BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2024**

Épreuve E5 - Conception et développement d'applications (option SLAM)

ANNEXE 7-1-B : Fiche descriptive de réalisation professionnelle (recto)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE	**N° réalisation : 1**
Nom, prénom : HANY Mathias	**N° candidat : 02113110453**
Épreuve ponctuelle ☐ **Contrôle en cours de formation** ☒	**Date :** 28 / 05 / 2024

Organisation support de la réalisation professionnelle
GSB

Intitulé de la réalisation professionnelle
Application de gestion de parcs en C#

Période de réalisation : 4 semestre **Lieu :** Lycée Henri-Matisse Cugnaux
Modalité : ☐ **Seul(e)** ☒ **En équipe**

Compétences travaillées
- ☒Concevoir et développer une solution applicative
- ☒Assurer la maintenance corrective ou évolutive d'une solution applicative
- ☒Gérer les données

Conditions de réalisation[1] (ressources fournies, résultats attendus)
Ressources Fournis : Cahier des charges
Attendus : Gant avant /aprés, Applicaton Opérationelle, manuel d'utilisateur, documentation du projet.

Description des ressources documentaires, matérielles et logicielles utilisées[2]
- GANT PROJECT : Pour planifier les tâches
- Looping : Pour modeliser
- BD MySql avec PhpMyAdmin : Héberger nos données
- Visual Studio : Pour développer
- Live Share pour partager mon code

Modalités d'accès aux productions[3] et à leur documentation[4]
https://github.com/Ilovanne/ProjetAnnee2-CSharp-LaboGSB.git

1En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

2Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

3Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

4Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

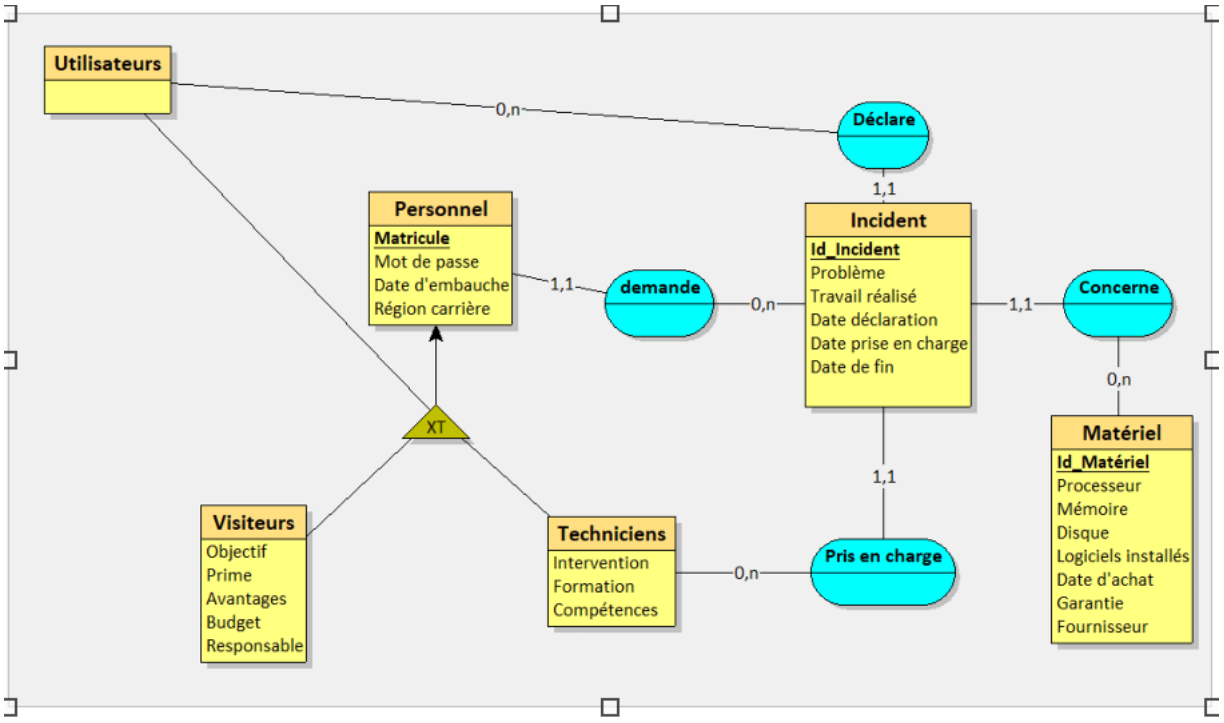
BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2024**

Épreuve E5 - Conception et développement d'applications (option SLAM)

ANNEXE 7-1-B : Fiche descriptive de réalisation professionnelle (verso, éventuellement pages suivantes)

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

MCD :



MLD :

Matériel = (Id_Matériel COUNTER, Processeur VARCHAR(50), Mémoire INT, Disque VARCHAR(50), Logiciels_installés VARCHAR(50), Date_d_achat DATE, Garantie VARCHAR(50), Fournisseur VARCHAR(50));

Personnel = (Matricule VARCHAR(50), Mot_de_passe VARCHAR(50), Date_d_embauche DATE, Région_carrière VARCHAR(50), #Id_Incident);

Visiteurs = (#Matricule, Objectif INT, Prime INT, Avantages VARCHAR(50), Budget INT, Responsable VARCHAR(50));

Techniciens = (#Matricule, Intervention DATE, Formation VARCHAR(50), Compétences VARCHAR(50));

Incident = (Id_Incident COUNTER, Problème VARCHAR(50), Travail_réalisé VARCHAR(50), Date_déclaration DATE, Date_prise_en_charge DATE, Date_de_fin DATE, #(#Matricule), #(#Matricule_1), #Id_Matériel)

Utilisateurs = (#Matricule);

Fonctionalités developpées :

Page de connexion : Le bouton connexion vérifie les informations écrites dans les textBox et regarde dans la base de données si elles correspondent à une personne enregistrée dedans. Si les identifiants sont valides, cela va vous rediriger vers la page dédiée sinon cela va faire une erreur de connexion.

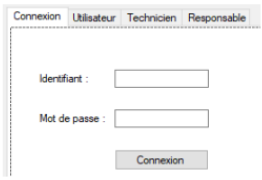
Page utilisateur : Le bouton valider permet de déclarer un incident qui sera enregistré dans la base de données et pourra être vu et prit en charge par un technicien. Le bouton afficher permet à l'utilisateur de voir l'avancée des incidents qu'il a déclaré.

Page technicien : Le bouton confirmer dans la section matériel permet d'enregistrer un matériel dans la base de données. Le bouton consulter permet d'afficher dans la listBox les informations d'un matériel sélectionné dans la comboBox. Le bouton supprimer permet lui de supprimer le matériel sélectionné dans une autre comboBox. Le bouton consulter dans la section incident permet de voir les incidents déclarés par les utilisateurs. Le bouton enregistrer permet à lui de prendre en charge un incident sélectionné dans une comboBox et mettre à jour son état.

Page responsable : Dans la section technicien ajouter, le bouton enregistrer permet d'ajouter à la base de données un technicien. Le bouton enregistrer dans la section technicien modifier, permet lui de changer des informations d'un technicien sélectionné dans une comboBox. Dans la section utilisateur modifier, le bouton enregistrer permet d'ajouter à la base de données un utilisateur. Le bouton enregistrer dans la section utilisateur modifier, permet lui de changer des informations d'un utilisateur sélectionné dans une comboBox. Dans la section visualiser, les boutons voir permettent de voir les statistiques respectable à chacun. Enfin, dans la section supprimer, les boutons supprimer permettent de supprimer un utilisateur et/ou un technicien de la base de données

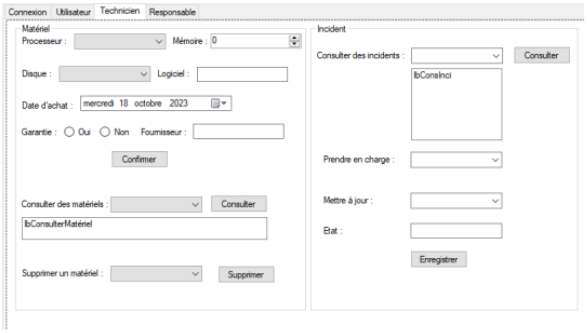
Screens Interface :

Page de connexion :

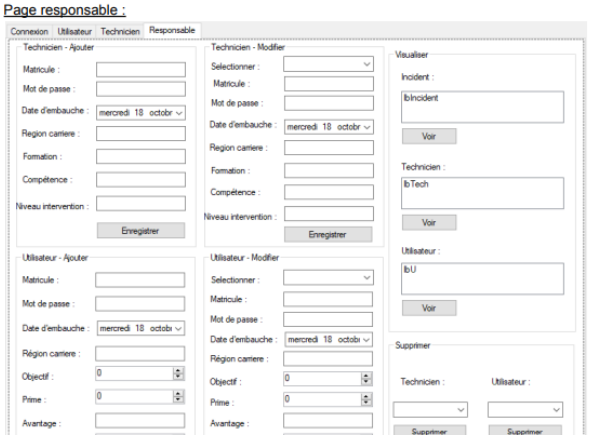


Page utilisateur :

Page technicien :



Page responsable :



Description des classes :

Classe BD : Cette classe contient des méthodes pour interagir avec la base de données MySQL utilisée dans l'application. méthodes principales :

- `VerifConn(string id, string mdp)` : Vérifie la connexion en comparant les identifiants fournis avec la base de données et renvoie le rôle de l'utilisateur ou une erreur de connexion.
- `AjoutTechnicien(Technicien unTechnicien)` : Ajoute un technicien à la base de données.
- `supprTechnicien(string id)` : Supprime un technicien de la base de données.
- `AjoutPersonnel(Personnel unPersonnel)` : Ajoute un membre du personnel à la base de données.
- `AjoutUtilisateur(Utilisateur unUtilisateur)` : Ajoute un utilisateur à la base de données.
- `ModifierUtilisateur(Utilisateur unUtilisateur)` : Modifie un utilisateur dans la base de données.
- `supprUtilisateur(string id)` : Supprime un utilisateur de la base de données.
- `AjoutIncident(Incident unIncident)` : Ajoute un incident à la base de données.
- `AjoutMateriel(Materiel unMateriel)` : Ajoute un matériel à la base de données.
- `GetIdMateriel()` : Récupère la liste des identifiants de matériel depuis la base de données.
- `GetUnMateriel(int id)` : Récupère les détails d'un matériel en fonction de son identifiant depuis la base de données.
- `SuppMateriel(int id)` : Supprime un matériel de la base de données en fonction de son identifiant.
- `GetMatriculeTechnicien()` : Récupère la liste des matricules de techniciens depuis la base de données.
- `GetMatriculeUtilisateur()` : Récupère la liste des matricules d'utilisateurs depuis la base de données.

- `affichIncident()` : Récupère la liste des incidents depuis la base de données.
- `majEtatIncident(int unId, int unEtat)` : Met à jour l'état d'un incident dans la base de données.

Ces méthodes permettent de gérer différentes opérations de manipulation des données dans la base de données MySQL utilisée par l'application.

classe Incident représente un incident dans l ‘application. Résumé des éléments qu'elle contient :

- **Attributs** :
  - `id`: L'identifiant de l'incident.
  - `probleme`: Description du problème associé à l'incident.
  - `materiel`: Description du matériel associé à l'incident.
  - `travailRealise`: Description du travail réalisé pour résoudre l'incident.
  - `dateDeclaration`: Date de déclaration de l'incident.
  - `datePriseEnCharge`: Date de prise en charge de l'incident.
  - `dateFin`: Date de fin de l'incident.
  - `matriculePerso`: Matricule du personnel associé à l'incident.
  - `idMateriel`: Identifiant du matériel associé à l'incident.
  - `matriculeTech`: Matricule du technicien associé à l'incident.
  - `etat`: État de l'incident.
- **Constructeurs** :
  - Ils permettent de créer des instances de la classe avec différentes combinaisons de paramètres.
- **Méthodes** :
  - Méthodes `get` et `set` pour accéder et modifier les attributs de l'incident.
  - `setEtat(int unEtat)` : Modifie l'état de l'incident.
  - `getEtat()` : Récupère l'état de l'incident.

La classe Materiel représente du matériel informatique et contient les propriétés suivantes :

- **Attributs** :
  - `idMateriel`: L'identifiant du matériel.
  - `processeur`: Le processeur du matériel.
  - `memoire`: La capacité mémoire du matériel.
  - `disque`: Le type de disque du matériel.
  - `logicielInstalles`: Les logiciels installés sur le matériel.
  - `datedAchat`: La date d'achat du matériel.
  - `garantie`: Un indicateur de garantie pour le matériel.
  - `fournisseur`: Le fournisseur du matériel.
- **Constructeurs** :
  - `Materiel(int idMateriel, string processeur, int memoire, string disque, string logicielInstalles, DateTime datedAchat, bool garantie, string fournisseur)`: Constructeur pour la classe avec ID spécifié.
  - `Materiel(string processeur, int memoire, string disque, string logicielInstalles, DateTime datedAchat, bool garantie, string fournisseur)`: Constructeur pour la classe sans ID spécifié.
- **Méthodes** :
  - Méthodes `get` et `set` pour accéder et modifier les attributs du matériel.

Cette classe fournit des fonctionnalités pour représenter et gérer des informations sur le matériel informatique, telles que le processeur, la mémoire, le disque, les logiciels installés, etc.

La classe Personnel représente le personnel du système et contient les propriétés suivantes :

- **Attributs** :
  - `matricule`: Le matricule du personnel.
  - `mdp`: Le mot de passe du personnel.
  - `dateEmb`: La date d'embauche du personnel.
  - `regCarriere`: La région de carrière du personnel.
  - `responsable`: Un indicateur pour déterminer si le personnel est responsable.
  - `Objectif`: L'objectif de l'utilisateur.
  - `Prime`: La prime de l'utilisateur.
  - `Avantages`: Les avantages de l'utilisateur.
  - `Budget`: Le budget de l'utilisateur.
  - `ResponsableMatricule`: Le matricule du responsable de l'utilisateur.
- **Constructeurs** :
  - `Personnel(string matricule, string mdp, DateTime dateEmb, string regCarriere, int responsable)`: Constructeur pour la classe Personnel.
  - `Personnel(string matricule, string mdp, DateTime dateEmb, string regCarriere, int responsable, int Objectif, int Prime, string Avantages, int Budget, string ResponsableMatricule)`: Constructeur pour les utilisateurs avec initialisation des propriétés spécifiques aux utilisateurs.
- **Méthodes** :
  - Méthodes `get` et `set` pour accéder et modifier les attributs du personnel.

Cette classe fournit des fonctionnalités pour représenter et gérer les informations relatives au personnel du système, telles que le matricule, le mot de passe, la date d'embauche, etc.

La classe Technicien hérite de la classe Personnel. Voici ce que fait cette classe :

1. **Propriétés spécifiques au technicien** :
    - Elle définit des propriétés supplémentaires spécifiques au technicien telles que la date d'entrée en fonction, la formation, les compétences et le type d'intervention.
2. **Constructeur** :
    - Elle a un constructeur qui prend des paramètres pour initialiser les propriétés de la classe de base `Personnel` ainsi que les propriétés spécifiques au technicien.
3. **Méthodes d'accès** :
    - Elle fournit des méthodes d'accès (getters) pour obtenir les valeurs des propriétés spécifiques au technicien.
4. **Méthodes de modification** :
    - Elle fournit des méthodes de modification (setters) pour mettre à jour les valeurs des propriétés spécifiques au technicien.

En résumé, la classe `Technicien` encapsule les informations propres à un technicien dans un système de gestion de services, tout en bénéficiant des fonctionnalités générales fournies par la classe de base `Personnel`.

La classe `Utilisateur` hérite également de la classe `Personnel`. Voici ce que fait cette classe :

1. **Propriétés spécifiques à l'utilisateur** :
    - Elle définit des propriétés supplémentaires spécifiques à l'utilisateur telles que l'objectif, la prime, les avantages et le budget.
2. **Constructeur** :
    - Elle a un constructeur qui prend des paramètres pour initialiser les propriétés de la classe de base `Personnel` ainsi que les propriétés spécifiques à l'utilisateur.
3. **Méthodes d'accès** :
    - Elle fournit des méthodes d'accès (getters) pour obtenir les valeurs des propriétés spécifiques à l'utilisateur.
4. **Méthodes de modification** :
    - Elle fournit des méthodes de modification (setters) pour mettre à jour les valeurs des propriétés spécifiques à l'utilisateur.

En résumé, la classe `Utilisateur` encapsule les informations propres à un utilisateur dans un système de gestion de services, tout en bénéficiant des fonctionnalités générales fournies par la classe de base `Personnel`.

classe `Visiteur`. Voici ce que fait cette classe :

1. **Propriétés** :
    - La classe `Visiteur` possède plusieurs propriétés privées telles que `obj`, `prime`, `avantages`, `budget`, `responsable` et `matricule`.

2. **Constructeur** :

   - La classe a un constructeur qui prend plusieurs paramètres pour initialiser les valeurs des propriétés de l'objet visiteur.

3. **Utilité** :

   - La classe `Visiteur` est une représentation d'un visiteur dans un système de gestion, mais elle est actuellement incomplète car la construction est arrêtée avant d'initialiser toutes les propriétés.