Documentation Technique

Table des matières

I. Introduction

Le laboratoire **Galaxy Swiss Bourdin (GSB)** est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen **Swiss Bourdin** (travaillant sur des médicaments plus conventionnels), lui-même déjà union de trois petits laboratoires.

Domaine d'étude

L'entreprise souhaite porter une attention nouvelle à sa force commerciale dans un double objectif qui est d'obtenir une vision plus régulière et efficace de l'activité menée sur le terrain auprès des praticiens.

Les déplacements et actions de terrain menées par les visiteurs engendrent des frais qui doivent être pris en charge par la comptabilité. On cherche à agir au plus juste de manière à limiter les excès sans pour autant diminuer les frais de représentation qui font partie de l'image de marque d'un laboratoire. Chez Galaxy, le principe d'engagement des frais est celui de la carte bancaire au nom de l'entreprise. Chez Swiss-Bourdin, une gestion forfaitaire des principaux frais permet de limiter les justificatifs. Pour tout le reste, le remboursement est fait après retour des pièces justificatives.

Une gestion unique de ces frais et remboursement pour l'ensemble de la flotte visite est souhaitée.

Les visiteurs récupèrent une information directe sur le terrain. Ceci concerne aussi bien le niveau de la confiance qu'inspire le laboratoire que la lisibilité des notices d'utilisation des médicaments ou encore les éventuels problèmes rencontrés lors de leur utilisation, etc. Ces informations ne sont actuellement pas systématiquement remontées au siège, ou elles le sont dans des délais jugés trop longs. Le service rédaction qui produit les notices souhaite avoir des remontées plus régulières et directes. Ceci permettra également au service labo-recherche d'engager des évaluations complémentaires.

L'application permet aux visiteurs médicaux lors de leurs déplacements et à tout moment de saisir et modifier leurs fiches de frais au forfait et hors forfait afin d'être remboursés.

L'objet de ce document est de définir les spécifications fonctionnelles détaillées de l'application GSB.

Les spécifications fonctionnelles détaillées ont pour but de décrire précisément l'activité et toutes les fonctionnalités prévues lors de la phase de conception sont précisées dans ce document en indiquant l'implémentation de ces fonctionnalités dans l'application.

II. CAHIER DES CHARGES

Définition de l'objet

Le suivi des frais est actuellement géré de plusieurs façons selon le laboratoire d'origine des visiteurs. On souhaite uniformiser cette gestion L'application doit permettre d'enregistrer tout frais engagé pour l'activité directe (déplacement, restauration et hébergement) et de présenter un suivi daté des opérations menées par le service comptable (réception des pièces, validation de la demande de remboursement, mise en paiement, remboursement effectué).

Forme de l'objet

L'application Desktop destinée aux visiteurs, délégués et responsables de secteur sera en ligne, accessible depuis un ordinateur. La partie utilisée par les services comptables sera aussi sous forme d'une interface Desktop.

Accessibilité/Sécurité

L'environnement doit être accessible aux seuls acteurs de l'entreprise. Une authentification préalable sera nécessaire pour l'accès au contenu.

Architecture

L'application respectera une architecture précise à définir.

Ergonomie

Les pages fournies ont été définies suite à une consultation. Des améliorations ou variations peuvent être proposées.

Codage

Le document "GsbWebTechnique " présente des règles de bonnes pratiques de développement utilisées par le service informatique de GSB pour encadrer le développement d'applications en PHP et en faciliter la maintenance. Les éléments à fournir devront respecter le nommage des fichiers, variables et paramètres, ainsi que les codes couleurs et la disposition des éléments déjà fournis.

Environnement

L'utilisation de bibliothèques, API ou frameworks est à l'appréciation du prestataire.

Modules

L'application présente deux modules : • enregistrement et suivi par les visiteurs • enregistrement des opérations par les comptables

Documentation

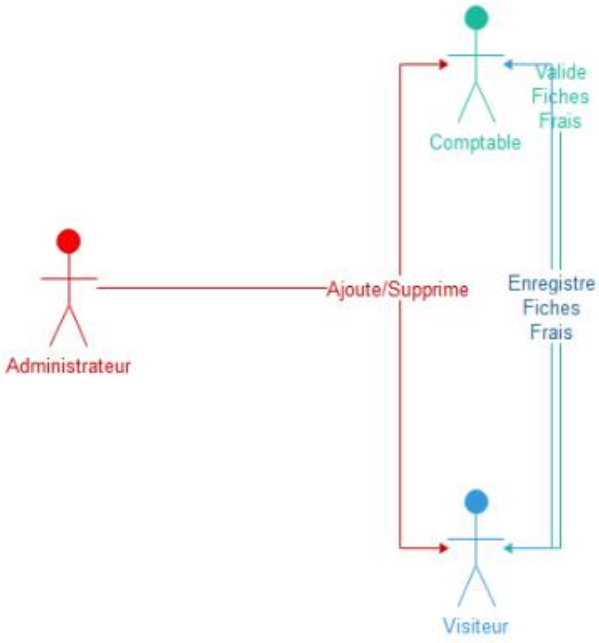
La documentation devra présenter l'arborescence des pages pour chaque module, le descriptif des éléments, classes et bibliothèques utilisées, la liste des frameworks ou bibliothèques externes utilisés

Responsabilités

Le commanditaire fournira à la demande toute information sur le contexte nécessaire à la production de l'application ainsi qu'une documentation et des sources exploitables pour la phase de test : base de données exemple, modélisation... Le prestataire est à l'initiative de toute proposition technique complémentaire. Le prestataire fournira un système opérationnel, une documentation technique permettant un transfert de compétence et un mode opératoire propre à chaque module.

III. DIAGRAMME DES ACTEURS DE L'APPLICATION

Le diagramme ci-dessous décrit le schéma des acteurs de l'application de gestion des frais.



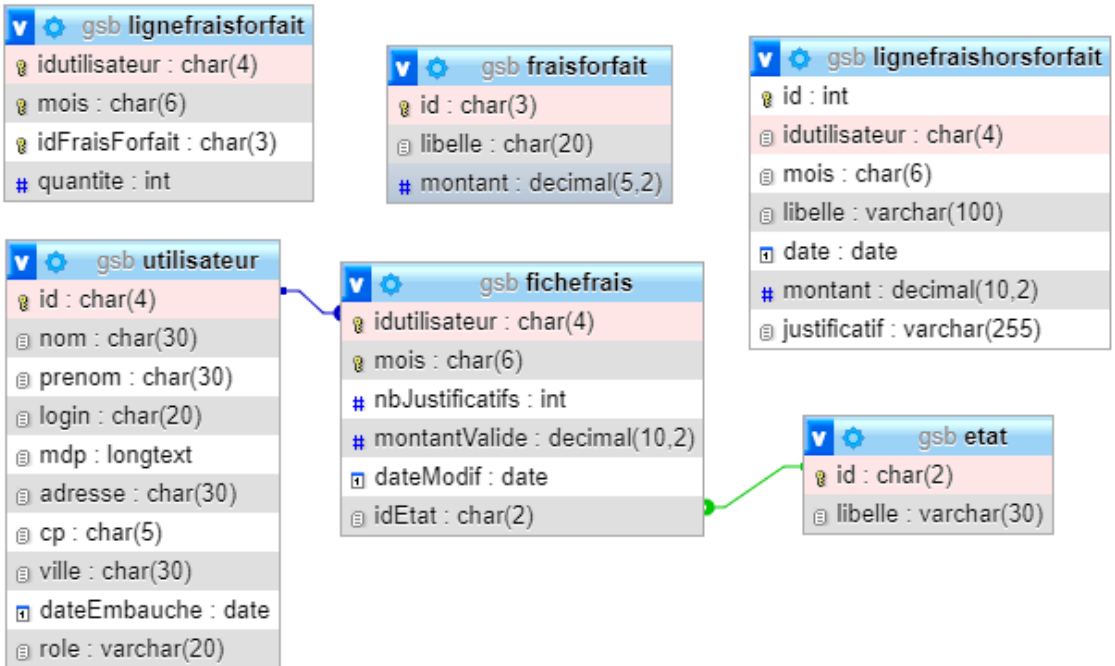
Les responsabilités de chacun des acteurs de l'application sont décrites ci-dessous :

- Le profil comptable permet de valider, de modifier et de refuser des fiches de frais

Le profil de comptable permet d'ajouter, de modifier et de supprimer des utilisateurs

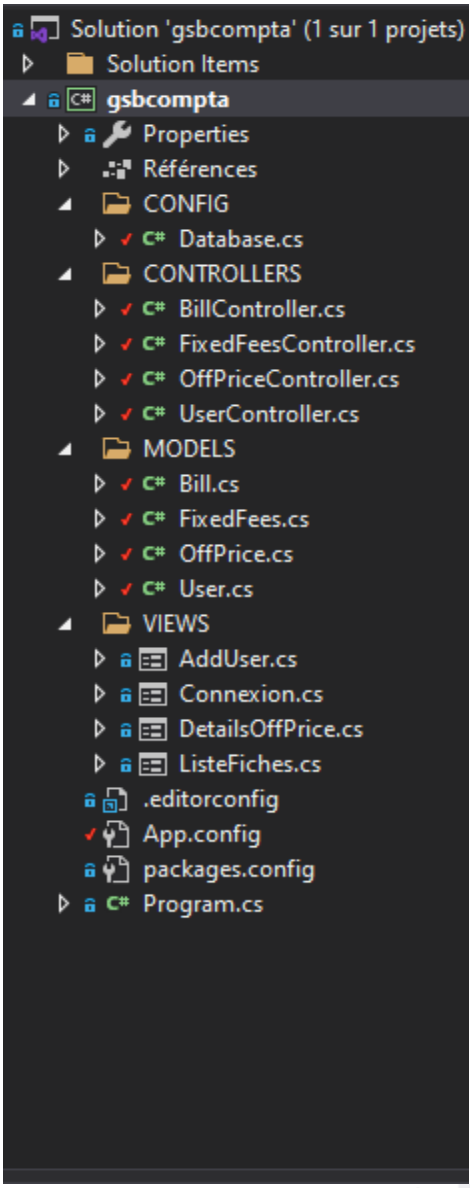
Modèle logique de données de l'outil

Le diagramme ci-dessous décrit le modèle de données de l'application de gestion de frais. Les informations manipulées dans l'outil n'impactent pas le modèle de données.



IV. ARBORESCENCE

L'application web a été réalisée avec le **Framework React.js** qui permet de créer des applications web et mobile en disposant d'une multitude de bibliothèque avec une communauté en pleine expansion :



Gsbcompta(Nom de notre application Desktop)

Properties: Ensemble des propriétés et des ressources ajoutées à notre projet.

Références : Une référence est essentiellement une entrée dans un fichier projet qui contient les informations dont Visual Studio a besoin pour localiser le composant ou le service.

Config : Contient le fichier nécessaire a la connexion a la base de données

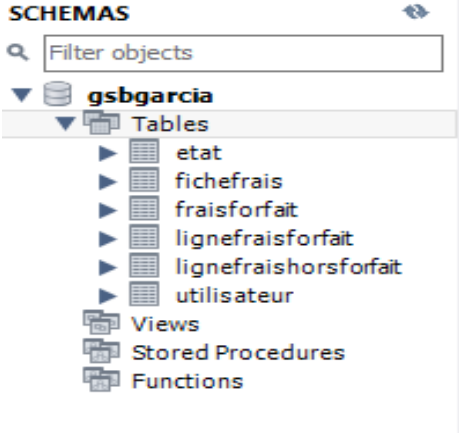
Controllers : Contient les controllers de nos fiches de frais et des utilisateurs afin d'exécuter les requêtes et de les renvoyer

Models :Permet de définir des classes qui seront réutilisés dans notre projet dans la Views et qui sera assigné une action grâce au controller

Views : Contient l'interface graphique de nos pages et l'assemblage de notre modèle MVC avec une assignation aux boutons et formulaires

Package.config :Stockage des dépendances du projet et de leurs versions

Program.cs : Permet le lancement de l'application avec la méthode static void Main()



La base de données est gérée sur MySQLWorkbench Avec une base de données stockée sur un serveur web. Elle contient 6 tables qui ont liées. Ex : Chaque fiche de frais et lié à un visiteur. Une fiche a elle-même une ou plusieurs lignes de frais forfait ou hors-forfait. La table utilisateur permet de définir les comptes d'accès à notre application web

V. LES DIFFERENTES FONCTIONNALITES

Authentification de l'utilisateur

Avant de pouvoir visualiser ses frais, le visiteur doit d'abord s'authentifier à l'aide de son login et de son mot de passe dans le formulaire de connexion. Lorsqu'un utilisateur est authentifié, son nom et son prénom apparaissent dans l'entête.

Lors de la demande de connexion (Quand le bouton "se connecter" a été cliqué avec les informations remplies au préalable)

```
1 référence
private void buttonLogin_Click(object sender, EventArgs e)
{
    String login = this.textBoxLogin.Text;
    String password = this.textBoxPassword.Text;
    Boolean logged = false;


    foreach(User u in uList)
    {
        if(u.Login == login && u.checkPassword(password))
        {
            logged = true;
        }
    }
    if(logged)
    {
        form = new ListeFiches();
        form.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Invalid Login or Password");
    }
```

Le bouton buttonLogin_Click déclenche l'action se connecter

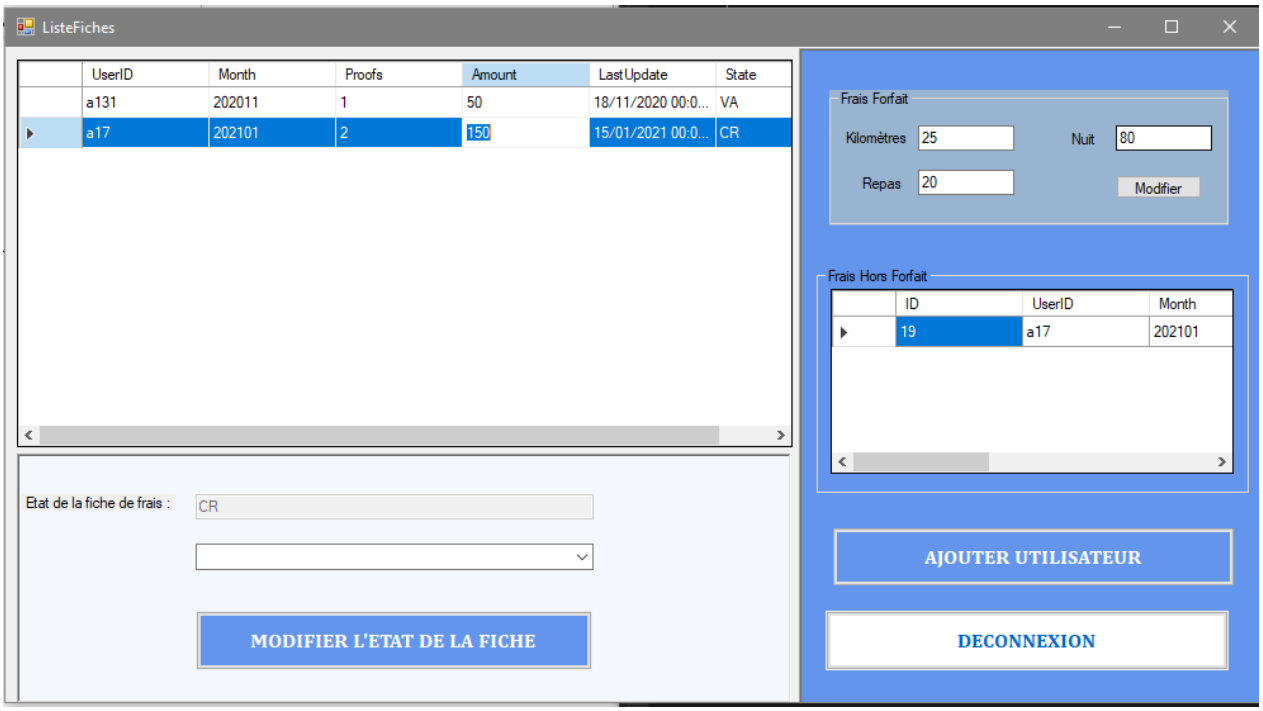
Les valeurs lues dans login et password sont récupérées et sont recherchées dans la base de données garce a UList

Si les informations ne sont pas conformes a celles existante au moment de a la connexion, un message d'erreur est renvoyé a la personne se connectant.

User dans le MODELS permet de vérifier si le password est bon grâce a sa methode checkPassword

Gestion de fiches de frais

Après s'être connecté, notre utilisateur peut modifier ses fiches de frais existantes ou en ajouter/supprimer



L'interface se présente ainsi et permet à l'utilisateur de renseigner les informations notamment les nuits/repas et le kilométrage pour se faire rembourser ses frais dans le forfait et appliquer le changement avec modifier

Les fiches de frais seront enregistrées dans la Base de données. Le comptable n'aura plus qu'a opérer pour gérer et rembourser ses fiches de frais avec également la possibilité d'acquisition de fichier envoyer par le praticien

```
2 références
public BillController()
{
    Db = new Database();
}

2 références
public List<Bill> getBills()
{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand();

        cmd.CommandText = "SELECT * FROM fichefrais";
        MySqlDataReader reader = cmd.ExecuteReader();
        Bills = new List<Bill>();

        while (reader.Read())
        {
            Bill b = new Bill(reader["idutilisateur"].ToString(), reader["mois"].ToString(), Int32.Parse(reader["nbJusti
            Bills.Add(b);
        }
        Db.Mysql.Close();
        return Bills;

    }
    catch (Exception e)
    {
        Console.WriteLine(e);
        return Bills;
    }
}
1 référence
public void updateStateBill(string userid, string month, string state)

{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand();

        cmd.CommandText = "UPDATE fichefrais SET idEtat = ('" + state + "' ) WHERE idutilisateur ='" + userid + "' AND m
```

ExecuteReader() est la méthode qui permet de récupérer toutes les fiches de frais

Methode qui est ensuite appelée dans Listesfiches pour récupérer nos données dans la base de données sous le nom de variables bList

Les Données qui sont inscrites par l'utilisateur sont stockées dans le MODELS sous la forme de variables qui récupèrent toutes les valeurs

Frais Forfaits

Dans ListFixedFees , on peut observer la requête de notre commande cmd.ExecuteReader() ; qui récupère les frais forfait de la fiche en fonction du nom d'utilisateur et du mois

```
1 référence
public List<FixedFees> getFixedFees(string userid, string month)
{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand();

        cmd.CommandText = "SELECT * FROM lignefraisforfait WHERE idutilisateur
        MySqlDataReader reader = cmd.ExecuteReader();
        FixedFees = new List<FixedFees>();

        while (reader.Read())
        {
            FixedFees ff = new FixedFees(reader["idutilisateur"].ToString(), r
            FixedFees.Add(ff);
        }
        Db.Mysql.Close();
        return FixedFees;

    }
    catch (Exception e)
    {
        Console.WriteLine("ERROR: " + e );
        return FixedFees;
    }
}

3 références
public void updateFixedFees(string userid, string month, string quantity, stri
{
    try
```

Ainsi , dans ListesFiches , lorsqu'un comptable selectionne une fiche , la méthode GetFixedFees est appelée pour désigner les frais forfaits aux champs correspondants. Cette méthode est appelée dans FixedFeesController qui envoie la requête correspondante a la quantité , l'utilisateur , le mois et les frais hors forfaits

```
1 référence
private void majFF()
{

    foreach (DataGridViewRow row in dataGridViewBills.SelectedRows)
    {
        userID = row.Cells[0].Value.ToString();
        month = row.Cells[1].Value.ToString();

        ffController = new FixedFeesController();
        ffList = ffController.getFixedFees(row.Cells[0].Value.ToString(),
            row.Cells[1].Value.ToString());
        this.textBoxFixedFeesKM.Text = ffList[0].Quantity.ToString();
        this.textBoxFixedFeesNUI.Text = ffList[1].Quantity.ToString();
        this.textBoxFixedFeesREP.Text = ffList[2].Quantity.ToString();
        opList = opController.getOffPrices(userID, month);
        this.dataGridViewOffPrice.DataSource = opList;

        txtBoxEtat.Text = row.Cells[5].Value.ToString();
    }
}

1 référence
private void dataGridViewOffPrice_SelectionChanged(object sender, EventArgs e)
{
    foreach (DataGridViewRow row in dataGridViewOffPrice.SelectedRows)
    {
```

Frais Hors Forfaits

Dans ListFiches , lorsque le comptable clique sur le bouton modifier , on utilise la méthode updateFixedFees pour mettre a jour les valeurs saisies dans la base de données

```
1 référence
private void buttonFixedFeesUpdate_Click(object sender, EventArgs
{
    String KM = this.textBoxFixedFeesKM.Text;
    String NUI = this.textBoxFixedFeesNUI.Text;
    String REP = this.textBoxFixedFeesREP.Text;
    ffController.updateFixedFees(userID, month, KM, "KM");
    ffController.updateFixedFees(userID, month, NUI, "NUI");
    ffController.updateFixedFees(userID, month, REP, "REP");
}
```

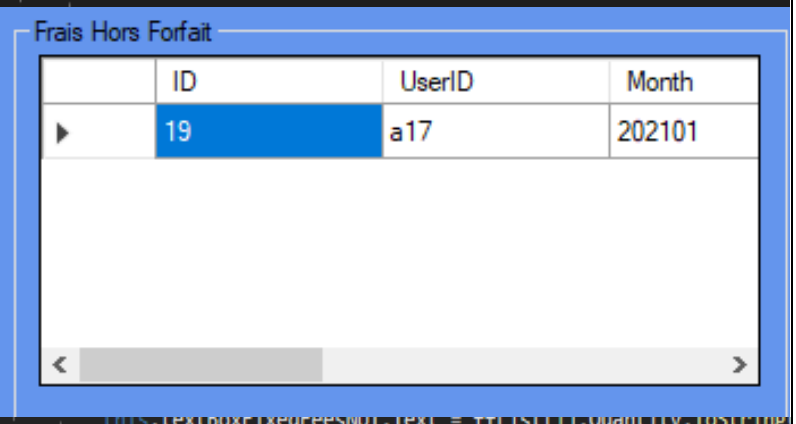
Gestion de fiches de frais

Dans FixedFeesController , on assigne la requête correspondante a la méthode pour pouvoir mettre a jour les données dans la base de données

```
3 références
public void updateFixedFees(string userid, string month, s
{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand();

        cmd.CommandText = "UPDATE lignefraisforfait SET q
        cmd.ExecuteNonQuery();

        Db.Mysql.Close();

    }
    catch (Exception e)
    {
        Console.WriteLine("ERROR: " + e);

    }
```

La création d'un nouvel utilisateur se passe tout d'abord dans la base de données .

L'administrateur ou comptable si les droits lui sont accordés pourra a sa guise ajouter des nouveaux utilisateurs en renseignant simplement les champs dont les plus importants (login,mdp)



Dans le tableau des frais hors forfait , lorsqu'on selectionne une ligne et qu'on veut la mettre a jour , on modifie la dataGridViewRow (soi les les colonnes d'une ligne) en changeant les valeurs par une nouvelles valeur

La méthode updateOffprice permet d'enregistrer les nouvelles valeurs grâce a sa requête SQL

```
        this.textBoxFixedFeesNUI.Text = ffList[1].Quantity.ToString
        this.textBoxFixedFeesREP.Text = ffList[2].Quantity.ToString
        opList = opController.getOffPrices(userID, month);
        this.dataGridViewOffPrice.DataSource = opList;

        txtBoxEtat.Text = row.Cells[5].Value.ToString();
    }
}

1 référence
private void dataGridViewOffPrice_SelectionChanged(object sender, Ev
{
    foreach (DataGridViewRow row in dataGridViewOffPrice.SelectedRo
    {
        DetailsOffPrice OPForm = new DetailsOffPrice(row.Cells[0].Va
        OPForm.Show();
        this.Close();

    }
}

1 référence
```

```
    }

    1 référence
    public void updateOffPrice(string id, string name, str
    {
        try
        {
            Db.Mysql.Open();

            MySqlCommand cmd = Db.Mysql.CreateCommand();

            cmd.CommandText = "UPDATE lignefraishorsforfai
            cmd.ExecuteNonQuery();

            Db.Mysql.Close();

        }
        catch (Exception e)
        {
            Console.WriteLine("ERROR: " + e);

        }

    }
```

Statut Fiche De Frais

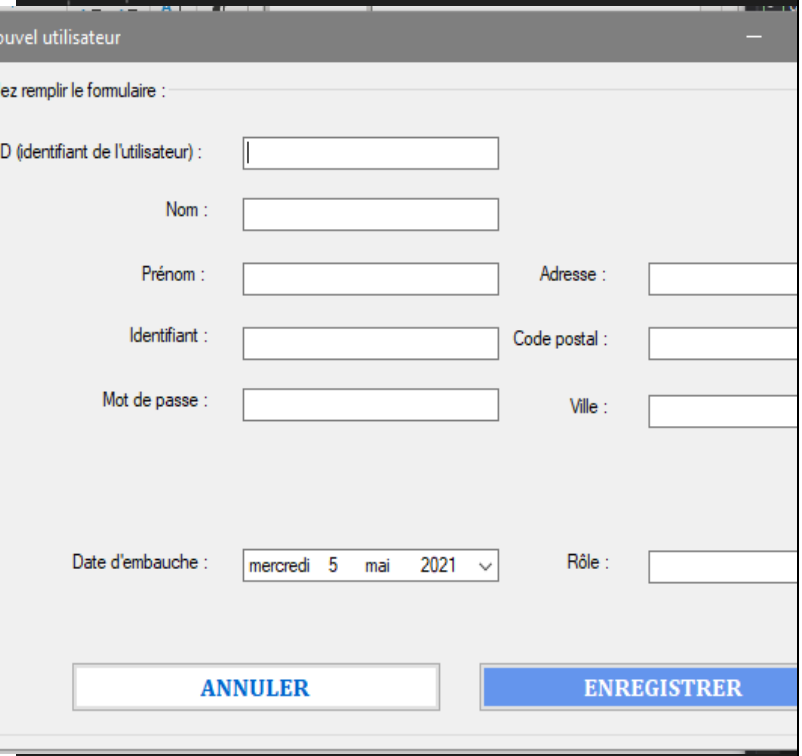
Le comptable modifie l'etat de la fiche de frais grâce au bouton modifier.

On appelle la methode UpdateStateBill qui met a jour l'etat de la fiche de frais grâce a sa requête sql dans le BillController

Getbills est tout de meme utiliser pour trouver les fiches de frais correspondantes a cet etat.

```
}
1 référence
public void updateStateBill(string userid, string m

{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand()

        cmd.CommandText = "UPDATE fichefrais SET id
        cmd.ExecuteNonQuery();

        Db.Mysql.Close();
    }
    catch (Exception e)
    {
        Console.WriteLine("ERROR: " + e);
```

Ajout Utilisateur



L'ajout d'un utilisateur s'opère lorsque le comptable appuie sur le bouton enregistrer qui créera un nouvel utilisateur grace a la méthode buttonAddUser_Click

Le controller recoit la quete SQL duquel le MODELS a enregistrer les données

```
}

1 référence
private void buttonAddUser_Click(object sender, EventArgs e)
{
    form = new AddUser();
    form.Show();
    this.Hide();
}

1 référence
public void addUsers(string id, string name, string firstn
{
    try
    {
        Db.Mysql.Open();

        MySqlCommand cmd = Db.Mysql.CreateCommand();

        cmd.CommandText = "INSERT INTO utilisateur(id, nom

        cmd.ExecuteNonQuery();

        Db.Mysql.Close();

    }
    catch (Exception e)
    {
        Console.WriteLine("ERROR: " + e);
```

```
0 références
public void AddUser(string id, string name, stri
{
    this.Id = id;
    this.Name = name;
    this.Firstname = firstname;
    this.Login = login;
    this.Password = password;
    this.Address = address;
    this.Zip = zip;
    this.City = city;
    this.DateEmployment = dateEmployment;
    this.Role = role;
}
```

```
1 référence
private void btnDisconnect_Click(object sender, EventArgs e)
{
    form1 = new Form1();
    form1.Show();
    this.Hide();

}
```

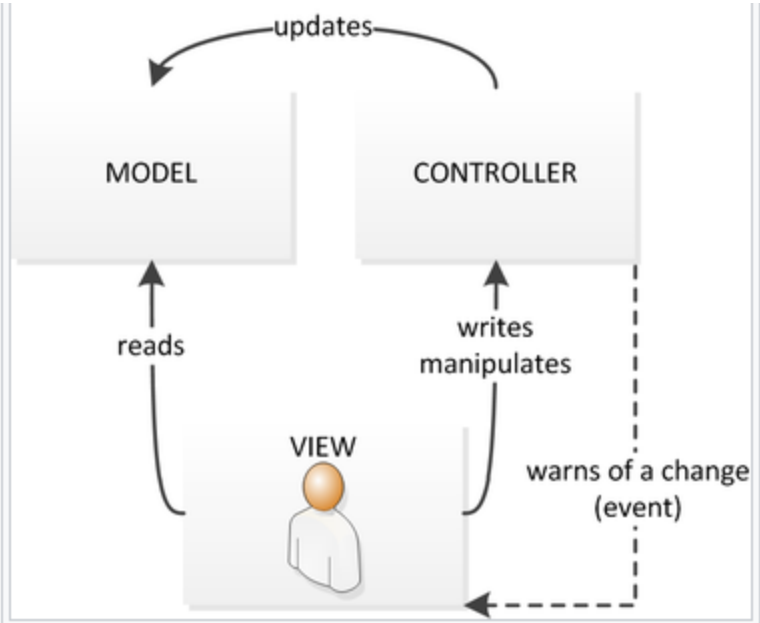
Deconnexion

Ce bouton permet de se déconnecter Et Affiche de nouveau la page de connexion lorsque le comptable se deconnecte

```
namespace gsbcompta
{
    9 références
    class Database
    {
        25 références
        public MySqlConnection Mysql { get; set; }

        4 références
        public Database()
        {
            this.connect();
        }

        1 référence
        private void connect()
        {
            string connectionString = "SERVER=127.0.0.1; DATABASE=gsb;
            this.Mysql = new MySqlConnection(connectionString);
        }
    }
```

Base De Données

Voici la base de données qui assure la connexion a notre base de données en localhost . D'où la nécessité d'utiliser Docker pour y avoir accès



Interactions entre le modèle, la vue et le contrôleur.

Pour bien comprendre ,le modèle proposé par notre projet est un modèle MVC qui permet trois modules avec trois responsabilités différentes : les modèles, les vues et les contrôleurs.

- Un modèle (Model) contient les données à afficher.
- Une vue (View) contient la présentation de l'interface graphique.
- Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.