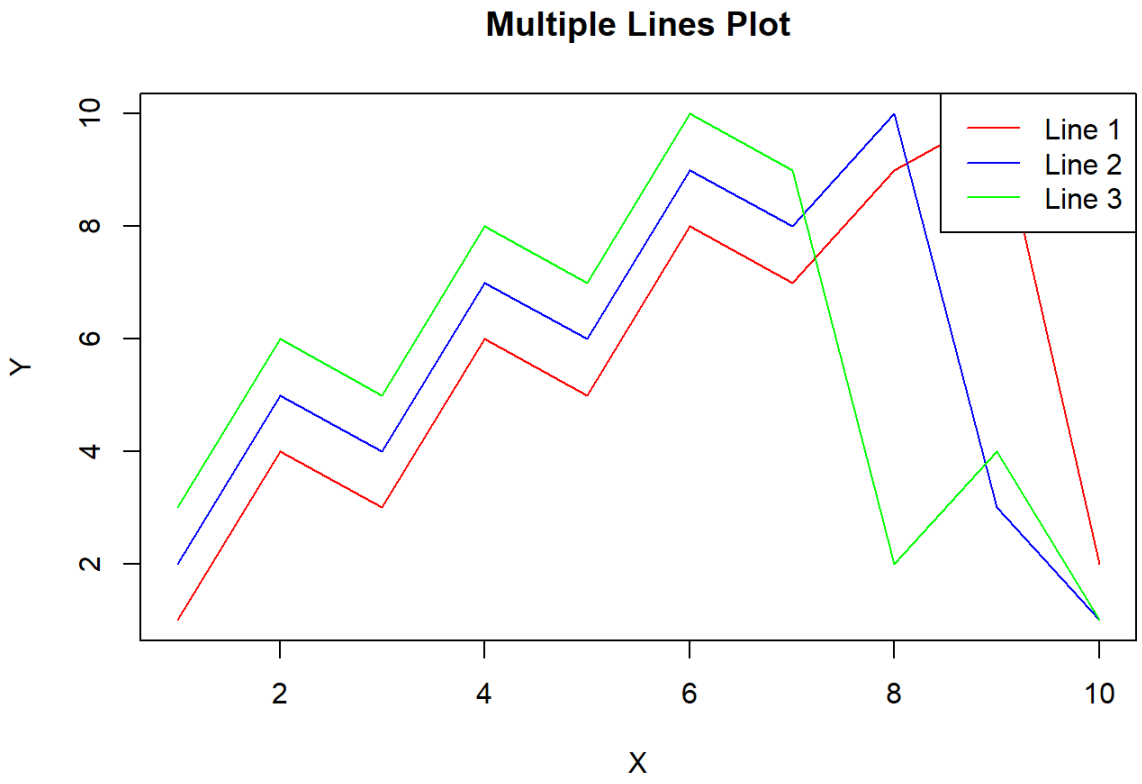
Plot those lines!



plot the line - Recherche Images

Mathieu Bamert – CID 2D
Lausanne - Vennes
32 périodes
Xavier Carrel

Table des matières

1 SPÉCIFICATIONS

1.1 Titre

Plot those lines !

1.2 Description

Mon projet s'appelle Plot Those Lines.

C'est une application écrite en C# avec WindowsForms qui affiche des graphiques de températures à partir de fichiers CSV. Elle permet de comparer les températures entre plusieurs villes de Suisse et de voir comment elles changent chaque jour. L'utilisateur peut importer plusieurs fichiers de données issus de MétéoSuisse, zoomer ou se déplacer dans le graphique, et choisir quelles stations il veut afficher. Le graphique est fait avec la bibliothèque ScottPlot, et j'utilise LINQ pour trier et calculer les valeurs minimales, maximales ou moyennes. Les données et la configuration peuvent être sauvegardées dans un fichier JSON, ce qui permet de rouvrir la session plus tard.

1.3 Domaine d'application

Pour mon projet, j'ai choisi de travailler dans le thème de la météorologie, et plus précisément sur l'étude des températures dans différentes régions de Suisse.

L'objectif de mon logiciel est de comparer les températures moyennes, minimales et maximales enregistrées chaque jour dans plusieurs localités suisses. Grâce à cette comparaison, il sera possible d'identifier les régions où il fait le plus chaud ou le plus froid, ainsi que d'observer les variations de température au fil du temps.

Les données météorologiques utilisées proviendront de MeteoSwiss (MétéoSuisse), la source officielle des mesures météorologiques en Suisse. Ces données permettront d'assurer la fiabilité et la précision des informations affichées dans le logiciel.

Le moyen de récupérer les données sont sur ce site : Donnée météo

1.4 Matériel et logiciels à disposition

- Un PC
- Accès à internet
- Github / Github Project

1.5 Objectifs produit

Le but de mon application est de pouvoir visualiser et de voir l'évolution chaque jour entre la température minimum, maximum, moyenne dans différents lieux en Suisse.

Les fonctionnalités principales sont :

- Importer des données depuis un fichier CSV de MétéoSuisse
- Ajouter autant de donnée qu'on veut sans perdre les anciennes données importer dans l'application
- Afficher des graphique grâce à une librairie qui se nomme ScottPlot

- Zoomer dans le graphique et pouvoir se déplacer à l'aide de la souris
- Affichage de plusieurs endroits en même temps.
- Possibilité de choisir quel graphe on veut afficher en les sélectionnant

1.6 Objectifs pédagogiques

Ce projet m'a aidé à m'améliorer en C# et à mieux comprendre la programmation orientée objet. J'ai aussi appris à manipuler des fichiers CSV et à utiliser LINQ pour filtrer et les données. C'était la première fois que j'utilisais ScottPlot, et j'ai découvert comment relier une interface WindowsForms à une librairie graphique. J'ai aussi appris à organiser mon travail sur GitHub, à planifier les étapes avec les issues et à faire des commits réguliers. Ce projet m'a permis de pratiquer des compétences concrètes : lire des données réelles depuis un fichier CSV et les afficher dans mon graphique.

2 PLANIFICATION INITIALE

Pour ce projet, j'ai utilisé GitHub Project pour organiser mes tâches. J'y ai créé plusieurs issues correspondant aux différentes étapes du développement. Chaque tâche contenait une description, une petite checklist et un lien vers le code ou les tests. Voici les grandes étapes que j'ai suivies : Création du dépôt GitHub et mise en place du projet Visual Studio. Recherche de données météo sur le site MeteoSwiss (CSV). Création des premières maquettes sous WinForms pour imaginer l'interface. Développement de la lecture de fichiers CSV et affichage des données dans le programme. Ajout de ScottPlot pour tracer les graphiques et tester le zoom et le déplacement. Création des filtres temporels et des calculs de min, max et moyenne. Ajout des fonctions mathématiques personnalisées (comme sin(x), x^2). Mise en place de la sauvegarde et du chargement au format JSON. Tests finaux et corrections avant la remise. Le projet a été développé sur plusieurs semaines, à raison de quelques heures par semaine. J'ai essayé de travailler régulièrement pour ne pas avoir trop de choses à faire à la fin. Chaque étape a été testée avant de passer à la suivante pour éviter les erreurs cumulées.

Lien GitHub: https://github.com/MBAMER/Plot_thos_lines_MathieuBamert

3 USER STORIES

Les User Stories m'ont aidé à mieux comprendre les besoins de l'utilisateur et à découper le projet en petites étapes. Elles m'ont permis d'organiser mon travail et de tester chaque fonctionnalité au fur et à mesure.

Affichage de séries temporelles - lien
En tant qu'utilisateur je veux une représentation graphique de plusieurs séries temporelles (time serie) simultanément.

Critères d'acceptance :

- Importer un fichier CSV (Météo suisse) en cliquant sur un bouton "importer des données depuis un fichier CSV".
- Quand j'importe un fichier et que le fichier n'est pas un fichier CSV valide, un message d'erreur s'affiche.
- Quand j'importe un fichier CSV, un message s'affiche et rajoute la liste de température du fichier CSV dans espace exprès pour à gauche

Flexibilité de l'affichage - lien
En tant qu'utilisateur, je veux pouvoir bénéficier d'une grande flexibilité d'affichage afin de pouvoir analyser mes données en détail.

Critères d'acceptance :

- On se déplacer dans mon graphique avec la souris
- On zoome sur mon graphique grâce à la molette
- Quand je clique sur une courbe, la courbe est la seule courbe afficher sur mon graphique

Importer une série de donnée - lien
En tant qu'utilisateur, je veux importer des séries de données de façon permanente. PTL me permet d'importer un ou plusieurs fichier CSV

Critères d'acceptance :

- On importer des données grâce à un fichier CSV
- Les données restent permanentes dans mon application. Quand je quitte mon application et que je la relance mes données restes comme je les ai laissés avant de quitter.

Affichage de plusieurs intervalles de temps pour une même donnée - lien
En tant qu'utilisateur, je veux afficher plusieurs intervalles de temps pour une même donnée. Par exemple, si j'ai une série temporelle pour l'année 2010 et une autre série (fichier, source) pour l'année 2011, je veux pouvoir afficher en une seule ligne les deux années consécutives.

Critères d'acceptance :

- On choisit une période à afficher dans le graphique
- Quand je click sur un bouton précis, mes séries se joigne
- On joint des séries en important un fichier csv

Mode pour afficher des fonctions - lien

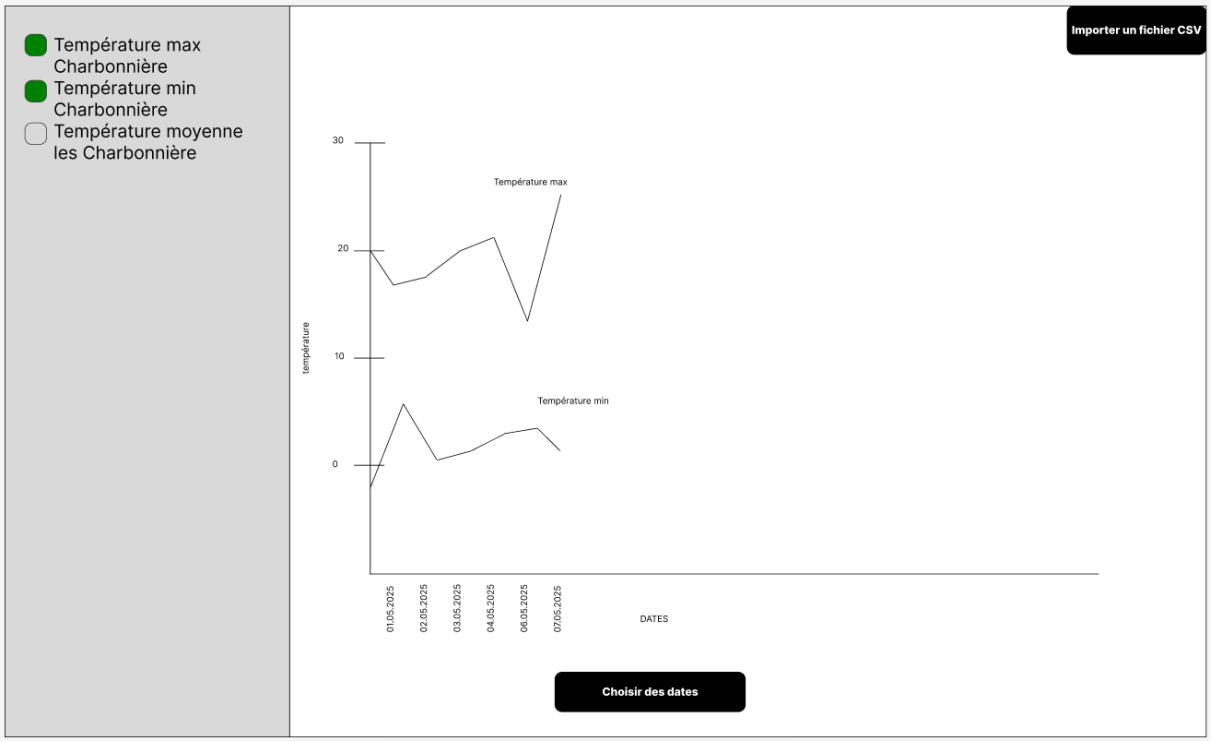
En tant qu'utilisateur, je veux aussi avoir un mode pour afficher des fonctions plutôt que des séries temporelles (onglet, option, ...). Par défaut, je peux voir x2, sinus(x), sin(x) + sin(3x)/3 + sin(5x)/5, x * sin(x). De plus, un champ texte me permet d'écrire une expression personnalisée qui sera exécutée avec Roslyn dynamiquement. Les tokens "sin", "cos","^" sont donc remplacés par leur équivalent C#.

Critères d'acceptance :

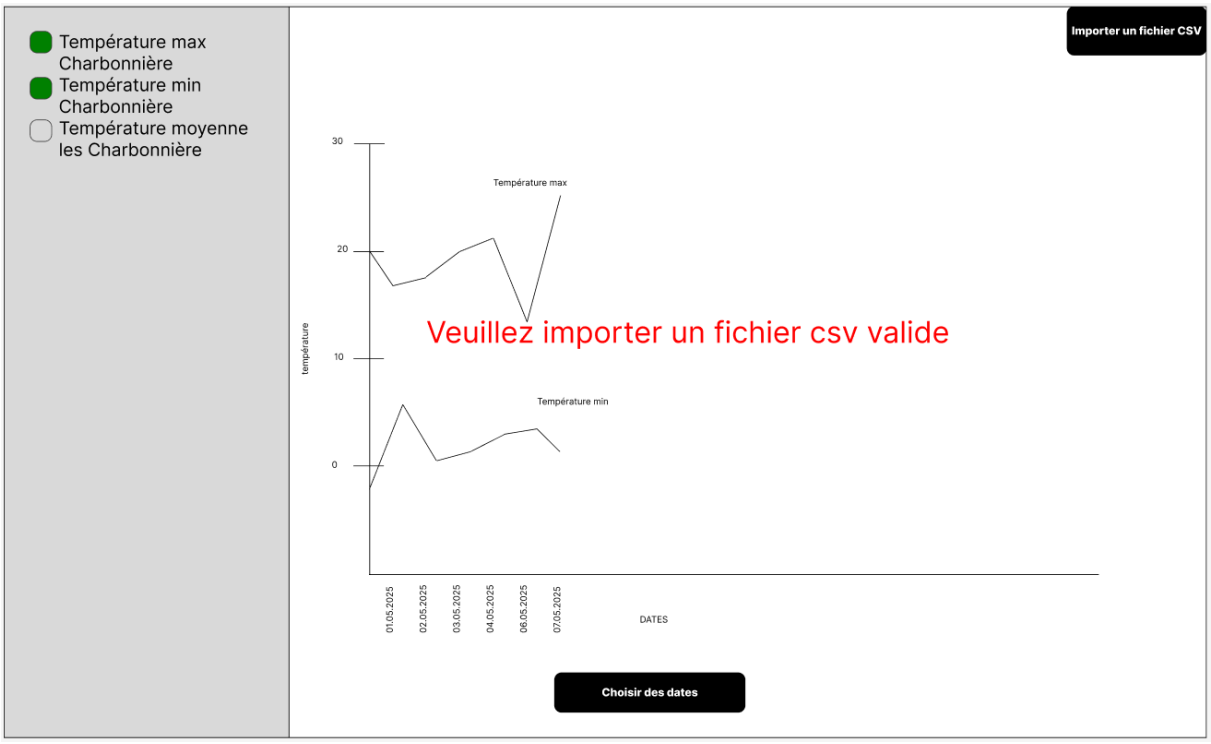
- On a une zone de saisie des expressions personnalisées

4 MAQUETTE

Avant de commencer à coder, j'ai fait plusieurs maquettes pour imaginer à quoi allait ressembler l'application.



Ici on peut voir la page principale, ce que l'utilisateur verra dés la première importation de données. En haut à gauche on peut voir les courbes qu'on veut afficher. Ils sont directement visibles dans le graphique.



Quand on importe un fichier pas valide on peut voir ci-dessus ce qui se passerait. Il y a un message qui s'affiche et les données n'ont pas été exporter

Quand on arrive à importer un fichier CSV il y a un message de validité qui s'affiche.

5 RAPPORT DE TEST

Fonctionnalité Test effectué Résultat attendu Résultat obtenu ✓/✗

Importation CSV Charger un fichier MeteoSwiss Les données s'affichent sur le graphique OK ✓

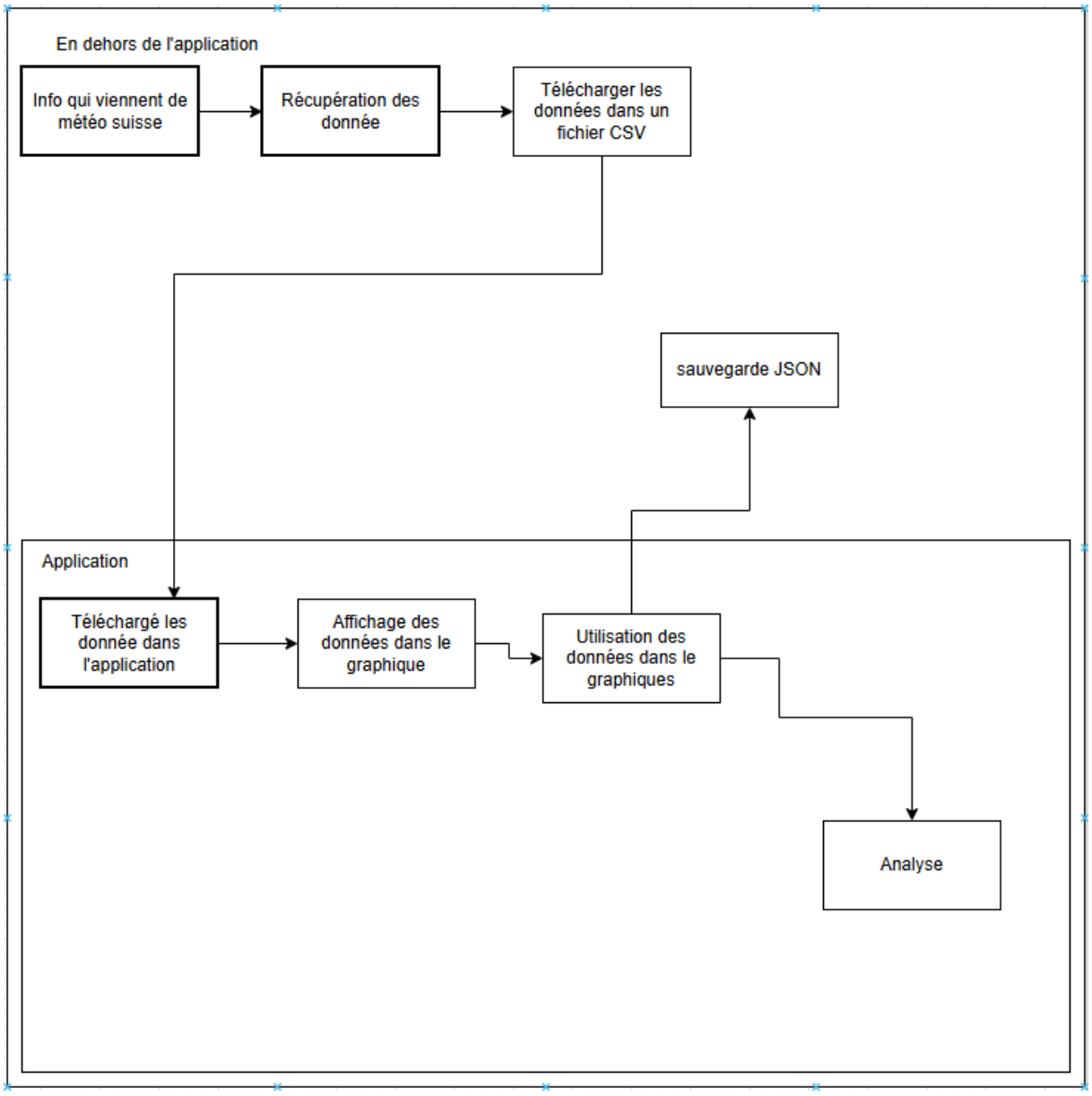
Affichage multiple Ajouter plusieurs stations Chaque station a sa couleur et s'affiche bien OK ✓

Filtres de dates Limiter l'affichage à une période donnée Le graphique s'adapte à la sélection OK ✓

Sauvegarde JSON Sauver la session Fichier JSON créé et lisible OK ✓

Chargement JSON Recharger la session Les mêmes courbes reviennent KO

6 SHEMA



7JOURNAL DE TRAVAIL

Le journal est juste ici :

m-planification-jnlt
rav-MathieuBamert.:

8 USAGE DE L'IA

J'ai utilisé un peu l'intelligence artificielle pendant mon projet, mais surtout pour m'aider à comprendre certains messages d'erreur dans Visual Studio. L'IA m'a aussi servi à reformuler quelques phrases de mon rapport ou à vérifier que mes explications étaient claires.

Finalement, l'IA a été un outil d'aide, mais pas une solution automatique. Tout le code et les idées principales du projet viennent de moi.

9 CONCLUSION

9.1 Conclusion personnelle

Ce projet m'a vraiment permis d'apprendre beaucoup de choses, autant sur la programmation en C# que sur la manière de gérer un vrai projet complet. Au début, je pensais que ce serait assez simple, mais j'ai vite compris qu'il fallait bien structurer le code et tester chaque partie pour éviter les erreurs.

Voir le graphique se tracer avec les températures de plusieurs villes, c'était motivant. J'ai aussi mieux compris comment manipuler des fichiers CSV, comment utiliser LINQ pour filtrer et comment sauvegarder des données avec JSON.

J'ai appris à travailler plus proprement, à corriger mes erreurs sans paniquer, et à écrire du code plus clair. Si je devais améliorer le programme, j'aimerais ajouter d'autres types de données (comme les précipitations) ou moderniser un peu l'interface.

Dans l'ensemble, je suis content du résultat final.

9.2 Conclusion Technique

Le développement de l'application Plot The Line a permis de concevoir un outil complet d'analyse et de visualisation de données météorologiques à partir de fichiers CSV. L'application, réalisée en C# sous WinForms, s'appuie sur la bibliothèque ScottPlot pour la représentation graphique interactive des températures (moyenne, maximum, minimum) en fonction du temps.

Sur le plan technique, plusieurs points essentiels ont été mis en œuvre :

Lecture et traitement des fichiers CSV : les données sont importées dynamiquement, validées et converties en listes typées (DateTime, double), avec gestion des séparateurs et des valeurs manquantes.

Utilisation de LINQ : la majorité des traitements (sélection, filtrage, génération d'items d'interface) repose sur des expressions LINQ, garantissant un code concis, lisible et maintenable.

Affichage graphique dynamique : chaque série de données est affichée via un FormsPlot, avec un rafraîchissement automatique selon les cases cochées. L'interface permet à l'utilisateur de visualiser plusieurs fichiers simultanément grâce à un système de couleurs cycliques.

Fiabilité et robustesse : les exceptions sont gérées proprement et les opérations de désérialisation ou d'importation sont protégées afin d'éviter tout plantage en cas de données invalides.

10 PROBLÈME TECHNIOQUE RENCONTRÉ

Mon principal problème a été la persistance des données. J'arrivais à afficher le nom de chaque graphe et mettre leur label dans le graphique. Mais l'application n'affichait pas les courbes. Le problème est sûrement dans les données. J'aurais beau essayé de demander à un camarade de classe ou à l'IA personne a réussi à résoudre ce problème. Je n'ai pas trouvé de réponse à ce problème.

10 SOURCES

Voici les différentes sources que j'ai utilisées pendant le projet : MeteoSwiss Open Data – https://www.meteosuisse.admin.ch/services-et-publications/applications/ext/telecharger-des-donnees-sans-savoir-coder.html#lang=fr&mdt=normal&pgid=&sid=&col=&di=&tr=&hdr=