|  |
| --- |
| Plot that lines |

Table des matières

[1 Analyse préliminaire 3](#_Toc180662353)

[1.1 Introduction 3](#_Toc180662354)

[1.2 Objectifs 3](#_Toc180662355)

[1.3 Gestion de projet 4](#_Toc180662356)

[2 Analyse / Conception 4](#_Toc180662357)

[2.1 Domaine 4](#_Toc180662358)

[2.2 Concept 4](#_Toc180662359)

[2.3 Analyse fonctionnelle 4](#_Toc180662360)

[2.4 Stratégie de test 4](#_Toc180662361)

[3 Réalisation 5](#_Toc180662362)

[3.1 Points de design spécifiques 5](#_Toc180662363)

[*3.1.1* *…* 5](#_Toc180662364)

[*3.1.2* *…* 5](#_Toc180662365)

[*3.1.3* *…* 5](#_Toc180662366)

[3.2 Déroulement 5](#_Toc180662367)

[3.3 Mise en place de l’environnement de travail 6](#_Toc180662368)

[3.4 Description des tests effectués 6](#_Toc180662369)

[3.5 Erreurs restantes 6](#_Toc180662370)

[4 Conclusions 6](#_Toc180662371)

[5 Annexes 7](#_Toc180662372)

[5.1 Journal de travail 7](#_Toc180662373)

# Analyse préliminaire

## Introduction

Ce projet est réalisé dans le cadre du module I323 afin d’apprendre la programmation fonctionnelle en créant une application console affichant des graphiques.

Dans ce projet l’application console permet d’afficher des graphiques concernant les dépenses militaires de chaque pays dans la devise décidé par l’utilisateur afin de comparer les dépenses effectuées par les différents pays sélectionné.

## Objectifs

* Création d’une interface graphique permettant de filtrer l’affichage des données.
  + Filtre sur les dates
  + Au moins 3 courbes
  + L’axe X correspond à la temporalité

## Gestion de projet

La méthode de projet utilisé est la méthode agile. Le chef de projet est Xavier Carrel.

La méthode Agile est une méthode de gestion de projet centré sur la flexibilité et la collaboration avec les membres de l’équipe et le client.

Elle fonctionne avec des cycles de développement court appelés « sprints ».

Dans le cadre de ce projet un seul sprint est effectué.

# Analyse / Conception

## Domaine

Les données récoltées sont les dépenses militaires de plusieurs pays selon leurs devises locales.

Le domaine des données est militaire et économique.

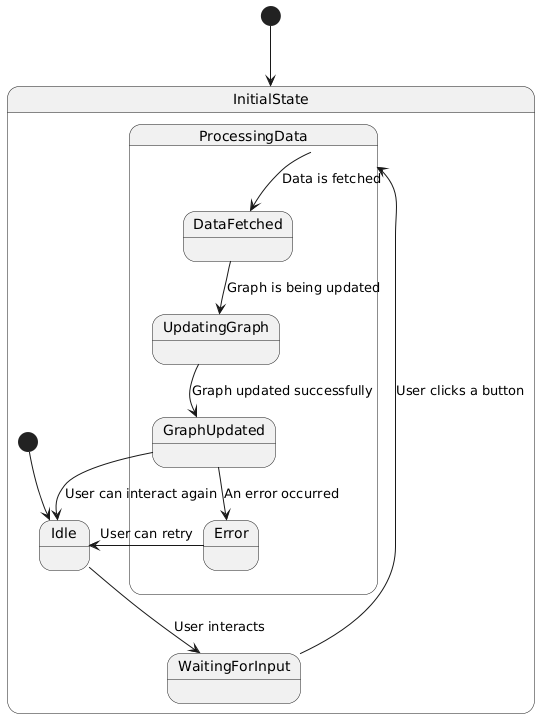
L’échelle de temps utilisé est d’une année en une année.

Le public destiné contient des économistes mais aussi n’importe quelle personne curieuse de connaître les dépenses militaires d’un pays en particulier et de les comparer à d’autres pays.

## Concept



1 - Diagramme de classe



2 - Diagramme d'état

## Analyse fonctionnelle

***Reprendre le contenu des User Stories d’IceScrum : Story + tests d’acceptance (avec IceTools) + maquettes***

## Stratégie de test

Des tests unitaires sont effectués avec MSTest, ils vérifient que les méthodes utilisées renvoient des valeurs correctes. Les méthodes faisant appel à une API ne sont pas testées ainsi que celles ne retournant aucune valeur.

**Ordre des tests :**

1. Tests unitaires :

Les tests unitaires vérifieront que les méthodes fonctionnent indépendamment du reste.

1. Tests d’intégration :

Les tests d’intégration sont effectués pour vérifier que l’appel à l’API fonctionne correctement.

1. Tests de validation :

Finalement les tests de validation seront effectués pour vérifier que le programme correspond bien aux exigences qui ont été spécifié dans les User Story.

**Moyen à mettre en œuvre :**

Des données de test pertinentes doivent être préparées.

Par exemple des noms de pays, des dates ou un fichier comportant des données.

Pour mettre en place ces données il faudra modifier les valeurs des constantes dans le code ou mettre en place un fichier .env.

Les constantes actuelles sont :  
FILEPATH qui contient l’emplacement du fichier contenant les données.

BEGINNINGYEAR qui contient la date de départ des données récoltés.

ENDINGYEAR qui contient la date de fin des données récoltés.

Afin de réduire le temps et l’effort nécessaire il serait préférable d’automatiser les tests dans un outil de CI/CD.

**Testeurs :**

Les testeurs éventuels de l’application seraient des utilisateurs finaux ou des camarades de classe, donc une communauté de développeur.

**Sprint Review :**

La sprint review peut se dérouler en présentiel comme en visioconférence.

Elle concernera toutes les personnes impliquées dans ce projet.

# Réalisation

## Points de design spécifiques

### Gestion des erreurs lors de l’appel à l’API

Au moment de faire appel à l’API dans le cas où une erreur avec l’API survient le graphique est quand même affiché avec les données locales.

### Utilisation de secret pour la clef API

Le jeton API est stocké dans un fichier .env. Dans le cas ou le fichier n’est pas trouvé les données locales sont utilisé.

### Traitement des devises non supportés

Affichage d’un message d’erreur si impossibilité d’utiliser l’API. L’application fonctionnera toujours avec les données locales.

***Ce chapitre est constitué de plusieurs sous-chapitre.***

***Chaque sous-chapitre explique un point de design technique particulier, quelque chose que vous avez dû inventer pour répondre au besoin et qui ne peut pas s’expliquer par de simples commentaires dans le code.***

***Il s’agit d’explications techniques sur le fonctionnement du système. Les explications sont appuyées par des diagrammes, ou de très brefs éléments de code.***

***NE PAS mettre ici des pratiques usuelles que tout professionnel de la branche connaît déjà. Par exemple, n’EXPLIQUEZ PAS ICI CE QU’EST LE PATTERN MVC.***

***Exemple (simplifié à l’extrême) : Protection contre des formulaires mal intentionnés ou modifiés***

* ***Au moment de générer le formulaire, le script php :***
  + ***Concatène les noms de tous les champs contenus dans le formulaire***
  + ***Calcule un hash SHA256 de la chaîne obtenue***
  + ***Ajoute un input nommé « CSRF » de type hidden dans le form***
* ***A la réception du POST du fromulaire***
  + ***Concatène les noms des indices de $\_POST***
  + ***Calcule un hash SHA256 de la chaîne obtenue***
  + ***Vérifie que la valeur du champ CSRF correspond***

## Déroulement

### Afficher le graphique

*En tant qu’utilisateur*

*Je veux sélectionner le pays*

*Pour afficher le graphique de ce pays*

Afin de réaliser cette user story j’ai décidé d’utiliser un Forms et d’utiliser le NuGet « ScottPlot » pour afficher le graphique. Une autre alternative aurait été d’utiliser les fonctions intégrées de base mais cela permettant moins de personnalisation j’ai décidé de m’orienter vers ScottPlot.

Les premières difficultés ont été l’utilisation de Forms, n’ayant pas effectué de projet avec cette technologie j’ai dû apprendre les syntaxes et comment cela fonctionnait.

Ensuite j’ai rencontré des difficultés avec le package ScottPlot car je souhaitais effectuer certaines choses que ScottPlot ne me permettait pas, par exemple l’affichage détaillé de chaque point.

### Effectuer un changement monétaire vers la même monnaie

*En tant qu'utilisateur*

*Je veux afficher les données dans la même devise*

*Afin de pouvoir les comparer*

Afin de réaliser cette user story j’ai décidé d’utiliser une API pour convertir les devises. J’ai premièrement rencontré des difficultés à trouver une API gratuite.

J’ai ensuite rencontré des difficultés à récupérer quelles devises étaient utilisé, mes données locales ne contenaient pas la devise mais le code du pays.

Le dernier problème rencontré venait d’une limite quotidienne de requête à l’API, cela m’a pris du temps avant de m’en rendre compte.

## Mise en place de l’environnement de travail

Le code source est stocké sur GitHub à l’adresse suivante : <https://github.com/joaberch/plot-that-lines.git>

Il peut être cloné avec cette commande :



3 - Commande pour cloner le dépôt

**Liste des fichiers :**

* README.md : Description du projet GitHub
* .gitignore :Décrit les fichiers à ne pas prendre en compte lors des commits.
* .env : Contient la clef de l’API.
* API\_MS.MIL.XPND.CN\_DS2\_fr\_csv\_v2\_3446916.csv : Donnée de test.

**Version des systèmes/outils :**

* IDE : Visual Studio Community 2022, Version 17.9.4
* Framework : .Net, version 8.0.202
* Bibliothèques externes : « ScottPlot.WinForms par Scott Harden », version 5.0.38

## Description des tests effectués

***Reprendre les tests d’acceptance d’IceScrum au moyen de la feuille ad hoc d’IceTools***

## Erreurs restantes

**Option de filtre :**

Description : Les options de filtre ne fonctionnent pas, elles avaient été anciennement implémentées dans un ancien commit mais la restructuration de l’application ne l’a pas restauré.

Conséquences : L’utilisateur ne peut pas appliquer d’options de filtre. Toutes les années sont affichés.

Actions envisagées : Récupérer les méthodes anciennement créées et les adapter au code actuel

**Changement de devise :**

Description : Lorsqu’un graphique est affiché et que la devise a modifié le graphique ne se met pas automatiquement à jour.

Conséquences : L’utilisateur ne peut pas automatiquement changer la devise, il devrait désélectionner et resélectionner le pays.

Action envisagée : Réafficher les graphiques lorsque la devise est modifiée.

# Conclusions

**Objectifs atteints :**

* L’application affiche un graphique contenant les données récupérées.
* La sélection des pays à afficher est fonctionnelle
* La conversion des devises en utilisant une API fonctionne.
* Dans le cas d’une erreur avec l’API les données sont toujours affichées selon les données locales

**Objectifs non-atteints :**

* Filtrage des données selon les années.
* Mise à jour automatique du graphique lors du changement de devise.

**Difficultés particulières :**

* Utilisation de l’API, intégrer l’API à été un problème pour gérer les erreurs de conversion et les données qui ne correspondaient pas à ce que l’API souhaitait.

**Suite possible :**

* Création et affinage des options de filtre.
* Mise à jour automatique du graphique selon la devise.
* Optimisation des requêtes à l’API, sauvegarder les données pour ne pas avoir à réeffectuer la même requête.
* Améliorer l’interface utilisateur pour la rendre plus simple à utiliser.

# Annexes

## Journal de travail

Le journal de travail est accessible dans le même dossier que le rapport.