PROGRAMMATION ORIENTÉE OBJET II

INF11207 (MS)

UNIVERSITÉ DU QUÉBEC À RIMOUSKI

DÉPARTEMENT DE MATHÉMATIQUES, DÍNFORMATIQUE ET DE GÉNIE

RAPPORT

Travail Pratique

ÉQUIPE :

Bah Amadou

PROFESSEUR :

Yacine Yaddaden, Ph. D.

Date : 5 Avril 2025

Table des matières

1 Introduction

Ce projet a pour objectif de dímplémenter une application console en C# pour gérer un magasin de fleurs. Il met en pratique des concepts avancés de la programmation orientée objet (POO), tels que les classes abstraites, les interfaces, l'héritage, la surcharge d'opérateurs, ainsi que l'utilisation des bibliothèques externes CsvHelper et Json.NET.

2 Technologies Utilisées

— **Langage** : C#
— **IDE** : Vs Code sur linux
— **Contrôle de version** : Git, GitHub
— **Bibliothèques** : CsvHelper, Json.NET
— **Modélisation UML** : NClass

3 Analyse et Conception

3.1 Diagramme de Classes

Le diagramme de classes UML comprend les entités suivantes :
— **Utilisateur (abstraite)** : id, nom, prenom, email, motDePasse
— **Client, Vendeur, Propriétaire, Fournisseur** : héritent de Utilisateur
— **Fleur** : nom, couleur, prix, description, quantité
— **Bouquet** : liste de fleurs, prixTotal
— **Commande** : client, vendeur, articles, statut
— **Facture** : commande, montantTotal, modePaiement

3.2 Relations

— Une **Commande** est associée à un **Client** et un **Vendeur**
— Une **Commande** contient des **Fleurs** ou des **Bouquets**
— Une **Facture** est liée à une **Commande**

4 Fonctionnalités Implémentées

4.1 Gestion des Utilisateurs

Différents rôles d'utilisateurs (client, vendeur, propriétaire, fournisseur) sont gérés avec leurs droits et tâches spécifiques.

4.2 Gestion des Fleurs

Les données sont importées depuis un fichier CSV via CsvHelper. Chaque fleur est caractérisée par un nom, une couleur, un prix, une description et une qunatité ajoutée lors du

chargement du fichier csv.

4.3 Création de Bouquets

Les bouquets peuvent être composés de plusieurs fleurs, et son prix est calculé automatiquement.

4.4 Gestion des Commandes

Les clients peuvent créer des commandes contenant des fleurs ou des bouquets. Les vendeurs gèrent les statuts de commande en les validant ou en les annulant. Si vailidée, Une facture en format pdf est automatiquement générée.

4.5 Facturation

Une facture est générée une fois le paiement effectué, contenant le détail de la commande et le mode de paiement.

5 Persistance des Données

5.1 Fichiers CSV

La bibliothèque CsvHelper est utilisée pour lire le fichier `fleurs_db.csv`. Voici un exemple :

```
Nom,Couleur,Prix,Description
Rose,Rouge,5.0,Symbole d'amour
Tulipe,Jaune,3.5,Color\'ee et vive
```

5.2 Fichiers JSON

Les données comme les commandes et factures sont sauvegardées en JSON à l'aide de Json.NET.

6 Exemples de Code

6.1 Chargement des Fleurs

```
public void LoadFleursFromCsv()
        {
            list_fleurs.Clear(); // Clear existing flowers to prevent duplic
            using (var reader = new StreamReader("fleurs_db.csv"))
            using (var csv = new CsvReader(reader, new CsvConfiguration(Cultu
            {
                HeaderValidated = null, // Skip header validation
```

```
            MissingFieldFound = null // Skip missing field validation
        }))
        {
            // Read records from CSV
            var records = csv.GetRecords<dynamic>(); // Use dynamic for a

            // Add records to the list
            foreach (var record in records)
            {
                // Access each field by header name (as defined in the CS
                string nom = record.Nom ?? "Unknown";
                string couleur = record.Couleur ?? "Unknown";
                string description = record.Caract ristiques ?? "Unknown
                double prix = record.PrixUnitaireCAD != null ? Convert.To
                int quantite = 10; // Default quantity for each flower

                // Create a new Fleur object and add it to the list
                Fleur fleur = new Fleur(nom, couleur, description, prix,
                list_fleurs.Add(fleur);
            }
            ToJson(); // Write to JSON file
        }

    }
```

6.2 Sauvegarde en JSON

```
public void ToJson()
    {
        try
        {
            string json = JsonConvert.SerializeObject(
                list,
                Formatting.Indented,
                new JsonSerializerSettings
                {
                    TypeNameHandling = TypeNameHandling.All,   // Stores t
                    ReferenceLoopHandling = ReferenceLoopHandling.Ignore
                }
            );

            string filePath = Path.Combine(Directory.GetCurrentDirectory(
            File.WriteAllText(filePath, json);
            //Console.WriteLine($"JSON file written to {filePath}");
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

7 Conclusion

Ce projet a permis de mettre en application les principes avancés de la POO en C#, tout en manipulant des données externes avec les bibliothèques CsvHelper et Json.NET. Le code est répartí de manière modulaire pour être réutilisable dans une future interface graphique.