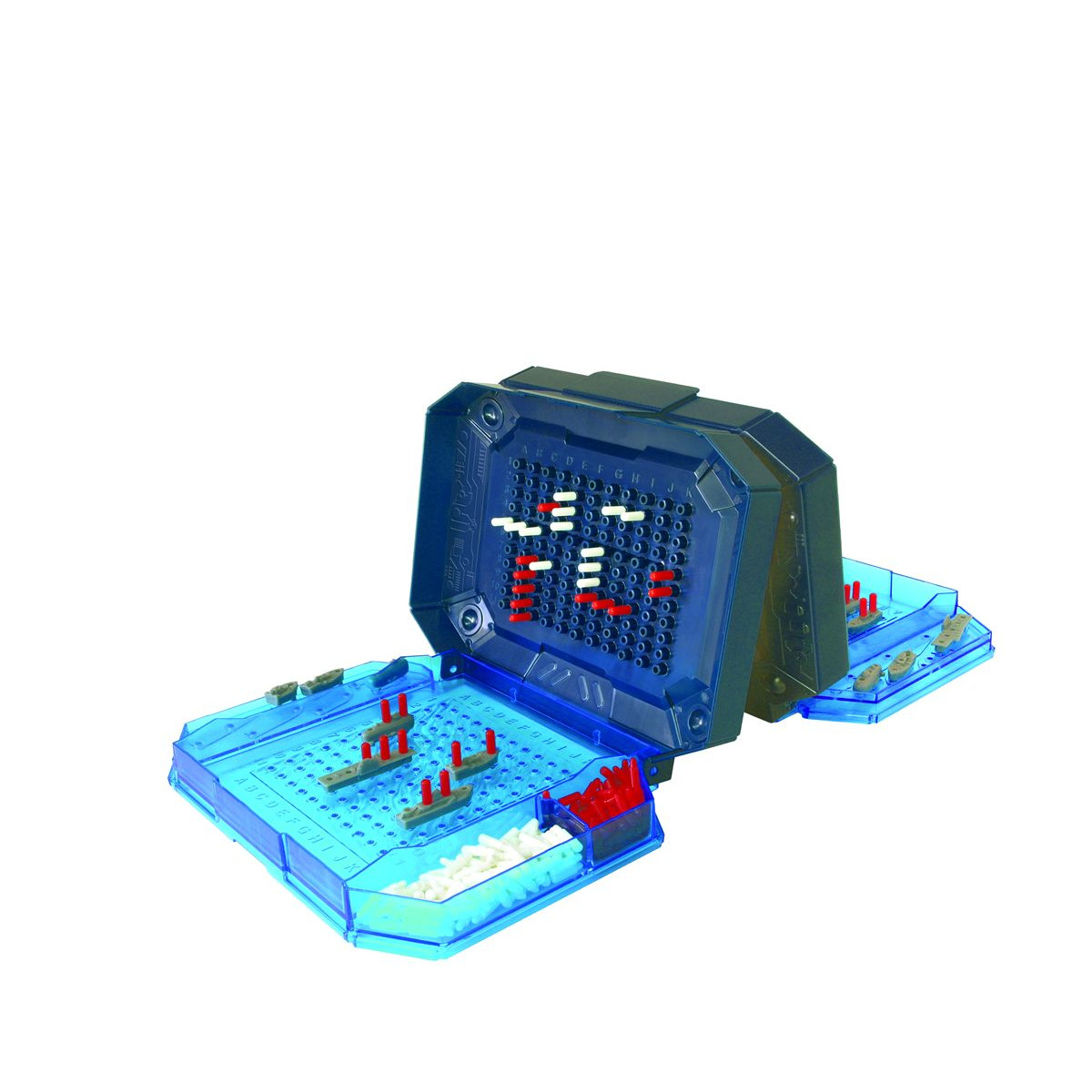


|  |
| --- |
| Bataille Navale |

**

*Meylan Benoit, Usan Sacha*

*Av. de la gare 14*

*1450, Ste-Croix*

*Benoit.Meylan@cpnv.ch, Sacha.Usan@cpnv.ch*



SI-C3b

*20.11.2018*

Table des matières

[1 Introduction 3](#_Toc531097758)

[1.1 Cadre, description et motivation 3](#_Toc531097759)

[1.2 Organisation 3](#_Toc531097760)

[1.3 Objectifs 3](#_Toc531097761)

[1.4 Planification initiale 3](#_Toc531097762)

[2 Analyse 4](#_Toc531097763)

[*2.1* Cahier des charges *détaillé* 4](#_Toc531097764)

[2.2 Analyse concurrentielle 4](#_Toc531097765)

[2.3 Cas d’utilisation 5](#_Toc531097766)

[2.4 Stratégie de test 7](#_Toc531097767)

[2.4.1 Tests 7](#_Toc531097768)

[2.4.2 Testeurs 7](#_Toc531097769)

[2.5 Planification 7](#_Toc531097770)

[3 Implémentation 7](#_Toc531097771)

[3.1 Dossier de conception 7](#_Toc531097772)

[3.1.1 Logiciels utilisés 7](#_Toc531097773)

[3.1.2 Diagramme UML 8](#_Toc531097774)

[3.2 Dossier de réalisation 9](#_Toc531097775)

[3.3 Dossier de réalisation 9](#_Toc531097776)

[3.3.1 Fichiers 9](#_Toc531097777)

[3.4 Description des tests effectués 9](#_Toc531097778)

[3.5 Erreurs restantes 9](#_Toc531097779)

[3.6 Dossier d'archivage 9](#_Toc531097780)

[4 Mise en service 10](#_Toc531097781)

[4.1 Installation 10](#_Toc531097782)

[4.2 Rapport de mise en service 10](#_Toc531097783)

[4.3 Liste des documents fournis 10](#_Toc531097784)

[5 Conclusions 10](#_Toc531097785)

[6 Annexes 10](#_Toc531097786)

[6.1 Sources – Bibliographie 10](#_Toc531097787)

[6.2 Journal de bord de chaque participant 10](#_Toc531097788)

[6.3 Manuel d'Utilisation 11](#_Toc531097789)

[6.4 Archives du projet 11](#_Toc531097790)

# Introduction

## Cadre, description et motivation

Dans le cadre du module « projet C# » nous avons décidé de réaliser une bataille navale qui dans un premier temps ne s’occupera que de la gestion des sauvegardes avec la gestion de fichiers. Dans un deuxième temps nous implémenterons la gestion du réseau qui permettra à deux joueurs connectés en LAN de jouer l’un contre l’autre.

## Organisation

Elève 1 : Usan, Sacha, Sacha.Usan@cpnv.ch

Elève 2 : Meylan, Benoit, Benoit.Meylan@cpnv.ch

Expert 1 : Andolfatto, Frederique, Frederique.Andolfatto@cpnv.ch

Expert 2 : Ithurbide, Julien, Julien.Ithurbide@cpnv.ch

## Objectifs

1. L’utilisateur aura la possibilité de charger une partie sauvegardée
2. L’utilisateur aura la possibilité de sauvegarder une partie en cours
3. Un utilisateur qui créer une partie pourra décider du nombre de navires que chacun aura.
4. Plusieurs utilisateurs pourront jouez en réseau local

## Planification initiale

|  |  |
| --- | --- |
| Date | Etape |
| 20.11.2018 | Début du projet et rendu de la définition du projet |
| 30.11.2018 | Fin du sprint 1 |
| 14.12.2018 | Fin du sprint 2 |
| 11.01.2018 | Fin du sprint 3 |
| 18.01.2018 | Finalisation du projet |
| 25.01.2018 | Préparation à la présentation |

# Analyse

## Cahier des charges *détaillé*

* Un menu permettant de lancer une nouvelle partie ou d’en charger une existante
* Lors du lancement d’une nouvelle partie on peut choisir le nombre de navires ainsi que leurs tailles
* Une partie se déroule suivant les règles de bases du jeu de bataille navale
* Une partie en cours peut être sauvegardée
* Une partie peut être créer soit en réseau soit en mode un joueur
* Dans le cas d’une partie en réseau si un joueur quitte la partie il perd.

## Analyse concurrentielle

## Cas d’utilisation

**Création d’une partie en mode un joueur**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| Le joueur se trouve sur le menu principal |  |
| Le joueur clique sur « Nouvelle partie » | L’utilisateur est redirigé sur un formulaire qui lui permet de paramétrer la partie (taille et nombre de navires, partie en réseau ?) |
| Le joueur a rempli les différents paramètres, sans cocher la case réseau et clique sur « Jouer » | Des bateaux sont placés aléatoirement sur la grille (non visible) et le jeu commence |

**Création d’une partie en réseau**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| Le joueur « A » se trouve sur le menu principal |  |
| Le joueur « A » clique sur « Nouvelle partie » | L’utilisateur est redirigé sur un formulaire qui lui permet de paramétrer la partie (taille et nombre de navires, partie en réseau ?) |
| Le joueur « A » a rempli les différents paramètres, et a coché la case réseau et clique sur « Jouer » | L’application le met en attente qu’un autre joueur ce connecte |
| Le joueur « B » se trouve sur le menu |  |
| Il clique sur le bouton « se connecter à une partie » | Un menu s’affiche avec des champs de configuration IP (et éventuellement un mot de passe) |
| Le joueur « B » a rempli les champs correctement (ip du joueur « A ») et clique sur ce connecter | Si la connexion est établie il se connecte à la partie du joueur « A ». Les joueurs peuvent placer leurs bateaux sur la grille |
| Les joueurs ont placé leurs bateaux et ont cliqué sur « Prêt » | La partie se lance, chacun des joueurs jouent à tour de rôle |

**Exemple d’un tour de jeu**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| C’est au tour du joueur « A » |  |
| Le joueur « A » clique en dehors de la grille | Rien ne se passe |
| Le joueur « A » clique sur une case, un bateau du joueur « B » s’y trouve | Un message « touché » est affiché sur l’écran du joueur « A ». Une croix est affichée sur l’emplacement où se trouvait le bateau sur l’écran du joueur « B ».  On vérifie si le joueur « B » a toujours des bateaux.  Si oui c’est au joueur B de jouer  Si non le joueur « A » a gagné |
| Dans le cas où le joueur « B » a encore des bateaux.  Le joueur « B » clique sur une case, aucun bateau du joueur « A » ne s’y trouve | Un message « Manqué » est affiché sur l’écran du joueur « B »  C’est autour du joueur « A » |

**Gagner la partie**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| C’est au tour du joueur « A » |  |
| Le joueur « A » clique sur une case de la grille, un bateau du joueur « B » s’y trouve | C’était le dernier bateau du joueur « B » il a donc perdu la partie. Retour pour chacun des joueurs sur le menu principale. |

**Sauvegarde d’une partie (1 joueur)**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| **Le joueur est en pleine partie** |  |
| **Il clique sur le bouton sauvegarder** | **Une fenêtre s’affiche le joueur doit rentrer un nom de partie** |
| **Le joueur a rentré un nom valide et clique sur ok** | **La partie est sauvegardée** |

**Charger une partie**

|  |  |
| --- | --- |
| Action utilisateur | Réaction de l’application |
| **Le joueur ce trouve sur le menu principal** | **Une fenêtre s’affiche avec la liste des parties sauvegardées** |
| **Le joueur sélectionne une partie puis clique sur le bouton « charger »** | **La partie se charge et le joueur est redirigé sur l’écran de jeu** |

**Bateau Coulé**

## Stratégie de test

### Tests

Lors de la phase de tests nous effectuerons cinq types de tests principaux.

1. Niveaux de tests
   1. Tests unitaires (est-ce que chaque partie fonctionnent individuellement)
   2. Tests d’intégration (est-ce que les différentes parties fonctionnent ensemble)
   3. Tests système (est-ce que les différentes fonctionnalités fonctionnent-elles chez le client)
2. Types de tests
   1. Test de robustesse (reprise après erreurs, gestion des erreurs)
   2. Test de performances (temps de réaction, écriture, lecture)
   3. Tests fonctionnels (est.ce )
   4. Test d’ergonomie (intuitif, accessible, simple d’utilisation, etc…)

### Testeurs

Des testeurs externes pourront nous aider, au quel cas nous le préciserons ci-dessous

## Planification

Pour ce projet nous avons décidé d’adopter une planification sous forme de sprints. Vous pourrez les retrouvez sur le repository du projet.

Notre repository : <https://github.com/SachaBenoit/Projet_Csharp/projects>

# Implémentation

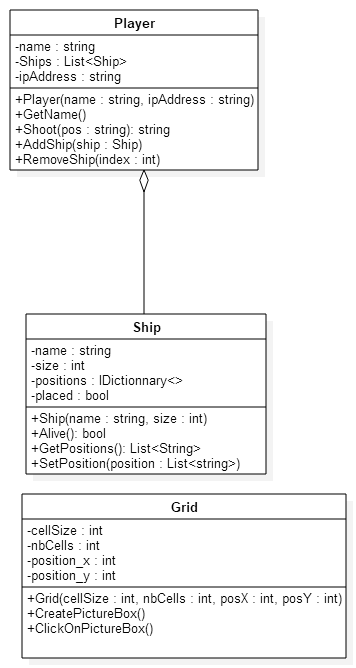
## Dossier de conception

### Logiciels utilisés

* Visual studio : développement
* StarUML : diagramme UML
* Word : documentation
* Pencil : maquette

### Diagramme UML

Ce diagramme représente les différentes classes nécessaire au programme.



## Dossier de réalisation

### Logiciel et système d’exploitation.

* Visual Studio 2017
* Windows 10 Education

### Numéro de version du produit

* V1.0

### Librairies

* WindowsForm : pour la partie graphique
* NewtonSoft.Json : pour la gestion des sauvegardes
* ActiveMQ : pour le multijoueur en réseau

### Classes

* FormMenu.cs
* FormNewPart.cs
* FormPart.cs
* Grid.cs : représente une grille
* Player.cs : représente un joueur
* Ship.cs : représente un bateau
* Target.cs : représente une case ciblée par le joueur

## Description des tests effectués

Pour chaque partie testée de votre projet, il faut décrire:

* les conditions exactes de chaque test
* les preuves de test (papier ou fichier)
* tests sans preuve: fournir au moins une description
* Il est recommandé de partir des Scénarios décrits dans l’anayse, complétés éventuellement par les modifications apportées à l’analyse.

Tous les tests ont été effectués par l’équipe de développement, nous n’avons malheureusement pas eu le temps de faire tester notre programme à des testeurs externes.

## Erreurs restantes

Cette partie répertorie les éventuelles erreurs restantes :

* Possibilité de d’empiler plusieurs bateaux l’un sur l’autre ou de les faire se croiser
* Le chargement d’une partie n’est pas implémenté
* Le multijoueur en réseau n’est pas implémenté

## Dossier d'archivage

Vous pouvez retrouver toutes les versions de notre projet dans notre repository sur git : <https://github.com/SachaBenoit/Projet_Csharp>

# Mise en service

## Liste des documents fournis

Fournir une procédure d’installation de votre produit.

Un utilisateur externe au projet (l’évaluateur de votre projet par exemple !) doit pouvoir:

* Prendre l’archive de votre projet telle que décrite en section 4.4
* Se procurer le matériel que vous avez décrit en 4.1
* Suivre votre procédure d’installation
* Se retrouver avec un système fonctionnel

# Conclusions

Développez en tous cas les points suivants:

* Objectifs atteints / non-atteints
* Comparaison entre ce qui avait prévu et ce qui s’est passé, en termes de planning et (éventuellement) de budget
* Points positifs / négatifs
* Difficultés particulières
* Suites possibles pour le projet (évolutions & améliorations)

# Annexes

## Sources – Bibliographie

* Documentation C# de MicroSoft : <https://docs.microsoft.com/en-us/dotnet/csharp/>
* ActiveMQ : <https://www.pmichaels.net/2016/09/29/a-c-programmers-guide-to-installing-running-and-messaging-with-activemq/>
* Gestion des threads : <https://openclassrooms.com/fr/courses/1243901-les-threads-en-net>
* Utilisation de la librairie newtonsoft : <https://www.newtonsoft.com/json/help/html/Introduction.htm>

## Manuel d'Utilisation

## Archives du projet

CD, DVD… dans une fourre en plastique.