|  |
| --- |
| Plot That Line |

Julien Mares

Table des matières

[1 Analyse préliminaire 3](#_Toc181176056)

[1.1 Introduction 3](#_Toc181176057)

[1.2 Objectifs 3](#_Toc181176058)

[1.3 Gestion de projet 4](#_Toc181176059)

[2 Analyse / Conception 4](#_Toc181176060)

[2.1 Domaine 4](#_Toc181176061)

[2.2 Concept 4](#_Toc181176062)

[2.3 Analyse fonctionnelle 4](#_Toc181176063)

[2.4 Stratégie de test 4](#_Toc181176064)

[3 Réalisation 4](#_Toc181176065)

[3.1 Points de design spécifiques 4](#_Toc181176066)

[*3.1.1* *…* 5](#_Toc181176067)

[*3.1.2* *…* 5](#_Toc181176068)

[*3.1.3* *…* 5](#_Toc181176069)

[3.2 Déroulement 5](#_Toc181176070)

[3.3 Mise en place de l’environnement de travail 5](#_Toc181176071)

[3.4 Description des tests effectués 6](#_Toc181176072)

[3.5 Erreurs restantes 6](#_Toc181176073)

[4 Conclusions 6](#_Toc181176074)

[5 Annexes 7](#_Toc181176075)

[5.1 Journal de travail 7](#_Toc181176076)

*NOTE L’INTENTION DES UTILISATEURS DE CE CANEVAS :*  
*Toutes les parties en italiques sont là pour aider à comprendre ce qu’il faut mettre dans cette partie du document. Elles n’ont donc aucune raison d’être dans le document final.*

*De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n’aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l’alourdir inutilement.*

# Analyse préliminaire

## 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 visée 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.

## Objectifs

### 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

### 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.

### 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

### Réalisation d’un rapport

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

## Gestion de projet

***Ce chapitre décrit la méthode de gestion de projet utilisée, ainsi que les éventuelles particularités requises par le contexte et/ou le chef de projet***

# Analyse / Conception

## 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 jours 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.

## Concept

* *Diagramme de classe*
* *Diagramme(s) d’état*

## Analyse fonctionnelle

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

## Stratégie de test

### Test d’acceptance

Il y a plusieurs tests d’acceptance pour chacune des User Stories, ils sont effectué par un dev qui va si 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.

### 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 tout les cas possible forcément mais on cherche une suffisance, on peut par exemple tester les extrêmes et 3-4 cas ordinaires, si ils réussissent on part du principe que c’est bon.

# Réalisation

## Points de design spécifiques

***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

***Résumer comment s’est passé la réalisation de chaque story, ses difficultés, les alternatives envisagées mais rejetées, ses surprises, …***

## Mise en place de l’environnement de travail

* ***Comment accéder au code source***
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*

***Ce chapitre décrit précisément comment un employé qualifié peut recréer l’environnement dans lequel vous avez effectué ce travail***

## Description des tests effectués

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

## Erreurs restantes

*S'il reste encore des erreurs :*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

***Reporter la*** [***dette technique***](https://www.premaccess.com/qu-est-ce-que-dette-technique-comment-la-maitriser/#:~:text=La%20dette%20technique%20survient%20quand,de%20plus%20en%20plus%20fr%C3%A9quents.) ***connue. S’appuyer sur la pratique des // TODO***

# Conclusions

*Développez en tous cas les points suivants :*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Journal de travail