



RAPPORT AP1_GSB



SOMMAIRE

- I. Contexte GSB
- II. Fonctionnalité présente
- III. BDD
- IV. Application
- V. Conclusion

I. Contexte GSB

Galaxy Swiss Bourdin (GSB) a été créé en 2009 par la fusion de Galaxy, qui se spécialise dans les maladies virales (SIDA, hépatites), et Swiss Bourdin, qui se concentre sur des médicaments plus traditionnels. Ce regroupement a créé une entreprise majeure dans le secteur pharmaceutique, avec un siège administratif à Paris et un siège social à Philadelphie, aux États-Unis. Le domaine pharmaceutique, caractérisé par de nombreuses fusions, génère de grandes rentrées de fonds, mais est critiqué pour ses pratiques de visite médicale considérées comme désinformées. À la suite de la fusion, GSB a mis en place une restructuration et des économies d'échelle pour optimiser ses activités, avec 480 visiteurs médicaux en France et 60 dans les DOM-TOM, répartis dans 6 secteurs géographiques.

La centralisation de l'infrastructure informatique de GSB à Paris inclut des services administratifs et des salles de serveurs sécurisées. Les informations, perçues comme stratégiques, sont répliquées chaque jour aux États-Unis. La direction des systèmes d'information occupe une position stratégique en intégrant des outils de recherche, de production et de communication avancés, tandis que la partie commerciale a été moins automatisée. Le réseau est divisé en VLAN afin de garantir la sécurité et la fluidité du trafic, avec des règles d'accès rigoureuses pour chaque VLAN.

GSB désire consolider sa capacité commerciale afin d'obtenir une meilleure compréhension de l'activité sur le terrain et de redonner confiance aux équipes après les fusions. Les patients, qui jouent un rôle essentiel dans l'information des prescripteurs sur les produits du laboratoire, ont un impact indirect sur les prescriptions médicales. Cette structure partagée par les délégués médicaux, qui découle de la politique de Galaxy, a pour objectif de standardiser et de moderniser les méthodes de visite médicale.

II. Fonctionnalité présente

L'application GSB développée intègre plusieurs fonctionnalités essentielles à la gestion des fiches de frais. Chaque fonction a été pensée pour répondre aux besoins métier tout en assurant une navigation intuitive et une sécurité renforcée.

Authentification des utilisateurs

- Vérification des identifiants via la base de données MariaDB.
- Redirection automatique vers l'interface correspondant au rôle de l'utilisateur :
 - Visiteur médical
 - Comptable
 - Administrateur (si présent)
- Interface simple de connexion (WinForms) avec retour d'erreur en cas d'identifiants invalides.

Gestion des fiches de frais

- Création automatique de la fiche de frais mensuelle lors de l'ajout du premier frais.
- Règle métier respectée : période de saisie autorisée du 11 du mois au 10 du mois suivant.
- Consultation des frais saisis, mois par mois.

Ajout de frais

- **Frais forfaitaires :**
 - Choix du type de frais (hébergement, repas, etc.) via liste déroulante.
 - Saisie de la quantité.
 - Calcul du montant basé sur le tarif unitaire stocké en base.
- **Frais hors forfait :**
 - Saisie libre du nom du frais.
 - Saisie du montant et de la date.
 - Vérification de validité des données (valeurs non nulles, montants positifs).

Consultation mensuelle

- Sélection du mois dans une ComboBox dynamique.
- Affichage des frais forfaitaires et hors forfaits dans des tableaux (DataGridView).

- Statut de validation visible pour chaque frais.

Validation comptable

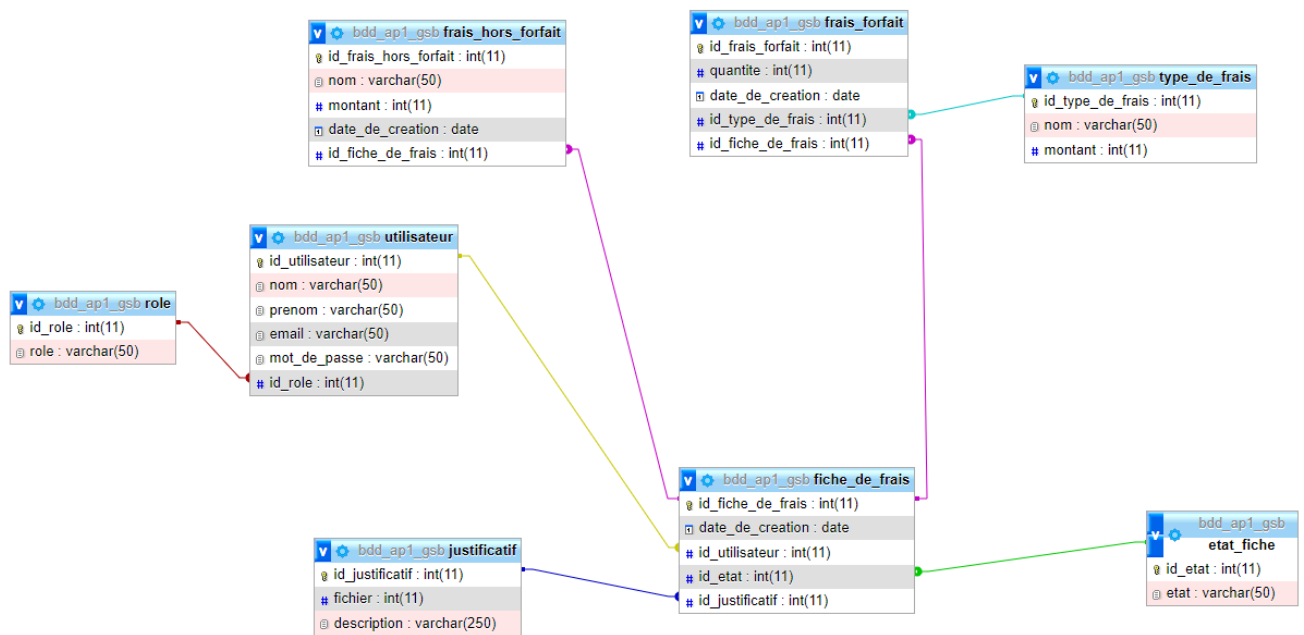
- Pour les utilisateurs avec rôle "comptable" :
 - Filtrage par utilisateur et mois.
 - Consultation des fiches avec possibilité de changer leur statut (VALIDER / REFUSER).
 - Mise à jour en base de l'état des fiches via boutons d'action.

Sécurité et fiabilité

- Utilisation de requêtes SQL paramétrées pour prévenir les injections.
- Gestion des erreurs et affichage de messages clairs à l'utilisateur.
- Architecture robuste et découplée entre logique métier, accès aux données et interface graphique.

III. BDD

Voici mon MPD :



Le modèle physique de données de l'application GSB a été conçu pour assurer la cohérence, la fiabilité et la clarté dans la gestion des fiches de frais. Il repose sur **neuf tables principales**, organisées selon les règles de la **troisième forme normale (3NF)**, garantissant une bonne séparation des responsabilités et une réduction de la redondance des données.

1. Table utilisateur

Contient les informations personnelles de chaque utilisateur.

- **Champs** : id_utilisateur, nom, prenom, email, mot_de_passe, id_role
- **Relation** : liée à la table role par id_role

2. Table role

Définit le rôle attribué à chaque utilisateur (ex : Visiteur, Comptable).

- **Champs** : id_role, role
- **Relation** : 1 rôle → plusieurs utilisateurs

3. Table fiche_de_frais

Représente une fiche mensuelle créée automatiquement pour chaque utilisateur.

- **Champs** : id_fiche_de_frais, date_de_creation, id_utilisateur, id_etat, id_justificatif
- **Relations** :
 - Avec utilisateur (auteur de la fiche)
 - Avec etat_fiche (statut de la fiche)
 - Avec justificatif (pièce jointe liée)

4. Table etat_fiche

Liste les états possibles d'une fiche de frais (En attente, Validée, Refusée, etc.).

- **Champs** : id_etat, etat
- **Relation** : 1 état → plusieurs fiches de frais

5. Table justificatif

Contient les fichiers associés à une fiche (nom et description).

- **Champs** : id_justificatif, fichier, description

6. Table frais_forfait

Stocke les frais standards (forfaitaires) saisis par l'utilisateur.

- **Champs** : id_frais_forfait, quantite, date_de_creation, id_type_de_frais, id_fiche_de_frais
- **Relations** :
 - Avec fiche_de_frais
 - Avec type_de_frais (nature du frais)

7. Table type_de_frais

Décrit chaque type de frais forfaitaire disponible.

- **Champs** : id_type_de_frais, nom, montant

8. Table frais_hors_forfait

Enregistre les frais non standards (libres).

- **Champs** : id_frais_hors_forfait, nom, montant, date_de_creation, id_fiche_de_frais
- **Relation** : liée à fiche_de_frais

Synthèse des relations clés

- Un **utilisateur** peut avoir plusieurs **fiches de frais**.
- Une **fiche de frais** peut contenir plusieurs **frais forfaitaires** ou **hors forfait**.
- Une **fiche de frais** a un seul **état** et peut être liée à un **justificatif**.
- Un **type de frais** peut être utilisé dans plusieurs enregistrements de frais forfaitaires.

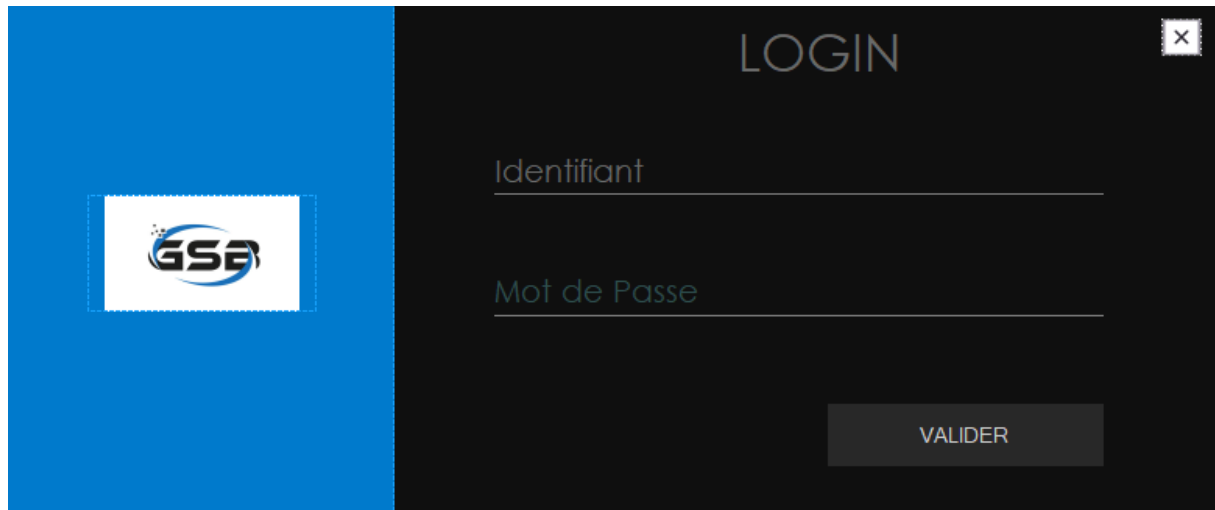
Conclusion

Ce MPD assure une organisation claire des données et une évolutivité du système. Il permet :

- Une gestion structurée des rôles et utilisateurs.
- Un suivi précis des frais par mois et par utilisateur.
- Un contrôle rigoureux des statuts et justificatifs associés aux fiches.

IV. Application

A. Page de connexion



Ceci est mon windows forms de ma page de connexion.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using MySql.Data.MySqlClient;
11
12 namespace API_GSB
13 {
14     4 références
15     public partial class login : Form
16     {
17         string mysqlCon = "server=127.0.0.1;uid=root;pwd=root;database=bdd_ap1_gsb";
18
19         1 référence
20         private void text_id_Enter(object sender, EventArgs e)
21         {
22             if (text_id.Text == "Identifiant")
23             {
24                 text_id.Text = "";
25                 text_id.ForeColor = Color.LightGray;
26             }
27         }
28     }
29 }
```

```

28 1 référence
29 private void text_mdp_Enter(object sender, EventArgs e)
30 {
31     if (text_mdp.Text == "Mot de Passe")
32     {
33         text_mdp.Text = "";
34         text_mdp.ForeColor = Color.LightGray;
35         text_mdp.UseSystemPasswordChar = true;
36     }
37
38 1 référence
39 private void text_mdp_Leave(object sender, EventArgs e)
40 {
41     if (text_mdp.Text == "")
42     {
43         text_mdp.Text = "Mot de Passe";
44         text_mdp.ForeColor = Color.DimGray;
45         text_mdp.UseSystemPasswordChar = false;
46     }
47
48 1 référence
49 private void pictureBox1_Click(object sender, EventArgs e)
50 {
51     Application.Exit();
52 }
53
54 private DataBase db = new DataBase();
55

```

```

56 1 référence
57 public login()
58 {
59     InitializeComponent();
60 }
61
62 1 référence
63 private void button_valider_Click(object sender, EventArgs e)
64 {
65     string email, user_password;
66     email = text_id.Text;
67     user_password = text_mdp.Text;
68     using (SqlConnection conn = db.GetConnection())
69     {
70         try
71         {
72             conn.Open();
73             MySqlCommand cmd = new MySqlCommand("SELECT utilisateur_id_utilisateur FROM 'utilisateur' WHERE mot_de_passe = '' + user_password + '' AND email = '' + email + '';", conn);
74             int dataId = Convert.ToInt32(cmd.ExecuteScalar());
75             if (dataId == 0)
76             {
77                 MessageBox.Show("L'email ou le mot de passe saisi n'est pas correct");
78                 conn.Close();
79                 return;
80             }
81             MySqlCommand command = new MySqlCommand("SELECT role FROM 'role' INNER JOIN utilisateur ON utilisateur_id_role = role.id_role WHERE utilisateur_id_utilisateur = '' + dataId + '';", conn);
82             string role = Convert.ToString(command.ExecuteScalar());
83             conn.Close();
84             Redirection(role, dataId);
85         }
86         catch
87         {
88             MessageBox.Show("Il y a eu un problème avec la base de données, veuillez recommencer");
89             conn.Close();
90         }
91     }
92 }

```

```

92 1 référence
93 private void Redirection(string role, int dataId)
94 {
95     Form newForm = null;
96
97     switch (role)
98     {
99         case "Visiteur":
100             newForm = new visiteur(dataId);
101             break;
102
103         case "Comptable":
104             newForm = new comptable(dataId);
105             break;
106
107         case "Administrateur":
108             newForm = new administrateur(dataId);
109             break;
110
111         default:
112             MessageBox.Show("Rôle non reconnu.");
113             return;
114     }
115
116     this.Hide();
117     newForm.ShowDialog();
118     this.Show();
119 }

```

Mon code représente une application Windows Forms écrite en C# qui permet aux utilisateurs de se connecter.

1. Imports et Déclarations :

- Les bibliothèques nécessaires sont importées, y compris celles pour la manipulation des formulaires Windows et la connexion à MySQL.

2. Classe login :

- Hérite de Form pour créer une fenêtre de connexion.

3. Variables :

- `mysqlCon` : Chaîne de connexion à la base de données MySQL.

4. Événements de Contrôle :

- **`text_id_Enter` et `text_mdp_Enter`** : Gèrent l'événement lorsque les champs de texte (pour l'identifiant et le mot de passe) sont activés. Si le champ contient le texte par défaut ("Identifiant" ou "Mot de Passe"), ce texte est effacé et la couleur du texte change.
- **`text_mdp_Leave`** : Gère l'événement lorsque le champ de mot de passe perd le focus. Si le champ est vide, il remet le texte par défaut et change la couleur du texte.
- **`pictureBox1_Click`** : Quitte l'application lorsque l'utilisateur clique sur `pictureBox1`.

5. Initialisation du Formulaire :

- **`login`** : Constructeur de la classe qui initialise les composants du formulaire.

6. Connexion et Authentification :

- **`button_valider_Click`** : Gère l'événement du clic sur le bouton de validation. Il récupère l'email et le mot de passe entrés par l'utilisateur et tente de se connecter à la base de données pour vérifier les informations d'identification.
 - Ouvre une connexion à la base de données MySQL.
 - Exécute une requête pour vérifier si les informations de connexion sont correctes.
 - Si les informations sont correctes, une deuxième requête est exécutée pour obtenir le rôle de l'utilisateur.
 - Appelle la méthode Redirection pour rediriger l'utilisateur vers une nouvelle fenêtre en fonction de son rôle.

7. Redirection Basée sur le Rôle :

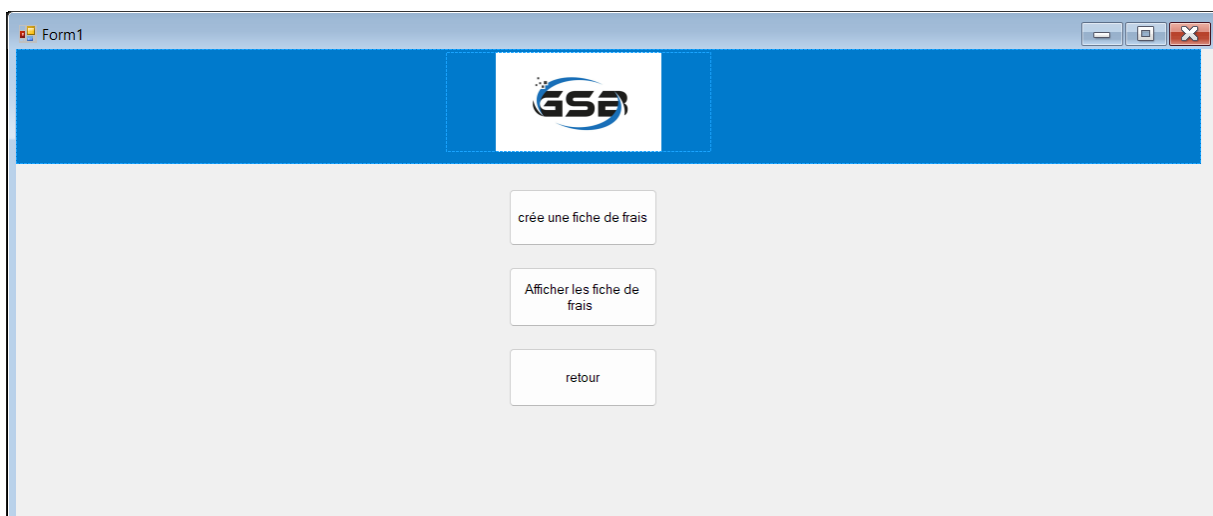
- **Redirection** : Méthode qui redirige l'utilisateur vers une fenêtre spécifique en fonction de son rôle (Visiteur, Comptable, Administrateur).
 - Crée une nouvelle instance de formulaire correspondant au rôle de l'utilisateur.
 - Cache le formulaire de connexion actuel et affiche le nouveau formulaire.

Ce code gère principalement l'authentification des utilisateurs et la redirection en fonction de leur rôle, avec une interface utilisateur simple pour la connexion. Quelques points à noter pour l'améliorer :

- **Sécurité** : Les requêtes SQL dans le code actuel sont vulnérables aux attaques par injection SQL. Utiliser des paramètres pour les requêtes SQL est fortement recommandé.
- **Gestion des Erreurs** : Le bloc catch est générique. Il pourrait être utile de capturer des exceptions spécifiques et de fournir plus de détails dans les messages d'erreur pour aider au débogage.

Mon code montre comment récupérer les informations de la base de données pour diriger chaque utilisateur vers le bon formulaire, leur permettant ainsi d'accéder à leur page personnelle et d'utiliser leurs fonctionnalités spécifiques.

B. Page utilisateur



C. Page ajoute frais



Le formulaire Ajouter_frais permet à l'utilisateur de choisir s'il souhaite ajouter un frais forfaitaire ou hors forfait. L'identifiant de l'utilisateur connecté est récupéré automatiquement à partir de la classe login et transmis aux formulaires d'ajout de frais correspondants. Deux boutons permettent d'ouvrir les formulaires Ajouter_frais_forfait et Ajouter_frais_hors_forfait, tandis qu'un troisième bouton permet de fermer la fenêtre.

D. Ajoute de frais forfait et hors forfait

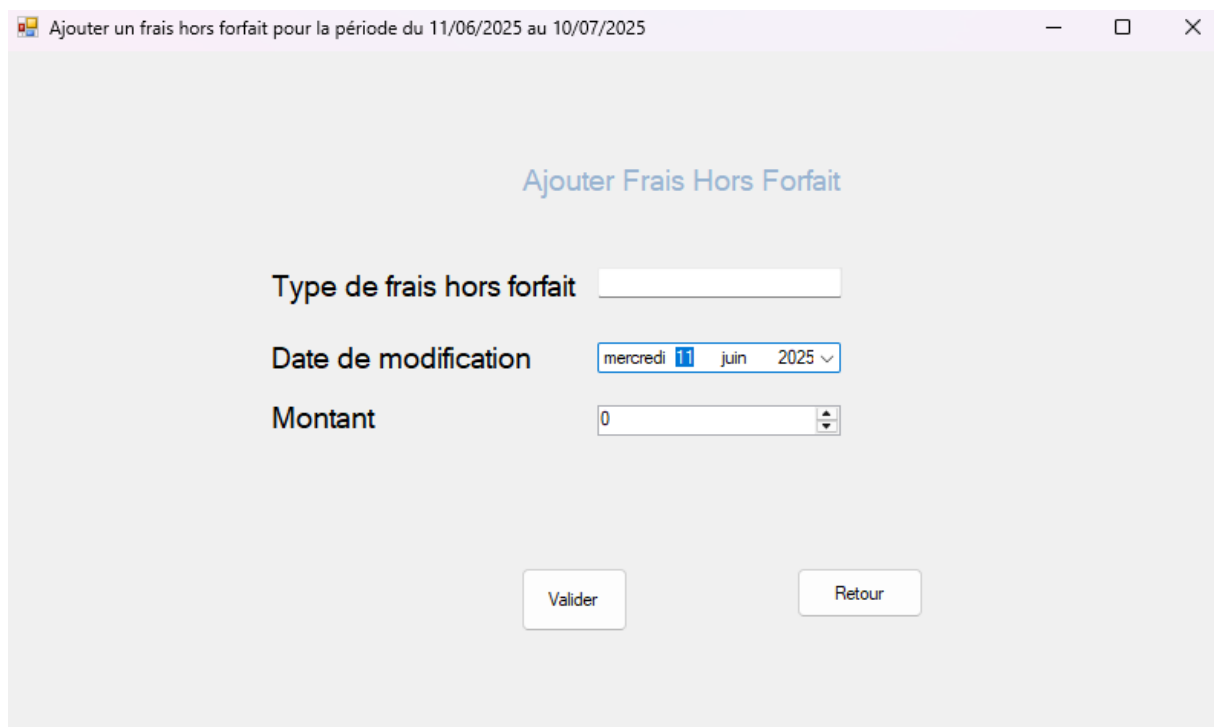
A screenshot of a web application window titled 'Ajouter un frais forfait pour la période du 11/06/2025 au 10/07/2025'. The window has a light purple header bar with the title and standard OS window controls. The main content area has a light gray background. At the top center, the text 'Ajouter Frais Forfait' is displayed in blue. Below this, there are three rows of form fields. The first row is labeled 'Type de frais forfait' and has a dropdown menu with 'Nuité' selected. The second row is labeled 'Date de modification' and has a date picker showing 'mercredi 11 juin 2025'. The third row is labeled 'Quantité' and has a numeric input field with '0' and a small up/down arrow icon. At the bottom of the form, there are two white rectangular buttons: 'Valider' on the left and 'Retour' on the right.

Ce formulaire permet à l'utilisateur connecté de saisir un **frais forfaitaire** (avec choix de type et quantité). À l'ouverture, la liste des types de frais est chargée dynamiquement depuis la table `type_de_frais` via une requête SQL.

Une période de saisie autorisée est automatiquement calculée, allant du **11 du mois en cours au 10 du mois suivant**, et appliquée au champ de date (DateTimePicker). Cette règle métier est contrôlée côté client dans l'interface.

Lors de la validation :

- Le formulaire vérifie les champs obligatoires.
- Il utilise une méthode `GetOrCreateFiche()` pour rechercher ou créer une fiche de frais (table `fiche_de_frais`) en fonction de la période.
- Les données du frais sont ensuite insérées dans la table `frais_forfait` via une requête SQL paramétrée, en sécurisant les entrées avec des `SqlParameter`.



The screenshot shows a web application window titled "Ajouter un frais hors forfait pour la période du 11/06/2025 au 10/07/2025". The main heading is "Ajouter Frais Hors Forfait". The form contains three input fields: "Type de frais hors forfait" (a text input), "Date de modification" (a date picker showing "mercredi 11 juin 2025"), and "Montant" (a numeric input with a spinner, showing "0"). At the bottom, there are two buttons: "Valider" and "Retour".

Ce formulaire permet la saisie d'un **frais hors forfait**, non prédéfini. L'utilisateur doit renseigner un **nom**, un **montant** et une **date**.

Comme pour les frais forfaitaires :

- La période de saisie est limitée du 11 au 10 du mois suivant.
- Le montant est validé côté client comme un nombre positif.

- Une fiche de frais mensuelle est automatiquement recherchée ou créée si elle n'existe pas déjà (GetOrCreateFicheDeFrais()).

L'insertion du frais est ensuite réalisée dans la table frais_hors_forfait via une requête SQL sécurisée avec des paramètres.

E. Consultation des fiche de frais

The screenshot shows a web application window titled 'fiche de frais'. At the top, there is a dropdown menu set to 'Juin'. Below it, the title 'Fiches de frais du 11 juin 2025 au 10 juillet 2025' is displayed. The main content area contains two data grids. The first grid, titled 'Frais Forfait', has columns: type_de_frais, quantite, montant_unitaire, date_de_creation, and statut_fiche. It lists five entries: 'Nuité' (2 units, 40), 'Nuité' (1 unit, 40), 'Nuité' (0 units, 40), 'Repas' (2 units, 25), and 'Kilométrique' (4 units, 1). The second grid, titled 'Frais Hors-Forfait', has columns: type_de_frais_hors, montant, date_de_creation, and statut_fiche. It lists three entries: 'cc' (6 units, 11/06/2025), 'salut' (100 units, 20/06/2025), and 'sdc' (100 units, 24/06/2025). A 'retour' button is located at the bottom right.

type_de_frais	quantite	montant_unitaire	date_de_creation	statut_fiche
Nuité	2	40	19/06/2025	VALIDER
Nuité	1	40	18/06/2025	VALIDER
Nuité	0	40	11/06/2025	VALIDER
Repas	2	25	11/06/2025	VALIDER
Kilométrique	4	1	02/07/2025	VALIDER

type_de_frais_hors	montant	date_de_creation	statut_fiche
cc	6	11/06/2025	VALIDER
salut	100	20/06/2025	VALIDER
sdc	100	24/06/2025	EN COURS

Ce formulaire permet d'afficher les frais forfaitaires et hors forfaits saisis par l'utilisateur, en fonction du mois sélectionné.

- La liste des mois est générée dynamiquement avec DateTimeFormat, et la sélection déclenche le rechargement des données.
- La période de recherche est définie selon une règle métier : du **11 du mois au 10 du mois suivant**.
- Deux méthodes (LoadFichesFraisForfait et LoadFichesFraisHorsForfait) exécutent des **requêtes SQL paramétrées** avec jointures pour charger les frais depuis la base MariaDB.
- Les résultats sont affichés dans deux DataGridView.
- Le contrôle des erreurs est géré par des blocs try/catch avec messages d'alerte.
- Toutes les requêtes utilisent des **paramètres sécurisés** pour éviter les injections SQL.

F. Comptable

Validation des fiches de frais

visiteur juin 2025 Rechercher

	type	quantite	montant_unitaire	statut
	Nuité	2	40	VALIDER
	Nuité	1	40	VALIDER
	Nuité	0	40	VALIDER
	Repas	2	25	VALIDER
	Kilométrique	4	1	VALIDER
*				

	type	montant	statut
	cc	6	VALIDER
	salut	100	VALIDER
	sdc	100	REFUSER
*			

Valider Refuser

Le formulaire comptable est destiné aux utilisateurs ayant un rôle de validation. Il permet de **consulter, filtrer et modifier** l'état des fiches de frais (forfaitaires et hors forfaits) soumises par les visiteurs médicaux.

Au chargement du formulaire (ComptableForm_Load) :

- La méthode ChargerListeVisiteurs() récupère tous les utilisateurs ayant un rôle `id_role = 3` (visiteurs) et les affiche dans une ComboBox.
- La méthode ChargerListeMois() génère dynamiquement les 12 derniers mois en format texte français (MMMM yyyy), affichés dans une autre ComboBox.

Lorsque l'utilisateur clique sur le bouton "**Rechercher**" :

- Le mois sélectionné est transformé en période exacte via `ObtenirPeriodeSelectionnee()` (du **11 du mois** au **10 du mois suivant**).
- Les méthodes `ChargerFraisForfait()` et `ChargerFraisHorsForfait()` exécutent chacune une requête SQL pour charger les frais correspondants :

- Requêtes avec INNER JOIN sur les tables fiche_de_frais, etat_fiche, et soit frais_forfait, soit frais_hors_forfait.
- Les résultats sont affichés dans deux DataGridView.
- Les identifiants internes (id_fiche_de_frais, id_frais_x) sont masqués à l'affichage.

Lorsque l'utilisateur sélectionne une ligne et clique sur "**Valider**" ou "**Refuser**" :

- La méthode ModifierStatutFicheSelectionnee() est appelée.
- Elle récupère l'id_fiche_de_frais de la ligne sélectionnée.
- Elle interroge la table etat_fiche pour obtenir l'id_etat correspondant au nouveau statut.
- Une requête UPDATE met à jour la fiche de frais avec ce nouveau statut.
- L'affichage est automatiquement rafraîchi après modification.

Ce formulaire assure une **gestion centralisée et sécurisée des validations comptables**, en respectant la logique métier des périodes et en offrant à l'utilisateur une interface claire pour valider ou refuser les frais saisis par les visiteurs.

G.Database

```
1 using MySql.Data.MySqlClient;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7 using System.Windows.Forms;
8
9 namespace AP1_GSB
10 {
11     14 références
12     public class DataBase
13     {
14         private string ConnectionString = "server=127.0.0.1;uid=root;pwd=root;database=bdd_ap1_gsb";
15
16         6 références
17         public MySqlConnection GetConnection()
18         {
19             try
20             {
21                 MySqlConnection connection = new MySqlConnection(ConnectionString);
22                 return connection;
23             }
24             catch (MySqlException ex)
25             {
26                 MessageBox.Show($"Erreur lors de la connexion à la base de données : {ex.Message}");
27                 return null;
28             }
29         }
30
31         2 références
32         public static string ConvertDateFormat(DateTime date)
33         {
34             string day = date.Day.ToString();
35             string month = date.Month.ToString();
36             string year = date.Year.ToString();
37             string returnValue = $"{year}-{month}-{day}";
38             return returnValue;
39         }
40
41         2 références
42         public static string DateFiche()
43         {
44             DateTime now = DateTime.Now;
45             if (now.Day < 11)
46             {
47                 now = now.AddMonths(-1);
48             }
49             string returnDate = $"{now.Month}-{now.Year}";
50             return returnDate;
51         }
52     }
53 }
54
55
56
57
```

Votre code représente une classe DataBase.

1. Imports et Déclarations :

- Les bibliothèques nécessaires sont importées, y compris celles pour la connexion à MySQL et la manipulation des formulaires Windows.

2. Classe DataBase :

- Déclare une chaîne de connexion (ConnectionString) pour se connecter à la base de données MySQL.

3. Méthode GetConnection :

- Crée et retourne une connexion MySQL à partir de la chaîne de connexion.
- En cas d'erreur lors de la création de la connexion, affiche un message d'erreur et retourne null.

4. Méthode ConvertDateFormat :

- Méthode statique qui prend une date (DateTime) et la convertit en une chaîne au format AAAA-MM-JJ.
- Sépare le jour, le mois et l'année, puis les combine dans le format souhaité.

5. Méthode DateFiche :

- Méthode statique qui détermine la période pour laquelle une fiche de frais est créée.
- Si le jour du mois actuel est inférieur à 11, recule d'un mois.
- Retourne une chaîne représentant le mois et l'année au format MM-AAAA.

En résumé, ce code fournit une classe utilitaire pour gérer les connexions à une base de données MySQL et pour manipuler les dates. Il inclut des méthodes pour établir une connexion à la base de données, convertir les dates en chaînes spécifiques et déterminer la période de création des fiches de frais.

V. Conclusion

Ce projet avait pour objectif de développer une application de gestion des fiches de frais destinée aux utilisateurs de l'entreprise GSB, avec une distinction claire entre les rôles des visiteurs médicaux et des comptables. Grâce à l'utilisation de **C# en WinForms** pour l'interface et **phpMyAdmin** pour la base de données, l'application permet de manière fluide :

- La **saisie contrôlée** des frais forfaitaires et hors forfaits,
- La **consultation mensuelle** des frais par l'utilisateur,
- La **validation ou le refus** des fiches de frais par un comptable.

L'ensemble des interfaces respecte les règles métiers imposées par l'entreprise, notamment la contrainte de période (du 11 au 10 du mois suivant), la création automatique des fiches mensuelles, et le suivi de l'état des frais.

Sur le plan technique, l'application repose sur une architecture robuste, avec :

- Des requêtes SQL sécurisées,
- Un découplage clair entre l'interface utilisateur et les accès à la base de données,
- Une bonne gestion des erreurs et des interactions utilisateurs.

Ce projet m'a permis de consolider mes compétences en **programmation orientée objet**, en **base de données relationnelle**, ainsi qu'en **développement d'interfaces graphiques**. Il démontre l'importance de combiner la logique métier, la rigueur technique et l'ergonomie dans la réalisation d'un outil professionnel fiable et fonctionnel.