

## Département Informatique

Filière :  
« Génie informatique »  
I-GI

# Rapport de projet AP4B



**Réalisé par :**

-MEZIANE Maryem.  
-BRUTON Lucas.  
-MARTIN Mickaël.

**Encadré par :**

-Mr. KAS Mohamed.  
-Mr. GECHTER Franck.

**Année Universitaire : 2021-2022**

# Remerciements

Il nous est agréable de nous acquitter d'une dette de reconnaissance auprès de toutes les personnes, dont l'intervention au cours de ce projet a favorisé son aboutissement.

Nous tenons à remercier tout particulièrement Mr.Kas Mohamed, notre professeur de TP pour sa disponibilité, ses précieux conseils et recommandations précieuses.

Nous tenons également à présenter nos sincères remerciements à nos professeurs pour nous avoir appris les démarches à suivre et les outils nécessaires pour la réussite de notre projet.

Pour finir, nous adressons nos remerciements à tout le corps enseignant de l'Université De Technologie De Belfort Montbéliard pour leurs efforts considérables, spécialement le département informatique.

# Liste des figures

<i>Figure 1 : activité d'un processus</i>	13
<i>Figure 2 : activités d'un processus de développement logiciel</i>	13
<i>Figure 3 : cycle de vie en Y</i>	15
<i>Figure 4 : méthode Scrum</i>	17
<i>Figure 5 : diagramme de contexte du système.</i>	26
<i>Figure 6 : diagramme de cas d'utilisation</i>	.26
<i>Figure 7: diagramme de séquence de lancement de l'application.</i>	27
<i>Figure 8: diagramme de séquence du lancement du panneau inférieur de l'affichage .</i>	28
<i>Figure 9: diagramme de séquence du lancement du panneau supérieur de l'affichage .</i>	28
<i>Figure 10 : diagramme de séquence de sélection de nombre des joueurs.</i>	29
<i>Figure 11 : diagramme de séquence d'initialisation du plateau KANAGUT .</i>	29
<i>Figure 12 : diagramme de séquence du panneau Attendre .</i>	30
<i>Figure 13: diagramme de séquence du panneau Action .</i>	30
<i>Figure 14 : diagramme de séquence du panneau titre.</i>	31
<i>Figure 15 : diagramme de séquence de l'affichage des UVs.</i>	32
<i>Figure 16 : diagramme de séquence de l'affichage des nouveaux choix des compétences.</i>	33
<i>Figure 17 : diagramme de séquence de récupération des nouvelles cartes compétences choisis.</i>	33
<i>Figure 18: diagramme de séquence de mise à jour des cartes compétences du joueur.</i>	34
<i>Figure 19: diagramme de séquence de mise à jour de choix de listener.</i>	35
<i>Figure 20: diagramme de séquence de déplacement de choix des compétence (partie 1).</i>	35
<i>Figure 21: diagramme de séquence de déplacement de choix des compétence (partie 2).</i>	35
<i>Figure 22: diagramme de séquence d'affichage de choix des compétence (partie 1).</i>	36
<i>Figure 23: diagramme de séquence d'affichage de choix des compétence (partie 2).</i>	37

<i>Figure 24: diagramme de séquence déplacement de choix des listener .</i>	37
<i>Figure 25: diagramme de séquence d'affichage de nouveaux choix des compétences .</i>	38
<i>Figure 26: diagramme de séquence de choix de parcours .</i>	38
<i>Figure 27: diagramme de séquence de choix de spécialité .</i>	39
<i>Figure 28: diagramme de séquence d'affichage du score .</i>	40
<i>Figure 29 : diagramme de classe du système (partie 1).</i>	41
<i>Figure 30 : diagramme de contexte du système(partie 2).</i>	42
<i>Figure 31 :Interface Choix de nombre des Joueurs.</i>	46
<i>Figure 32 :Interface Choix Carte UVs et Compétences.</i>	46
<i>Figure 33 :Interface Nouveau choix de compétences.</i>	47
<i>Figure 34 :Interface déplacement choix de compétences.</i>	47
<i>Figure 35 :Interface d'ajout des nouveaux Choix des Compétences.</i>	48
<i>Figure 36 :Interface de Choix des spécialité.</i>	48
<i>Figure 37 :Interface affichage du score.</i>	49

# Sommaire

## Table des matières

<b>Remerciements .....</b>	<b>2</b>
<b>Liste des figures.....</b>	<b>3</b>
<b>Sommaire .....</b>	<b>5</b>
<b>Résumé.....</b>	<b>7</b>
<b>Abstract.....</b>	<b>8</b>
<b>Introduction.....</b>	<b>9</b>
<b>Partie I : Contexte général du projet .....</b>	<b>11</b>
<b>Introduction : .....</b>	<b>11</b>
I. Présentation du projet .....	11
1. Origine du projet : .....	11
II. Conduite et gestion de projet .....	11
III. Gestion du projet : .....	12
1. Ingénierie et Démarche .....	12
IV. Contraintes du projet : .....	20
1. Planification du projet : .....	20
Diagramme de Gantt : .....	20
<b>Conclusion : .....</b>	<b>21</b>
<b>Partie II : Étude préliminaire et analyse des besoins.....</b>	<b>23</b>
<b>Introduction : .....</b>	<b>23</b>
I. Étude de l'existant.....	23
1. L'existant : .....	23
2. Solution proposée : .....	23
I. Description des besoins : .....	23
1. Les besoins fonctionnels : .....	23
2. Les besoins techniques : .....	24
<b>Conclusion : .....</b>	<b>24</b>
<b>Partie III : Conception du projet et architecture.....</b>	<b>26</b>
<b>Introduction : .....</b>	<b>26</b>
I. Architecture du projet : .....	26

1. Identification des acteurs : .....	26
2. Diagramme de contexte : .....	26
3. Diagramme de cas d'utilisation : .....	27
4. Diagramme de séquence boîte noire : .....	28
Initialisation du plan de jeu kanagut pour commencer a mettre en place les autres composants du jeu. .....	31
5. Diagramme de classe global : .....	39
<b>Partie IV : Réalisation et mise en œuvre</b> .....	43
I. Environnement de développement : .....	43
1. Outils logiciels : .....	43
2. Technologies de développement : .....	44
3. Langages : .....	45
II. Présentation de l'application .....	45
<b>BILAN</b> .....	51
I. Evaluation générale : .....	51
II. Problèmes rencontrés et résolution : .....	51
1. Problèmes rencontrés : .....	51
2. Résolution des problèmes : .....	51
III. Compétences acquises : .....	51
Conclusion .....	52
<b>Webographie</b> .....	53

# Résumé

Le présent document constitue le fruit d'un travail accompli dans le cadre du projet du AP4B qui nous a été confié lors de notre 5 -ème semestre à l'UTBM.

L'objectif du projet était de participer activement à la conception et la mise en œuvre du jeu Kanagawa adapté à un étudiant de l'UTBM.

Ce projet suit les pratiques de développement Agile et sa phase d'analyse et de conception qui a été abordée à l'aide du langage de modélisation UML et réalisé avec le processus de développement SCRUM.

De fil en aiguille, l'application a été réalisée et testée. Et cela représente la fin d'une expérience fructueuse qui s'est étalée sur deux mois en équipe, dont le résultat était concluant et très réussi.

# Abstract

This document is a result of work completed as part of the AP4B project which we were responsible for during our 5th semester at UTBM.

The objective of the project was to participate in the design and implementation of the Kanagawa game adapted to a UTBM student.

This project follows Agile development practices and its analysis and design phase which was developed using the UML modelling language and carried out with the SCRUM development process.

One thing leading to another, the application was carried out and tested. And this represents the end of a fruitful experience that spanned two months as a team, the result of which was conclusive and very successful.



# Introduction

Dans le cadre de notre formation en branche en Ingénierie Informatique, à l'Université de Technologie De Belfort Montbéliard (UTBM), nous sommes amenés à réaliser un projet Java qui a pour but principal de nous préparer à être des meneurs de projet et de développer l'esprit d'initiative, de créativité, d'innovation, d'organisation et de dynamisme du groupe.

Le document ci-dessous va définir tout d'abord le projet avec ses buts et ses finalités, ensuite la planification, puis l'analyse des besoins, l'architecture et la conception de l'application, et enfin un bilan pour tirer les objectifs atteints, les perspectives ainsi que les compétences acquises.

# Contexte général du projet

# Contexte général du projet

## Introduction :

Ce chapitre introductif a pour objet d'exposer le cadre général du projet en présentant l'origine de l'idée, les objectifs et enjeux, la méthodologie adoptée, l'équipe et les tâches de chaque membre ainsi que les contraintes et la planification du projet.

## I. Présentation du projet

### 1. Origine du projet :

L'objectif de ce projet n'est bien évidemment pas de refaire le jeu Kanagawa en Java à l'identique mais de s'inspirer de ce jeu pour en faire une adaptation selon notre sensibilité en intégrant des éléments de notre parcours à l'UTBM. On peut imaginer par exemple que chaque carte correspond à une UV et que le but de chaque joueur est de finaliser son cursus.

## II. Conduite et gestion de projet

La gestion de projet est une démarche visant à organiser de bout en bout le bon déroulement d'un projet. Lorsque la gestion de projet porte sur un ensemble de projets concourant à un même objectif, on parle de gestion de programme.

En pratique, le projet est tourné vers l'objectif final, il doit être adaptable à des modifications fréquentes, mais maîtrisé et planifié. Donc toute modification doit rester planifiée. Et notamment, le projet doit rester dynamique et équilibrer continuellement les contraintes techniques et de délai.

### III. Gestion du projet :

#### 1. Ingénierie et Démarche

La gestion de projet est une démarche visant à organiser de bout en bout le bon déroulement d'un projet. Lorsque la gestion de projet porte sur un ensemble de projets concourant à un même objectif, on parle de gestion de programme. En pratique, le projet est tourné vers l'objectif final, il doit être adaptable à des modifications fréquentes, mais maîtrisé et planifié. Donc toute modification doit rester planifiée. Et notamment, le projet doit rester dynamique et équilibrer continuellement les contraintes techniques, de coût et de délai.

Pour notre cas, nous étions sollicités pour prendre en charge deux contraintes majeures : livrer un produit de qualité et livrer à temps. Compte tenu de ces contraintes, de nos objectifs et des caractéristiques de notre mission, nous avons opté pour la méthodologie SCRUM avec des bonnes pratiques de XP pour mener notre projet, le choix de cette démarche avait pour raison le découpage du projet en plusieurs itérations. A chaque itération une partie du projet sera traitée et à la fin de l'itération devra être livrée.

##### a. Qualité du logiciel :

Le génie logiciel procède en plusieurs étapes dans la fabrication d'un programme informatique. Ceci permet de garantir la qualité du produit et le respect des besoins de l'utilisateur tout en respectant les délais fixés au départ.

Ainsi, notre application doit répondre aux principaux facteurs de qualité logiciel :

- **La validité** : c'est la capacité du logiciel à remplir exactement les tâches énoncées lors de sa spécification.
- **La fiabilité et la robustesse** : c'est l'aptitude du logiciel à fonctionner en continu et même dans des conditions anormales.
- **La compatibilité** : le logiciel doit pouvoir être combiné très facilement à d'autres logiciels.

- **La réutilisabilité** : l'application doit pouvoir être utilisée dans son entièreté ou en partie dans un autre projet.
- **L'efficacité** : le logiciel doit pouvoir utiliser efficacement les ressources matérielles.
- **L'extensibilité** : le logiciel doit être facile à maintenir, il doit pouvoir accepter l'ajout de nouvelles fonctionnalités.
- **La portabilité** : le logiciel doit pouvoir s'exécuter sous plusieurs environnements différents.
- **L'intégrité** : le logiciel doit pouvoir protéger efficacement son code entier ou une partie de son code contre les accès non autorisés.

Ainsi, afin d'assurer une qualité logicielle, il est nécessaire de maîtriser le processus de réalisation du logiciel. Afin d'y parvenir, nous avons :

- Identifié les objectifs à atteindre
- Défini le processus (définir l'activité ou l'ensemble d'activités qui utilise des ressources pour convertir des éléments d'entrée en éléments de sortie possédant une valeur ajoutée)
- Défini les rôles, les missions et les objectifs de chacun
- Mesuré et contrôlé les actions de chacun

Finalement la qualité du processus de développement est la garantie de la qualité du produit.

#### b. Activité et processus de développement du logiciel :

Un processus de développement est réalisé en se basant sur le concept de diviser pour mieux régner. Il désigne l'ensemble d'activités coordonnées et régulées, en partie ordonnées, dont le but est de créer un produit.



*Figure 1 : activité d'un processus*

Dans mon cas et généralement dans le cadre des projets de développement logiciel les activités sont communes :

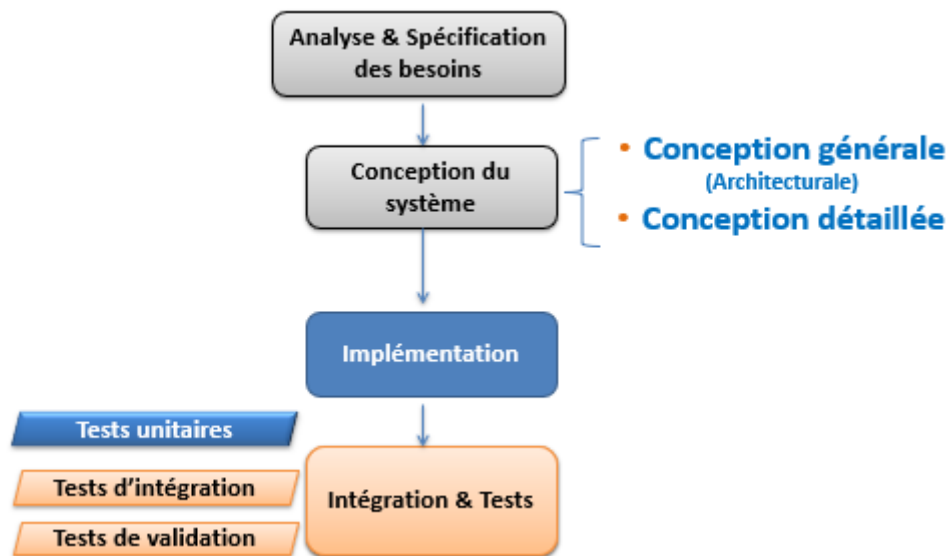


Figure 2 : activités d'un processus de développement logiciel

### c. Les cycles de vie d'un logiciel :

Afin de mettre en place un logiciel de qualité, il faut bien définir un processus de développement de qualité qui permet d'organiser les activités, guider les développeurs et fournir les moyens de développement et de maintenance pour répondre aux besoins instables du client.

Dans ce sens il existe deux approches pour un cycle de vie de logiciel :

Modèles classiques	<ul style="list-style-type: none"> <li>• Modèles stricts</li> <li>• Etapes très clairement définies</li> <li>• Fonctionne bien avec les gros projets et les projets gouvernementaux</li> </ul>
Méthodes agiles	<ul style="list-style-type: none"> <li>• Modèles incrémentaux et itératifs</li> <li>• Petites et fréquentes livraisons</li> <li>• Accent sur le code et moins sur la documentation</li> <li>• Convient aux projets de petite et moyenne taille</li> </ul>

Table 1 : différences entre les méthodes classiques et les méthodes agiles

- **Les modèles classiques :**

Nous avons rassemblé les modèles classiques dans le tableau suivant :

Modèle classiques	Principe
<b>Modèle en cascade</b>	Il définit des phases séquentielles à l'issue de chacune desquelles des documents sont produits pour en vérifier la conformité avant de passer à la suivante
<b>Modèle en V</b>	Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception.
<b>Cycle de vie incrémentale</b>	Se base sur l'approche « diviser pour régner » dont chaque incrément constitue un composant indépendant du logiciel
<b>Cycle de vie en spirale</b>	Il consiste à coder un peu (prototype), à tester puis à recommencer.
<b>Cycle de vie en Y</b>	S'articule autour de trois branches : <ul style="list-style-type: none"> <li>- Une branche technique : identifier les outils et les composants techniques à utiliser</li> <li>- Une branche fonctionnelle : identifier les exigences fonctionnelles</li> <li>- Une branche de conception et réalisation</li> </ul>

*Table 2 : Les modèles classiques*

### **Cycle de vie adopté :**

Nous avons choisi d'adopter un cycle de vie en Y dû à son pilotage de risque, sa nature itérative et incrémentale ainsi que la simplicité de son intégration avec la méthode agile Scrum qui sera introduite par la suite.

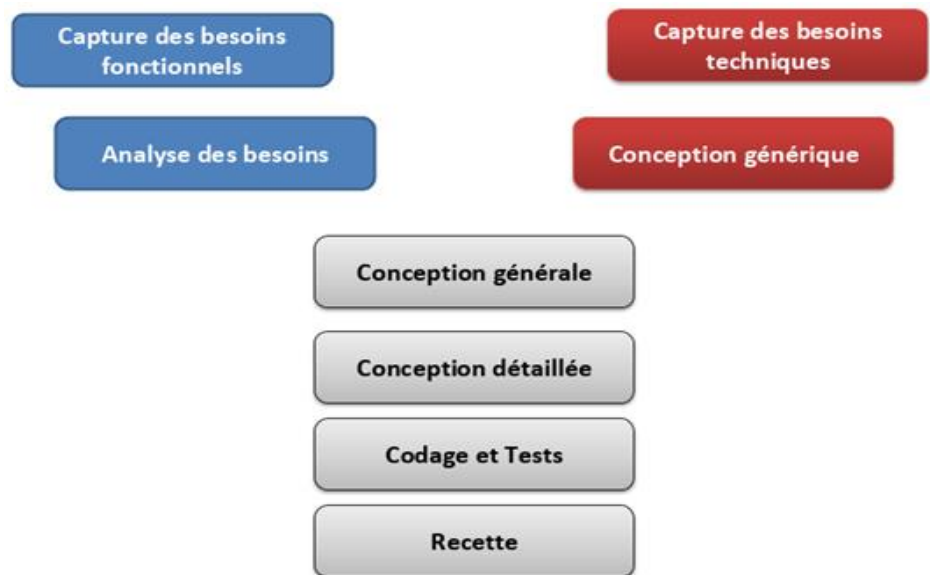


Figure 3 : cycle de vie en Y

### Les Méthodes agiles :

Parmi les principales méthodes agiles :

L'eXtreme Programming (XP)	Le principe fondamental de la méthode XP est de faire collaborer étroitement tous les acteurs du projet et d'opter pour des itérations de développement très courtes. La planification des tâches reste très souple et l'estimation des charges simplifiée par des projections à très court terme. Ainsi la correspondance entre ce qu'attend le client et les réalisations est garantie. Les fonctionnalités sont livrées régulièrement, afin d'être testées et validées au travers de prototypes opérationnels.
<b>Agile Unified Process (Agile UP ou AUP)</b>	<p>La méthode est divisée en quatre phases :</p> <ul style="list-style-type: none"> <li>- Lancement : identification du périmètre du projet, définition de la ou des architectures potentielles pour le système, implication des intervenants et obtention du budget.</li> <li>- Conception : définir l'architecture du système et démonstration de sa pertinence.</li> </ul>



	<ul style="list-style-type: none"> <li>- Réalisation : développement du logiciel lors d'un processus incrémental dans l'ordre de priorité des fonctionnalités.</li> <li>- Livraison : validation et déploiement du système en production.</li> <li>-</li> </ul>
<b>Méthode Scrum</b>	Cette méthodologie agile consiste à réaliser des fonctionnalités par itération en incluant la participation du client. Chaque itération peut durer de deux à quatre semaines, à la fin de chaque sprint un produit fonctionnel doit être livré.

*Table 3 : les méthodes agiles*

### **Méthode adoptée :**

Dans le cadre de mon projet et afin d'assurer le bon déroulement des différentes phases de ce dernier, j'ai opté pour la méthode agile Scrum pour la conception et le développement de mon système, en effet le processus Scrum s'adapte parfaitement à la décomposition de mon projet, il se base sur les avantages suivants :

- Plus de souplesse et de réactivité
- Grande capacité d'adaptation pour que l'application implémentée puisse répondre aux nouvelles exigences pendant le développement
- Satisfaire au mieux les besoins du client(utilisateur)
- Itératif et incrémental : le projet sera divisé en unités incrémentales appelées itérations. Le temps de développement de chaque itération est faible (deux semaines en moyenne), fixé et strictement respecté.

Scrum définit trois rôles principaux :

- **Le Product Owner** : qui est responsable du produit et représente les utilisateurs finaux.
- **Le Scrum master** : assure les tâches suivantes :
  - ◆ S'assurer que Scrum est bien appliquée et respectée.
  - ◆ Encourager l'équipe à apprendre et à progresser pour qu'elle soit fonctionnelle, productive et créative durant le projet.
  - ◆ Éliminer les obstacles pouvant perturber la progression du travail.

→ **L'équipe du projet** : composée de 4 membres. Elle regroupe tous les rôles nécessaires au projet, nous étions chargés d'accomplir toutes les tâches ayant une relation avec la conception et le développement. Le rôle principal de l'équipe est :

- ◆ Transformer les besoins exprimés en fonctionnalités utilisables.
- ◆ Livrer régulièrement une version fonctionnelle du produit.

La figure ci-dessous présente le déroulement de la gestion des projets par Scrum.



Figure 4 : méthode Scrum

#### a. Scénarii :

→ **Backlog produit** (ou catalogue des besoins ou scénarii) :

Élaboration d'une application KanaGUT qui se compose des modules suivants :

#### Quitter plate-forme inscription (1) :

- Le joueur choisit une colonne d'enseignements qui contient 1 carte.
- Le joueur fait son choix de compétences pour le tour pour pouvoir ajouter la carte à son parcours.
- Le joueur ajoute la carte à son parcours.

#### Quitter plate-forme inscription (2) :

- Le joueur choisit une colonne d'enseignements qui contient 2 cartes.
- Le joueur place les deux cartes dans ses compétences.

#### **Quitter plate-forme inscription (3) :**

- Le joueur choisit une colonne d'enseignements qui contient 3 cartes.
- Le joueur place deux cartes dans ses compétences.
- Le joueur fait son choix de compétences pour le tour pour pouvoir ajouter la 3e carte à son parcours.
- Le joueur ajoute la dernière carte à son parcours.
- Le joueur prend une spécialité de diplôme grâce aux nouveaux crédits acquis durant son tour.

#### **Quitter plateforme inscription (4):**

1. Le joueur choisit une colonne d'enseignements qui contient 2 cartes
2. Le joueur tente de placer les deux cartes dans son parcours. Ceci échoue puisqu'il n'a pas choisi les bonnes compétences. Les 2 cartes sont jetées .

#### **Attendre :**

- Le joueur attend au premier tour des choix des cartes d'enseignements pour pouvoir prendre deux cartes au prochain tour.

#### **Lancer le jeu :**

- Le joueur lance le jeu en sélectionnant le nombre de joueurs pour la partie.
- Le programme prépare tout le nécessaire pour le bon fonctionnement de la partie
- Les joueurs commencent à jouer.

#### **Arrêt du jeu :**

- Le joueur clique sur la croix de la fenêtre du programme pour arrêter la partie en cours.

→ **Sprint Backlog** : A chaque fois on doit se focaliser sur une partie du backlog qui est le sprint, pour notre cas nous allons commencer par le module de gestion des filières après la fin du sprint en passant au module suivant, et ainsi de suite jusqu'à la réalisation du dernier module.

#### **→ Sprint (itération) :**

- ◆ Développement des fonctionnalités du Backlog de sprint.
- ◆ Aucune modification du Backlog de sprint n'est possible.

- ◆ Mêlée quotidienne (Rencontre quotidienne).
- ◆ Point de contrôle quotidien de l'équipe.

→ **Produit livrable** : livrer au Product Owner à la fin du sprint le module réalisé.

## IV. Contraintes du projet :

Après le recensement du périmètre du projet, il a fallu énumérer et évaluer les contraintes que pourrait induire afin de les réduire ou de les éliminer par la suite.

Ainsi, les contraintes que nous jugeons ardues et stimulantes sont axées autour de :

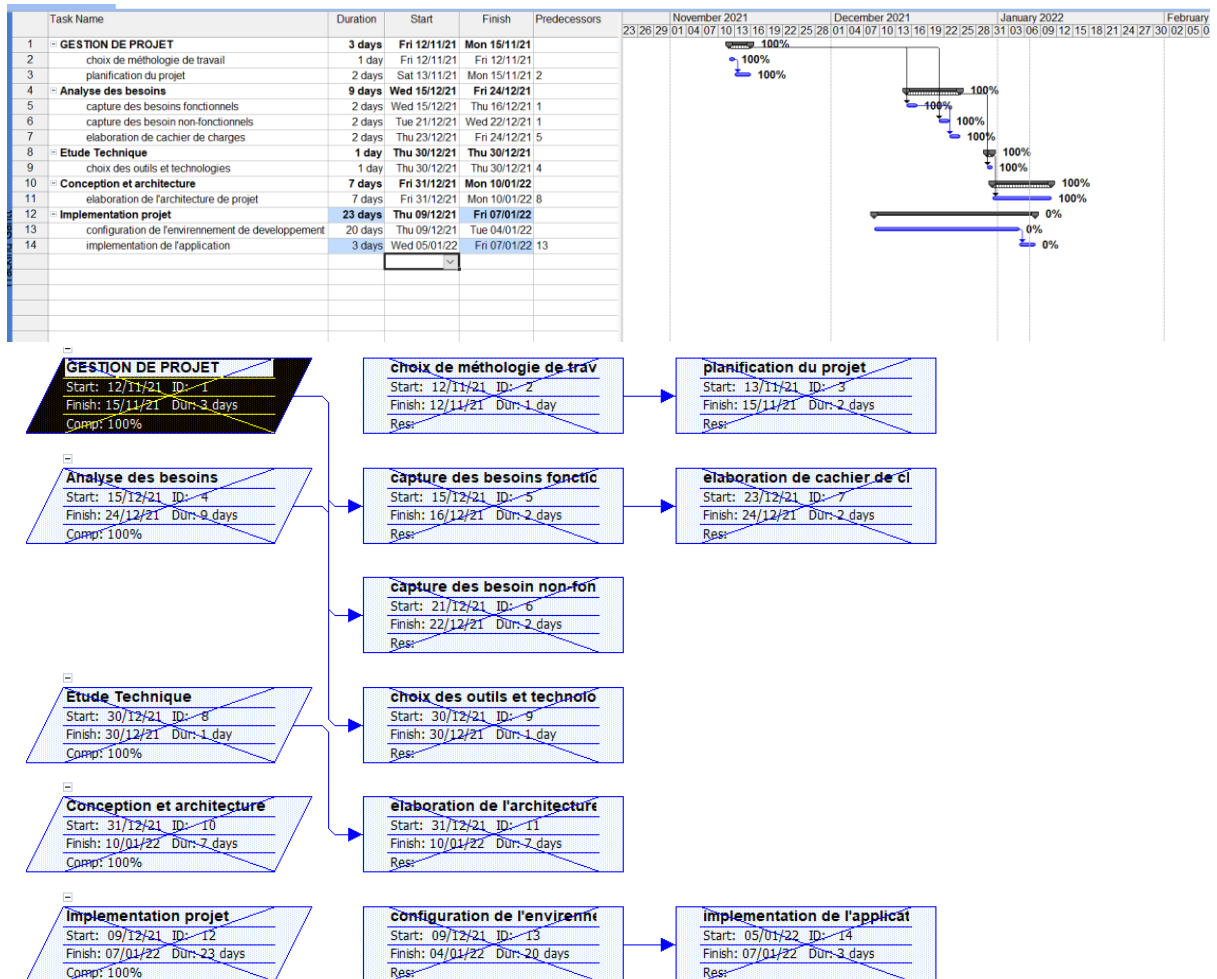
- **Contrainte de délai de réalisation** : la durée doit être répartie entre les différentes tâches à effectuer
- **Contraintes de conception et de développement** : l'application en vue de réalisation devra être flexible et facile à évoluer afin de pouvoir à tout moment y intégrer de nouvelles fonctionnalités.

### 1. Planification du projet :

La planification est une phase avant-projet. Elle consiste à délimiter le périmètre temporel du projet et prévoir le déroulement des tâches tout au long de la période allouée au projet. Nous avons prévu que notre projet sera planifié comme suit :

#### Diagramme de Gantt :

Ce planning nous a permis de maîtriser la gestion du temps alloué pour la réalisation du projet.



## Conclusion :

Après avoir présenté le projet de façon générale et la méthode de travail nous passerai à l'étude préliminaire qui s'avère primordiale pour le bon déroulement du projet.

# Étude préliminaire et analyse des besoins

# Étude préliminaire et analyse des besoins

## Introduction :

Ce chapitre est décomposé en trois sections importantes. Tout d'abord, une étude préliminaire sur l'existant, ses limites et la solution proposée. Par la suite une description détaillée des besoins fonctionnels et non fonctionnels de la solution.

### I. Étude de l'existant

#### 1. L'existant :

Kanagawa est un jeu de cartes stratégique et poétique dans lequel vous devez réaliser la plus belle estampe. Développez votre Atelier pour immortaliser les plus beaux sujets au fil des saisons, et devenez le plus prestigieux élève du peintre Hokusai. Sélectionnez avec soin vos cartes d'enseignement, placez-les dans votre atelier ou dans votre estampe. Peignez les plus beaux sujets et devenez le digne héritier d'Hokusai !

#### 2. Solution proposée :

Dans le but d'adapter le jeu Kanagawa à un étudiant de L'UTBM nous avons pensé à rendre l'atelier de cette jeu un projet d'étude dans lequel les joueurs (étudiants) choisissent leurs formations, leurs compétences ainsi que leurs propres parcours. Et celui qui aura le plus de points à la fin du jeu est nommé gagnant.

### I. Description des besoins :

#### 1. Les besoins fonctionnels :

L'étude de l'existant réalisée au préalable nous a permis de mieux cerner les besoins fonctionnels auxquels notre solution doit répondre. Notre application se compose de plusieurs interfaces. Et chaque partie de l'application a des besoins fonctionnels spécifiques.

L'application permet de répondre aux besoins suivants :

## 2. Les besoins techniques :

Les besoins techniques sont les principaux facteurs de qualité logiciel que nous avons indiqué précédemment :

- La portabilité
- La validité
- La fiabilité et la robustesse
- La compatibilité
- La réutilisabilité
- L'efficacité
- L'extensibilité
- L'intégrité

## Conclusion :

L'étude préliminaire s'est avérée obligatoire pour mieux détailler les besoins et simuler le fonctionnement de notre solution, elle vient avant la conception détaillée qui regroupe les diagrammes nécessaires et l'architecture utilisée dans la réalisation de l'application, c'est donc le sujet du chapitre suivant.



# Architecture et conception

# Conception du projet et architecture

## Introduction :

Dans ce chapitre, j'expose d'abord l'architecture technique et l'architecture applicative de la solution, avant d'entamer ensuite la phase de conception de chaque module de l'application.

## I. Architecture du projet :

L'architecture est l'ensemble des aspects techniques et applicatifs qui sont importants pour un logiciel. Le choix des architectures influe sur la réussite ou l'échec d'un projet. Dans cette partie, j'expose l'architecture technique cible de la solution ainsi que l'architecture applicative.

### 1. Identification des acteurs :

Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. Dans cette partie, nous allons énumérer les acteurs susceptibles d'interagir avec notre application. Nous distinguons quatre acteurs :

- les joueur (étudiants à l'UTBM)

### 2. Diagramme de contexte :

Le diagramme de contexte est un diagramme UML qui constitue une étape intermédiaire entre le cahier des charges et la construction des premiers cas d'utilisation. Il présente le système à modéliser, en général sous la forme d'une boîte noire et les différents acteurs qui interagissent avec ce système.

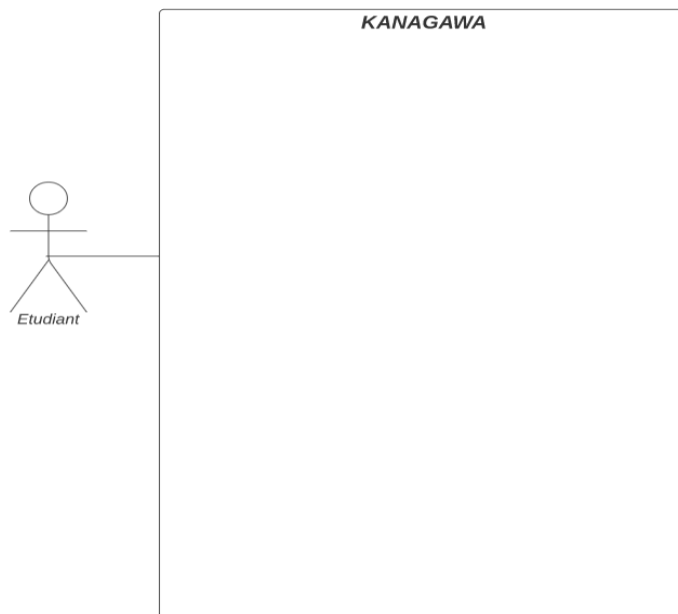


Figure 5 : diagramme de contexte du système.

### 3. Diagramme de cas d'utilisation :

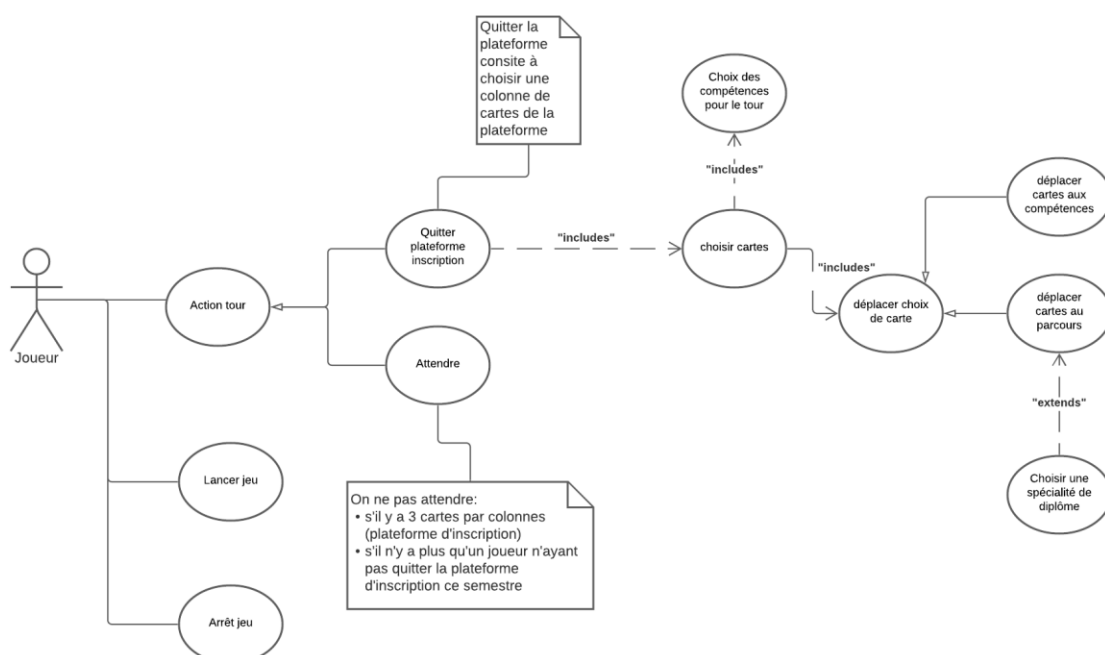


Figure 6 : diagramme de cas d'utilisation.

Le diagramme de cas d'utilisation ci-dessus présente les ensembles des cas d'utilisation. Dans ce module, le système doit permettre au joueur d'accéder facilement au jeu et de commencer son parcours à l'UTBM.

#### 4. Diagramme de séquence boîte noire :

Pour bien détailler le diagramme de cas d'utilisation nous allons présenter le diagramme de séquence boîte noire pour montrer comment nous allons réaliser les cas d'utilisation de ce module. Nous allons présenter le diagramme de séquence qui décrit le processus de jouer et affichage des interfaces.

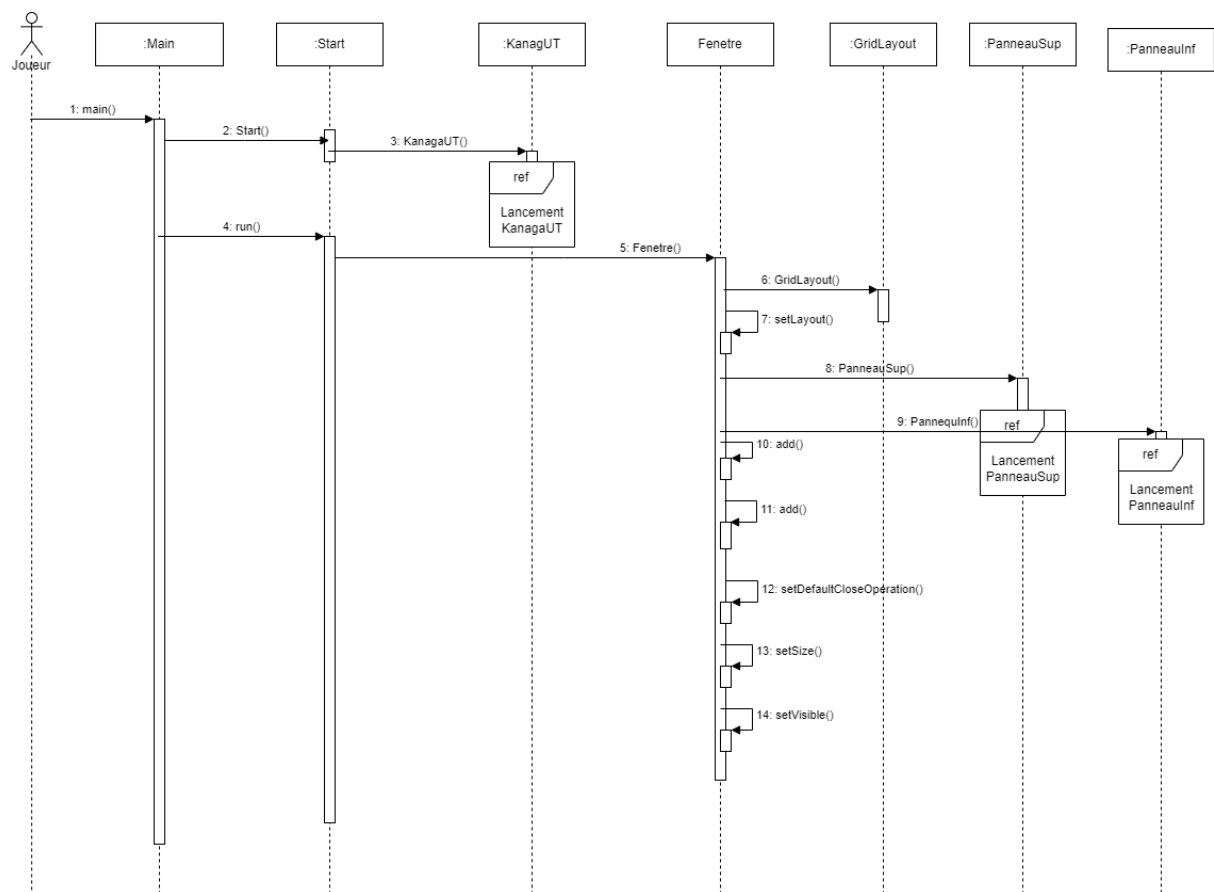


Figure 7: diagramme de séquence de lancement de l'application.

Ce diagramme représente l'interaction entre les objets pour Lancer l'application. Après interprétation du code et le lancement de main il fait appel au différente classe de l'application pour assurer le bon déroulement du jeu.

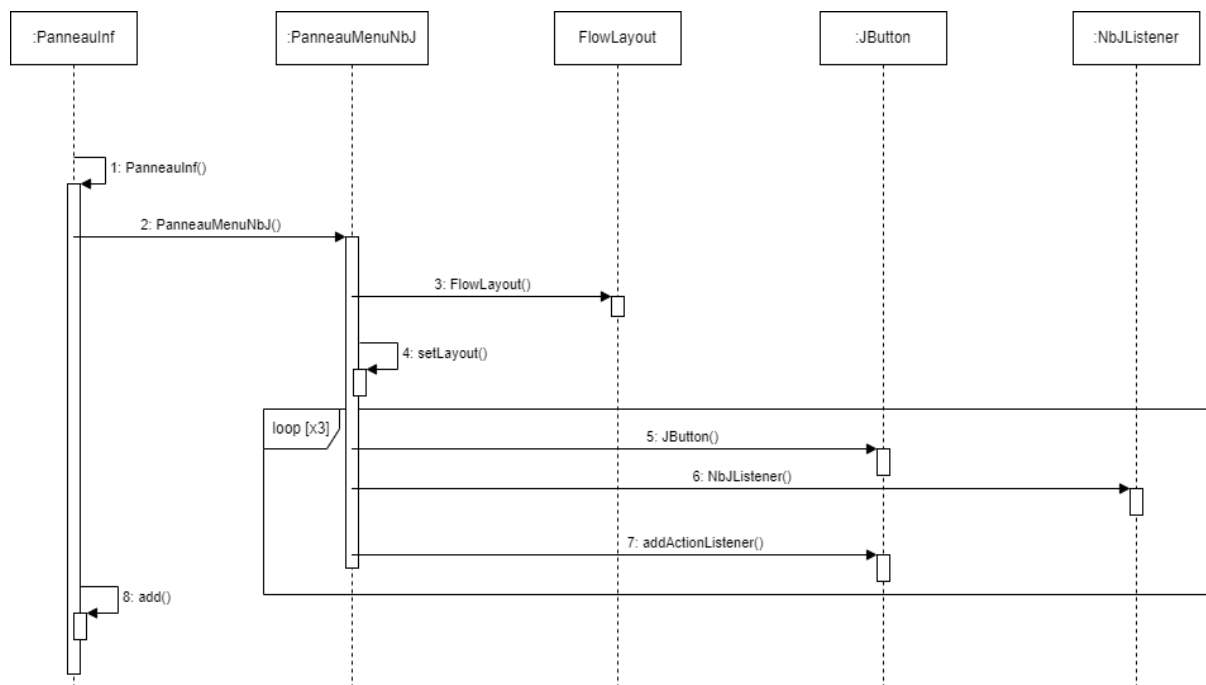


Figure 8: diagramme de séquence du lancement du panneau inférieur de l’affichage du nombre des joueurs .

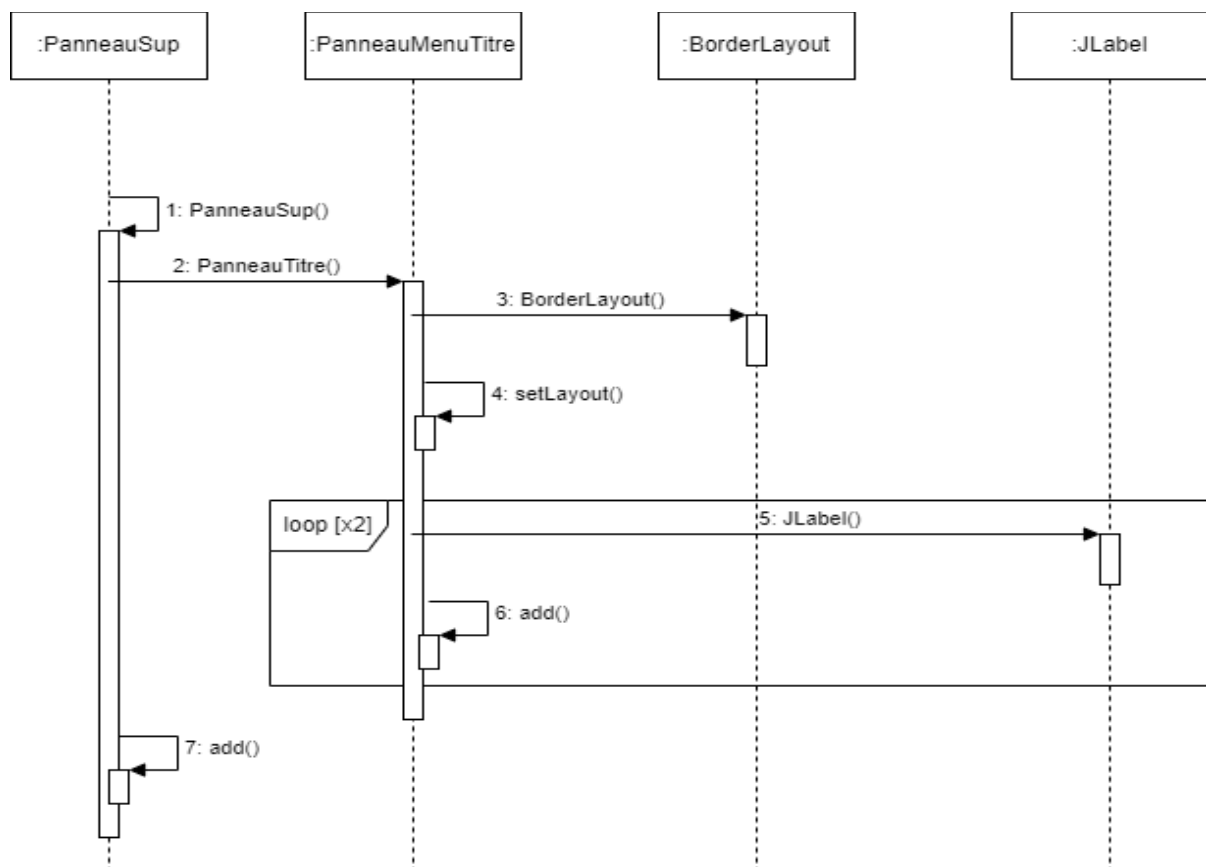


Figure 8: diagramme de séquence du lancement du panneau supérieur de l’affichage du 1ere interface.

Lancement des objets nécessaires pour former une interface correcte pour l'application.

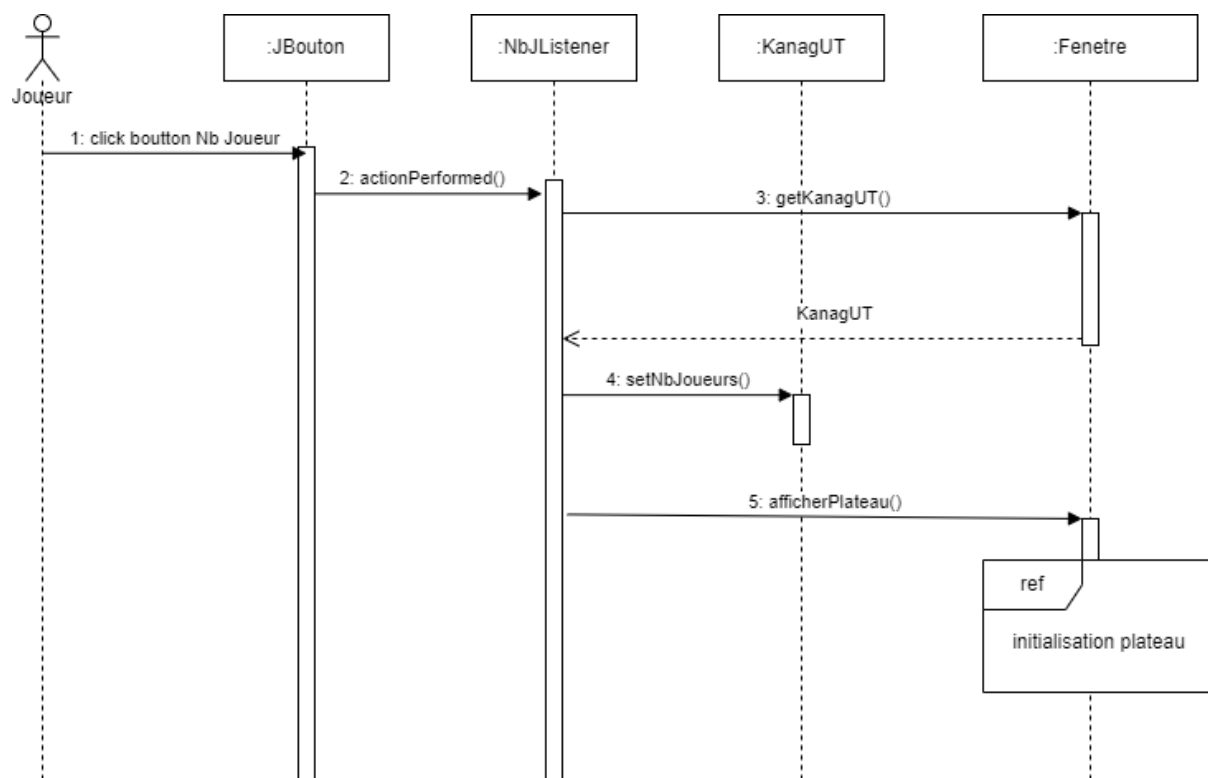


Figure 9 : diagramme de séquence de sélection de nombre des joueurs .

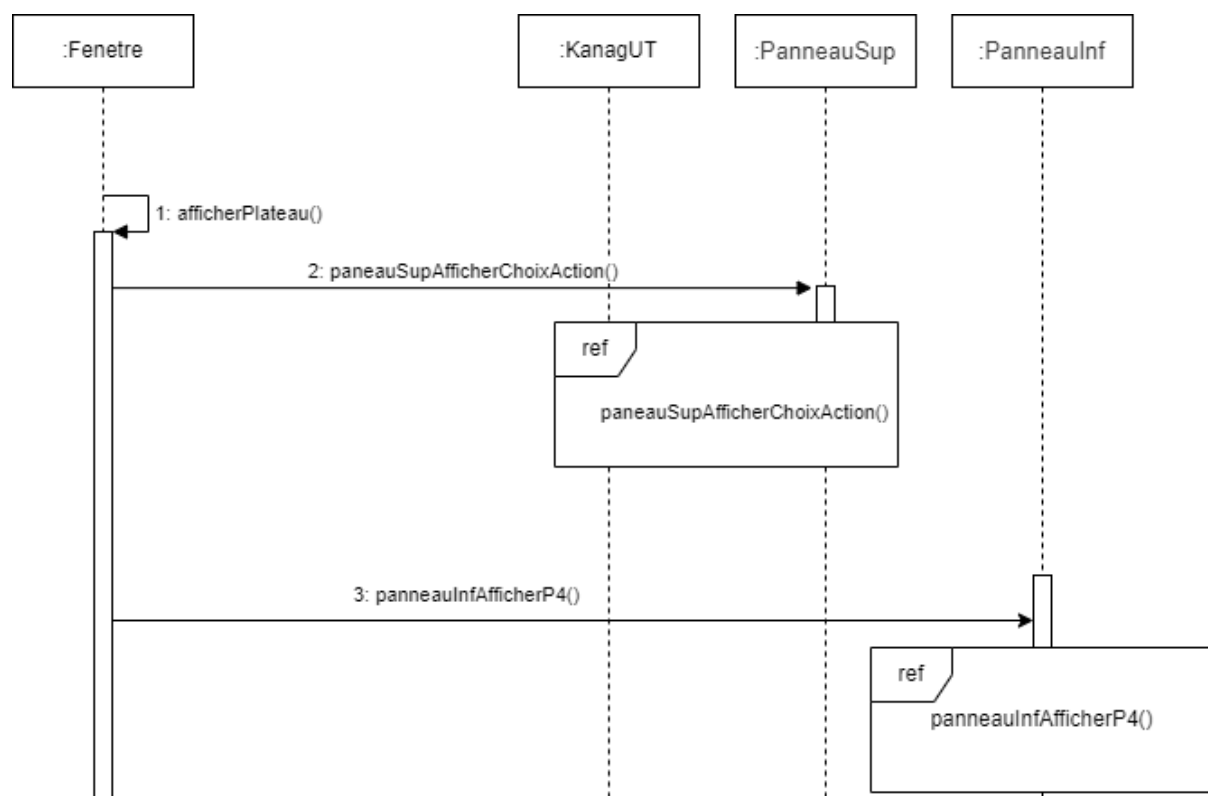


Figure 10 : diagramme de séquence d'initialisation du plateau KANAGUT .

Initialisation du plan de jeu kanagut pour commencer a mettre en place les autres composants du jeu.

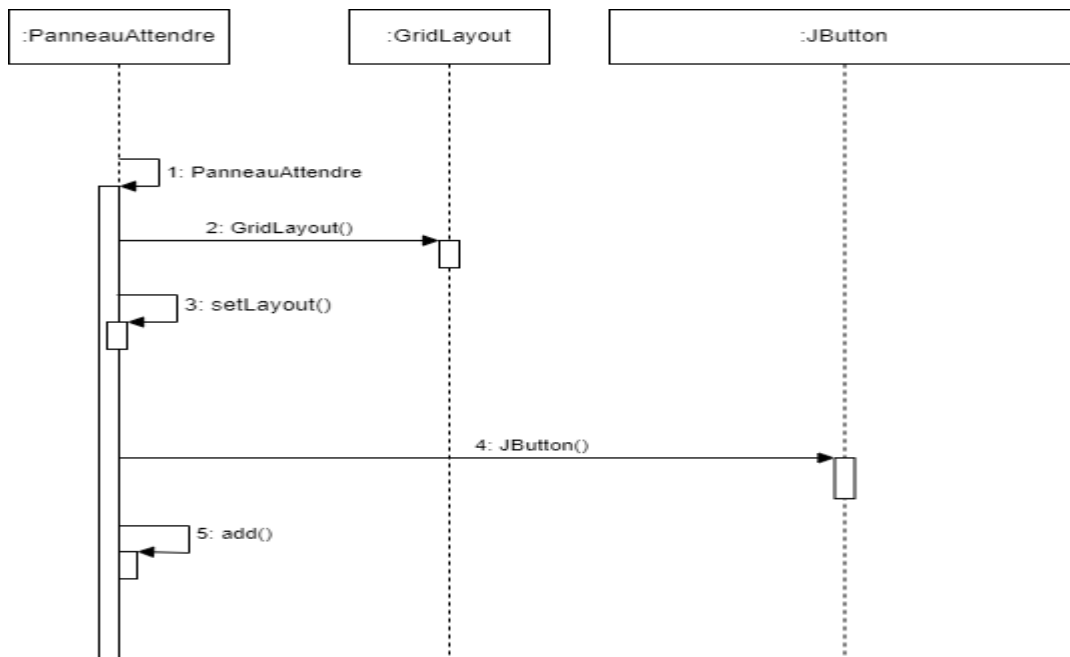


Figure 11 : diagramme de séquence du panneau Attendre .

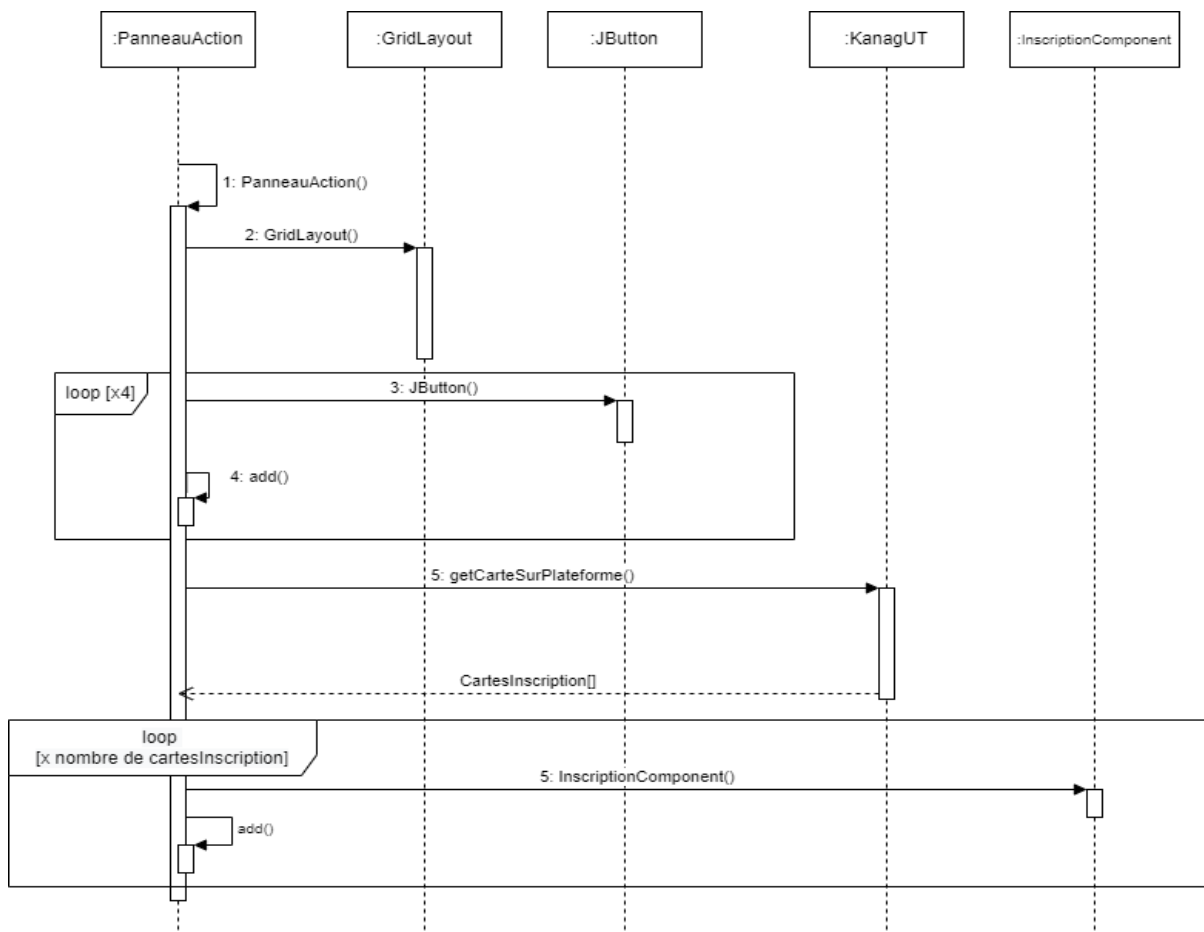


Figure 12 : diagramme de séquence du panneau Action .

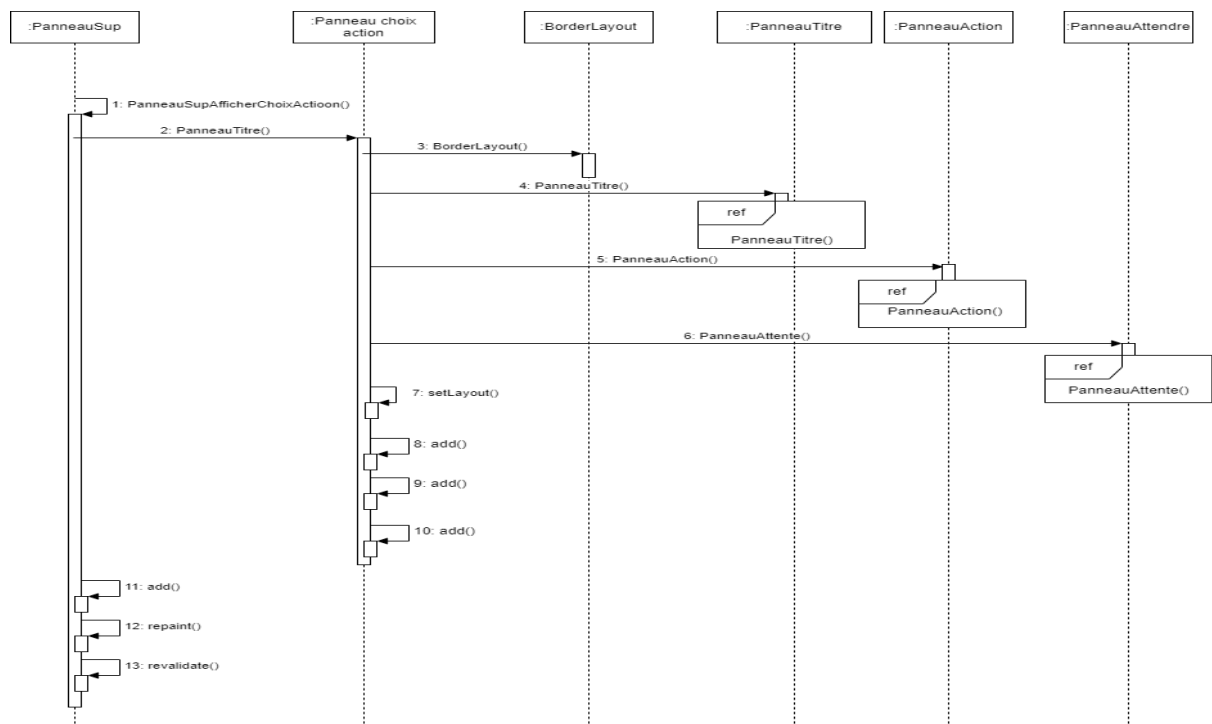


Figure 13 : diagramme de séquence du panneau pour choisir l'Action .

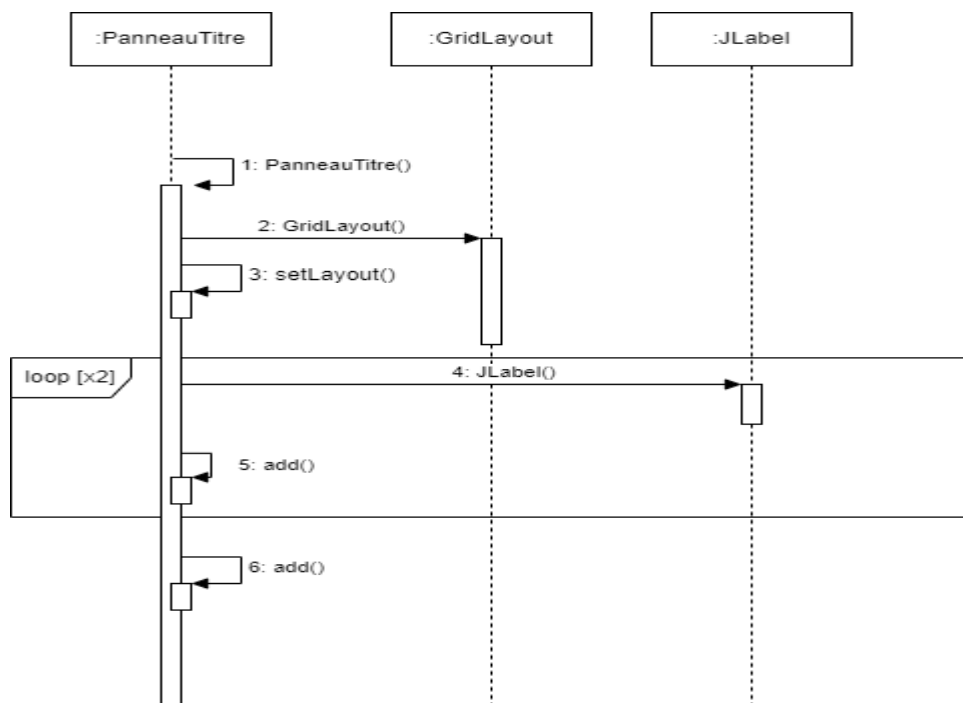


Figure 14 : diagramme de séquence du panneau titre.



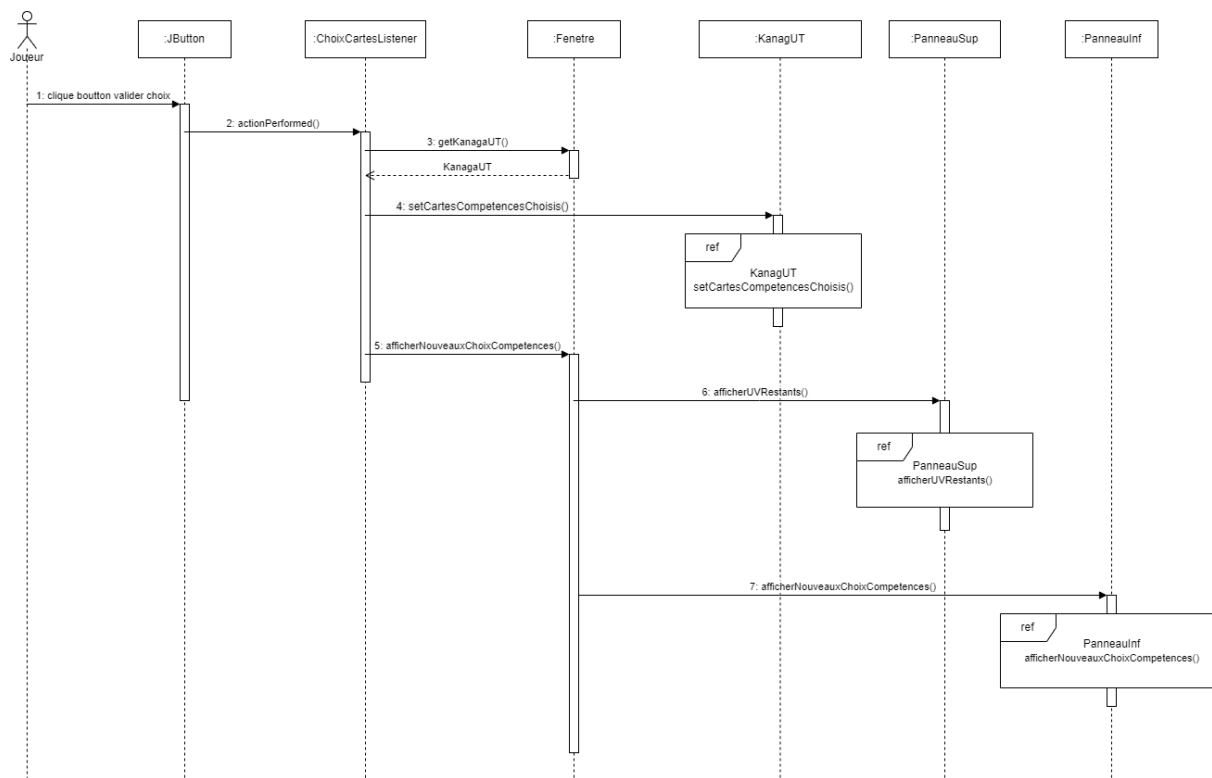


Figure 14 : diagramme de séquence de l'interface d'ajout de choix de compétences.

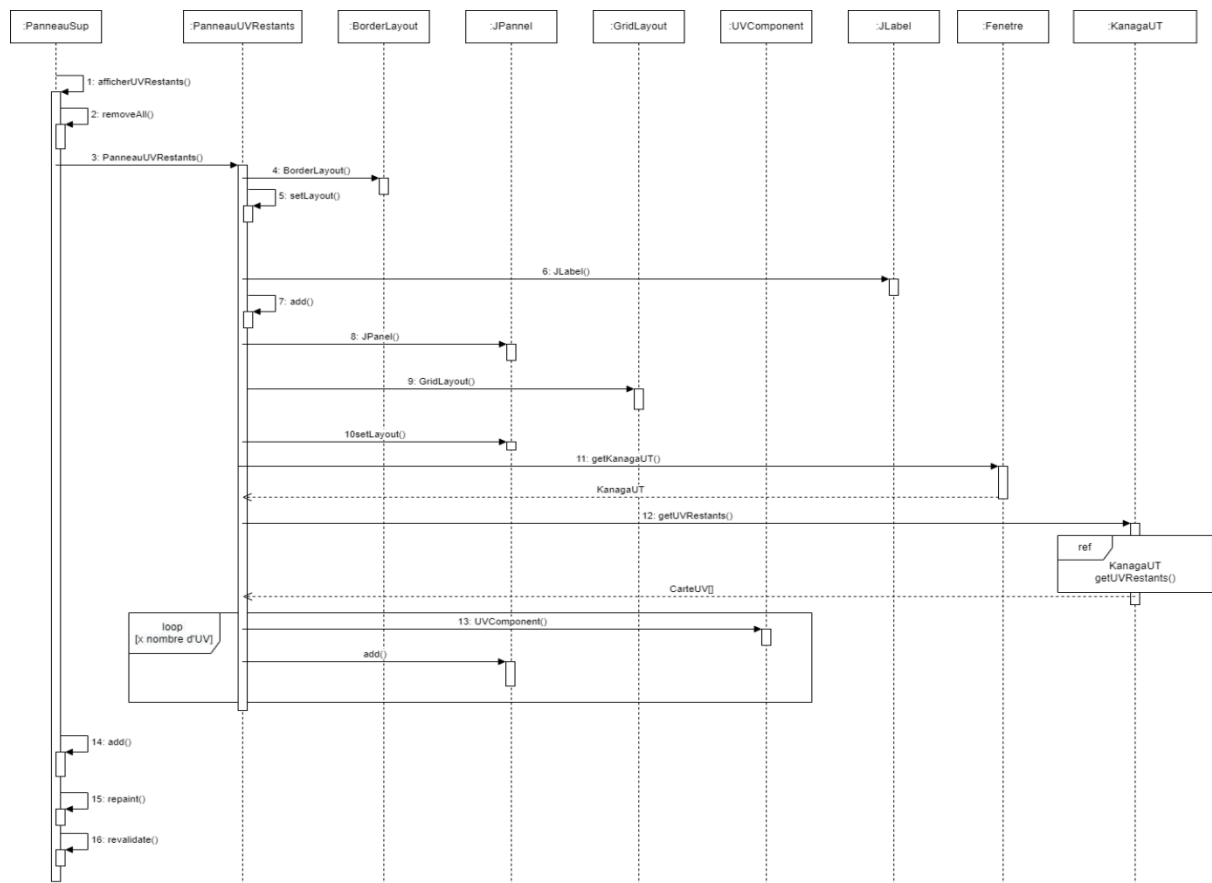


Figure 15 : diagramme de séquence de l'affichage des UVs.



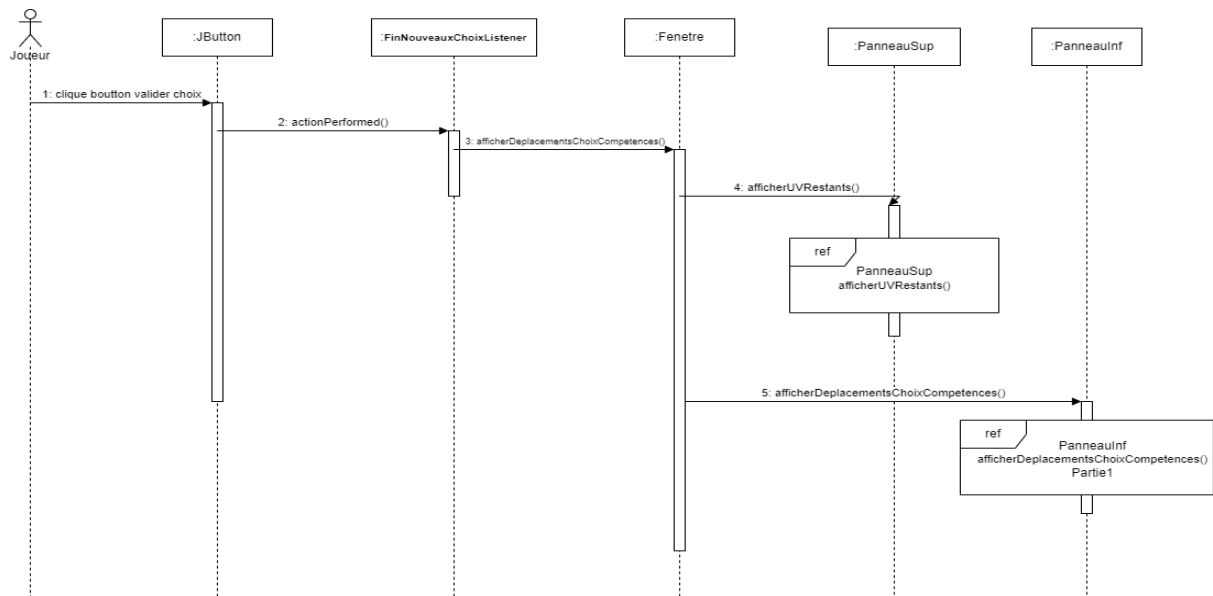


Figure 19: diagramme de séquence de mise à jour de choix de listener.

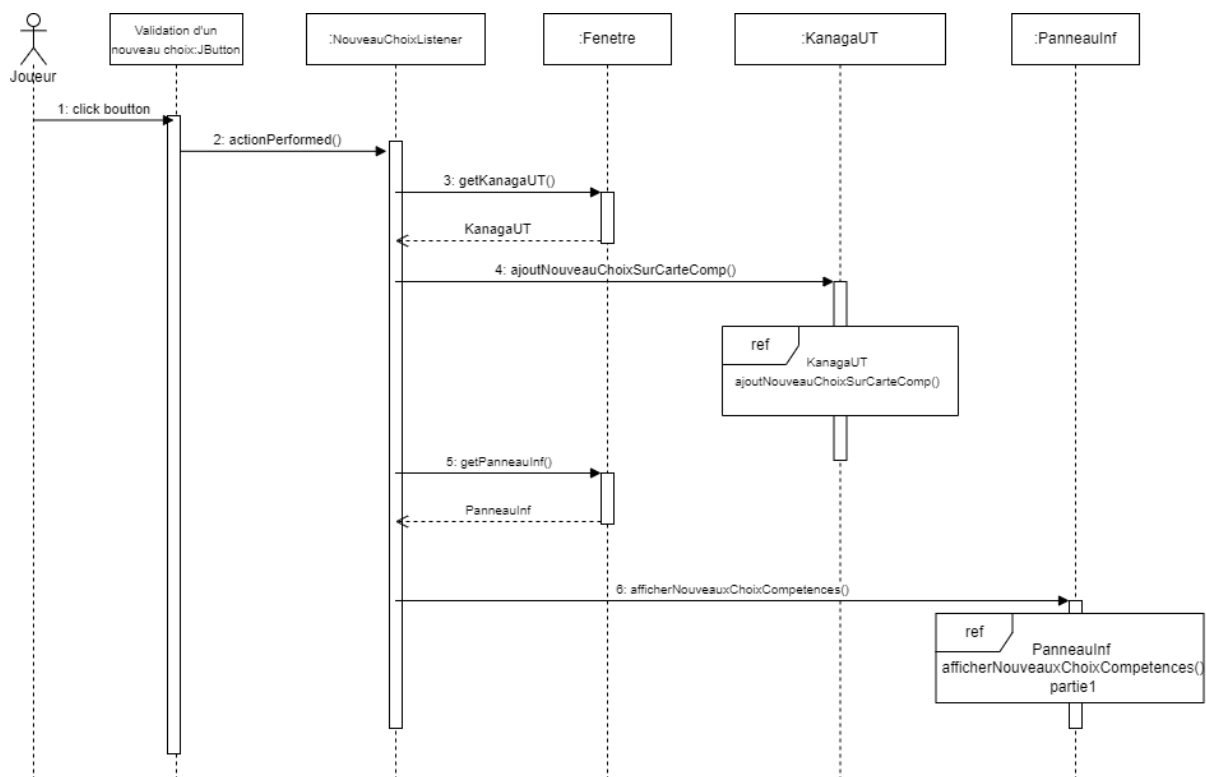


Figure 20: diagramme de séquence de nouveau choix de listener .



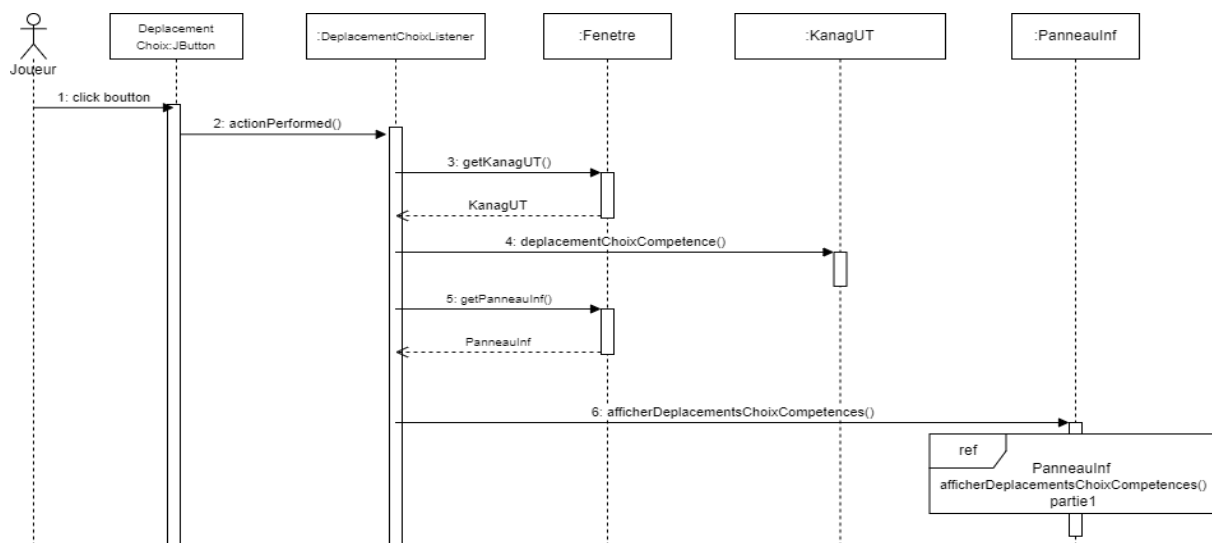


Figure 24: diagramme de séquence déplacement de choix des listener .

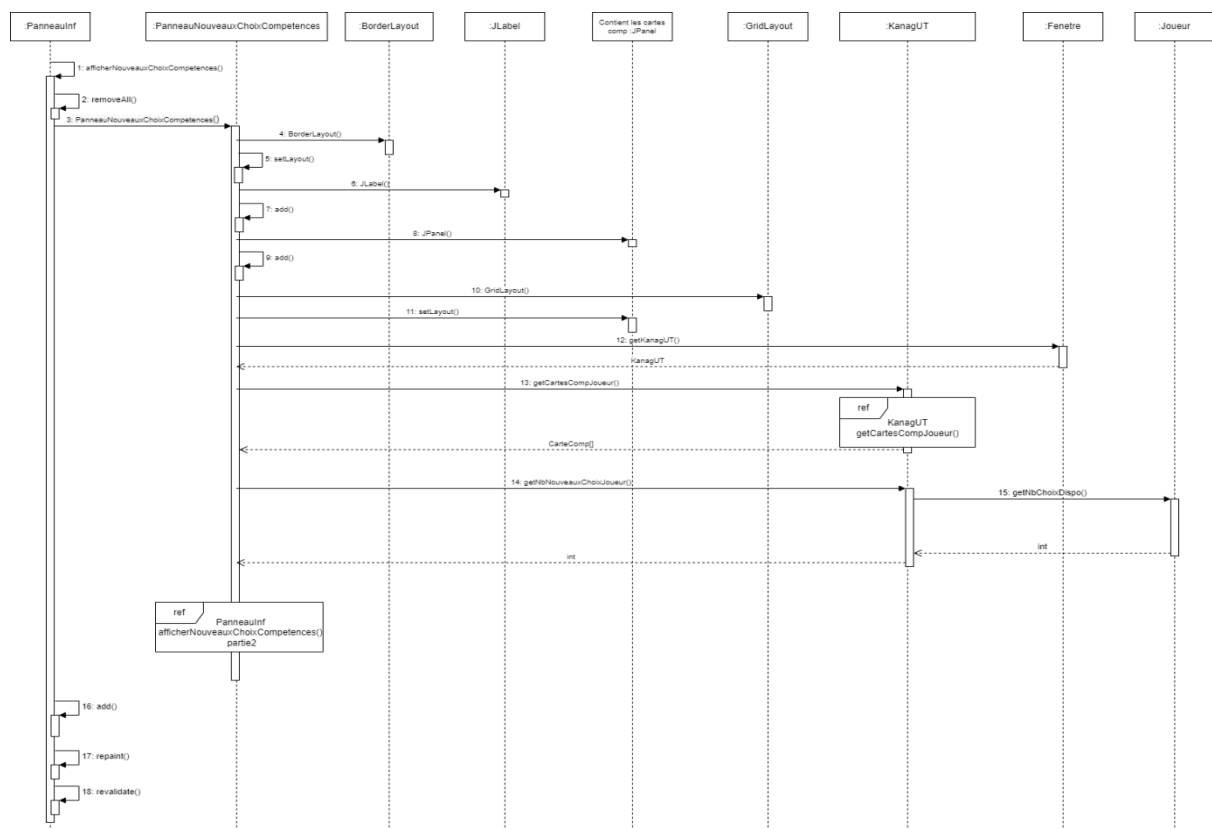


Figure 25: diagramme de séquence d'affichage de nouveaux choix des compétences .

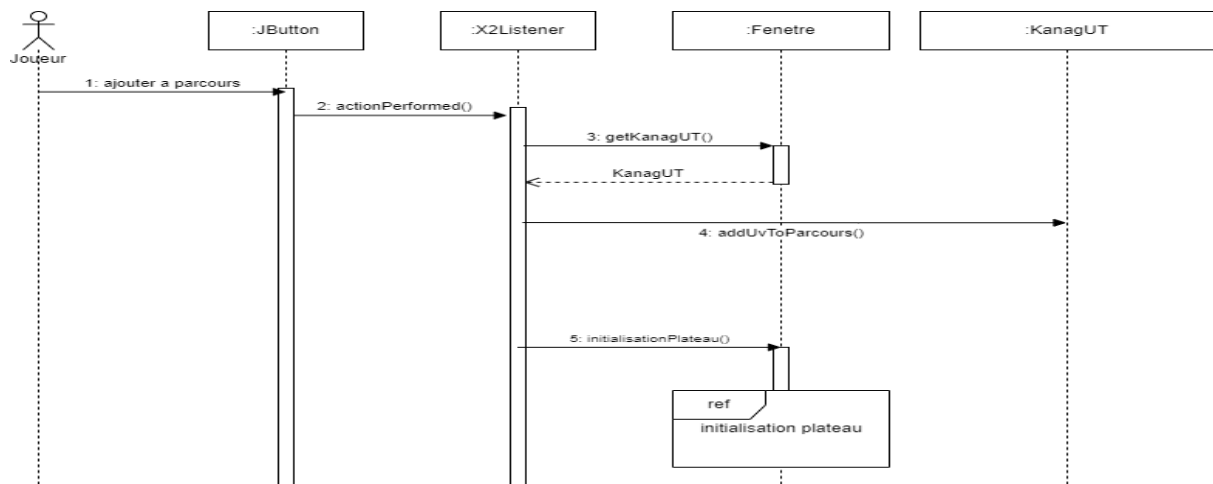


Figure 26: diagramme de séquence de choix de parcours .

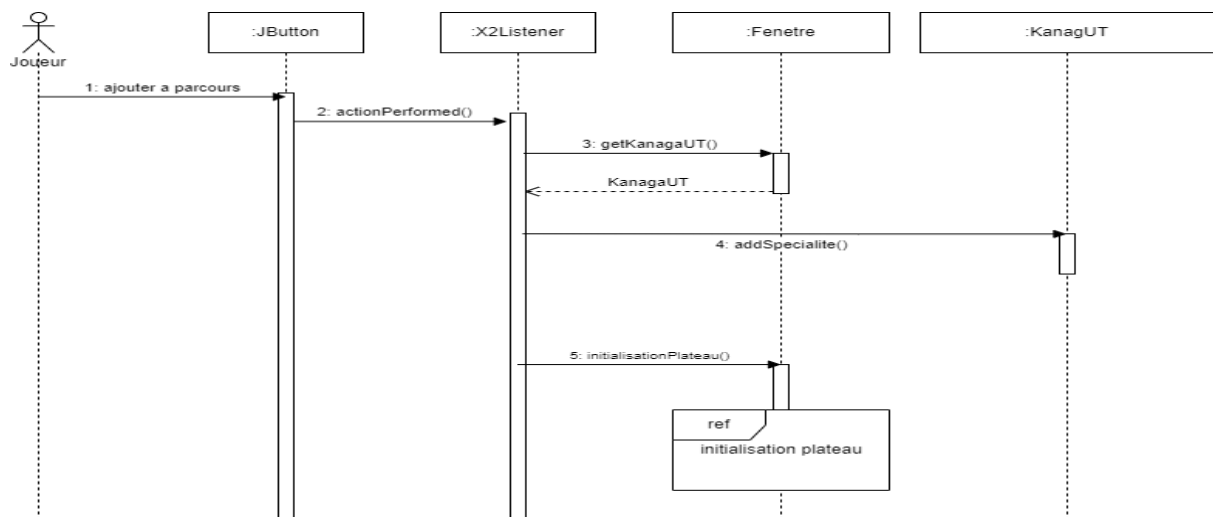


Figure 27: diagramme de séquence de choix de spécialité .

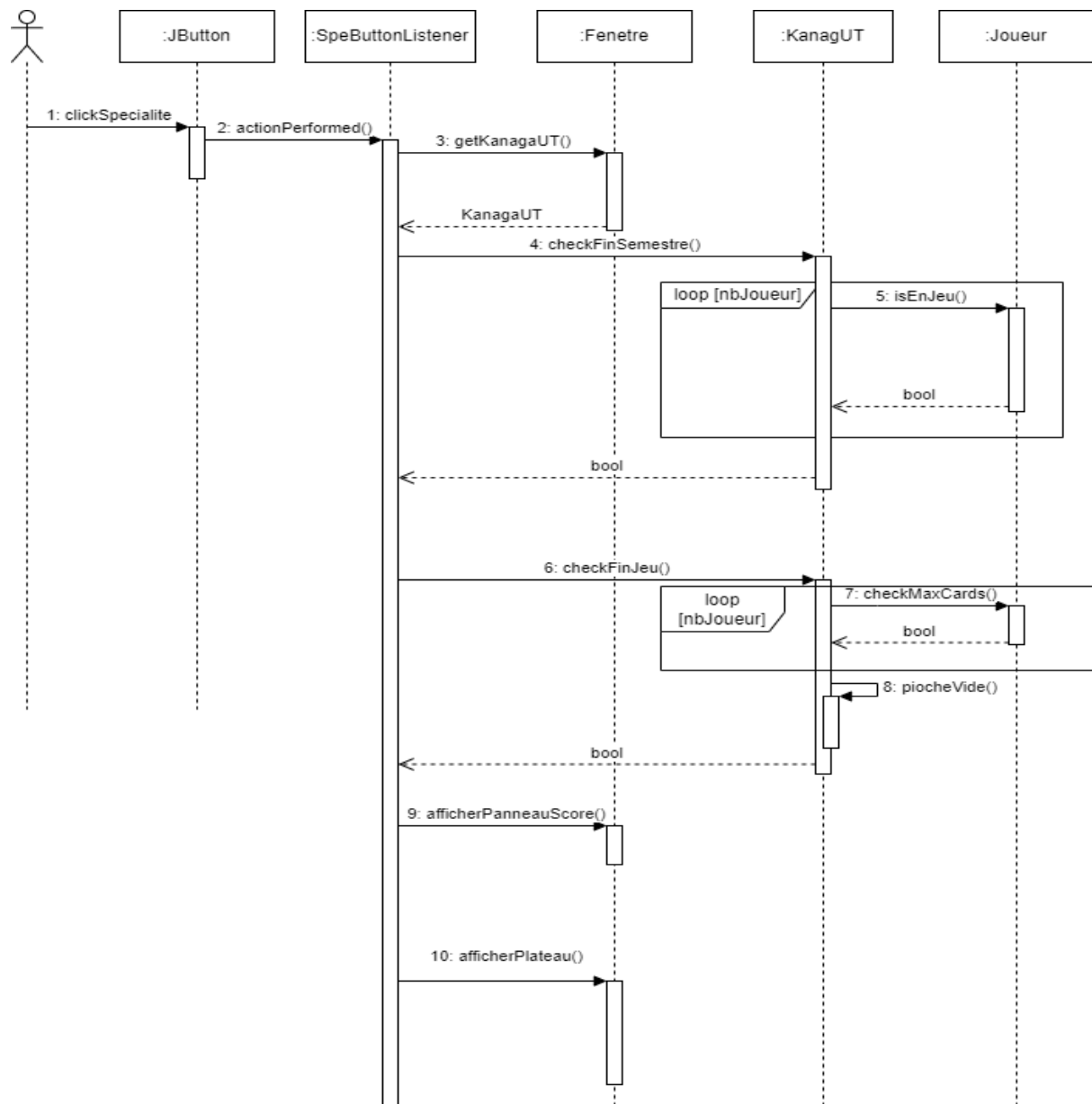


Figure 28: diagramme de séquence d’affichage du score .

## 5. Diagramme de classe global :

Le diagramme de classe global décrit l’ensemble des entités persistantes de l’application. Il contient la classe principale KanaGUT qui englobe toute les méthode dont un joueur a besoin pour valider son parcours dans le jeu ce dernier est représenter par une classe joueur liée à des autres classes (Spécialité, Parcours, Acqui, Score, Spécialisation, Carte des Compétences , Carte des UVs...) qui permet le bon déroulement du jeu.

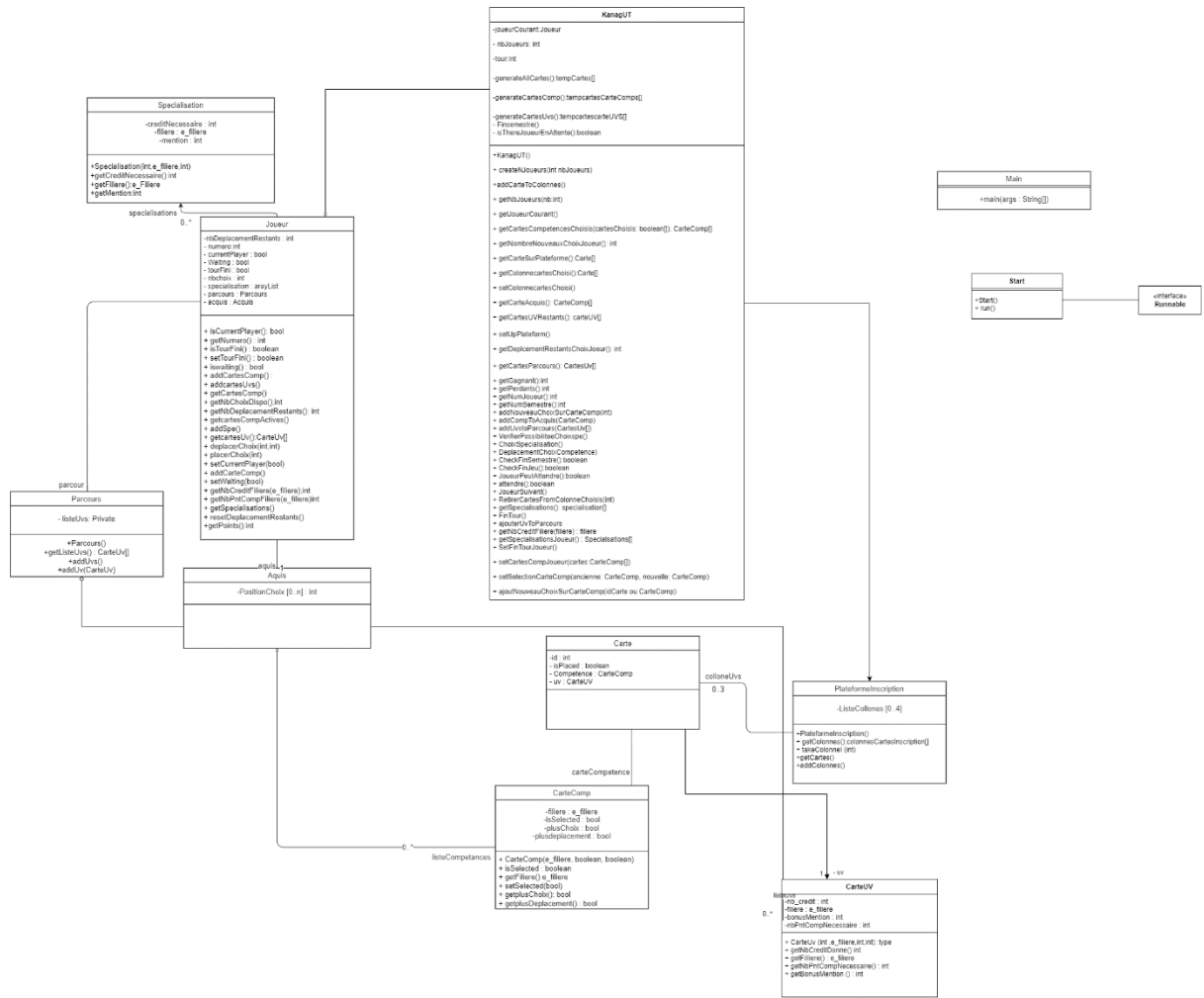


Figure 29 : diagramme de classe du système (partie 1).





# Réalisation et mise en œuvre

# Réalisation et mise en œuvre



## Introduction :

Cette partie est consacrée à la phase de réalisation de l'application. Ainsi, elle contiendra une présentation de l'environnement technique du projet et les différentes étapes de mise en œuvre de ce dernier. Enfin, quelques captures sont exposées pour achever cet élément.

## I. Environnement de développement :

### 1. Outils logiciels :

Pour la réalisation, nous avons eu recours aux outils suivants :

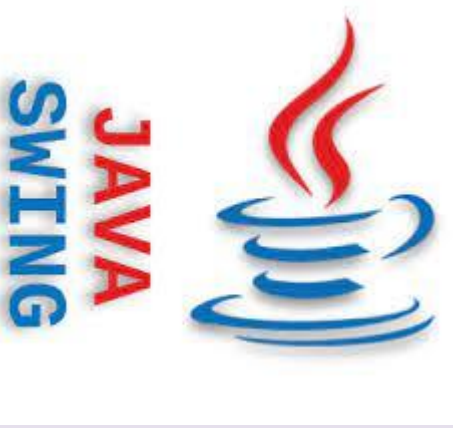
Outil	Présentation
	Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.
	<b>diagrams.net</b> (anciennement draw.io) est un logiciel de diagramme en ligne gratuit. Vous pouvez l'utiliser comme créateur d'organigrammes, logiciel de diagramme de réseau, pour créer UML en ligne, comme outil de diagramme ER, pour concevoir un schéma de base de données, pour créer BPMN en ligne, en tant que créateur de schéma de circuit, etc. draw.io peut importer des fichiers .vsdx, Gliffy™ et Lucidchart™.





Git est un système de gestion de versionnement décentralisé. Il est notamment utilisé pour le noyau Linux ou pour PHP.

Git permet notamment de "commiter" localement puis de pousser aux autres développeurs un ensemble de commit locaux. Il permet également d'utiliser un workflow de développement en soumettant par exemple l'envoi de code à l'approbation d'un des développeurs. La faculté de Git à créer des branches facilement ainsi que de permettre leur administration de façon simple en fait un outil de choix dans le cadre de développement de projets open source.

## 2. Technologies de développement :

Technologies	Présentation
	<p>Swing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes, inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures.</p>

### 3. Langages :

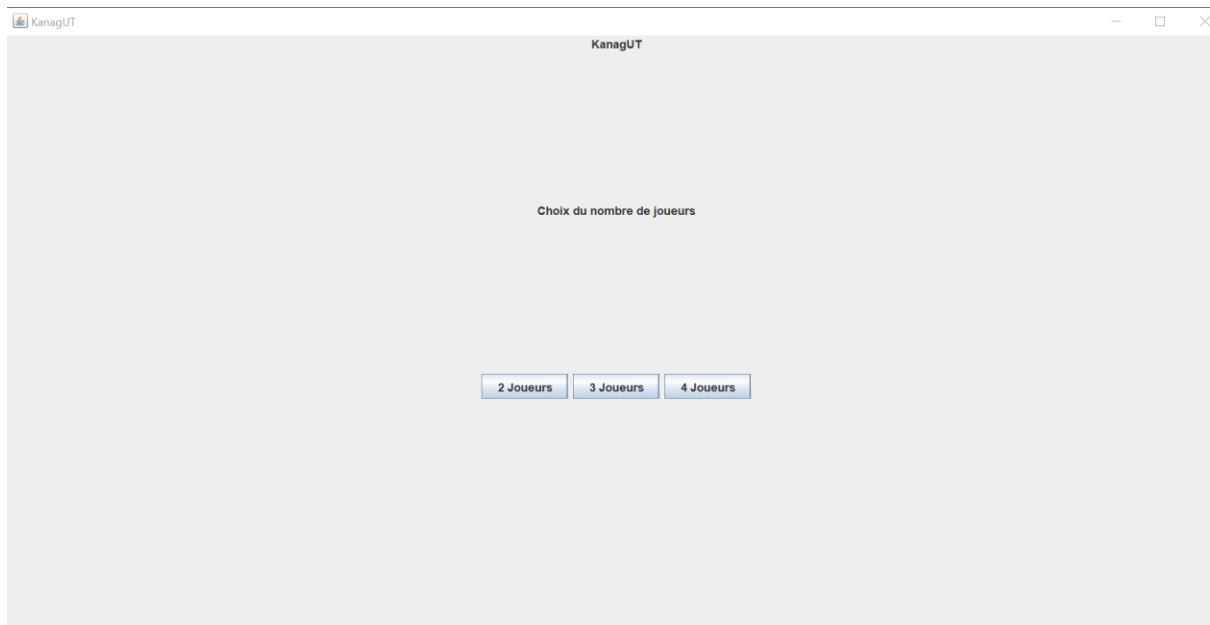
Langages	Présent
	Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.
	Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton. Une particularité de ce langage est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation.

## II. Présentation de l'application

Nous exposerons quelques interfaces de notre application, en essayant à chaque fois de décrire les différents objets interactifs mis à la disposition de l'utilisateur.

### Interface KANAGUT:

Le Jeu commence par une interface du choix de joueur; les acteurs nécessaires pour le bon déroulement du jeu.



c

Après avoir choisi le nombre des Joueurs, chacun d'eux à son tour est amené sur le plateau du jeu KanaGUT à fin de choisir des cartes UV et compétences pour l'obtention des crédits nécessaires pour accéder à une spécialité.

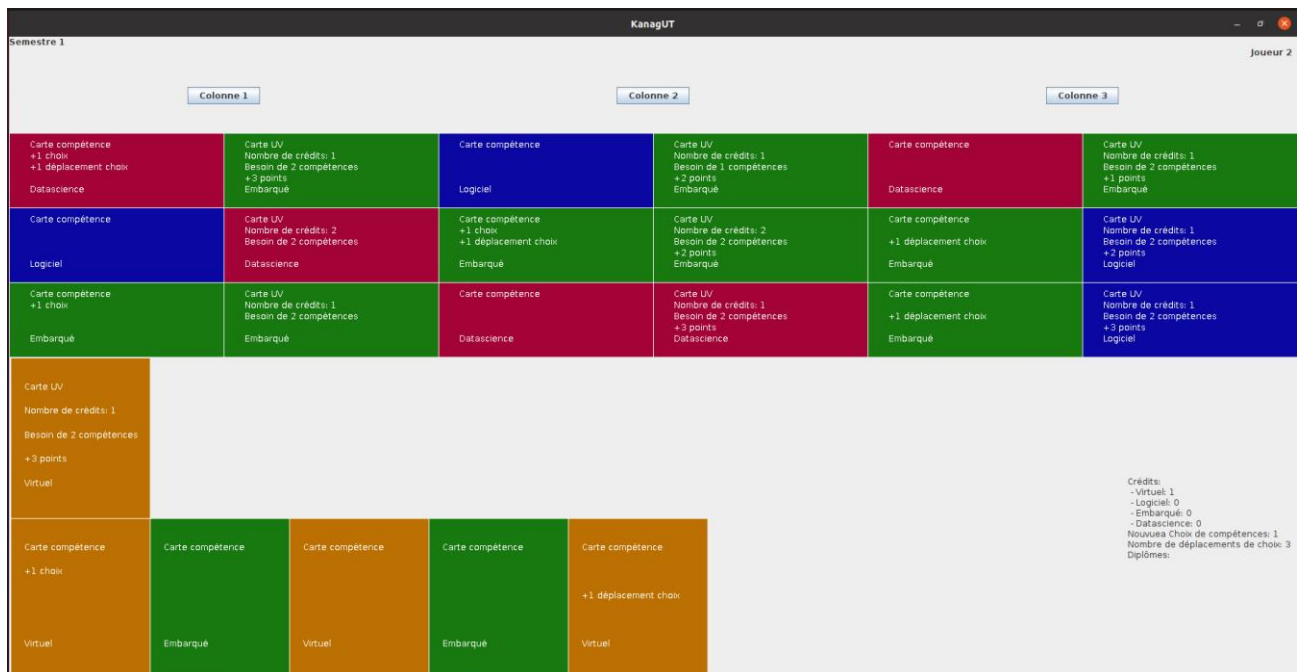


Figure 32 :Interface Choix Carte UVs et Compétences.

En choisissant une colonne des Cartes le joueur pourra déplacer ses choix à son parcours.

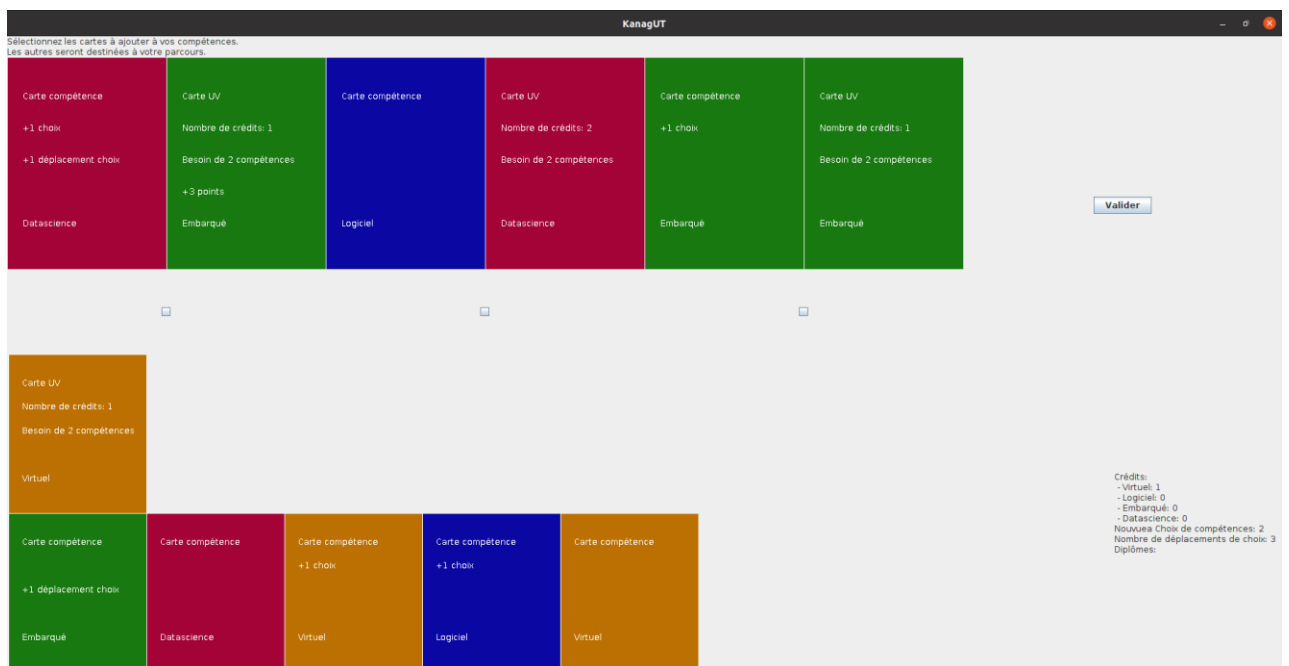


Figure 33 :Interface Nouveau choix de compétences.

Comme il peut choisir des nouveaux compétences.

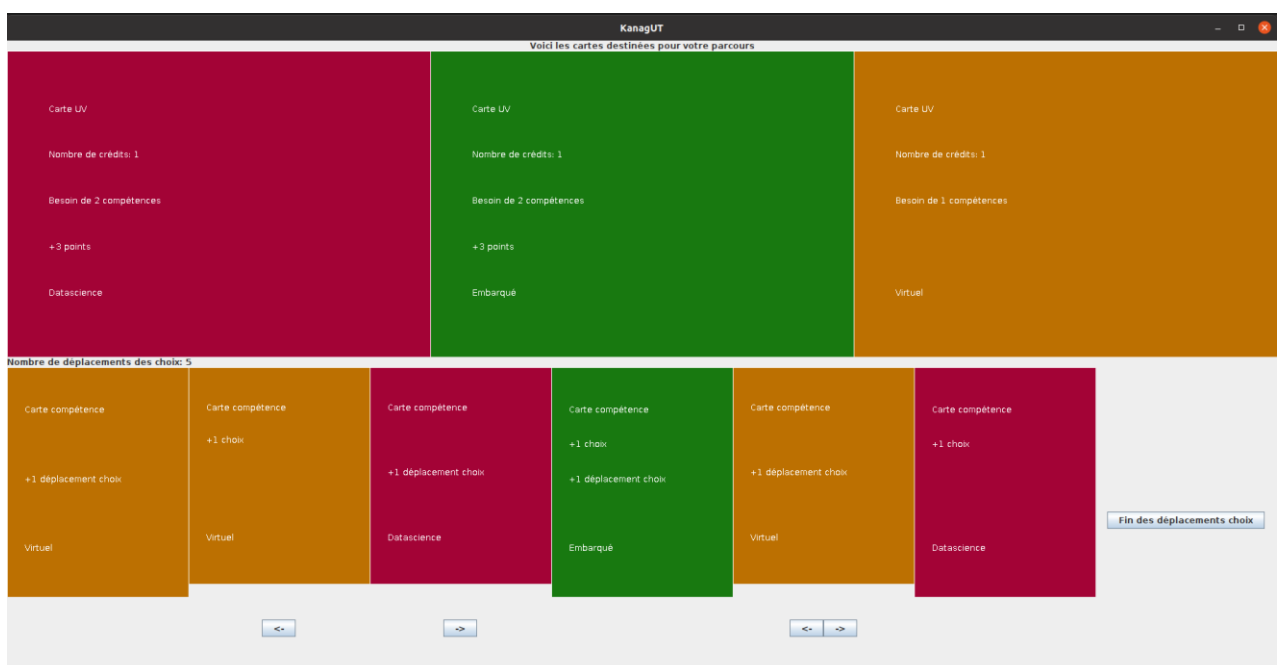


Figure 34 :Interface déplacement choix de compétences.

Et les ajouter par la suite à son parcours suivie .

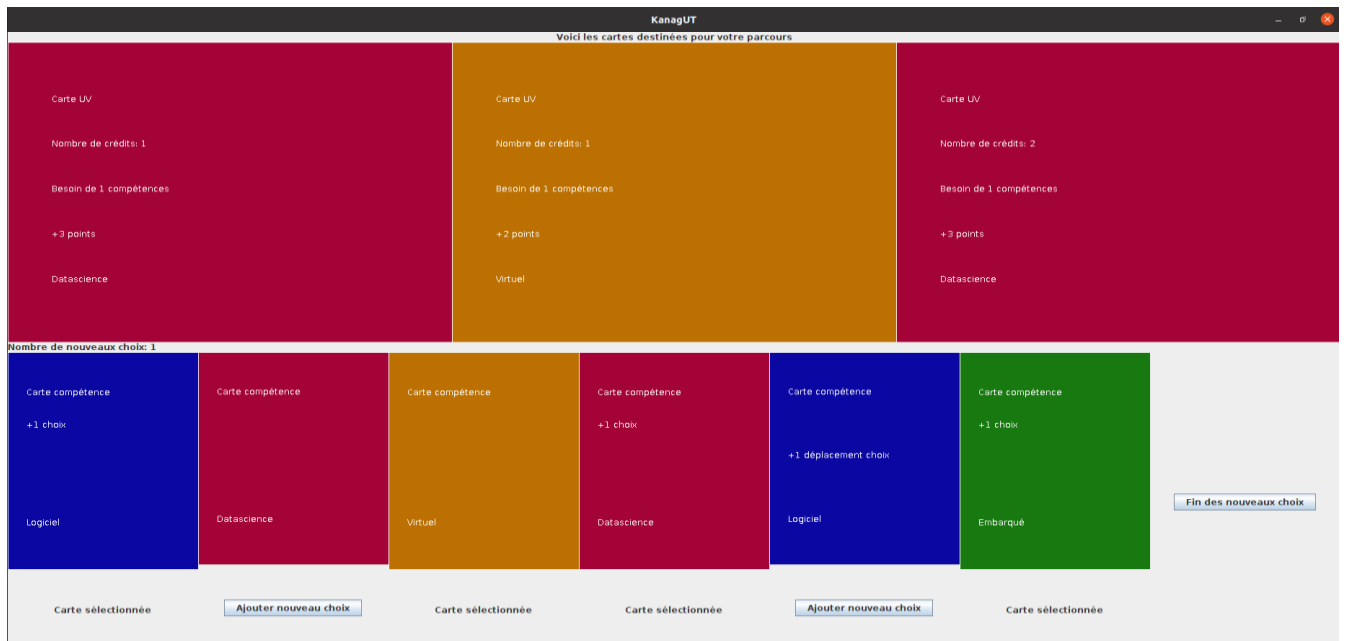


Figure 35 :Interface d'ajout des nouveaux Choix des Compétences.

Avec l'obtention des crédits nécessaires le joueur aura donc la possibilité de choisir une des spécialités proposées ( le choix de la spécialité se fait en se basant sur ses UVs et ses compétences).

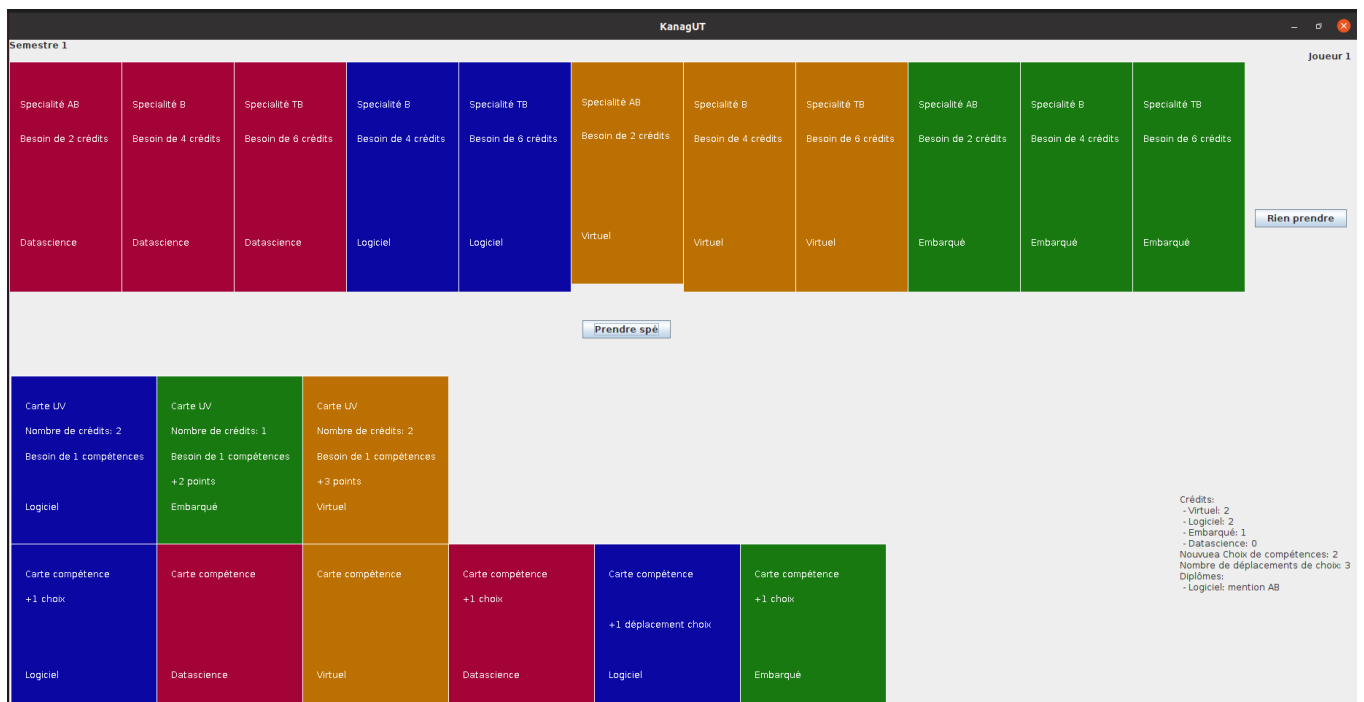
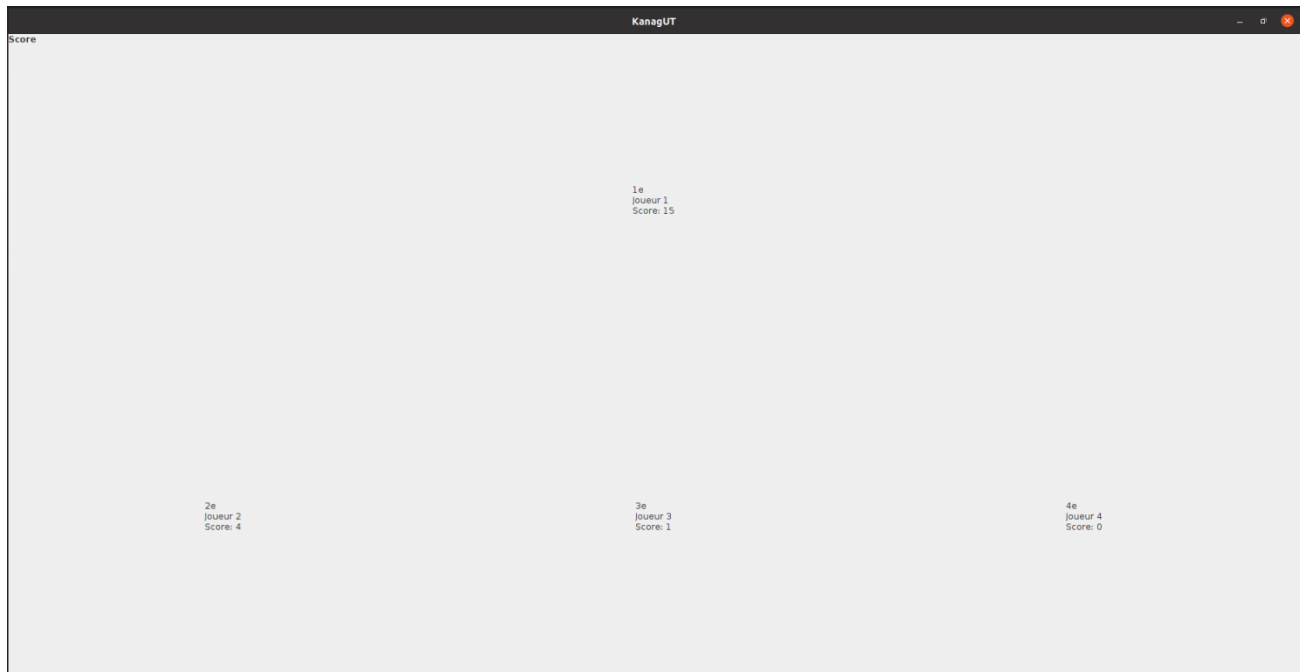


Figure 36 :Interface de Choix des spécialité.



En accomplissant toutes les étapes et à une condition précis on arrive à la fin du jeu. Cette dernière est représenté par une interface qui génère le Score et la position de chacun des joueurs dans le jeu.



*Figure 36 :Interface affichage du score.*

# Bilan

# BILAN

## I. Évaluation générale :

Une fois la réalisation faite, nous avons été satisfaits par le résultat obtenu. Nous avons réussi à créer une application agréable dans son utilisation. D'autant plus, vu que nous avons travaillé avec de nouvelles technologies nous avons en effet acquis de nouvelles connaissances.

## II. Problèmes rencontrés et résolution :

### 1. Problèmes rencontrés :

Pendant la réalisation de ce projet, nous avons rencontré quelques problèmes, parmi lesquels nous citons :

- **Travail avec des nouvelles technologies :**

Nous avons choisi de travailler avec JAVA SWING vu que d'une part nous avons eu l'occasion d'utiliser JAVA SWING dans des projets, et d'autre part nous voulions apprendre à utiliser cette bibliothèque JAVA après avoir étudié JAVA AWT dans un module et découvrir la différence entre eux.

### 2. Résolution des problèmes :

Concernant le problème du travail avec de nouvelles technologies, nous avons pu franchir cet obstacle en effectuant des recherches, et en étudiant mais surtout grâce aux cours disponibles sur internet.

## III. Compétences acquises :

En un premier temps nous avons acquis de nouvelles connaissances en développant avec de nouvelles technologies. En un deuxième temps, nous avons appris à être plus autonome, et à s'autoformer pour non seulement réaliser l'application mais aussi être à la hauteur afin de satisfaire les besoins de l'entreprise.

## Conclusion

Ce projet nous a beaucoup apporté d'un point de vue technique aussi bien que personnel. Il nous a permis de mettre en avant nos compétences en gestion, en analyse et conception ainsi qu'en développement. Un tel projet est considéré comme un jeu de société pour les étudiants de l'UTBM.

Pour mener à bien ce projet, nous avons opté pour utiliser le concept de génie logiciel. Ce concept, désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation. L'architecture informatique mise en place permet d'exploiter l'environnement de connaissances et de mettre à disposition des étudiants un jeu représentant leurs parcours au sein de l'UTBM en ajoutant un grand nombre de fonctionnalités secondaires.

Évidemment, il était nécessaire de mettre en place un logiciel de qualité. Pour cela, nous avons choisi d'adopter un cycle de vie en Y non seulement à cause de son pilotage de risque mais aussi sa nature itérative et incrémentale ainsi que la simplicité de son intégration avec la méthode agile « Scrum ». En effet, le processus Scrum étant Itératif et incrémental s'adapte parfaitement à la décomposition de notre projet.

Ce travail a été réalisé durant une période de deux mois, la première semaine a consisté en l'étude des besoins. Elle a permis de proposer différents scénarios d'utilisation et de fonctionnalités de projet demandés. La semaine suivante a été consacrée à la modélisation de l'application et à la constitution des enjeux. Les six dernières semaines nous ont permis de concevoir et d'implémenter l'application, Le déroulement n'a pas été aussi linéaire, car la démarche a nécessité beaucoup d'aller-retour entre les différents modules.

D'un point de vue personnel, ce projet nous a permis d'évoluer nos compétences de travail en équipe.

# Webographie

- <https://www.jmdoudoux.fr/java/dej/chap-swing.htm>
- <https://www.w3schools.com>
- <https://www.w3schools.com>
- <https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-FR.pdf>
- <https://app.diagrams.net/?src=about#Hlilith-m%2FAPP-QCM%2Fmain%2FUntitled%20Diagram.drawio>
- <https://app.diagrams.net>