Rapport de Projet

De Pina Correia Ryan – CID3A

Lausanne, Vennes – 32p

Table des matières

1. Spécification

1.1 Introduction

1.1.1 Description du domaine

Ce projet s'inscrit dans le **domaine de la visualisation de données financières et temporelles**. Il vise à fournir un outil d'analyse graphique pour suivre l'évolution des valeurs boursières de différentes entreprises.

Contexte métier : - **Analyse financière** : Suivi des cours d'actions en temps réel via l'API Twelve Data - **Comparaison de performances** : Visualisation simultanée de plusieurs entreprises (Tesla, Apple, Microsoft, etc.) - **Analyse temporelle** : Étude de l'évolution des prix sur différentes périodes (2020-2025) - **Données boursières** : Prix d'ouverture, fermeture, plus haut, plus bas, volume d'échanges

Applications concrètes : - Investisseurs souhaitant comparer les performances de différentes actions - Analystes financiers étudiant les tendances du marché - Étudiants en finance découvrant les marchés boursiers - Traders recherchant des opportunités d'investissement

1.1.2 Description du projet

Concevoir un logiciel permettant d'afficher et d'analyser des séries temporelles sous forme graphique.
L'utilisateur pourra importer des données externes (CSV, JSON, API) et comparer plusieurs jeux de données simultanément.

1.1.3 Objectifs pédagogiques

Ce projet a pour but de développer les compétences suivantes :

1. Programmation fonctionnelle (ICT-323) - Utilisation exclusive de **LINQ** (pas de boucles `for`) - Implémentation d'**extensions C#** personnalisées (`Filter`, `ForEachDo`) - Application des concepts de programmation fonctionnelle (immutabilité, fonctions pures) - Manipulation de collections avec les méthodes LINQ (`Select`, `Where`, `OrderBy`, etc.)

2. Consommation d'API REST - Intégration de l'API **Twelve Data** pour récupérer des données boursières - Utilisation de **HttpClient** pour effectuer des requêtes HTTP - Parsing de réponses **JSON** avec `Newtonsoft.Json` - Gestion des erreurs et des limites de l'API (quotas de requêtes)

3. Visualisation de données - Utilisation de la librairie **Chart** de Windows Forms - Affichage de graphiques linéaires avec plusieurs séries - Gestion de l'axe temporel (dates) et de l'axe des valeurs (prix) - Implémentation de fonctionnalités interactives (zoom, filtres, statistiques)

4. Gestion de projet - Utilisation de **GitHub** pour le versioning et la gestion de projet - Rédaction d'un **journal de travail** régulier - Création de **User Stories** avec tests d'acceptance - Planification et suivi de l'avancement du projet

5. Architecture logicielle - Séparation des responsabilités (Service API, Interface utilisateur) - Gestion de l'état de l'application (marques, données, zoom) - Optimisation des performances (cache, chargement asynchrone) - Tests unitaires pour valider les fonctionnalités critiques

1.3 Matériel et logiciels à disposition

- **Visual Studio 2022**
- **GitHub**
  - Project
  - Repository (repo)
  - Roadmap
  - User stories
- **Python**
- **API : Twelve Data**
- **Figma**

1.4 Prérequis

- **ICT-323 Programmation fonctionnelle**
- **C#**
- **API REST (HttpClient)**
- **Gestion de projet**
- **Passion Lecture (mobile)**
- **C335 (flashcards)**

1.5 Cahier des charges

Gestion de projet

Les directives spécifiques à la gestion de projet vous seront fournies séparément par votre chef de projet.
Dans tous les cas, les points suivants doivent être respectés :

1. Vous êtes inscrit au projet **P_FUN** sur Marketplace :
   Lien vers Marketplace
2. Vous devez effectuer une **analyse sous forme de User Stories (US)**
   - Inclure les tests d'acceptance
   - Inclure des maquettes
   - Cette étape doit être réalisée avant de commencer à coder
3. Vous devez fournir une **planification simple** de la réalisation de ces US avant de commencer à coder
4. Vous devez tenir un **Journal de travail Jdt.md**

1.5.1 Objectifs et portée (SMART)

Objectif 1 : Réaliser un programme informatique de qualité

- Organisé (namespace, classes, commit log, …)
- Compacté (pas de copié/collé)
- Optimisé (structures adaptées)
- Testé (tests unitaires)
- Commenté
- Complet (code, modèle de données, maquettes PDF, exécutable, …)

Objectif 2 : Prouver que vous êtes digne de confiance dans la gestion d'un projet

- Journal de travail à jour
- Pro-activité (poser des questions au client, faire des démonstrations, utiliser Git pour versioning)

1.5.2 Caractéristiques des utilisateurs et impacts

- Public cible : utilisateurs ayant besoin d'analyser des données temporelles (finance, météo, santé, sport, etc.).
- Impacts attendus :
  - Gain de temps pour la comparaison de données.
  - Accessibilité via un outil ergonomique.
  - Flexibilité d'affichage des graphiques pour répondre à des besoins variés.

1.5.3 Fonctionnalités requises

- Afficher plusieurs séries temporelles simultanément.
- Offrir une flexibilité d'affichage pour analyser les données en détail.
- Importer des séries de données de manière permanente (CSV, JSON, API).
- Comparer plusieurs intervalles de temps pour une même donnée.
- Mode fonctions permettant d'afficher des expressions mathématiques (préconfigurées et personnalisées via Roslyn).

1.5.4 Contraintes

- Utiliser **LINQ** (pas de boucle `for`).
- Implémenter **au moins 2 extensions C#**.
- Utiliser une **librairie graphique** (Forms, MAUI, Uno, WPF, …).
- Utiliser une **librairie de visualisation de données** (ex. ScottPlot).

1.5.5 Travail à réaliser par l'apprenti

- Réalisation du code source et des fonctionnalités principales.
- Mise en place des tests unitaires et de validation.
- Rédaction du rapport PDF (introduction, planification, tests, journal de travail, usage de l'IA, conclusion).
- Publication sur GitHub (release incluant code + rapport).

1.5.6 Si le temps le permet

- Ajouter des options avancées d'affichage (filtres, zoom interactif, mise en forme personnalisée).
- Développer des connecteurs supplémentaires vers d'autres APIs publiques.

- Optimiser les performances pour de très grands jeux de données.

1.5.7 Méthodes de validation

- Vérification par tests unitaires et manuels.
- Démonstrations intermédiaires au client/chef de projet.
- Validation des User Stories (tests d'acceptance).
- Comparaison avec les spécifications initiales.

1.6 Points évalués

- Respect des **objectifs SMART**.
- Qualité et organisation du **code source**.
- Capacité à utiliser **GitHub** pour la gestion de projet.
- Exhaustivité et régularité du **journal de travail**.
- Pertinence de la **planification** et respect des délais.
- Clarté et qualité du **rapport PDF**.
- Autonomie, pro-activité et communication avec le client.

1.7 Validation et conditions de réussite

- Le projet est considéré comme réussi si :
  - Une **release GitHub** est disponible avec le code complet et fonctionnel.
  - Le **rapport PDF** contient toutes les sections requises (introduction, planification, tests, journal, IA, conclusion).
  - Les fonctionnalités principales (affichage de séries temporelles, importation, flexibilité d'affichage, mode fonctions) sont opérationnelles.
  - Les **tests unitaires** confirment le bon fonctionnement des composants critiques.
  - Le **journal de travail** est tenu régulièrement et reflète l'évolution du projet.
  - Le projet respecte les contraintes techniques et la planification validée avec le client.

---

2. Planification Initiale

2.1 Méthodologie de projet

- **Approche :** Développement par itérations avec User Stories
- **Outils de gestion :**
  - **GitHub Project :** Project #4
  - **Repository :** Plot-those-line
  - **Issues & Roadmap :** Suivi des tâches et bugs
- **Durée totale :** 32 périodes
- **Suivi :** Journal de travail détaillé → Consulter le JDT

2.4 Suivi et contrôle

- **GitHub Project Board :** Tableau de bord Project #4
  - **Backlog** : User Stories en attente
  - **In Progress** : Tâches en cours de développement
  - **Done** : Fonctionnalités terminées et testées
- **Journal de travail quotidien :** JDT détaillé
- **Issues GitHub :** Suivi des bugs et améliorations
- **Commits réguliers :** Messages explicites et atomiques
- **Points réguliers** avec le chef de projet

---

3. Analyse

3.1 Explication API

Twelve Data

Cette API n'était pas la première api que j'ai utilisé, au début du projet j'avais utilisé l'API Alpha Vantage mais le soucis, c'est que pour une Api gratuite elle me fournissait que 25 requête par jour. Le soucis c'est que dès que je testais 3 - 4 fois mon graphique por savoir si tout s'affichait je ne pouvais plus rien utiliser. Après un moment de recherche j'ai trouvé cette api qui me fournit les données voulues et parfaitement bien. Elle me laisse + de liberté et me laisse plus de requête et j'ai une meilleur vision de ce que je peux utiliser par jour via leur interface web + complète.

3.2 Gestion de l'API

Structure de réponse Twelve Data

L'API Twelve Data retourne les données au format JSON avec la structure suivante :

```
{
  "meta": {
    "symbol": "TSLA",              // Nom de la marque (abréviation)
    "interval": "1day",            // Données par jour
    "currency": "USD",             // Monnaie utilisée
    "exchange_timezone": "America/New_York",
    "exchange": "NASDAQ",
    "mic_code": "XNGS",
    "type": "Common Stock"
  },
```

```
  "values": [
    {
      "datetime": "2025-09-23",     // Date de l'ouverture
      "open": "439.88000",          // Prix à l'ouverture des actions
      "high": "440.97000",          // Le plus haut dans la journée
      "low": "423.72000",           // Le plus bas dans la journée
      "close": "425.85001",         // Prix à la fermeture des actions
      "volume": "83211500"          // Nombre d'actions échangées
    },
    {
      "datetime": "2025-09-22",
      "open": "431.10999",
      "high": "444.98001",
      "low": "429.13000",
      "close": "434.20999",
      "volume": "97108800"
    }
  ],
  "status": "ok"                    // Statut de la requête
}
```

Utilisation dans Plot-Those-Line

Ces données Twelve Data sont ensuite traitées pour générer les graphiques :

- **Close (Prix de clôture)** : Valeur principale affichée sur le graphique (axe Y)
- **High/Low** : Pour visualiser la volatilité (Average = Moyenne)
- **DateTime** : Axe temporel (X) du graphique

Clarification des valeurs affichées :

L'axe Y du graphique représente le **cours de l'action en USD (dollars américains)**. Il s'agit du prix de clôture quotidien de l'action, c'est-à-dire le dernier prix auquel l'action a été échangée lors de la fermeture du marché boursier.

Exemple de lecture : - Si le graphique affiche "425.85 USD" pour Tesla le 23/09/2025, cela signifie qu'une action Tesla coûtait 425,85 dollars américains à la clôture du marché ce jour-là.

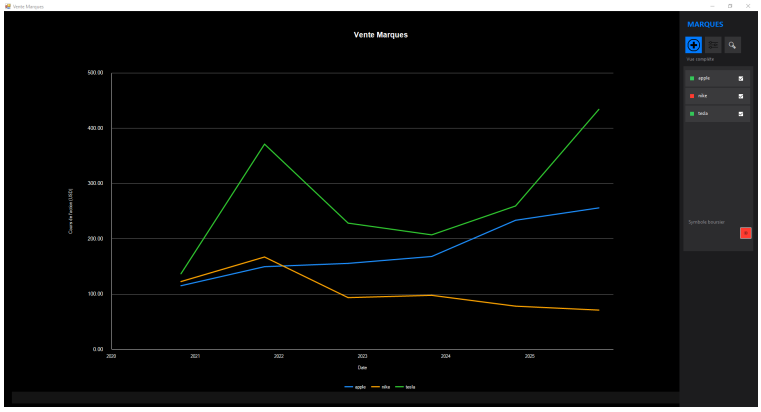
Statistiques affichées : - **Min** : Prix de clôture le plus bas sur la période affichée - **Max** : Prix de clôture le plus haut sur la période affichée
- **Moy (Moyenne)** : Prix de clôture moyen sur la période affichée

---

3.3 Conception des tests

j'ai fait un projet annexe qui contient MTest, et qui est en réference avec mon projet principal. Il contient des tests d'acceptence. - 5.1 Dossier des tests

3.4 Affichage graphique (Code)




```
private void SetupChart(Panel parent)
{
    chart = new Chart { Dock = DockStyle.Fill, BackColor = Color.Black };
    var chartArea = new ChartArea("main")
    {
        BackColor = Color.Black,
        BorderColor = Color.White,
        BorderWidth = 1
    };

    // Ajuster la position et la taille du graphique pour éviter le débordement
    chartArea.Position.X = 5; // Marge gauche
    chartArea.Position.Y = 5; // Marge haute
    chartArea.Position.Width = 85; // 85% de la largeur disponible (réduire pour éviter le débordement)
    chartArea.Position.Height = 90; // 90% de la hauteur disponible

    // Configuration de l'axe X avec affichage des mois et années
    chartArea.AxisX.MajorGrid.Enabled = false;
    chartArea.AxisX.LabelStyle.ForeColor = Color.White;
    chartArea.AxisX.LineColor = Color.White;
    chartArea.AxisX.Title = "Date";
    chartArea.AxisX.TitleForeColor = Color.White;
    chartArea.AxisX.IntervalType = DateTimeIntervalType.Months;
    chartArea.AxisX.Interval = 1; // Afficher chaque mois
    chartArea.AxisX.LabelStyle.Format = "MMM"; // Format court des mois (Jan, Fév, Mar, etc.)
    chartArea.AxisX.LabelStyle.Angle = -45; // Incliner les labels pour éviter le chevauchement

    chartArea.AxisY.MajorGrid.LineColor = Color.Gray;
    chartArea.AxisY.LabelStyle.ForeColor = Color.White;
    chartArea.AxisY.LineColor = Color.White;
    chartArea.AxisY.Title = "Cours de l'action (USD)";
    chartArea.AxisY.TitleForeColor = Color.White;
    chartArea.AxisY.LabelStyle.Format = "#,0.00";

    // Définir les limites de l'axe X (copiée du test qui fonctionne)
    var minDate = new DateTime(2020, 9, 24);
    var maxDate = new DateTime(2025, 9, 23);
    chartArea.AxisX.Minimum = minDate.ToOADate();
    chartArea.AxisX.Maximum = maxDate.ToOADate();

    // Forcer l'affichage de toute la plage
    chartArea.AxisX.ScaleView.Zoomable = false;
    chartArea.AxisY.ScaleView.Zoomable = false;

    // Ajouter des marges pour éviter le débordement
    chartArea.InnerPlotPosition.X = 10; // Marge gauche
    chartArea.InnerPlotPosition.Y = 10; // Marge haute
    chartArea.InnerPlotPosition.Width = 80; // Largeur réduite
    chartArea.InnerPlotPosition.Height = 80; // Hauteur réduite
```

4.1 Dossier de réalisation

4.1.1 Architecture du projet

Repository GitHub : RyanDPC/Plot-those-line

Structure des fichiers : Plot-those-line/ ├── ActionMarque/ # Application principale C# | ├── TwelveDataService.cs # Service API Twelve Data | ├── Form1.cs # Interface utilisateur | ├── AlphaVantageService.cs # Service API alternatif | └── ActionMarque.csproj # Configuration projet ├── docs/ # Documentation | ├── jdt.md # Journal de travail | └── Livrables/rapport.md # Rapport de projet └── script/ # Scripts d'automatisation ├── generate-jdt.py # Génération JDT automatique └── export-rapport.bat # Export PDF du rapport

4.1.2 Fonctionnalités implémentées

Suivi détaillé : GitHub Project Board

Fonctionnalité	Statut	Commit	Issue
Service Twelve Data API	Terminé	`TwelveDataService.cs`	-
Interface graphique	Terminé	`Form1.cs`	-
Gestion multi-granularité	Terminé	Enum `DataGranularity`	-
Filtrage par dates	Terminé	Période 2020-2025	-
Parsing JSON robuste	Terminé	Gestion erreurs	-
Documentation automatisée	Terminé	Scripts Python/Batch	-

---

5. Tests

5.1 Dossier des tests

Version de l'application testée : 1.0.0

Date du test : 8 Octobre 2025

Nom du testeur : Ryan De Pina

Scénario 1

 Description

En tant que utilisateur voulant comprendre les relations entre données

Je souhaite comparer des séries différentes comme le prix et le temps

Pour voir si les variations de prix sont liées au temps mis dans le de marché

 Tests d'acceptance

Test validé

Contexte : L'utilisateur charge les données de prix et temps.

Action : Il les affiche sur un même graphique avec deux axes.

Résultat attendu : Les deux séries sont visibles avec un axe pour chaque.

Test validé

Contexte : L'utilisateur filtre une période.

Action : Il vérifie la cohérence des données affichées.

Résultat attendu : Les données restent synchronisées et claires.

Scénario 2

 Description

En tant que Un analyste de données dans une grande entreprise

Je souhaite Afficher plusieurs séries temporelles simultanément (par exemple, l'évolution des actions de plusieurs entreprises sur une période donnée)

Pour Comparer l'évolution de différentes actions ou données financières sur la même période, facilitant ainsi les décisions stratégiques basées sur les tendances du marché.

 Tests d'acceptance

Test validé

Contexte : L'analyste a téléchargé des données boursières pour différentes entreprises (par exemple, Tesla, Apple, Microsoft).

Action : L'analyste sélectionne les séries temporelles correspondant aux actions de ces entreprises sur les 5 dernières années.

Résultat attendu : Une représentation graphique s'affiche avec toutes les séries temporelles superposées, permettant une comparaison facile entre les entreprises.

Scénario 3

 Description

En tant que Un analyste de données dans une entreprise financière

Je souhaite Bénéficier d'une flexibilité d'affichage, notamment pouvoir zoomer ou filtrer mes séries temporelles en fonction de la granularité souhaitée (ex: vue annuelle, mensuelle, hebdomadaire)

Pour Analyser mes données en détail, détecter des anomalies ou des tendances cachées, et avoir une meilleure compréhension du comportement du marché.

 Tests d'acceptance

Test validé

Contexte : L'analyste a sélectionné une série temporelle représentant l'évolution du prix de l'action Apple sur les 10 dernières années.

Action : L'analyste applique un zoom sur les 2 dernières années et filtre les données pour afficher une vue mensuelle.

Résultat attendu : L'affichage se met à jour pour montrer uniquement les données mensuelles sur les 2 dernières années.

Scénario 4

 Description

En tant que Un data engineer dans une grande entreprise technologique

Je souhaite Importer de manière permanente des séries de données provenant de diverses sources (CSV, JSON, API) dans l'application

Pour Assurer une mise à jour continue des données boursières de plusieurs entreprises et les rendre disponibles pour les analyses en temps réel.

 Tests d'acceptance

Test validé

Contexte : Le data engineer souhaite importer les données boursières des actions de Google depuis une API prise avec Twelve Data.

Action : Ecrire l'entreprise voulue sur la zone de texte.

Résultat attendu : Si l'entreprise est sur l'API, les séries temporelles sont visibles dans l'application avec les bonnes valeurs.

---

6. Usage de l'IA

Dans ce projet, j'ai utilisé l'intelligence artificielle de manière ciblée pour m'assister dans différents aspects du développement. Voici un récapitulatif détaillé de mon utilisation :

6.1 Rapport et documentation

J'ai utilisé l'IA uniquement pour le visuel et pour m'aider à implémenter le style de l'ETML pour l'en-tête, car je voulais créer un template markdown pour de futurs projets. L'IA m'a permis de mettre en place un en-tête professionnel avec le style ETML, incluant le logo et la mise en forme appropriée.

6.2 Développement du code

Pour les extensions au niveau syntaxe des extensions, j'ai fait appel à l'IA car je ne savais pas comment les écrire proprement. L'IA m'a aidé à identifier quelles fonctionnalités d'extensions seraient utiles pour l'application, ce qui m'a permis d'implémenter des extensions C# fonctionnelles et bien structurées.

6.3 Interface utilisateur (CSS)

J'ai également utilisé l'IA pour le CSS de l'application afin qu'elle soit assez belle à utiliser et à regarder. L'IA m'a assisté dans la conception d'un design moderne et professionnel pour l'interface utilisateur. Cela incluait la sélection de couleurs harmonieuses, l'optimisation de la disposition des éléments, l'amélioration de l'expérience utilisateur globale et l'adaptation responsive des composants.

7. Conclusion

7.1 Bilan des fonctionnalités demandées

Fonctionnalités implémentées :

- Affichage de séries temporelles simultanées
- Importation de données via API Twelve Data
- Interface graphique avec gestion des marques (ajout/suppression)
- Calcul et affichage des statistiques (Min, Max, Average)
- Filtrage par dates (période 2020-2025)
- Gestion multi-granularité des données
- Parsing JSON robuste avec gestion d'erreurs

7.2 Bilan de la planification

J'ai essayé de suivre ma planification et de suivre ma vitesse de travail. Finalement, avec du travail supplémentaire à la maison, j'ai pu finir plus tôt le 1 octobre 2025. Les prochains moments de cours seront surtout pour peaufiner et corriger les erreurs émises par l'enseignant.

7.3 Bilan personnel

J'ai particulièrement apprécié la réalisation de ce rapport car cela m'a permis de créer un template réutilisable pour mes futurs projets GitHub. Cette approche de documentation structurée me sera très utile pour organiser mes travaux. Concernant la partie développement, j'ai constaté que l'utilisation d'une API avec C# présente certaines difficultés pour moi. Je trouve que la visualisation et la compréhension du code sont plus complexes dans ce langage comparé à d'autres frameworks que j'ai pu utiliser. L'utilisation de Visual Studio 2022 a également représenté un défi supplémentaire. N'étant pas familier avec cet environnement de développement, je l'ai trouvé moins intuitif que d'autres IDE que j'utilise habituellement. Néanmoins, cette expérience m'a permis de découvrir un nouvel outil professionnel et de sortir de ma zone de confort technologique, ce qui enrichit mon profil de développeur.

8. Divers

8.1 Journal de travail

- **Suivi :** Journal de travail détaillé → Consulter le JDT