Rapport Projet en C#

Laboratoire GSB

Kottoyev Zoubayr
Ourici Ismail

BTSSIO2

Sommaire

Introduction

Contexte du Projet : Ce rapport est rédigé dans le contexte du projet de développement d'une application logicielle en C# (C-sharp) nommée "Laboratoire GSP". L'application est conçue pour gérer des incidents, du matériel, des utilisateurs, des techniciens, et des responsables dans un environnement de laboratoire ou similaire. Le projet implique l'utilisation d'une base de données MySQL pour stocker et gérer ces informations.

Objectif du Rapport : L'objectif principal de ce rapport est de documenter de manière exhaustive le projet "laboratoire GSP". Il vise à présenter une vue d'ensemble du projet, à expliquer son architecture logicielle et à mettre en évidence le rôle crucial de la classe `Bd` dans la gestion de la base de données. Le rapport cherchera à expliquer comment les différentes classes du projet interagissent et à montrer comment les opérations de base de données sont effectuées, notamment l'ajout, la modification et la récupération de données. De plus, ce rapport peut également inclure des captures d'écran de l'interface utilisateur pour illustrer l'utilisation de l'application dans un contexte pratique.

Classes du Projet

Classe de Base de Données

Chaine de Connexion MySQL : La classe contient une chaîne de connexion MySQL (connStr) qui spécifie les détails de la connexion à la base de données, tels que l'adresse du serveur, le nom de la base de données, le nom d'utilisateur et le mot de passe.

Méthodes d'Ajout de Données :

- Ajouterunincident(Incident unIncident): Cette méthode ajoute un nouvel incident à la base de données. Elle prend en compte les informations sur l'incident, telles que l'ID de l'utilisateur, l'ID du matériel, la description, le statut et la date de déclaration.
- Ajouterunresponsable(Responsable unResponsable): Cette méthode ajoute un nouveau responsable à la base de données. Elle enregistre l'ID du responsable et son mot de passe.
- Ajouterunmateriel(Materiel unMateriel): Cette méthode ajoute un nouveau matériel à la base de données, en enregistrant l'ID du matériel, le nom et la description.
- Ajouterunutilisateur(Utilisateur unUtilisateur): Cette méthode ajoute un nouvel utilisateur à la base de données, avec des informations telles que l'ID de l'utilisateur, le nom, le prénom, la fonction, le niveau d'habilitation, le mot de passe et l'ID du responsable.
- Ajouteruntechnicien(Technicien unTechnicien): Cette méthode enregistre un nouveau technicien dans la base de données, avec des détails tels que l'ID du technicien, le nom, le prénom, le niveau d'intervention, le mot de passe, la formation, les compétences et l'ID du responsable.

Méthodes de Récupération de Données :

- getLesIncidents(): Cette méthode récupère la liste des incidents depuis la base de données. Elle parcourt les enregistrements de la table "incident" et crée des objets de type Incident pour chaque incident enregistré.
- getLesMateriel(): Cette méthode récupère la liste des matériels depuis la base de données. Elle parcourt les enregistrements de la table "materiel" et crée des objets de type Materiel pour chaque matériel enregistré.

- **getLesTechnicien()**: Cette méthode récupère la liste des techniciens depuis la base de données. Elle parcourt les enregistrements de la table "technicien" et crée des objets de type Technicien pour chaque technicien enregistré.
- **getLesUtilisateur()**: Cette méthode récupère la liste des utilisateurs depuis la base de données. Elle parcourt les enregistrements de la table "utilisateur" et crée des objets de type Utilisateur pour chaque utilisateur enregistré.

Méthodes de Vérification :

- **verifUtil(string id, string mdp)**: Cette méthode vérifie l'authentification d'un utilisateur en comparant l'ID de l'utilisateur et le mot de passe avec les enregistrements de la base de données.
- **verifTech(string id, string mdp)**: Cette méthode effectue une vérification similaire pour un technicien.
- **verifRes(string id, string mdp)**: Cette méthode vérifie l'authentification d'un responsable.

Méthodes de Mise à Jour et de Suppression :

- **SuppMateriel(int id)**: Cette méthode supprime un matériel de la base de données en fonction de son ID.
- **ModifStatutPris(int id)**: Cette méthode met à jour le statut d'un incident en le marquant comme "Pris".
- **ModifStatutTraite(int id)**: Cette méthode met à jour le statut d'un incident en le marquant comme "Traité".
- **ModifTech(int id)**: Cette méthode effectue une mise à jour sur un technicien.
- **SuppTech(int id)**: Cette méthode supprime un technicien de la base de données.
- **SuppUtilisateur(int id)**: Cette méthode supprime un utilisateur de la base de données.

Classes "Incident" et "Matière"

Classe "Incident" :

La classe "Incident" représente un incident dans votre application. Elle contient plusieurs attributs et deux constructeurs pour initialiser ces attributs. Voici une description plus détaillée :

- Attributs :
  - idIncident (int) : L'identifiant unique de l'incident.
  - description (string) : La description de l'incident.
  - statut (string) : Le statut de l'incident, qui peut être par exemple "déclaré", "pris" ou "traité".
  - dateDeclaration (string) : La date de déclaration de l'incident.
  - dateResolution (string) : La date de résolution de l'incident.
  - travailARealiser (string) : Une description du travail à réaliser pour résoudre l'incident.
  - idMateriel (string) : L'identifiant du matériel associé à l'incident.
  - idTechnicien (string) : L'identifiant du technicien chargé de l'incident.
  - idUtilisateur (string) : L'identifiant de l'utilisateur ayant déclaré l'incident.
- Constructeurs :
  - Le premier constructeur prend en compte tous les attributs de l'incident et permet de les initialiser lors de la création d'une instance de la classe.
  - Le deuxième constructeur simplifie la création d'un incident en fournissant seulement la description, la date de déclaration, l'ID du matériel et l'ID de l'utilisateur. Il initialise automatiquement le statut à "déclaré".

Classe "Materiel" :

La classe "Materiel" représente un matériel dans votre application. Elle contient trois attributs et un constructeur pour initialiser ces attributs. Voici une description plus détaillée :

- Attributs :
  - idMateriel (int) : L'identifiant unique du matériel.
  - nom (string) : Le nom du matériel.
  - description (string) : La description du matériel.
- Constructeur :

- Le constructeur permet d'initialiser les attributs de la classe "Materiel" lors de la création d'une instance en fournissant l'identifiant du matériel, son nom et sa

Classe "Utilisateur"

La classe "Utilisateur" représente un utilisateur dans votre application. Elle contient plusieurs attributs et deux constructeurs pour initialiser ces attributs. Voici une description plus détaillée :

- Attributs :

  - idUtilisateur (int) : L'identifiant unique de l'utilisateur.
  - nom (string) : Le nom de l'utilisateur.
  - prenom (string) : Le prénom de l'utilisateur.
  - fonction (string) : La fonction de l'utilisateur.
  - niveauHabilitation (string) : Le niveau d'habilitation de l'utilisateur.
  - motDePasse (string) : Le mot de passe de l'utilisateur.
  - idResponsable (int) : L'identifiant du responsable de l'utilisateur.

- Constructeurs :

  - Le premier constructeur prend en compte tous les attributs de l'utilisateur et permet de les initialiser lors de la création d'une instance de la classe. Cela permet de créer un utilisateur avec un identifiant, un nom, un prénom, une fonction, un niveau d'habilitation, un mot de passe et un identifiant de responsable.
  - Le deuxième constructeur simplifie la création d'un utilisateur en fournissant tous les attributs à l'exception de l'identifiant du responsable. Cela peut être utile lorsque l'identifiant du responsable n'est pas encore connu ou n'est pas applicable.

- Propriétés (Getters et Setters) :

  - Chaque attribut de la classe "Utilisateur" est associé à une propriété (getter et setter) qui permet d'accéder à la valeur de l'attribut ou de la modifier.

Classe "Technicien" et Classe "Responsable

Classe "Technicien" :

La classe "Technicien" représente un technicien dans votre application. Voici une description détaillée de cette classe :

- Attributs :
  - idTechnicien (int) : L'identifiant unique du technicien.
  - nom (string) : Le nom du technicien.
  - prenom (string) : Le prénom du technicien.
  - niveauIntervention (string) : Le niveau d'intervention du technicien.
  - motDePasse (string) : Le mot de passe du technicien.
  - formation (string) : La formation du technicien.
  - competences (string) : Les compétences du technicien.
  - idResponsable (int) : L'identifiant du responsable du technicien.
- Constructeur :
  - Le constructeur de la classe "Technicien" prend en compte tous les attributs de la classe et permet de les initialiser lors de la création d'une instance. Cela permet de créer un technicien avec un identifiant, un nom, un prénom, un niveau d'intervention, un mot de passe, une formation, des compétences et un identifiant de responsable.
- Propriétés (Getters et Setters) :
  - Chaque attribut de la classe "Technicien" est associé à une propriété (getter et setter) qui permet d'accéder à la valeur de l'attribut ou de la modifier.

Classe "Responsable" :

La classe "Responsable" représente un responsable dans votre application. Voici une description détaillée de cette classe :

- Attributs :
  - idResponsable (int) : L'identifiant unique du responsable.
  - motDePasse (string) : Le mot de passe du responsable.
- Constructeur :
  - Le constructeur de la classe "Responsable" prend en compte les attributs de la classe et permet de les initialiser lors de la création d'une instance. Cela permet de créer un responsable avec un identifiant et un mot de passe.
- Propriétés (Getters et Setters) :
  - Chaque attribut de la classe "Responsable" est associé à une propriété (getter et setter) qui permet d'accéder à la valeur de l'attribut ou de la modifier.

Programmes Principaux

Programme connection

1. Le formulaire de connexion est créé en utilisant Windows Forms. Il possède des composants graphiques tels que des boutons, des zones de texte, et des boutons radio.
2. L'événement button1_Click est déclenché lorsque l'utilisateur clique sur le bouton "Connexion".
3. Dans cet événement, le code vérifie quelle option (utilisateur, technicien, ou responsable) a été sélectionnée à l'aide de boutons radio.
4. Ensuite, il appelle la classe Bd pour vérifier les informations de connexion en fonction de l'option sélectionnée. La méthode verifUtil, verifTech, ou verifRes est utilisée pour cela.
5. Si l'authentification réussit (c'est-à-dire si la méthode de vérification renvoie une valeur supérieure à zéro), le code ouvre une nouvelle fenêtre correspondante (utilisateur, techniciens, ou resonsable).

Programme Utilisateur

1. Le formulaire principal de l'utilisateur est créé en utilisant Windows Forms. Il possède des composants graphiques tels que des zones de texte, des boutons, une liste déroulante (listBox), etc.
2. L'événement button1_Click est déclenché lorsque l'utilisateur clique sur le bouton "Déclarer Incident". Dans cet événement, le code récupère les informations saisies par l'utilisateur dans les zones de texte (ID utilisateur, poste, description) et crée un nouvel objet Incident avec ces informations.
3. Ensuite, le code appelle la méthode Ajouterunincident de la classe Bd pour ajouter l'incident à la base de données.
4. L'événement button2_Click est déclenché lorsque l'utilisateur clique sur le bouton "Afficher Incidents". Dans cet événement, le code appelle la méthode getLesIncidents de la classe Bd pour récupérer la liste des incidents depuis la base de données.
5. Ensuite, il parcourt la liste des incidents et les affiche dans la liste déroulante (listBox) de l'interface utilisateur.

Programme "Technicien"

Initialisation du formulaire :

La classe techniciens est un formulaire qui semble être utilisé par les techniciens. Lors de son initialisation, le constructeur techniciens() est appelé, et le formulaire est créé.

Affichage des Incidents Déclarés :

Lorsque l'utilisateur clique sur le bouton "Afficher Incidents Déclarés" (button5_Click), le code récupère la liste des incidents déclarés à l'aide de la méthode Bd.getLesIncidents(). Ensuite, il parcourt cette liste et ajoute les détails de chaque incident déclaré à une zone de liste déroulante (comboBox2).

Affichage des Incidents Pris en Charge :

Lorsque l'utilisateur clique sur le bouton "Afficher Incidents Pris en Charge" (button6_Click), le code récupère la liste des incidents "Pris" à l'aide de la méthode Bd.getLesIncidents(). Ensuite, il parcourt cette liste et ajoute les détails de chaque incident pris en charge à une autre zone de liste déroulante (comboBox3).

Ajout d'un Nouveau Matériel :

Lorsque l'utilisateur clique sur le bouton "Ajouter Matériel" (button1_Click_1), le code récupère les données saisies par l'utilisateur (ID de matériel, nom, description), crée un objet Materiel avec ces données, puis utilise la méthode Bd.Ajouterunmateriel(unMateriel) pour ajouter ce matériel à la base de données.

Affichage des Matériels :

Lorsque l'utilisateur clique sur le bouton "Afficher Matériels" (button7_Click), le code récupère la liste des matériels à l'aide de la méthode Bd.getLesMateriel(). Ensuite, il parcourt cette liste et ajoute les détails de chaque matériel à une zone de liste déroulante (comboBox1).

Suppression de Matériel :

Lorsque l'utilisateur sélectionne un matériel dans la zone de liste déroulante (comboBox1) et clique sur le bouton "Supprimer Matériel" (button2_Click), le code supprime le matériel sélectionné de la base de données à l'aide de la méthode Bd.SuppMateriel(idMateriel).

Prise en Charge d'un Incident :

Lorsque l'utilisateur sélectionne un incident dans la zone de liste déroulante (comboBox2) et clique sur le bouton "Prendre en charge" (button3_Click), le code met à jour le statut de l'incident en le marquant comme "Pris" à l'aide de la méthode Bd.ModifStatutPris(idIncident).

Traitement d'un Incident :

Lorsque l'utilisateur sélectionne un incident dans la zone de liste déroulante (comboBox3) et clique sur le bouton "Marquer comme Traité" (button4_Click), le code met à jour le statut de l'incident en le marquant comme "Traité" à l'aide de la méthode Bd.ModifStatutTraite(idIncident).

Programme "Responsable"

Initialisation du formulaire :

La classe responsable est un formulaire utilisé par les responsables. Lors de son initialisation, le constructeur responsable() est appelé, et le formulaire est créé.

Ajout d'un Nouveau Technicien :

Lorsque le responsable clique sur le bouton "Ajouter Technicien" (button1_Click_1), le code récupère les données saisies par l'utilisateur (ID de technicien, mot de passe), crée un objet Technicien avec ces données et des valeurs par défaut pour les autres attributs (nom, prénom, niveau d'intervention, formation, compétences, ID du responsable), puis utilise la méthode Bd.Ajouteruntechnicien(unTechnicien) pour ajouter ce technicien à la base de données.

Affichage des Techniciens :

Lorsque le responsable clique sur le bouton "Afficher Techniciens" (button2_Click), le code récupère la liste des techniciens à l'aide de la méthode Bd.getLesTechnicien(). Ensuite, il parcourt cette liste et ajoute les identifiants des techniciens à deux zones de liste déroulante (comboBox1 et comboBox2).

Suppression d'un Technicien :

Lorsque le responsable sélectionne un technicien dans la zone de liste déroulante (comboBox2) et clique sur le bouton "Supprimer Technicien" (button3_Click), le code supprime le technicien sélectionné de la base de données à l'aide de la méthode Bd.SuppTech(idTechnicien).

Ajout d'un Nouvel Utilisateur :

Lorsque le responsable clique sur le bouton "Ajouter Utilisateur" (button8_Click), le code récupère les données saisies par l'utilisateur (ID d'utilisateur, mot de passe), crée un objet Utilisateur avec ces données et des valeurs par défaut pour les autres attributs (nom, prénom, fonction, niveau d'habilitation), puis utilise la méthode Bd.Ajouterunutilisateur(unUtilisateur) pour ajouter cet utilisateur à la base de données.

Affichage des Utilisateurs :

Lorsque le responsable clique sur le bouton "Afficher Utilisateurs" (button7_Click), le code récupère la liste des utilisateurs à l'aide de la méthode Bd.getLesUtilisateur(). Ensuite, il parcourt cette liste et ajoute les identifiants des utilisateurs à deux zones de liste déroulante (comboBox4 et comboBox3).

Suppression d'un Utilisateur :

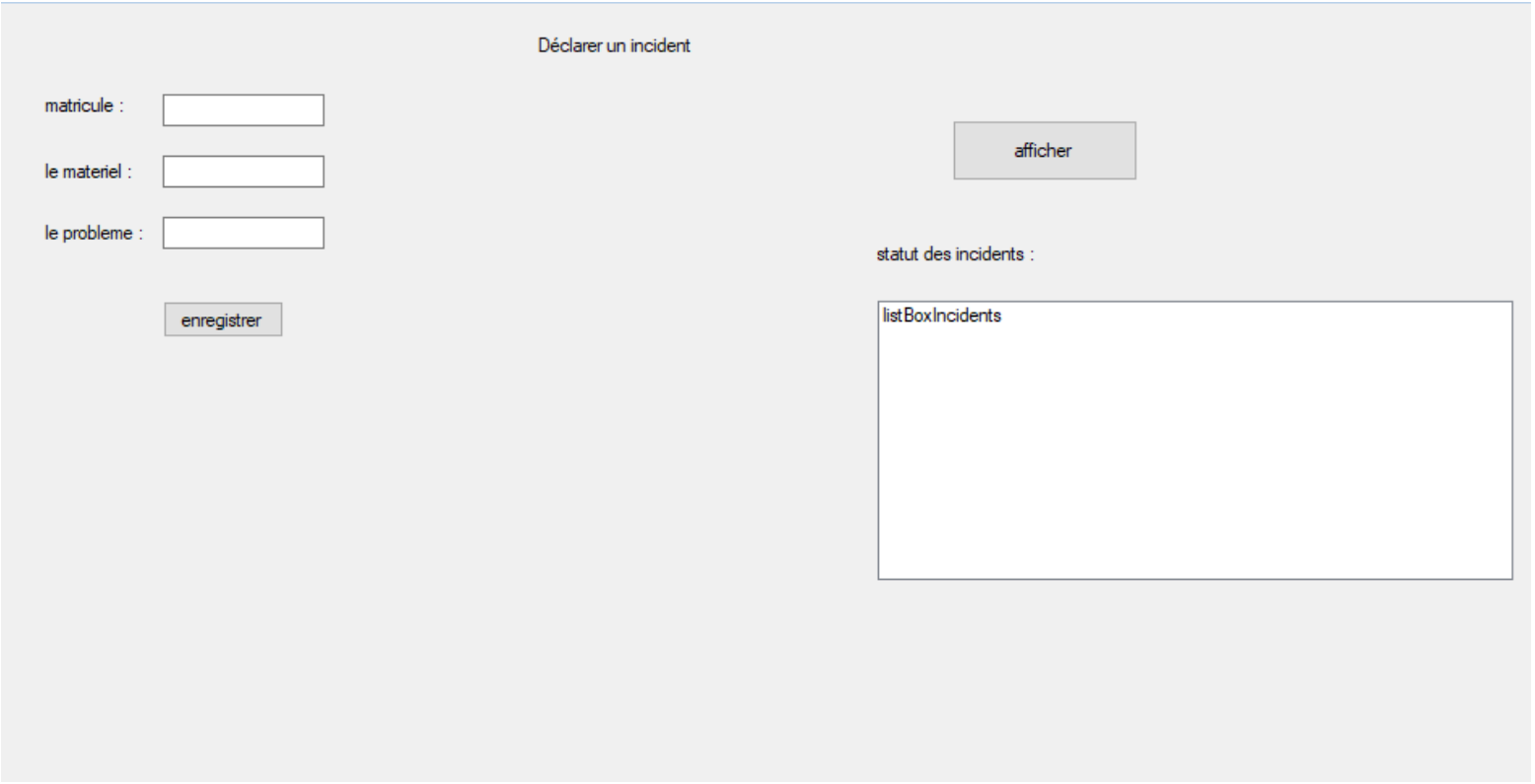
Lorsque le responsable sélectionne un utilisateur dans la zone de liste déroulante (comboBox3) et clique sur le bouton "Supprimer Utilisateur" (button5_Click), le code supprime l'utilisateur sélectionné de la base de données à l'aide de la méthode Bd.SuppUtilisateur(idUtilisateur).

Interfaces

Interface "Connexion"

L'interface de connexion est une fenêtre de l'application qui permet aux utilisateurs d'entrer leur nom d'utilisateur et leur mot de passe, en choisissant leur type d'accès (utilisateur, technicien, responsable). En appuyant sur le bouton "Connexion", le système vérifie ces informations et redirige l'utilisateur vers la section correspondante de l'application en fonction de son rôle. Cette interface est essentielle pour garantir la sécurité et l'accès approprié aux fonctionnalités de l'application.

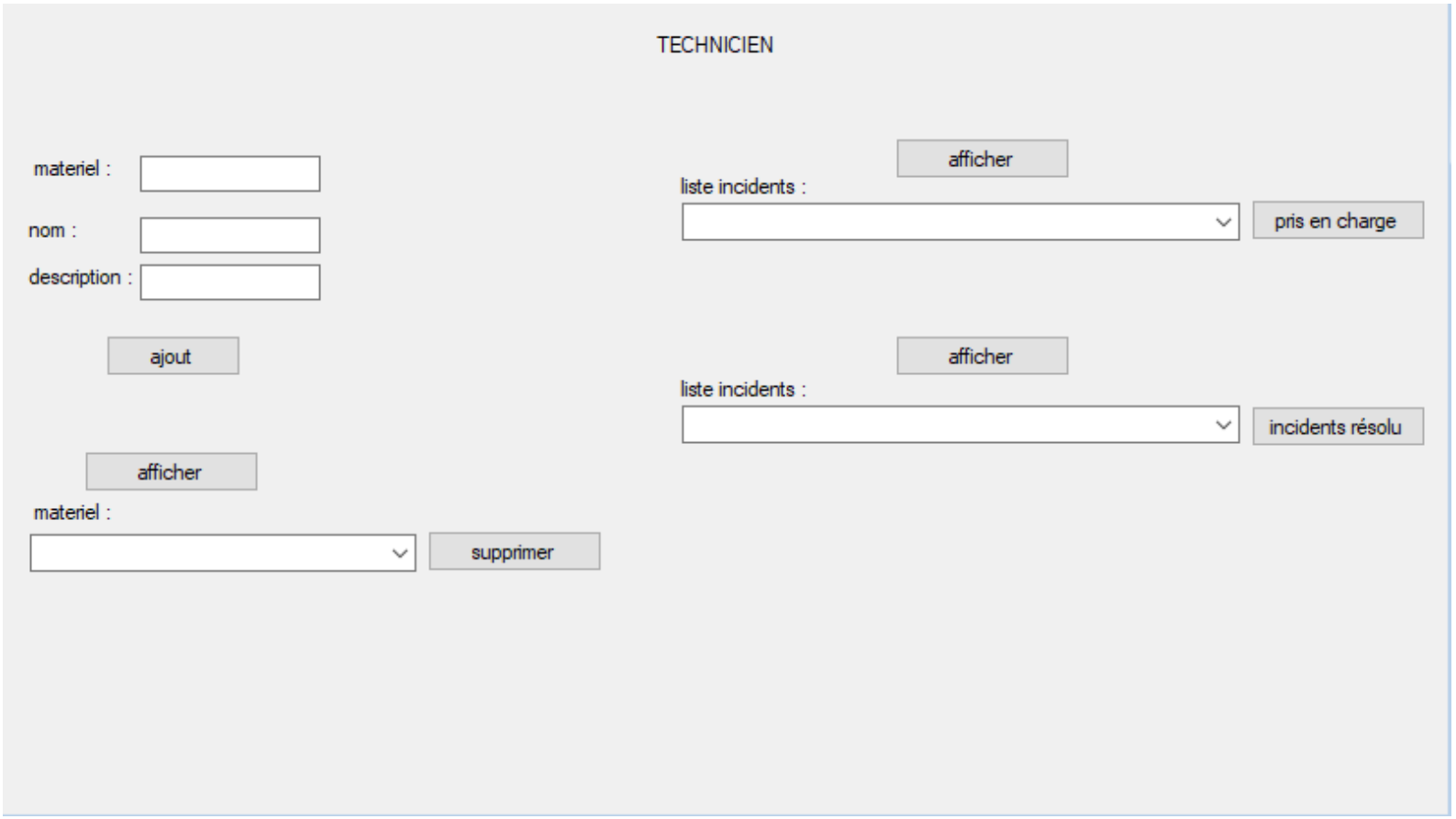
Interface "Utilisateur"



La déclaration d'incident peut inclure des informations telles que la description de l'incident, la date de déclaration, et d'autres détails pertinents. Une fois qu'un incident est déclaré, il peut être suivi pour connaître son état d'avancement, par exemple, s'il a été "déclaré", "pris" ou "traité". Cela permet aux utilisateurs de suivre les incidents et de savoir à quel stade de résolution ils se trouvent.

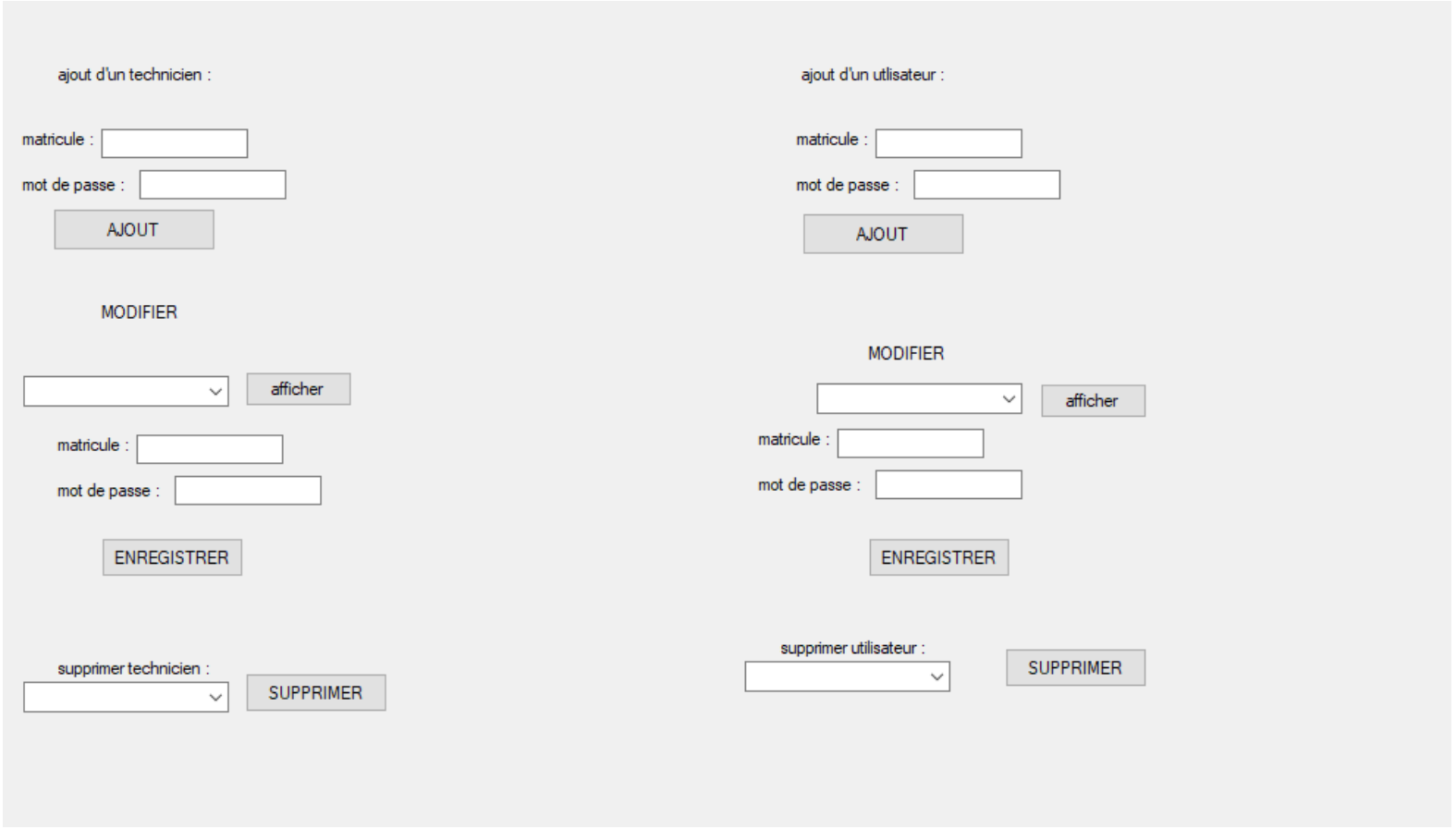
En résumé, votre interface utilisateur facilite la création et la gestion d'incidents, offrant une vue d'ensemble des incidents existants et de leur statut actuel. Cela peut être particulièrement utile dans un contexte de gestion des problèmes et des tâches au sein de votre application.

Interface Technicien



1. **Affichage des Incidents à Traiter :** Les techniciens peuvent accéder à une liste d'incidents qui leur sont assignés ou qui doivent être traités. Cette liste peut inclure des détails tels que la description de l'incident, la date de déclaration, et son statut actuel.
2. **Mise à Jour de l'État des Incidents :** Les techniciens peuvent avoir la possibilité de mettre à jour l'état des incidents, par exemple en marquant un incident comme "pris" lorsqu'ils commencent à travailler dessus, ou en le marquant comme "traité" une fois le problème résolu.
3. **Ajout de Nouveaux Matériels :** Les techniciens peuvent ajouter de nouveaux équipements ou matériels à la base de données, en fournissant des informations telles que l'identifiant du matériel, le nom et la description.
4. **Suppression de Matériels :** Ils peuvent également avoir la possibilité de supprimer des matériels de la base de données si cela est nécessaire.
5. **Affichage des Matériels Existant :** Une liste des matériels existants peut être affichée pour que les techniciens puissent consulter les équipements disponibles.

Interface "Responsable"



Affectation des Techniciens : Les responsables peuvent assigner des techniciens spécifiques à des incidents en fonction de leur disponibilité et de leur expertise.

Ajout de Nouveaux Techniciens : Les responsables ont la possibilité d'ajouter de nouveaux techniciens à la base de données en fournissant leurs informations personnelles, telles que le nom, le prénom, le niveau d'intervention, le mot de passe, la formation et les compétences.

Suppression de Techniciens : Si nécessaire, les responsables peuvent supprimer des techniciens de la base de données.

Ajout de Nouveaux Utilisateurs : Ils peuvent également ajouter de nouveaux utilisateurs à l'application, en fournissant leurs informations de base, telles que le nom, le prénom, la fonction, le niveau d'habilitation et le mot de passe.

Suppression d'Utilisateurs : Les responsables peuvent supprimer des utilisateurs de la base de données en cas de besoin.

Structure Base de Données

modèle logique de données (MLD)

Responsable = (ID_responsable VARCHAR(50), mot_de_passe VARCHAR(50));
Matériel = (ID_materiel VARCHAR(50), Nom VARCHAR(50), Description VARCHAR(50));
Utilisateur = (ID_utilisateur_ VARCHAR(50), Nom VARCHAR(50), Prénom VARCHAR(50), Fonction VARCHAR(50), Niveau_habilitation VARCHAR(50), mot_de_passe VARCHAR(50), #ID_responsable);
Technicien = (ID_technicien VARCHAR(50), Nom VARCHAR(50), Prenom VARCHAR(50), NiveauIntervention VARCHAR(50), mot_de_passe VARCHAR(50), Formation VARCHAR(50), Competences VARCHAR(50), #ID_responsable);
Incident = (ID_incident VARCHAR(50), Description VARCHAR(50), Statut VARCHAR(50), Date_declaration VARCHAR(50), Date_resolution VARCHAR(50), travail_a_réaliser VARCHAR(50), #ID_materiel, #ID_technicien, #ID_utilisateur_);

Schéma conceptuel de données(MCD)

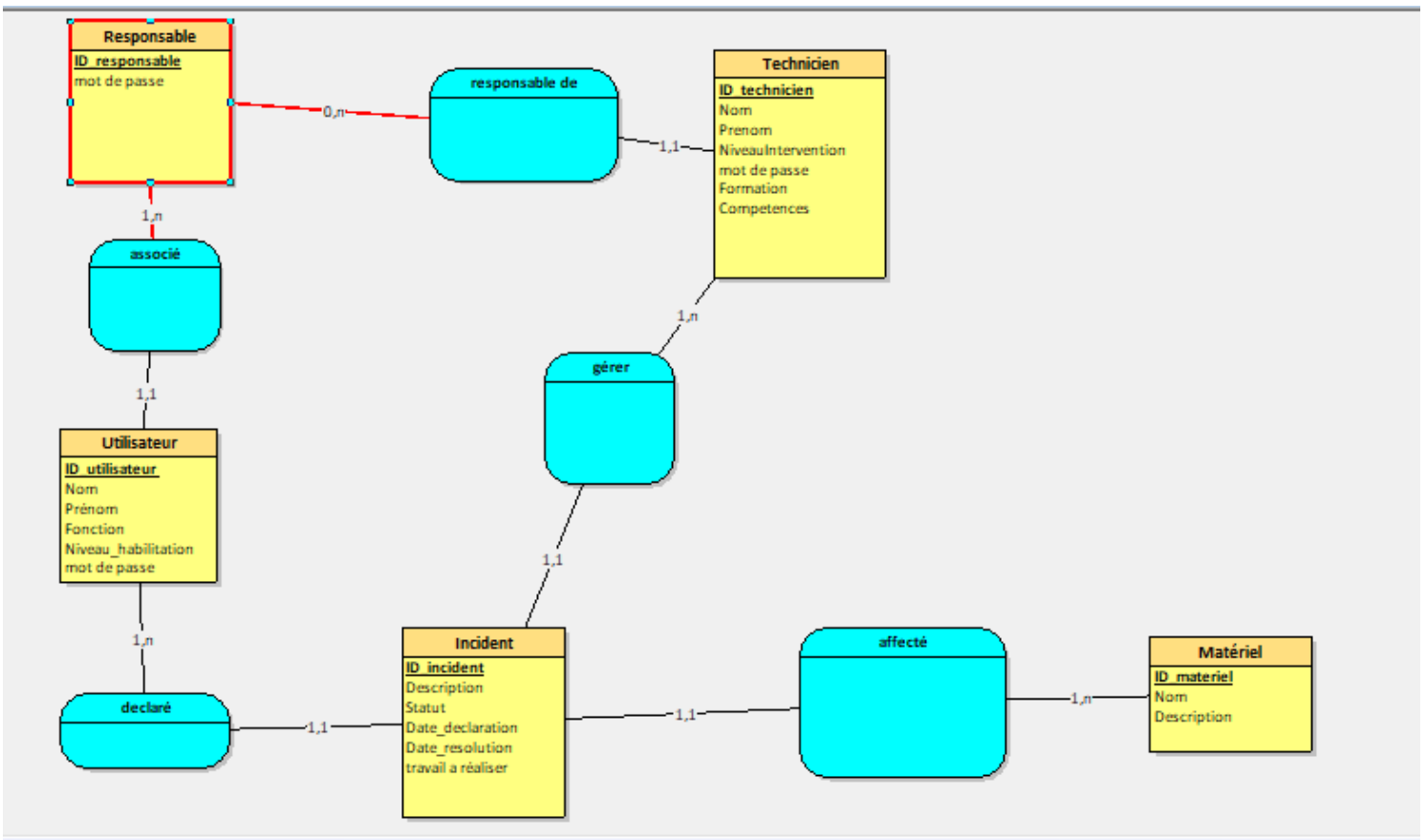
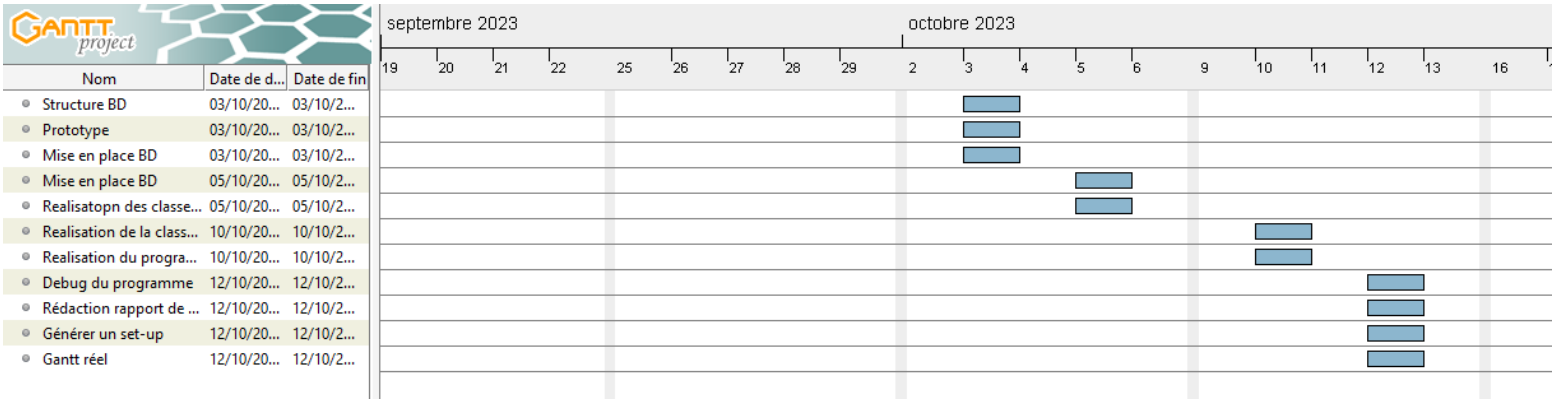
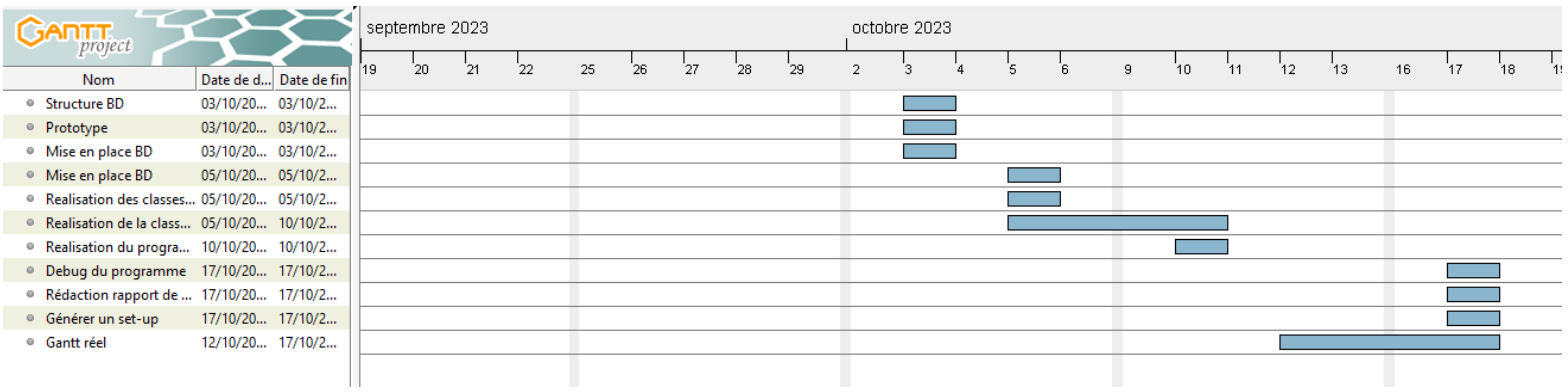


Diagramme De Gantt

Gantt Prévisionnelle



Gantt Finale



Conclusion

Ce rapport a couvert en détail le projet "Laboratoire GSP" en C#. L'application permet une gestion complète des incidents, du matériel, des utilisateurs, des techniciens et des responsables dans un environnement de laboratoire.

L'application repose sur une classe de base de données (Bd) pour gérer efficacement les opérations de base de données. Les différentes classes, telles que "Incident," "Materiel," "Utilisateur," "Technicien," et "Responsable," ont été présentées en détail.

Les programmes principaux ont été examinés, montrant comment chaque rôle d'utilisateur interagit avec l'application pour accomplir des tâches spécifiques..

En résumé, le projet "Laboratoire GSP" démontre l'efficacité de la programmation orientée objet et de la gestion de base de données pour la résolution de problèmes pratiques dans un environnement de laboratoire. Ce rapport sert de guide complet pour les parties prenantes du projet.