

BTS SERVICES INFORMATIQUES AUX ORGANISATION – SESSION 2023

EPREUVE E5 – CONCEPTION ET DEVELOPPEMENT D'APPLICATIONS (Option SLAM)

Projet 1 : StockManager

Contexte :

Le besoin primaire évoqué par L'épicerie du village était que pour une si petite quantité de stock, elle ne trouvait de solution lui permettant de gérer au plus simple ses stocks. Désormais grâce à cet outil, la directrice de la boutique pourra désormais se connecter sur l'application pour pouvoir gérer les entrées de ses produits, gérer les utilisateurs ou encore suivre son stock en ajoutant les commandes amenant les sorties d'articles du stock. L'entreprise n'ayant pas beaucoup de moyen ni de ressources informatiques souhaite pouvoir utiliser l'application sur le poste fixe présent dans la pièce dédiée au stock.

Sommaire :


1. Introduction
2. Architecture
3. Technologies Utilisées
4. Installation
5. Base de données


1. Introduction


Ce projet est une application client lourd de gestion des stocks. Il permet de gérer les entrées dans le stock ainsi que les commandes reçues par l'entreprise. Vous pourrez y retrouver les opérations du CRUD en liaison avec la base de données. Cela permet, d'ajouter, modifier, supprimer ou simplement consulter les données relatives aux commandes, au stock, aux fournisseurs, aux catégories d'articles du stock ou des utilisateurs. Cette application a été conçu pour pouvoir tourner sur une machine locale pour simplifier ces tâches à l'entreprise qui utilisait auparavant un simple tableur.

2. Architecture

Pour développer cette application j'ai utilisé le modèle MVC (Model View Controller). Cela m'a permis de séparer distinctement les modèles d'objet présent en base de données, l'interface et les fichiers qui gèrent le traitement des données. Cette application est reliée à une base de données pour permettre le stockage des données. Vous pourrez retrouver dans le [répertoire Github](#) du projet un dossier « Model » répertoriant les classes du projet, un dossier « View » qui contient les fichiers relatifs à l'interface graphique ainsi qu'un dossier « Controller » qui contient tous les fichiers avec les fonctions pour le traitement des données. Enfin, le répertoire doc répertorie tous les fichiers utiles pour l'utilisation du projet.


 master ▾


 1 branch

 0 tags


Go to file











Add file ▾

 Code ▾

 **tenjustin** correct

026518a yesterday

 12 commits

 Controller	correct	yesterday
 Model	day before E5	3 weeks ago
 View	correct	yesterday
 lib	before e5	3 weeks ago
 .gitattributes	Ajouter .gitattributes, .gitignore et README.md	last month
 .gitignore	Ajouter .gitattributes, .gitignore et README.md	last month
 GestionStock.csproj	datacategories	last month
 GestionStock.sln	correct	yesterday
 Program.cs	before e5	3 weeks ago
 README.md	Ajouter .gitattributes, .gitignore et README.md	last month

Vous retrouverez également les fichiers relatifs au répertoire Github ainsi que les fichiers relatifs au développement du projet sur Visual Studio et pour finir, le fichier Program.cs qui gère le lancement de l'application.

3. Technologies utilisées

Pour développer ce projet, j'ai décidé d'utiliser le langage C# pour approfondir mon apprentissage de ce langage très demandé dans le monde professionnel ainsi que pour apprendre à utiliser l'IDE Visual Studio. Le choix de cette technologie m'a permis de réaliser une application desktop entière. Pour la base de données de mon application, j'ai utilisé une base de données Microsoft SQL Server hébergé en local car cela était plus simple à mettre en place en local pour le développement.

4. Installation

Pour installer l'application en local vous pouvez cloner le répertoire Github du projet (<https://github.com/tenjustin/GestionStockPPE>). Vous aurez juste à copier le script SQL du répertoire /doc dans votre serveur local MySQL.

Changez ensuite la variable `connectionString` du fichier `/Controller/DBConnexion.cs` pour la chaîne de caractère de connexion à votre base de données :

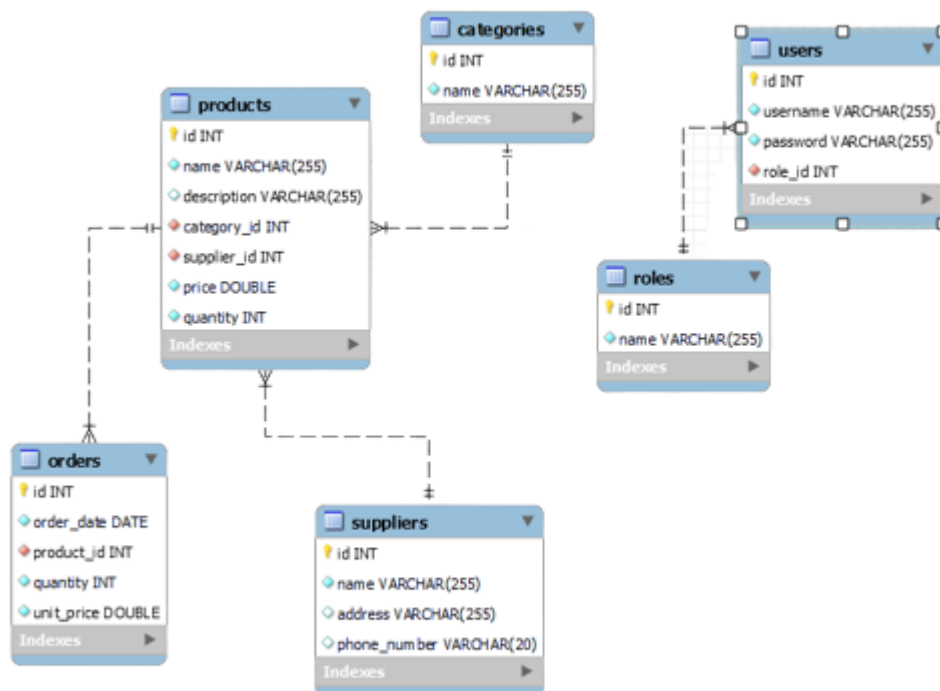
```
30 références
internal class DBConnexion
{
    //Chaîne de caractère à adapter à votre base de données
    private string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Justin\BDD\GestionStockP

    30 références
    public SqlConnection GetConnection()
    {
        SqlConnection connection = new SqlConnection(connectionString);
        return connection;
    }
}
```

Vous pouvez ensuite lancer le projet en appuyant sur la touche « f5 ». Vous aurez accès alors à la version de développement de l'application.

5. Base de données

Voici le schéma de la base de données de l'application :



Mise en place d'un déclencheur lors de la mise à jour des commandes pour mettre à jour automatiquement le stock :

```

--Declencheur pour maj des commandes (maj du stock auto)
CREATE TRIGGER majStockUpdateOrder ON orders AFTER UPDATE AS
BEGIN
    Declare @deltaQty INT
    declare @oldQty INT
    declare @newQty INT
    declare @qtyUpdt INT
    set @qtyUpdt = (Select quantity from products where id=(Select product_id from inserted))
    set @oldQty = (Select quantity from deleted)
    set @newQty = (Select quantity from inserted)
    IF (@oldQty >= @newQty)
    BEGIN
        SET @deltaQty = @oldQty - @newQty
        Update products SET quantity = @qtyUpdt + @deltaQty Where id = (Select product_id from inserted)
    END
    ELSE
    BEGIN
        SET @deltaQty = @newQty - @oldQty;
        Update products SET quantity = @qtyUpdt - @deltaQty Where id = (Select product_id from inserted)
    END
END

```