Prise en main de Visual Studio et découverte de C#

**Disclaimer : Il est important de comprendre le code présent dans le TP que vous allez copier-coller.**

**Il est aussi important de comprendre comment les actions sont imbriquées pour arriver au résultat final. Sans cette compréhension, vous ne pourrais pas progresser.**

## Partie 1 : Prise en main de différents éléments

### Phase 1 : Création du projet

Attention, vous devez lancer Visual Studio 2022 et non Visual Studio Code. Visual Studio doit déjà être installé sur les postes.

Une image contenant texte, Police, capture d’écran, logo

Description générée automatiquement

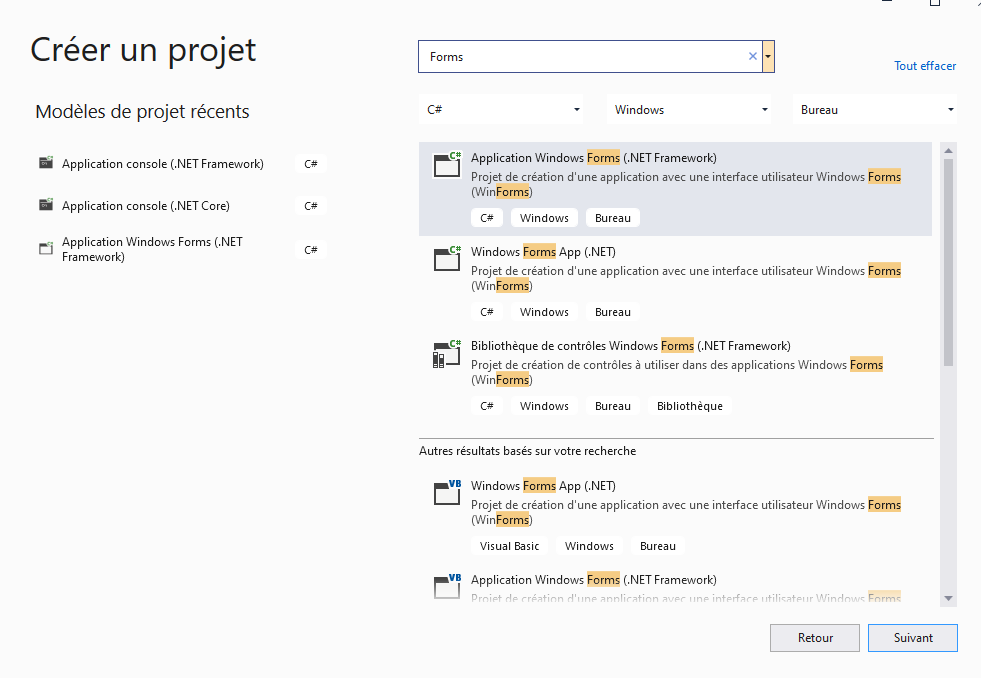
Vous devez alors avoir l’écran suivant : Une image contenant texte, capture d’écran, Police

Description générée automatiquement

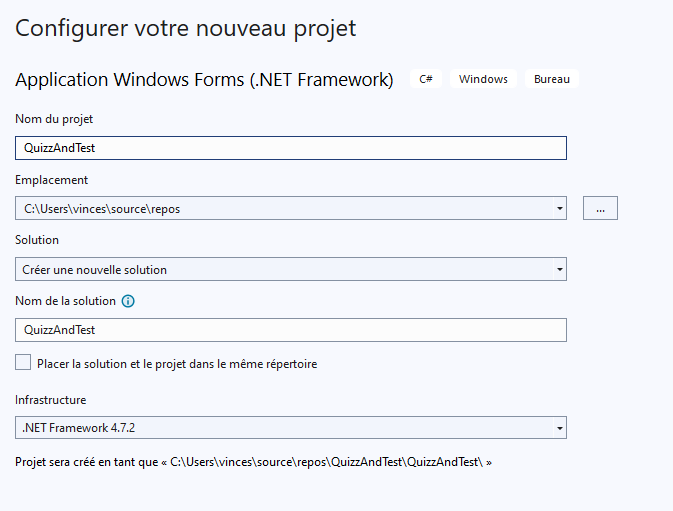
Pour créer votre application, vous devez créer un « projet » :

- En choisissant le lien dédié sur la page précédente,

L’IDE vous demande alors de choisir le type de projet que vous souhaitez mettre en place. Nous travaillerons avec le type « Application Windows Forms (.NET Framework)».



L’interface vous demande le nom de votre projet, par défaut WindowsFormsApp1. Renommez le nom du projet afin de ne pas laisser le nom par défaut ainsi que **l'emplacement du projet** (Dans votre dossier personnel ou sur votre clé USB par exemple).

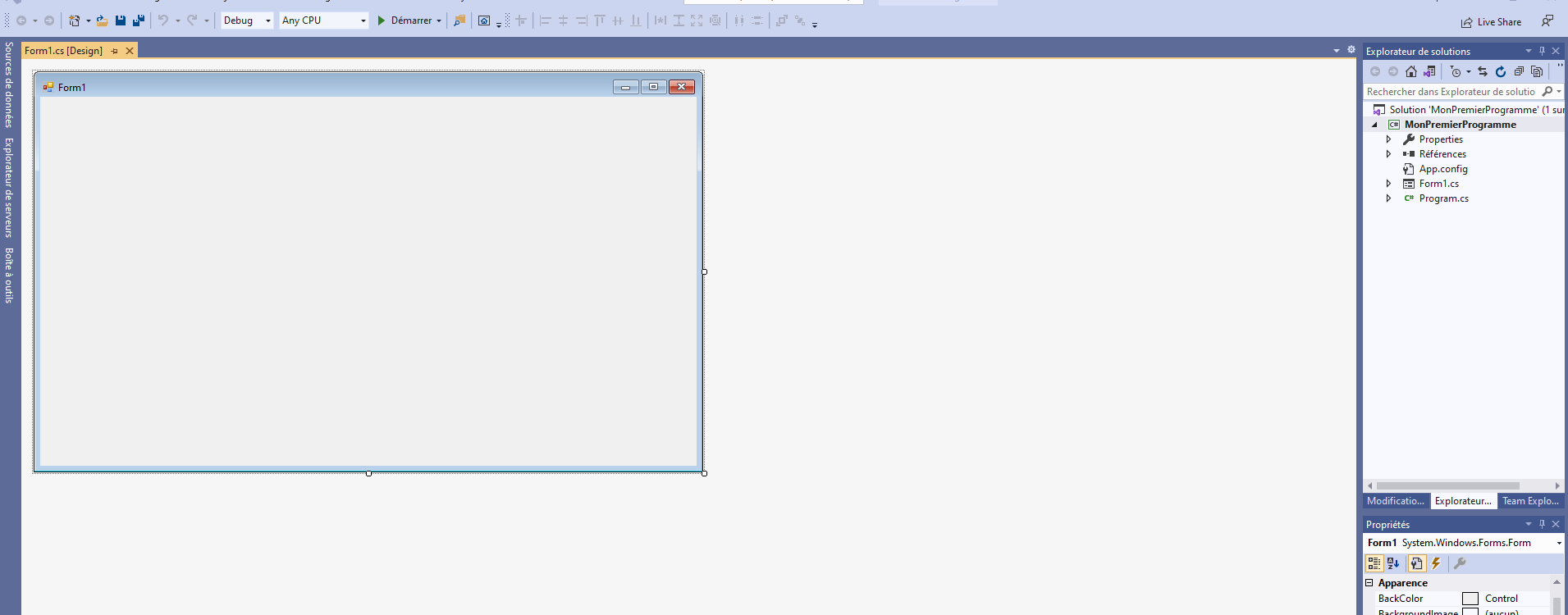


Selon la version de Visual Studio, il est possible que certains types de projets n’apparaissent pas. Vous utiliserez principalement les types qui suivent :

- **Application Windows Form** : des projets basés sur des interfaces Windows simples,

- **Bibliothèques de classes** : des projets pour créer vos propres composants.

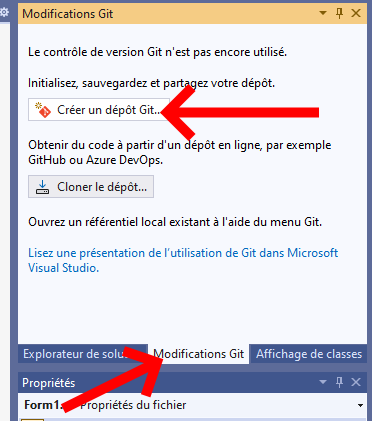
L’interface de l’IDE s’ouvre enfin, elle se présente ainsi (vue partielle) :

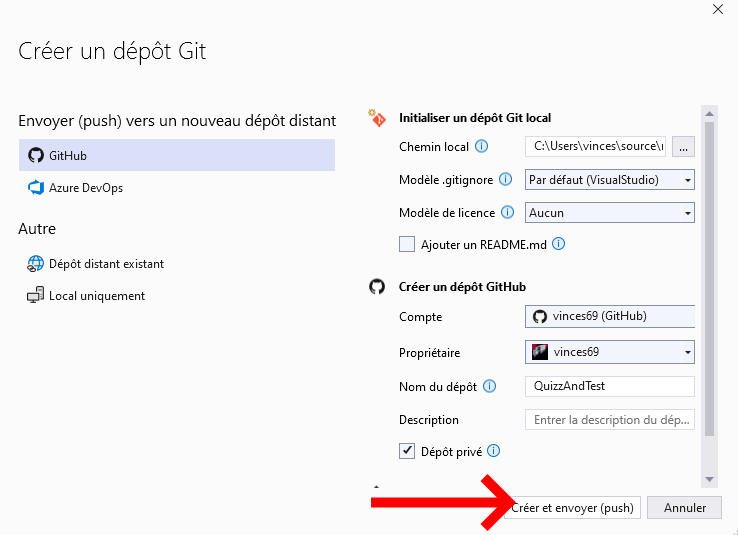


### Phase 2 : Enregistrement du projet

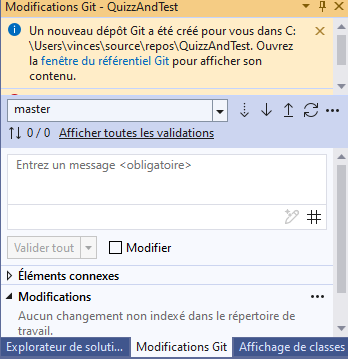
Pour enregistrer le projet pour la première fois, il suffit d’utiliser la démarche traditionnelle sous Windows (menu Fichier – Enregistrer Tout) ou utiliser le raccourcis clavier CTRL + la touche « S ».

Il est fortement conseillé d’enregistrer votre projet sur GitHub. Pour cela il faut aller dans





Maintenant votre projet est sauvegardé sous GitHub et vous pouvez faire des Commit et Push depuis cet écran.

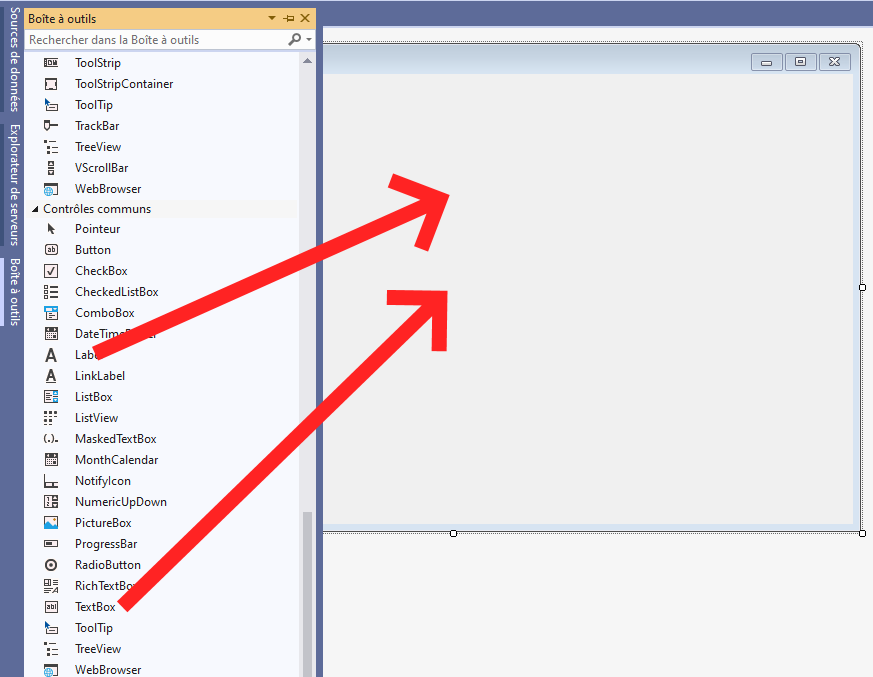


### Phase 3 : Insertion de contrôles commun dans une fenêtre

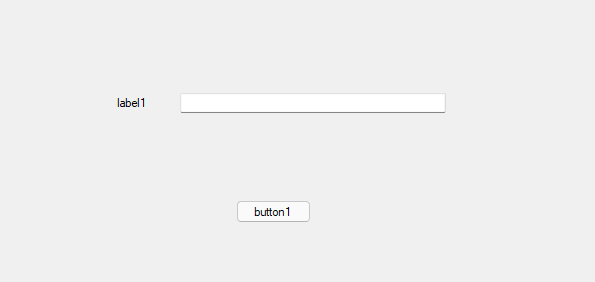
Pour insérer un contrôle, il faut ouvrir la boite à outil. C’est là que tous les contrôles que vous utiliserez seront présents. (Il faut être sur une vue pour que ce menu apparaisse).

### 

Maintenant vous devez disposer sur l’écran un label, un bouton et une textbox. Pour cela il suffit de faire un glisser déposer depuis la barre d’outils vers votre écran. **Attention, le volet « Boite à outils » n’apparait que pour le mode design.**



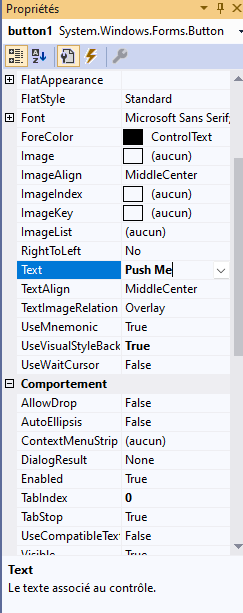
Une fois réalisé, essayez d’avoir le résultat suivant :



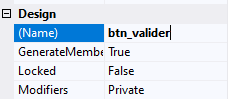
### Phase 4 : Personnalisation des contrôles positionné dans la fenêtre

Pour modifier le texte du bouton il faut aller dans les paramètres du bouton. Pour cela, il faut sélectionner le bouton en cliquant dessus. Les paramètre du bouton devrait apparaitre sur le côté droit (S’il n’est pas apparu, utiliser le raccourcis ALT+ENTRER).

Utiliser le champ « Text » pour changer le texte du bouton

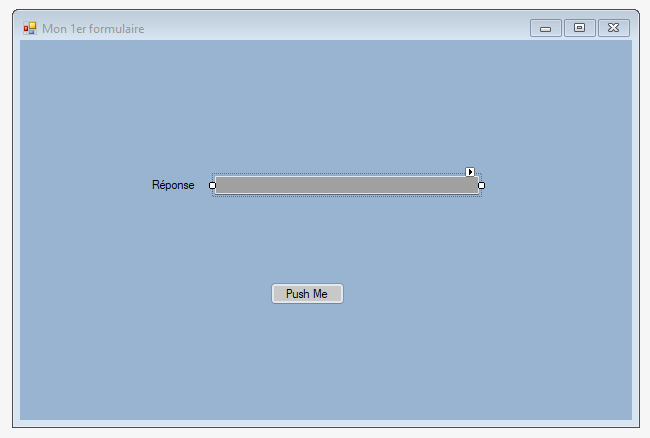


Changer le nom du bouton « button1 » en « btn\_valider ».



Changer le nom du textbox en « txt\_afficher » et le nom du label en « lbl\_rep »

Dans les propriétés du textbox, cherchez la propriété enable et passez-la à « False ». Puis tentez de reproduire l’affiche suivant grâce aux différents paramètres dans les propriétés.



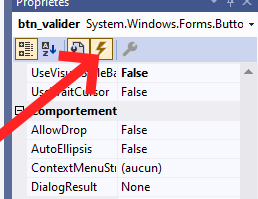
**Il est important de nommer correctement les objets que vous créez. En effet, je vous conseille de nommer les objets de cette façon : « Type d’objet » \_ « Fonction ».**

**Cette bonne pratique vous fera gagner un temps précieux surtout lorsqu’il y a beaucoup de type d’objet sur le même formulaire.**

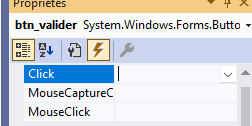
**IL est essentiel de nommer de la même façon l’ensemble des objets que vous créez**

### Phase 5 : Mise en place de code dans l’application

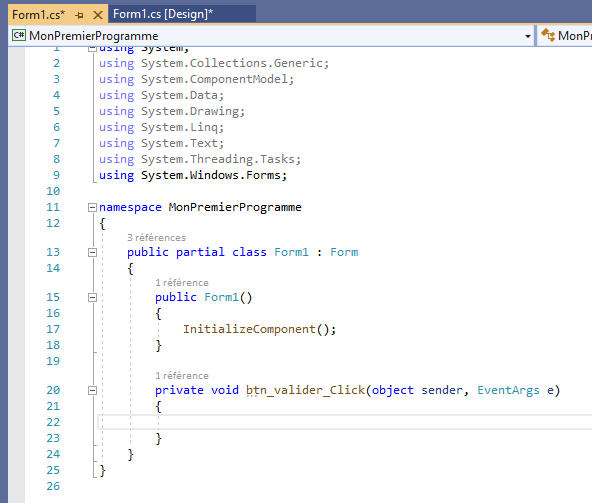
Pour afficher un message lors de l’appuis du bouton, il faut cliquer sur le bouton et cliquer sur le bouton évènement dans les propriétés.



Double cliquez sur le champ à coté de Click

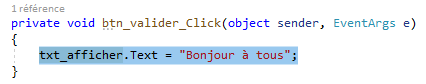


L’écran suivant devrait apparaitre



Insérez le code suivant :

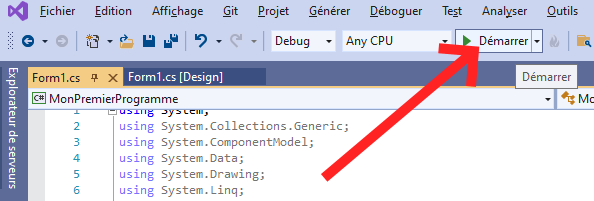
txt\_afficher.Text = "Bonjour à tous"; //Bonjour le commentaire



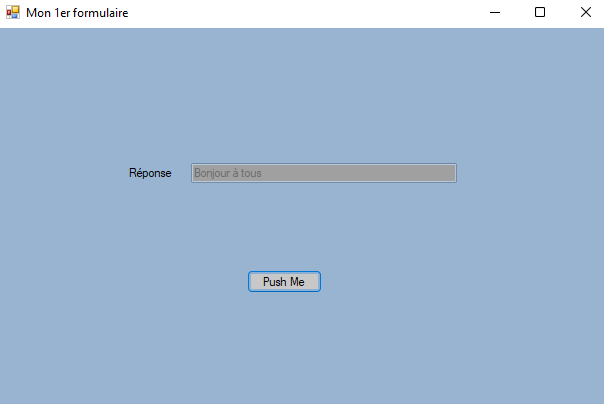
Ici nous utilisons une action (l’appuis sur le bouton) pour que le code s’exécute.

### Phase 6 : Test de l’application

Pour lancer l’application il faut cliquer sur le bouton démarrer.



L’écran suivant devrais s’afficher et lors du clic sur le bouton le message devrait s’afficher dans la textbox.

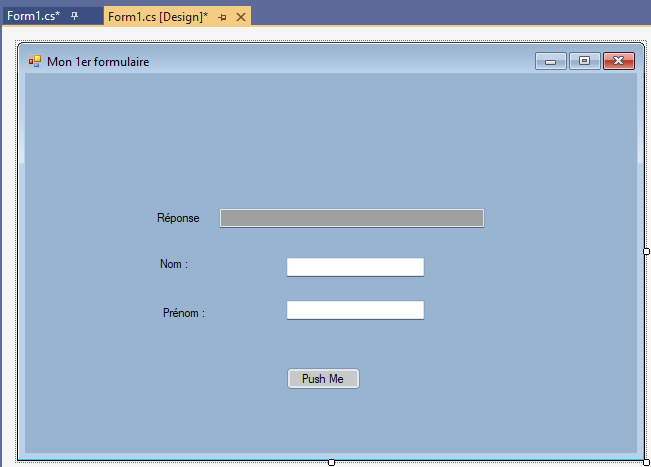


**Si vous avez fermé votre onglet design il suffit de double cliquer sur le form correspondant.**

**Si vous avez fermé votre onglet contenant le code, il suffit de sélectionner le form et d’appuyer sur F7**

### Phase 7 : Récupérer des données dans des textbox et afficher un message

Reproduire l’affichage suivant en n’oubliant pas de renommer les textbox que vous venez d’ajouter « txt\_ <nom à afficher> »



Modifier votre code par celui-ci :

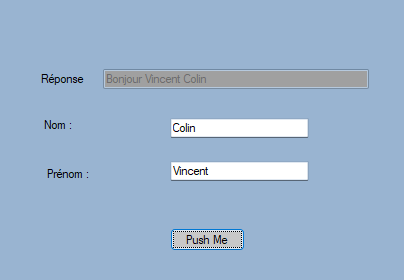
string result = "Bonjour ";

result += txt\_nom.Text ;

txt\_afficher.Text = result;

Vérifier que lorsque vous remplissez votre nom et que vous cliquez sur le bouton le message « Bonjour <Votre nom> » apparait.

Modifier votre code pour que le message « Bonjour <Votre prénom>  <Votre nom> »



### Phase 8 : Contrôle des données dans les textbox

Pour forcer l’utilisateur à remplir les champs on utilise le code suivant :

if (txt\_nom.Text != "")

{

// votre code

}

Adaptez le code pour tester si tous les champs sont remplis.

### Phase 9 : Afficher un message dans une boite de dialogue à l’utilisateur lorsque le nom ou le prénom n’est pas remplis

Pour afficher une boite de message à l’utilisateur, on utilise le code suivant : MessageBox.Show("Aucun nom n'est rentré", "Erreur",MessageBoxButtons.OK,MessageBoxIcon.Warning);

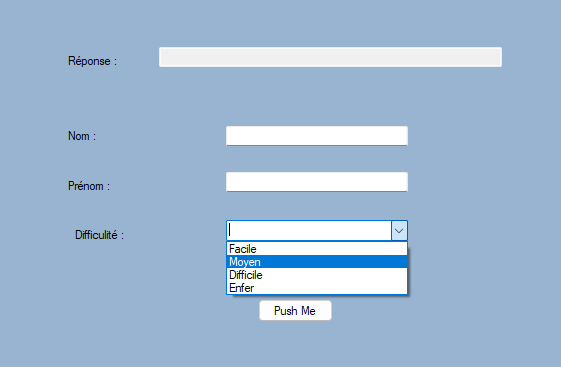
Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

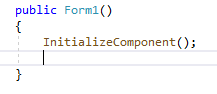
Affichez une boite de dialogue lorsque le nom ou le prénom n’est pas rentré

### Phase 10 : Utilisation d’un combobox

Reproduire l’affichage suivant :

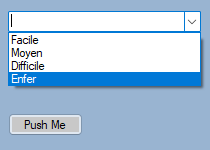


Pour charger un combobox au lancement de l’écran il faut ajouter son code ici :



Pour ajouter des éléments on utilise le code suivant : <Nom du combobox>.Items.Add("item");

Ajouter les items suivant dans le combo box : Facile, Moyen, Difficile, Enfer



Lorsque la difficulté « Enfer » est sélectionné, le fond d’écran doit devenir rouge et le texte de chaque label doit être en gras.

Pour cela on va utiliser l’événement SelectIndexChanged du combobox

Pour connaitre la valeur sélectionné on utilise le code suivant : <Nom du combobox>.SelectedItem

Pour changer la couleur de fond de la Form on utilise le code suivant : this.BackColor = Color.Red;

Pour mettre en gras le text d’un label on utilise le code suivant :

<Nom du label>.Font = new Font(<Nom du label>.Font, FontStyle.Bold);

Pour retrouver la couleur d’origine lorsque une autre valeur que « Enfer » est sélectionnée on utilise le code suivant : Color.FromArgb(153, 180, 209);

Remettre les labels en normal lorsqu’une autre valeur que « Enfer » est sélectionnée.

Ajouter au texte lorsque l’on appuis sur le bouton : \r\n La difficultés sélectionnée est : <Valeur sélectionnée>

Interdire la saisie de texte en modifiant la propriété DropDownStyle

### Phase 11 : Résolution d’une erreur typique

Si vous double cliquer sur le champ nom

Une image contenant capture d’écran, texte, ligne, Rectangle

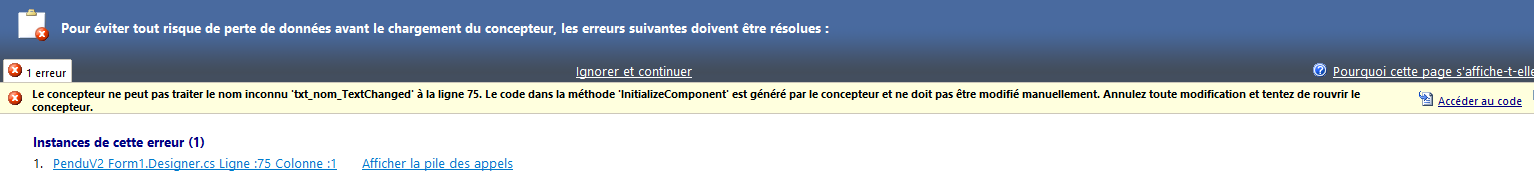
Description générée automatiquement

Vous devriez arriver sur l’écran où sont stockées les différentes actions.

Une image contenant texte, ligne, Police, Tracé

Description générée automatiquement

Vous venez de créer une action non désirée. Si vous effacez le code généré, vous obtiendrez l’erreur suivante en retournant sur la page de génération de l’affichage.



Pour résoudre cette erreur, il faut cliquer sur le lien Accéder au code

Une image contenant texte, Police, capture d’écran, Rectangle

Description générée automatiquement

Puis il faut effacer seulement la ligne qui possède l’erreur

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

**ATTENTION SI VOUS EFFACER> PLUSIEURS LIGNES QUI NE SONT PAS EN ERREUR, VOTRE PROJET RISQUE DE NE PLUS FONCTIONNER !!**

En retournant dans le Design le formulaire devrais réapparaitre.

### Phase 12 : Création d’un nouveau form et liaison

Nous allons créer un nouveau Form, pour cela il faut cliquez droit sur le nom de votre projet et choisir : Ajouter > Nouvel élément

Il faut ensuite choisir Formulaire WindowForm

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

**N’oubliez pas de changer le nom de votre formulaire !! Puis cliquez sur Ajouter**

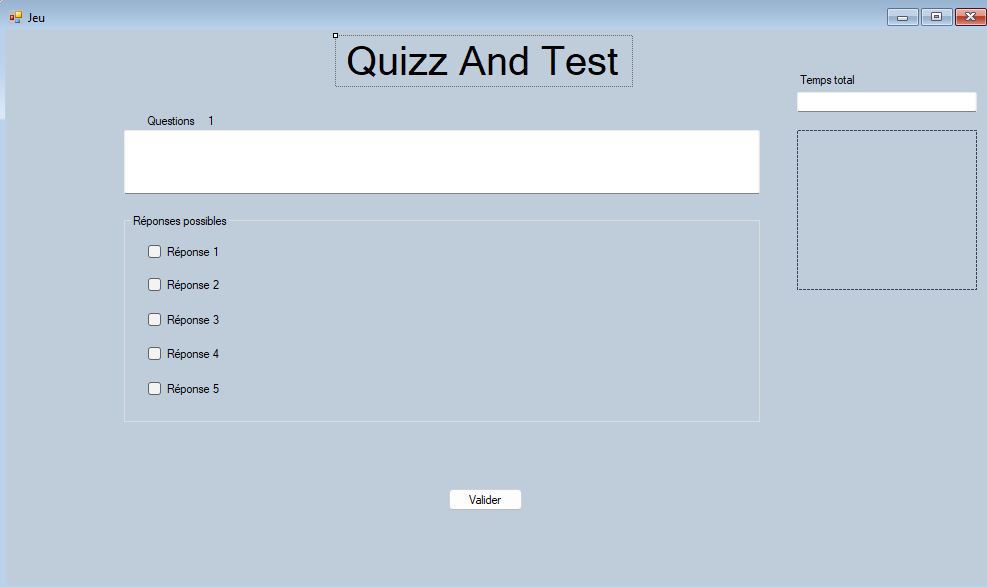
Normalement le nouveau formulaire devrait apparaitre dans votre explorateur de solution.

**Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement**

En double cliquant sur votre nouveau formulaire vous pourrez accéder au design

Reproduire l’affichage suivant :



**N’oubliez pas de bien nommer vos boutons, les 2 textbox, ainsi que la picturebox.**

**La picturebox permettra de faire défiler des images permettant de voir si la réponse est bonne ou non.**

**La grande textbox permettra d’afficher les questions.**

Les textbox ne doivent pas pouvoir être cliquable.

### Phase 13 : Modification de l’ancien form

Modifiez le Form1 pour qu’il ait cet aspect

Une image contenant texte, capture d’écran, logiciel, conception

Description générée automatiquement

Des erreurs dans votre code devraient apparaitre, corrigez-les !!

Codez le contrôle de contenu des champs Nom, Prénom et Difficulté lors de l’appuis sur le bouton valider.

Pour contrôler qu’un combobox à une valeur sélectionnée, on utilise le code suivant <combobox>. SelectedIndex > -1

### Phase 14 : Appel d’un form à un autre

Une fois tous les contrôles réalisés, vous pouvez faire appel à votre 2éme Form en utilisant le code suivant :

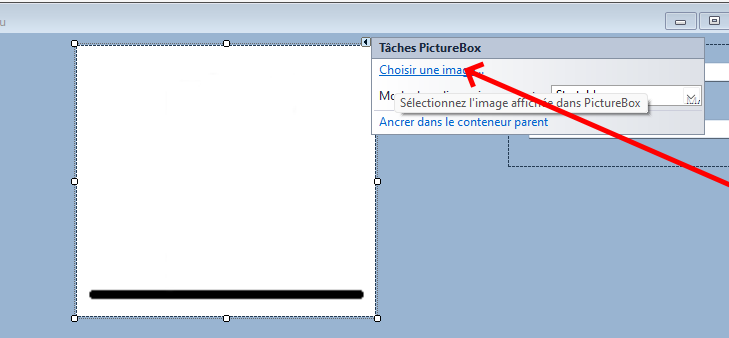
Jeu J = new Jeu();

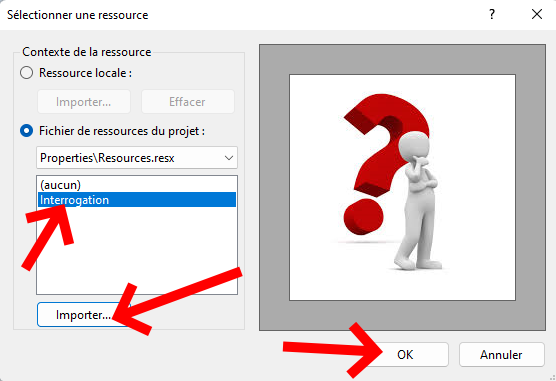
J.Show();

this.Hide();

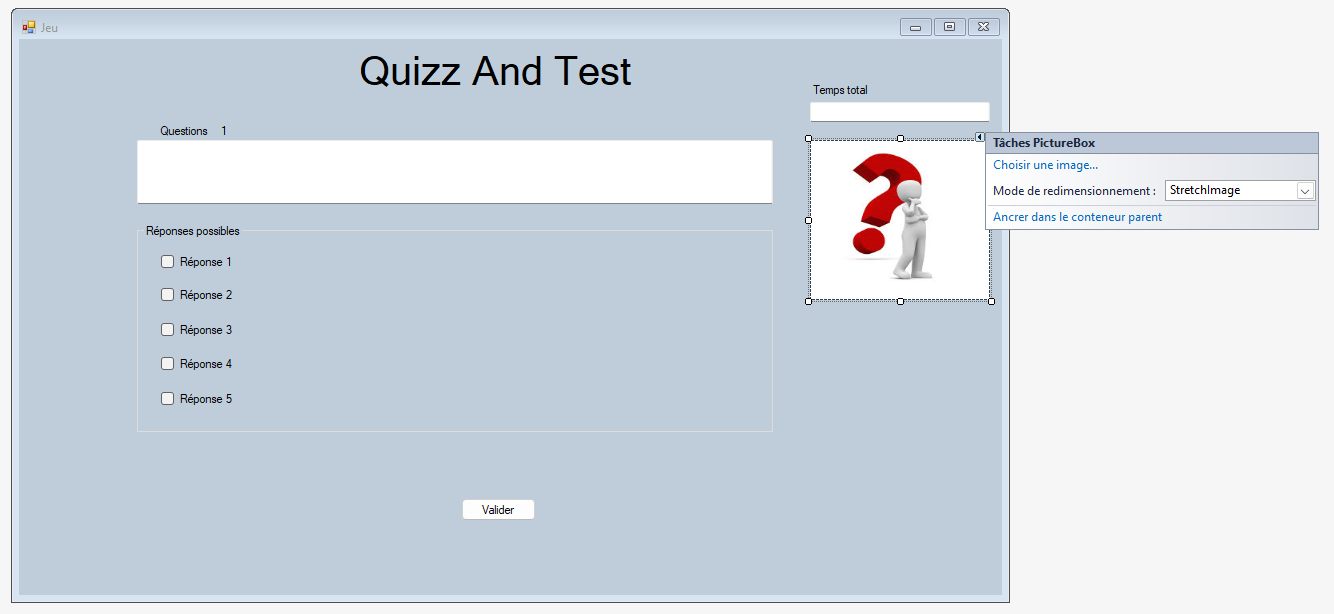
### Phase 15 : Affichage d’une image par défaut dans le pictureBox

Pour ajouter les images il vous suffit de les importer





Puis une fois l’image importées, reproduisez l’écran suivant.



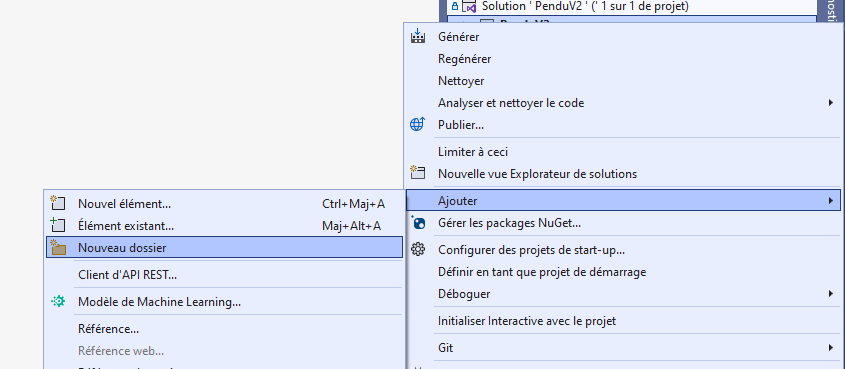
Ajouter 2 images supplémentaires afin de permettre à l’utilisateur de savoir si il a choisi la bonne réponse ou non.

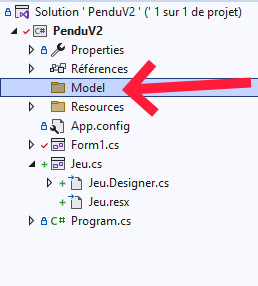
Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Maintenant nous allons réaliser le code permettant de jouer. Pour cela nous allons créer un nouveau fichier contenant le code que nous allons utiliser dans le formulaire jeu.

Nous allons ajouter un dossier qui va contenir l’ensemble de nos objets que nous allons créer





Maintenant nous allons créer la **classe** « Question ». Cette classe va nous permettre de formater les données afin de n’oublier aucuns éléments d’une question.

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Ensuite il faut recopier le code suivant :

Une image contenant texte, capture d’écran, ligne

Description générée automatiquement

Maintenant nous allons créer la **classe** « Partie » en utilisant la même façon que l’étape précédente.

Une partie possède :

* Un score de type numérique
* Une difficulté de type numérique
* Le nombre de questions de la partie de type numérique
* Une liste de questions de type List<Question>
* Le numéro de la bonne réponse
* Le numéro de la questionx

Créer les attributs correspondants.

Créer un constructeur permettant d’initialiser les attributs.

### Phase 16 : Créer le constructeur de la classe Partie

Créez dans la classe partie et complétez le constructeur suivant :

public Partie(List<Question> ListeQuestions)

{

//Mettre le score à 0

//Mettre la difficulté à 0

//Mettre le numéro de question à 0

//Attribuez la liste des questions au bon attribut

//Comptez le nombre de question dans la liste des questions avec la méthode count

}

### Phase 17 : Coder le quizz

Maintenant vous allez créer une méthode permettant de changer l’image en fonction de la réponse choisi par l’utilisateur.

private void changerImg(PictureBox PbImage,bool BonneReponse,bool raz)

{

if (!raz)

{

if (BonneReponse)

{

PbImage.Image = Properties.Resources.vrai;

}

// Code à completer !!!!!!!!!!!!

}

}

Vous allez créer une nouvelle méthode publique qui prend en paramètre la réponse de l’utilisateur, la réponse valide de la question et une PictureBox . Appelez cette fonction « validerReponse ».

public void validerReponse(int reponse, PictureBox PbImage)

{

if (reponse == reponseValid)

{

calculerScore(true); // Il faut créer cette méthode privé qui ajoute +1 au score à chaque bonne réponse

changerImg(PbImage, true, false);

}

//code à compléter

}

### Phase 18 : Changer les questions

Dans votre classe partie, vous allez coder 3 méthodes permettant de changer les questions.

Cette première méthode permettra de tester s’il reste des questions ou si c’est la fin de partie.

public void changerQuestion( TextBox txt\_affichage, CheckBox ckb\_reponse1, CheckBox ckb\_reponse2, CheckBox ckb\_reponse3, CheckBox ckb\_reponse4, CheckBox ckb\_reponse5,Form formulaire, GroupBox gd\_reponse, PictureBox PbImage)

{

if (// On teste si il reste des questions )

{

aleatoireReponse(txt\_affichage, gd\_reponse);

// On décoche les 5 checkbox

}

else

{

//appel de la méthode de fin de partie qui sera réalisé plus tard

}

}

Cette méthode permet de mettre les propositions de réponse dans un ordre aléatoire.

private void aleatoireReponse(TextBox txt\_affichage, GroupBox gd\_reponse)

{

int bonneReponse = //récupérer la bonne réponse à la question

txt\_affichage.Text = //Afficher l’énoncé de la question

List<int> reponseAleatoire = new List<int>() { 1, 2, 3, 4, 5 };

Random rnd = new Random();

for (int i = 1; i <= 5; i++)

{

int randIndex = rnd.Next(reponseAleatoire.Count);

int random = reponseAleatoire[randIndex];

reponseAleatoire.Remove(random);

string reponse = "";

//Réaliser un switch permettant d’affecter à reponse la proposition sélectionnée de manière aléatoire

getCheckBox(i,gd\_reponse).Text = reponse;

if (bonneReponse == random)

{

reponseValidQuestion = i;

}

}

}

Méthode permettant de retourner une des 5 checkbox

private CheckBox getCheckBox(int indice,GroupBox gd\_reponse)

{

foreach (Control c in gd\_reponse.Controls)

{

if (c.GetType() == typeof(CheckBox) && c.Name == "ckb\_reponse" + indice.ToString())

{

return ((CheckBox)c);

}

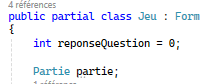
}

return null;

}

### Phase 19 : Liaison de notre Classe Question et Partie avec notre formulaire Jeu

Allez dans le code du formulaire Jeu. Nous allons instancier la classe Partie, ce qui nous permettra de créer un objet que nous pourrons utiliser dans le code du formulaire. On utilise le code suivant :



partie = new Partie(); //à placer dans le constructeur de la classe partie

**Question : Expliquez la raison de la déclaration séparée de l’instanciation ?**

Code du constructeur du form Jeu

public Jeu()

{

InitializeComponent();

List<Question> ListeQuestions = new List<Question>();

ListeQuestions.Add(new Question("Quelle est la capitale de la France ?", 1, 1, "Paris", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de l'Espagne ?", 4, 1, "Paris", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de l'Allemagne ?", 3, 1, "Paris", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de l'Italie ?", 5, 1, "Paris", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de l'Angleterre ?", 2, 1, "Paris", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de la Belgique ?", 1, 1, "Bruxelles", "Londres", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de la Suisse ?", 2, 1, "Bruxelles", "Berne", "Berlin", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale du Luxembourg ?", 3, 1, "Bruxelles", "Berne", "Luxembourg", "Madrid", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale du Portugal ?", 4, 1, "Bruxelles", "Berne", "Luxembourg", "Lisbonne", "Rome"));

ListeQuestions.Add(new Question("Quelle est la capitale de l'Autriche ?", 5, 1, "Bruxelles", "Berne", "Luxembourg", "Lisbonne", "Vienne"));

partie = new Partie(ListeQuestions);

partie.changerQuestion(txt\_affichage,ckb\_reponse1,ckb\_reponse2,ckb\_reponse3,ckb\_reponse4,ckb\_reponse5,this,gb\_reponse,pb\_question);

}

On met le code juste après la ligne « InitializeComponent(); », comme cela le code s’exécutera lorsque la fenêtre sera généré

### Phase 20 : Gestion du bouton Valider

Double cliquer sur le bouton « Valider » de votre formulaire jeu pour créer l’événement Click

Une image contenant texte, ligne, Police, capture d’écran

Description générée automatiquement

Ajouter le code suivant :

partie.validerReponse(reponseQuestion, pb\_question);

partie.numeroQuestion++;

partie.changerQuestion(txt\_affichage, ckb\_reponse1, ckb\_reponse2, ckb\_reponse3, ckb\_reponse4, ckb\_reponse5, this, gb\_reponse,pb\_question);

lbl\_numero.Text = (partie.numeroQuestion+1).ToString();

reponseQuestion = 0;

Vous pouvez désormais tester si votre code marche.

### Phase 21 : Décocher les autres cases à cocher lorsque l’on choisit une autre réponse

Pour cela vous devez lancer l’événement sur le click de toutes les cases à cocher.

//Boucle permettant de décocher toutes les cases à cocher du formulaire

foreach (var box in gb\_reponse.Controls.OfType<CheckBox>())

{

box.Checked = false;

}

((CheckBox)sender).Checked = true;

**//Mettre dans une variable la réponse choisis par l’utilisateur**

### Phase 22 : Gérer la fin de partie

Pour cela nous allons vérifier à la fin du code dans l’événement « Changer de question » si il reste encore des questions dans la liste de question. Une fois toutes les questions réalisées, un message apparait et propose de rejouer une partie et affichant le score réalisé durant la partie.

Vous allez créer 1 méthode dans votre classe Partie ayant pour nom finDePartie

Dans la méthode FinDePartie, il faut :

* Déclarer une variable msg de type DialogResult
* Puis on affiche le message suivant :

msg = MessageBox.Show("Votre score est de " + score + ".\r\n Voulez vous rejouer", "Fin de la partie"

, MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);

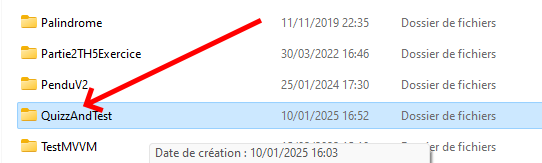
* Si oui alors (msg == DialogResult. Yes)
  + - On remet à 0 le score
    - On remet l’image par défault
    - On affiche la 1ère question de la liste
* Si non on revient à l’écran précédent
  + - Form1 Accueil = new Form1();
    - Accueil.Show();
    - formulaireJeuActif.Hide();

Appelez la nouvelle méthode la où il est nécessaire de l’appeler.

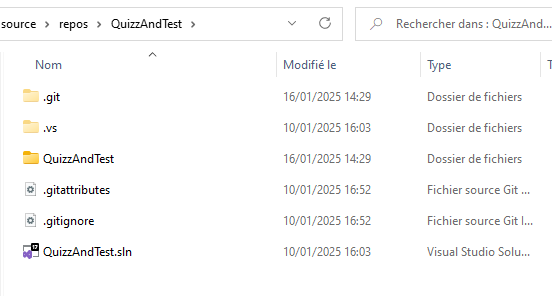
Une fois fini, tester si tout fonctionne bien.

**N’oubliez pas de garder le projet une fois terminé nous le réutiliseront dans le TP suivant**

Si vous souhaitez sauvegarder votre projet dans votre espace personnel ou sur une clé USB il faut copier **l’ensemble du dossier** qui contient le projet.



Dans ce dossier vous devez avoir l’arborescence suivante pour que le projet puisse être ouvert et fonctionner sur une autre machine.



SI VOUS NE COPIEZ PAS L’ENSEMBLE DU PROJET, VOUS NE POURREZ PAS CONTINUER A TRAVAILLER AVEC CE PROJET.