Groupe 1A – 1.2.A

Mathieu MERILLON
Valentin SAVREUX
Léo ROUILLARD
Benjamin SAINT-MARS
Lucas MIRLOUP
Ladislas WALCAK
Nolan MARTINS
Baptiste GARRIDO

M1207
Projet *Duel sur la toile*
Création d'une application

DOSSIER DE PRÉSENTATION

Tuteur : **M^me^ Adobet**
Année : **2017 - 2018**

iUT'O
IUT ORLÉANS
INFORMATIQUE

SOMMAIRE

I - Introduction

Dans le cadre de notre formation à l'IUT Informatique d'Orléans, nous avons l'occasion de mettre en application nos connaissances dans la plupart des modules abordés durant l'année, notamment ceux en Informatique et Gestion de Projet, grâce à un projet en groupe.

Le projet consiste à développer une plateforme de jeux en ligne pour la société *DUEL SUR LA TOILE*. Cette société est fictive, c'est une simulation du monde professionnel. On peut ainsi se représenter concrètement la demande d'un client, en dégager un cahier des charges, une analyse complète des besoins pour concevoir l'application de manière détaillée.

Cette plateforme est divisée en deux modules utilisateurs et deux modules jeux. Parmi les modules utilisateurs, on retrouve :

Le module Administrateur permet la gestion des différents jeux et des différents utilisateurs. On peut s'en servir pour mettre à jour un jeu ou désactiver (bannir) un joueur.

Le module Joueur permet à un utilisateur de jouer, d'envoyer et de recevoir des messages, de la part d'autres utilisateurs. Il peut communiquer, envoyer des invitations à jouer à n'importe quel jeu disponible sur la plateforme et accepter celles qu'il reçoit.

Parmi les modules jeux, il y a :

Le module Puissance 4 où 2 joueurs se défient pour aligner 4 pions de leur couleur respective dans un plateau de 7 cases par 6. La partie est terminée lorsque toutes les cases sont remplies sans qu'il n'y ait de puissance 4, ou lorsqu'un des 2 a réussi.

Le module Mastermind où 2 joueurs doivent trouver une combinaison de couleurs le plus rapidement possible. Les joueurs n'ont pas de contraintes d'essais. Il faut simplement être le plus rapide. L'application permet d'affronter un joueur sans qu'il soit obligé de jouer en même temps. Par exemple, un joueur peut inviter un autre à faire mieux que lui pour trouver la même combinaison.

Ces modules, qui représentent les 4 grandes fonctionnalités de l'application, devront posséder une interface, ergonomique, que nous développeront lors des dernières phases de développement, et qui découlera des analyses réalisées durant les premières phases de développement.

Ce dossier décrit justement la démarche utilisée durant les phases de développement.

II - Implémentation de la Base de Données

A) Dictionnaire des données

La plateforme de jeux nécessite une base de données. Pour créer une base de données fiable et claire, nous avons écrit un dictionnaire des données.

Un dictionnaire des données sert à recenser les différentes données qui seront stockées dans la base de données, ainsi que toutes les informations utiles sur ces données. Avec le dictionnaire des données, on nomme chaque donnée afin de ne pas la confondre avec d'autres.

Entité / Association	Nom	Definition	Structure	Type	Quantité	Exemple	Commentaire
User	idU	Identifiant du joueur	Number	élémentaire	-	1256	L'identifiant est unique et invariable
	pseudoU	Pseudonyme du joueur	VarChar	élémentaire	-	XxX_D4rK5a5uk3_XxX	Le pseudo est unique mais variable
	emailU	Email du joueur	VarChar	élémentaire	-	nom.prenom@exemple.xx	l'adress email est unique mais variable
	passwordU	Le mot de passe du joueur	VarChar	élémentaire	-	Squeezie45	Le mot de passe est unique mais variable
	activatedU	Statut d'activation de l'utilisateur	Boolean	élémentaire	2	true	Un utilisateur désactivé ne peut pas se connecter sur la plateforme
	typeU	Le type de l'utilisateur	Number	élémentaire	3	2	L'utilisateur peut-être (0) admin / (1) développeur / (2) joueur
Asso : Stat (User->Game)	nbWin	Le nombre de parties gagnées par le joueur sur le jeu	Number	élémentaire	-	15	
Game	idG	Identifiant du jeu	Number	élémentaire	-	111	Le fichier exécutable aura comme nom son idG
	nameG	Nom du jeu	VarChar	élémentaire	-	Puissance4	
	descriptionG	Description du jeu	VarChar	élémentaire	-	Le but du jeu est d'aligner 4 pions de la même couleur sans que sont adversaire y parvienne.	
	fileG	Le fichier jar contenant l'exécutable	LONGBLOB	élémentaire	-	0101100011010101010000001101111011...	
	versionG	Le numero de la version actuelle du jeu	Float	élémentaire	-	10.21	Construction : "V" + code
	typeG	Le type du jeu	VarChar	élémentaire	4	score	Un jeu peut avoir pour type : Score / Speed / Turn-Based
InstanceGame	numP	Le numero de la partie	Number	élémentaire	-	32	
	stateP	L'avancement de la partie	Number	élémentaire	-	-2	-2 (abandonnée) / -1 Fini / Score ou numManche >=0
Asso : Send	hasBeenRead	Lecture du message par le destinataire	Boolean	élémentaire	2	FALSE	false signifie que le message n'a pas été lu
Message	idM	Identifiant du message	Number	élémentaire	-	26548	Construction : id expéditeur + "-" + id receveur
	dateM	Date d'envoi	DateTime	élémentaire	-	26/03/18	
	contentM	Le contenu du message	VarChar	élémentaire	-	Salut, tu veux jouer avec moi ?	

On définit chaque donnée avec ce qu'elle va représenter, on indique également à quel type de structure la donnée appartient-elle (si la donnée est un nombre entier, un nombre décimal, du texte, un élément avec 2 possibilités, une suite binaire, ou encore une date).

La donnée va également posséder un type (élémentaire pour les valeurs que l'on ne peut déduire purement à l'aide d'autres, ou calculable si elle est déductible). On indique également la quantité supposée de possibilités que prend la donnée (par exemple le Pseudo du joueur peut avoir une infinité de valeur, alors que la donnée activateU qui indique si le compte est activé ou non ne peut avoir que deux possibilités : True ou False (Vrai ou Faux)).

On donne ensuite un exemple de valeurs que peut prendre la donnée. Puis on redonne une description plus détaillée qui explique entièrement à quoi sert la valeur.

On a séparé les données contenus dans le dictionnaire en entités (quelque chose qui a une existence concrète) et associations (relations entre plusieurs entités) après les avoir toutes listées dans un tableur. Une association est signalée par les termes "Asso: " avant son nom.

Chaque entité possède une “clé primaire”, une valeur qui ne peut être présente qu’une fois pour ce champ dans la base de données, souvent nommé id ou num suivi du terme qui la distingue. Par exemple, idU est la clé primaire de l’entité User et il ne peut donc n’y avoir qu’un seul utilisateur portant l’idU 1254. Cette clé primaire permet de faire en sorte qu’il ne puisse pas avoir de “conflits” dans la base de données entre deux utilisateurs dans cet exemple, mais cela vaut pour toutes les entités.

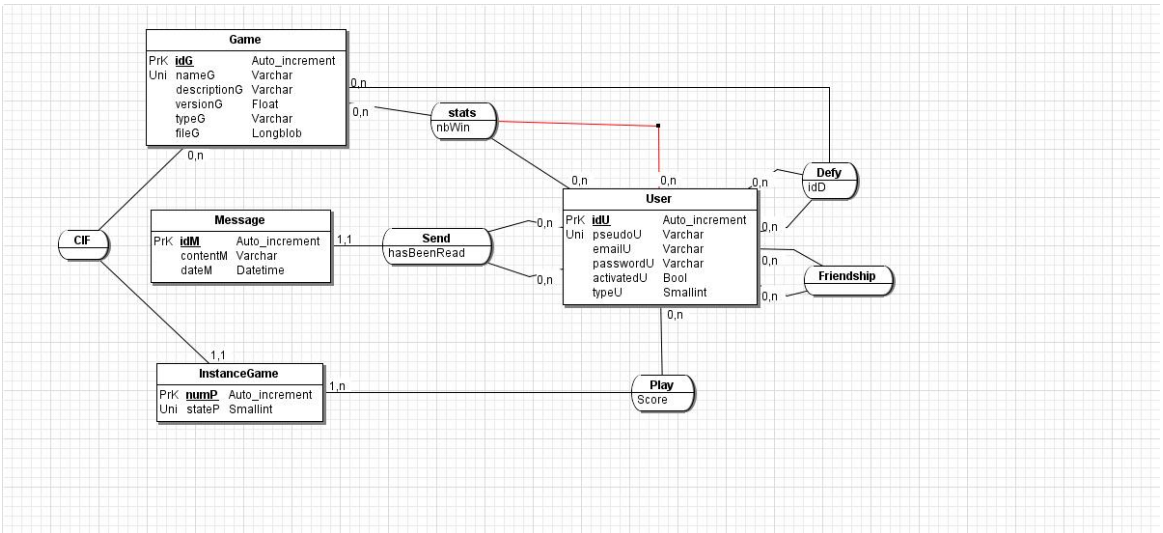
Une requête est une commande permettant d’interroger une base de données.

Requêtes SQL : Le SQL est un langage servant à gérer des bases de données. Les requêtes SQL sont des commandes écrites en SQL afin de stocker, de récupérer et d’agencer des informations dans une base de données.

B) MCD

Le *MCD* (ou Modèle Conceptuel des Données) est un schéma servant, lors du développement, à décrire certaines des *interactions* entre différentes données présentes dans le système. Il sert principalement à connaître les liens possibles entre chaque entité de notre application pour ensuite les transformer en une base de données. De plus, ce schéma permet d'éviter au maximum la *redondance d’informations* en essayant de séparer les éléments. Il permet donc une *optimisation* en donnant une vue d’ensemble des différentes données de l’application.

Ce diagramme est divisé en tables, appelées “*entités*”, qui représentent la plupart du temps un objet distinct de l’application. Chacune de ces entités contient des “attributs” ayant différents types (nombres, caractères, booléen, etc…) représentant des *informations*. Ces tables ont une ou plusieurs “*instances*”, c’est-à-dire qu’il peut y avoir plusieurs fois la même entité avec des attributs différents. Voici ci-dessous la première version de la phase 1 de notre MCD et de notre script de création de la base de données.



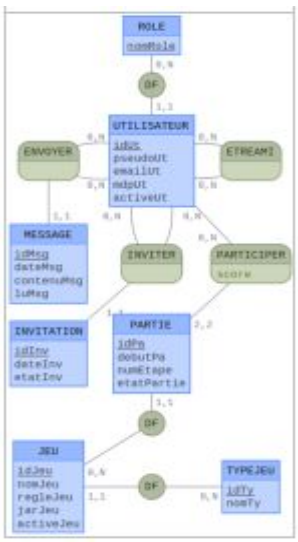
1.0 MCD initial

```sql
DROP TABLE DEFY;
DROP TABLE FRIENDSHIP;
DROP TABLE STATS;
DROP TABLE SEND;
DROP TABLE PLAY;
DROP TABLE MESSAGE;
DROP TABLE INSTANCEGAME;
DROP TABLE GAME;
DROP TABLE USER;


CREATE TABLE USER(
  idU INT UNIQUE AUTO_INCREMENT,
  pseudoU VARCHAR(20),
  emailU VARCHAR(255),
  passwordU VARCHAR(50),
  activatedU BOOLEAN,
  typeU SMALLINT,
  PRIMARY KEY (idU)
);

CREATE TABLE GAME(
  idG INT UNIQUE AUTO_INCREMENT,
  nameG VARCHAR(20),
  descriptionG VARCHAR(400),
  versionG FLOAT,
  typeG VARCHAR(20),
  fileG TEXT,
  PRIMARY KEY (idG)
);

CREATE TABLE INSTANCEGAME(
  numP INT UNIQUE AUTO_INCREMENT,
  stateP INT,
  idG INT,
  PRIMARY KEY (numP),
  FOREIGN KEY (idG) REFERENCES GAME(idG)
);

CREATE TABLE MESSAGE(
  idM INT UNIQUE AUTO_INCREMENT,
  dateM DATETIME,
  contentM VARCHAR(250),
  PRIMARY KEY (idM)
);

CREATE TABLE PLAY(
  numP INT,
  score INT,
  idU INT,
  PRIMARY KEY (numP,idU),
  FOREIGN KEY (idU) REFERENCES USER(idU),
  FOREIGN KEY (numP) REFERENCES INSTANCEGAME(numP)
);
```

```sql
CREATE TABLE SEND(
  idM INT UNIQUE AUTO_INCREMENT,
  idSender INT,
  idReceiver INT,
  hasBeenRead BOOLEAN,
  FOREIGN KEY (idSender) REFERENCES USER(idU),
  FOREIGN KEY (idReceiver) REFERENCES USER(idU),
  FOREIGN KEY (idM) REFERENCES MESSAGE(idM),
  PRIMARY KEY (idSender,idReceiver,idM)
);

CREATE TABLE STATS(
  nbWin INT,
  idG INT,
  idU INT,
  idOpponent INT,
  PRIMARY KEY(idG,idU,idOpponent),
  FOREIGN KEY (idU) REFERENCES USER(idU),
  FOREIGN KEY (idG) REFERENCES GAME(idG),
  FOREIGN KEY (idOpponent) REFERENCES USER(idU)
);

CREATE TABLE FRIENDSHIP(
  idUser1 INT UNIQUE,
  idUser2 INT UNIQUE,
  FOREIGN KEY (idUser1) REFERENCES USER(idU),
  FOREIGN KEY (idUser2) REFERENCES USER(idU)
);

CREATE TABLE DEFY (
  idD INT UNIQUE AUTO_INCREMENT,
  idUser1 INT,
  idUser2 INT,
  idG INT,
  FOREIGN KEY (idUser1) REFERENCES USER(idU),
  FOREIGN KEY (idUser2) REFERENCES USER(idU),
  FOREIGN KEY (idG) REFERENCES GAME(idG)
);
```

1.1 Script initial de création de BD

Afin d'avoir une base conceptuellement correcte pour tous les groupes, le corps enseignant nous a fourni un MCD et un script de création de base de données pour toutes les phases suivantes. Les voici respectivement :



1.2 MCD final

```sql
CREATE TABLE USER(
  idU INT UNIQUE AUTO_INCREMENT,
  pseudoU VARCHAR(20),
  emailU VARCHAR(255),
  passwordU VARCHAR(50),
  activatedU BOOLEAN,
  typeU SMALLINT,
  PRIMARY KEY (idU)
);

CREATE TABLE GAME(
  idG INT UNIQUE AUTO_INCREMENT,
  nameG VARCHAR(20),
  descriptionG VARCHAR(400),
  versionG FLOAT,
  typeG VARCHAR(20),
  fileG TEXT,
  PRIMARY KEY (idG)
);

CREATE TABLE INSTANCEGAME(
  numP INT UNIQUE AUTO_INCREMENT,
  stateP INT,
  idG INT,
  PRIMARY KEY (numP),
  FOREIGN KEY (idG) REFERENCES GAME(idG)
);

CREATE TABLE MESSAGE(
  idM INT UNIQUE AUTO_INCREMENT,
  dateM DATETIME,
  contentM VARCHAR(250),
  PRIMARY KEY (idM)
);

CREATE TABLE PLAY(
  numP INT,
  score INT,
  idU INT,
  PRIMARY KEY (numP,idU),
  FOREIGN KEY (idU) REFERENCES USER(idU),
  FOREIGN KEY (numP) REFERENCES INSTANCEGAME(numP)
);

CREATE TABLE SEND(
  idM INT UNIQUE AUTO_INCREMENT,
  idSender INT,
  idReceiver INT,
  hasBeenRead BOOLEAN,
  FOREIGN KEY (idSender) REFERENCES USER(idU),
  FOREIGN KEY (idReceiver) REFERENCES USER(idU),
  FOREIGN KEY (idM) REFERENCES MESSAGE(idM),
  PRIMARY KEY (idSender,idReceiver,idM)
);
```

```sql
CREATE TABLE STATS(
  nbWin INT,
  idG INT,
  idU INT,
  idOpponent INT,
  PRIMARY KEY(idG,idU,idOpponent),
  FOREIGN KEY (idU) REFERENCES USER(idU),
  FOREIGN KEY (idG) REFERENCES GAME(idG),
  FOREIGN KEY (idOpponent) REFERENCES USER(idU)
);

CREATE TABLE FRIENDSHIP(
  idUser1 INT UNIQUE,
  idUser2 INT UNIQUE,
  FOREIGN KEY (idUser1) REFERENCES USER(idU),
  FOREIGN KEY (idUser2) REFERENCES USER(idU)
);

CREATE TABLE DEFY (
  idD INT UNIQUE AUTO_INCREMENT,
  idUser1 INT,
  idUser2 INT,
  idG INT,
  FOREIGN KEY (idUser1) REFERENCES USER(idU),
  FOREIGN KEY (idUser2) REFERENCES USER(idU),
  FOREIGN KEY (idG) REFERENCES GAME(idG)
);
```

1.3 Script final de création de Bd

III - Analyse, description, conception

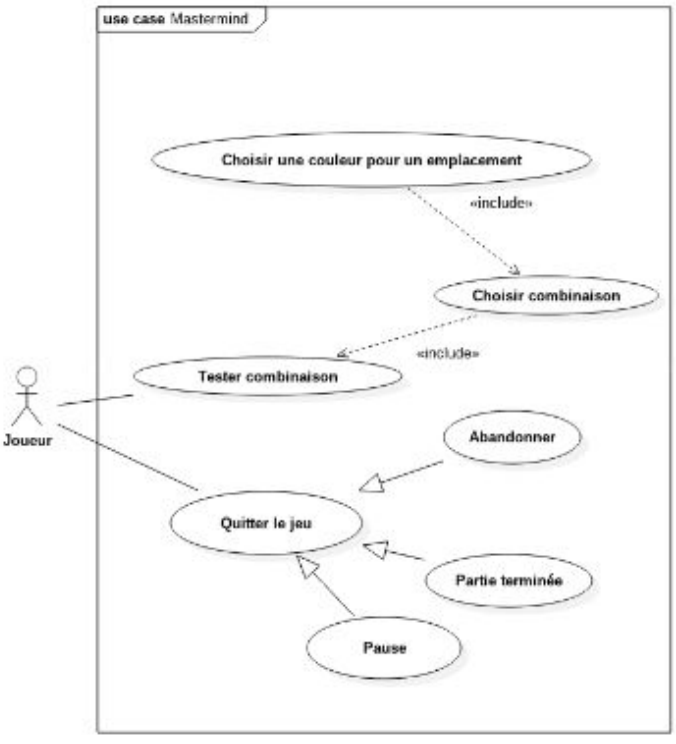
La description rapide des différents modules que nous devons implémenter ne nous permet pas de les développer. Nous devons concevoir, de façon beaucoup plus détaillée, des éléments d'analyse décrivant en profondeur le système voulu.

L'analyse détaillée est une méthode très importante chez l'informaticien, puisqu'elle lui permet d'avoir à l'avance un aperçu à la fois global, précis et sous différents angles, de tout ce qu'il va falloir implémenter (en terme de codage) dans le projet. Pour la même raison, elle lui facilite grandement la maintenance de son programme, puisqu'il peut localiser très simplement une ligne de code à changer, même parmi des milliers de lignes.

Pour ce faire, nous utilisons différents diagrammes (diagrammes de Cas d'Utilisation ou CU, diagramme de Séquence, diagramme de décomposition par tâches,...) ainsi que d'autres éléments comme des descriptions textuels décrivant chacun des modules.

A) Cas d'Utilisations (CU)

Un CU est une action impliquant le système étudié. Dans l'exemple ci-dessous, "Tester combinaison" est un CU concernant le système Mastermind.
Les différents CU qui composent un diagramme ont des liens entre eux, mis en évidence par différentes flèches. Celle avec «include» signifie que le CU où celle-ci va nécessiter l'exécution du CU d'où elle part. Par exemple il faut choisir une combinaison pour la tester. Une autre flèche est utilisée aussi. C'est le symbole des Cas Particuliers. Par exemple, "Quitter un jeu" peut être fait en mettant pause, en abandonnant ou en terminant une partie.



B) Diagramme des cas d’utilisation :

Un diagramme des cas d’utilisation est un diagramme permettant de modéliser les différentes fonctionnalités d’un système (en l'occurrence ici une plateforme de jeux) ainsi que les différents acteurs qui vont utiliser le système, pour que l’on puisse ainsi savoir plus facilement et plus rapidement comment utiliser le système, comment il fonctionne, qui a accès à quelles fonctionnalités, etc.

Celui de l’application “Duel sur la toile” contient deux acteurs, c’est à dire deux types de personnes qui vont interagir avec le système : l’utilisateur (le joueur) et l’administrateur qui est également un utilisateur. L’administrateur a donc accès aux cas d’utilisation des utilisateurs mais dispose aussi de ses propres cas d’utilisation.

L’administrateur a donc la possibilité de gérer les jeux présent sur cette plateforme (en ajouter ou en supprimer) puis gérer les joueurs (comme par exemple supprimer des comptes joueurs).

Les cas d’utilisation de l’utilisateur peuvent être séparés en deux parties : une partie concernant ses parties de jeux sur la plateforme (comme par exemple consulter ses statistiques ou celles de ces adversaires) et l’autre partie concernant ses amis (comme créer une liste d’amis ou leur envoyer des messages par exemple).

C) Description textuelle :

À partir du diagramme de CU, on peut aller plus loin dans la description du fonctionnement d’un module, grâce à la description textuelle. Comme son nom l’indique, cette analyse va décrire les scénarios correspondant à l’action en question et peut être réaliser à la place ou en complément du diagramme de séquence. Voici par exemple la description textuelle de l’action “Quitter le jeu” :

Sommaire d’authentification :

- *Titre* : Quitter la partie
- *Acteur* : Joueur
- *Résumé* : Le joueur quitte la partie, mettant les scores à jour et notifiant les deux joueurs des résultats
- *Date* : 23/04/2018
- *Responsable* : Groupe 1A12.B
- *Version* : 1.0
- *Précondition(s)*: Deux joueurs connectés ont lancé partie de Mastermind. La partie vient de s’achever.
- *Post-condition(s)*: /

Scénario nominal :

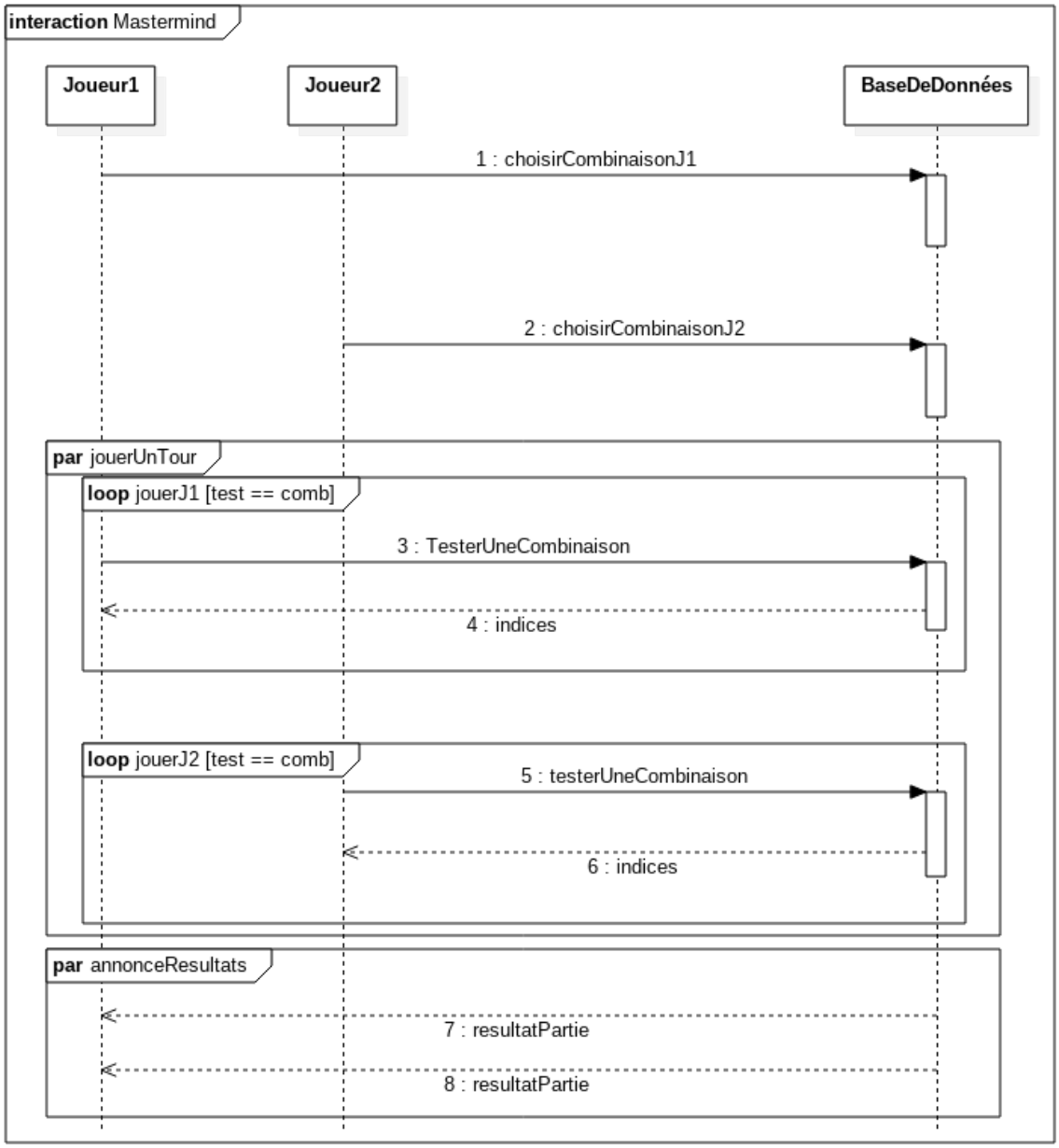
Acteur	Système
4- Clique sur le bouton affiché à l'écran	1- Notifie le vainqueur de sa victoire et le vaincu de sa défaite 2- Notifie l'autre joueur que l'adversaire a quitté la partie. 3- Affiche un bouton "Fin de partie" 5- Notifie le joueur n'ayant pas encore quitté la partie du départ de son adversaire 6- Met les statistiques des joueurs à jour.

Il existe plusieurs types de scénarios :

Le scénario nominal est celui qui advient dans le meilleur des cas, sans aucun problème ; le scénario alternatif décrit une action que l'utilisateur peut potentiellement faire, qui n'est pas exactement prévue par le module, mais qu'il peut tout de même faire; le scénario d'exception se met en place lorsqu'il y a un problème dans le processus (par exemple dans cette application, si un utilisateur tape un mauvais mot de passe pour se connecter à son compte, ce scénario se mettra en place).

D) Diagramme de séquence :

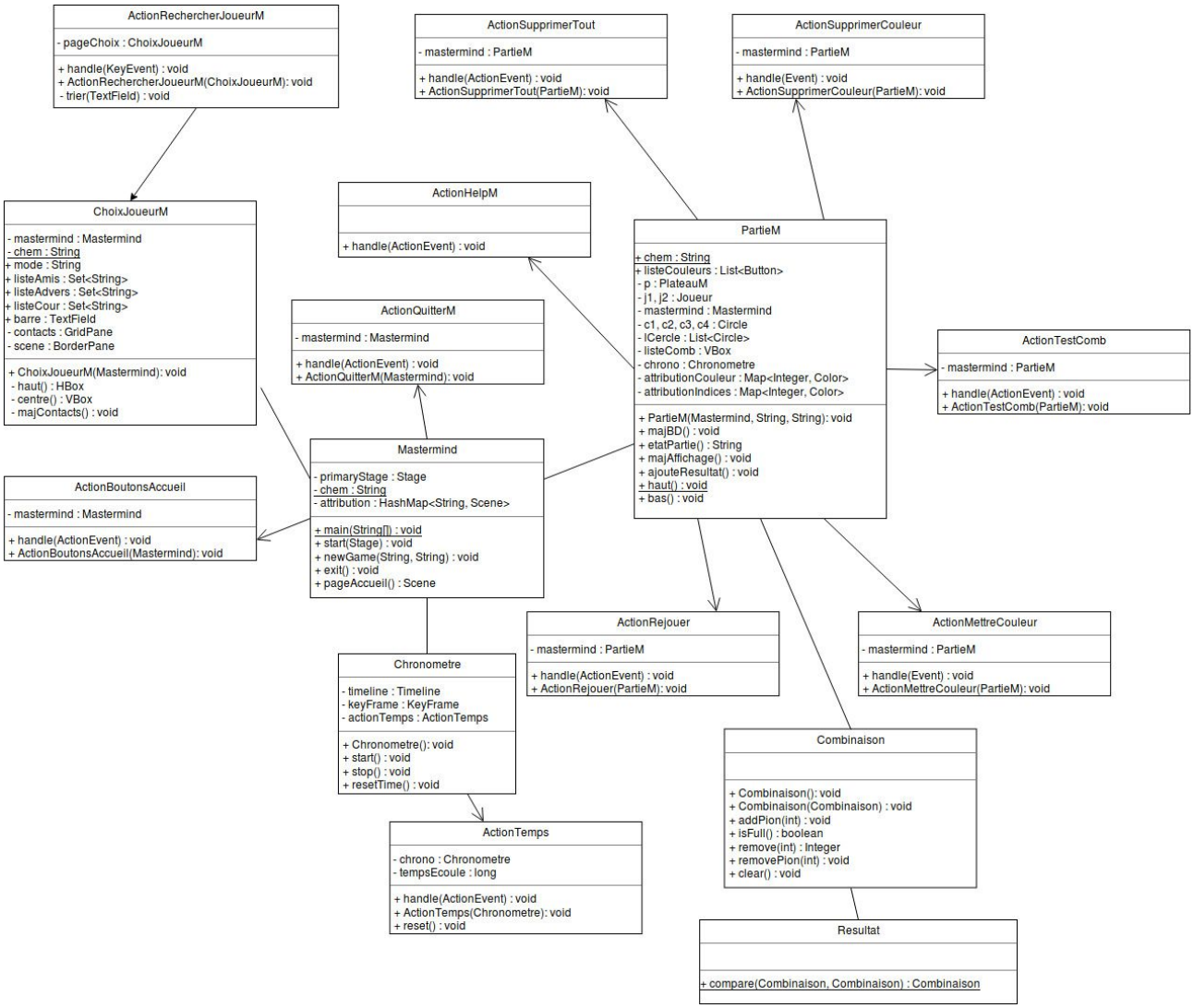
Une autre méthode d'analyse est le diagramme de séquence. C'est une représentation temporelle des objets et de leurs interactions.



E) Diagramme de classes

Le diagramme de classes est l'outil d'analyse le plus poussé. Si détaillé qu'on peut en déduire le codage des fonctionnalités. Mais pour la même raison, il est très long et difficile de l'expliquer.

Voici le diagramme de classes du module Mastermind.



Ce que chaque informaticien doit savoir de cette analyse, c'est que chaque bloc est une classe, avec un nom, des attributs et des méthodes (des fonctions qui s'appliquent à cette classe), les traits sont des associations qui représentent la connexion entre ces classes dans le programme. Les flèches indiquent qu'une classe est utilisée dans une autre.

F) Maquettage

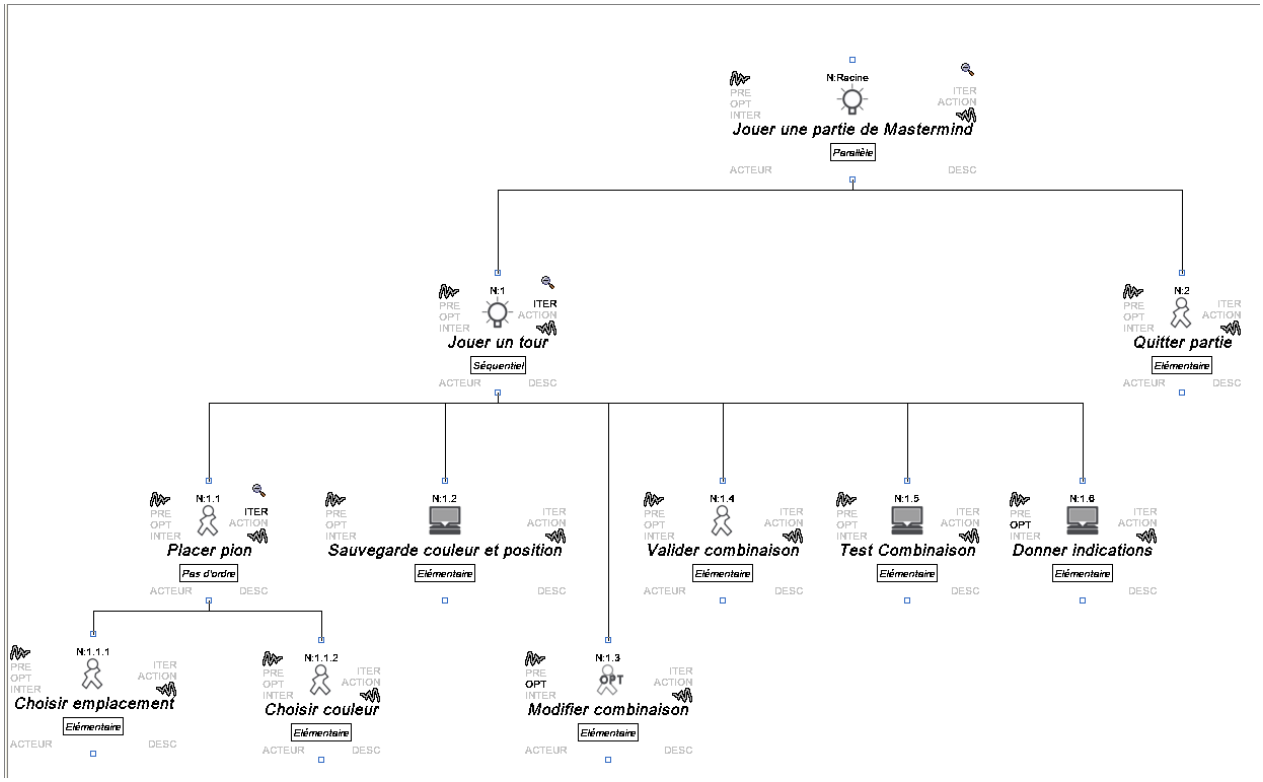
L'analyse détaillée est l'étape qui précède le maquettage, où nous choisissons et mettons en forme, visuellement et conceptuellement, les fonctionnalités de notre programme.

Dans cette partie, nous allons décrire notre démarche pour construire les maquettes des différentes fonctionnalités de l'application et notre première approche de l'implémentation des modules, en partant de l'analyse détaillée.

Lorsque le programme à développer n'est pas très intuitif (qu'il n'est pas simple de le visualiser à l'esprit), on effectue une décomposition grâce au modèle des tâches/

Un modèle des tâches est un schéma (réalisé sur le logiciel KMADe) permettant de voir l'enchaînement des tâches réalisé lors de l'exécution du module. Il reprend les éléments de la description textuelle et les assemble dans l'ordre adéquat.

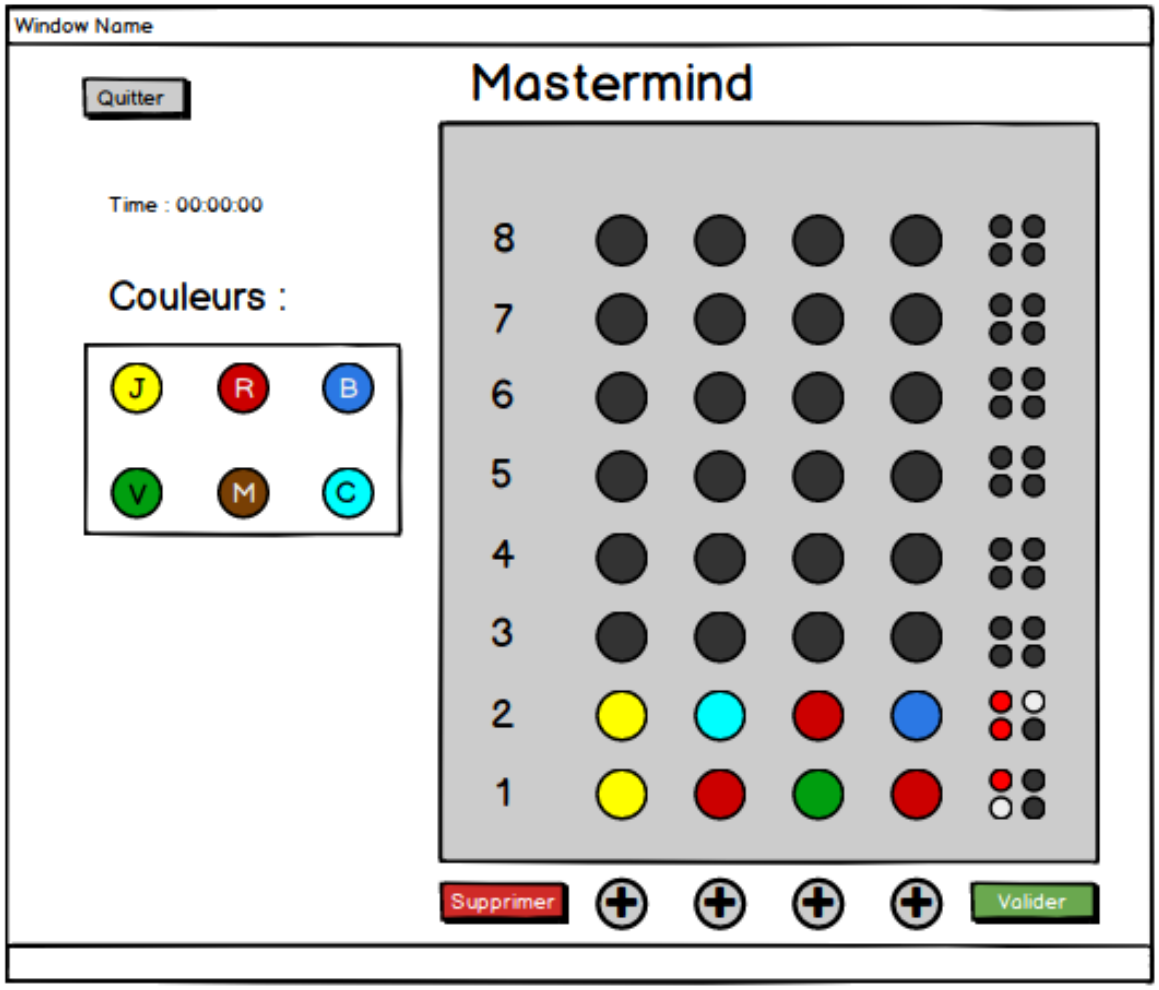
En exemple, nous avons pris le modèle des tâches du Mastermind :



De cette manière, le déroulement des différentes fonctionnalités et cette fois clarifié, sans aucune ambiguïté possible. On peut alors en déduire une première maquette de l'application.

Une maquette est une représentation mentale d'un module ergonomique et fonctionnel projeté sur un support visuel. Ce n'est qu'un brouillon, un modèle pour l'interface réalisé, néanmoins celle-ci pourra comporter des différences avec la maquette.

Voici notre maquette de la page du plateau pour le module Mastermind :



IV - Implémentation des modules

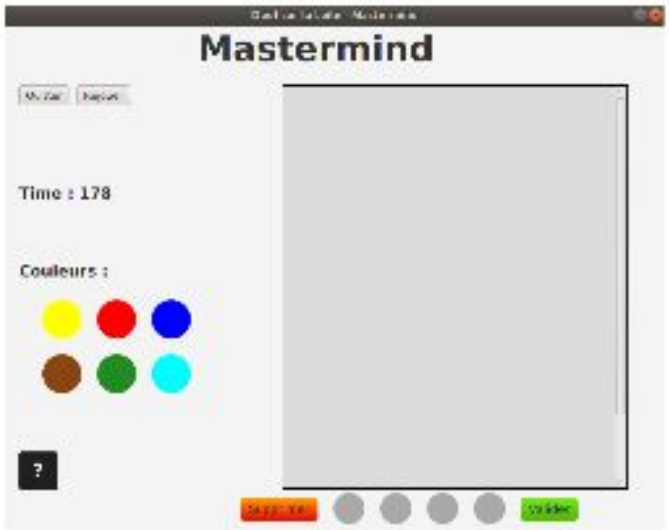
À partir de la BD fournie, de l'analyse détaillée et des maquettes, nous avons commencé le lundi 11 juin le développement de tous les modules. En terme d'organisation, nous avons fait en sorte de travailler sur les 4 modules en même temps. Nous allons dans ce chapitre expliquer comment nous avons implémenté les pages de la plateforme, ajouté des interactions et fait fonctionner les algorithmes qui modélisent les différents modules.

Voici ci-dessous les trois pages qui composent le module Mastermind, après développement.

Page d'accueil du Mastermind :

Page de choix de joueur :

Plateau du Mastermind :



Nous avons utilisé le langage JavaFX pour coder tous les modules de la plateforme. Nous avons utilisé la façon de coder “Modèle-Vue-Contrôleur”, qui sépare nos fichiers en trois catégories, les Modèles qui eux gèrent les données utilisées par l’application, les Vues qui sont l’affichage graphique des modèles, et les Contrôleurs qui gèrent les différentes interactions entre l’utilisateur et la vue, modifiant ainsi les données du modèle et l’affichage.

Néanmoins, nous n’avons pas eu le temps de lier les jeux à la base de données. Si nous avions pu, nous aurions utilisé la bibliothèque Java JDBC, qui permet de faire des fonctions en Java utilisables pour la gestion des bases de données, ainsi que des requêtes SQL. L’état des parties auraient été sauvegardés dans un fichier JSON pour pouvoir retrouver les informations à propos de la partie facilement et de pouvoir charger les parties interrompus facilement. Nous n’avons pas eu le temps d’implémenter ce lien avec la base de données car nous avons rencontré des problèmes avec l’implémentation de la base de données elle-même.

Le module Joueur est complètement lié à la base de données et peut la mettre à jour ou même y ajouter des données. Pour lier ce module à la base de données, nous avons créé une API en MySQL. Elle prend la forme d’une bibliothèque appelée par tous les autres modules, une seule connexion à la base de données est effectuée par instance de l’application.

Le module Administrateur a été partagé en trois parties bien distinctes : la partie graphique, la partie action des joueurs et enfin la partie gérant les données. Ce module est relié à la base de donnée grâce à l’API MySQL développé par le groupe. Chaque action faite sur l’administration a un lien avec les informations de la base de données. Le module administrateur fais de nombreuses requêtes de recherche dans la base de données pour afficher les informations des joueurs. De plus, il effectue des requêtes de mise à jour, notamment lors de l’édition d’un profil.

Les statistiques des jeux et des joueurs que n’ont pas pu être programmées car nos jeux ne sont pas mis en ligne.

V - Partie IHM - Choix d'ergonomie

Dans ce chapitre, nous allons détailler la composition des différentes fenêtres de notre application. Les choix d'ergonomie sont très importants afin d'évaluer la qualité du programme. Nous allons ainsi parler des 3 facteurs humains qui fondent nos choix : l'Apprenabilité, la Flexibilité et la Robustesse. Pour chacun d'eux, nous allons mettre en avant les principes ergonomiques auxquels nous avons pensés.

A) L'Apprenabilité

Ce facteur humain est le premier qui est jugé par l'utilisateur, puisqu'il s'opère dès l'ouverture de l'application. En effet, les premières observations de celui-ci vont lui permettre de s'orienter et de comprendre les différentes fonctionnalités, et ainsi d'évaluer sa facilité à prendre en main le logiciel. Pour ce projet, nous avons insisté sur l'apprenabilité grâce aux principes de Familiarité, de Guidage et d'Observabilité.

Ci-dessus la fenêtre d'inscription à la plateforme, pour illustrer nos choix pour l'Apprenabilité.

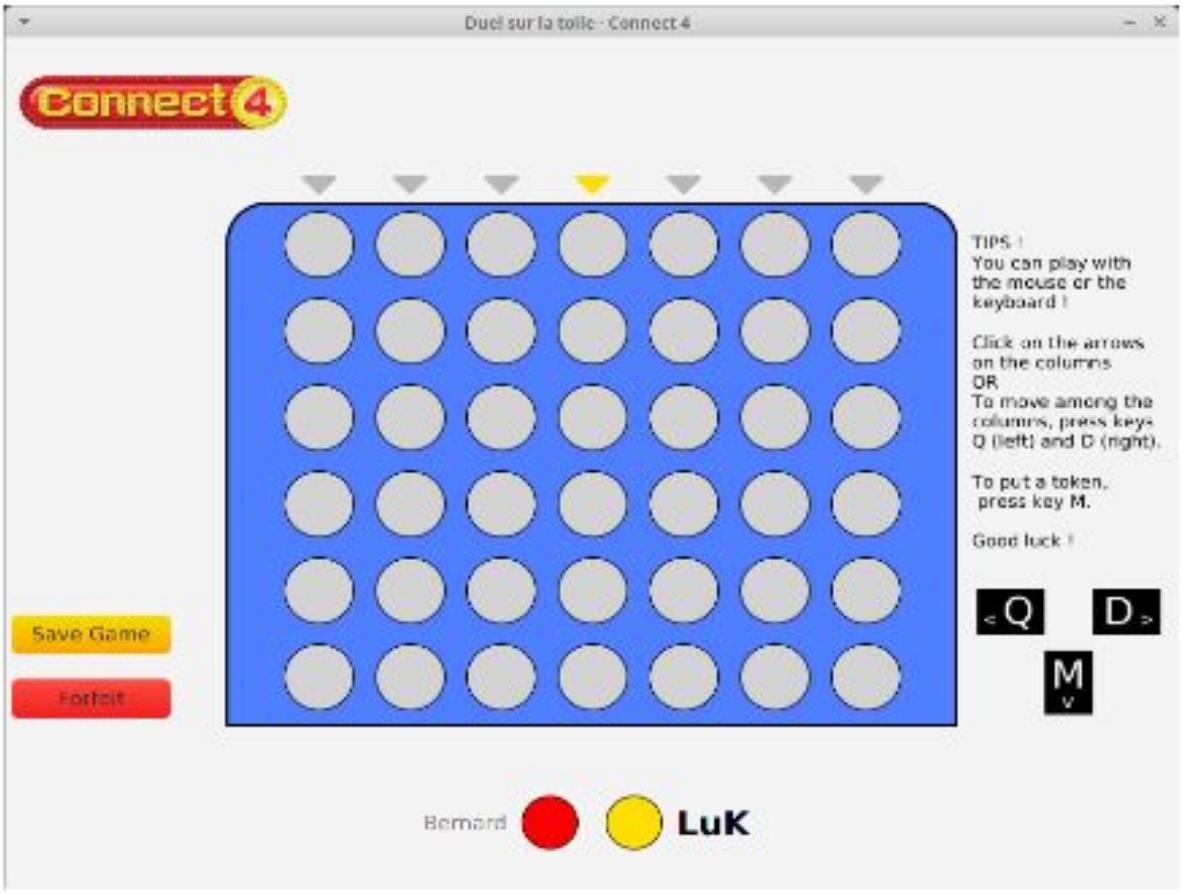
Tout d’abord, la Familiarité vient de la position des éléments sur cette page. En effet, sur de nombreux sites ou applications, l’inscription se fait à droite de la page et la connexion généralement à gauche. Ainsi, nous avons mis les champs à remplir de ce côté là afin que l'utilisateur connaisse ce genre de disposition. De plus, les labels sont aussi une forme de familiarité car l’utilisateur est habitué à devoir remplir des champs en dessous.

Ici, le Guidage vient principalement des zone de texte à droite. L’utilisateur voit qu'à gauche aucun texte est à entrer et donc se dirige vers la droite. Il y a un enchaînement de zone de texte avec une indication de ce qu’il doit rentrer. En cas d’ambiguïté, le futur inscrit peut voir quelques informations en plus grâce aux infobulles (qui apparaissent après 3 secondes où la souris est au-dessus du champ). Une fois qu’un champ est rempli, il doit passer à celui d’en dessous et est guidé grâce à l’ordre vertical du haut vers le bas de ces zone de texte. Enfin, après tous les champs, nous avons mis un bouton “S’inscrire” sur lequel l’utilisateur doit cliquer. Le guidage est encore présent car depuis le début il suit l’ordre vertical des champs et donc la prochaine étape est ce bouton. Enfin, le guidage est aussi renforcé grâce aux textes au dessus des champs à remplir indiquant le titre du champ et ce qu’il doit y mettre.

Enfin, on a augmenté l’Observabilité : la partie pour l’inscription se situe uniquement à droite de la page alors que le reste (pour revenir ou le logo) se situe à gauche. Il y a donc deux parties bien distinctes que l’utilisateur peut repérer.

B) La Flexibilité

Ce facteur humain reflète la capacité de l'application à s'adapter en fonction du style d'utilisation d'un utilisateur. En effet, un débutant qui joue sur une plateforme numérique doit tout aussi facilement s'amuser qu'un expert, qui a déjà connu plusieurs logiciels comme celui-ci.



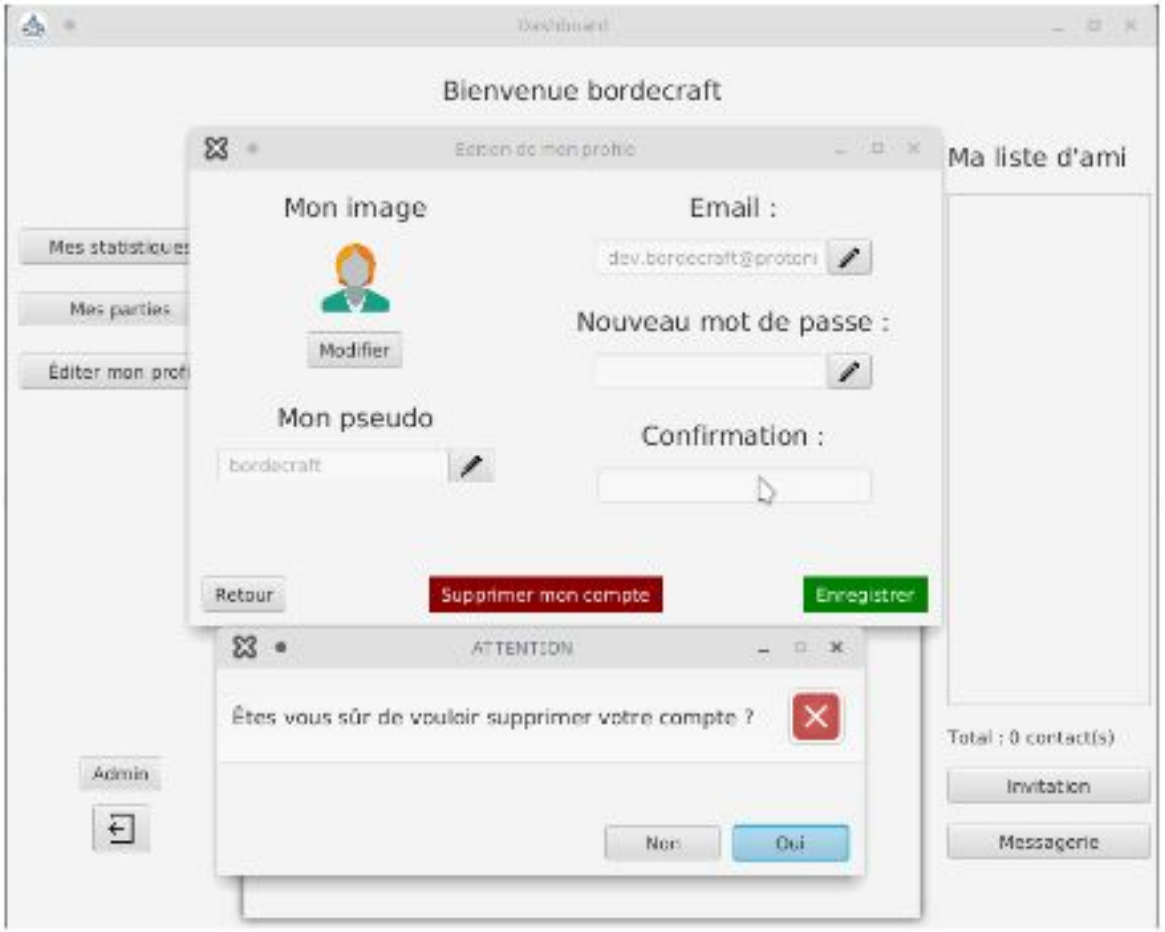
Ci-dessus la fenêtre d'une partie de Puissance 4, qui nous permettra d'illustrer nos choix pour la Flexibilité.

Premièrement, l'adaptation du plateau du jeu aux différents styles d'utilisations provient de la possibilité d'utiliser la souris pour ajouter un pion coloré, comme le feraient principalement les nouveaux joueurs, ou en utilisant le clavier, avec les 3 touches indiquées à droite, ce qui est pratique pour le joueur expérimenté. Ensuite, le guidage, avec des conseils d'utilisations écrits à droite de la fenêtre (qui s'ajoutent aux règles du jeu dans la page de choix de joueur) permettent d'éviter toute ambiguïté sur cette pratique.

Ainsi, un logiciel flexible sera intéressant pour plus de personnes, un point fort non négligeable pour la vente par exemple.

C) Le Robustesse

Ce troisième facteur humain traduit la réalisation des différentes fonctionnalités face aux problèmes possibles. Une application robuste limite les risques de plantage et d'erreurs d'inattention de la part de l'utilisateur. La Robustesse intervient donc avant et après toute erreur possible



Pendant notre développement, nous avons mis l'accent sur la robustesse de notre application. Pour se faire, nous avons mis en place un guidage de l'utilisateur très simple et compréhensible, permettant de ne pas se perdre.

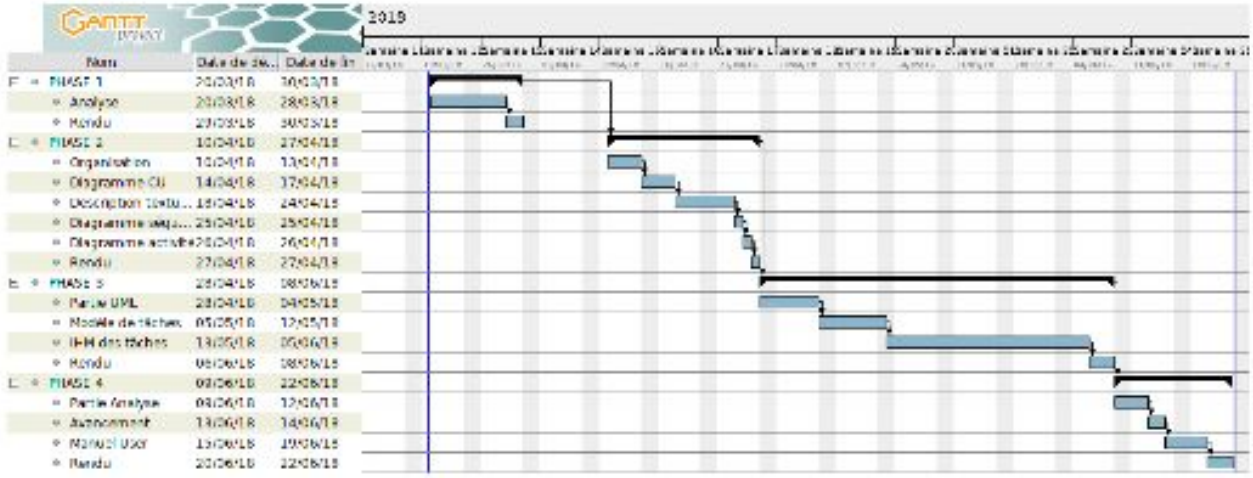
Pour se faire, nous avons développé un IHM léger, peu surchargé en informations, ainsi que plusieurs pop-up de prévention, pensé pour être le plus compréhensible possible, même par les personnes les moins expérimentés.

De plus, dans les cas conduisant à des modifications importantes, les actions doivent être confirmées, allant du simple pop-up "oui/non" jusqu'à la confirmation du mot de passe pour les actions très importantes.

VI - État d'avancement du projet

Dans cette avant-dernière partie, nous allons détailler notre avancement vis-à-vis de notre organisation, ce qui a été fait, ce qui n'a pas été fait et pourquoi.

Lors de la phase 2 du projet, nous devions établir un planning et une répartition des tâches, afin de s'organiser pour les 2 phases suivantes. Nous avons utilisé deux outils, le diagramme de GANTT pour le planning et la matrice RACI pour la répartition des tâches, dont vous pouvez voir ci-dessous les versions.



MATRICE RACI Groupe 1A - 1.2.A

	Mathieu	Lucas	Valentin	Léo	Nolan	Baptiste	Benjamin	Ladislas
Responsable	4	24	31	31	31	31	31	31
Acteur	25	8	-	-	-	-	-	-
Consulté	4	2	2	2	2	2	3	2
Informé	1	2	2	2	2	2	1	2

Phase	Groupe de Tâches 1									Responsable	Acteur	Consulté	Informé	Total	Cumulé
1	Dico	Tâche 1	A4	R3	R3	R3	R3	R3	R3	R3	18	4	-	-	22	22
	MCD	Tâche 2	A2	R3	R3	I1	I1	I1	R3	I1	9	2	-	3	14	36
	Diag CU	Tâche 3	A2	I1	I1	R3	R3	R3	I1	R3	9	2	-	3	14	50
	BD Script/Requêtes	Tâche 4	C1	A2	I1	I1	I1	I1	R1	I1	1	2	1	4	8	58
	Rendu Phase 1	Tâche 5	A1	C1	R1	R1	R1	R1	C1	R1	4	1	2	-	7	65
2	Diag CU	Tâche 6	A1	R2	R2	R2	R2	R2	R2	R2	12	1	-	-	13	78
	Desc textuelle	Tâche 7	I1	A1	R2	R2	R2	R2	R2	R2	10	1	-	1	12	90
	Diag Séquence	Tâche 8	A1	I1	R1	R1	R1	R1	R1	R1	5	1	-	1	7	97
	Diag Activité	Tâche 9	R1	A1	R1	R1	R1	R1	R1	R1	6	1	-	-	7	104
	Rendu phase 2	Tâche 10	A2	R2	R2	R2	R2	R2	R2	R2	12	2	-	-	14	118
3	UML	Tâche 11	A2	R3	C1	R2	C1	R2	C1	R2	7	2	3	-	12	130
	Modèle des tâches	Tâche 12	R3	A2	R2	C1	R2	C1	R2	C1	9	2	2	-	13	143
	IHM	Tâche 13	A2	R4	R4	R4	R4	R4	R4	R4	24	2	-	-	26	169
	Compte rendu	Tâche 14	C2	A1	R1	R1	R1	R1	R1	R1	5	1	2	-	8	177
	Rendu phase 3	Tâche 15	A2	R2	R2	R2	R2	R2	R2	R2	12	2	-	-	14	191
4	Dossier Analyse	Tâche 16	C1	A1	R2	C1	R2	C1	R2	C1	6	1	3	-	10	201
	État d'avancement	Tâche 17	A1	C1	C1	R2	C1	R2	C1	R2	4	1	4	-	9	210
	Manuel utilisateur	Tâche 18	A2	R2	R2	R2	R2	R2	R2	R2	12	2	-	-	14	224
	Rendu phase 4	Tâche 19	A3	R3	R3	R3	R3	R3	R3	R3	18	3	-	-	21	245

A) Ce que nous avons pu faire

Nous avons pu réaliser la plupart des IHM prévues, n'excluant seulement que celles étant en liens avec des fonctionnalités non implémentées.

Les deux jeux demandés ainsi qu'un troisième (Jeu des 21 Bâtonnets) ont été implémentés sous forme de JAR exécutables.

L'intégralité des IHM du module Administrateur ont été développées, ne laissant que celles dédiées aux statistiques, ainsi qu'à la mise à jour des jeux dans la base de données, non utilisables pour le moment.

Pour le module Joueur, une première version d'un système de liste d'amis a été implémentée, lié à un chat fonctionnel, ainsi qu'un système d'édition de profil et d'un accès à l'interface administrateur, pour les utilisateurs étant administrateurs.

Enfin, un "dashboard", le bureau du Joueur, contenant la liste des jeux disponibles à l'installation ainsi qu'un autre contenant la liste des jeux installés est présent, permettant le lancement des jeux en local.

B) Ce que nous n'avons pas pu faire

Les jeux sont fonctionnels en local, c'est-à-dire que 2 joueurs peuvent y jouer sur la même interface, sur le même ordinateur. Mais ils ne sont pas reliés à la base de données, ne permettant donc pas de jouer "en ligne". Autrement dit, les jeux ont une vie limitée à une seule ouverture. En les quittant, aucune sauvegarde n'est effectuée et aucune statistique n'est retenue, ce qui est dommage, mais provient de nos choix suite à notre gestion du temps restant.

Ainsi, dans le module Joueur, les fonctionnalités en rapport avec les statistiques de parties des joueurs, le nombre de parties effectuées, le nombre de victoires…, ne sont pas fonctionnelles.

C'est la fin de la description de notre état d'avancement, nous allons pouvoir conclure ce dossier.

VII - Bilan

Malgré une évidente déception de la part de l'équipe quant au non fonctionnement de certaines fonctionnalités citées auparavant, il en sort tout de même un sentiment de réussite dû à l'ensemble des implémentations fonctionnelles. On peut lancer et s'amuser sur 3 jeux, dont 1 bonus, nous pouvons créer notre compte avec un avatar, ajouter des amis à notre liste et discuter avec eux sur la messagerie en ligne. Un administrateur peut désactiver un compte, éditer son profil.

Durant ce projet, nous étions une équipe de 8. Nous avons donc décidé, dans le but d'optimiser le développement, de diviser le groupe en 4 duos, chacun se spécialisant un module précis. Ainsi, pendant les 3 premières phases notamment, chaque binôme était à l'aise avec son module. Le binôme du Puissance 4, composé de Mathieu et Lucas, formait aussi le groupe chef de projet, puisque Mathieu s'occupait, en plus du jeu, d'organiser le groupe, de répartir les tâches, de rassembler la production et d'en concevoir le rendu, et Lucas s'occupait d'organiser la gestion du Git ainsi que de nous créer un serveur SQL propre au groupe, indépendant de l'IUT.

Pendant la quatrième phase, les duos sont restés globalement les mêmes. Le Puissance 4 ayant été terminé le plus tôt, Mathieu a pu aider l'autre jeu à être implémenté, et Lucas s'est occupé de notre BD. Un problème dans la Base de Données, lors de la deuxième semaine, nous a fait perdre du temps. C'est pour cela que certains membres du groupe ont dû être présent sur plusieurs modules différents, afin de palier à ce problème. Mais notre organisation a souvent été fragile face à ce genre d'imprévus, et notre gestion du temps a souvent été sous pression. Les derniers jours de développement ont été les plus difficiles, notamment en devant choisir quelles fonctionnalités ne pourront pas être programmées. C'est ainsi que le temps donné pour réaliser ce projet, de deux semaines, nous a paru trop court, n'ayant pas eu le temps de compléter tous les éléments du cahier des charges.

Avec ce projet, nous avons pu apprendre à mieux nous organiser en équipe relativement nombreuse (nous étions 8 dans notre groupe, ce qui était un record par rapport à notre expérience en informatique). Nous avons pu approfondir nos connaissances acquises lors de nos cours à l'IUT'O, notamment en Java, JavaFX et dans l'utilisation de la bibliothèque Java JDBC pour gérer nos bases de données.

Cette expérience de projet, presque en autonomie complète, a été fructueuse, puisqu'elle nous a permis de mieux imaginer comment se déroule un projet en entreprise, et donc de nous préparer à notre vie professionnelle. Être contraint par le temps, devoir le gérer efficacement et de manière réaliste, s'adapter et faire des choix quand il y a des imprévus, sont une partie des difficultés qui fondent le monde des projets informatiques. Les vivre sera inoubliable pour chaque membre de notre groupe, mais surtout enrichissant.

Si certaines tensions ont pu apparaître à la fin du projet, à cause de la pression et de la fatigue, nous nous sommes globalement bien entendus. Nous avons pu travailler ensemble et nous ne sommes pas déçus de ce projet de fin de première année de DUT Informatique.