

Plot That Line

Julien Mares

Table des matières

1	Analyse préliminaire	4
1.1	Introduction	4
1.2	Objectifs	4
1.2.1	Programme manipulant et affichant des données.....	4
1.2.2	Planification claires (à l'aide des User Stories)	4
1.2.3	Utilisation de Github en bonne et due forme	4
1.2.4	Réalisation d'un rapport	4
2	Analyse / Conception.....	4
2.1	Domaine.....	4
2.2	Concept.....	4
2.2.1	Diagramme de class.....	4
2.2.2	Diagramme(s) d'état.....	6
2.3	Analyse fonctionnelle.....	6
2.3.1	Graphique température	6
2.3.2	Identification des courbes météorologiques par ville	7
2.3.3	Liste Ville	7
2.3.4	Triage des données à analyser.....	8
2.3.5	Synchronisation données souhaité avec les données local	8
2.4	Stratégie de test.....	8
2.4.1	Test d'acceptance	8
2.4.2	Test unitaires	8
3	Réalisation	9
3.1	Points de design spécifiques.....	9
3.2	Déroulement	9
3.2.1	User Story - Graphique température	9
3.2.2	User Story – Identification des courbes météorologiques par ville.....	9
3.2.3	Technical Story – Synchronisation données souhaité avec les données locales 9	
3.2.4	User Story – Triage des données à analyser	10
3.2.5	User Story - Liste Ville.....	10
3.3	Mise en place de l'environnement de travail.....	10
3.3.1	Accès au code source	10
3.3.2	Les fichiers du programme.....	10
3.3.3	Outils utilisés	11
3.4	Description des tests effectués	11
3.5	Erreurs restantes	13
3.5.1	TODO #1 Manque 1er jour de la période sélectionné	13
4	Conclusions	13
4.1.1	Objectifs atteints / non-atteints.....	13
4.1.2	Points positifs / négatifs	13
4.1.3	Difficultés particulières	13
4.1.4	Suite possible	13
5	Annexes	14

5.1	Journal de travail	14
-----	--------------------------	----

1 Analyse préliminaire

1.1 Introduction

Ce projet, effectué dans un cadre scolaire à l'ETML, permet l'approfondissement des connaissances en c#, et est un premier pas vers la programmation fonctionnelle ainsi que l'utilisation de LINQ. L'objectif visé était de faire un programme allant manipuler des données avec LINQ et en les affichants de manière graphique, Dans ce projet (C# WinForms) les données que nous allons traiter sont des données météorologiques qu'on va chercher via l'API « openmeteo ». Elles sont traitées avec LINQ et l'affichage se fait via l'extension ScottPlot.

1.2 Objectifs

1.2.1 Programme manipulant et affichant des données

Programme c# Win Forms, utilisant les extensions LINQ et ScottPlot, LINQ servira à manipuler et traiter les données tandis que ScottPlot

1.2.2 Planification claires (à l'aide des User Stories)

Avoir une vision claire du projet grâce à des User Stories compréhensible et des tests d'acceptance en suffisance.

1.2.3 Utilisation de Github en bonne et due forme

Les commits Github doivent être atomique et bien nommé, le gitignore est correcte et permet une bonne gestion des fichiers à versionner

1.2.4 Réalisation d'un rapport

Le rapport répond à tous les points qu'il aborde avec précision.

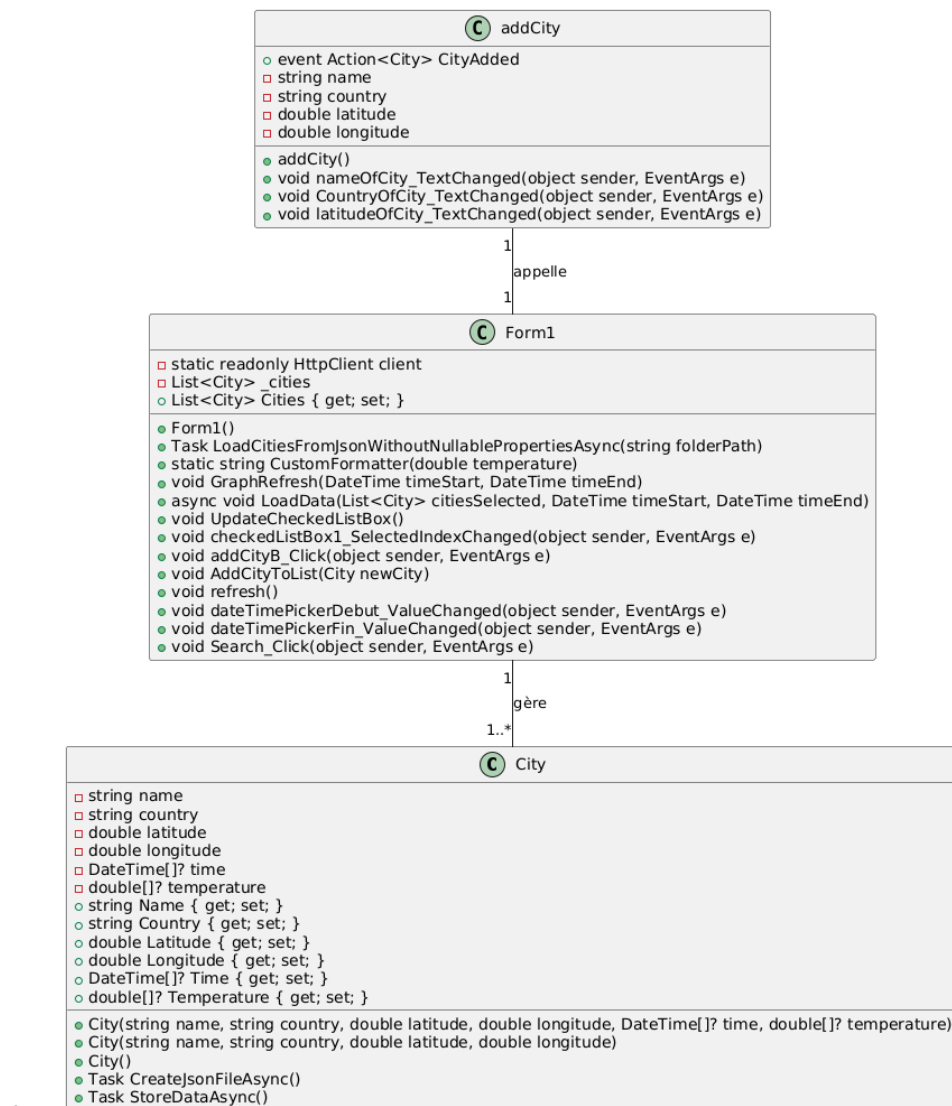
2 Analyse / Conception

2.1 Domaine

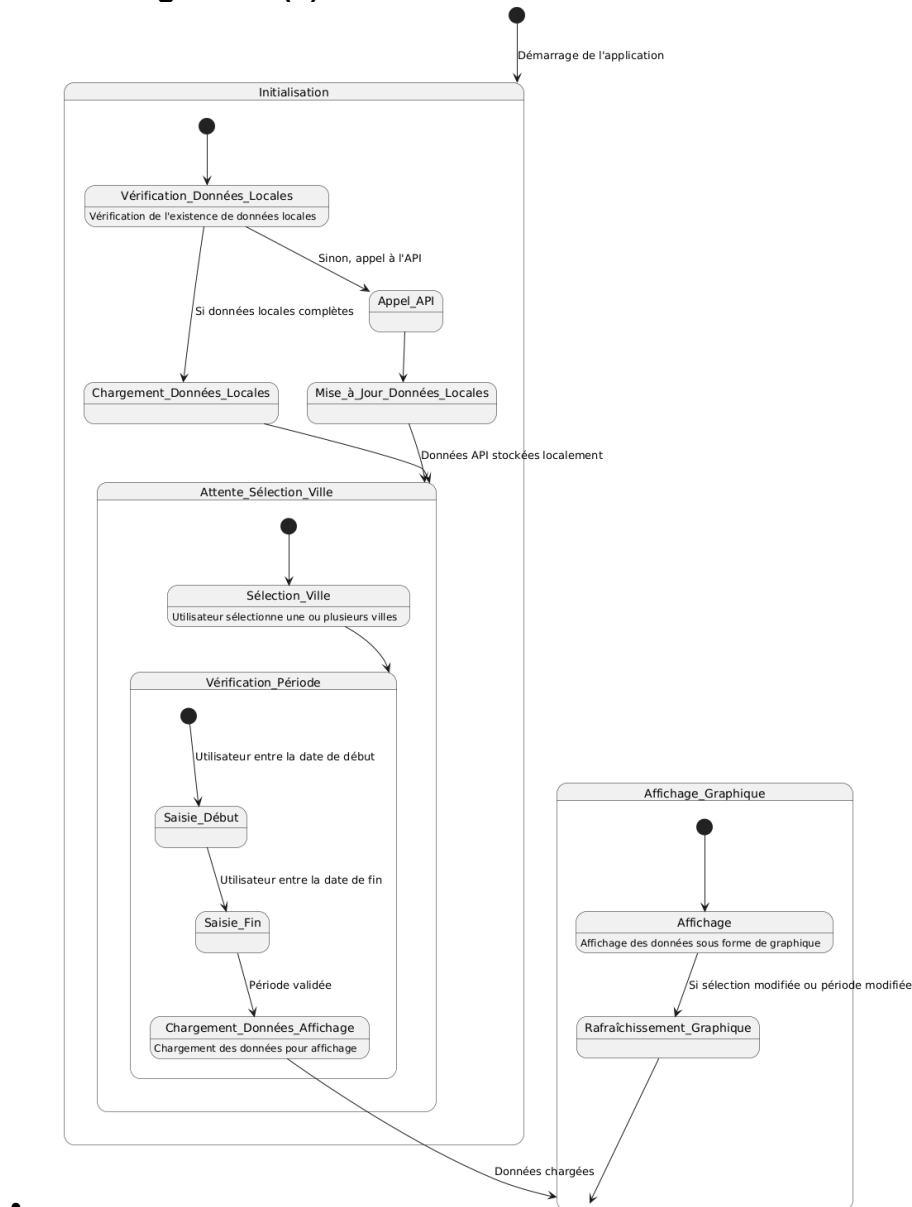
Les données utilisées concernent la météo partout dans le monde depuis 1950, elles représentent la température d'un endroit, définis par sa latitude et sa longitude pour chaque jour à minuit. Pour ces données l'échelle de temps pertinentes dépend de son utilisateur, un « data analyser » pourrait vouloir analyser les données depuis 50 ans alors qu'une mère de famille pourrait vouloir se remémorer la température de ses dernières vacances à Paris. C'est donc un programme sans publique cible définis mais où chaque utilisateur va l'utiliser à sa manière.

2.2 Concept

2.2.1 Diagramme de class



2.2.2 Diagramme(s) d'état



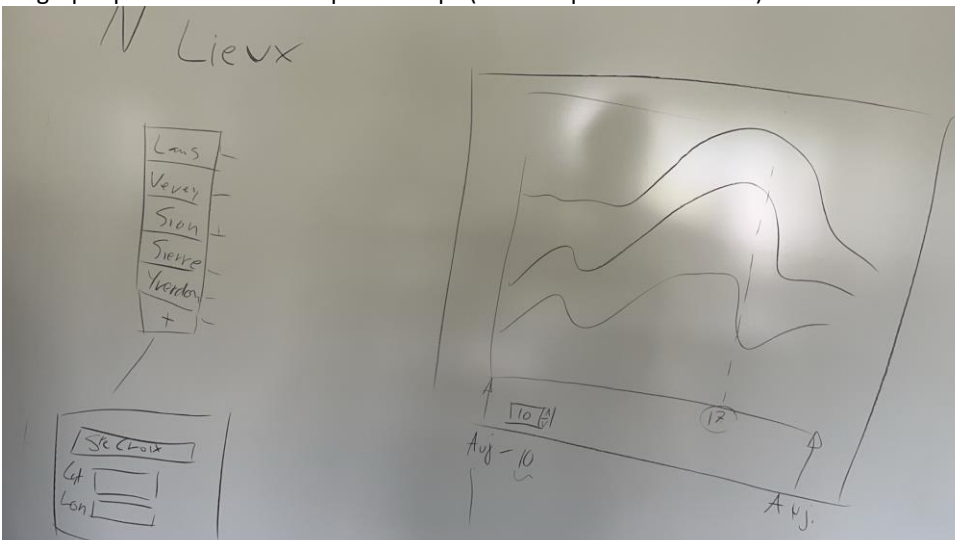
2.3 Analyse fonctionnelle

2.3.1 Graphique température

En tant qu'utilisateur Je veux pouvoir chercher la température d'un endroit selon une certaine période Afin d'avoir une vision globale de l'évolution climatique à travers le temps

Tests d'acceptance:

Echelle temps	Quand je suis sur l'application Je peux recevoir les données météo d'aujourd'hui jusqu'à X nombres de jours dans le passé (x est choisis par l'utilisateur) Cela me fait
---------------	--

un graphique concernant ce laps de temps (Voir Maquette ci-dessous)	
	
Lisibilité	Etant en face du graphique Je peux lire les unités du graphe Afin de pouvoir me faire une idée des valeurs
Compréhensibl e	Etant en face du graphique Je comprends facilement quelle courbe correspond à quelle ville

2.3.2 Identification des courbes météorologiques par ville

(Auteur: Julien Mares)

En tant qu'utilisateur Je veux pouvoir identifier facilement à quelle ville correspond chaque courbe sur le graphique Afin de pouvoir comparer les données météorologiques entre plusieurs villes sans confusion.	
Tests d'acceptance:	
Affichage des checkboxes colorées	Étant donné que j'ai une liste de villes disponibles, Quand j'affiche les checkboxes pour chaque ville, Alors chaque checkbox a une couleur unique qui correspond à la couleur de la courbe sur le graphique.
Sélection checkbox	Étant donné que je coche ou décoche une checkbox, Quand je l'a regarde Alors je remarque que son apparence change (couleur / icône check / etc..)
Différenciation visuelle des courbes	Étant donné que plusieurs courbes sont affichées pour différentes villes, Quand je regarde le graphique, Alors chaque courbe est distincte par sa couleur pour éviter toute confusion.

2.3.3 Liste Ville

(Auteur: Julien Mares)

En tant qu'utilisateur J'ai besoin d'une liste de ville que je peux sélectionner Afin de connaître leurs données météo correspondantes	
Tests d'acceptance:	
Selection ville	Sur la liste des villes Quand je clique sur une ville Cela montre qu'elle est sélectionné (icône Check ou fond de couleur par exemple)
Affichage graphique données ville	Une fois que j'ai sélectionné les villes que je voulais Quand je vais sur le graphique Il y a une courbe correspondante à chacune des villes sélectionnées
Ajout ville	Quand je suis sur la page d'accueil du programme Quand je clique sur le bouton "+" ou "Ajouter une ville", en dessous de la liste Cela m'ouvre un formulaire où il faut remplir la latitude et longitude de la ville
Ajout ville 2	Une fois que j'ai remplis le formulaire d'ajout d'une ville en mettant sa latitude et longitude Quand je clique sur "Ajouter la ville" Cela me ferme le formulaire et la ville est ajoutée à la liste

2.3.4 Triage des données à analyser

(Auteur: Julien Mares)

En tant qu'utilisateur J'aimerais pouvoir modifier certains paramètres de trie Afin que le graphique affiche les courbes selon mes souhaits

Tests d'acceptance:

date	Quand je suis sur l'application Je sélectionne une date de début et de fin Cela modifie le
Picker	graphique en faisant en sorte qu'il englobe la durée entre ma date de début et de fin
City	Quand je suis sur l'application Je sélectionne différentes villes (1 ou n) Cela m'affiche
selection	plusieurs courbes sur le graphique correspondante chacune d'elle à une ville

2.3.5 Synchronisation données souhaité avec les données local

(Auteur: Julien Mares)

Récupérer tous les fichiers json en local et en faire des objets

Tests d'acceptance:

Récupération légère	Quand mon programme se lance Il va vérifier dans un dossier où son stocké tout les ajouts de ville de l'application Afin d'aller récupérer les données (nom ville, pays, latitude, longitude) et en crée un objet pour chacun
Affichage courbe (Si fichier json contient les données requis)	Etant donné que j'ai sélectionné une ville et des dates Quand j'appuie sur le bouton (Search qui execute la méthode loaddata) Cela va remplir l'objet de la ville avec les données météo contenu dans le fichier json de la ville correspondantes
Affichage courbe (Si fichier json ne contient pas les données requis)	Etant donné que j'ai sélectionné une ville et des dates Quand j'appuie sur le bouton (Search qui execute la méthode loaddata) Cela va remplir essayer de remplir l'objet de la ville avec les données météo contenu dans le fichier json, mais si elles ne sont pas suffisantes on va faire un appel à l'API afin de récupérer les données manquante.
Mise à jour des données local	Étant donné que des données manquent dans le fichier JSON, Lorsque l'application les récupère via l'API, Alors elle met à jour correctement le fichier JSON local.

2.4 Stratégie de test

2.4.1 Test d'acceptance

Il y a plusieurs tests d'acceptance pour chacune des User Stories, ils sont effectué par un dev qui va s'il est réussi cocher le test est réussi ou non. Evidemment ces tests ne sont pas exhaustif, on peut toujours en rajouter mais il est important de savoir à quel moment les tests sont suffisants. Les tests d'acceptance, dans des projets fait pour un client doivent obligatoirement être valider par le client.

2.4.2 Test unitaires

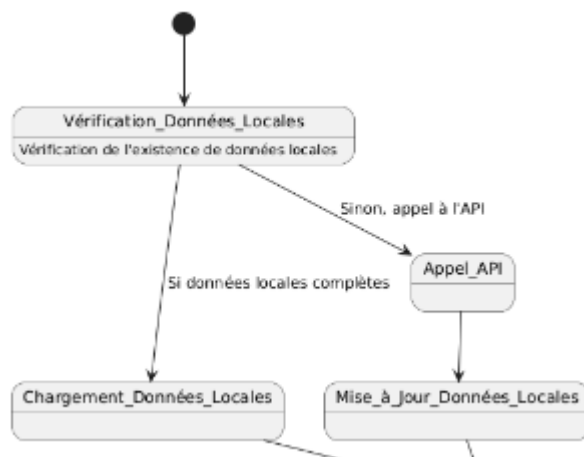
Les tests unitaires sont des tests sous forme de code que l'on va exécuter afin de s'assurer que le code (une méthode par exemple) fonctionne correctement, ils doivent être effectué régulièrement afin de prévenir le plus tôt possible si un changement dans le code a eu un impact négatif insoupçonnée. Les tests unitaires ne doivent pas vérifier absolument tous les cas possibles forcément mais on cherche une suffisance, on peut par exemple tester les extrêmes et 2-3 cas ordinaires, s'ils réussissent on part du principe que c'est bon.

3 Réalisation

3.1 Points de design spécifiques

Dans mon programme, qui utilise des milliers des données chaque requêtes (potentiellement), la rapidité pour aller chercher les données est primordial. Cependant faire une requête API à chaque fois que l'utilisateur fais ou change sa demande est excessif, imaginons que Patrick recherche les données météorologiques du 1 janvier 2004 au 1 janvier 2024 pour 3 villes, Lausanne, Paris et Rome et qu'ensuite il change d'avis, il veut maintenant les données jusqu'au 4 janvier 2024 aller rechercher des dizaines de milliers pour simplement ajouter 4 jours au graphique est aberrant.

C'est pour cela que le programme va stocker les données demandées par l'utilisateur localement sous forme de fichier json, ainsi s'il refait une requête dont les données sont stockées sur la machine elle sera nettement plus rapide.



En procédant comme cela, on améliore la vitesse du programme dans le cas où la machine possède les données souhaitées mais aussi, on devient moins dépend de l'API.

3.2 Déroulement

3.2.1 User Story - Graphique température

Cette User Story a été plutôt simple à mettre en place, grâce à ScottPlot tout est fait rapidement et simplement avec un résultat ergonomique, c'est un première pour moi d'afficher des données avec un programme et j'ai été agréablement surpris par ScottPlot. Malheureusement un des tests d'acceptance n'a pas été respecté, pour le test d'acceptance "Echelle temps" j'ai finalement changé de solution par rapport à celle demandé par le test pour plusieurs raisons. Premièrement l'API utilisé ne contient pas les données des 2 jours précédents donc il est impossible d'avoir les données d'aujourd'hui et hier, deuxièmement je pense qu'un utilisateur de cette application ne l'utiliserait pas pour connaître forcément la météo des jours passé récent, mais il l'utiliserait plus pour en tirer des conclusions sur des laps de temps de temps antérieur. C'est pour cela que j'ai opté pour une solution où l'utilisateur peut sélectionner un laps de temps en mettant une date de début et une date de fin.

3.2.2 User Story – Identification des courbes météorologiques par ville

Cette User Story aussi a été plutôt simple, toujours grâce à ScottPlot, malheureusement le premier test d'acceptance, "Affichage des checkboxes colorées"

a été raté car il n'est pas possible de modifier la couleur d'un seul item dans une checkBoxList (on peut modifier la couleur du tout mais pas chaque item différemment), mais en réalité il y a une légende pour chaque courbe sur le graphique qui donne la couleur de la courbe correspondantes, donc on obtient quand même l'informations voulus via une feature de ScottPlot qui résous plutôt bien notre problème.

3.2.3 Technical Story – Synchronisation données souhaité avec les données locales

Cette partie-là du site m'a pris pas mal de temps non pas à cause de sa complexité mais du fait que jusqu'à un bon moment dans le projet je ne m'étais pas posé et définis cette Story ainsi que ses tests d'acceptance, je travaillais sans penser à cette question ce qui faisais que je m'emmêlais les pinceaux. Finalement une fois que j'ai mis à plat mes pensées en créant cette Story tout m'a paru plus claire et simple, elle m'a vraiment montré une force potentielle de la méthode Scrum.

3.2.4 User Story – Triage des données à analyser

Elle a été simple à mettre en place, hormis un petit problème que j'ai rencontré, lorsque qu'on utilise un date Picker de Win forms il faut savoir qu'en cliquant dessus afin de l'ouvrir et de sélectionner une date, cela appelle directement la méthode "dateTimePickerFin_ValueChanged" qui est censé être appelé à un changement de date en lui donnant une valeur pas décidé par l'utilisateur, il a donc été important de spécifier au programme de ne pas exécuter la méthode cité plus haut si la date de début est après, chronologiquement parlant, la date de fin. Sinon cela fait planter le programme pour des raisons de logique dans le code.

3.2.5 User Story - Liste Ville

User Story simple à mettre en place, la seule vraie difficulté a été de pouvoir ajouter à la liste Cities la ville ajoutée via le formulaire addCity, pour cela il a fallu crée une méthode qui renvoie l'objet city crée dans le formulaire et ensuite crée une méthode qui récupère un objet city et l'ajoute à la liste.

3.3 Mise en place de l'environnement de travail

3.3.1 Accès au code source

En installant la release de ce repos : <https://github.com/JulienETML-hub/Plot-that-line>, en l'extrayant, vous trouverez le code source dans le dossier PlotThatLine2/Code

3.3.2 Les fichiers du programme

- **.vs/** - Dossier contenant des fichiers de configuration spécifiques à Visual Studio. Utilisé pour stocker les informations de débogage et de session.
- **bin/** - Dossier contenant les fichiers compilés (exécutables et bibliothèques) générés après la compilation du projet.
- **datasets/** - Dossier contenant les fichiers de données, probablement des fichiers JSON ou CSV, utilisés pour stocker les données météorologiques des villes.
- **obj/** - Dossier contenant les fichiers temporaires créés lors de la compilation, utilisé pour le build incrémental de Visual Studio.
- **TestResults/** - Dossier contenant les résultats des tests unitaires exécutés sur le projet.

- **addCity.cs** - Fichier source C# contenant le code pour gérer l'ajout de villes dans l'application.
- **addCity.Designer.cs** - Fichier généré automatiquement pour définir l'interface utilisateur de la fenêtre d'ajout de villes.
- **addCity.resx** - Fichier de ressources contenant des informations comme les chaînes de texte pour l'interface utilisateur dans **addCity.cs**.
- **City.cs** - Fichier source C# définissant la classe **City**, avec les propriétés comme le nom de la ville et les données météo associées ainsi que les méthodes gérant les données locales (création fichier, chargement données).
- **Form1.cs** - Fichier source C# contenant le code pour la fenêtre principale de l'application, incluant la logique de chargement et d'affichage des données.
- **Form1.Designer.cs** - Fichier généré automatiquement pour l'interface de la fenêtre principale (**Form1**), avec les éléments graphiques et leur disposition.
- **Form1.resx** - Fichier de ressources pour la fenêtre principale, utilisé pour stocker les chaînes de texte et autres ressources de l'interface.
- **PlotThatLine2.csproj** - Fichier de projet C# qui contient les informations de configuration du projet, comme les dépendances, les versions, et les options de build.
- **PlotThatLine2.csproj.user** - Fichier de configuration spécifique à l'utilisateur, qui stocke les paramètres personnalisés pour le projet (non essentiel pour le code).
- **PlotThatLine2.sln** - Fichier de solution Visual Studio qui contient les informations de la solution, listant tous les projets inclus (ici, **PlotThatLine2**).
- **Program.cs** - Point d'entrée principal de l'application C#. Contient la méthode **Main()** pour démarrer l'application.

3.3.3 Outils utilisés

- *Windows 10*
- *IceScrum*
- *Suite Office*
- Visual Studio 2022 (Projet : Application Windows Forms C#) [extension: LINQ, ScottPlot (WinForms), Newtonsoft.Json]
- *PlantUML.com*
- *GitHub (github Desktop)*

3.4 Description des tests effectués

3.4.1.1 [Triage des données à analyser](#)

date Picker	Quand je suis sur l'application Je sélectionne une date de début et de fin Cela modifie le graphique en faisant en sorte qu'il englobe la durée entre ma date de début et de fin	OK 25 Oct
City selection	Quand je suis sur l'application Je sélectionne différentes villes (1 ou n) Cela m'affiche plusieurs courbes sur le graphique correspondante chacune d'elle à une ville	OK 25 Oct

3.4.1.2 Identification des courbes météorologiques par ville

Affichage des checkboxes colorées	Étant donné que j'ai une liste de villes disponibles, Quand j'affiche les checkboxes pour chaque ville, Alors chaque checkbox a une couleur unique qui correspond à la couleur de la courbe sur le graphique.	KO 26 Oct
Selection checkbox	Étant donné que je coche ou décoche une checkbox, Quand je l'a regarde Alors je remarque que son apparence change (couleur / icone check / etc..)	OK 24 Oct
Différenciation visuelle des courbes	Étant donné que plusieurs courbes sont affichées pour différentes villes, Quand je regarde le graphique, Alors chaque courbe est distincte par sa couleur pour éviter toute confusion.	OK 24 Oct

3.4.1.3 Synchronisation données souhaité avec les données locales

Récupération légère	Quand mon programme se lance Il va vérifier dans un dossier où son stocké tous les ajouts de ville de l'application Afin d'aller récupérer les données (nom ville, pays, latitude, longitude) et en crée un objet pour chacun	OK 24 Oct
Affichage courbe (Si fichier json contient les données requis)	Etant donné que j'ai sélectionné une ville et des dates Quand j'appuie sur le bouton (Search qui execute la méthode loaddata) Cela va remplir l'objet de la ville avec les données météo contenues dans le fichier json de la ville correspondante	OK 24 Oct
Affichage courbe (Si fichier json ne contient pas les données requis)	Etant donné que j'ai sélectionné une ville et des dates Quand j'appuie sur le bouton (Search qui execute la méthode loaddata) Cela va remplir essayer de remplir l'objet de la ville avec les données météo contenues dans le fichier json, mais si elles ne sont pas suffisantes on va faire un appel à l'API afin de récupérer les données manquantes.	OK 24 Oct
Mise à jour des données local	Étant donné que des données manquent dans le fichier JSON, Lorsque l'application les récupère via l'API, Alors elle met à jour correctement le fichier JSON local.	OK 24 Oct

3.4.1.4 Liste Ville

Selection ville	Sur la liste des villes Quand je clique sur une ville Cela montre qu'elle est sélectionnée (icone Check ou fond de couleur par exemple)	OK 23 Oct
Affichage graphique données ville	Une fois que j'ai sélectionné les villes que je voulais Quand je vais sur le graphique Il y a une courbe correspondantes à chacune des villes	OK 23 Oct
Ajout ville	Quand je suis sur la page d'accueil du programme Quand je clique sur le bouton "+" ou "Ajouter une ville", en dessous de la liste Cela m'ouvre un formulaire où il faut remplir la latitude et longitude de la ville	OK 23 Oct
Ajout ville 2	Une fois que j'ai remplis le formulaire d'ajout d'une ville en mettant sa latitude et longitude Quand je clique sur "Ajouter la ville" Cela me ferme le formulaire et la ville est ajoutée à la liste	OK 23 Oct

3.4.1.5 Graphique température

Echelle temps	Quand je suis sur l'application Je peux recevoir les données météo d'aujourd'hui jusqu'à X nombres de jours dans le passé (x est choisis par l'utilisateur) Cela me fait un graphique concernant ce laps de temps (Voir Maquette sur GitHub)	ko 27 Oct
Lisibilité	Etant en face du graphique Je peux lire les unités du graphe Afin de pouvoir me faire une idée des valeurs	OK 25 Oct
Compréhensible	Etant en face du graphique Je comprends facilement quelle courbe correspond à quelle ville	OK 25 Oct

3.5 Erreurs restantes

3.5.1 TODO #1 Manque 1er jour de la période sélectionné

Quand on sélectionne une durée, la courbe affichée englobe la durée voulue sauf le premier jour. C'est une erreur simple à régler je pense malheureusement je n'ai plus le temps.

Les conséquences sur l'utilisateurs ne sont pas énormes, il n'a qu'à sélectionner un jour avant la période qu'il veut analyser et cela fera l'affaire.

Il faudrait refactoriser le code de manière à régler le problème.

4 Conclusions

4.1.1 Objectifs atteints / non-atteints

Programme manipulant et affichant des données	Atteint
Planification claires (à l'aide des User Stories)	Non-Atteint
Utilisation de Github en bonne et due forme	Atteint
Réalisation d'un rapport	Atteint

4.1.2 Points positifs / négatifs

Pour conclure, je suis plutôt content du résultat de mon projet même s'il est très simple et basique, mais je suis surtout très content car durant ce projet j'ai appris plein de choses sur lesquels je butais avant, par exemple je ne savais pas faire un gitignore et à chaque projet cela m'embêtait et j'ai enfin pris le temps de m'y atteler et je pense qu'à partir de maintenant je le ferai bien dès le début. Aussi je me rends compte maintenant (en fin de projets) qu'écrire les tâches sur IceScrum aurait été une très bonne option, bien que je ne l'ait pas vraiment fait durant ce projet en voyant l'efficacité d'IceTools je pense que pour les prochains j'essaierai de plus me poser au début et d'écrire des tâches claires dès le début. Au-delà de ces deux choses qu'on avait déjà plus ou moins appris par le passé (git, icescrum), j'ai apprécié travailler sur ce projet (Comme la plupart des projets de dev du genre) et j'ai pu imaginer comment la programmation fonctionnelle pourrait être utile, même si pour l'instant je trouve que c'est un très bon complément (à la POO par exemple), j'ai du mal à me projeter un projet 100% fonctionnel.

4.1.3 Difficultés particulières

L'utilisation de IceScrum au début, je pense que pour plusieurs raisons j'ai repoussé l'utilisation d'IceScrum ce qui m'a été un vrai poison durant le projet, la première raison est par peur de l'engagement je pense, je craignais d'écrire des tests et de devoir m'y tenir, deuxièmement par mon manque de connaissance en la matière (WinForms & ScottPlot), je ne savais pas vraiment ce que j'allais être capable de faire. Et finalement par l'attrance du code, qui a deux déclencheurs, le premier basique, je préfère coder que faire des User Stories, mais aussi au début du projet j'ai pris un jour entier pour choisir quelles données et API j'allais utiliser, j'avais l'impression d'être en retard sur le projet et cela m'a poussé à me précipiter.

4.1.4 Suite possible

Comme amélioration potentiel je vois le spectre de données à analyser, actuellement il analyse uniquement la température mais l'API utilisé permet d'accéder à plein d'informations qu'il pourrait mettre à profit. Et évidemment une refonte visuelle ne

serait pas de refus, par exemple la sélection des dates n'est vraiment pas pratique, pouvoir sélectionner la date de début et de fin sur le même date picker le serait déjà plus par exemple.

5 Annexes

5.1 Journal de travail

Voir annexes (Dans le dossier "Doc") ou [Journal de travail](#)