I- Règles d'utilisations

a- Règles liées à la connexion

La connexion se fait par l'intermédiaire d'un compte utilisateur, via le pseudo et le mot de passe de l'utilisateur plus précisément. Si aucun utilisateur n'existe dans la base de données, les identifiants par défauts pour la connexion sont admin pour le pseudo et admin comme mot de passe. Les identifiants par défauts peuvent être modifiés en modifiants les contenus des variables `DEFAUL_PSEUDO` et `DEFAUL_PASSWORD` contenu dans le fichier AppConstants.cs dans le dossier **Customs** se trouvant à la racine du projet.

NB : Si un utilisateur existe dans la base de données, les identifiants par défauts ne sont plus considérés lors de la connexion.

b- Règles liées à la maintenance

L'ajout d'une maintenance est conditionné par l'existence d'un type de maintenance, de ce fait, si aucun type de maintenance n'existe, l'ajout ne pourra pas être réalisé. Vous serez constamment redirigé vers la page d'ajout de type de maintenance.

c- Règles à l'ensemble des données.

Seule la personne qui à créer une donnée peut la modifier ou la supprimer, mais l'information est visible par l'ensemble des utilisateurs.

Seul l'administrateur peut créer un compte utilisateur, l'interface est paramétrée pour valider cette règle.

II- Explication du code.

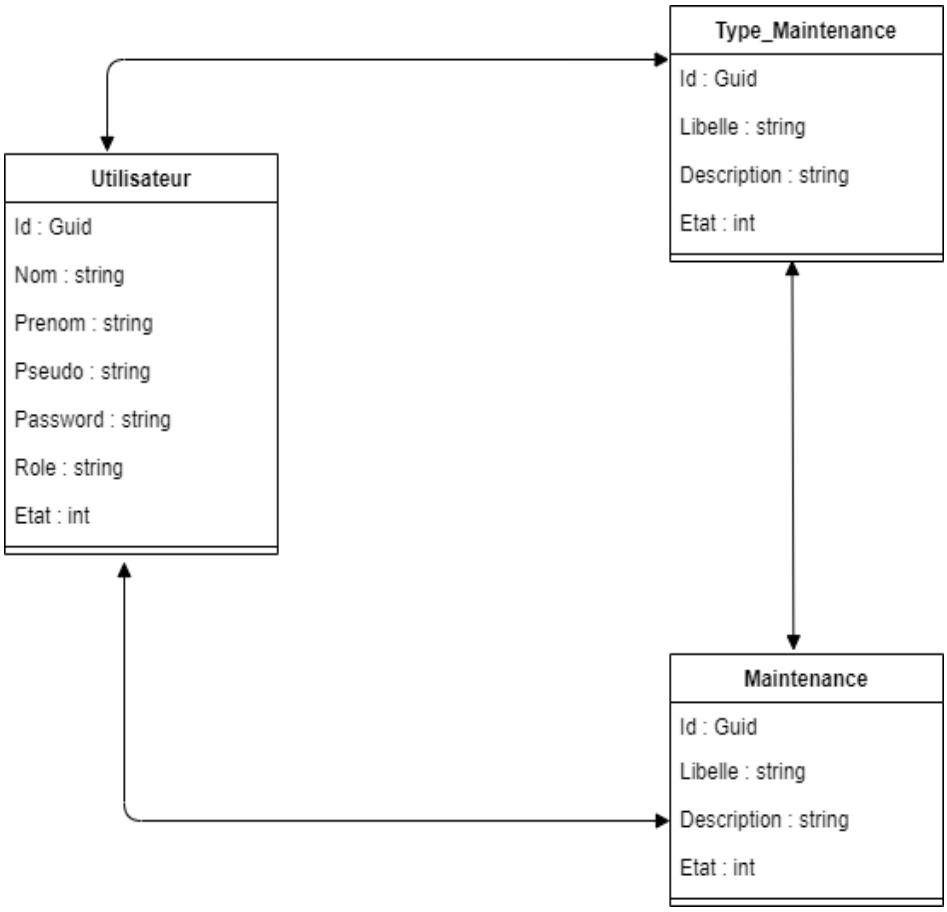
- L'ensemble des interfaces se trouve dans le dossier **Pages** se trouvant à la racine du projet.
- Le dossier **Services** contient la classe pour communiquer avec la base de données.
- Le dossier **Interfaces** contient l'interface utilisée pour injecter les services dans les différentes pages. (Si vous ne comprenez pas le terme « Injecter », recherchez **Injection de dépendance** vous aurez de quoi vous occuper).
- L'image qui suit est le diagramme de classe du projet.

  Explication des classes utilisées :
  - Utilisateur : Pour gérer les différentes personnes pouvant réaliser des actions dans l'application.
  - TypeMaintenance : Pour gérer les différents types de maintenances. Comme une maintenance évolutive…
  - Maintenance : Pour gérer les maintenances.

  Explications des propriétés pouvant porter confusion :
  - Id : Au lieu d'utiliser les entiers comme vous le fait habituellement les **GUID** sont plus appropriés pour gérer tout ce qui est clé unique.
  - Role : C'est une propriété de type UtilisateurRole qui est une énumération, de ce fait qui a une valeur finie.
  - Etat : Propriété pour gérer l'état d'une donnée. Exemple : 0 pour dire que c'est une nouvelle donnée, 1 pour dire que la donnée a été supprimée.

  Explication des différentes relations entre les classes.

  - Un utilisateur peut gérer **un ou plusieurs** Maintenance ou TypeMaintenance. De ce fait la clé de l'utilisateur doit migrer dans la table Maintenance et TypeMaintenance.
  - Un TypeMaintenance peut être utilisé par plusieurs Maintenances, de ce fait la clé de TypeMaintenance doit migrer vers Maintenance.


Type_Maintenance
Id : Guid
Libelle : string
Description : string
Etat : int
Utilisateur
Id : Guid
Nom : string
Prenom : string
Pseudo : string
Password : string
Role : string
Etat : int
Maintenance
Id : Guid
Libelle : string
Description : string
Etat : int