Documentation Technique

Application ASL

Table des matières

[Présentation de l’application 2](#_Toc72811797)

[I. Architecture et application 2](#_Toc72811798)

[II. Base de données 3](#_Toc72811799)

[III. C# DAO Main activity 4](#_Toc72811800)

[a) Ateliers 4](#_Toc72811801)

[b) Stand 4](#_Toc72811802)

[c) Inscription 4](#_Toc72811803)

[d) Liste 5](#_Toc72811804)

[e) Partenaire 5](#_Toc72811805)

[IV. C# Class 6](#_Toc72811806)

[g) Atelier 6](#_Toc72811807)

[h) Stand 6](#_Toc72811808)

[i) Participant 6](#_Toc72811809)

[j) Bénévoles 10](#_Toc72811810)

[V. Fonction C# DAO 11](#_Toc72811811)

[a) DAOAtelier 11](#_Toc72811812)

[b) DAOStand 11](#_Toc72811813)

[c) DAOParticipant 12](#_Toc72811814)

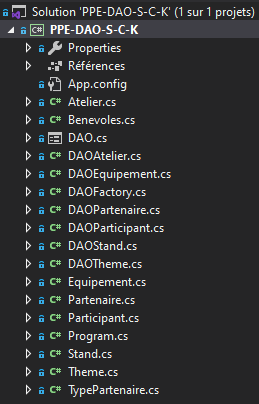
# Présentation de l’application

## Architecture et application

Il s’agit d’une application sous c# (client lourd)

L’application est séparée entre 2 parties :

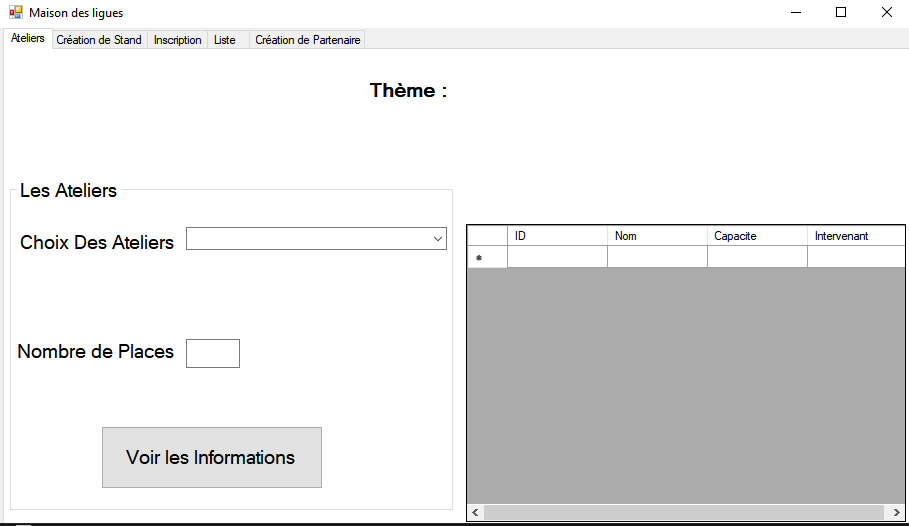
### Partie Codage :

L’application est au format objet

Les Class définisse les attribue constructeur et méthode de chaque objet

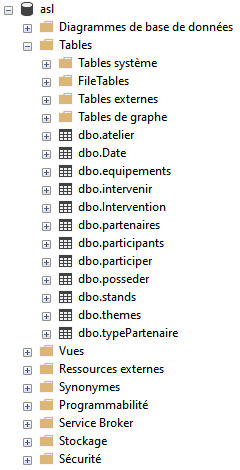
Les Class DAO définisse les méthodes qui effectue des requêtes destinées à la bdd « asl » de Sql Serveur.

### Partie Graphique :

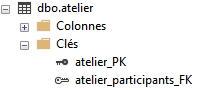
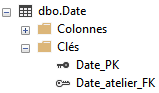


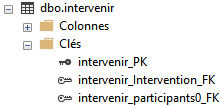
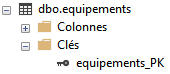
## Base de données

Il s’agit de la base de données de l’application, elle est sous SQL Serveur :

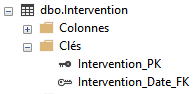


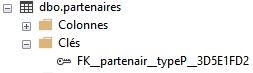
Relations entre les tables :

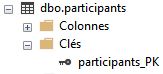


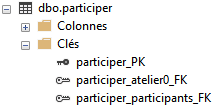


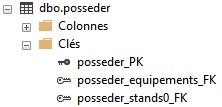
(Relations entre les tables Suite : )

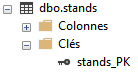


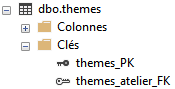


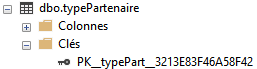












## C# DAO Main activity

### Ateliers

Une image contenant texte

Description générée automatiquement

Lorsque l’on entre dans la page Atelier, les données seront rentrée automatiquement, il faut néanmoins préparer une condition où les données ne se répètent pas en boucle.

Il y’a un seul bouton dans la page.

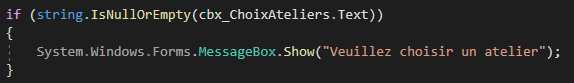
Une image contenant texte

Description générée automatiquement

Lorsque l’utilisateur clique dessus, il se déclenchera plusieurs évenements :



-On vérifie qu’un atelier est choisi



-On rempli les données de l’atelier dans un tableau et on met en évidence la capacité de celui-ci. Une image contenant texte, capture d’écran, écran

Description générée automatiquement

-Enfin, on affiche les thèmes qui correspond à l’atelier que l’utilisateur.

Pour cela , on utilise une fonction conditionnelle switch case avec les ateliers.

### Stand

private void Btn\_creationStand\_Click(object sender, EventArgs e)

Cette fonction permet de créer un Stand dès que l’on clique sur le bouton.

private void txt\_Nalle\_KeyPress(object sender, KeyPressEventArgs e)

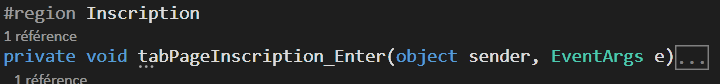
private void txt\_Nordre\_KeyPress(object sender, KeyPressEventArgs e)

private void txt\_surface\_KeyPress(object sender, KeyPressEventArgs e)

private void tbx\_nbrSiege\_KeyPress(object sender, KeyPressEventArgs e)

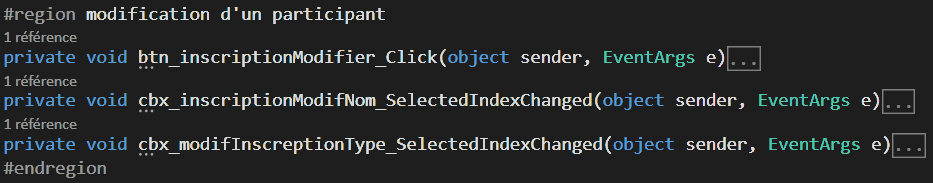
Ces fonctions permettent à l’utilisateur de l’application de ne pas entrer autre chose que des caractères numériques dans les textbox, txt\_Nalle, txt\_Nordre, txt\_surface, tbx\_nbrSiege

### Inscription



tabPageInscription\_Enter(…) : s’active en entrant dans la page du formulaire.

Est utilisé pour insérer les données dans les différents selecte dès l’entrée dans la page.

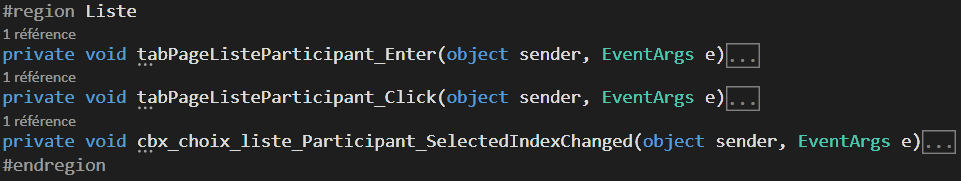


Btn\_inscriptionModifier\_Click( … ) : s’active aux cliques sur le bouton. Est utilisé pour crée où modifier un participant, bénévoles ou intervenant, en local et en BDD.

Cbx\_inscriptionModifNom\_SelectedIndexChanged(…) : s’active au changement d’indexe. Est utiliser pour savoir quel participant et sélectionné et renseigné c’est donné dans les champs du formulaire… ou permets de sélectionnée la création d’un nouveau participant si l’index 0 est sélectionné.

cbx\_modifInscreptionType\_SelectedIndexChanged(…) ; s’active au changement d’indexe. Est utilisé pour savoir si le type bénévole est sélectionné, auquel cas le formulaire ajoute un champ.

### Liste



tabPageListeParticipant\_Enter(…) : s’active en entrant dans la page du formulaire. Est utilisé pour insérer les données dans le sélect « cbx\_choix\_liste\_Participant » dès l’entrée dans la page.

tabPageListeParticipant\_Click(…) : est une erreur ( miss clique)

cbx\_choix\_liste\_Participant\_SelectedIndexChanged(…) : s’active au changement d’index. Est utilisé pour alimenter le Data Gris View (DGV\_ListeParticipant) en fonction de la liste sélectionnée.

### Partenaire

## C# Class

### Atelier

Les attribues utilisés :

Une image contenant texte

Description générée automatiquement

Il y a un constructeur sans paramètres et un avec. Une image contenant texte

Description générée automatiquement

Les différentes méthodes hors getter et setter sont :

ajouterParticipant(Participant participant) : ajoute un objet Participant dans la liste des participants

supprimerParticipant(Participant participant) : supprime un objet Participant dans la liste des participants

allAteliers() : récupère touts les ateliers en une liste formé par la requête de la Base de donnée de DAOAtelier

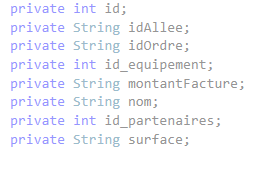
participe() : Récupère le résultat de la requête inner join de DAOAtelier pour avoir la liste des participants d’un atelier

Une image contenant texte

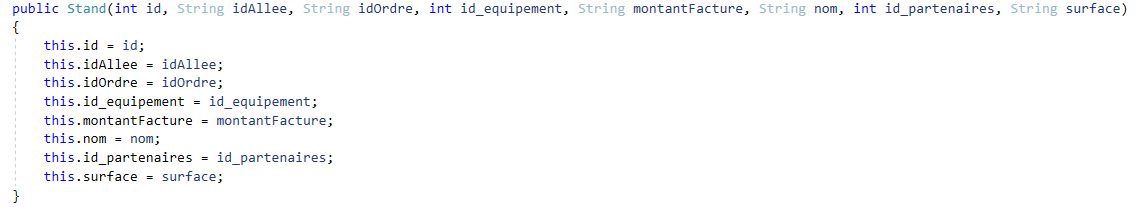
Description générée automatiquement

### Stand

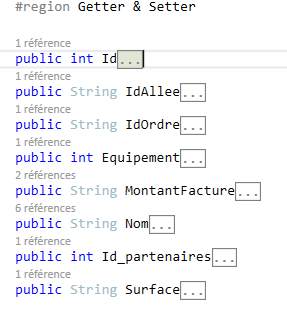
Attribue :



Constructeur :



Getter, setter (Permet de lire ou écrire les données des attributs) :

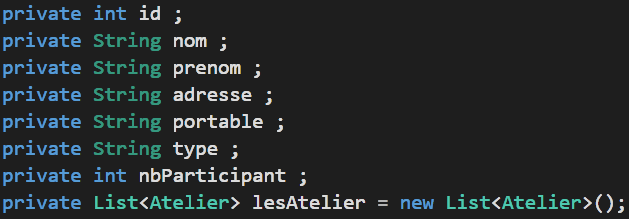


public void ajoutdbStand()

Permet d’ajouter un Stand

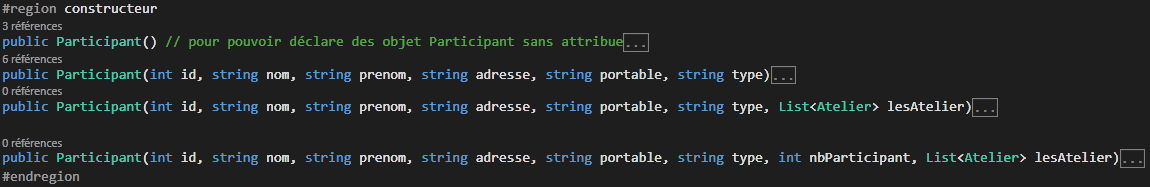
### Participant

Attribue :

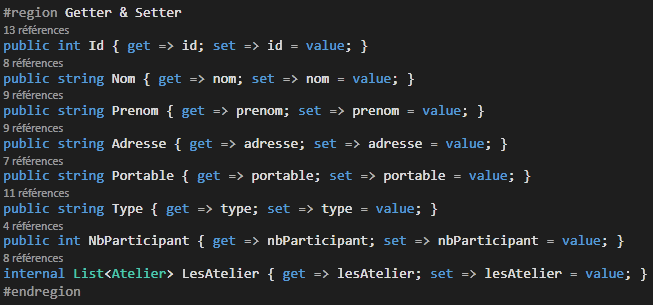


Constructeur :

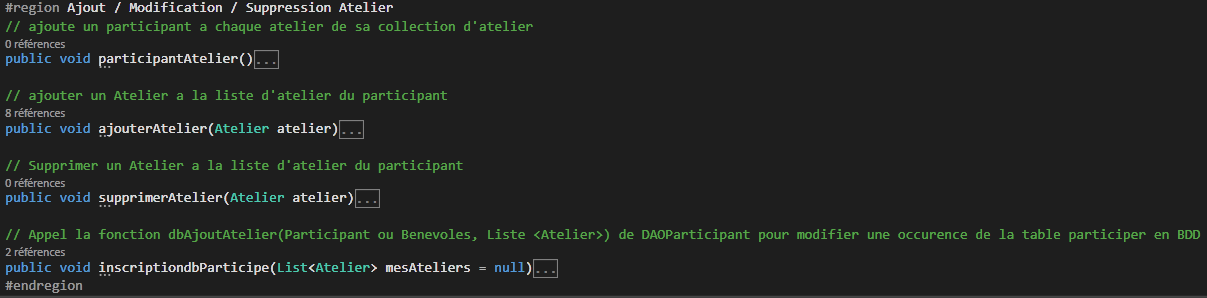
Il existe sous 4 formes (redéfinition) pour faire du polymorphisme en fonction du nombre d’attribue donnée



Actuellement la dernière redéfinition est inutile, car nbParticipant n’est pas utilisé.



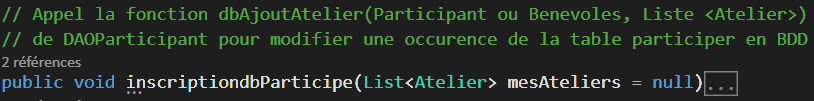
Permet de lire ou écrire les données des attributs.

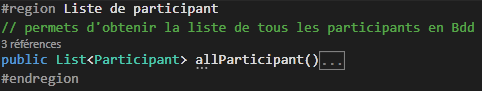


participantAtelier() : le participant sélectionné est ajouté à chaque atelier de sa liste d’ateliers.

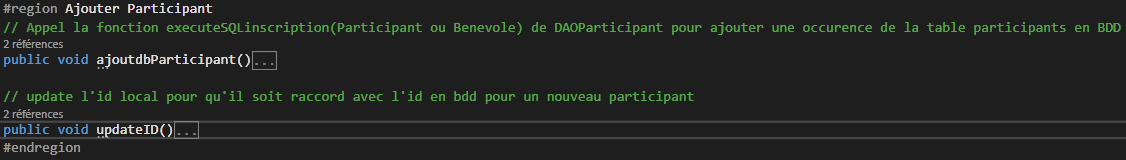
ajouterAtelier() : on ajoute un atelier à la collection « lesAtelier ».

supprimerAtelier() : on supprime un atelier de la collection « lesAtelier ».



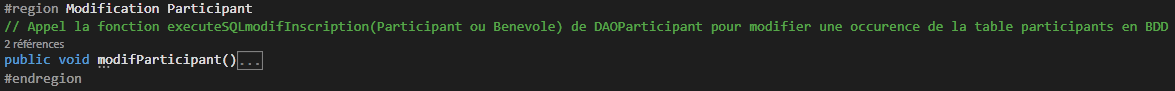


allParticipant() : récupère la collection de tous les participants existant en bdd.

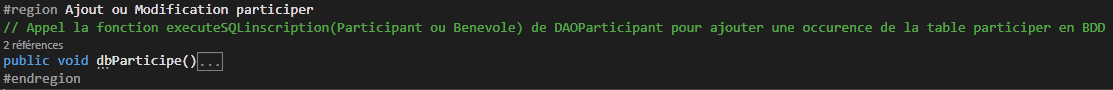


ajoutdbParticipant() : appel une fonction de DAOParticipant pour ajouter une occurrence a la table participants ( le participant, intervenant ou bénévole qui appelle la fonction)

updateID() : update l’id local du dernier inscrit pour qu’il correspondant a l’id en bdd.



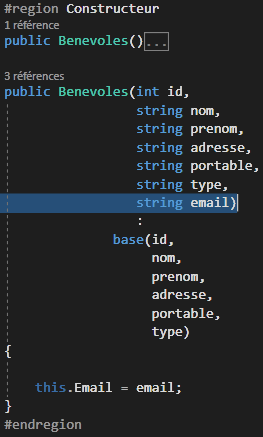
modifParticipant() : appelle une fonction de DAOParticipant pour modifier (update) une occurrence d’un participant ( ou benevole ou intervenant) en bdd (table participants).



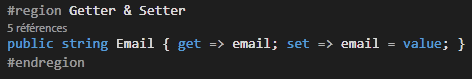
dbParticipe() : appel une fonction de DAOParticipant pour modifier, supprimé ou crée une ou plusieurs occurrences dans la table participer.

### Bénévoles





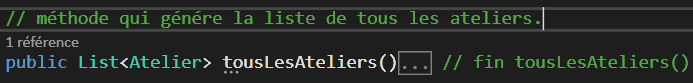
2 constructeurs, pour initier un bénévole vide (pour des besoins, pratique dans les fonctions liées avec participant) ou avec toutes les informations de base nécessaire a la construction d’un participant (constructeur le moins complet, omis le constructeur sans attribue) + l’attribue unique des bénévoles

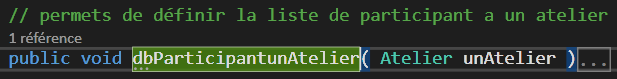


Pour valoriser ou observer la valeur de « email ».

## Fonction C# DAO

### DAOAtelier





### DAOStand

public void AjouterStand(Stand unStand )

Insère un stand dans la BDD.

public void ModifierStand(int id\_partenaire , String montant, int idStand)

Modifie l’id\_partenaire et le montant facturé dans la BDD.

public int bddUpdateID()

Prend l’id max de la table équipement.

public List<Stand> listeStand()

Cette fonction retourne la liste des stands.

public int getIdStand(string type)

Cette fonction retourne l’id stands.

public String getMontantStand(string type)

Cette fonction retourne le montant facturé

### DAOParticipant

