

**Programmation Orientée Objet II**

# INF11207 (MS)

UNIVERSIT´E DU QUEBEC A` RIMOUSKI

D´EPARTEMENT DE MATH´EMATIQUES,

D’INFORMATIQUE ET DE G´ENIE

RAPPORT

# Travail Pratique 01

**E´QUIPE :**

Feriel Batache

Amina Missoum

Emile Santerre

**PROFESSEUR :**

## Yacine Yaddaden, Ph. D.

Table des matières

**Table des matières**

[1. Introduction 3](#_Toc194610045)

[2. Description du projet 4](#_Toc194610046)

[2.1. Modélisation UML 4](#_Toc194610047)

[2.2. Fonctionnalités principales 4](#_Toc194610048)

[3 Technologies utilisées 5](#_Toc194610049)

[3.1. Environnement de développement et outils 5](#_Toc194610050)

[3.2. Langage et bibliothèques 5](#_Toc194610051)

[4 Défis rencontrés 6](#_Toc194610052)

[4.1. Adaptation des technologies et interopérabilité des outils 6](#_Toc194610053)

[4.2. Tests fonctionnels et débogage 6](#_Toc194610054)

[4.3. Gestion du temps et priorisation des tâches 6](#_Toc194610055)

[5 Conclusion 7](#_Toc194610056)

# 

# 

1 Introduction

# 1. Introduction

Dans le cadre du cours de Programmation Orientée Objet II (INF11207), le projet **FlowerShop** a été conçu pour créer une application console en C# permettant de gérer un magasin de fleurs. Cette application a pour but de fournir une solution logicielle pour la gestion de plusieurs aspects d’une boutique de fleurs, tels que la gestion des utilisateurs, la vente de fleurs et de bouquets, la gestion des commandes, la génération des factures, ainsi que la gestion d’inventaire.

Lors du développement de cette application, nous avons mis en pratique plusieurs concepts de la programmation orientée objet (POO), incluant l’héritage, les classes abstraites, ainsi que la surcharge d’opérateurs, etc. Ces concepts permettent d’assurer la modularité, la réutilisabilité et l’évolutivité du code.

De plus, le projet explore l’utilisation de bibliothèques externes pour la gestion des fichiers CSV et JSON, facilitant l’importation et la persistance des données. Grâce à cette approche, nous avons pu intégrer une architecture permettant non seulement de gérer les fleurs et les bouquets, mais aussi de suivre les commandes des clients, gérer les différents utilisateurs du système, générer automatiquement des factures et maintient l’inventaire à jour.

Ce rapport décrit en détail l'implémentation du projet **FlowerShop**, en expliquant les technologies utilisées, les défis rencontrés lors du développement, ainsi que les solutions mises en place pour y faire face. Il se termine par une analyse des apprentissages réalisés et des perspectives d’amélioration pour l’avenir.

.

2 Description du projet

# 2. Description du projet

Le projet FlowerShop a pour objectif de développer une application permettant de gérer un magasin de fleurs en offrant une gestion complète des différentes entités et processus commerciaux. L’application permet de gérer plusieurs types d’utilisateurs, chacun ayant des rôles et permissions spécifiques. Le propriétaire dispose d’un accès total pour administrer le magasin, les vendeurs gèrent les commandes et interagissent avec les clients, tandis que ces derniers peuvent consulter le catalogue de fleurs et de bouquets, passer des commandes et effectuer des paiements. Les fournisseurs, quant à eux, sont responsables de l’approvisionnement en fleurs et peuvent mettre à jour le stock selon les besoins du magasin.

L’application inclut également un module de gestion des fleurs et des bouquets. Chaque fleur est caractérisée par un nom, une couleur dominante, un prix unitaire et une description. La création de bouquets personnalisés est une fonctionnalité clé, permettant aux clients mais aussi aux vendeurs et au propriétaire, de composer des arrangements floraux sur mesure. Le prix total d’un bouquet est calculé automatiquement en additionnant les prix des fleurs qui le composent, avec l’ajout de frais fixes pour la main-d’œuvre et l’inclusion d’une carte personnalisée.

La gestion des commandes et des factures est un autre aspect central du projet. Chaque commande est associée à un client et traitée par un vendeur. Une fois validée et payée, la commande donne lieu à une facture qui récapitule le numéro de facture, les identifiant du client et du vendeur, la date d’émission et le montant total payé. Cette approche garantit une traçabilité des transactions réalisées au sein du magasin.

Le stockage des données est assuré par l’utilisation du format JSON, garantissant une persistance des informations entre les sessions d’utilisation de l’application. Les fichiers JSON permettent de sauvegarder les informations relatives aux utilisateurs, aux bouquets, aux commandes et aux factures de manière structurée et accessible. En complément, un fichier CSV est utilisé pour l’importation initiale des fleurs, facilitant ainsi l’intégration de nouvelles données dans l’application.

Enfin, une bibliothèque réutilisable dédiée à la gestion des bouquets de fleurs a été développée. Cette bibliothèque centralise les fonctionnalités liées à la création et à la manipulation des bouquets, permettant une extension future du projet vers une interface graphique qui pourra exploiter ces fonctionnalités sans nécessiter de modifications majeures du code existant.

Pour mieux comprendre le fonctionnement de chaque fonctionnalité du programme et savoir comment s’en servir, il est recommandé de se référer aux vidéos d’explication fournies.

3 Technologies utilisées3 Technologies utilisées

## **3.1. Environnement de développement et outils**

* **Microsoft Visual Studio 2022 Community** : a été utilisé comme environnement principal pour le développement du projet en **C#**, permettant un travail efficace avec les fonctionnalités intégrées pour le débogage et les tests.
* **Git & GitHub** : ont été utilisés pour la gestion des versions et pour assurer un travail collaboratif fluide, avec la possibilité de suivre et de restaurer les versions du code à chaque étape du projet.
* **NClass** : a été utilisé pour la conception du diagramme UML, permettant de modéliser les différentes entités et leurs relations au sein du système, ce qui a facilité la structure du code.

## **3.2. Langage et bibliothèques**

* **C#** : un langage orienté objet qui permet une meilleure organisation du code et l’utilisation de concepts comme l'héritage, les classes abstraites et les interfaces.
* **CsvHelper** : a été utilisé pour gérer l'importation et l'exportation des données dans des fichiers **CSV**, permettant de charger facilement les informations des fleurs disponibles et des commandes passées.
* **Json.NET** : a été utilisé pour la gestion des fichiers **JSON**, garantissant la sérialisation et la désérialisation des données afin de les sauvegarder et les restaurer à chaque lancement de l’application.

4 Défis rencontrés

# 4 Défis rencontrés

Tout au long du développement de l'application, plusieurs défis ont été rencontrés, nécessitant des ajustements techniques et une gestion minutieuse des ressources.

## **4.1. Adaptation des technologies et interopérabilité des outils**

L’un des premiers défis a été l’intégration des bibliothèques externes, notamment pour la gestion des fichiers **JSON** et **CSV**. Il a fallu adapter le code pour s’assurer que les données provenant des fichiers CSV puissent être correctement mappées aux objets du projet, comme les fleurs et les commandes. De plus, la gestion de la sérialisation et de la désérialisation des données en JSON a posé des difficultés. Il était essentiel de garantir que les objets stockés en JSON puissent être correctement restaurés à chaque exécution, ce qui a demandé une attention particulière à la gestion des structures de données et des formats.

## **4.2. Tests fonctionnels et débogage**

Le processus de **tests fonctionnels** a été crucial pour assurer la stabilité de l’application. Chaque fonctionnalité devait être testée indépendamment pour s'assurer qu’elle ne causait pas de conflits avec d'autres parties du code. Lorsqu'un problème était détecté, il était impératif de le corriger rapidement, puis de tester à nouveau l’ensemble des fonctionnalités pour garantir que les modifications n’introduisaient pas de bugs. Un autre défi lié à cette étape a été le processus de **débogage** pour résoudre les problèmes complexes qui se sont présentés, en particulier lors de l’intégration des fichiers CSV et JSON.

## **4.3. Gestion du temps et priorisation des tâches**

La gestion du temps et l’organisation des tâches ont également constitué un défi majeur. Le développement de l’application a nécessité une planification minutieuse pour prioriser les fonctionnalités essentielles, comme la gestion des fleurs, des commandes et des utilisateurs, avant d’ajouter des améliorations supplémentaires. La planification des tests et des validations a aussi été un point clé pour éviter des erreurs de dernière minute qui auraient pu compromettre la qualité et le délai de livraison du projet.

5 Conclusion

# 5 Conclusion

Ce projet nous a permis de développer une application complète et fonctionnelle, intégrant plusieurs des concepts de la programmation orientée objet en C#. Nous avons non seulement appris à structurer une application de façon claire et modulaire, mais aussi à gérer des données externes à travers les formats JSON et CSV. L'utilisation de ceux-ci a facilité la gestion et la persistance des données, rendant l’application robuste et évolutive.

Nous avons également développé une vraie collaboration de groupe, en utilisant Git pour suivre notre progression et en partageant les tâches efficacement.

Dans une perspective d’amélioration, l’ajout d’une interface graphique utilisant Windows Forms ou WPF serait une évolution naturelle du projet. Une autre amélioration possible serait d’intégrer une base de données relationnelle pour optimiser la gestion des informations stockées. En somme, ce projet a été une expérience illustrant la complexité et les défis réels du développement logiciel.