Projet Plot That Line

Mateen Salem Khalil
22/10/2024

Table des matières

Analyse préliminaire

1. Introduction

Ce projet a pour but de développer une application permettant aux utilisateurs d'analyser des données de jeu provenant de deux sources principales : le jeu League of Legends (LoL), développé par Riot Games, et les statistiques des jeux les plus populaires sur la plateforme Steam. L'application se concentre notamment sur l'affichage de graphiques interactifs utilisant la bibliothèque OxyPlot pour faciliter la visualisation des données.

Lexique

Pour être sûr de comprendre tous les termes utilisés dans ce rapport quelque mot ont besoin d'être mit au clair :

Dans League of Legends, un **invocateur** est un joueur qui possède un compte et un profil de jeu.

League of Legends est un jeu multijoueur en ligne de type MOBA (Multiplayer Online Battle Arena), où les joueurs incarnent des champions avec des compétences spécifiques. Développé et publié par Riot Games, LoL est l'un des jeux en ligne les plus populaires au monde, avec un système de progression permettant aux invocateurs d'améliorer leurs compétences et de concourir en équipes.

Riot Games est une société de développement de jeux vidéo, principalement connue pour League of Legends.

2. Objectifs

- Réaliser un programme de qualité qui a pour but de pouvoir inspecter le profil d'un joueur du jeu League of Legends, ainsi d'afficher les nombres de joueurs des 3 jeux les plus jouer sur Steam.
- Utilisation de user stories lors du développement pour garder un cap clair quant aux fonctionnalités du programme.
- Utiliser un système de versioning de code ainsi que de commit bien nommer et atomique.
- Mise en place d'un journal de travail pour retracer le temps passé sur le projet

- Création d'une documentation apportant des précisions sur le projet

3. Gestion de projet

Les outils suivants ont été utiliser pour la gestion de ce projet :

- IceScrum, pour crée et planifier les user stories.
- GitHub, pour le suivie et le versioning du code.
- Excel, pour le journal de travail.
- Word, pour la rédaction de ce rapport.

Analyse / Conception

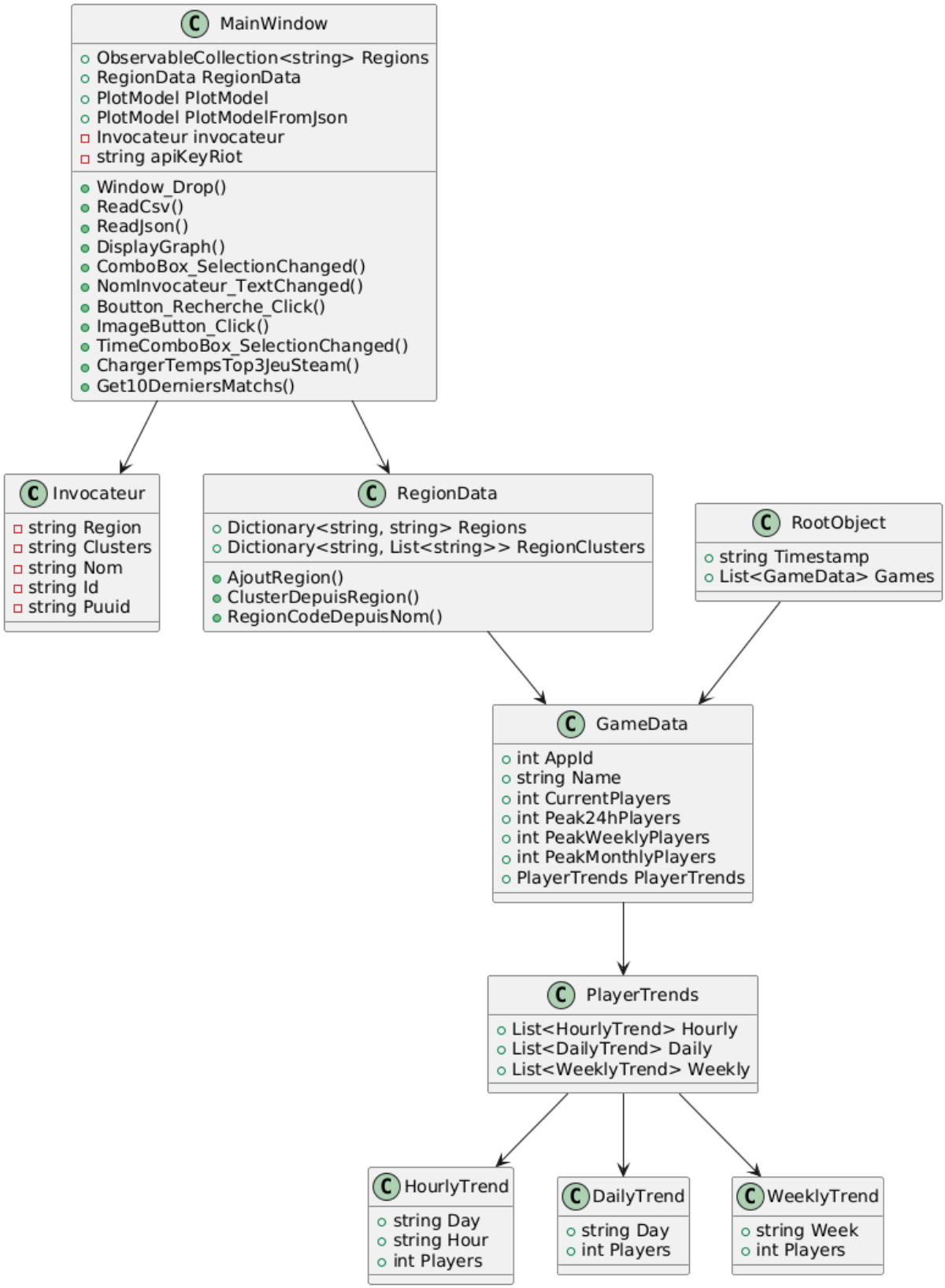
1. Domaine

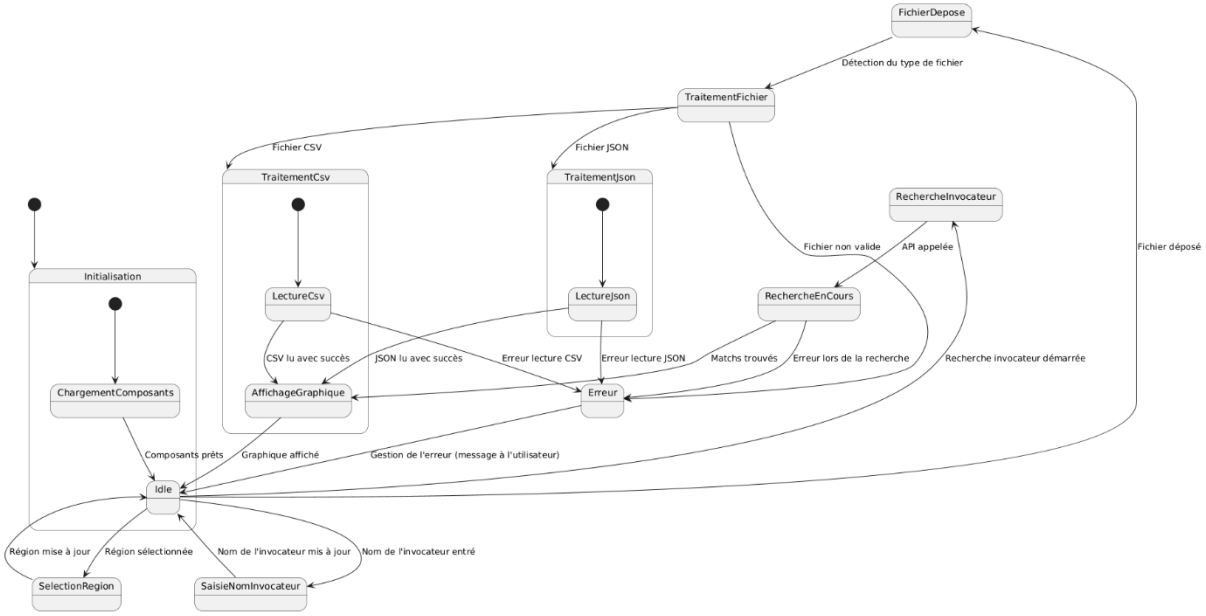
Les données choisies sont des statistiques fictives crée de manière procédurale (pour en avoir le plus grand nombre) à l'aide de git copilot sur les 3 jeux le plus jouer de Steam (Counter Strike : Global Offensive, Dota 2, Team Fortress 2), les données choisies sont :

- App_id, qui est un identifiant unique pour identifier un jeu
- Name, le nom du jeu
- Current_players, le nombre de joueur connecter en ce moment
- Peak_24h_players, le nombre record de joueurs connecter ces 24 dernières heures
- Peak_weekly_players, le nombre record de joueurs connecter ces 7 dernier jours
- Peak_monthly_players, le nombre record de joueurs connecter ce dernier mois
- players_trends, qui est un objet qui contient les listes suivantes :
    - hourly, qui contient des objets composés de :
        - day, la date au format ISO 8601
        - hour, l'heure au format xx:xx
        - players, le nombre de joueurs
    - daily, qui contient des objets composés de :
        - day
        - players

Ces données me permettent d'avoir des stastistiques de l'ordre de quelque semaine/jour.

2. Concept


MainWindow
ObservableCollection<string> Regions
RegionData RegionData
PlotModel PlotModel
PlotModel PlotModelFromJson
Invocateur invocateur
string apiKeyRiot
Window_Drop()
ReadCsv()
ReadJson()
DisplayGraph()
ComboBox_SelectionChanged()
NomInvocateur_TextChanged()
Boutton_Recherche_Click()
ImageButton_Click()
TimeComboBox_SelectionChanged()
ChargerTempsTop3JeuSteam()
Get10DerniersMatchs()
Invocateur
string Region
string Clusters
string Nom
string Id
string Puuid
RegionData
Dictionary<string, string> Regions
Dictionary<string, List<string>> RegionClusters
AjoutRegion()
ClusterDepuisRegion()
RegionCodeDepuisNom()
RootObject
string Timestamp
List<GameData> Games
GameData
int AppId
string Name
int CurrentPlayers
int Peak24hPlayers
int PeakWeeklyPlayers
int PeakMonthlyPlayers
PlayerTrends PlayerTrends
PlayerTrends
List<HourlyTrend> Hourly
List<DailyTrend> Daily
List<WeeklyTrend> Weekly
HourlyTrend
string Day
string Hour
int Players
DailyTrend
string Day
int Players
WeeklyTrend
string Week
int Players



3. Analyse fonctionnelle

Manière d'afficher les nombres de joueurs steam

(Auteur: Mateen Khaiy)

Je veux voir le nombre de joueurs des trois jeux les plus joués sur steam par heure sur une durée de 1 semaine (maquette: schéma graphique)	
Tests d'acceptance:	
Lancement de l'app	Depuis mon desktop, Quand je double-clique l'icône Je vois le graphique: (maquette "schéma graphique")
couleur distincte par jeu	dans la mainWindow quand le programme affiche les nombres de joueurs pour les top 3 jeux steam chaque jeu a une ligne dans le graphique qui lui est assigné avec une couleur unique et propre a lui même
type de graphique	dans la mainWindow quand le programme affiche les nombres de joueurs pour les top 3 jeux steam le graphique est de type zone (maquette "graphique de zone")

légende pour chaque ligne	dans la mainWindow quand je clique sur une courbe une boite de texte affiche le nom du jeu, la date, le nombre de joueur

afficher historique de partie pour invocateur

(Auteur: Mateen Khaiy)

je veux voir les 5 dernières partie de l'invocateur don j'ai données la region, le nom et l'id (voir photo liée)	
Tests d'acceptance:	
ordre d'affichage de manière antéchronologique	dans la invocateurHistoryWindow les parties sont afficher de la plus récentes au moins récentes
scroll bar existante	dans la invocateurHistoryWindow a droite des parties listées il y a une scroll bar qui me permet de naviguer verticalement entre les parties
contenant des parties dans l'historique	dans la invocateurHistoryWindow le résumé des parties contient une icon du champion que l'invocateur a joué, un texte (win/loose) qui indique si il a gagner ou perdu la partie, et les nombre de kill, de morts, et d'assistance qu'il a eu durant cette partie.

recherche invocateur

(Auteur: Mateen Khaiy)

en tant qu'utilisteur je veux rechercher un invocateur précis	
Tests d'acceptance:	
recherche réussite	dans mainWindow, avec des infos d'invocateur correctes, quand je clique recherche (loupe, voir maquette schema) la fenêtre

	invocateurHistoryWindow (maquette) s'ouvre et affiche ses 5 dernières partie
message d'erreur	dans mainWindow, avec des infos d'invocateur incorrectes, quand je clique recherche (loupe, voir maquette schéma) il affiche un message d'erreur disant "invocateur non trouvé" (voir image liée)
invocateur précédemment rechercher	dans ma invocateurHistoryWindow quand j'ai au préalable déjà rechercher un invocateur et que j'utilise la barre de recherche à nouveau et que le bouton on/off est régler du coter "historique" le programme me propose tous les invocateurs que j'ai déjà rechercher depuis le lancement du programme (voir maquette)

invocateurs favoris

(Auteur : Mateen Khaiy)

je veux pouvoir enregistrer mes invocateurs favoris afin de pouvoir les rechercher rapidement à l'avenir	
	Tests d'acceptance :
bouton favoris	dans ma invocateurHistoryWindow il y a un bouton d'invocateur favoris (maquette : schéma bouton favoris)
Ajout aux favoris	dans ma invocateurHistoryWindow quand j'ajoute un invocateur au invocateur favoris le bouton favori se rempli (maquette : schéma bouton favoris activé)
recherche invocateur favoris	dans mainWindow quand le bouton on/off est cocher du coter "favoris" quand je sélectionne ma search bar, les invocateurs qui sont favoris sont lister (maquette : invocateur favoris liste)

Exporter les données des graphiques au format CSV

(Auteur : Mateen Khaiy)

je veux exporter les données des graphiques au format CSV	
Tests d'acceptance :	
bouton d'export	sur l'interface de n'importe quelle graphique un bouton "exporter en CSV" se trouve en bas à droite du graphique (maquette : bouton export)
données exportées dans un fichier CSV	quand un graphique est affiché quand l'utilisateur clique sur le bouton "Exporter en CSV" les données du graphique sont exportées dans un fichier CSV qui se trouve dans le dossier "téléchargement" de l'utilisateur

4. Stratégie de test

Notre programme contient 3 tests qui permet d'être sûr que les fonctionnalités principales du projet fonctionnent correctement.

a. TestRegionSelectionChanged

Permet de tester si l'item choisie dans la liste a choix est stocker correctement dans l'objet invocateur

```
[TestMethod]
0 références
public void TestRegionSelectionChanged()
{
    var thread = new Thread(() =>
    {
        // Arrange
        var mainWindow = new MainWindow();
        var comboBox = new ComboBox { ItemsSource = mainWindow.Regions };
        comboBox.SelectedItem = "Europe West";

        // Act
        mainWindow.ComboBox_SelectionChanged(comboBox, null);

        // Assert
        Assert.AreEqual("Europe West", mainWindow.invocateur.Region);
    });
}
```

b. TestNomInvocateur_TextChanged

Permet de tester si le programme fait bien la différence entre le pseudo et l'id quand il est saisi.

```
[TestMethod]
| 0 références
public void TestNomInvocateur_TextChanged()
{
    var thread = new Thread(() =>
    {
        // Arrange
        var mainWindow = new MainWindow();
        var textBox = new TextBox { Text = "Summoner#1234" };

        // Act
        mainWindow.NomInvocateur_TextChanged(textBox, null);

        // Assert
        Assert.AreEqual("Summoner", mainWindow.invocateur.Nom);
        Assert.AreEqual("#1234", mainWindow.invocateur.Id);
    });
    thread.SetApartmentState(ApartmentState.STA);
    thread.Start();
    thread.Join();
}
```

c. TestReadJson

Permet de tester si le programme arrive bel et bien à transformer un fichier JSON en liste de point

```
[TestMethod]
0 références
public void TestReadJson()
{
    var thread = new Thread(() =>
    {
        // Arrange
        var mainWindow = new MainWindow();
        string jsonContent = @"
        {
            'Games': [
                {
                    'player_trends': {
                        'Hourly': [
                            { 'Day': '2023-10-01', 'Hour': '10:00', 'Players': 500 },
                            { 'Day': '2023-10-01', 'Hour': '11:00', 'Players': 600 }
                        ]
                    }
                }
            ]
        }";

        string tempFilePath = Path.GetTempFileName();
        File.WriteAllText(tempFilePath, jsonContent);

        // Act
        var dataPoints = mainWindow.ReadJson(tempFilePath);

        // Assert
        Assert.AreEqual(2, dataPoints.Count);
        Assert.AreEqual(500, dataPoints[0].Y);
        Assert.AreEqual(600, dataPoints[1].Y);

        // Clean up
        File.Delete(tempFilePath);
    });
}
```

Réalisation

1. Gestion et traitement des fichiers JSON pour l'affichage graphique

Dans le programme RiftMetrics l'une des fonctionnalités principales est de permettre aux utilisateurs de drag-and-drop des fichiers au format JSON.

Problème :

Les fichiers JSON ont des structures imbriquées, le défi était donc de désérialiser ces fichiers de manières correcte et structuré

Solution technique

Nous avons utilisé la bibliothèque json.NET pour gérer la désérialisation des fichiers JSON. Ces derniers sont déposés par les utilisateurs, puis lus, puis convertis en objet RootObject contenant les données sur les jeux données dans le fichier JSON.

a. **Lecture du fichier JSON et désérialisation**
   - Lorsqu'un fichier JSON est déposé, son contenu est lu et est transformé en objet

b. **Transformation des données en affichage graphique**
   - Les données de l'objet sont transformées en liste de point prêt à être afficher sous forme de graphique

2. Récupération des données d'un invocateur

Une autre fonctionnalité du programme RiftMetrics est de récupérer des données d'un invocateur pour pouvoir ensuite les utilisés dans des requêtes vers l'API de Riot Games.

Il est important de noter que pour cause de nombreux problèmes qui sont survenus au cours du développement, l'application final ne comporte pas de fonctionnalité qui utilise l'api de Riot Games (comme pour afficher les 5 dernières parties de l'invocateur par exemple), ce chapitre est uniquement se concentre uniquement sur la collecte des données que l'invocateur fournis (comme son pseudo et sa région)

Problème :

Le problème principal était d'être sûr que les données rentrer par l'invocateur était correcte et apte à être utiliser dans des requêtes à l'API de Riot Game, les données spécifier sont :

- La saisi d'un pseudo d'invocateur valide et existant.
- La sélection d'une région parmi les 15 proposées par Riot Games
- Un ID d'invocateur valide qui est saisie à la suite du pseudo, la séparation entre le pseudo et l'ID est marqué par le caractère « # » (exemple : teemo#EUW, dans cette exemple teemo est le pseudo et EUW est l'ID)

Solution technique :

La solution mise en place pour récupérer les données de l'invocateur se fait en deux étapes :

a. Sélection de la région

- l'utilisateur sélectionne la région a la quelle l'invocateur qu'il désire rechercher appartient parmi une liste déroulante.
- Une fois que l'utilisateur à sélectionner une région, cette dernière est associée à l'objet Invocateur

b. Saisie du pseudo et de l'ID de l'invocateur

- L'utilisateur entre le pseudo de l'invocateur dans un champ de texte. Si le pseudo contient un ID séparé par un symbole « # » (exemple : teemo#EUW), le programme découpe correctement le pseudo en deux parties : le nom de l'invocateur et son ID.
- A chaque fois que le contenu du champ de texte change, le programme exécute la méthode suivante :

```
public void NomInvocateur_TextChanged(object sender, TextChangedEventArgs e)
{
    TextBox textBox = sender as TextBox;

    if (textBox != null)
    {
        string text = textBox.Text;
        int index = text.IndexOf('#');

        if (index >= 0)
        {
            invocateur.Nom = text.Substring(0, index);
            invocateur.Id = text.Substring(index);
            Debug.WriteLine("Nom de l'invocateur : " + invocateur.Nom);
            Debug.WriteLine("Id de l'invocateur : " + invocateur.Id);
        }
        else
        {
            invocateur.Nom = text;
            invocateur.Id = string.Empty;
        }
    }
}
```

3. Déroulement

Au cours du développement de l'application plusieurs problèmes ont été rencontrer, voici quelque unes :

- API Riot Games :
- Affiche des 3 jeux les plus joués :
- Drag-and-Drop de fichier :

4. Mise en place de l'environnement de travail

Le code source de mon programme, ainsi que les fichiers relatifs aux documentations sont disponible dans mon repo GitHub. Ce dernier contient deux répertoires.

le premier se nomme « Code » et contient le code source.

Le deuxième se nomme « Documentation » et contient mon journal de travail, les spécifications du projet ainsi que mon rapport de projet.

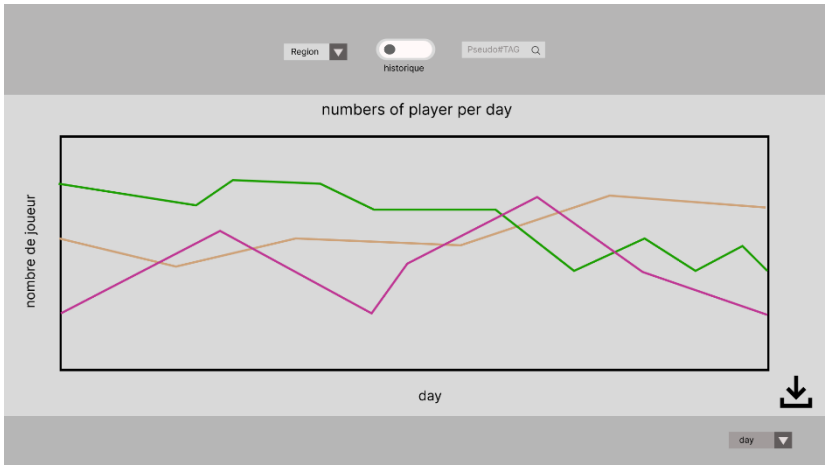
Durant ce projet j'ai utilisé les outils suivants :

- Visual studio 2022
- iceScrum
- GitHub Desktop
- GitHub

5. Description des tests effectués

Exporter les données des graphiques au format CSV

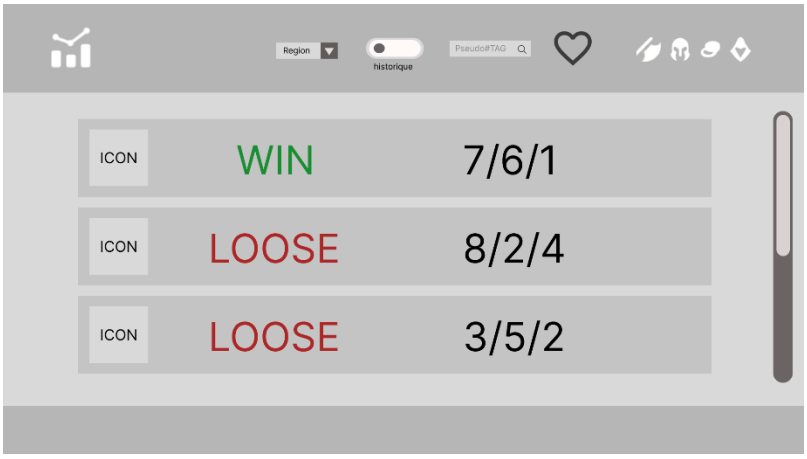
bouton d'export	sur l'interface de n'importe quelle graphique un bouton "exporter en CSV" se trouve en bas à droite du graphique (maquette : bouton export)	ko
données exportées dans un fichier CSV	quand un graphique est affiché quand l'utilisateur clique sur le bouton "Exporter en CSV" les données du graphique sont exportées dans un fichier CSV qui se trouve dans le dossier "téléchargement" de l'utilisateur	ko



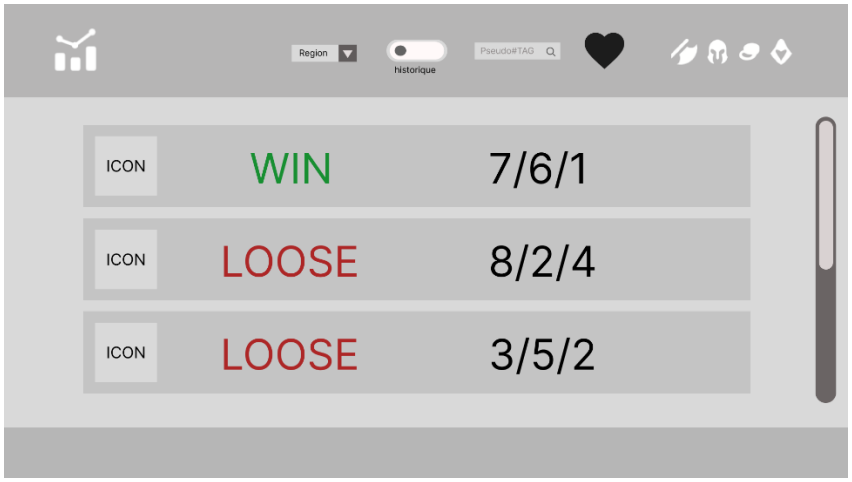
maquette 1 bouton export

invocateur favoris

bouton favoris	dans ma invocateurHistoryWindow il y a un bouton d'invocateur favoris (maquette : schéma bouton favoris)	ko
Ajout aux favoris	dans ma invocateurHistoryWindow quand j'ajoute un invocateur au invocateur favoris le bouton favori se rempli (maquette : schéma bouton favoris activé)	ko
recherche invocateur favoris	dans mainWindow quand le bouton on/off est cocher du coter "favoris" quand je sélectionne ma search bar, les invocateurs qui sont favoris sont lister (maquette : invocateur favoris liste)	ko



maquette 2 bouton favoris



maquette 3 bouton favoris activé

maquette 4 invocateur favoris liste

recherche invocateur

recherche réussite	dans mainWindow, avec des infos d'invocateur correctes, quand je clique recherche (loupe, voir maquette schéma) la fenêtre	ko

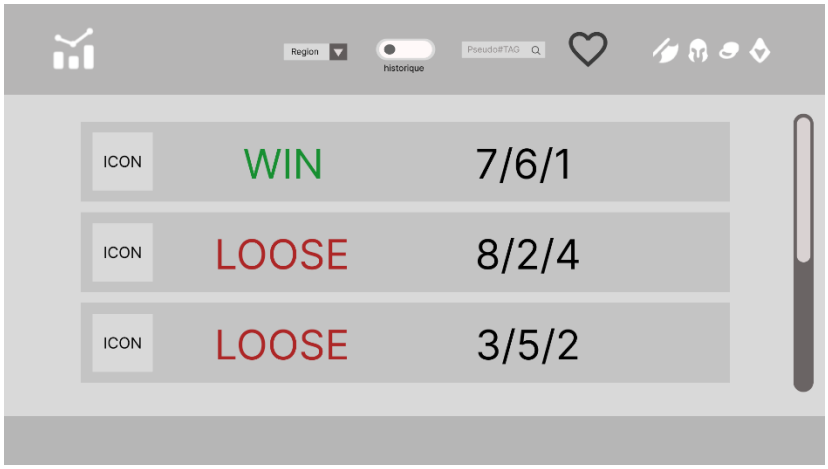
	invocateurHistoryWindow (maquette : recherche invocateur précedent) s'ouvre et affiche ses 5 dernières partie	
message d'erreur	dans mainWindow, avec des infos d'invocateur incorrectes, quand je clique recherche (loupe, voir maquette schéma) il affiche un message d'erreur disant "invocateur non trouvé" (maquette : invocateur non trouvé)	ko
invocateur précédemment rechercher	dans ma invocateurHistoryWindow quand j'ai au préalable déjà rechercher un invocateur et que j'utilise la barre de recherche à nouveau et que le bouton on/off est régler du coter "historique" le programme me propose tous les invocateurs que j'ai déjà rechercher depuis le lancement du programme (maquette : recherche invocateur précedent)	ko

maquette 5 recherche invocateur précedent

maquette 6 invocateur non trouvé

afficher historique de partie pour invocateur

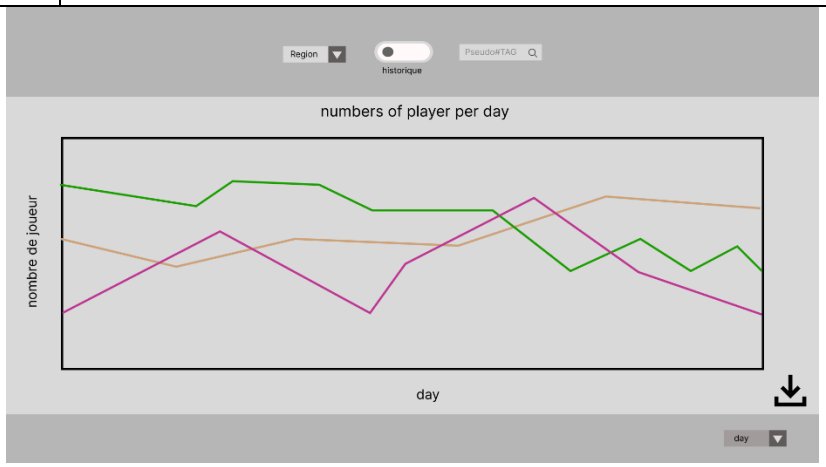
ordre d'affichage de manière antéchronologique	dans la invocateurHistoryWindow les parties sont afficher de la plus récentes au moins récentes	ko
scroll bar existante	dans la invocateurHistoryWindow à droite des parties listées il y a une scroll bar qui me permet de naviguer verticalement entre les parties	ko
contenant des parties dans l'historique	dans la invocateurHistoryWindow le résumé des parties contient une Icon du champion que l'invocateur a joué, un texte (Win/loose) qui indique si il a gagné ou perdu la partie, et les nombre de kill, de morts, et d'assistance qu'il a eu durant cette partie (maquette : invocateur historique).	ko



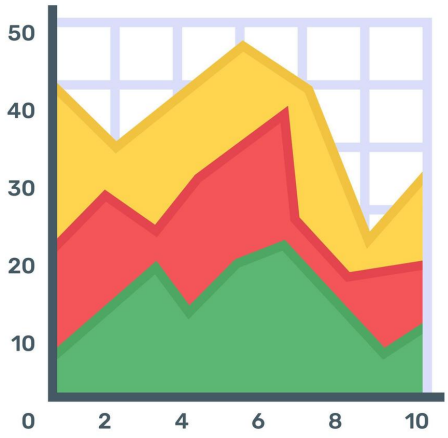
maquette 7 invocateur historique

Manière d'afficher les nombres de joueurs Steam

Lancement de l'app	Depuis mon desktop, Quand je double-clique l'icône Je vois le graphique : (maquette "schéma graphique")	OK 1 Nov
couleur distincte par jeu	dans la mainWindow quand le programme affiche les nombres de joueurs pour le top 3 jeux Steam chaque jeu a une ligne dans le graphique qui lui est assigné avec une couleur unique et propre a lui même	ko
type de graphique	dans la mainWindow quand le programme affiche les nombres de joueurs pour le top 3 jeux Steam le graphique est de type zone (maquette "graphique de zone")	OK 1 Nov
légende pour chaque ligne	dans la mainWindow quand je clique sur une courbe une boite de texte affiche le nom du jeu, la date, le nombre de joueur	OK 1 Nov



maquette 8 schéma graphique



maquette 9 graphique de zone

6. Erreurs restantes

Problème de communication avec l'API de Riot Games

La communication avec l'api de Riot Games pour prendre les informations de l'invocateur entrée par l'utilisateur n'a pas pu être finaliser. Quand le programme essaye de faire la requête, une erreur indique que l'url n'est pas correcte. Cela est probablement dû à une mauvaise compréhension de la documentions de Riot Games.

System.UriFormatException : 'Invalid URI: The hostname could not be parsed.'

message erreur api

Sans la connexion à l'api, les fonctionnalités d'affichage des données de jeu récentes et des statistiques d'invocateur ne sont pas disponibles. L'utilisateur ne peut donc pas obtenir les informations sur les parties récentes de l'invocateur rechercher.

Actions envisagées ou possibles :

Implémenter des messages d'erreurs détaillés pour informer l'utilisateur en cas de problème de communication avec l'API.

S'inspirer d'autres projet tier qui utilise l'api de Riot Games pour comprendre comment l'utiliser et l'implémenter dans le code du programme de manière correcte.

Fonctionnalité de Drag and Drop pour les fichiers CSV non testée

La fonctionnalité de drag-and-drop de fichier csv n'a pas encore été tester. Elle permettrait aux utilisateurs de glisser un fichier CSV directement dans l'application pour visualiser ses données sous forme de graphique. Actuellement, il est possible que la fonctionnalité ne fonctionne pas comme prévu.

Si cette fonctionnalité de fonctionne pas, cela obligerait l'utilisateur a utilisé d'autre moyen pour afficher leurs données en un format graphique.

Actions envisagées ou possibles :

Étudier comment d'autres applications implémentent et gèrent la fonctionnalité de drag and drop

Conclusion

Ce projet RiftMetrics a permis d'atteindre plusieurs objectifs clés, mais certains aspects restent à finaliser.

Objectifs atteints / non-atteints

Les objectifs principaux ont été réalisés, notamment :

Affichage des Top 3 jeux Steam : Lecture et visualisation des données JSON dans un graphique OxyPlot.

Gestion d'erreurs : Des messages d'erreur informatifs ont été intégrés pour améliorer l'expérience utilisateur.

Cependant, certains objectifs n'ont pas été atteints :

Communication avec l'API de Riot Games : La connexion pour afficher les informations d'un invocateur reste incomplète.

Drag and Drop de fichiers CSV : La fonctionnalité de glisser-déposer nécessite des tests et des ajustements.

Points positifs et difficultés

Le projet a permis de travailler sur WPF, un domaine peu exploré, et d'aborder la manipulation JSON et la programmation fonctionnelle.

Cependant, la prise en main de WPF a été complexe, et l'intégration de l'API de Riot Games s'est révélée difficile. Également quand j'ai publier mon code pour l'avoir sous forme d'exécutable, plusieurs fonctionnalité avait changer, par exemple, quand je lance mon application .exe elle n'affiche pas les données du JSON que je lui ai donnée et comme je me suis rendu compte de cela tardivement lors du développement du projet je n'ai pas pu le corriger

Suites possibles

Les améliorations possibles incluent la finalisation du drag and drop pour les fichiers CSV et l'intégration complète de l'API de Riot Games.

Utilisation de ChatGPT

Durant ce projet j'ai souvent utilisé ChatGPT pour m'aider car wpf était un nouveau domaine que j'ai trouvé difficile à prendre en main. Je ne l'ai par exemple utilisé pour qu'il m'aide à désérialiser des fichiers json.

Annexes

1. Journal de travail