AKOUVI Brice

AGERT Petru-Luigi

Rapport Projet : Base De Données et interopérabilité

I. Structure de la Solution

a. Classes

Création des 9 classes correspondantes aux entités du schéma E/A.

Une classe **Contrôle** représentant les différents utilisateurs ainsi que leurs mots de passe respectifs, cette classe intègre la connexion Sql :

```
public static MySqlConnection Connect()
{
    try
    {
        string bdd = "SERVER=localhost;PORT=3306;DATABASE=Velomax;";
        bdd += $"UID={UID};PASSWORD={PSWD}";
        MySqlConnection connexion = new MySqlConnection(bdd);
        connexion.Open();
        return connexion;
    }
    catch (MySqlException e) { }

    return null;
}
```

Une interface **IElement** permettant l'intégration des méthodes **Ajout()**, **Suppression()**, **Modification()** aux différentes classes.

b. Fenêtres

Création de deux fenêtres, la première représentant le choix de l'utilisateur et l'autre représentant l'accueil permettant d'accéder à tous les modules de l'interface.

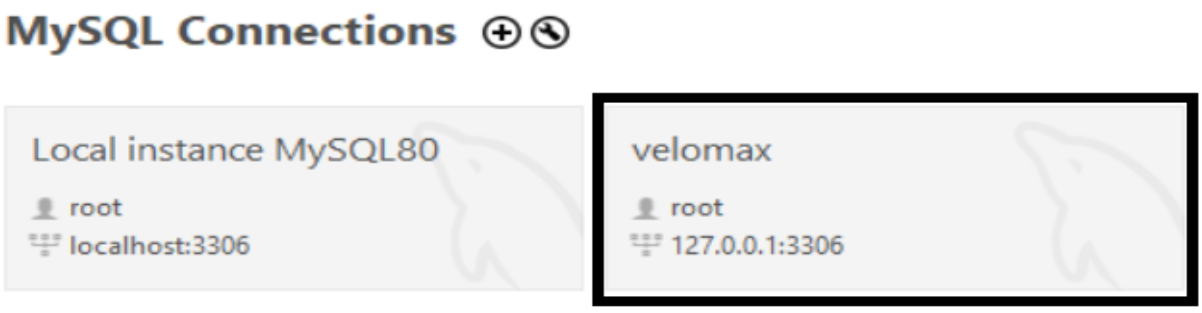
c. Pages

Création d'une page par module demandé, une page 'pop-up' vient s'ajouter au module statistique pour afficher celles des **Commandes**.

II. MySql :

a. Nouvelle connexion

Création d'une nouvelle connexion MySql Root/Root, qui matérialise le nouvel utilisateur de l'interface wpf. Cette connexion MySql/C# rends accessible les bases de données dans VS avec une relation client/serveur.



b. Requêtes

Les requêtes SQL ont été utilisées avec leurs implémentations dans les commandes WPF.

```
1 référence
public void fill_grid1()
{
    string req = "select count(*) nbclient from (select no_i from individu union select no_b from boutique) as t;";
    MySqlDataReader reader = Controle.Requete(req, true);
    if (reader.Read())
    {
        lbl_nbclient.Content = reader.GetInt32("nbclient").ToString();
    }
}
```

III. WPF :

a. Commandes WPF

Les Commandes d'actions sont construites à partir du langage *Sql* et utilise la connexion à la base de donnée Velomax.

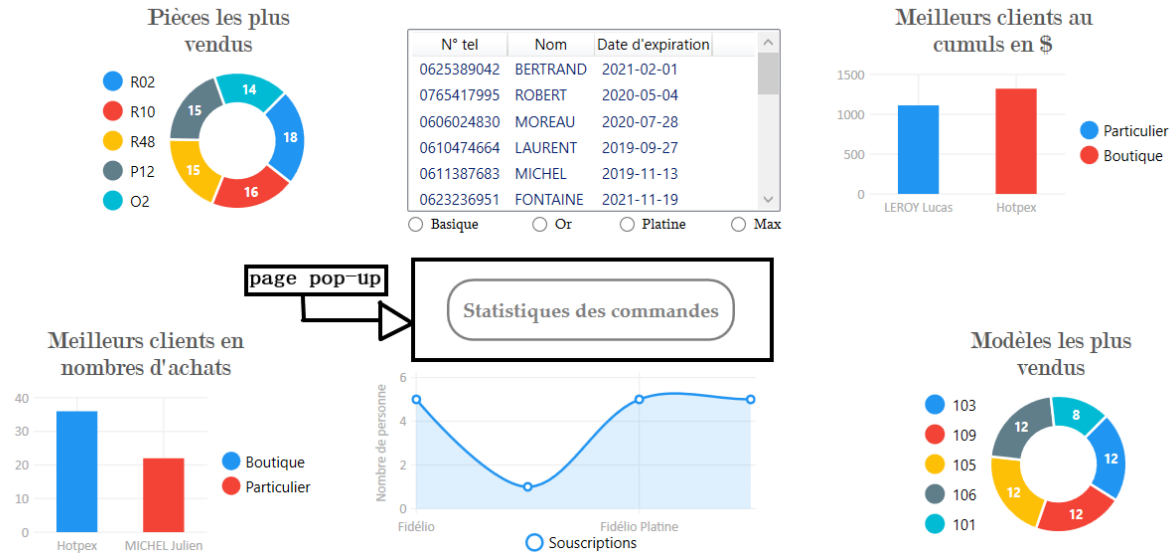
```
2 références
public void update_listclient()
{
    string req = "select * from individu";
    req += " order by convert(SUBSTRING_INDEX(no_i,'I',-1), signed integer) desc;";
    MySqlDataReader reader = Controle.Requete(req, true);
    if(reader != null)
    {
        List<Individu> lsti = new List<Individu>();
        while (reader.Read())
        {
            Individu i = new Individu((string)reader["no_i"], reader["nom_i"].ToString(), reader["adresse_i"].ToString(),
                reader["couriel_i"].ToString(), (int)reader["tel_i"], reader["prenom_i"].ToString());
            lsti.Add(i);
        }
        listview_indiv.ItemsSource = lsti;
    }

}
```

(La méthode ci-dessus permet la mise à jour de la liste des clients (par ordre alphabétique).)

IV. Choix

a. Modules Statistiques



Aperçu de différentes statistiques de l'entreprise à l'aide de graphique fournis par le dictionnaire de ressources WPF.

b. Stock

```
1 référence
private void add_velo(object sender, RoutedEventArgs e)
{
    Commande current = (Commande)listview_com.SelectedItem;
    if (box_velo.SelectedItem != null && box_taille.SelectedItem != null && box_quantite_velo.Text != "" && current != null)
    {
        string numvelo = box_velo.SelectedItem.ToString();
        string taille = ((ListBoxItem)box_taille.SelectedItem).Content.ToString();
        int.TryParse(box_quantite_velo.Text, out int quantite);
        current.AjoutElement(numvelo, quantite, taille, false);
        lstview_comp.ItemsSource = get_composition_commande(current);

    }
}
```

Lorsque l'on ajoute une pièce, un vélo à une commande, le stock diminue automatiquement.

```
public void fill_modele()
{
    MySqlDataReader reader = Controle.Requete(reqs[1], true);
    List<Stock_Modele> lstm = new List<Stock_Modele>();
    if (reader != null)
    {
        while (reader.Read())
        {
            Stock_Modele sm = new Stock_Modele
            {
                No = (string)reader["no_m"],
                Nom = (string)reader["nom_m"],
                Categorie = (string)reader["categorie"],
                Grandeur = (string)reader["label"],
                Stock = reader.GetInt32("stock")
            };
            lstm.Add(sm);

        }
        listview_modele.ItemsSource = lstm;
    }
}
```

c. Export Xml/Json

```
7 références
public void serializetoJSON<T>(string file, List<T> lst)
{
    StreamWriter sr = new StreamWriter("JSON_Export/" + file);
    JsonTextWriter jsonWriter = new JsonTextWriter(sr);

    JsonSerializer serializer = new JsonSerializer();
    serializer.Serialize(jsonWriter, lst);

    jsonWriter.Close();
    sr.Close();
    MessageBox.Show("Table exportée avec succès ! A trouver dans le bin/debug");

}
```

```
6 références
public void serializetoXML<T>(string file, List<T> lst)
{
    XmlSerializer xs = new XmlSerializer(typeof(List<T>));
    StreamWriter sr = new StreamWriter("XML_Export/" + file);
    xs.Serialize(sr, lst);
    sr.Close();
}
```

L'exportation peut se faire dans le sens xml/json mais aussi json/xml, ainsi que sur toutes les

tables.

```
1 référence
private void export_indiv(object sender, MouseButtonEventArgs e)
{
    string req = "select * from individu";
    req += " order by convert((SUBSTRING_INDEX(no_i,'I',-1), signed integer) desc;";
    MySqlDataReader reader = Controle.Requete(req, true);
    List<Individu> lsti = new List<Individu>();

    if (reader != null)
    {
        while (reader.Read())
        {
            Individu i = new Individu((string)reader["no_i"], reader["nom_i"].ToString(), reader["adresse_i"].ToSt
                reader["couriel_i"].ToString(), (int)reader["tel_i"], reader["prenom_i"].ToString());
            lsti.Add(i);
        }

    }

    serializetoJSON<Individu>("individu.json", lsti);
    serializetoXML<Individu>("individu.xml", lsti);
}
```