Dossier de programmation 3 :
SONTOT Alexis

Sommaire

A) Compétences :

A4.1.6 Gestion d'environnements de développement et de test
A4.1.7 Développement, utilisation ou adaptation de composants logiciels
A4.1.8 Réalisation des tests nécessaires à la validation d'éléments adaptés ou développés
A4.1.9 Rédaction d'une documentation technique

B) Structure des données :

Classe Globale :

```
class Global
{
    public static int nbMaxNiveaux = 42 ;
    public static int plafondScore = 666;
    public static ClassJoueur leJoueurTest;
    public static Communaute laCommunauté = new Communaute();


}
```

Classe Communauté :

```
private List<ClassJoueur> lesJoueurs;
private List<Message> lesMessages;
```

Classe message :

```
private string dateMessage;
private string JoueurEnvoyeur;
private string contenuMessage;
```

C) Explication des algorithmes :

1. Classe Communauté et classe Message :

La classe communauté est simple :
Elle contient le constructeur qui l'initialise avec une liste de joueur vide et une liste de message vide elle contient deux getters un pour obtenir la liste des joueurs et un pour la liste des messages et également 2 setters pour ajouter joueurs et messages a la communauté :

```
public Communaute()
{
    this.lesJoueurs = new List<ClassJoueur>();
    this.lesMessages = new List<Message>();
}

//getter
public List<Message> getMessage()
{
    return this.lesMessages;
}

public List<ClassJoueur> getJoueurs()
{
    return this.lesJoueurs;
}
```

```
//setter
public void ajouterJoueur(ClassJoueur leJoueur)
{
    this.lesJoueurs.Add(leJoueur);
}

public void ajouterMessage(Message leMessage)
{
    this.lesMessages.Add(leMessage);
}
```

La classe message contient elle le constructeur et 3 getters pour le joueur envoyeur le contenu du message ainsi que la date d'envoi du message :

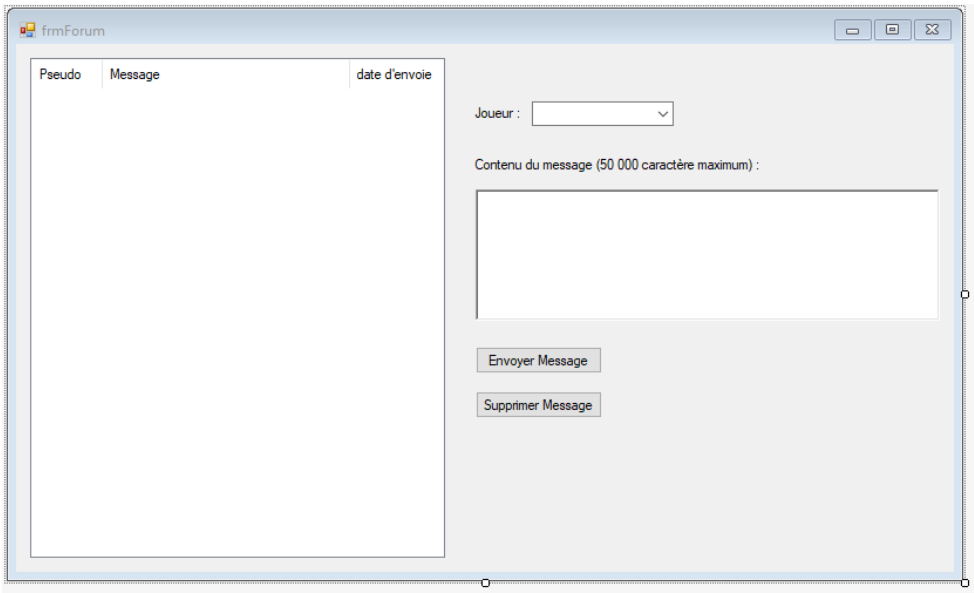
```
public Message(string leJoueur , string leContenuMessage)
{
    this.JoueurEnvoyeur = leJoueur;
    this.contenuMessage = leContenuMessage;
    this.dateMessage = DateTime.Now.ToShortDateString();
}

//getter
public string getJoueur() { return this.JoueurEnvoyeur; }
public string getContenuMessage() { return this.contenuMessage; }
public string getDate() { return this.dateMessage; }
```

2. La modification de la classe ClassJoueur :

La modification est simple un getter et un setter ainsi qu'un attribut privé pour gérer si le joueur est influenceur :

```
private Boolean Influenceur;

public ClassJoueur(string lenom,string leprenom,string lepseudo, string lemail, PictureBox lavatar )
{
    this.nom = lenom;
    this.prenom = leprenom;
    this.pseudo = lepseudo;
    this.email = lemail;
    this.avatar = lavatar;
    this.niveau = 1;
    this.score = 0;
    this.Influenceur = false;


}
public bool estInfluenceur()
{
    return this.Influenceur;
}

public void setInfluenceur(Boolean vraifaux)
{
    this.Influenceur = vraifaux;
}
```

3. Form Forum et classement :



Au load du formulaire on alimente le combobox avec les pseudos de tous les joueurs de la communauté puis on alimente la listview des messages contenu dans la communauté

```
private void frmForum_Load(object sender, EventArgs e)
{
    btEnvoyer.Enabled = false;

   foreach (ClassJoueur leJoueur in Global.laCommunauté.getJoueurs())
    {
        cbJoueur.Items.Add(leJoueur.getPseudo());
    }

    foreach (Message leMessage in Global.laCommunauté.getMessage())
    {

        ListViewItem ligne = new ListViewItem();
        ligne.Text = leMessage.getJoueur();
        ligne.SubItems.Add(leMessage.getContenuMessage());
        ligne.SubItems.Add(leMessage.getDate());

        lvMessage.Items.Add(ligne);
    }
}
```

Le bouton permettant d'envoyé le message créer un nouveau message qu'il enregistre directement dans la classe communauté de global puis pour chaque message il affiche dans la listview tous les messages présents dans la communauté sachant que je clear la listview au début pour éviter de réafficher le même message 2 fois

```
private void btEnvoyer_Click(object sender, EventArgs e)
{
    Global.laCommunauté.ajouterMessage(new Message(cbJoueur.Text, rtbMessage.Text));
    lvMessage.Items.Clear();
    foreach (Message leMessage in Global.laCommunauté.getMessage())
    {

        ListViewItem ligne = new ListViewItem();
        ligne.Text = leMessage.getJoueur();
        ligne.SubItems.Add(leMessage.getContenuMessage());
        ligne.SubItems.Add(leMessage.getDate());

        lvMessage.Items.Add(ligne);
    }

}
```

Puis pour la suppression des messages j'ai fais en sorte que quand on clic sur un message le bouton de suppression est Enabled

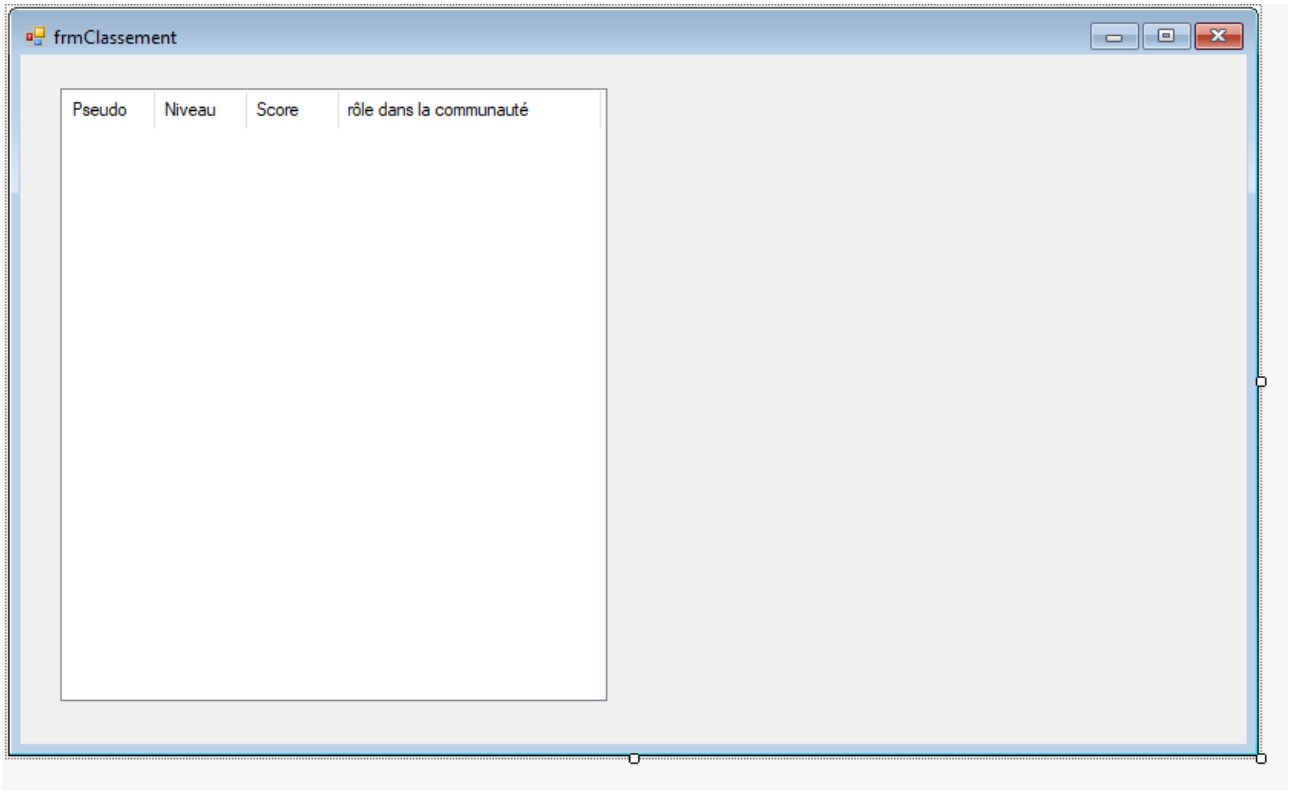
```
private void lvMessage_SelectedIndexChanged(object sender, EventArgs e)
{
    btSupp.Enabled = true;
}
```

Et donc pour la suppression je supprime le message dans la communauté qui a le même index que celui de la listview, je clear la listview et la réalimente :

```
private void btSupp_Click(object sender, EventArgs e)
{
    int idMessage = lvMessage.TabIndex;
    Global.laCommunauté.getMessage().RemoveAt(idMessage);

    lvMessage.Items.Clear();
    foreach (Message leMessage in Global.laCommunauté.getMessage())
    {

        ListViewItem ligne = new ListViewItem();
        ligne.Text = leMessage.getJoueur();
        ligne.SubItems.Add(leMessage.getContenuMessage());
        ligne.SubItems.Add(leMessage.getDate());

        lvMessage.Items.Add(ligne);
    }
}
```

Puis pour le form Classement :

Il s'agit d'une simple listview affichant pseudo niveau et score des joueurs de la communauté ainsi que s'ils sont influenceur (le premier joueur du classement est désigné comme influenceur)




```
IEnumerable<ClassJoueur> triJoueur = from leJoueur in Global.laCommunauté.getJoueurs() orderby leJoueur.getNiveau() descending, leJoueur.getScore() descending select leJoueur;
```

Cette ligne permet de trier les joueurs de la communauté par ordre décroissant du niveau et du score

```
triJoueur.First().setInfluenceur(true);
```

Cette ligne rend le premier joueur du classement comme influenceur

```
foreach(ClassJoueur Joueur in triJoueur)
{
    if(Joueur != triJoueur.First())
    {
        Joueur.setInfluenceur(false);
    }
    ListViewItem ligne = new ListViewItem();
    ligne.Text = Joueur.getPseudo();
    ligne.SubItems.Add(Joueur.getNiveau().ToString());
    ligne.SubItems.Add(Joueur.getScore().ToString());
    if(Joueur.estInfluenceur() == true)
    {
        ligne.SubItems.Add("influenceur de la communauté");
    }
    else
    {
        ligne.SubItems.Add("joueur de la communauté");
    }
    lvClassement.Items.Add(ligne);
}
```

Puis on affiche le tous dans la listview.
L'encadre rouge et la en cas de changement d'influenceur elle force tous les joueurs qui ne sont pas premier a ne pas/plus être influenceur.

4. Modification des forms

J’ai simplement rajouté une combobox à chaque form pour choisir le joueur à modifier
En ajoutant ce code à chaque fois :

```
private void cbJoueur_SelectedIndexChanged(object sender, EventArgs e)
{
    Global.leJoueurTest = Global.laCommunauté.getJoueurs().ElementAt(cbJoueur.TabIndex);
    btChangerAvatar.Enabled = true;
}
```

E) Jeu d'essai :

1	Ouverture des différentes form et menus via le menu MDI	Valider
2	Form Forum entièrement fonctionnelle	Valider
3	Tous les anciens form fonctionnelle	Valider
4	Système de « connexion » a chaque anciens form pour choisir le joueur	Valider

F) Conclusion :

Pour conclure le dp3 a été entièrement fais cependant étant peu guider certaine partie ne sont peut être pas comme attendu j’ai essayer de faire au plus comme je pensais que ca devait être j’ai un peu galérer pour le classement des joueurs mais je pense m’en être bien sorti sinon dans l’ensemble j’ai appris pas mal de truc et ce fut un dossier de programmation intéressant.