Université du Québec à Rimouski

Travail Pratique I

8INF957 - Programmation objet avancée

Professeur – Yacine Yaddaden, Ph. D.

Application pour Médecin et diagnostique par IA

→ Mamadou Mouslim Diallo
→ Benjamin Lapointe-Pinel
→ FateMeh Bashardoustjoubjarkouli

TABLE DES MATIÈRES

INTRODUCTION

Ce travail a été réalisé dans le cadre des TPs dans le cours de Programmation Orientée Objet Avancée. Le travail consiste à concevoir une application de bureau destinée aux médecins qui est permettra aux médecins de faire des diagnostiques par l'intelligence artificielle à l'aide de l'algorithme **KNN**(K-Nearest Neighbors).

Pour la réalisation de notre projet conformément à l'énoncé, nous avons utilisé plusieurs méthodes apprises en cours notamment la technique de la **programmation orientée objet** pour définir les modèles de classes dans notre code, la conception des interfaces graphiques avec le **Framework WPF** (Windows Presentation Fundation) combiné à l'algorithmique des plus proches voisins KNN(K-Nearest Neighbors) pour la réalisation des résultats souhaités.

De ce fait, les technologies utilisées sont **Visual Studio 2022 Community** qui facilite grandement la performance de notre code en l'analysant et en le simplifiant ainsi que la librairie externe **CSVHelper** que nous avons installée dans les *Tools* avec le gestionnaire *Nuget* pour faciliter l'upload de nos fichiers de tests et de références.

En plus nous avons opté pour le design pattern **MVVM** afin d'organiser les éléments de notre logiciel avec chacun son rôle. Cela a été fondamental pour structurer notre programme. Voir la capture.

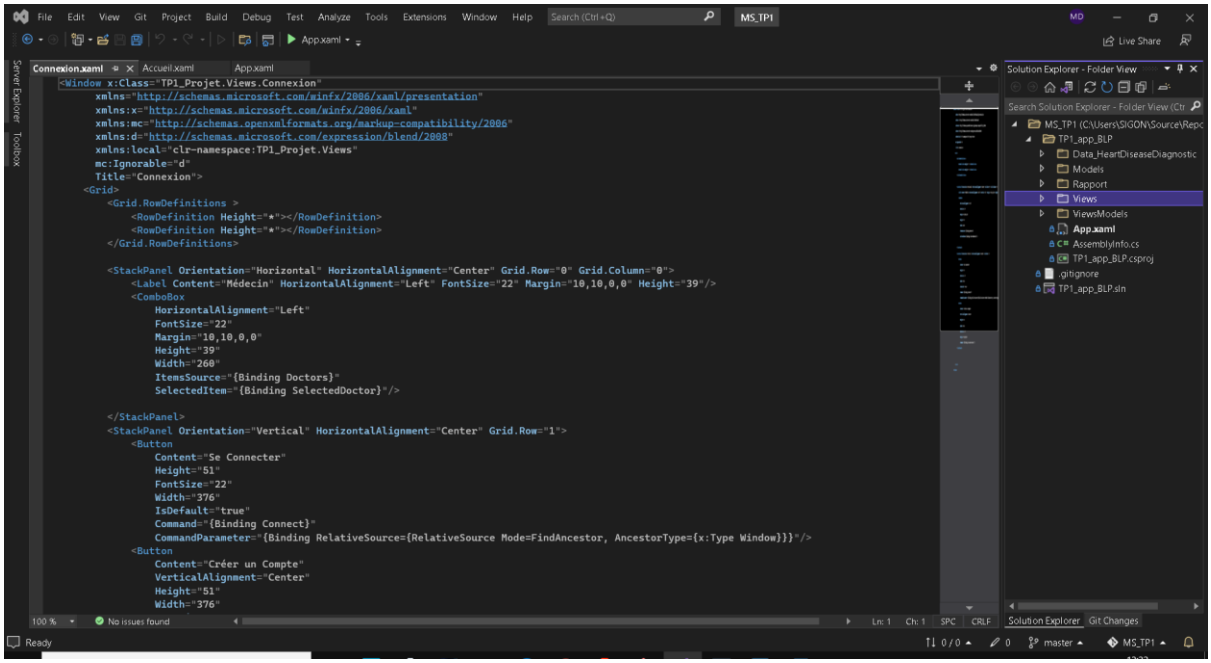

Figure 1 - Structure de notre programme avec MVVM

Si vous voulez apprendre à coder, le meilleur moyen c'est d'écrire le plus possible de code. C'est ce que nous tenter de faire ici en utilisant l'IDE VS Community qui est un outil puissant

qui intègre plusieurs langages POO notamment en Csharp où des milliers d'applications roulent.

Le langage de programmation C# permet de créer de nombreux types d'applications, par exemple :

- Applications métiers pour capturer, analyser et traiter les données
- Applications web dynamiques accessibles à partir d'un navigateur web
- Jeux 2D et 3D
- Applications financières et scientifiques
- Applications cloud
- Applications mobiles

Pour revenir à notre projet, l'objectif dans un premier temps est de pouvoir créer le profil d'un médecin qui pourra par la suite se connecter et ensuite les modifier ses informations. Nous avons créé un modèle de classe pour le médecin et l'interface associée.

Par la suite, nous nous sommes intéressés à la page d'accueil une fois la connexion effectuée. Ce second volet consiste à naviguer entre les 3 onglets *Informations*, *Diagnostique* et *Configuration IA.* Pour chaque cas, un contrôle utilisateur a été associé.

L'onglet *Informations* affiche les données du médecin qui sont modifiables. L'onglet *Diagnostique* contient un bouton où nous avons la possibilité d'ajouter un patient et voir ses informations.

Ce sont ses informations qui sont entre autres des valeurs, du texte, des combobox qui seront ajoutées pour faire le diagnostic. L'objectif est d'appliquer le KNN pour avoir le résultat.

Sur l'onglet *Configuration IA,* nous avons travaillé sur les paramètres KNN ainsi que les boutons d'importations des fichiers ***train.csv*** et ***test.csv*** pour la classification et l'évaluation. Nous utilisons une technique d'apprentissage supervisé.

Pour ceux qui ont plongé la main dedans, l'intelligence artificielle ou le modèle de machine Learning a pour but de faire des prédictions.

Nous avons implémenté KNN combiné avec un algorithme de tri appelé tri Shell pour trouver le taux de reconnaissance. Cela a été possible en appliquant une évaluation de performance de l'algorithme en utilisant les données de références et d'évaluation.

MODÉLISATION UML

Dans la conception de notre application, nous avons ajouté d'autres modèles de classes ainsi que leurs propriétés. Nous avons ainsi appliqué pour certaines, un système d'héritage et d'association.

Nous avons implémenté la Classe ***Person*** avec son constructeur pour afficher les données comme le nom et le prénom du médecin ou du patient en faisant un héritage avec les classes ***Doctor*** et ***Patient***.

L'implémentation de la classe ***Diagnostic*** qui hérite de l'interface ***IDiagnostic***. Dans le constructeur, nous avons instancié ***Features*** comme float où nous avons stocké des valeurs en déclarant dans le code le *type de douleur, le nombre de gros vaisseaux, la dépression et la thalassémie.*

Dans notre ViewModel qui assure la liaison avec la Vue, plusieurs classes ont été créées comme vous le constatez dans le diagramme. ***DoctorEditorViewModel*** est une classe mère de ***AccueilViewModel*** qui assure l'interaction entre différents onglets ainsi que les actions effectuées dans les différentes fenêtres.

De l'autre, ***PatientViewModel*** et ***DoctorEditorViewModel*** sont deux autres classes qui contiennent les données du médecin et du patient. Elles héritent de INotifyPropertyChanged et permettent la liaison des interfaces.

Pour la connexion, nous avons lié ***ConnexionViewModel*** à l'interface de connexion qui déclenche l'action de créer un compte et de pouvoir se connecter.

Pour résumer, dans la modélisation UML de notre projet, voici les entités et classes qui entrent en jeu.

Nous avons :

1. La classe ***Doctor***
2. La classe ***Patient***
3. La classe ***Person***
4. La classe ***Diagnostic*** qui implémente ***IDiagnostic***
5. La classe ***KNN*** qui implémente ***IKNN***
6. La classe AccueilViewModel
7. La classe DoctorEditorViewModel
8. La classe PatientViewModel
9. La classe ConnexionViewModel

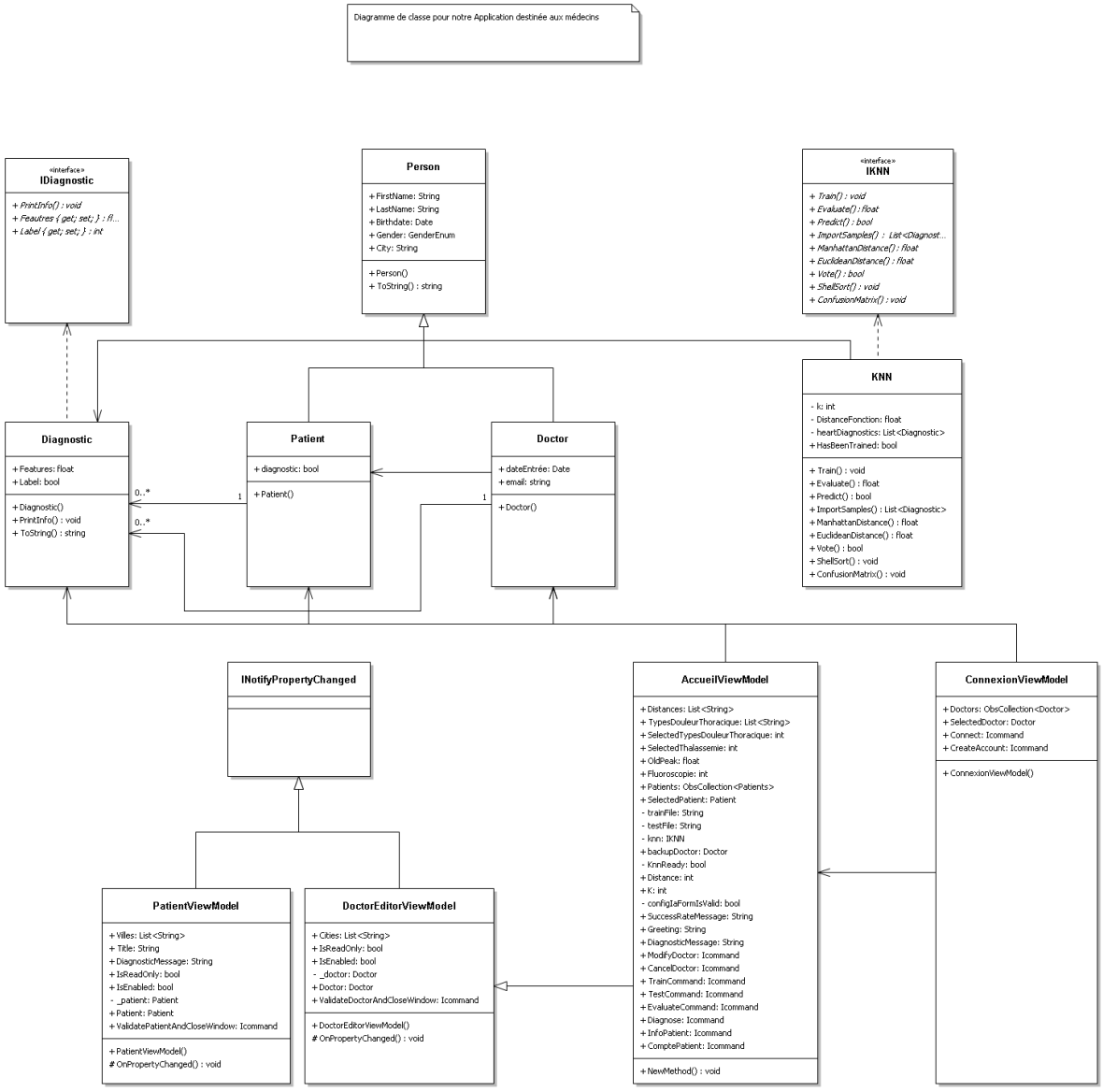

Figure 2 - Capture du diagramme de classe du projet

APERÇU DE L'APPLICATION (GUI)

Notre application destinée aux médecins dispose de plusieurs interfaces graphiques interconnectées conformément à notre énoncé. Nous avons ajouté au fur et à mesure d'autres interfaces selon le besoin.

Voyons la description :

INTERFACE DE CONNEXION

C'est la première fenêtre qui s'affiche lors de l'exécution du logiciel. Cette interface a été créée dans *Views* et organisée grâce au **StackPanel** et au **Grid.** Nous avons créé la première

interface graphique WPF où nous avons la possibilité d'ajouter le profil du médecin afin qu'il puisse se connecter en tant que médecin et de pouvoir ensuite modifier ses informations.

Cette interface est bindée avec la fenêtre *Créer un compte* et l'interface *Accueil* regroupant les 3 onglets. Voici la capture.

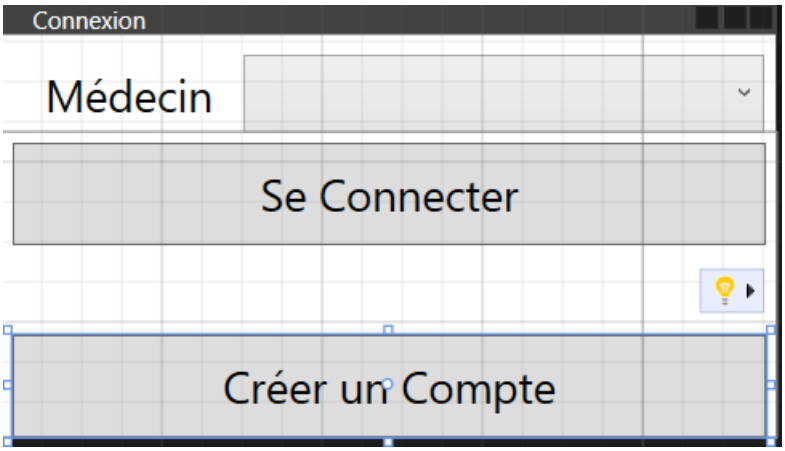

Figure 3 - Interface de Connexion

Sur nos deux boutons, nous avons fait un binding à deux variables (Connect & CreateAccount) dans le code que nous avons créé dans le ViewModel.

Le premier aspect consiste à créer un compte (le profil du médecin) afin de remplir ses informations. Puis, vous aurez la possibilité d'ajouter le médecin puis vous connecter.

Voici un extrait du code.

```
public Doctor SelectedDoctor { get; set; }

1 reference
public ICommand Connect { get; private set; }
1 reference
public ICommand CreateAccount { get; private set; }

1 reference
public ConnexionViewModel(IEnumerable<Doctor> doctors)
{
    Doctors.AddRange(doctors);
    SelectedDoctor = Doctors[0];

    Connect = new RelayCommand<Window>(window =>
    {
        var accueil = new Accueil(SelectedDoctor);
        accueil.Show();
        window.Close();
    });

    CreateAccount = new RelayCommand(() =>
    {
        var createAccount = new CreateAccount();
        bool? result = createAccount.ShowDialog();
        if (result.HasValue && result.Value)
        {
            Doctors.Add(createAccount.doctorEditorViewModel.Doctor);
        }
    });
}
}
```

Figure 4 - extrait code pour la connexion

Une fois connecté, vous allez sur l'interface Accueil.

L'INTERFACE ACCUEIL.

Elle représente l'interface de « Bienvenue » où nous avons trois 3 onglets de navigation : *informations, diagnostique, configuration IA. Voici la capture.*

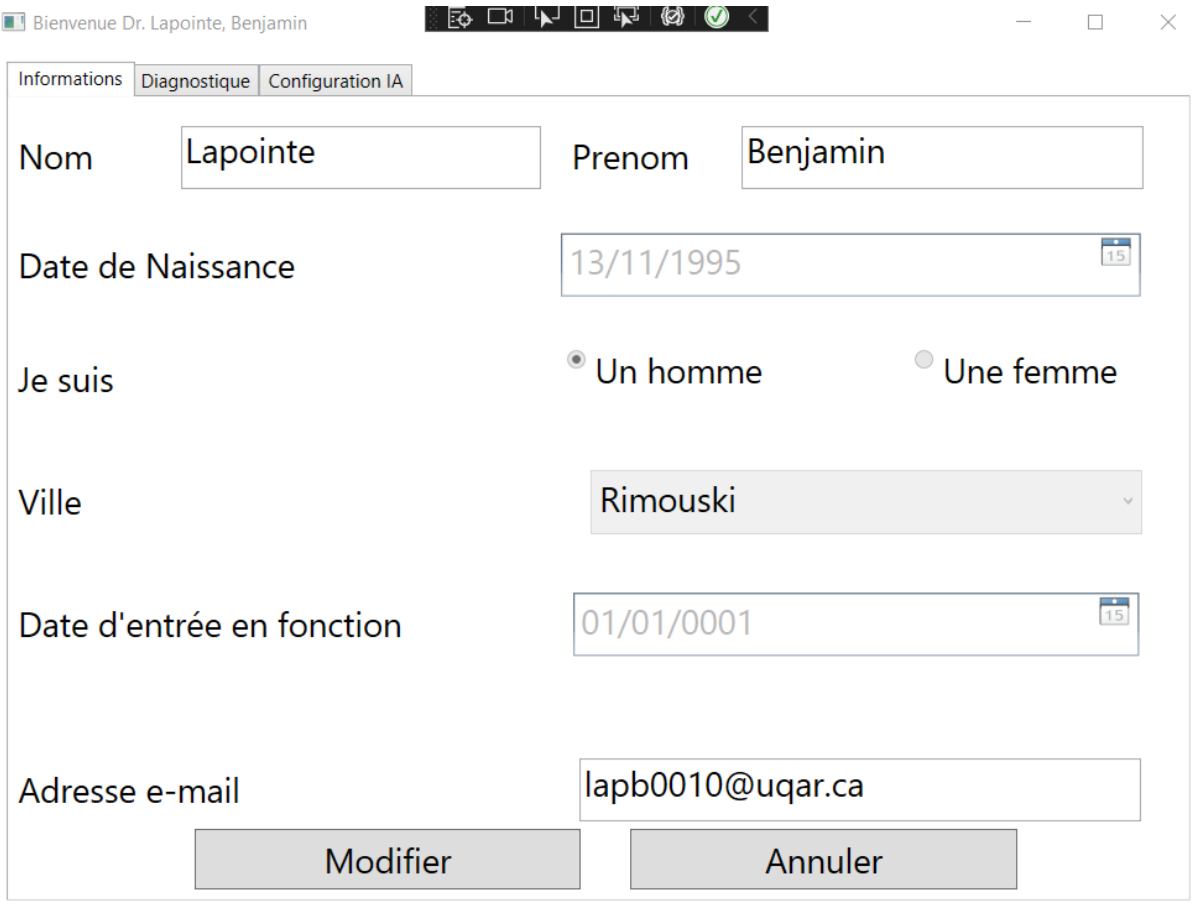

Figure 5 – Fenêtre Accueil

Dans le système de navigation entre les onglets et la liaison avec les interfaces correspondantes, pour ne pas nous répéter, nous avons utilisé des UserControls auxquels nous faisons appel pour la synchronisation des données.

Sur l'onglet ***Informations*** nous avons les informations du médecin où nous avons la possibilité de *modifier* ou *annuler*.

Sur l'onglet ***Diagnostique*** où nous avons des boutons pour trouver les informations sur le patient, ajouter un patient sans pouvoir modifier ses informations, définir le type de douleur thoracique, Thalassémie dans des Combobox. Ce sont ses informations (valeurs, texte) qui seront ajoutées pour que le médecin puisse faire son diagnostic.

On a aussi implémenté les zones de saisies du nombre de dépressions ST induites et le nombre de gros vaisseaux. Voir la capture.

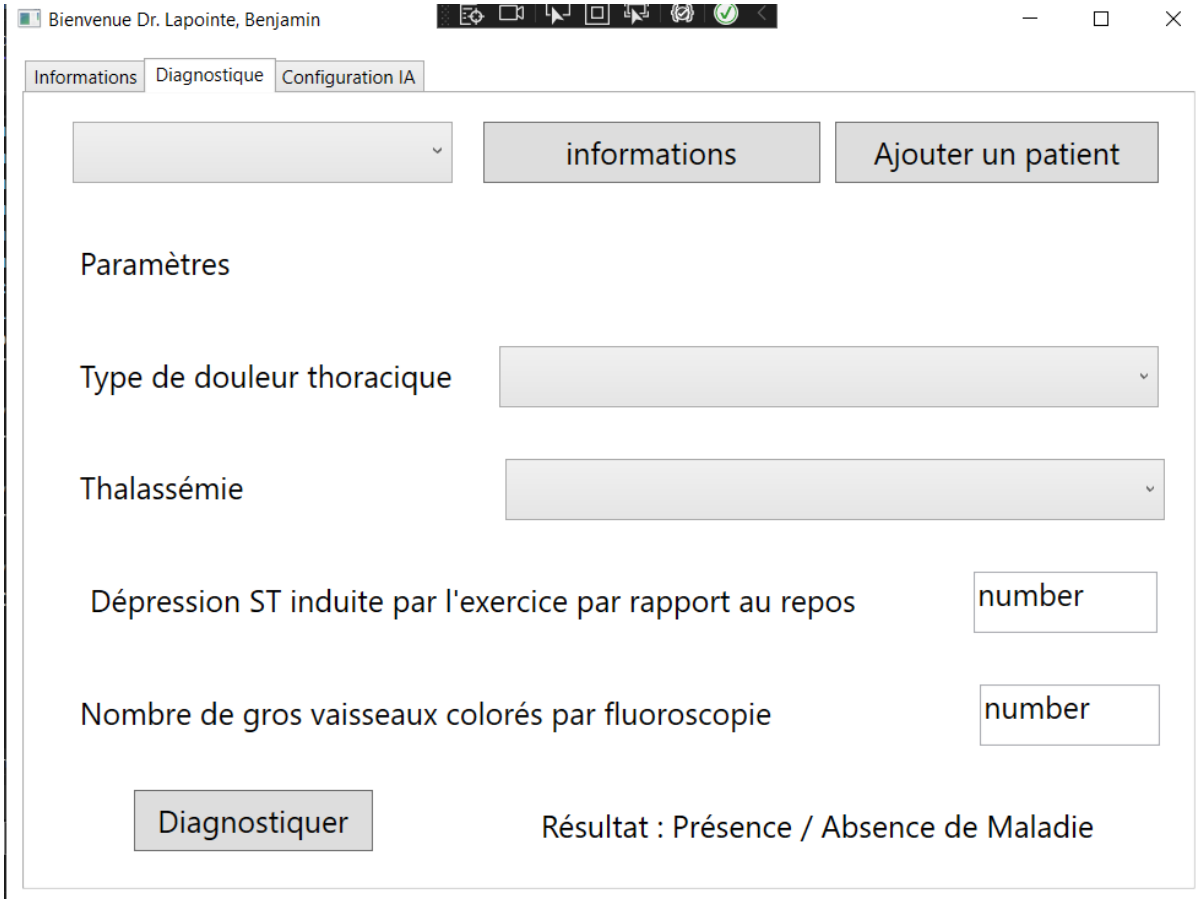

Figure 6 - Onglet diagnostique

Sur l'onglet ***Configuration IA*** où nous avons deux boutons pour ajouter nos deux fichiers ***train.csv*** et ***test.csv.*** Il suffit de cliquer sur Parcourir et les charger depuis votre ordinateur. Aussi les paramètres K et le type de distance soient *Manhattan* ou *Euclidienne*.

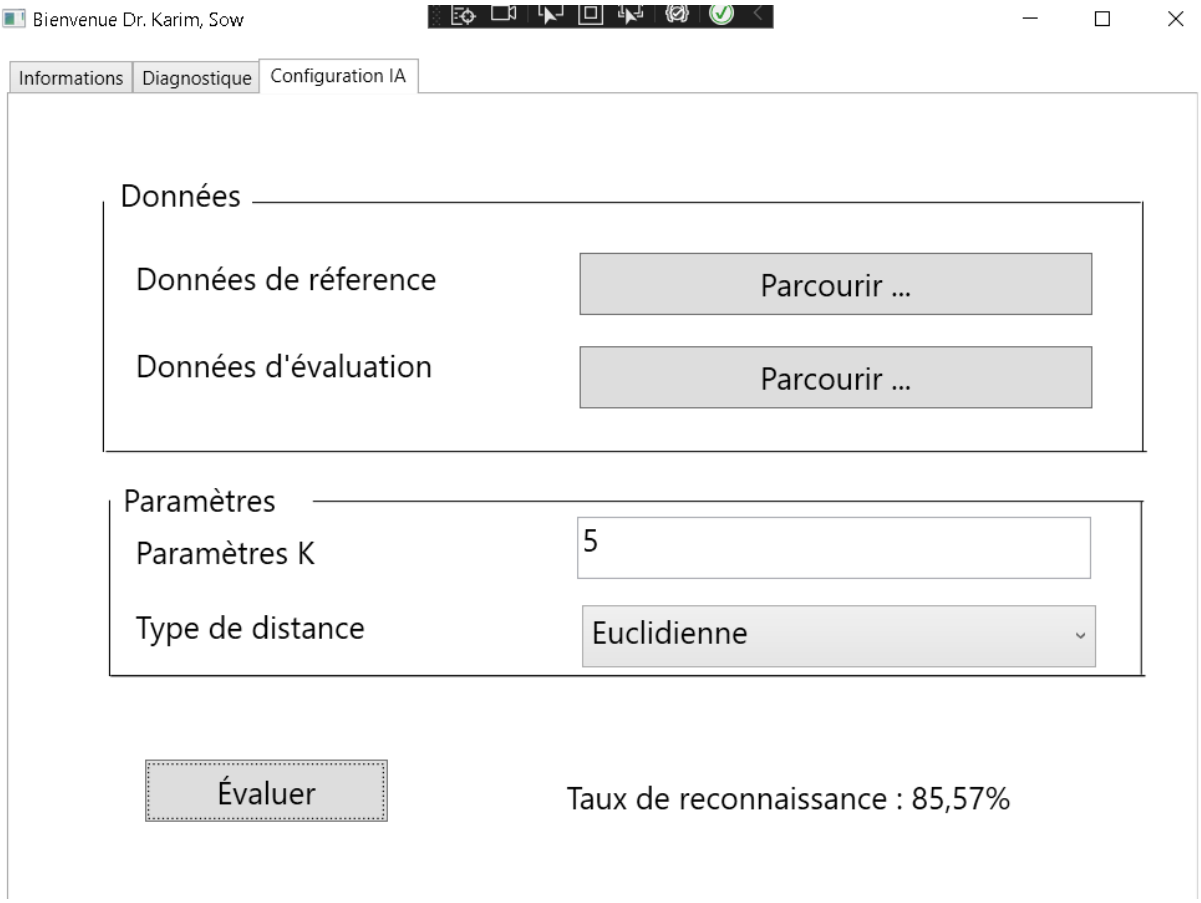

Figure 7 - Onglet Configuration IA

Une fois que les données sont entrées et que la distance est choisie, vous pouvez évaluer le taux de reconnaissance en cliquant sur le bouton *Evaluer*. Cela va faire appel aux données qui se trouvent dans AccueilViewModel pour trouver le résultat de l'algorithme en faisant de la classification. Ça donne le résultat du taux de reconnaissance en pourcentage.

INTERFACES POUR LE PATIENT

Après le travail sur Médecin, nous sommes occupés du cas du patient. En utilisant presque la même procédure, nous avons créé des interfaces pour le patient, des modèles de classe, et la liaison à l'aide des classes dans le ViewModel.

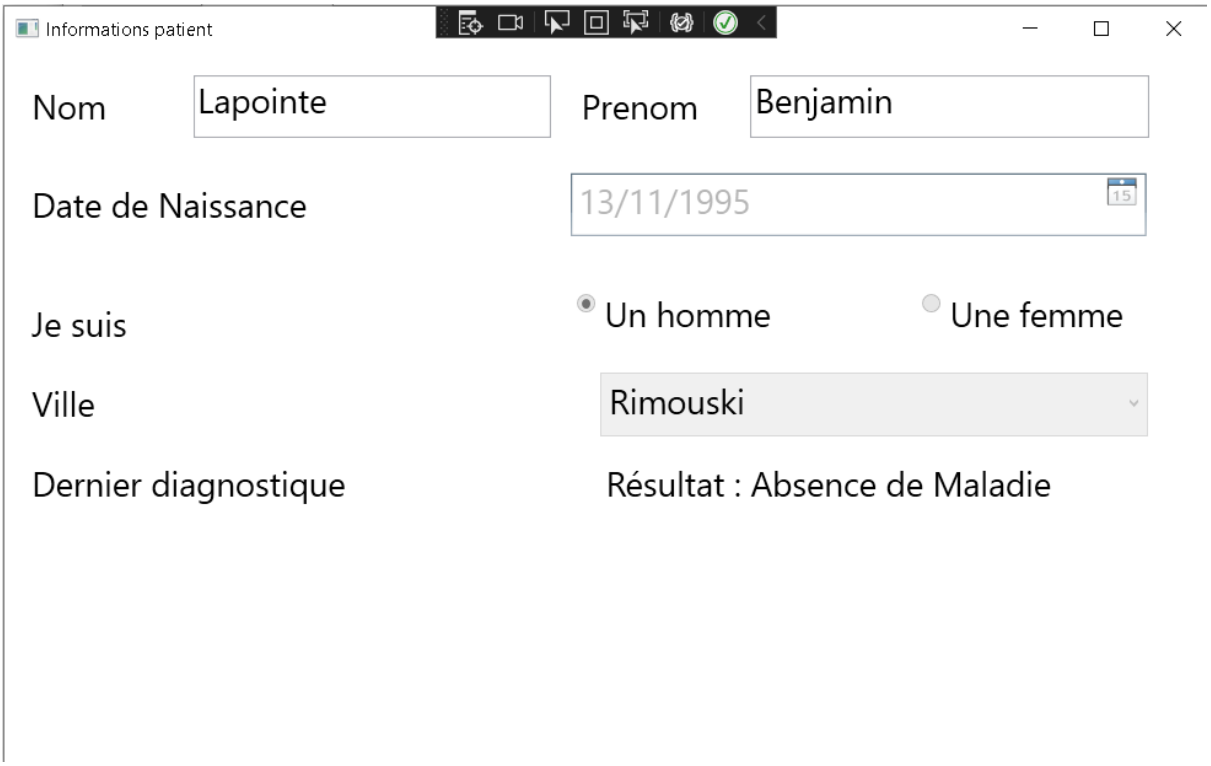

Figure 8 - Interface Informations du patient

Nous avons implémenté le bouton ***Informations*** pour afficher les informations du patient, et le bouton ***Ajouter un patient*** pour créer un compte pour le patient et ainsi pouvoir faire le diagnostic en affichant le résultat dans la fenêtre.

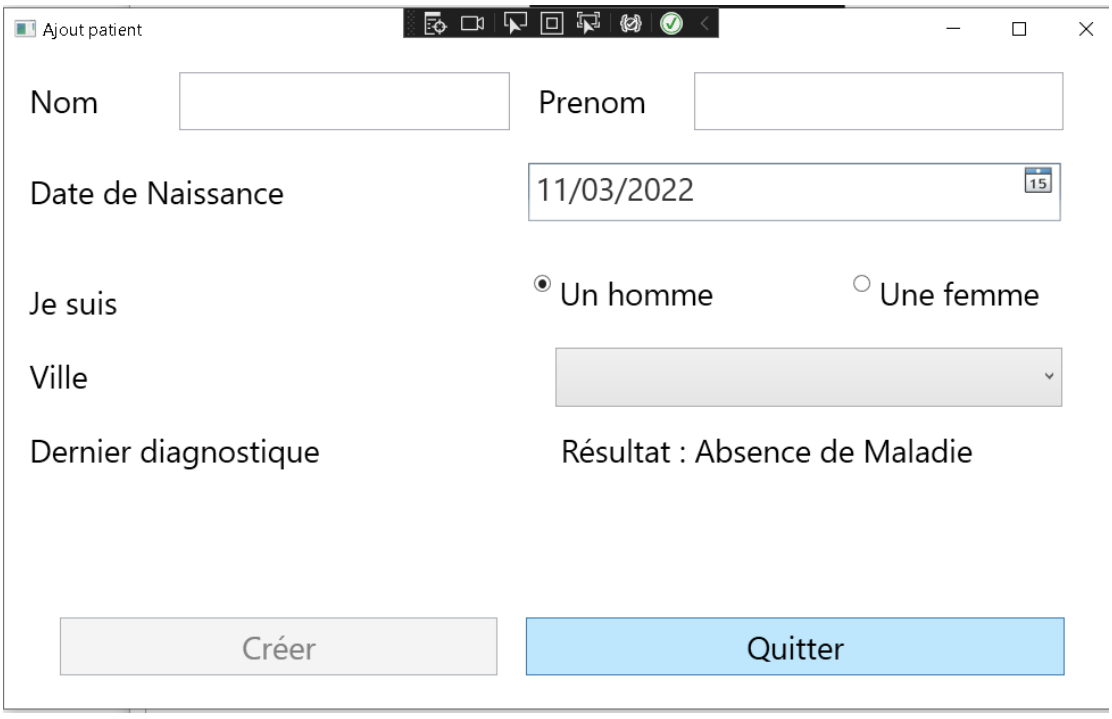

Figure 9 - Interface Ajouter Patient

ALGORITHME DU k-NN ET RÉSULTATS

L'implémentation de l'algorithme KNN connu sous le nom de K-Nearest Neighbors – les plus proches voisins – a connu plusieurs étapes.

KNN est un algorithme qui utilise une technique simple et a pour objectif de faire des prédictions. Pour se faire, on utilise des échantillons déjà étiquetés provenant du fichier *train.csv.*

Pour faire une prédiction d'un échantillon, on trouve ses K plus proches voisins (d'où le nom), puis on regarde quelle est la majorité des étiquettes voisins. La majorité déterminera la prédiction de l'étiquette de l'échantillon en question.

Voici les détails d'implémentation pour un échantillon.

IDENTIFIER TOUTES LES DISTANCES DEPUIS L'ÉCHANTILLON

Dans un ensemble d'apprentissages de plusieurs dimensions, on utilise une formule pour calculer la distance de tous les voisins. On peut utiliser la distance euclidienne, la distance la plus courte entre deux points :

$$\text{euclidienne}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

Ou la distance de Manhattan, qui décrit plutôt la distance qu'un conducteur de taxi emprunterait dans une ville quadrillée :

$$\text{manhattan}(p,q) = \sqrt{|p_1 - q_1| + |p_2 - q_2| + \cdots + |p_n - q_n|}$$

Dans les deux cas, l'indice de p et q représente les différents symptômes d'un diagnostic.

TROUVER LES K PLUS PROCHES VOISINS

Pour trouver les K plus proches voisins dans cette liste de distance, il suffit de les trier en ordre, et de sélectionner les k premiers dans la liste. Dans ce travail, le tri Shell a été utilisé. Le tri Shell est une optimisation du tri insertion.

Le tri par insertion itère sur chaque élément du début vers la fin. Pour chaque élément, il va trouver sa place vers le début, décalant ainsi tous les autres vers la fin.

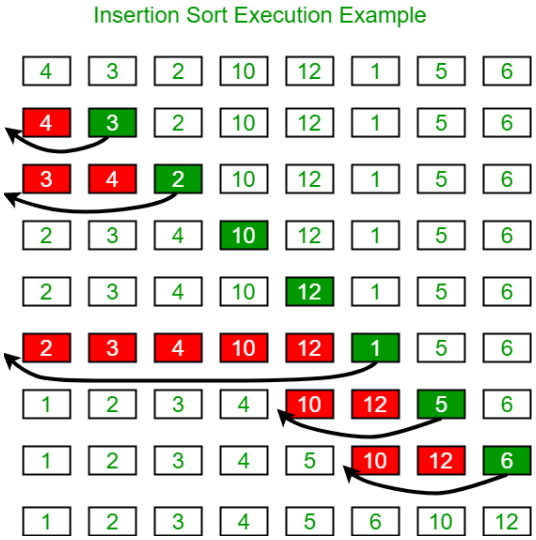

Figure 10 – Tri par insertion

Le tri Shell quant à lui fait aussi un tri par insertion, mais sur un sous-tableau qui saute tous les n éléments. Le tableau est ainsi retrié plusieurs fois, et n diminue à chaque fois, selon différentes fonctions, jusqu'à ce qu'il soit arrivé à 1, pour un tri par insertion classique final. Ainsi, la liste serait déjà presque triée lors de cette dernière étape.

Il existe plusieurs fonctions pour faire diminuer n, mais nous utilisons la fonction originale de Shell:

$$n = \left[\frac{N}{2^k}\right]$$

Où N est la taille du tableau, et k l'itération actuelle.

Une fois le tableau trié, on a choisi les k plus près, et on peut voir quel est la majorité du diagnostic, et ainsi, faire une prédiction.

Dans le cadre de notre application, avec un ensemble d'apprentissages train.csv, des données d'évaluation test.csv, un **k=6**, et une distance euclidienne, on arrive à une précision de diagnostic de **84.54%. Voir capture.**

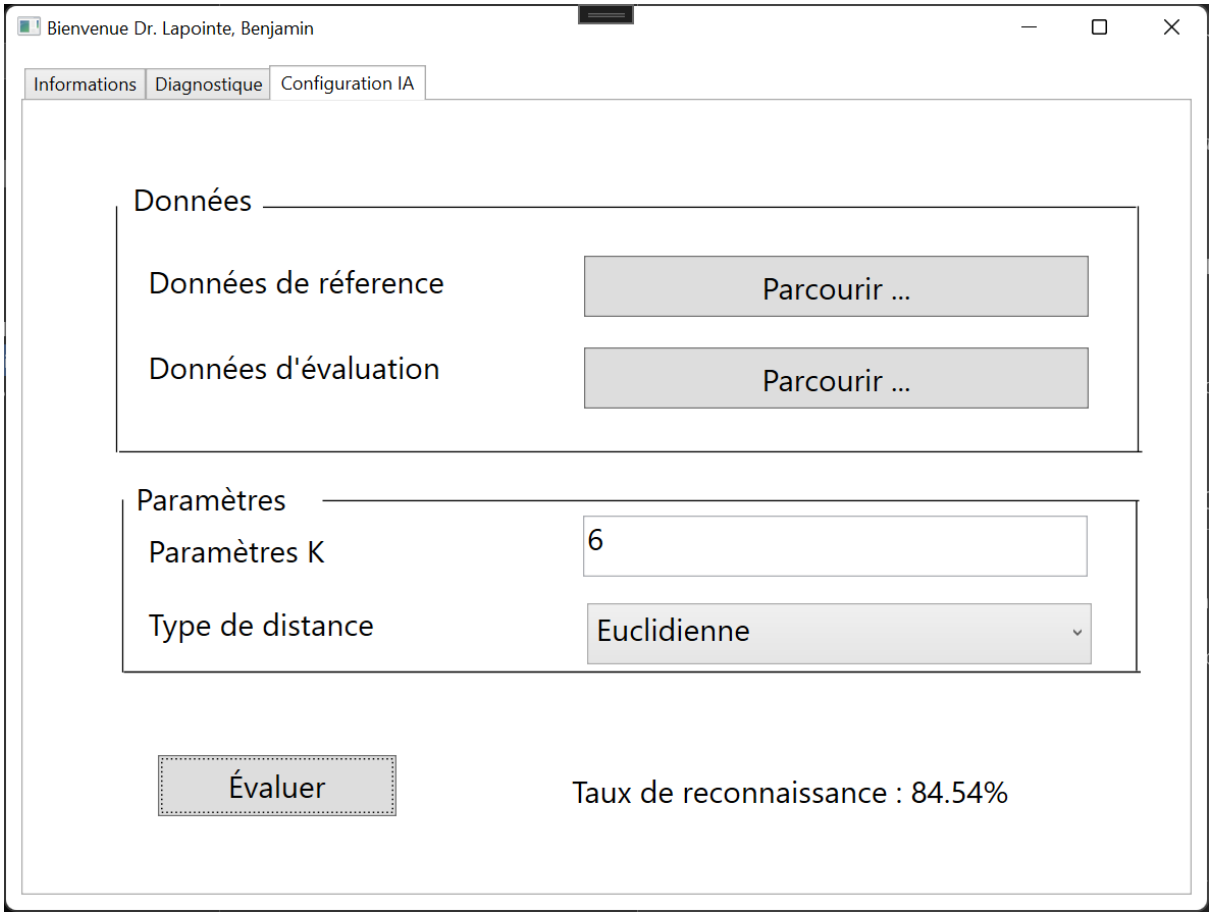

Figure 11 - Résultat Évaluation KNN

PROBLÈMES ET DIFFICULTÉS RENCONTRÉES

Comme tout autre projet, dans la réalisation de notre projet (TP1), nous nous sommes confrontés à certains problèmes, c'est le cas de :

- L'interconnexion des interfaces avec le binding et faire le survol d'une fenêtre à une autre (au début) ;
- Plus ou moins de difficulté à communiquer de vive voix (chacun travaillait et faisait son commit sur le serveur) ;

Mais en dépit de toutes les difficultés, les occupations, nous avons pu nous dépasser pour aboutir à ce résultat. La documentation sur internet en a largement contribué.

CONCLUSION

Dans ce projet, nous avons pu analyser l'énoncé, posé nos questions, implémenté au fur et à mesure dès le début pour arrive à ce résultat.

Nous avons pu implémenter KNN avec tri Shell, créé des modèles de classes correspondantes, relié nos interfaces en utilisant ces technologies citées ci haut.

Nous remercions, M. Yacine, pour ce TP1 qui nous fait largement découvrir le contour de ce langage ainsi l'outil WPF pour la conception des interfaces graphiques qui demandent beaucoup de soins et de concentration.

MERCI !

RÉFÉRENCES

CsvHelper: https://joshclose.github.io/CsvHelp

https://en.wikipedia.org/wiki/Euclidean_distance

https://fr.wikipedia.org/wiki/Distance_de_Manhattan

https://fr.wikipedia.org/wiki/Tri_par_insertion

https://fr.wikipedia.org/wiki/Tri_de_Shell

https://www.geeksforgeeks.org/insertion-sort/