PROJET DE PROGRAMMATION

de

FLAMENT Theo, MANCELLON Adam

MARTORANA Aurore et LAVALLEE Jérémy

Groupe AI

https://gitlab.com/borninachenil/projet-prog.git

L2 Informatique

Faculté des Sciences

Université de Montpellier

16 Avril 2022

SOMMAIRES

I. INTRODUCTION

Au cours du mois de juillet 2021, selon les statistiques de Médiamétrie pour le Web, ce fut près de 9,4 millions de visiteurs, soit environ 18 pourcents des Français qui se sont rendus sur un site de jeux de société en ligne. Avec un temps de consultation en moyenne de 1 heure 27 minutes par personne pour le mois, ces chiffres s'expliquent par l'actualité avec la présence de nombreuses pubs sur le développement des jeux de société en ligne.

Ce projet consiste a développer un jeu de « Qui-est-ce ? » générique. Tout d'abord, nous devons développer le jeu de base puis réaliser un générateur selon les événements a gérer. Une fois cette base faite, nous devons faire évoluer notre jeu pour permettre la gestion du type d'affrontement, améliorer l'interface, ajouter des fonctionnalités ...

Pour effectuer ce projet, nous avons utilisé des outils collaboratifs ou de développement pour la répartition des tâches. Nous avons également fait un choix de langage, organisée une architecture permettant à l'utilisateur de jouer, élaborées des fonctionnalités pour aider à la saisie des personnages, réalisées des particularités techniques menant à des extensions de notre jeu, puis nous avons rencontré des difficultés techniques et organisationnelles.

L'objectif de ce projet est donc de créer par nos propres moyens une application proposant un jeu de société générique ayant plusieurs fonctionnalités tels que jouer contre un ordinateur, un autre joueur, ou avec des personnages cachés ...

La première partie abordera les technologies utilisées et l'organisation du groupe, la seconde partie présentera les fonctionnalités permettant à l'utilisateur de jouer, la troisième partie soulignera le générateur aidant à la saisie des personnages, la quatrième partie démontrera les fonctionnalités avancées puis la dernière partie expliquera les difficultés rencontrées lors du projet et les bénéfices apportés.

II. TECHNOLOGIES UTILISÉES ET ORGANISATION

Pour commencer, nous allons voir la gestion de notre groupe pour répartir les différentes tâches et les outils employés. Tout d'abord, il est important que le projet ne se scinde pas en sous-projet indépendant au sein de notre groupe. Pour cela, nous avons pour chaque étape, réparties des tâches à effectuer en groupe ou tout seul. En effet, pour attribuer toutes ces tâches nous avons tout au long de notre projet utilisé Discord, qui est un logiciel de messagerie instantanée. Après, plusieurs heures de discussions et d'organisation, nous avons choisi plusieurs sites permettant la distribution du travail pour nous faciliter la tâche, le langage que nous allons utiliser et un rythme de travail commun à tous, étant pendant les heures de cours prédéfinis et quelques heures dans les week-ends.

Pour continuer, nous avons choisi un langage à notre application qui est JAVA. Ce dernier est un langage de programmation orienté objet très répandue et à la base de la plupart des applications en réseau. Pour pouvoir développer en JAVA, nous avons utilisé « Eclipse » qui est un environnement de développement intégré libre, extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre un langage de programmation tels que JAVA. Pour notre projet, nous avons utilisé quelques Framework qui est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un squelette de programme. Et nous avons aussi utilisé quelques bibliothèques qui sont des collections de routines, qui peuvent être déjà compilées et prêtes à être utilisées par des programmes. Les bibliothèques sont enregistrées dans des fichiers semblables, voire identiques aux fichiers de programmes, sous la forme d'une collection de fichier de code objet rassemblés, accompagnée d'un index permettant de retrouver facilement chaque routine. Ce dernier est un ensemble d'instruction qui prend en charge une certaine opération et produit un résultat.

Ainsi, nous avons utilisé JavaFX qui est un Framework et une bibliothèque d'interface utilisateur, qui nous permet de créer une interface graphique pour des applications. Dans notre code nous avons utilisé une architecture spécifique de

JavaFX. « L'application » est la classe principale de notre programme JavaFX. Chaque programme JavaFX doit étendre la classe Application. Sa méthode de démarrage est le point d'entrée principal de l'application, c'est la première méthode à être appelée une fois que le système est prêt. Une application JavaFX se compose de deux conteneurs : « Stage » et « Scene ». Le premier indiqué est le conteneur de niveau supérieur, la fenêtre principale de l'application. « Scene » est le conteneur de contenu visuel, qui est organisé dans un graphique de scène. Ce dernier est un arbre hiérarchique de nœuds qui représente tous les éléments visuels de l'interface utilisateur de l'application. Les implémentations concrètes de nœuds incluent des primitives graphiques, des contrôles, des gestionnaires de mise en page, des images ou des médias. Il est possible de manipuler la scène en modifiant les propriétés du nœuds, donc nous avons animés les nœuds pour appliquer des effets, faire des transformations et modifier leur opacité.

De plus, nous avons utilisé JSON qui est un format de fichier standard ouvert et un format d'échange de données qui utilise du texte lisible par l'homme pour stocker et transmettre des objets de données composés de paires attribut-valeur et de tableaux. Il s'agit d'un format de données commun avec diverses utilisations dans l'échange électronique de données, y compris celui des applications. Pour lier JAVA et JSON, nous avons utilisé GSON qui est une bibliothèque Java qui peut être utilisée pour convertir des objets Java en leur représentation JSON. Il peut également être utilisé pour convertir une chaîne JSON en un objet Java équivalent. GSON peut travailler avec des objets Java arbitraires, y compris des objets préexistants dont vous n'avez pas de code source. Nous avons utilisé cette bibliothèque car elle nous a permis de fournir des mécanismes faciles à utiliser tels que « toString() » pour convertir Java en JSON, autoriser les représentations personnalisées pour les objets ou encore générer une sortie JSON compacte et lisible.

Mais encore, nous avons utilisé quelques librairies standard de Java. Tout d'abord, nous avons utilisé « Java.lang », qui fournit des classes qui sont fondamentales pour la conception du langage Java. Les classes les plus importantes sont « Object », qui est la classe mère de toutes les classes, dont les instances représentent les classes lors de l'exécution. Souvent, il est nécessaire de représenter une valeur de type primitive sous forme d'objet. Ainsi, il existe des classes 'enveloppe' qui vont représenter un type primitif qui sont soit des Boolean, des Character, des Integer, Long, Float et Double. Il existe aussi la classe Math qui fournit des fonctions

mathématiques. La classe String qui permet de faire des opérations usuelles sur les chaînes de caractères. La classe Runtime, Process, System ... qui fournissent des méthodes pour réaliser des opérations systèmes qui gèrent le chargement dynamique des classes ou autre. Et la classe Throwable qui englobe tous les objets qui représentent les erreurs et exceptions. Ensuite, nous avons utilisé « Java.io » qui est conçu selon deux concepts très simples : un double découpage lecture/écriture et texte/binaire d'une part, et le design pattern Decorator d'autre part. Le premier indique la fonction des classes, et le second la façon dont elles peuvent être assemblées en une chaîne de lecture ou d'écriture. De plus, nous avons utilisé « Java.util » qui contient des classes utilitaires, en particulier celles qui servent à manipuler les classes dates, collections, listes, Maps, vecteurs, Hastable, Random ... Ces classes ne sont pas associées à d'autres classes par généralisation spécialisation et donc qu'elles sont directement liées à la classe « Object » du paquetage « Java.lang ». De plus, ces classes respectent les interfaces comme celle de JavaFX. Elle est également associé au paquetage « Java.io » car elle permet de pouvoir sauvegarder et restaurer l'état d'un objet.

Pour continuer cette partie, nous allons voir les multiples sites que nous avons choisis pour notre collaboration tout au long de notre projet.

D'une part, notre premier outil de collaboration est Trello, qui est un site de gestion de projet en ligne. Ce site repose sur une organisation des projets en forme de planche listant des cartes, chacune représentant des tâches de travail. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre. Nous avons commencé notre expérience sur Trello en créant un projet. Sur la page de projet du Trello, nous avons créé trois planches. La première sous le nom de « À faire » qui est composées de toutes les tâches à faire permettant l'avancement de l'application. La seconde planche porte comme nom « En cours » et est composées de toutes les tâches que nous sommes en train de faire permettant l'amélioration de notre application. Pour finir, il reste une dernière planche ayant comme nom « Fait » représentant toutes les étapes du projet qui sont effectuées avec succès. Par exemple, nous avons de multiples cartes comme l'interface, les boutons ... Chaque étape effectuée a une assignation qui permet de savoir la personne qui a effectué cette tâche et sur quelle page. Cette fonctionnalité est pratique dans le cas où l'un de nous travaille également sur cette tâche ou sur la même page pouvant engendrer des conflits de code. Trello nous a beaucoup aidés pour la répartition des tâches

dans le groupe et pour l'organisation de notre travail.

D'autre part, notre second outil de collaboration est GitLab. Ce site est un logiciel libre de forge basée sur GIT. Ce dernier est un logiciel de gestion de version décentralisé. GitLab propose de nombreuses fonctionnalités comme le Wiki, qui est une application web permettant la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web ou d'une application. GitLab utilise aussi la fonctionnalité d'un système de suivi des bugs, c'est un logiciel permettant d'effectuer un suivi des bugs signalés dans le cadre d'un projet de développement web. Ces deux fonctionnalités nous ont énormément aidés pour améliorer la qualité de notre site. De plus, GitLab permet de piloter des dépôts de code source et permet de vérifier à chaque modification de notre code source que le résultat des modifications ne produit pas de régression dans notre application qu'on développe en parallèle. Et c'est cette dernière fonctionnalité qui nous a permis une grande organisation de travail, une grande avancée, mais aussi de grande difficulté. En effet, lorsque nous avons lancé un projet et que nous avons incorporé des fichiers contenant tout le code source de notre application, chaque personne de notre groupe peut voir tout le code source. Si l'un de nous change un bout du code, il doit utiliser une commande « git commit » qui capture un instantané les changements, c'est-à-dire qu'il soit intégrer les nouveaux morceaux de code sur le GitLab. Dès que cela est fait, les autres membres du groupe reçoivent une notification disant les modifications faites et sur quel fichier. Cette particularité nous a vraiment aidée dans notre travail, car recevoir une notification lorsque l'un de nous améliore notre site, permet d'avancer en groupe, de récupérer ce qu'il a changé pour que chacun de nous puisse aussi avancer et surtout de collaborer.

Par la suite, nous allons voir les fonctionnalités que nous avons utilisé pour permettre aux utilisateurs de jouer à notre jeu.

III. FONCTIONNALITÉS PERMETTANT À L'UTILISATEUR DE JOUER

Pour commencer, nous allons voir l'application et l'architecture de base de notre site. Tout d'abord, notre application de base doit permettre de nombreuses interactions possibles pour les utilisateurs. Pour mener une bonne description, nous allons expliquer l'architecture phase par phase dans l'ordre que nous l'avons faite.

Ainsi, lorsqu'on rentre dans notre application, on apparaît directement dans le menu principal qui est la page centrale de notre projet. Cette dernière est composée d'un « Menu principal » en haut, puis de deux boutons présentant deux options différentes. Sur le premier bouton est inscrit « Nouvelle partie/ Charger partie », en effet en appuyant sur ce bouton, l'utilisateur arrivera sur une page ou il pourra soit charger une partie soit en créer une nouvelle. S'il choisit de créer une partit alors une petite fenêtre s'ouvre. Cette dernière permet de choisir le mode de difficulté, c'est-à-dire qu'il y a deux boutons avec comme signification « Facile » ou « Normal ». S'il choisit de charger une partie déjà commencer, cela est possible grâce au fichier JSON de sauvegarde. Le deuxième bouton présent sur le menu principal est « L'extension de notre jeu ». Pour continuer, une fois qu'une partie est en cours l'utilisateur arrive sur la page de jeu composé de plusieurs boutons et de l'interface graphique présentant les nombreux personnages du jeu. En effet, en bas de la page il y a la présence de cinq menus déroulants. Ces derniers permettent à l'utilisateur de formuler des questions. Deux de ces menues déroulants sont dédiés au choix du « type d'attributs », par exemple la couleur des cheveux, des yeux, le genre du personnage, les accessoires visibles mais aussi l'expression du visage. Avec cela, deux autres menus déroulants permettent de compléter la question car ils permettent de faire un choix parmi les attributs possibles soit liés avec ceux demandés juste avant. Donc si l'utilisateur demande la couleur des cheveux alors dans les deuxièmes menus déroulants, il pourra compléter avec « Noirs, Bouclés, Chauves, Blancs, Blonds, Lisses, Roux, Châtains, Brins ». Si l'utilisateur demande la couleur des yeux alors il pourra compléter avec « Marrons, Gris, Bleu,

Noirs ». Mais encore, s'il demande le genre de personnage alors il pourra compléter avec « Homme, Femme », pour les expressions du visage « Négative, positive » et pour finir, s'il demande les accessoires visibles, il pourra compléter avec « Boucle d'oreille, rouge à lèvre, lunettes, chapeau, barbe, moustache, couette, nœuds, béret, casquette ». Pour finir la demande d'une question, il y a le dernier menu déroulant qui permet de sélectionner un connecteur logique donc une plus grande diversité de question qui est possible. Ces connecteurs logiques sont « Et, Ou, non ». Une fois sa question formulée, l'utilisateur valide sa question grâce au bouton dédié et reçois par la suite sa réponse. Le jeu choisis un personnage aux hasards parmi tous ceux présents devant l'utilisateur. Lorsque l'utilisateur va valider sa question, notre jeu va barrer les personnages. Par exemple, si l'utilisateur demande si la personne a les cheveux « Blond ET bouclés », alors si le personnage choisis de notre jeu n'a pas ses attributs alors il va barrer toutes les personnes ayant les cheveux blonds et bouclés. Le jeu se termine uniquement lorsque l'utilisateur trouve le bon personnage avec le moins de question possible. Et pour finir, l'utilisateur peut aussi quitter ou sauvegarder une partie en appuyant sur le bouton dédié à cette fonctionnalité.

Pour poursuivre, nous allons voir le format du fichier JSON. En effet, il y a deux types de fichier JSON nécessaire au fonctionnement du jeu. Il y a celui qui permet la sauvegarde d'une partie et celui qui contient toutes les informations des personnages. Tout d'abord, les fichiers de sauvegarde contiennent le chemin du ficher JSON des personnages de la partie et toutes les informations de la partie comme le nombre de lignes (« nbL ») et de colonnes (« nbC »), le personnage que l'utilisateur doit réussir à deviner, la liste des personnages déjà cochés(« coches » et le nombre de tours(« nbT »). Par exemple, pour notre application nous avons :

```
{
  "nbL" : 4,
  "nbC" : 6,
  "nbT" : 0,
  "entM" : 0,
  "jsonP" : "c://Users/theof/eclipse-workspace/testInterface/src/main/resources/Entite.json",
  "coches" : [1,7]
}
```

Ensuite, il y a les fichiers de personnages qui contiennent les informations nécessaires pour tous les personnages. Pour chaque personnage, nous pouvons trouver un identifiant par un numéro unique, son nom, le chemin de l'image («

path ») puis tous les attributs physiques tels que les cheveux yeux . . .. Par exemple, pour un personnage de notre application nous avons :

```
"5" : {
  "nom" : ["Luc"],
  "path" : ["file::src/main/resources/img/personnages/imageonline-co-split-image-5.png"],
  "cheveux" : ["noirs", "bouclés"],
  "yeux" : ["marrons"],
  "genre" : ["homme"],
  "accessoires" : ["barbe"],
  "expression du visage" : ["positive"]
},
```

Maintenant, nous allons voir les différentes formes de requêtes que nous avons utilisés. Tout d'abord, comme préciser précédemment lorsqu'un joueur lance le jeu il fait quelques manipulations pour arriver jusqu'au moment où il va commencer la partie. Pour réussir jusqu'à là, nous avons fait une requête sous la forme d'une fenêtre de choix de fichier permettant donc de commencer une partie avec le mode de difficulté que l'utilisateur veut. Mais également, nous avons fait des requêtes pour toutes les questions présentes dans les menus déroulant. Il y a également des requêtes présentent pour tous les boutons liés à l'interface soit les onclick permettant de cocher des entités, pour pouvoir quitter ou sauvegarder une partie.

Pour finir, nous allons voir les structures de données, des classes et des variables que nous avons utilisé. En effet, pour commencer nous pouvons parler de la classe Entité. Pour cette dernière, nous avons choisi d'utiliser un format spécial qui est : HashMap<String, ArrayList<string» permettant de stocker leurs attributs. Nous avons choisi ce type de format car certains attributs peuvent avoir plusieurs valeurs comme : un ''accessoires" peut être un ''chapeau" mais aussi une paire de ''lunette". Le chemin de leur image et leur nom sont stockés dans ce HashMap et ils ont un attribut booléen nommé « barre » qui est vrai lorsque le personnage est coché et rend faux dans le cas contraire.

Ensuite, nous avons la classe Partie qui a pour attributs toutes les données de la partie comme le nom de la sauvegarde qui sera noté dans le fichier JSON pour garder toutes les informations de la partie, le nombre de lignes et de colonnes, l'instance d'Entité qui représente le personnage mystère soit celui que le joueur va devoir deviner. Il y a aussi la présence d'un bouton permettant au joueur de quitter ou sauvegarder la partie. Il y a un compteur de tour qui sert de score dans le jeu de base et ce dernier s'arrête uniquement quand il ne reste que le

personnage mystère non coché, ou si l'utilisateur coche le personnage mystère. Il existe deux constructeurs dans la classe Partie, le premier permet de créer une nouvelle partie et l'autre permet de charger une partie sauvegardée. Concernant les méthodes, getAllEntite permet d'obtenir un tableau d'instance d'Entite à partir d'un chemin de fichier de personnages, des constructeurs paramétré différent, les méthodes get, la méthode coche permettant de changer le booléen d'une instance d'Entite et incrémente ou décrémente le compteur de personnages cochés, une méthode de sauvegarde ainsi que getAllKeys qui retourne un tableau avec tous les types d'attributs de personnages et getAllChoicesByKey qui retourne un arrayList de tous les attributs d'un type spécifié.

Et enfin, nous avons la classe PartieFacile qui est la classe fille de Partie. Donc cette classe contient en plus une ArrayList des entités éliminées automatiquement en attribut. En effet, ceci est la particularité du mode facile de notre jeu de base, où il y a la possibilité de connaître le nombre de personnage éliminés quand l'utilisateur tape une question avant qu'il valide, et les personnages éliminés sont barrés automatiquement par l'ordinateur après validation. Pour continuer, il y a aussi la présence de deux méthodes, étant listeElimine qui retourne une arrayList des personnages eliminés et eliminesParTour qui elimine les personnages automatiquement selon la question et le connecteur logique.

Pour la suite, nous allons voir les fonctionnalités que nous avons utilisé pour aider à la saisie des personnages.

IV. GÉNÉRATEUR POUR LA SAISIE DES PERSONNAGES

Pour commencer cette partie, nous allons voir toutes les fonctionnalités et aspects du générateur qui permet à la saisie des personnages de notre jeu de base. Tout d'abord, notre générateur est donc une classe de notre jeu de base contenant plusieurs bibliothèques. Ces dernières sont « import com.google.gson.* ; / import java.io.* ; / java.util.* ; ». La première bibliothèque est simple et basée sur Java pour sérialiser des objets Java en JSON et vice versa. Il s'agit d'une bibliothèque open source développé par Google. La seconde est une bibliothèque qui va nous fournir des classes pour manipuler des flux de données soit nécessaires à la création, lecture, écriture et traitement des flux. Puis la dernière est une bibliothèque contenant des classes de collection héritées, des modèles d'événements, des installations de date et d'heure, d'internationalisation et diverses classes d'utilité.

Pour continuer, la classe Générateur contient plusieurs attributs soit trois entiers étant les colonnes, les lignes et le nombre d'attribut à donner par entité. Ensuite nous avons une instance de la classe Entité et un attribut « String » qui est le chemin des images des personnages. Nous avons aussi un constructeur qui se sert d'une fonction pour récupérer tous les personnages du fichier JSON de base qui est constitué seulement d'une liste de numéro pour chaque entité, et le chemin de leur image pour chaque. Il y a la présence d'une méthode pour donner un nouvel attribut à toutes les entités par exemple pour ''accessoires" et la présence d'une autre méthode nommée « supprK » qui permet de supprimer toutes les entités. Il y a aussi une méthode qui permet de renvoyer un personnage à partir d'un chemin d'image, ce qui est pratique pour retrouver le personnage correspondant lorsqu'un utilisateur va appuyer sur une image pour la barrer. Cette méthode se sert du fait que chaque personnage a toujours le chemin de son image dans ses attributs, si aucun personnage n'est attribué à cette image alors ça envoie un nouveau personnage mais vide puis fait un message d'erreur.
De plus, il y a aussi la présence d'une méthode qui permet d'afficher toutes les entités en fonction du nombre de ligne et de colonnes choisis. Mais encore, il y a la présence de « saveJsonEntite » qui est une méthode de sauvegarde où lorsque l'utilisateur va appuyer sur le bouton de sauvegarde, cette méthode va rentrer en

vigueur. Donc toutes les mises à jour de la partie seront garder, pour cela on va supprimer le fichier correspondant à la liste des personnages qui est un attribut de la classe Générateur, quand nous avons fini de modifier tous les personnages, nous faisons une sauvegarde en créant un nouveau fichier qui créer le JSON correspondant. Donc si un utilisateur veut reprendre sa partie, il suffit juste que lors qu'on demande de reprendre une partie sauvegardée, il choisit se nouveau fichier créer. Donc la méthode « Save » créer le fichier des entités dans son dossier associé ainsi que son fichier d'exécution associé dans son dossier respectif. Et c'est l'utilisateur qui choisi le nom de sa sauvegarde car elle est propre à chaque utilisateur.
Pour finir, il y a également la présence de la même interface que le jeu de base mais en changeant juste la méthode « Onclick » des images et en remplaçant les combobox des questions par une TextArea pour pouvoir ajouter des clés d'attributs. Cette nouvelle méthode Onclick ouvre une fenêtre pour ajouter tous les attributs d'un personnage sélectionné.

Par la suite, nous allons voir les fonctionnalités avancées que nous avons éventuellement fait pour notre projet.

V. FONCTIONNALITÉS AVANCÉES

Pour continuer, nous allons voir les fonctionnalités avancées que nous avons choisis pour notre projet. Durant notre projet, nous avons eu énormément de problème lié à la crise sanitaire mais aussi lié à des problèmes d'organisations et d'apprentissage car nous sommes partit de zéro pour réaliser ce projet. Cependant, nous avons eu quand même de multiples idées plutôt varié. Nous allons donc vous les présenter par ordre chronologiques.

En premier lieu, lors du début du projet soit la première séance, nous avons comme idée le fait de faire un mode 1 contre 1, c'est-à-dire pouvoir faire un face à face contre un autre joueur en réseau. De base nous avons décidé de faire le mode de jeu étant jouable seulement dans un réseau local, cependant lorsque nous avons décidé du langage de programmation que nous allions utiliser pour le projet soit Java, nous avons préféré abandonner cette idée. En effet, notre manque de connaissance en programmation réseau avec Java combiné avec toutes les technologies indispensables que nous avons utilisé pour le jeu de base comme JavaFX, GSON ..., ne nous permettait pas de faire cette extension car cela nous semblait être une difficulté trop grande pour le temps mis à notre disposition.

En second lieu, nous avons eu comme idée le fait d'utiliser un langage naturel avec notamment l'idée d'utiliser une API de Google. Cette dernière est un programme logiciel qui permet à des développeurs d'accéder à des données et ressources informatiques. Pour transmettre clairement des requêtes et des réponses, les API obéissent à des règles et à des méthodes spécifiques. Cette extension était très intéressante mais après énormément de recherche et de compréhension nous avons vu que c'était très complexe comme système.

En troisième lieu, nous avons eu l'idée d'une extension étant un classement des scores avec des pseudos demandées lors de l'enregistrement du classement à la façon d'une borne arcade. Cette extension avait de nombreux avantages comme la facilité de la réalisation, ne nécessitait pas d'apprendre de nouvelles langues ou technologies et le concept est assez universel. Cependant, lorsque nous l'avions comparé avec les exemples demandés pour la réalisation du projet, notre idée nous semblait un peu simpliste et basique. Nous nous disions que cela aurait presque

l'air de faire partie du jeu de base plutôt qu'une extension. Malgré cela, nous avons gardé cette idée dans un coin au cas où nous n'aurions pas d'autres idées.

En dernier lieu, nous avons fini par arriver à l'idée d'une extension d'un mode de jeu contre l'ordinateur, le gagnant étant désigné par le nombre de tours afin d'éviter que le joueur ou l'ordinateur soit avantagé selon qui commence. Cette idée présentait certains avantages de la précédente, tels que le fait que nous n'avons pas besoin d'apprendre une nouvelle langue ou technologie. Même si elle était plus compliquée d'un classement elle semblait plus réalisable que les premières idées que nous avons trouvé, tout en étant moins simpliste et plus intéressante pour le joueur. Cette idée nous a convaincu par la richesse des fonctionnalités possibles, le joueur aurait pu par exemple choisir la difficulté de l'algorithme contre lequel il allait jouer ou même choisir lui-même le personnage mystère pour mettre au défi l'ordinateur et voir le nombre minimal de coup qu'il suffisait pour deviner à coup sur le personnage. Donc nous sommes partit sur cette extension. Cependant, par manque de temps dû à tous les problèmes énumérés avant, nous avons choisi de faire une extension où le joueur choisis donc une difficulté, puis choisis un personnage parmi tous ceux qui seront exposés devant lui. A partir de là, le jeu sera donc inversé, c'est-à-dire que c'est l'ordinateur qui va poser des questions et le joueur qui va répondre jusqu'au moment où l'ordinateur va trouver le bon résultat.

Pour finir ce mémoire, nous allons voir en dernier lieu une conclusion revenant sur les différentes difficultés que nous avons rencontrées et les bénéfices que nous pouvons en tirés de ce projet.

VI. BÉNÉFICE ET DIFFICULTÉ DU PROJET

Pour commencer, nous avons rencontré énormément de difficultés pendant notre projet. En effet, nous l'avons dit avant, lors de notre projet nous avons fait le choix d'utiliser plusieurs bibliothèques et Framework tels que JavaFX ou GSON ... Les difficultés techniques de notre projet étaient de comprendre toutes les différentes fonctionnalités des bibliothèques, la compréhension et réussir à coder. Tous les membres de notre groupe n'avaient jamais utilisé JavaFX comme interface graphique, ni l'utilisation du fichier Json de manière optimale. De plus, nous avons eu du mal à trouver un outil adapté et facile d'utilisation pour convertir Json en HashMap, finalement nous avons choisie GSON. Ce dernier était l'une de nos plus grandes difficultés car c'était une toute nouvelle bibliothèque qu'il fallait totalement maitriser. Un autre problème était les nombreux conflits sur git, qui nous ont pris énormément de temps pour les résoudre. Des personnes de notre groupe ont également eu des problèmes avec le setup du projet sur Eclipse car c'était également nouveau pour eux.
Pour finir, l'une des plus grandes difficultés était la répartition du travail. Malgré, plusieurs sites pour l'organisation de notre travail, nous avons dû apprendre de nombreuses bibliothèques, maîtriser le langage à la perfection ... donc chaque membre du groupe allaient à son allure pour apprendre, ce qui a durcit la répartition du travail. Certes au début de notre projet, nous avons eu de nombreux créneaux permettant de se voir. Cependant, nous sommes dans une année particulière avec la présence de la Covid-19. Donc, à cause de cette dernière, nous avons pris le choix de ne pas beaucoup se voir pour la sécurité de soi-même et de notre famille. En contrepartie, toutes les semaines, nous avons fait des visioconférences, mais également des appels pour communiquer et s'entraider.

Pour continuer, nous allons voir les différents bénéfices que tout ce projet nous a apportés dû à tout ce qu'on réussit à faire fonctionner. D'une part, nous avons fait ce projet en groupe. Donc ce qui a le mieux fonctionné est le fait que dans le groupe, chaque personne à travailler, participer, aider et encourager les autres. Mais aussi le fait d'avoir réussi à finir ce projet dans le délai imposé et d'avoir complété tous les objectifs fixés. De plus, nous avons réussi malgré la crise

sanitaire à nous coordonner dans nos travails et de maximiser nos ressources pour finir ce projet. De plus, nous avons eu énormément de problème sur git, car comme nous avons utilisé de nombreuses bibliothèques et Framework que de base nous ne connaissons pas énormément, donc cette particularité nous a permis de résoudre et de comprendre de nombreux problèmes liés à git, ce qui nous a permis d'acquérir énormément d'expérience dans ce domaine.
Pour finir, ce qui a le mieux fonctionné dans ce projet est sans hésitation le fait d'avoir acquis un grand savoir. Effectivement, nous avons fait le choix d'utiliser Java comme langage, de multiples bibliothèques et Framework. Donc, après un grand travail de documentation, nous avons réussi à gérer et comprendre toutes les fonctionnalités de GSON, JavaFX, Éclipse ..., ce qui nous a permis d'apprendre et de gagner de l'expérience. Mais aussi de s'améliorer dans le développement d'application en reliant toutes ses fonctionnalités entre-elles pour donner notre projet finalement.

cm!

Pour conclure, à travers tout le mémoire, nous savons comment nous nous sommes organisé et réparti le travail avec tous les outils de collaboration comme Trello, malgré la crise sanitaire toujours présente. Nous avons également vu une description détaillée des fonctionnalités permettant à toute personne de jouer à notre jeu ainsi que l'aide à la saisie des personnes. Mais encore, nous avons vue toutes les différentes particularités techniques menant à des extensions de notre projet, puis nos difficultés durant tout le projet et les bénéfices que nous avons pu en conclure. Pour finir, nous pouvons dire que c'est une expérience extraordinaire à faire durant notre année de Licence Informatique, surtout en groupe, car ça nous a permis d'acquérir énormément d'expérience.