Plot That Line

Belkhiria Sofiène

Table des matières

1 Analyse préliminaire

1.1 Introduction

Le projet Plot That Line se déroule en parallèle du module I323 et a pour but pédagogique l'utilisation du paradigme fonctionnel avec la librairie LinQ en C#.
Le projet en lui-même consiste à développer une application graphique d'analyse de donnée en C#, le jeu de donné et la technologie d'affichage sont quant à eux au choix.

1.2 Objectifs

- Développement d'une application d'analyse des statistiques NBA.
- Elaboration d'un fil rouge de développement à base de users stories de qualité.
- Mise en place d'un historique de développement du projet fiable via des commits bien nommé et atomique.
- Mise en place d'un historique du temps via un journal de retraçant le temps passé sur les différents pants du projet
- Développement de maquette basse fidélité de l'application
- Elaboration d'une documentation expliquant le déroulement les objectifs ainsi que le fonctionnement du projet.

1.3 Gestion de projet

La gestion de ce projet se fera avec les outils suivants :

- IceScrum pour l'élaboration des user stories.
- GitHub/desktop afin de suivre l'évolution du code ainsi que son accessibilité.
- Excel pour le journal de travail.
- Word pour la création du rapport de projet.

A noté que la planification se fera aussi à l'aide des différentes user stories.

2 Analyse / Conception

2.1 Domaine

Les données choisies sont les statistiques de base de chaque match NBA de la saison 2023-2024, ces statistiques sont :

- La date du match.
- L'heure de début du match local au lieu du match.
- L'équipe Visiteuse.
- Les points de l'équipe visiteuse.
- L'équipe à domicile.
- Les points de l'équipe à domicile.

Ces données seront donc affichées sur une échelle d'une semaine chaque jour représentant la moyenne de point par match que l'équipe marque sur l'ensemble de la saison pour le jour spécifique.

Ces données pourront alors être utiliser pour :

- Des tentatives de prédiction de résultats par les différents organismes de paris.
- Les équipes techniques des différentes équipes NBA afin de mieux se préparer pour les matchs.

- Les analystes de statistique NBA avancé afin d'émettre des théories quant à la raison des résultats des équipes sur l'ensemble de la saison.

Le programme devrait lire les données de différentes manières :

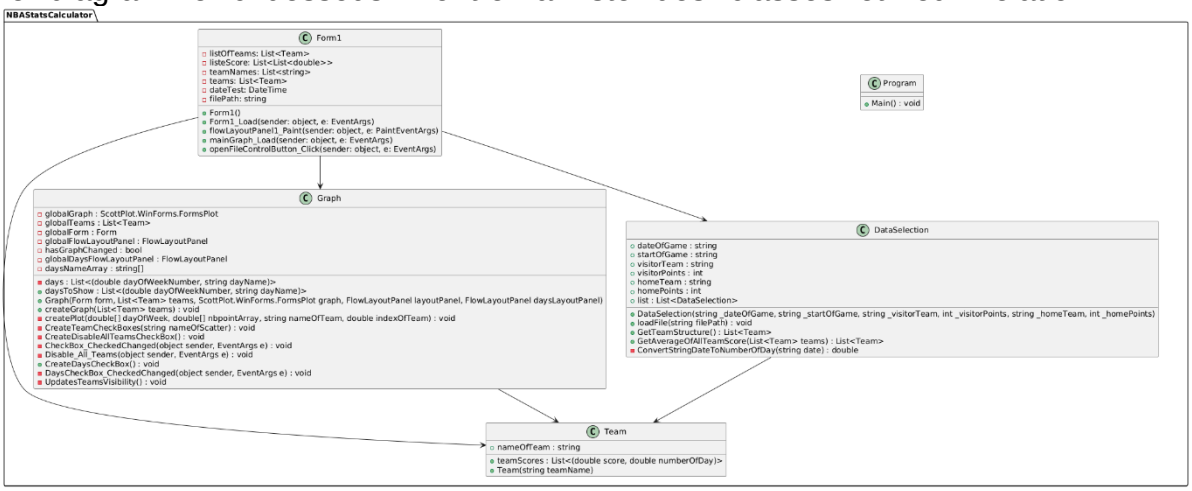
- Via des requêtes API renvoyant une réponse au format JSON
- Via la lecture d'un fichier JSON ayant des clefs identiques que le canevas ci-dessous.


```
{
  "Date": "Tue Oct 24 2023",
  "Start": "7:30p",
  "Visitor": "Los Angeles Lakers",
  "VPTS": 107,
  "Home": "Denver Nuggets",
  "HPTS": 119,
  "Attend": "Box Score",
  "LOG": "",
  "Arena": 19842,
  "Notes": "2:17"
},
```

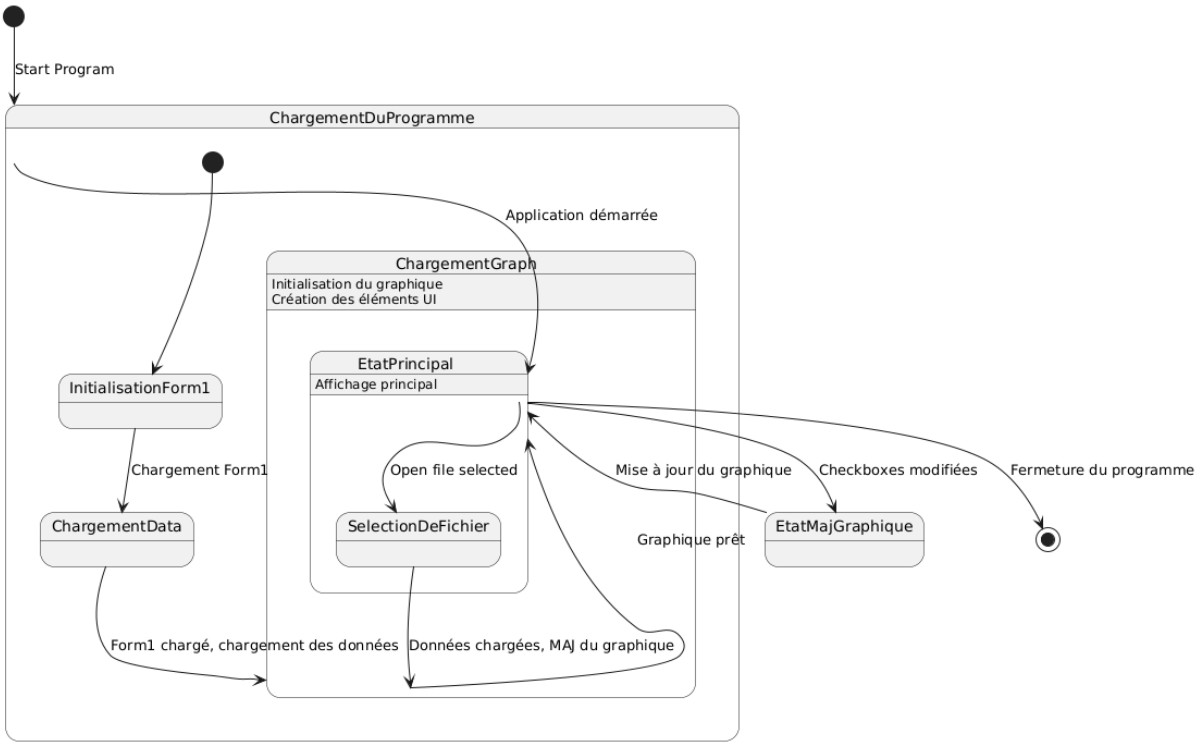

A noté que les champs : « Attend, LOG, Arena, Notes » ne sont pas obligatoire.

2.2 Concept

1. Le diagramme ci-dessous montre la liste des classes et leur relation :



2. Le diagramme ci-dessous montre les différents états du programme :



2.3 Analyse fonctionnelle

2.3.1 Graphe à multiple ligne

(Auteur : Sofiène Belkhiria)

En tant qu'utilisateur Je veux un graphe avec plusieurs courbes où chaque courbe représente une équipe Afin de pouvoir comparer les équipes	
Tests d'acceptance :	
Nombre de Ligne sans filtre	Sur la page de graphique Quand il est affiché et que aucun filtre n'est appliqué L'utilisateur voit 30 courbes différentes
Affichage label	Sur la page de graphique Quand il est affiché et que aucun filtre n'est appliqué L'utilisateur voit une partie label contenant autant de label que de courbe
Zoom sur graph	Sur la page de graphique Quand l'utilisateur scroll vers l'avant avec la molette sur le graph Le graph zoom
Dézoom sur graph	Sur la page de graphique Quand l'utilisateur scroll vers l'arrière avec la molette sur le graph Le graph dézoom

2.3.2 Filtrage des lignes

(Auteur : Sofiène Belkhiria)

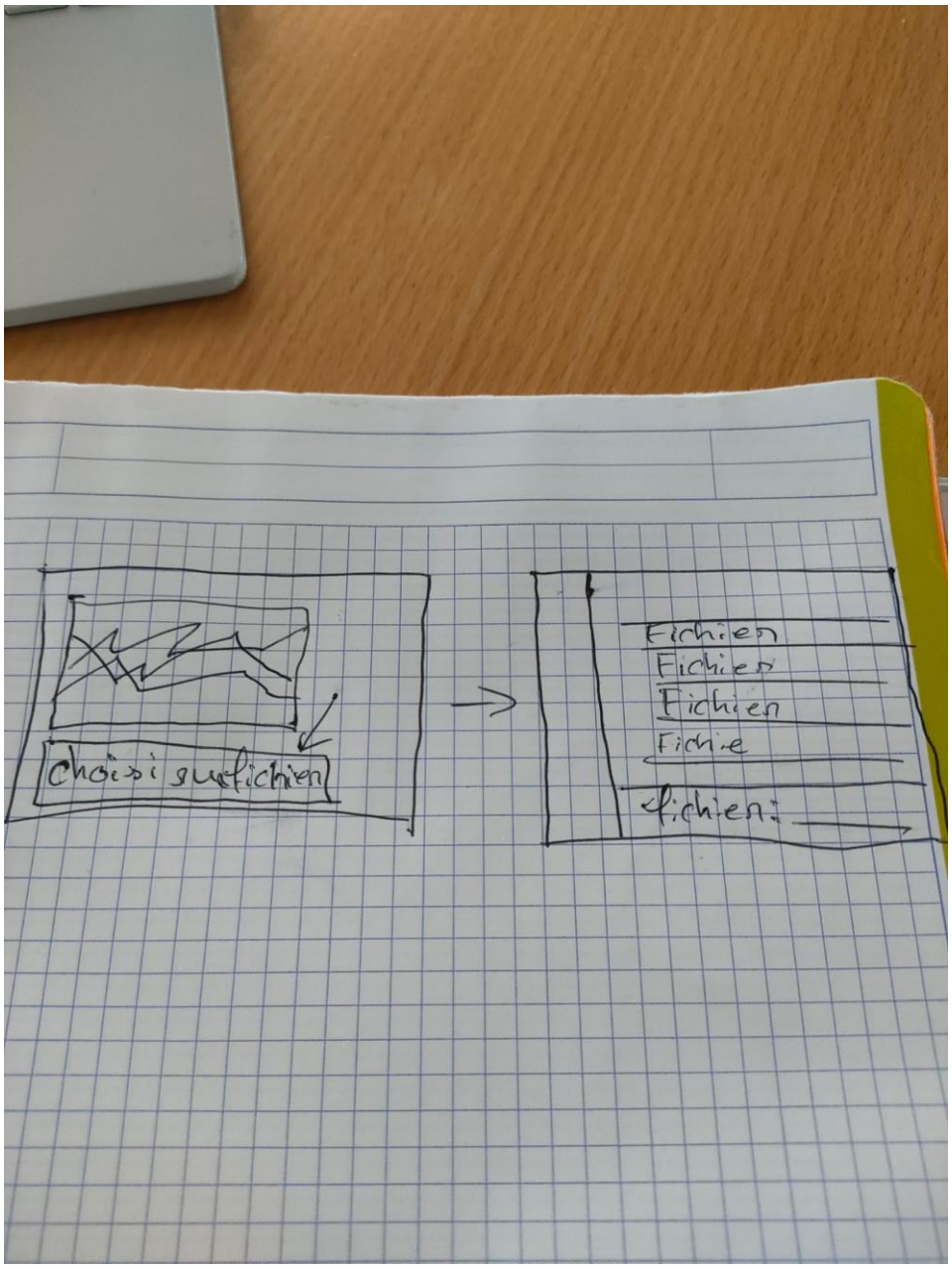
En tant qu'utilisateur Je veux choisir les courbes qui sont affichée Afin de pouvoir mieux comparer et trouver les choses	
Tests d'acceptance :	
Check box jour	Sur la page du graphe Quand le graph est affiché et qu'aucun filtre n'est choisi La checkBox affiche que tous les jours sont sélectionné

Check box équipe	Sur la page du graphe Quand le graph est affiché et qu'aucun filtre n'est choisi La legendes des équipes contient autant d'équipe que d'équipe sélectionnée
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique sur une équipe dans la partie label Le graph affiche uniquement l'équipe sur laquelle l'utilisateur a cliqué
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique sur une équipe dans la partie label La couleur de la ligne affichée est la même que la couleur de l'équipe ayant était cliquée dans la partie label
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique plusieurs fois sur la même équipe dans la partie filtre Les informations de la ligne ne changent pas
Selection des équipes à ne pas afficher	Sur la page du graphe Quand l'utilisateur décoche la checkBox d'une équipe L'équipe d'ont la checkbox a été décocher disparait
Selection des équipes à afficher	Sur la page du graphe Quand l'utilisateur coche la checkBox d'une équipe L'équipe d'ont la checkbox a été coché apparait
Affichage jour	Sur la page du graphe Quand l'utilisateur coche la checkBox d'un jour la statistique qui correspond au jour apparait
Désaffichage jour	Sur la page du graphe Quand l'utilisateur décoche la checkBox d'un jour la statistique qui correspond au jour disparait

2.3.3 Choix du fichier de graph

(Auteur : Sofiène Belkhiria)

En tant qu'utilisateur Je veux pouvoir choisir le fichier qui va être charger dans le graph Afin de pouvoir voir les statistique des saisons qui m'intéressent	
Tests d'acceptance :	
Affichage du menu de choix de fichier	Sur la page principale Quand l'utilisateur clique sur le bouton "Choisir un fichier" Un menu permettant de choisir un fichier apparait (Voir maquette)
Extension des fichier	Sur la page principale Quand l'utilisateur à cliquer sur "Choisir un fichier" et que l'explorateur de fichier s'ouvre l'utilisateur ne peut choisir que des fichiers de type JSON
Mise à jour du graph	Sur la page de graph Quand l'utilisateur change de fichier Les statistique affichée sur le graph s'adapte au fichier choisi
Affichage par défaut	Sur la page de graph Quand l'utilisateur ne choisit pas de fichier Le graph est vide
Affichage des filtres par équipe	Sur la page de graph Quand un utilisateur choisi un fichier La checkbox des équipes contient autant d'équipe qu'il y d'équipe dans le fichier



2.3.4 CheckBox et Ligne Adaptative

(Auteur : Sofiène Belkhiria)

En tant qu'utilisateur, Je veux que les checkbox d'équipe et les courbe s'adapte aux équipes présentes dans mon fichier de donnée.	
Tests d'acceptance :	
Nombre de Ligne	Sur le graphique quand aucun filtre n'est appliqué je vois autant de ligne que d'équipe dans mon jeu de donnée
Nombre de checkBox	Sur le page du graphique Quand un fichier a été chargé je vois autant de checkbox d'équipe que d'équipe dans mon jeu de donnée et je vois une checkBox supplémentaire permettant de tout supprimer.
Nom des checkBox	Quand les checkbox sont affichée le nom des checkBox correspond aux noms des équipes de mon fichier de donnée

2.4 Stratégie de test

Afin de tester et de vérifier au mieux le fonctionnement de l'application plusieurs tests ont été mis en place tels que :

- Test unitaire afin de vérifier la logique du code.
- Test par les paires afin de vérifier la solidité du code.
- Test par des proche non informaticien afin de vérifier l'affordance et la solidité de l'application en cas d'utilisation inattendue.

La stratégie de test à été accès sur les 3 fonctionnalité les plus important du programme qui sont :

- La génération du graphique.
- L'importation des données via un fichier. json adéquat.
- Les différents filtres du graphique

Chaque type de test permets de vérifier ces points de différentes manières.

1. Génération du graphique.
    a. Le test unitaire si dessous vérifie que le graphique générer n'est pas vide :

```
[TestMethod()]
0 références
public void createGraphTest()
{
    ///Arange
    Form form = new Form();
    ScottPlot.WinForms.FormsPlot graphPlot = new ScottPlot.WinForms.FormsPlot();
    FlowLayoutPanel layoutPanel = new FlowLayoutPanel();
    FlowLayoutPanel daysLayoutPanel = new FlowLayoutPanel();

    Team team1 = new Team("Equipe1");
    Team team2 = new Team("Equipe2");
    team1.teamScores = new List<(double score, double numberOfDay)>
    {
        (89,1),(100,2)
    };
    team2.teamScores = new List<(double score, double numberOfDay)>
    {
        (110,1),(95,2)
    };
    List<Team> teams = new List<Team>();
    Graph graph = new Graph(form, teams, graphPlot, layoutPanel, daysLayoutPanel);
    graph.daysToShow = new List<(double, string)>
    {
        (1, "lundi"), (2, "mardi")
    };
    ///Act
    teams.Add(team1);
    teams.Add(team2);
    graph.createGraph(teams);
    ///Assert
    Assert.IsTrue(graphPlot.Plot.GetPlottables().ToList().Count() > 0, "Le graphique est vide après l'appel de createGraph.");
}
```

    b. Le test par les paires à permis de vérifier que peu importe combien de fois on charge le même fichier le graphique ne change pas.
    c. Le test par des utilisateurs a permis de tester et d'assurer le bon fonctionnement du graphique malgré des zoom et dézoome continue.
2. Importation des données via un fichier.
    a. Le test unitaire ci-dessous vérifie que la classe DataSelection.cs ne soit pas vide et que les variables contenant les informations des équipes soit

du bon type une fois que la méthode LoadData ait été effectuée :

```
[TestMethod()]
0 références
public void loadFileTest()
{
    ///Arrange
    DataSelection data = new DataSelection("", "", "", 0, "", 0);
    string filePath = @"C:\Users\BILEL\Documents\PlotThatLine\App\NBAStatsCalculator\teamData.json";
    ///Act
    data.loadFile(filePath);
    ///Assert
    Assert.IsInstanceOfType(data.homeTeam,typeof(string));
    Assert.IsInstanceOfType(data.visitorTeam, typeof(string));
    Assert.IsInstanceOfType(data.homePoints, typeof(int));
    Assert.IsInstanceOfType(data.visitorPoints, typeof(int));
    Assert.IsInstanceOfType(data.dateOfGame, typeof(string));
    Assert.IsTrue(data.list.Count > 0);
}
```

b. Le test par les paires a permis d'assurer la gestion d'erreur du code dans le cas où les clefs du fichier. json ne correspondent pas à celle attendue ou que les valeurs de ses clef ne soient pas du bon type.

3. Filtres du graphique.
   a. Le test unitaire ci-dessous permet de vérifier que la listes des équipes afficher dans le graphique soit équivalente à la liste des équipes devant être afficher dans les filtres :

```
public void createGraphTeamsFilterTest()
{
    Form form = new Form();
    ScottPlot.WinForms.FormsPlot graphPlot = new ScottPlot.WinForms.FormsPlot();
    FlowLayoutPanel layoutPanel = new FlowLayoutPanel();
    FlowLayoutPanel daysLayoutPanel = new FlowLayoutPanel();

    Team team1 = new Team("Equipe1");
    Team team2 = new Team("Equipe2");
    team1.teamScores = new List<(double score, double numberOfDay)>
    {
        (89,1),(100,2)
    };
    team2.teamScores = new List<(double score, double numberOfDay)>
    {
        (110,1),(95,2)
    };
    List<Team> teams = new List<Team>();
    Graph graph = new Graph(form, teams, graphPlot, layoutPanel, daysLayoutPanel);
    graph.daysToShow = new List<(double, string)>
    {
        (1, "lundi"), (2, "mardi")
    };
    ///Act
    teams.Add(team1);
    teams.Add(team2);
    graph.createGraph(teams);
    ///Assert
    Assert.IsTrue(graphPlot.Plot.GetPlottables().ToList().Count() == layoutPanel.Controls.OfType<CheckBox>()
        .ToList()
        .Where(c => c.Checked)
        .ToList()
        .Count(), "Le nombre de ligne du graph et le nombre de checkBox cochée n'est pas le même");
}
```

   b. Les différents tests par les paires ont testé la cohérence d'affichage entre le filtre des équipes et du filtre sur jours (Capacité des deux types de filtre à être activer en même temps).
   c. Les différents tests réaliser par les utilisateurs ont permis d'identifier certains bugs liés à l'utilisation de plusieurs filtres, exemple :
      - Un crash de l'application avait lieu si tous les jours été décoché.

2.5 Ajout test possible

Bien que ces tests soient efficaces ils ne sont néanmoins pas suffisants dû à leur petit nombre, vous trouverez donc ci-dessous une liste de test pouvant être ajouter au programme afin de diminuer le risque de mauvaise logique des méthodes du code :

- Ajout d'un test sur la méthode CreateGraph afin de vérifier la logique sur le filtre concernant les jours sélectionner par l'utilisateur soit correct.
- Ajout d'un test sur la méthode loadFile afin de vérifier la gestion d'erreur dans le cas ou l'utilisateur charge un fichier. json non valide.
- Ajout d'un test sur la méthode CreateGraph afin de vérifier que les données dans le graphique soit bien basé sur la liste d'équipe qui lui est fournie.

2.6 Donnée à prévoir

Lien vers un fichier. json correctement formé et remplit afin de pouvoir inscrire le lien du fichier dans le test de la méthode loadFile.

3 Réalisation

3.1 Points de design spécifiques

Durant la réalisation du projet, plusieurs contraintes et surprise sont apparue, ces contraintes ont pu ralentir le développement de l'application voir changer la logique même de celle-ci afin de les résoudre/contourné.

Vous trouverez donc ci-dessous la liste des plus grandes contraintes rencontrée.

3.1.1 Récupération des données via API

Contexte/Problème

La première étape à avoir été implémentée était la lecture des données via un fichier. json cependant, au départ, une API devait être utiliser, celle-ci n'a néanmoins pas pu être utiliser du au format monétaire de celle-ci, en effet, pour pouvoir récupérer les données des saison passée un abonnement payant était nécessaire et les données de la saison actuelle ne pouvait pas non plus être utiliser dû à l'intersaison.

Conséquence

Cet imprévu a forcé le redémarrage du projet deux fois, la première fois 3 semaine après le début du projet afin de lire un fichier .CSV puis une deuxième fois 3 jours plus tard afin de lire un fichier. JSON, ce changement a été effectué afin de rendre compatible les données lues par l'application et les données récupérer depuis l'API afin facilité son intégration dans le futur.

3.1.2 Conversion date format chaine de caractère en numéros de jour

Contexte

Suite de la récupération des données brut à partir du fichier. JSON, il était nécessaire de les filtrée afin de ne garder que les données souhaitées à savoir :

- Nom de l'équipe.

- Liste contenant tous les scores de l'équipe et le numéro du jour de la semaine ou le score a été effectué.

Problème
Cependant les dates du fichier sont sous un format particulier comme ceci :

```
"Date": "Tue Oct 24 2023",
```

Solution
Une méthode « ConvertStringDateToNumberOfDay » a donc été créée afin d'effectuer la conversion.

3.1.3 Création d'un graphique

Nécessité
Lors de la création du graph :
- Ordonnée de manière croissante la liste des scores en fonction du numéro du jour du score.
- Conversion de toutes les valeurs numériques en « double » nécessaire au fonctionnement de Scott Plot.

3.1.4 Affichage des jours sur l'axe X du graphique

Problème
Un imprévu a été rencontré lors de la création du graphique, en effets Scott Plot n'accepte que des valeurs de type « double » sur l'axe X.

Solution
Après une heure de recherche la méthode « SetTicks » à été découvertes permettant d'afficher un tableau de « string » tant qu'un tableau de « double » permet de servir d'index.

Exemple :

```
globalGraph.Plot.Axes.Bottom.SetTicks(dayOfWeek, daysToShow.Select(d => d.dayName).ToArray());
```

- dayOfWeek sers de tableaux d'index.
- daysToShow… sers de tableaux de « string ».

3.1.5 Filtre des jours affichée

Problème
Scott Plot ne permet pas de rendre invisible un point précis d'une ligne.

Conséquence
Il est donc impossible de désafficher un jour sans recréer toute la ligne.

Solution
La solution a donc été de modifier la création d'une ligne afin qu'elle prenne en compte les jour voulant être affiché par l'utilisateur voici les étapes effectuées afin d'y parvenir :

- Création d'une liste contenante tous les jours ainsi que leurs numéros.
- Création d'une liste contenante tous les jours coché par l'utilisateur.
- Attribution d'un numéro de jour à chaque jour coché a partir de la liste de base.

```
globalDaysFlowLayoutPanel.Controls.OfType<CheckBox>().ToList().ForEach(c =>
{
    if (c.Checked)
    {
        daysChecked.Add((days
            .Where(d => d.dayName == c.Text)
            .Select(d => d.dayOfWeekNumber).Single(), c.Text));
    }
});
```

- Suppression de toutes les lignes du graphique.
- Création de nouvelle ligne en prenant en compte du nombre de jour à affiché.

```
createPlot(t.teamScores
    .Where(ts => daysToShow
    .Select(d => (double)d.dayOfWeekNumber)
    .Contains(ts.numberOfDay))
    .Select(ts => (double)ts.numberOfDay)
    .ToArray(),
    t.teamScores
    .Where(ts => daysToShow
    .Select(d => (double)d.dayOfWeekNumber)
    .Contains(ts.numberOfDay))
    .Select(ts => (double)ts.score)
    .ToArray(), t.nameOfTeam, i);
i++;
```

- Désafficher les équipes n'étant cochée.

3.1.6 Filtre des équipes affichée

Problème

Scott Plot ne permet pas de savoir quand l'utilisateur clique sur un des labels de la partie Label de graph.

Conséquence

- Impossibilité de gérer les filtres d'équipe directement dans la partie label et obligation de créer une partie dédiée dans le form.
- Impossibilité de réaliser certains tests de la story « Filtrage des ligne ».

Solution

Création d'un panelle dédié contenant des checkboxs dans le form afin de savoir quelle équipe afficher.

3.2 Déroulement

Majoritairement les user stories se sont bien passé sans problèmes cependant certaines d'entre elle ont été ponctué par des imprévus :

3.2.1 Choix du fichier de graph

Contexte
Bien que la réalisation de cette story à été plutôt rapide, son ajout a créé certains bugs dans le code spécialement quand l'utilisateur charge un fichier et que le graph n'est pas vide.

Problème
- Création de ligne par de dessus les anciennes sans les supprimer.
- Création des certaines checkbox en doublant.

Conséquence
- Filtre inutilisable.
- Ralentissement du programme.
- Diminution de la lisibilité du graphique.

Solution
Suppression des contenus des flowLayoutPannel et du graphique dès qu'un fichier est chargé.

Conclusion
En conclusion cette story s'est bien passé car la solution au problème était simple et intuitive cependant il est important de noter qu'il serait dans le future nécessaire de changer le code afin d'exclure la création des différentes checkbox de la class « Graph » et rendre leur création uniquement dans la class « Form1 ».

3.2.2 Filtrage des lignes

Contexte
Le filtrage des lignes devant être affiché est une des fonctions principales du programme et bien que le filtre des équipes à affiché s'est passé sans problème le filtre des jours a lui posé bien plus de problème que prévu.

Problème
- Scott Plot ne permet pas de rendre invisible un point précis d'une ligne.
- Nécessité de recréer toutes les lignes du graphe.
- Modification de la création de des lignes du graphique afin de prendre en compte les jours devant être affiché nécessaire.

Solution
- Modification de la méthode « CreateGraph » afin de l'adapter au nombre de jour devant être affiché.
- Ajout d'une méthode « UpdatesTeamsVisibility » afin d'actualiser la visibilité des lignes venant d'être créés.

Conclusion
Cette story a été la plus longue a effectué, considérer comme simple au départ l'impossibilité de manipuler la visibilité des points d'une ligne l'a rendue bien plus complexe et longue qu'attendue.

3.2.3 Graphe à multiple ligne

Contexte
Bien qu'intuitive et logique cette story reste néanmoins obligatoire et nécessaire pour le bon fonctionnement du programme.

Conclusion
Cette story n'a posé aucun réel défi du a son évidence et sa logique simple de plus Scott Plot étant naturellement conçus pour gérer plusieurs lignes cette story a été très facilement réalisée.

3.3 Mise en place de l'environnement de travail

Tout l'environnement de travail à été placé sur GitHub, le répertoire est subdivisé en quatre partie :

- App est le répertoire contenant l'entièreté du code source de l'application.
- Docs contient toute la documentation du projet du journal de travail au rapport de projet.
- Maquette contient toutes les maquettes nécessaire a la compréhension du projet.
- TeamData contient les différents fichiers de donnée ayant été utiliser à différent moment du projet.

3.4 Description des tests effectués

3.4.1 Sprint 1

3.4.1.1 Choix du fichier de graph

Affichage du menu de choix de fichier	Sur la page principale Quand l'utilisateur clique sur le bouton "Choisir un fichier" Un menu permettant de choisir un fichier apparait (Voir maquette)	OK 30 Oct
Extension des fichier	Sur la page principale Quand l'utilisateur à cliquer sur "Choisir un fichier" et que l'explorateur de fichier s'ouvre l'utilisateur ne peut choisir que des fichiers de type JSON	OK 30 Oct
Mise à jour du graph	Sur la page de graph Quand l'utilisateur change de fichier Les statistique affichée sur le graph s'adapte au fichier choisi	OK 30 Oct
Affichage par défaut	Sur la page de graph Quand l'utilisateur ne choisit pas de fichier Le graph est vide	OK 30 Oct
Affichage des filtres par équipe	Sur la page de graph Quand un utilisateur choisi un fichier La checkbox des équipes contient autant d'équipe qu'il y d'équipe dans le fichier	OK 30 Oct

3.4.1.2 Graphe à multiple ligne

Nombre de Ligne sans filtre	Sur la page de graphique Quand il est affiché et que aucun filtre n'est appliqué L'utilisateur voit 30 courbes différentes	OK 30 Oct
Affichage label	Sur la page de graphique Quand il est affiché et que aucun filtre n'est appliqué L'utilisateur voit une partie label contenant autant de label que de courbe	OK 30 Oct

Zoom sur graph	Sur la page de graphique Quand l'utilisateur scroll vers l'avant avec la molette sur le graph Le graph zoom	OK 30 Oct
Dézoom sur graph	Sur la page de graphique Quand l'utilisateur scroll vers l'arrière avec la molette sur le graph Le graph dézoom	OK 30 Oct

3.4.1.3 CheckBox et Ligne Adaptative

Nombre de Ligne	Sur le graphique quand aucun filtre n'est appliqué je vois autant de ligne que d'équipe dans mon jeu de donnée	OK 1 Nov
Nombre de checkBox	Sur le page du graphique Quand un fichier à été chargé je vois autant de checkbox d'équipe que d'équipe dans mon jeu de donnée et je vois une checkBox supplémentaire permettant de tout supprimer.	OK 1 Nov
Nom des checkBox	Quand les checkbox sont affichée le nom des checkBox correspond aux noms des équipes de mon fichier de donnée	OK 1 Nov

3.4.1.4 Filtrage des lignes

Check box jour	Sur la page du graphe Quand le graph est affiché et qu'aucun filtre n'est choisi La checkBox affiche que tous les jours sont sélectionné	OK 30 Oct
Check box équipe	Sur la page du graphe Quand le graph est affiché et qu'aucun filtre n'est choisi La legendes des équipes contient autant d'équipe que d'équipe sélectionnée	OK 1 Nov
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique sur une équipe dans la partie label Le graph affiche uniquement l'équipe sur laquelle l'utilisateur a cliqué	ko 30 Oct
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique sur une équipe dans la partie label La couleur de la ligne affichée est la même que la couleur de l'équipe ayant était cliquée dans la partie label	ko 30 Oct
Affichage une équipe	Sur la page du graphe Quand l'utilisateur clique plusieurs fois sur la même équipe dans la partie filtre Les informations de la ligne ne changent pas	OK 1 Nov
Selection des équipes a ne pas afficher	Sur la page du graphe Quand l'utilisateur décoche la checkBox d'une équipe L'équipe d'ont la checkbox à été décocher disparait	OK 30 Oct
Selection des équipes à afficher	Sur la page du graphe Quand l'utilisateur coche la checkBox d'une équipe L'équipe d'ont la checkbox à été coché apparait	OK 30 Oct
Affichage jour	Sur la page du graphe Quand l'utilisateur coche la checkBox d'un jour la statistique qui correspond au jour apparait	OK 30 Oct
Désaffichage jour	Sur la page du graphe Quand l'utilisateur décoche la checkBox d'un jour la statistique qui correspond au jour disparait	OK 30 Oct

3.5 Erreurs restantes

Plus aucune erreur n'est présente dans le projet, ce qui veut dire que sa compilation s'effectue sans problème, cependant il y a plusieurs points suscitant de la refactorisation afin d'améliorer les performance et l'évolutivité du code.

3.5.1 Dette technique

UpdatesTeamsVisibility

Le code de la méthode UpdatesTeamsVisibility est redondant avec une partie du code de la méthode CheckBox_CheckedChanged la solution serait de placer ce code redondant dans une méthode unique et de l'appeler quand nécessaire, il serait aussi nécessaire de n'utiliser cette méthode que quand cela est nécessaire.

Méthode 1

```
private void UpdatesTeamsVisibility()
{
    //TODO: Refactore pour que le code de cette methode et la methode CheckBox_CheckedChanged ne soit plus redondant
    //Liste des équipe à ne pas afficher
    globalFlowLayoutPanel.Controls.OfType<CheckBox>().ToList().Where(c => !c.Checked).ToList().ForEach(c =>
    {
        globalGraph.Plot.GetPlottables().ToList().ForEach(scatter =>
        {
            if (scatter.LegendItems.Select(s => s.LabelText).FirstOrDefault().ToString() == c.Text)
            {
                scatter.IsVisible = !scatter.IsVisible;
                globalGraph.Refresh();
            }

        });
    });
}
```

Méthode 2

```
private void CheckBox_CheckedChanged(object sender, EventArgs e)
{
    CheckBox changedCheckBox = sender as CheckBox;
    globalGraph.Plot.GetPlottables().ToList().ForEach(scatter =>
    {
        if (scatter.LegendItems.Select(s => s.LabelText).FirstOrDefault().ToString() == changedCheckBox.Text)
        {
            // Inverse la visibilité de la ligne
            scatter.IsVisible = !scatter.IsVisible;
            globalGraph.Refresh();
        }

    });
}
```

Création des checkbox

Un des points les plus important à changer dans le code est l'emplacement des méthodes de création des checkbox afin de les rendre externe a la class graph.cs et de les déplacer dans le form lui-même.

CreatePlot

Cette méthode est celle qui crée chaque ligne du graphique, cependant, le type de ligne change en fonction du nombre de jour a affiché, il serait donc utile de rendre

CreatePlot supérieur et placer chaque type de ligne dans une méthode différente qui serait un paramètre de la méthode CreatePlot.

Code

```
private void createPlot(double[] dayOfWeek, double[] nbpointArray, string nameOfTeam,double indexOfTeam)
{
    if(dayOfWeek.Length > 1)
    {
        var scatt = globalGraph.Plot.Add.Scatter(dayOfWeek, nbpointArray);
        scatt.LegendText = nameOfTeam;
    }
    else
    {
        //TODO:Changer les labels de l'axe X pour inclure le nom des équipes
        var coloumn = globalGraph.Plot.Add.Bar(indexOfTeam, nbpointArray[0]);
        coloumn.LegendText = nameOfTeam;
    }
    // Evite la redondance des checkBox (31 = le nombre d'equipe + le bouton tout supprimer)
    if(globalFlowLayoutPanel.Controls.Count < 31)
        CreateTeamCheckBoxes(nameOfTeam);
    globalGraph.Plot.Axes.Bottom.SetTicks(dayOfWeek, daysToShow.Select(d => d.dayName).ToArray());
    globalGraph.Plot.Axes.AutoScale();
}
```

LoadGraphic

Cette méthode charge le graphique dans le form cependant étant donné la manière dont les checkbox sont créer à chaque fois que cette méthode est appelée tous les FlowLayoutPannel doivent être vidé pour ne pas causer de problème quand un nouveau fichier est chargé il serait donc préférable de modifier cette partie afin de ne plus avoir a vidé les différents panneaux.

Code

```
public void createGraph(List<Team> teams)
{
    double i = 0;
    this.globalTeams = teams;
    // Label du graph
    globalGraph.Plot.XLabel("Jour de la semaine");
    globalGraph.Plot.YLabel("Nombre de points");
    //Crée une ligne pour chaque équipe
    if(daysToShow.Count >=1)
    {
        teams.ForEach(t =>
        {
            t.teamScores = t.teamScores.OrderBy(ts => ts.numberOfDay).ToList();
            createPlot(t.teamScores
                .Where(ts => daysToShow
                .Select(d => (double)d.dayOfWeekNumber)
                .Contains(ts.numberOfDay))
                .Select(ts => (double)ts.numberOfDay)
                .ToArray(),
                t.teamScores
                .Where(ts => daysToShow
                .Select(d => (double)d.dayOfWeekNumber)
                .Contains(ts.numberOfDay))
                .Select(ts => (double)ts.score)
                .ToArray(), t.nameOfTeam, i);
            i++;
        });
    }
    else
    {
        return;
    }
    i = 0;
    // Rafraîchissement et ajout au form
    globalGraph.Refresh();
    globalForm.Controls.Add(globalGraph);
    //TODO: Ne faire appel a cette methode que si nécessaire (uniquement si les il n'y pas 7 jour à afficher et que certaines équipe sont déchoché)
    UpdatesTeamsVisibility();
}
```

3.5.2 Conséquences sur l'utilisation du produit

Malgré les dettes techniques les conséquences sur l'utilisation du produit sont nul, cependant il reste néanmoins important d'effectuer ces changements afin d'améliorer les performance et l'évolutivité du code.

4 Conclusions

4.1 Objectif atteints / non-atteints

Tous les objectifs principaux du projet ont été réalisé le logiciel permet de :

- D'importer un jeu de donnée sous forme d'un fichier .JSON.
- Créer un graphique s'adaptant au jeu de donné.
- Filtrer les informations du graphique sur la l'axe du temps et plus.
- Le programme utilise en quantité l'extension LINQ.
- Une documentation de projet est disponible.
- Une gestion de projet via IceScrum à été effectuée bien qu'insuffisante sur le début du projet.

4.2 Points positifs / négatifs

Bien que retissant au début, dû à mon inexpérience avec l'extension LINQ ce projet m'a permis de mieux la comprendre et de mieux me rendre compte de son utilité et efficacité, j'ai au fur et à mesure du projet appris à mieux la maîtriser et ait pu apprendre beaucoup de nouvelle chose utile tel que :

- OffType().
- Cast ().
- Key après un groupBy().
- L'utilisation de la variable sender lors de l'utilisation d'une checkbox

4.3 Difficulté particulière

Les principales difficultés que j'ai rencontré lors du projet on été lié a mon analyse préalable qui a été insuffisante, effectivement ma plus grosse erreur à été de ne pas avoir plus analysé l'API que j'ai choisie ce qui m'a couter un recommencement total du projet 3 semaine après son début.

Dans le future je ferai en sorte de me concentrer sur l'analyse des besoins et des données du projet avant de commencer le code afin d'éviter de rencontrer des problèmes provenant de la base même du projet de plus afin d'éviter de mauvaise surprise lors de la création du code je me concentrerai plus sur la création de user stories afin de mieux gérer la création des différentes méthodes.

4.4 ChatGPT

ChatGPT a été un outil précieux durant la totalité du projet, effectivement il m'a permis de découvrir pas mal de nouvelle méthode utile tel que :

- Cast ().
- OffType().

En plus de m'avoir permis d'accélérer la compréhension de certaines erreur et exception produite durant le projet.

Il m'a aussi permis de réaliser certaines tâches en un temps record tel que :

- La création d'un diagramme de classe et d'états à partir de mon code source.
- La génération des commentaires des différentes méthodes du programme.

4.5 Suite possible

Dans le cas d'une suite, le focus serait porté sur la création d'un volume permanent permettant à l'utilisateur de ne pas avoir à charger un fichier a chaque ouverture de programme, en plus de cela, cette fonctionnalité permettrait de mettre en place un historique des jeux de donné charger dans le programme afin de ne pas avoir besoin de les charger manuellement.

Un autre point important serait la mise en place d'une communication entre le programme et une API afin de pouvoir avoir une évolution des statistiques en temps réelle.

5 Annexes

5.1 Journal de travail

Voir le journal de travail se trouvant sur GitHub.