|  |
| --- |
| Université du Québec à Rimouski |
| Travail Pratique I |
| INF11207 - Programmation orientée objet II  Professeur – Yacine Yaddaden, Ph. D. |

|  |
| --- |
| ⭢ Boutin, Frédérik  ⭢ Létourneau, Gabriel  ⭢ Milliard, Alex |

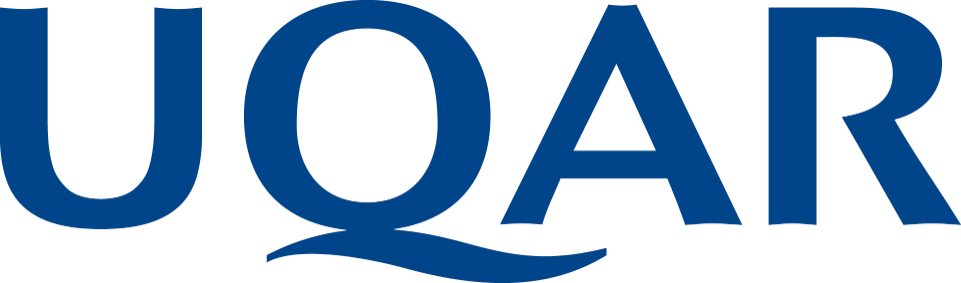


Table des matières

[Introduction 2](#_Toc102067608)

[Page d’accueil 2](#_Toc102067609)

[Nouvel utilisateur 2](#_Toc102067610)

[Fenêtre principale 2](#_Toc102067611)

[Onglet Prédiction 2](#_Toc102067612)

[Onglet Performances 3](#_Toc102067613)

[Onglet Paramètres 3](#_Toc102067614)

[Onglet Historique 3](#_Toc102067615)

[Onglet Mise à Jour (MAJ) Utilisateur 3](#_Toc102067616)

[Problèmes et Difficultés rencontrées 3](#_Toc102067617)

[Conclusion 4](#_Toc102067618)

[Références 4](#_Toc102067619)

# Introduction

Dans ce travail pratique, l’objectif était de se familiariser avec la création d’interfaces graphiques en WPF, la notion d’architecture MVVM, ainsi que l’utilisation d’ORM pour la communication avec la base de données. L’architecture MVVM a pour but de séparer la partie donnée de la partie vue d’une interface, par l’intermédiaire de View Models. Pour ce faire, nous devions utiliser le programme fait dans le travail pratique 1 et permettant d’obtenir une valeur de qualité de vin en fonction de quartes paramètres d’un échantillon. Nous devions lui ajouter une interface graphique permettant de faire le lien entre chaque utilisateur et ses données, via une base de données MySQL.

Cette interface graphique se présente sous la forme de trois fenêtres WPF, soit la première fenêtre permettant à l’utilisateur de se connecter à un compte, la deuxième fenêtre pour se créer un compte s’il n’en possède pas et la troisième fenêtre permettant d’entrer les fichiers et les informations utiles pour obtenir la qualité du vin.

# Page d’accueil

Comme mentionné en introduction, un utilisateur peut sélectionner son profil dans la boîte de commande de cette fenêtre. Dans le cas où il n’en possède pas, le bouton « Créer » lui permettra d’accéder à une deuxième fenêtre pour la création d’un compte. Une fois son compte sélectionné via son nom complet, il sera redirigé vers la fenêtre principale en appuyant sur le bouton « Connecter ». La mise à jour des noms d’utilisateurs se fait de manière dynamique, à mesure qu’il y a création de compte, de part une liaison à un objet Users dans le ViewModel. Nous avons utilisé la propriété IteamTemplate et DataTemplate dans la ComboBox de la fenêtre afin d’afficher uniquement le prénom et le nom d’utilisateur avec une liaison directement sur les deux mutateurs du même objet Users du ViewModel.

# Nouvel utilisateur

Cette page est accédée à la suite d’un clique sur le bouton ajouter et est obligatoire en l’éventualité où aucun utilisateur n’existe au sein de la base de données liées à l’application. Sur cette page, l’ensemble des informations requises pour la création d’un utilisateur (classe « Users » au sein de l’application) sont demandées et requises pour la création. Une fois les informations remplies, toutes ces informations sont stockées au sein de la base de données et ID utilisateur est aussi automatiquement créé et attribué au nouvel utilisateur. La création de l’utilisateur ferme aussi cette page et rouvre la page d’accueil afin de permettre de continuer d’utiliser l’application normalement.

# Fenêtre principale

## Onglet Prédiction

Au moment de l’accès à cette page, l’utilisateur courant est sauvegardé comme une variable accessible dans l’application au complet afin d’être en mesure d’effectuer diverses étapes à venir. Cet onglet demande tout simplement à l’utilisateur d’entrer les 4 valeurs liées à son vin dans les champs textes demandés. L’ensemble de ces champs contiennent un délai d’une seconde pour permettre à l’utilisateur d’entrer des chiffres après la virgule au besoin avant que l’application les entre en variable. Cependant, même si l’utilisateur remplit les quatre champs, le bouton permettant d’effectuer une prédiction ne sera pas cliquable.

Pour continuer, l’utilisateur se doit aussi de remplir tous les champs de l’onglet Paramètres (voir la section associée ci-dessous pour de plus amples détails sur cet onglet). À la suite de cette étape, le bouton prédiction devient accessible. Au moment du clique, ce bouton fait appel à la librairie KNN pour effectuer le calcul et affiche une image à côté de ce dernier selon la qualité évaluée. Ce bouton met aussi l’onglet historique à jour afin d’ajouter la plus récente prédiction associée à l’utilisateur courant en faisant appel à la même fonction utilisée pour générer les données trouvées à cet endroit.

## Onglet Performances

Cet onglet permet d’évaluer la performance de l’algorithme KNN dans les prédictions. Pour effectuer cette évaluation, il est premièrement nécessaire de remplir tous les champs de l’onglet paramètres. Il suffit ensuite d’importer un fichier test en utilisant le bouton approprié. À la sélection de ce fichier, le pourcentage de précision sera affiché ainsi que la matrice de confusion liée à cette évaluation de performance. Cette matrice est affichée au sein d’un datagrid qui est liée à une classe intitulée « LigneMatriceConfusion » qui permet de gérer cet affichage, le choix de passer ainsi a été réalisé suite à une rencontre avec le professeur.

## Onglet Paramètres

Dans cet onglet, l’utilisateur doit premièrement importer le fichier d’entraînement pour l’algorithme en appuyant sur le bouton ouvrir qui lui ouvrira une fenêtre permettant de parcourir les fichiers de l’ordinateur et de trouver le fichier approprié. À l’ouverture du fichier, un datagrid au centre de la page se remplit avec les données qui ont été importées du fichier choisit. Finalement, l’utilisateur doit choisir la valeur k (mise par défaut à 1) et sélectionné l’algorithme de tri (aucune valeur par défaut afin de forcer le choix). La réalisation de ces étapes sont nécessaires pour l’onglet prédiction ainsi que l’onglet performances (tel que mentionné précédemment).

## Onglet Historique

Cet onglet n’est qu’un énorme datagrid permettant d’afficher l’ensemble des prédictions réalisées par l’utilisateur courant (en utilisant la variable globale mentionnée précédemment), ces données sont tirées directement de la base de données en triant par l’identifiant unique de l’utilisateur. L’ensemble de ces données sont mises à jour chaque fois que l’utilisateur réalise une nouvelle prédiction afin de l’ajouter aux données présentées ici.

## Onglet Mise à Jour (MAJ) Utilisateur

Cet onglet est essentiellement un clone de la page nouvel utilisateur à la différence prêt que les différents champs sont remplis avec les informations de l’utilisateur courant en utilisant à nouveau la variable globale discutée précédemment. L’ensemble des champs sont modifiables et lorsque l’utilisateur appuie sur modifier l’ensemble des données sont mises à jour au sein de la base de données associée à l’application. Une boîte de message apparaît aussi à ce moment pour indiquer que la mise à jour a été effectué avec succès.

# Problèmes et Difficultés rencontrées

Le premier problème majeur rencontré fut la difficulté d’effectuer un binding sur les boutons de type radio pour le choix du sexe dans la page de création d’un nouvel utilisateur. Initialement, nous avons trouvé une solution en ajoutant des attributs pour chaque sexe dans la classe Users (voir code en commentaire). Quoique cette solution était fonctionnelle, elle résultait en du code peu satisfaisant et mal optimisé. Après avoir consulté le professeur, une meilleure solution qui utilise un convertisseur a été expliquée et démontrée, c’est cette solution qui a été utilisée dans le cadre du projet.

Un autre problème rencontré, était l’image résultante de l’opération de l’onglet prédiction. À nouveau ici, nous avons consulté le professeur qui nous a fait créer un attribut au sein de la classe prédictions qui ne sera pas ajouté à la base de données. Cet attribut est le chemin vers une des trois images qui sont présentes au sein du dossier views et change selon des conditions if/else dépendantes du résultat de la prédiction.

Finalement un autre problème majeur rencontré fut le binding des données liées à l’onglet performances. Ces données sont liées à un élément de type dictionary dans la bibliothèque fournit et nous éprouvions des difficultés à les utiliser. Après une consultation avec le professeur, la solution fut de créer une classe liée directement à ces données et de créer une « observable collection » contenant trois listes des données de la matrice de confusion.

# Conclusion

À la vue de notre interface graphique, il est possible de voir qu’elle respecte toutes les exigences du travail pratique. En effet, le résultat final présente 3 fenêtres permettant à un utilisateur d’effectuer la création d’un compte, la connexion à celui-ci afin d’utiliser le programme de mesure de qualité du vin. Ce même utilisateur doit seulement entrer son fichier comparatif, les informations sur son ou ses vin(s) (prend la forme d’un fichier .csv pour plusieurs vins ou quatre paramètres pour un seul échantillon) ainsi que les paramètres k et le choix d’algorithme de tri. Finalement, nous respectons le modèle MVVM, car aucune vue ne communique directement avec nos modèles. Nous assurons également la persistance des données par la communication avec la base de données MySQL.

# Références

1. Yaddaden, Yacine *INF-11207 – séance 13 – Interface graphique – introduction au design pattern MVVM* (**2022)** Université du Québec à Rimouski
2. Yaddaden, Yacine *INF-11207 – séance 14 – Interface graphique – initiation à l’ORM Entity Framework* (**2022)** Université du Québec à Rimouski