PROJET DE GROUPE
AGENDA A.S

Résumé

Ce projet a été réalisé à 2 dans le cadre du cours de SGBD en 2ème année.

Adrien MOUSTY

moustyadrien@hotmail.com

Table des matières

Analyse UML

Description

L'analyse UML comprend plusieurs diagrammes. Ils ont été demandés afin que l'utilisateur ayant commandé cette application puisse comprendre ce que nous voulons faire. Cela a aussi permis au client de mieux exprimer ses besoins.

Exemples

Diagramme des use-case

Interaction entre les acteurs et le système

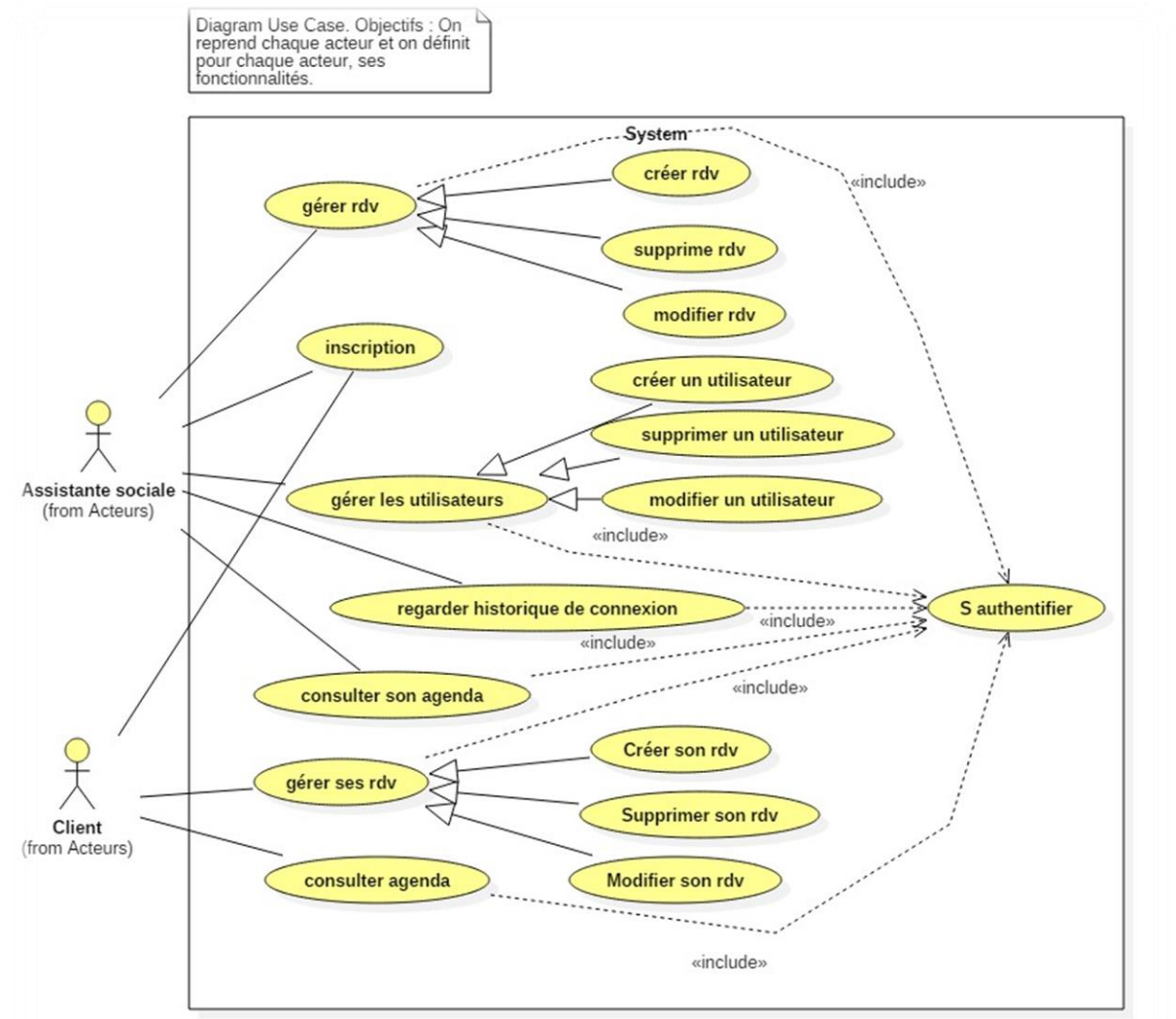
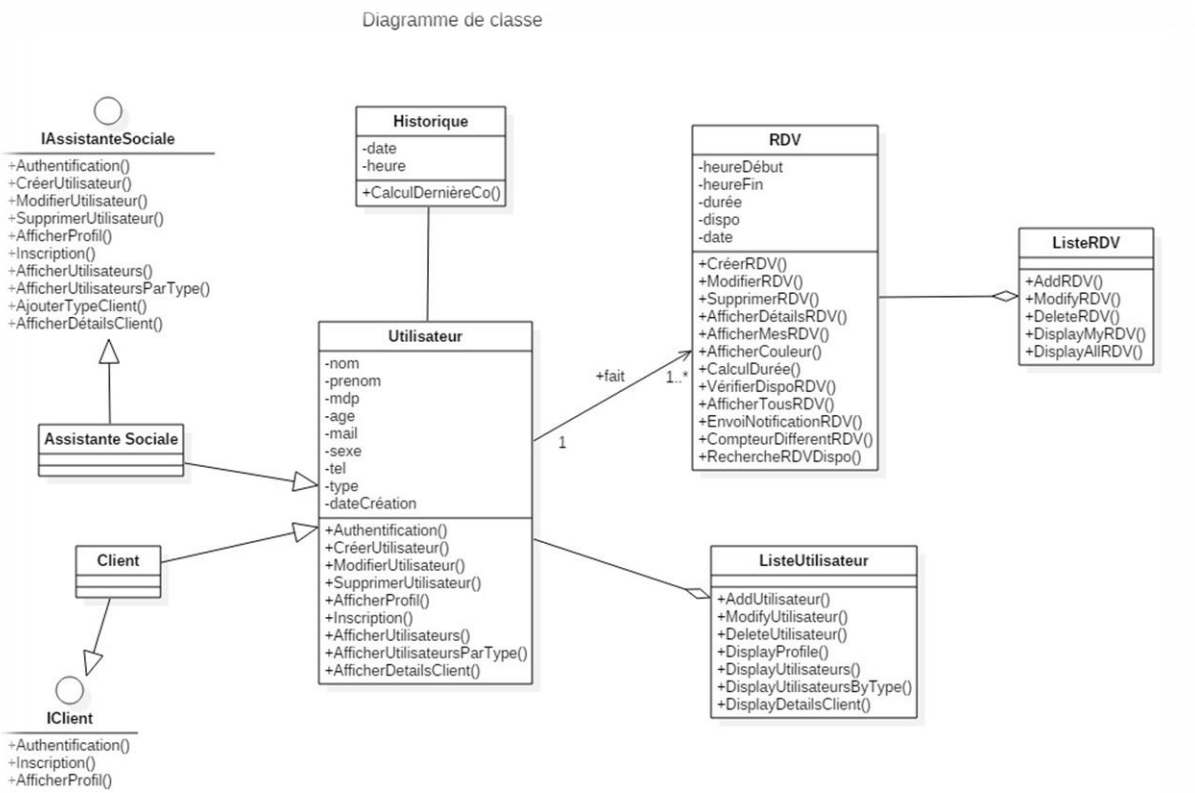


Diagramme de classe

Définit les méthodes et les attributs de la classe



GUI

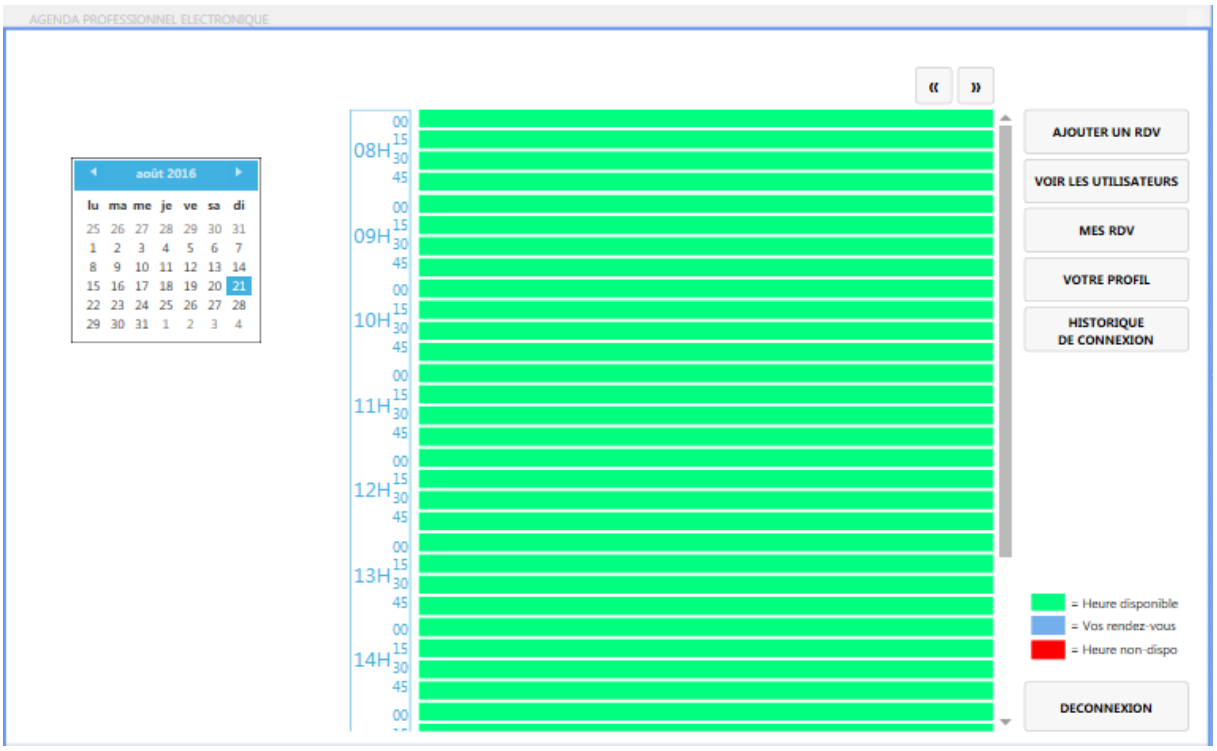
Description

Le but est de rendre l'interface pour l'utilisateur (aussi bien AS que le client) user-friendly.

Nous avons développé un système de textboxs colorées indiquant clairement quand un rendez-vous est libre ou non.

Il a été développé en WPF.

Exemples



Code behind

Description

Ce projet a principalement été développé en C# côté serveur et SQL pour gérer les données. Le tout est lié grâce au LINQ to SQL.

Langages

C#

Gestion de l'inscription – vérification des informations au préalable + utilisation de lambda expression.

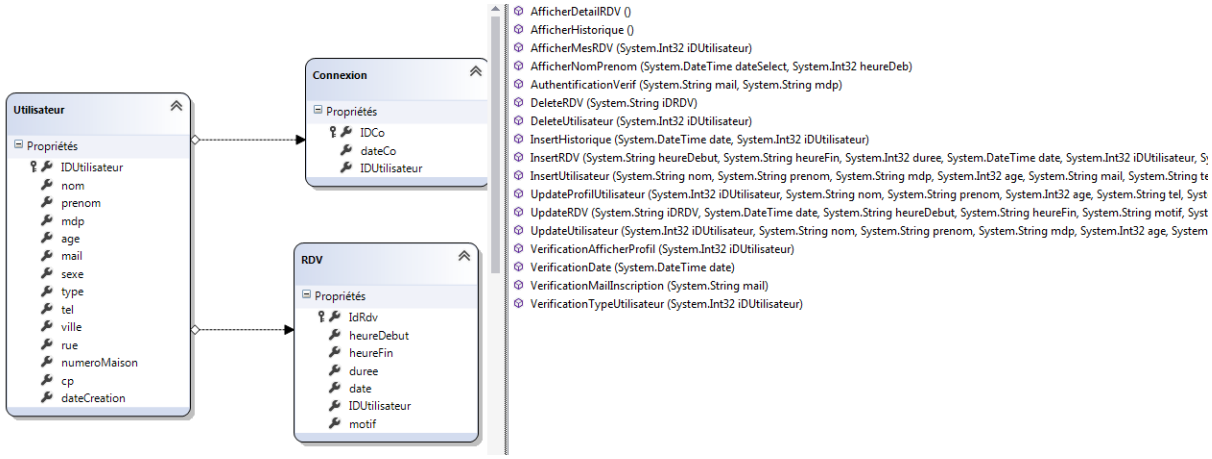
```
196         //Inscription d'un utilisateur à l'agenda
197         public async void Inscription(FenetreInscription fenetreInscription)
198         {
199             try
200             {
201                 DataClassesDataContext db = new DataClassesDataContext();
202                 //Appel de la procédure stockée pour vérifier si il existe déjà un mail correspondant à celui entré lors de l'inscription
203                 //var query = db.VerificationMailInscription(fenetreInscription.textBoxMail.Text);
204 
205                 List<Utilisateur> lambda = (db.Utilisateur).ToList();
206                 if (fenetreInscription.textBoxNom.Text.Trim() == "" || fenetreInscription.textBoxPrenom.Text.Trim() == "" ||
207                     fenetreInscription.textBoxMdp.Text.Trim() == "" || fenetreInscription.textBoxAge.Text.Trim() == "" ||
208                     fenetreInscription.comboBoxSexe.SelectedItem == null || fenetreInscription.textBoxPrenom.Text.Trim() == "" ||
209                     fenetreInscription.comboBoxType.SelectedItem == null || fenetreInscription.textBoxCP.Text.Trim() == "" ||
210                     fenetreInscription.textBoxNum.Text.Trim() == "" || fenetreInscription.textBoxRue.Text.Trim() == "" ||
211                     fenetreInscription.textBoxVille.Text.Trim() == "")
212                 {
213                     await fenetreInscription.ShowMessageAsync("", "Veuillez remplir les champs manquants.", MessageDialogStyle.Affirmative, fenetreInscription.MetroDialogOptions);
214                 }
215                 //Au lieu d'utiliser une procédure stockée pour la vérification de l'adresse mail, je fais une expression lambda
216                 //Si il existe déjà un utilisateur avec un email pareil que celui entré -> erreur
217                 else if (lambda.Any(x => x.mail == fenetreInscription.textBoxMail.Text))
218                 {
219                     await fenetreInscription.ShowMessageAsync("", "Il existe déjà un identifiant tel que le votre, veuillez choisir un autre mail.", MessageDialogStyle.Affirmative, fenetreInscription.Metr
220                 }
221                 //Sinon on enregistre les infos de l'utilisateur
222                 else
223                 {
224                     //Vérification de l'email
225                     string emailEntré = fenetreInscription.textBoxMail.Text;
226                     Regex regx = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");
227                     Match match = regx.Match(emailEntré);
228                     if (match.Success)
229                     {
230                         //On associe chaque attribut de la table user aux textbox entré
231                         this.mail = emailEntré;
232                         this.nom = fenetreInscription.textBoxNom.Text;
233                         this.prenom = fenetreInscription.textBoxPrenom.Text;
```

Interface pour l'assistante sociale ainsi que le client.

```
8   namespace AgendaAssistanteSociale.Classes
9   {
10      interface IAssistanteSociale
11      {
12          //Afficher les différents profils des utilisateurs
13          void AfficherDetailsClient(Utilisateur utilisateurSelectionné, FenetreDetailsUtilisateur fenetreDetailsUtilisateur);
14          //Créer un utiisateur
15          void CréerUtilisateur(FenetreAjouterUtilisateur fenetreAjouterUtilisateur);
16          //Modifier un utilisateur
17          void ModifierUtilisateur(Utilisateur utilisateur, FenetreModifierUtilisateur fenetreModifierUtilisateur);
18          //Supprimer un utilisateur
19          void SupprimerUtilisateur(Utilisateur utilisateurSelectionné);
20          //Modifier un RDV
21          void ModifierRDV(FenetreDetailsRDV fenetreDetailsRDV, string IDRDV);
22          //Supprimer un RDV
23          void SupprimerRDV(string IDRDV);
24          //Afficher les détails d'un RDV
25          void AfficherDetailsRDV(string heureRecup, DateTime dateSelec);
26      }
27  }
28
```

SQL

Utilisation d'un DBL ainsi que des procédures stockées



Appel de la procédure stockée « VerificationDate(Param) »

```
//On vérifie les rdv ou la date est égale au jour selectionné(grâce à une procédure stockée), si il y en a ->
if (db.VerificationDate(JourSelec).Any())
{
    foreach (var elem in db.VerificationDate(JourSelec))
    {
        // Pour les AS, tous les RDV sont en bleu.
        if (idUsrConnecte == elem.IDUtilisateur || db.VerificationTypeUtilisateur(idUsrConnecte).Any())
        {
            monRDV = true;
        }
        else
        {
            monRDV = false;
        }
        string textBlockDebut = "textBlock_" + elem.heureDebut;
        string textBlockFin = "textBlock_" + elem.heureFin;
        string textBlockIntervale = textBlockDebut;
        int heureIntervale = Convert.ToInt32(elem.heureDebut); // 1200

        // Compte le nombre de textblock reprénsentant les plages horaire
        AfficherHorairePris(FC.Can_RDVTextBlock.Children.Count, textBlockDebut, textBlockFin, textBlockIntervale, elem.heureDebut
        premierPassage = false; // -> S'il y a plusieurs RDV il ne mettra pas celui qui a été noté en bleu
    }
}
```

Résumé

- Travail effectué en binôme.

- Phase d’analyse du problème au préalable, ensuite phase de codage puis la phase de débug.

- Nous avons rencontré une assistante sociale pour mieux connaître ses besoins.

- Travail présenté oralement devant un jury, le tout en anglais.

- Délais imparti : 2 mois.

- Note finale : 17/20