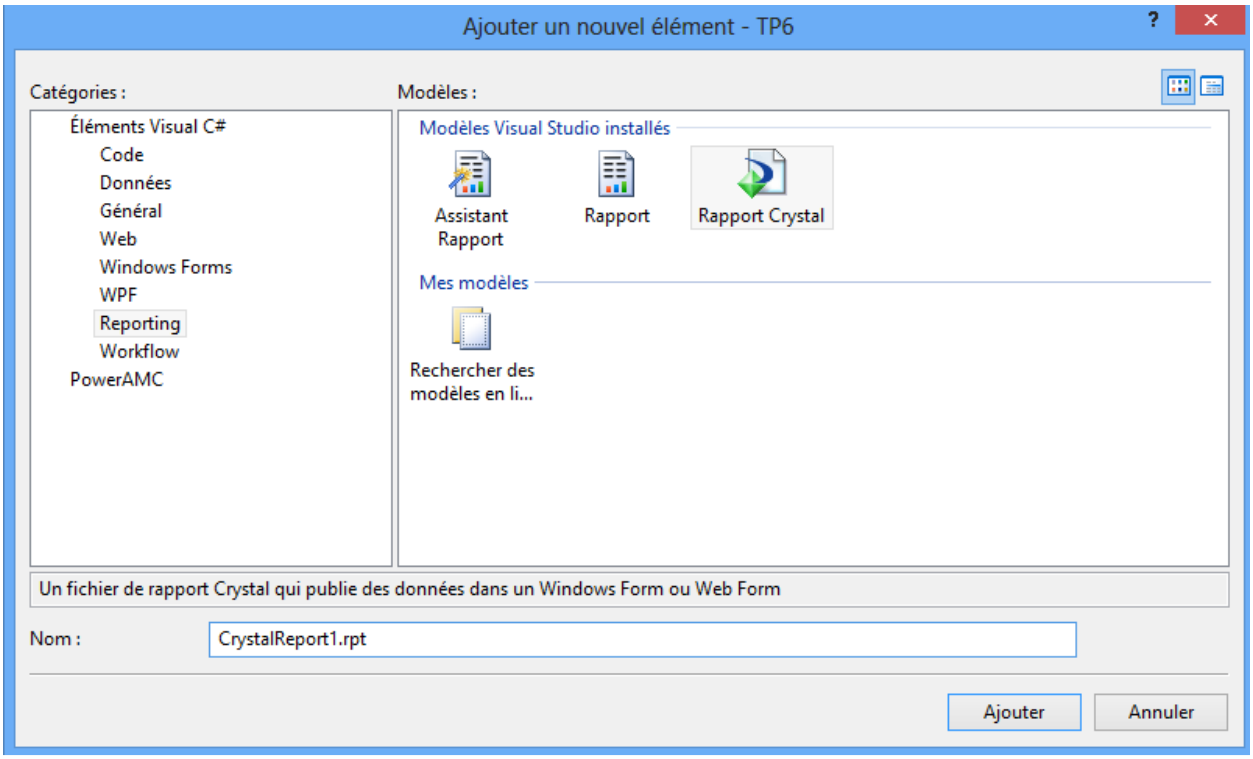
L'objectif est d'afficher la liste des clients dans un rapport:

1. Créer une nouvelle interface ***frmRapClient***, et ajouter à partir de la boite à outils dans l'interface un composant ***CrystalReportViewer***:

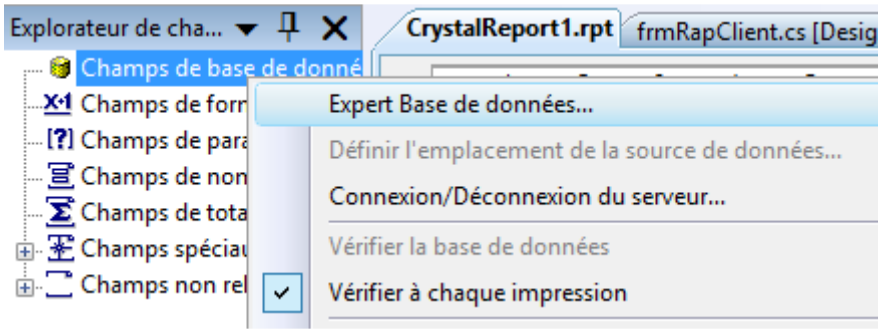


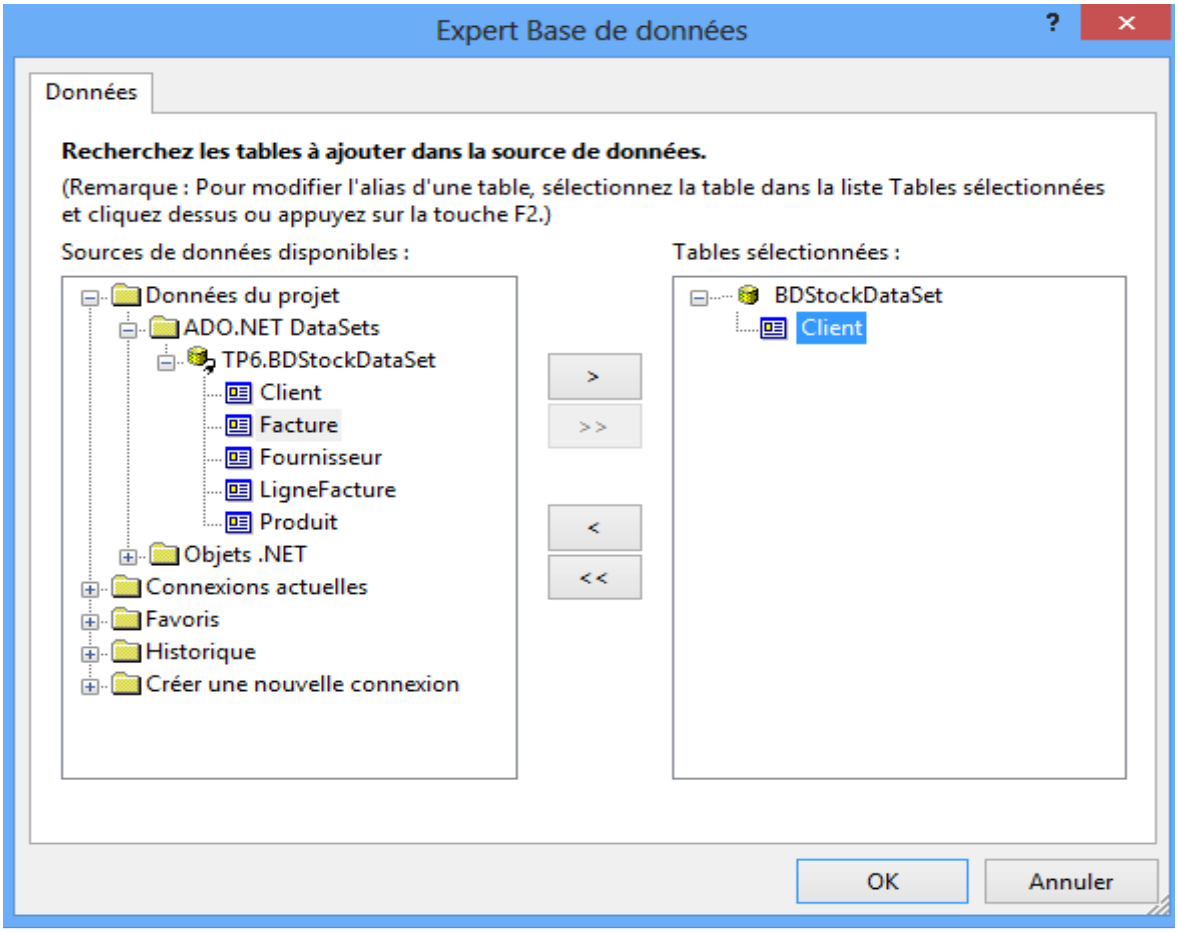
Changer la propriété ***WindowState=Maximized***

2. Ajouter un nouveau rapport vide :

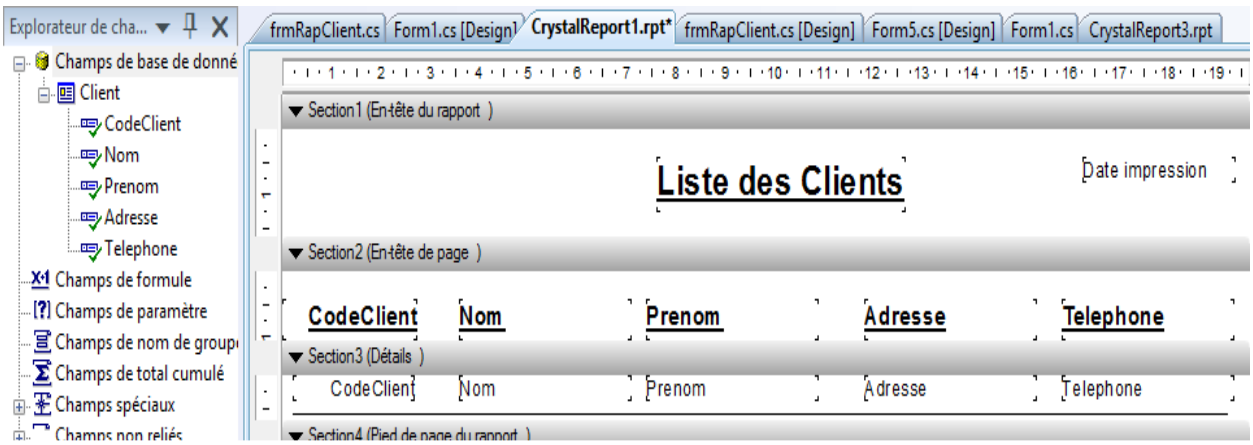


3. Définir les champs de la base de données pour le rapport:





4. Avec la souris, glisser les champs de la base vers la section 3 et préparer le rapports suivant:

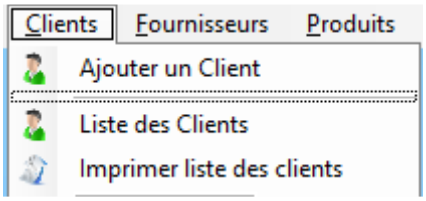


5. Dans le formulaire ***frmRapClient***, changer le constructeur de l'interface afin qu'il accepte un rapport pour l'initialiser:

```
//bibliothèque à ajouter au debut du code
using CrystalDecisions.CrystalReports.Engine;
//constructeur de la classe
public frmRapClient(ReportDocument rpt)
{
        InitializeComponent();
        crystalReportViewer1.ReportSource = rpt;

}
```

6. Dans le menu principal, ajouter un élément pour afficher le formulaire.




```
//Ajouter la bibliothèque:
using System.Data.SqlClient;
using CrystalDecisions.CrystalReports.Engine;
//le menu Imprimer liste des clients
SqlConnection cnx;
SqlCommand cmd;
SqlDataAdapter adap;
DataSet dset;
ReportDocument rpt;
rpt = new CrystalReport1();
cnx = new SqlConnection();
cnx.ConnectionString = "Data Source=.\\sqlexpress;Initial
Catalog=BDStock;Integrated Security=True";
cnx.Open();
cmd = new SqlCommand();
cmd.CommandText = "select * from Client";
cmd.Connection = cnx;
adap = new SqlDataAdapter(cmd);
dset = new DataSet();
adap.Fill(dset, "client");
rpt.SetDataSource(dset.Tables[0]);
frmRapClient frm = new frmRapClient(rpt);
frm.ShowDialog();
```