Rapport du projet – Plot Those Lines! (Crypto Edition)

1. Introduction

1.1 Objectifs pédagogiques

Ce projet est conçu pour me permettre de mettre en pratique les compétences acquises dans les modules de programmation. L'accent est mis sur :

- la programmation orientée objet avec C# ;
- l'utilisation de LINQ comme approche moderne de traitement des données ;
- la mise en œuvre de tests unitaires afin d'assurer la qualité du code ;
- la gestion de projet avec GitHub (projets, commits, versioning) ;
- la production d'une documentation professionnelle (Rapport, JDT, README).

L'objectif pédagogique est donc double : d'un côté apprendre à développer une application concrète avec un client, et de l'autre acquérir des bonnes pratiques professionnelles en termes de planification, organisation et communication technique.

1.2 Objectifs produit

Le produit attendu est une application Windows Forms qui permet de visualiser des séries temporelles sous forme de graphiques.

- L'utilisateur pourra importer et afficher les données issues d'une API externe (CoinGecko).
- L'application doit être capable d'afficher plusieurs séries sur un même graphique (par ex. Bitcoin + Ethereum).
- L'utilisateur doit pouvoir choisir la période d'analyse (7, 30, 90 jours).
- Des fonctions supplémentaires permettront d'analyser les données de manière flexible (zoom, légendes, choix de couleurs).

Le produit n'est pas seulement une démonstration technique : il doit être utilisable et compréhensible par un utilisateur non technique qui souhaite visualiser facilement l'évolution des cryptomonnaies.

2. Domaine et sources de données

J'ai choisi le domaine des cryptomonnaies pour plusieurs raisons :

- C'est un secteur en forte croissance et très médiatisé.
- Les données financières et temporelles y sont particulièrement adaptées à la représentation graphique.
- Il existe plusieurs sources de données fiables et gratuites.

2.1 Sources de données

- CoinGecko API (https://www.coingecko.com/en/api)
  CoinGecko est une plateforme reconnue dans le domaine des cryptos. Son API gratuite fournit des données historiques et en temps réel au format JSON. C'est cette source que j'utiliserai pour alimenter mon application.

- Investing.com – Live Charts (https://www.investing.com/charts/live-charts)
  En analysant cette plateforme, j'ai pu m'inspirer des fonctionnalités de visualisation qui pourraient être intéressantes à intégrer (choix d'un actif, intervalle de temps, graphiques multi-séries).

2.2 Justification du choix

J'ai préféré l'utilisation d'une API JSON plutôt que de fichiers CSV pour plusieurs raisons :

1. Les données sont toujours à jour.
2. L'API permet une intégration dynamique sans téléchargement manuel.
3. Le format JSON est mieux adapté à la sérialisation en objets C#.

3. Analyse et planification

3.1 User Stories

Voir README.md
(section User Stories) et le GitHub Project associé.

Exemples:

- Importer des données JSON depuis l'API CoinGecko.
- Afficher un graphique temporel avec ScottPlot.
- Comparer plusieurs cryptomonnaies.
- Documentation (Rapport, JDT, README, Planification).

3.2 Planification

J'ai planifié le projet sur **8 semaines** pour un total de 24 à 32 périodes, avec une marge de sécurité.

Consulter le fichier séparé Planification.md

4. Réalisation

4.1 Choix techniques

- **Framework** : C# Windows Forms (facile à mettre en œuvre et compatible avec ScottPlot).
- **API externe** : CoinGecko (format JSON).
- **Parsing JSON** : System.Text.Json.
- **Manipulation de données** : LINQ (remplace les boucles for).
- **Graphiques** : ScottPlot (librairie simple et efficace).

4.2 Organisation du code

- Namespace principal : PTL_Crypto.
- Classes :
  - CryptoPrice (modèle de données).
  - ApiClient (récupération JSON).
  - PlotManager (gestion graphique).
- Méthodes d'extension :
  - Conversion timestamp → DateTime.
  - Conversion liste JSON → séries exploitables par ScottPlot.

5. Tests

- **Tests unitaires**
- Vérification du parsing correct du JSON depuis CoinGecko API.
- Vérification de la conversion timestamp → DateTime via les méthodes d'extension.
- Vérification de la sélection et du filtrage des périodes (7, 30, 90 jours) avec LINQ.
- Vérification de la transformation des données JSON en séries exploitables par ScottPlot.
- **Tests d'acceptation**
- Lancer l'application, saisir "bitcoin" et vérifier l'affichage du graphique pour 30 jours.

- Comparer Bitcoin + Ethereum et s'assurer que 2 courbes distinctes apparaissent correctement.
- Saisir un identifiant invalide et vérifier la gestion de l'erreur (MessageBox ou Label).

6. Journal de travail (JDT)

Le Journal de travail est disponible dans JDT.xlsx

Il contient les tâches réalisées à chaque séance, la durée et un commentaire (progrès, difficultés, corrections).

7. Utilisation de l'IA

- Recherche d'API adaptées (CoinGecko, alternatives).
- Reformulation des User Stories.
- Assistance technique pour LINQ et Windows Forms(un des source d'information théorique).

Toutes les suggestions ont été vérifiées, adaptées et comprises avant intégration.

8. Conclusion

J'ai appris à :

- Travailler avec une API externe (CoinGecko).
- Manipuler des données JSON avec LINQ.
- Générer et afficher des séries temporelles avec ScottPlot.

Ce projet m'a permis de lier théorie et pratique, en réalisant un outil concret d'analyse visuelle des cryptomonnaies.

9. Références

- CoinGecko API: https://www.coingecko.com/en/api
- Exemple d'interface: https://www.investing.com/charts/live-charts
- ScottPlot: https://scottplot.net
- Documentation Microsoft LINQ: https://learn.microsoft.com/dotnet/csharp/linq
- CoinMarketCap: https://coinmarketcap.com