

Duel sur la toile

2019 - 2020

Simon CAVILLON

Maxime FEVRIER

Gwendal HERICOURT

Andrew MARY HUET DE BAROCHEZ

Nino SPINOLA

Landry VALETOUX

Groupe 11A



I. Sommaire

I.	Sommaire	1
II.	Introduction	2
1.	Objectifs du projet	2
2.	Contenu du rapport	3
III.	Base de données	3
1.	Dictionnaire des données	3
2.	Modèle Conceptuel de Données	5
3.	Modèle Logique de Données	7
4.	Les requêtes	8
IV.	Analyse de l'application et des modules	10
1.	Module joueur et administrateur	10
2.	Module Puissance 4	13
V.	Maquettage des modules	16
1.	Diagramme de classe des modules	16
2.	Modèle des tâches	17
3.	Interactions Homme-Machine	19
VI.	Réalisation	23
1.	Réalisation du module Joueur et Administrateur	23
2.	Réalisation du module Puissance 4	30
3.	Manuel utilisateur	32
VII.	Gestion du projet	33
1.	Matrice RACI	33
2.	Diagramme de GANTT	34
3.	Etat d'avancement du projet	35
	a. Etat d'avancement au 01/06	35
	b. Etat d'avancement au 19/06	36
4.	Bilan collectif	38
5.	Fiches individuelles	39
	a. Simon CAVILLON	39
	b. Maxime FEVRIER	41
	c. Gwendal HERICOURT	43
	d. Andrew MARY HUET DE BAROCHEZ	45
	e. Nino SPINOLA	48
	f. Landry VALETOUX	49
VIII.	Conclusion	50
IX.	Annexe	51

II. Introduction

1. Objectifs du projet

Le but de ce projet est de réaliser une plateforme en ligne qui serait capable d'héberger différents jeux. Sur cette dernière il y aurait au moins deux types d'utilisateurs qui représentent deux grands modules distincts :

- **Les joueurs** : qui pourront jouer entre eux sur les différents jeux proposés par la plateforme grâce à un système d'authentification avec un mot de passe et un pseudonyme. Consulter leur profil et leurs statistiques ainsi que ceux des autres joueurs mais également la possibilité d'envoyer des messages grâce à un système de messagerie et la possibilité d'avoir une liste d'amis commune sur la plateforme.
- **Les administrateurs** : qui auront les mêmes droits que les joueurs, mais dont leur but principal serait d'administrer la plateforme, de gérer ses utilisateurs et de consulter les statistiques des jeux.

Nous avons également la mission d'implémenter dans un module le jeu du **Puissance 4** en adaptant les contraintes techniques aux deux modules ci-dessus. Le jeu contiendra des parties auxquelles des utilisateurs pourront se connecter avec un nombre maximum de **2 joueurs par parties**. Une partie sera représentée sous forme d'une grille virtuelle où des pions pourront être alignés par les 2 joueurs participants à la partie, lorsqu'un joueur arrive à aligner **4 jetons de suite** ou que la **grille soit remplie**, dans l'une des deux conditions énoncées la partie se termine avec un gagnant ou non. On pourra notifier qu'une partie possède un état et donc la possibilité de **mettre en pause** celle-ci afin de permettre aux utilisateurs jouant à la partie de la reprendre plus tard.

Chaque module possède une interface Homme-Machine (IHM) afin de rendre le contact avec l'utilisateur compréhensible et ergonomique. Elles seront utilisées pour suivre l'avancement d'une partie et permettre l'affichage des différentes informations comme les statistiques ou le système de messagerie. C'est également un moyen pour permettre à l'administrateur de pouvoir gérer le jeu sans avoir des connaissances détaillées sur le code source et donc sur le langage Java.

2. Contenu du rapport

Pour répondre à ces objectifs, nous aurons besoin, dans un premier temps, de modéliser une **base de données** (BD) qui nous permettra de stocker toutes les informations relatives aux 2 modules que nous avons énoncé ci-dessus. Cela nous permettra aussi d'effectuer des statistiques pour les joueurs et les administrateurs.

Nous allons ensuite devoir réaliser une **analyse** de la future application et des modules. Cela pourra être fait avec la conception de diagrammes pour avoir un aperçu de notre vision de la plateforme et du jeu à réaliser par la suite.

Nous devons ensuite implémenter ces modules à partir de la BD en pensant évidemment à l'**IHM**. C'est à dire, à prendre en compte le point de vue de l'utilisateur et avoir une interface ergonomique simple d'utilisation.

Il faudra ensuite faire un **bilan** sur notre avancée afin de repérer les décalages avec le planning prévisionnel, ce qu'il y a à améliorer, faire fonctionner, etc. Finalement, nous ferons un bilan de groupe sur le déroulement de ce projet.

III. Base de données

1. Dictionnaire des données

Nous avons dans un premier temps listé toutes les données pour répondre aux besoins du client. Il faut préciser sur chaque attribut leur :

- Nom : nécessaire pour cibler la donnée dans la BD,
- Définition : ce que représente la donnée,
- Structure : le type de donnée (numérique/alphanumérique/booléen),
- Type : si la donnée est calculable ou non,
- Quantification : le nombre de donnée dans la BD,
- Exemple : un exemple de donnée,
- Et un commentaire (facultatif).

Nous avons donc abouti à un dictionnaire de données qui nous convient à la page suivante.

Nom	Définition	Structure	Type	Quantification	Exemple	Commentaire
idPer	Identifiant d'une Personne	Numérique	élémentaire		1	on incrémente l'id pour chaque nouvel utilisateur
pseudoPer	Pseudonyme d'une Personne	Alphanumérique	Non-calculable		xx_DarkKik0oLoL_Xx	Pas de caractères spéciaux, seulement des lettres et des chiffres
mdpPer	Mot de passe d'une Personne	Alphanumérique	Non-calculable		1Er29elop	Limiter à 30 caractères
mailPer	Adresse mail d'une Personne	Alphanumérique	Non-calculable		exemple@mail.com	
avatarPer	La source de l'avatar d'une Personne	Alphanumérique	Non-calculable		./images/avatar.png	
etatPer	L'état d'une Personne	Alphanumérique	Calculable		En partie	
idCat	Identifiant d'une Catégorie	Numérique	élémentaire	au moins deux (joueur, admin)	2	
nomCat	Nom de la Catégorie	Alphanumérique	Non-calculable	au moins deux (joueur, admin)	Administrateur	
idJeu	Identifiant du jeu	Numérique	élémentaire	1 (mais il peut en avoir plusieurs)	42	
nomJeu	Le nom du Jeu associé	Alphanumérique	Non-calculable	1 (mais il peut en avoir plusieurs)	puissance 4	
srcJeu	Chemin vers les fichiers du jeu	Alphanumérique	Non-calculable	1 (mais il peut en avoir plusieurs)	.././jeu/example.java	
dateInv	La date et l'heure d'envoi d'une invitation	Date	Calculable		01/01/20 13:56	
idPar	Identifiant d'une Partie actuellement jouée	Numérique	élémentaire		37	
datePar	La date et l'heure de la création de la partie	Date	Calculable		27/03/20 17:21	
srcEtatPar	Chemin vers les données de l'état de la partie	Alphanumérique	Non-calculable		./jeu/partiesEnCours/p1.json	Lien vers un fichier .json qui contient les données d'une partie en cours
etatPar	Indique si une partie est en cours ou fini	Alphanumérique	Non-calculable		En cours	
idGagnantPar	L'id du joueur qui a gagné la partie	Numérique	élémentaire		1	Initialisé à 0 (un identifiant qui n'existe pas) jusqu'à la fin de la partie
idPerdantPar	L'id du joueur qui a perdu la partie	Numérique	élémentaire		2	Initialisé à 0 (un identifiant qui n'existe pas) jusqu'à la fin de la partie
scoreJ1Par	Le score du joueur 1	Numérique	élémentaire		16	Initialisé à 0
scoreJ2Par	Le score du joueur 2	Numérique	élémentaire		3	Initialisé à 0
tempsPar	Temps de la partie	Date	Non-calculable		10:53	
dateMsg	La date et l'heure d'envoi d'un message	Date	Calculable		05/12/19 15:32	
texteMsg	Le contenu du message	Alphanumérique	Non-calculable		Bonjour !	
luMsg	Si le message a été lu ou pas	Booléen	Non-calculable		TRUE	
idMsg	L'id du message	Numérique	élémentaire		1	

Tableau 1 : Dictionnaire de données

2. Modèle Conceptuel de Données

Une fois le dictionnaire des données suffisamment avancé, nous avons commencé à réaliser un Modèle Conceptuel des Données (MCD) afin d'avoir un aperçu global de la base de données.

Nous avons créé le MCD à l'aide de *Mocodo online*, un logiciel en ligne d'aide à la conception de bases de données relationnelles. Cela nous a permis de faire les associations et donc de repérer les données redondantes ou manquantes et les erreurs dans les tables. Nous avons donc pu optimiser et corriger notre dictionnaire de données.

Nous avons finalement abouti au MCD correspondant à la figure 1.

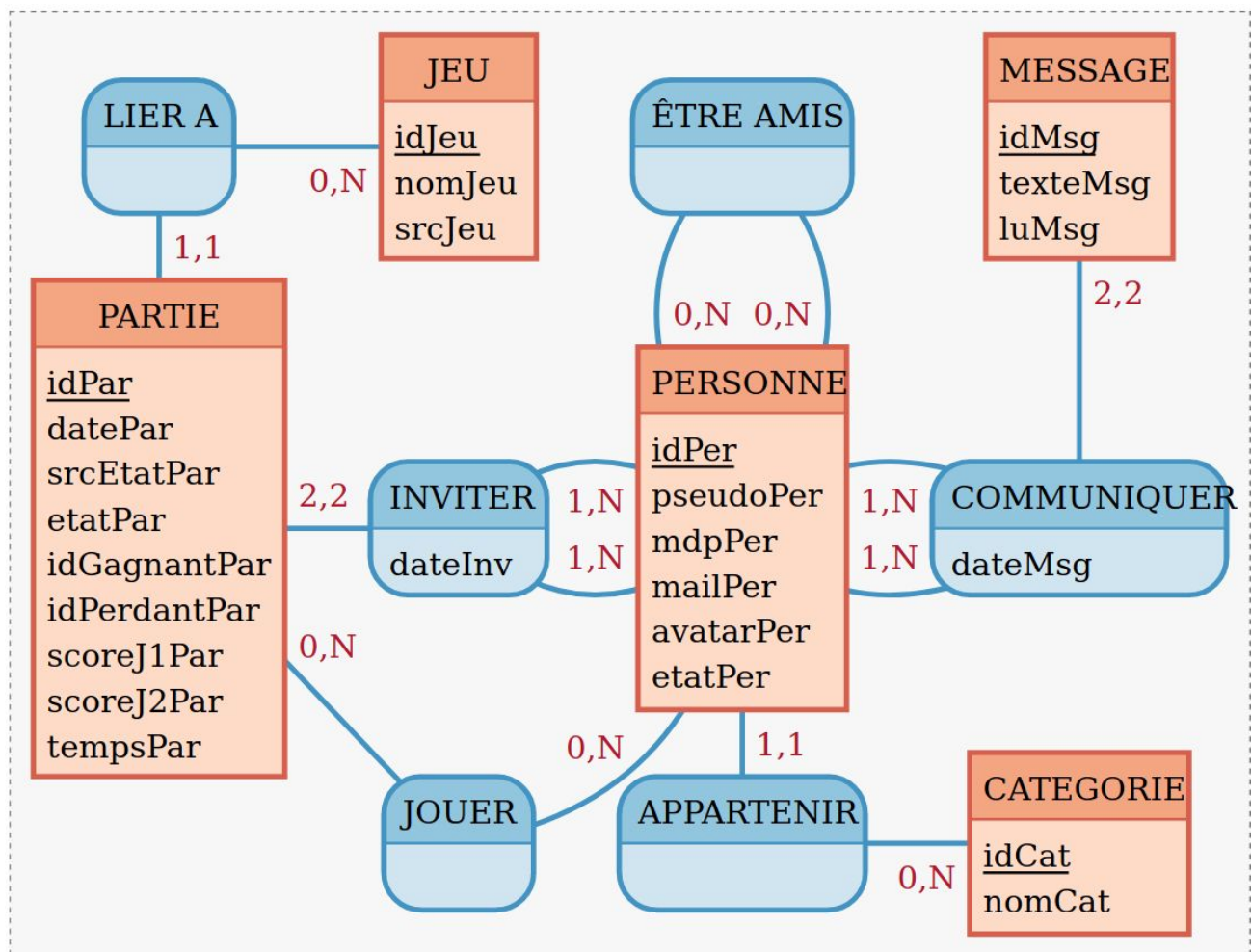


Figure 1 : Modèle Conceptuel des Données original

On nous a ensuite proposé une correction du MCD (figure 2). Le MCD fourni est plus succinct que le nôtre et les relations entre les tables sont plus claires et simples à implanter. En effet, la table JEU que nous avons choisi n'est pas présente ici et une table INVITATION est apparue. La table JEU n'est pas réellement nécessaire mais nous l'avons adopté dans l'optique d'implémenter plusieurs jeux à la plateforme. Tandis que la nouvelle table INVITATION simplifie les requêtes que nous pourrions faire.

C'est pour cela que nous avons décidé de se baser sur ce MCD pour programmer notre plateforme de jeu.

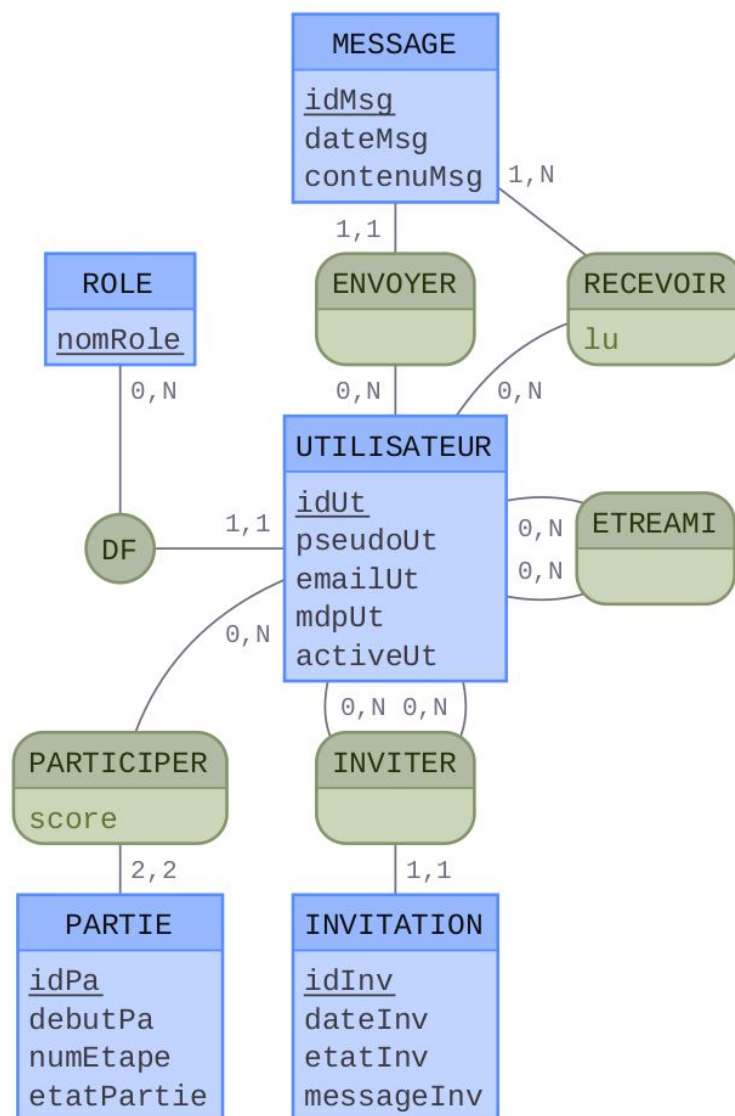


Figure 2 : Modèle Conceptuel des Données corrigé

3. Modèle Logique de Données

Après la réalisation du MCD, il nous faut synthétiser les tables en figurant notamment leurs identifiants soulignés et les clés étrangères précédées d'un '#'. Là encore, nous avons utilisé les scripts de *Mocodo online* en leur ajoutant des styles, on remarquera que notre MLD est constitué de 9 tables ainsi que 15 clés primaires et 12 clés étrangères. Chaque table est reliée à au moins une autre grâce à ces clés.

Nous avons réalisé cette modélisation pour mieux comprendre et analyser la base de données afin de mieux la programmer.

```
JEU ( idJeu, nomJeu, srcJeu )
ÊTRE AMIS ( #idPer, #idAmis)
MESSAGE ( idMsg, texteMsg, luMsg )
PARTIE ( idPar, datePar, srcEtatPar, etatPar, idGagnantPar, idPerdantPar,
scoreJ1Par, scoreJ2Par, tempsPar, #idJeu )
INVITER ( #idPer, #idDest, #idPar, dateInv )
PERSONNE ( idPer, pseudoPer, mdpPer, mailPer, avatarPer, etatPer, #idCat )
COMMUNIQUER ( #idPer, #idDest, #idMsg, dateMsg )
JOUER ( #idPer, #idPar )
CATEGORIE ( idCat, nomCat )
```

Figure 3 : Modèle Logique de Données

4. Les requêtes

Pour vérifier que la base de données que nous avons implémenté est un minimum cohérente et permet de faire quelques requêtes basiques, nous avons écrit les requêtes demandées par les professeurs par rapport à notre MCD original.

```
select idMsg, texteMsg
from (MESSAGE natural join COMMUNIQUER), PERSONNE
where iddest = PERSONNE.idper and pseudoper="iuto" and lumsg=false;
```

La requête ci-dessus nous permet de **renvoyer une liste des messages qui n'ont pas été lus par l'utilisateur iuto**. On remarquera qu'ici on a parenthésé la jointure naturelle entre la table **Message** et la table **Communiquer** afin de récupérer les messages qui ont communiqué à une date précise, puis on ajoutera la table **Personne** pour qu'on puisse vérifier que les messages communiqués n'ont pas été lus et qu'ils appartiennent au joueur "iuto".

```
select idPar
from PARTIE natural join JOUER natural join PERSONNE
where etatPar = "En cours" and pseudoPer = "iuto";
```

La requête ci-dessus permet de **renvoyer la liste des parties en cours de l'utilisateur iuto**. On remarquera qu'ici nous faisons une jointure naturelle entre les tables **Partie** et **Jouer** afin de récupérer les parties jouées, puis nous faisons une jointure naturelle à nouveau pour récupérer les parties jouées par des personnes. On viendra vérifier l'état de la partie à l'aide d'une condition where et également le pseudo de la personne.

```
select count(idPar) as nbPartieGagnée
from JOUER natural join PARTIE natural join PERSONNE P1 natural join
PERSONNE P2 natural join JEU
where P1.pseudoPer= "iuto" and P2.pseudoPer= "iutc" and idgagnantpar =
P1.idPer and nomJeu= "Puissance 4";
```

La requête ci-dessus permet de **renvoyer le nombre de parties gagnées par le joueur iuto contre le joueur iutc pour le jeu Puissance 4**. On remarquera qu'ici nous utilisons un agrégat count() afin de compter l'occurrence de idPar qui représentera les parties. On viendra renommer l'information sous la forme nbPartieGagnée. On viendra réaliser une jointure naturelle entre la table Jouer et la table **Partie** afin de récupérer les parties qui ont été jouées, puis on ajoutera l'information que nous voulons 2 personnes dans notre requête. Pour cela nous avons effectué un renommage. Enfin nous vérifions que le jeu est bien le Puissance 4 et que le gagnant est bien la personne possédant le pseudo "iuto" et tout ça grâce à une condition where.

IV. Analyse de l'application et des modules

Nous avons utilisé pour l'analyse de l'application le logiciel *StarUML* afin de modéliser un diagramme des cas d'utilisations. Nous avons choisi cet outil car il est spécialement dédié à la modélisation de diagrammes respectant les conventions UML auxquelles nous avons préalablement été formé.

1. Module joueur et administrateur

Le module Joueur et Administrateur se concentre sur les interactions que peut avoir l'utilisateur avec le jeu mais également le Serveur qui est un acteur secondaire, tous les cas d'utilisations sont des extensions d'un cas d'utilisation commun à tous qui est Connexion, cela veut dire que l'utilisateur doit se connecter pour pouvoir interagir avec la plateforme et donc le serveur.

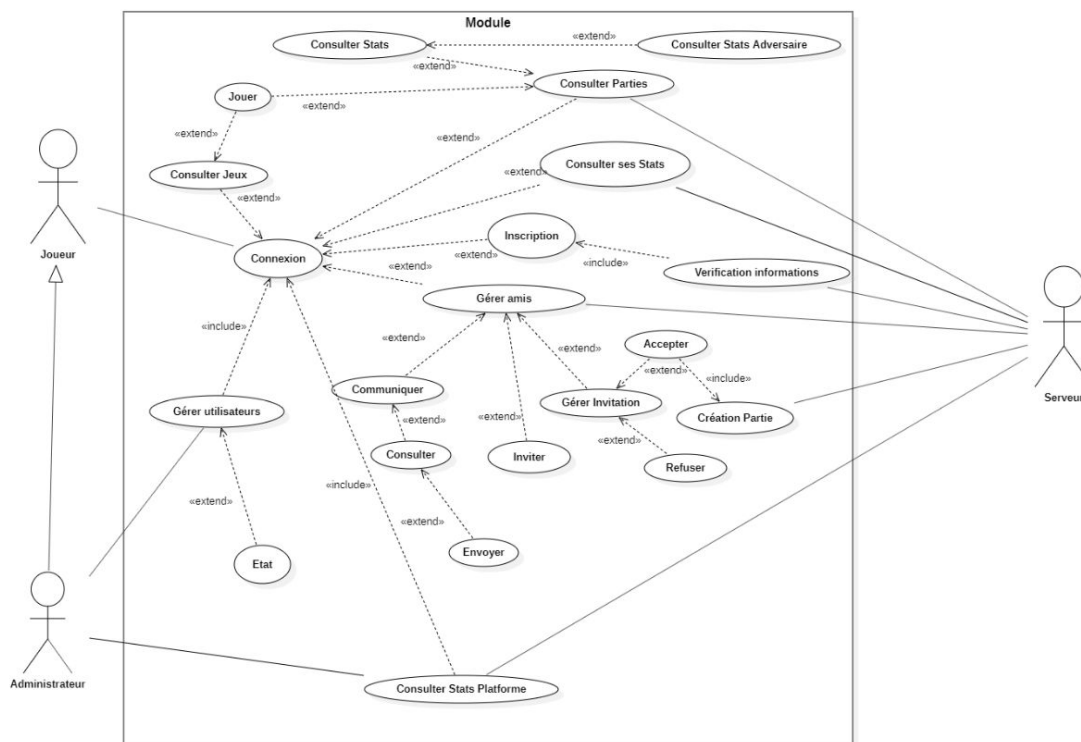


Figure 4 : Diagramme des cas d'utilisation du module Joueur et Administrateur

On peut constater dans l'exemple ci-dessus, que le joueur aura accès aux fonctionnalités de l'application qu'après s'être connecté.

Ensuite, à partir du diagramme des cas d'utilisations, nous avons produit une description textuelle à l'aide de l'outil *Google Doc*, un éditeur de texte, dans le but d'apporter plus de précision dans notre analyse et d'être davantage explicite dans notre rapport.

Cela nous permet d'avoir une idée claire et précise du déroulement des fonctionnalités de l'application. Mais aussi de la façon dont nous pourrions les intégrer lors de la réalisation et la programmation du projet. En effet, dans l'exemple ci-dessous, on peut voir l'enchaînement d'actions prévues pour le système et l'utilisateur lorsqu'il se connecte à la plateforme dans un scénario nominal.

CAS N°1
<p>Nom : Se connecter</p> <p>Acteur(s) : Joueur (et Administrateur)</p> <p>Description : Pour utiliser les différentes fonctionnalités de la plateforme, l'utilisateur doit s'authentifier.</p> <p>Pré-conditions : L'utilisateur doit s'être inscrit sur la plateforme.</p> <p>Démarrage : L'utilisateur à lancer l'application.</p>
DESCRIPTION
<p>Scénario nominal :</p> <ol style="list-style-type: none"> 1. Le système affiche la fenêtre de connexion. 2. L'utilisateur peut remplir les champs correspondant à son pseudonyme et son mot de passe avant de valider. 3. Le système vérifie l'authenticité des informations et si le compte est bloqué. 4. Le système connecte l'utilisateur à la plateforme. 5. Le système affiche la fenêtre de l'accueil. <p>Scénarios alternatifs :</p> <p>2.a. L'utilisateur décide de s'inscrire sur la plateforme.</p> <p>Scénarios d'exception :</p> <p>3.a. Les informations ne correspondent pas à la base de données ou le compte à été bloqué par un administrateur. Revenant ainsi à l'étape 2 du scénario nominal.</p> <p>Fin : Scénario nominale : 2. (sur décision de l'utilisateur), 5. (par le système)</p>
COMPLÉMENTS
<p>Performance attendue :</p> <p>La vérification des données et la connexion doit se faire en moins de 10 secondes.</p>

Tableau 2 : Description textuelle d'un cas d'utilisation du module Joueur et Administrateur

Afin d'effectuer le diagramme de séquence du scénario nominal appartenant à la partie de l'analyse de l'application, nous avons également utilisé le logiciel *StarUML* pour les mêmes raisons que ci-dessus. En voici un exemple :

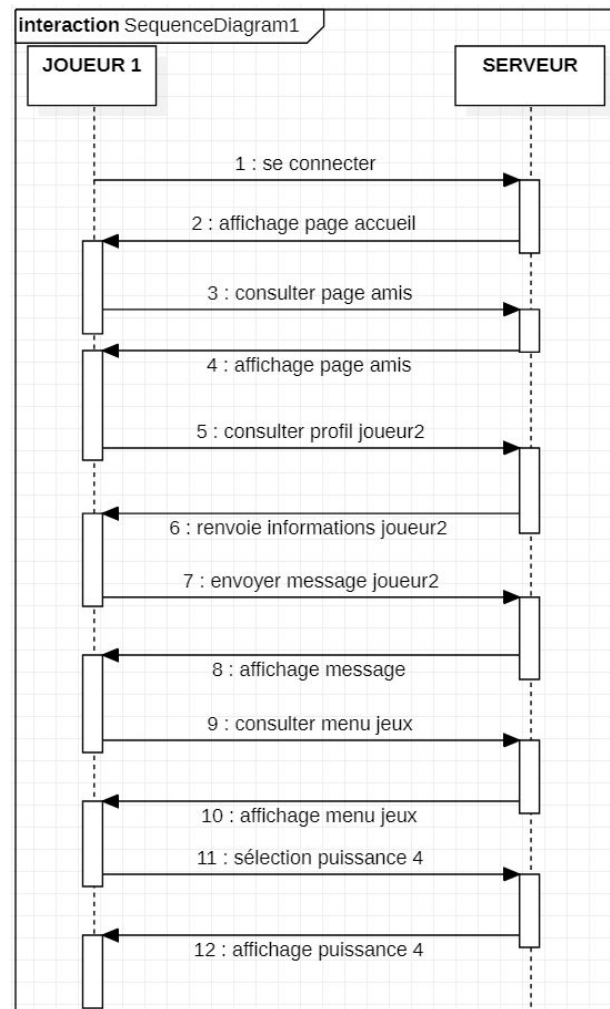


Figure 5 : Diagramme de séquence du scénario nominal du module Joueur et Administrateur

On peut constater dans l'exemple ci-dessus, que pour avoir accès aux différentes fonctionnalités l'étape "se connecter" est obligatoire. Ensuite, chaque action de Joueur1 implique un message synchrone entre Joueur1 et le Serveur.

2. Module Puissance 4

Dans ce module, la partie concernée est une fois que le Joueur est dans une partie de Puissance 4.

Nous avons également fait un diagramme des cas d'utilisation pour le module Puissance 4.

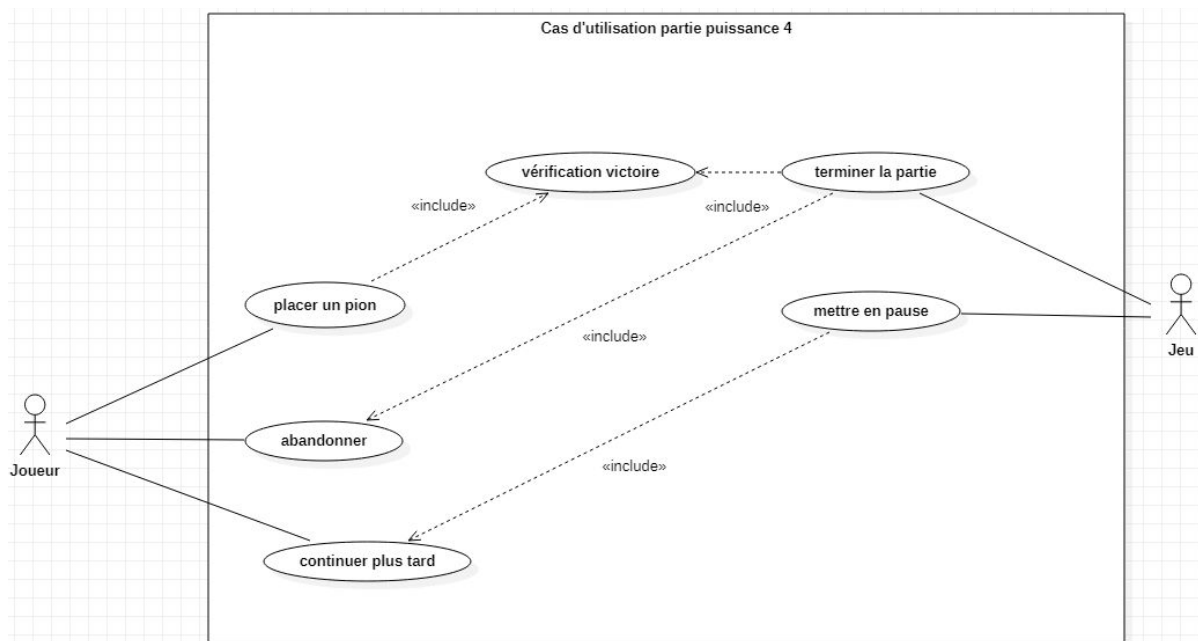


Figure 6 : Diagramme des cas d'utilisations du module Puissance 4

On peut voir qu'il est moins fournis que le premier, le Joueur ne dispose que de 3 choix possibles : placer un pion, abandonner la partie ou la mettre en pause. Chacun de ces choix a un effet différent sur la partie.

CAS N°1
Nom : Placer un pion Acteur(s) : Joueur Description : Le joueur doit pouvoir placer un pion sur le cadrillage de jeu.
Pré-conditions : Il faut que ce soit le tour du joueur Démarrage : L'utilisateur joue son tour
DESCRIPTION
Scénario nominal : <ol style="list-style-type: none"> 1. Le système indique le tour du joueur 2. Le système met en surbrillance les colonnes disponibles pour poser un pion 3. Le joueur sélectionne une colonne 4. Le système ajoute un pion au niveau de la colonne sélectionné à la première ligne en partant du haut 5. Le système joue une animation où le pion tombe jusqu'à ce qu'il collisionne avec un autre pion ou le bas du cadrillage
Scénarios alternatifs : 2.a. L'utilisateur ne pose pas de pion avant la fin du temps imparti et une colonne est sélectionné au hasard
Scénarios d'exception : 3.a. Aucune colonne n'est disponible, la partie se termine
Fin : Scénario nominale : 5. (par le système)
COMPLÉMENTS
Performance attendue : L'animation des pions ne doivent durer qu'une demie seconde.

Tableau 3 : Premier cas de la description textuelle du module Puissance 4

Pour la description textuelle du module Puissance 4 dont on peut voir un exemple ci-dessus avec le tableau 3, il était question d'apporter plus de précision quant au diagramme des cas d'utilisation. On peut la généraliser par le fait que nous devons gérer une partie de puissance 4 avec les fonctionnalités suivante : placer un pion, vérifier une victoire, terminer/abandonner une partie, la mettre en pause et continuer plus tard.

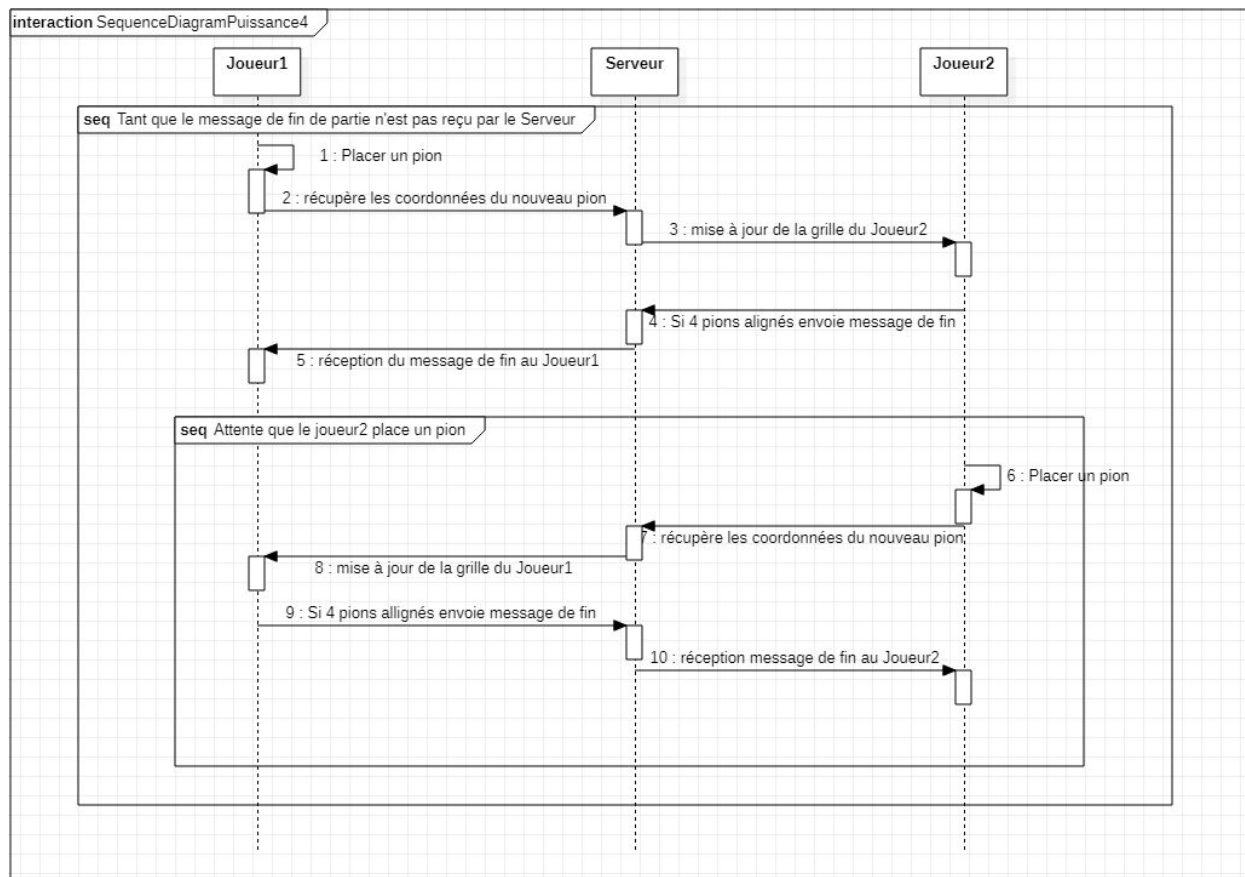


Figure 7 : Diagramme de séquence du scénario nominal du module Puissance 4

Pour le diagramme de séquence du module puissance 4 ci-dessus, nous avons identifié 3 différents acteurs afin de faciliter la création ainsi que la compréhension du diagramme : Le **Joueur1**, le **Serveur** et le **Joueur2**.

La partie ne s'arrête pas tant que le signal de fin de partie n'est pas reçu par le serveur.

Quand le **Joueur1** place un pion, les coordonnées du pion sont récupérées par le **Serveur** puis l'affichage du **Joueur2** est mis à jour. Le **Serveur** va ensuite vérifier si une ligne s'est formée. Si oui, le message de fin est transmis à l'autre joueur qui a donc gagné et la boucle s'arrête. Si non, on répète cette étape mais cette fois en inversant les rôles des deux joueurs c'est à dire que c'est maintenant le **Joueur2** qui place un pion.

V. Maquettage des modules

Cette partie du projet est dédiée à la conception de l'IHM de notre projet. Elle comprend une analyse UML (diagramme des classes), le modèle des tâches ainsi que le maquettage de l'application à partir de notre analyse des modules. Cela nous permettra d'avoir une vision de l'apparence qu'aura notre projet et de nous guider lors de sa réalisation.

1. Diagramme de classe des modules

Pour commencer, nous avons utilisé le logiciel *StarUML* pour faire le diagramme des classes. L'objectif est de modéliser au plus détaillé l'ensemble des modules constituant notre projet ainsi que leurs relations.

Comme nous pouvons le constater dans la figure ci-dessous, nous avons mis en place une classe commune pour chaque page afin d'éviter la redondance des attributs et des méthodes. Nous nous sommes par ailleurs abstenus d'ajouter les getters et setters pour gagner en visibilité.

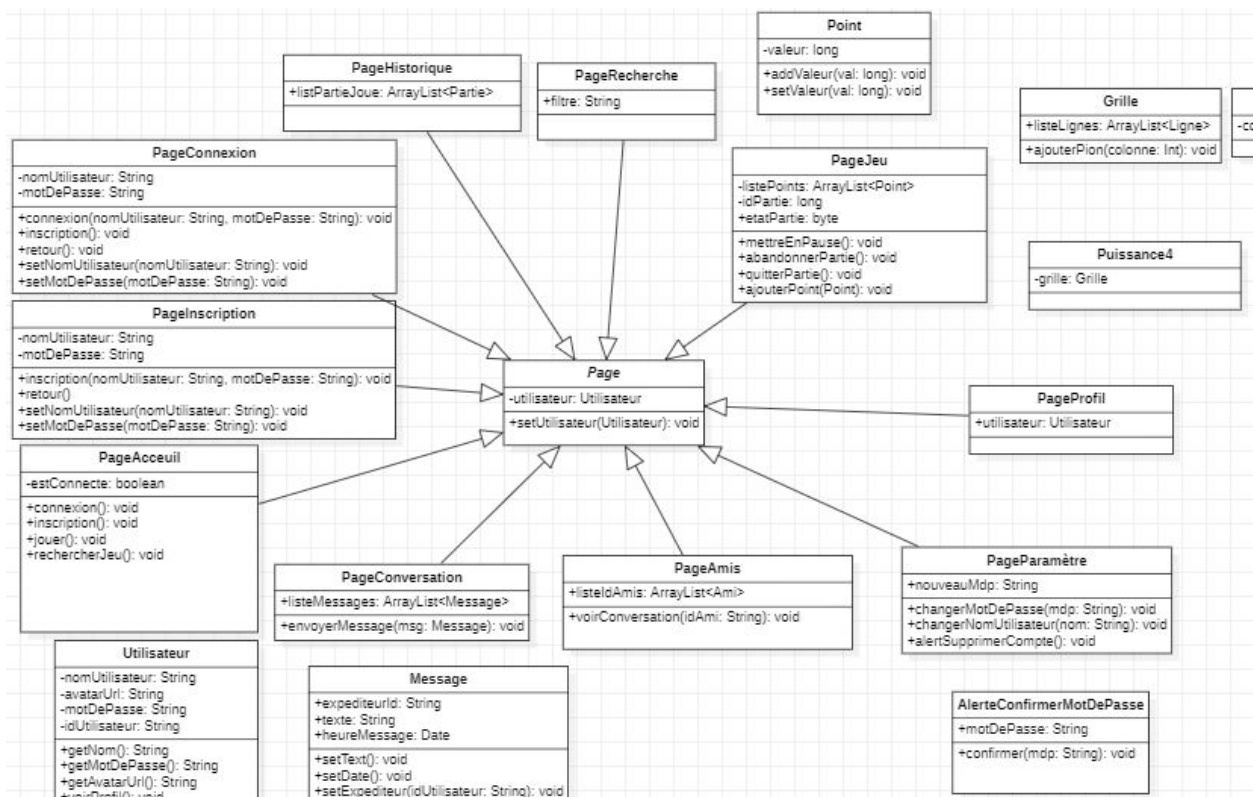


Figure 8 : Extrait du diagramme des classes (pour gagner en visibilité)

2. Modèle des tâches

Pour nous aider à réaliser les maquettes du projet, nous avons fait le modèle des tâches. Cela nous permet de décomposer en sous-tâches les tâches que nous avons défini dans notre analyse des modules et leur ordonnancement dans le temps pour atteindre le but souhaité.

Pour cela, nous avons utilisé *diagrams.net*, un outil en ligne pour faire rapidement des diagrammes clairs et explicites.

Les noms des cas d'utilisations sont représenté par des cases violettes. Chacune des tâches sont numérotées afin de les distinguer et les hiérarchiser.

Par exemple sur la figure 9 (sur la page suivante), la tâche initiale est "Consulter Jeux" et ses sous-tâches sont "Récupération des informations...", "Affichage de la liste..." et "Sélectionner un jeu" qui a son tour possède des sous-tâches.

Ces sous-tâches peuvent être exécuté de différentes façons. De manière **séquentielle** où elles seront exécutées l'une après l'autre. De manière **alternative** où l'utilisateur aura le choix de les traités dans le désordre. Enfin, de façon **élémentaire** qui indique que la tâche n'a pas de suite et qu'elle se termine. On parlera de scénario final pour la dernière description.

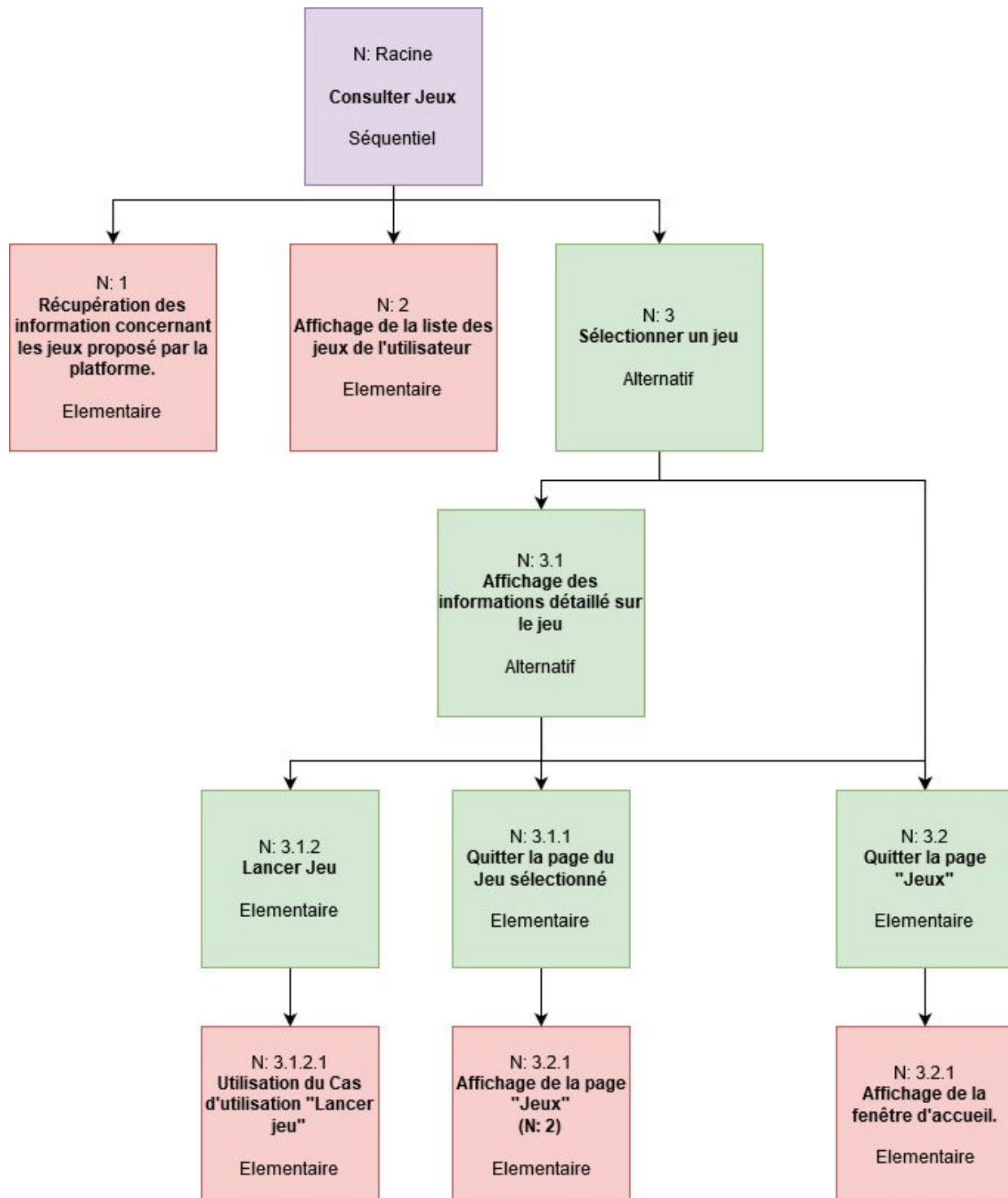


Figure 9 : Modèle des tâches de "Consulter Jeux"

3. Interactions Homme-Machine

Afin de concevoir les maquettes IHM de notre application, nous avons utilisé *figma.com*, un outil de design d'interface collaboratif en ligne. Cela nous a permis de tous travailler en même temps sur les maquettes en voyant les modifications des autres membres en temps réel et de faire des commentaires.

Ces IHM nous permettent d'avoir une vision globale de ce à quoi l'application finale pourra ressembler et de faciliter la réalisation de l'interface avec l'utilisateur.

Nous avons commencé par décider de la charte graphique et de la direction graphique pour l'application. Nous avons donc retenu un design assez plat (Flat Design). La couleur dominante que nous avons choisi est le orange, utilisé pour les onglets sélectionnés et les boutons pour indiquer qu'une action est possible et le violet comme couleur secondaire pour compléter le orange si besoin. Comme couleur de fond, nous avons retenu des nuances de gris avec une écriture blanche afin que l'utilisation et la lecture sur l'application soit le plus agréable possible. Pour réaliser ces maquettes, nous nous sommes inspirés d'applications déjà existantes comme *Discord*, *Blizzard* et *Steam*. Nous avons même choisis comme nom pour notre application *Vapeur* en référence à la plateforme *Steam*.

Après avoir réalisé la première maquette tous ensemble afin de définir la direction graphique que nous allons prendre et garder une cohérence avec toutes les pages. Notamment la barre de navigation qui reste la même pour toutes les pages mais dont le texte de l'onglet sélectionné prend la couleur dominante pour informer l'utilisateur où il se trouve. Le but de l'IHM étant de guider un maximum l'utilisateur pour éviter qu'il se perde sur la plateforme. Nous nous sommes, ensuite, répartis l'ensemble des maquettes.

La maquette ci-dessous représente la page jeux qui sert d'accueil et affiche la bibliothèque de jeux de l'utilisateur. La liste des jeux est à gauche et ils peuvent être sélectionnés afin d'afficher leurs informations et lancer une partie. La miniature du jeu est affichée suivie de son titre et son type. Éventuellement, l'appréciation du jeu avec un nombre d'étoiles. Ensuite, une description du jeu accompagné de captures d'écran.

Finalement, l'utilisateur peut lancer une partie de deux façons. Avec les boutons "Search for a game" et "Invite a friend". Le premier lance une recherche de joueur, jusqu'à ce qu'un autre utilisateur de la plateforme lance lui aussi une recherche pour le Puissance 4. Dans ce cas, le jeu est lancé et une partie est créée entre les deux joueurs. Le fonctionnement du deuxième bouton est expliqué à la page suivante.

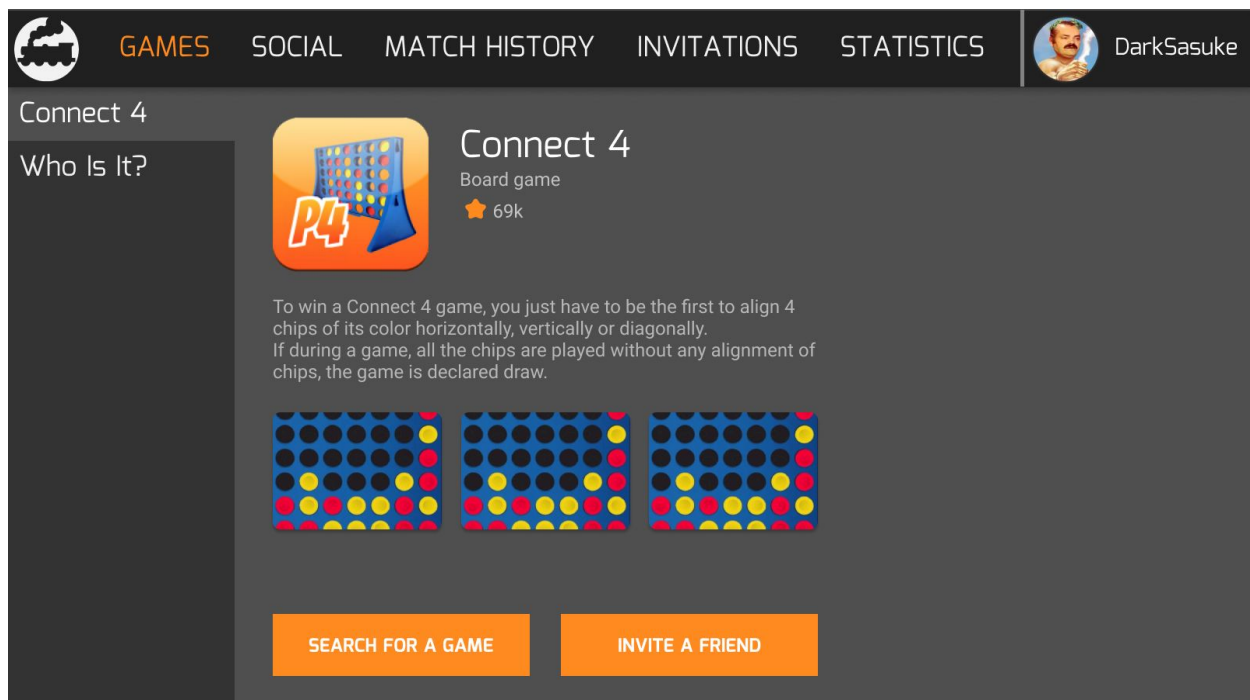


Figure 10 : Maquette de la page des jeux

Le bouton "Invite a friend" permet à l'utilisateur de choisir un autre joueur de dans sa liste d'amis s'affichant à droite de l'écran. Une invitation lui est alors envoyée. S'il accepte cela suivra la même procédure citée précédemment. S'il refuse, l'initiateur de l'invitation recevra une notification lui informant ce refus.

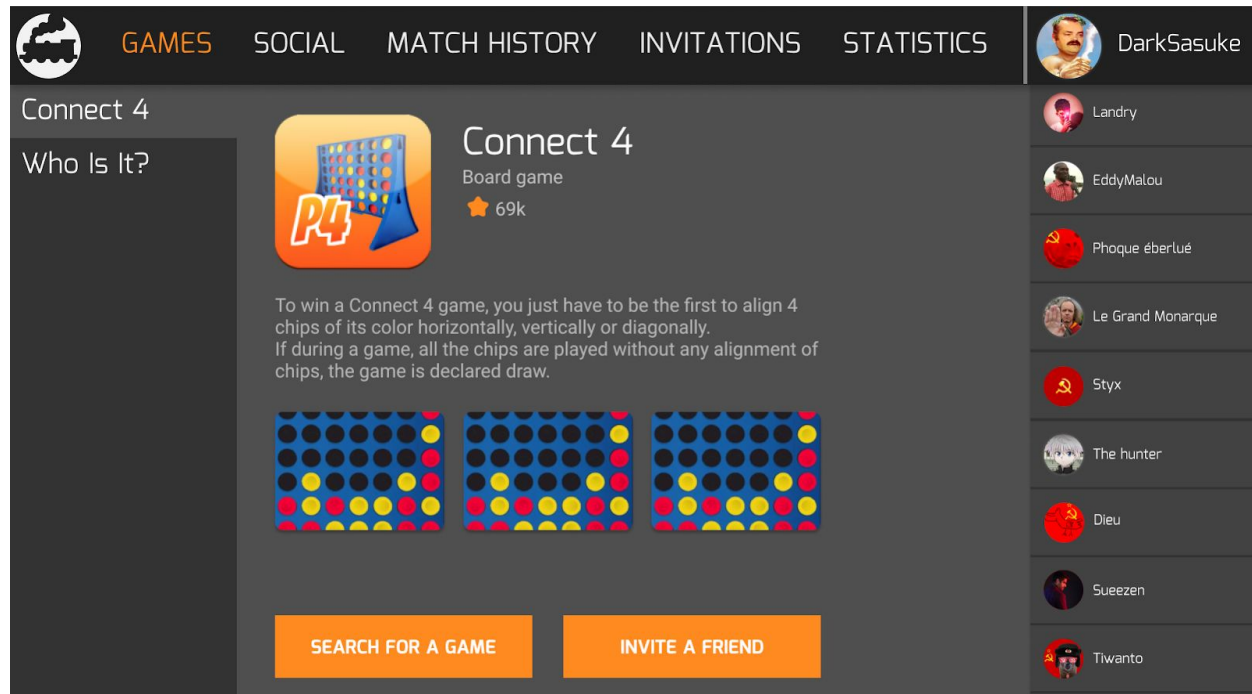


Figure 11 : Maquette puissance 4, invitation de joueur

Concernant l'IHM du module Puissance 4, nous avons choisi de garder un affichage très simple et concis, nous avons donc gardé le nécessaire. On peut voir le nom des deux joueurs avec leur couleur ainsi que le nombre de jetons restant.

Il y a également un bouton pour pouvoir abandonner, et un autre pour proposer de mettre en pause une partie pour la reprendre plus tard. Afin de choisir où placer son jeton, il y a des flèches qui indiquent les colonnes disponibles. Il y a aussi un chronomètre indicateur du temps qui sera sauvegardé dans la base de données afin d'avoir des statistiques relatives à la durée des parties.

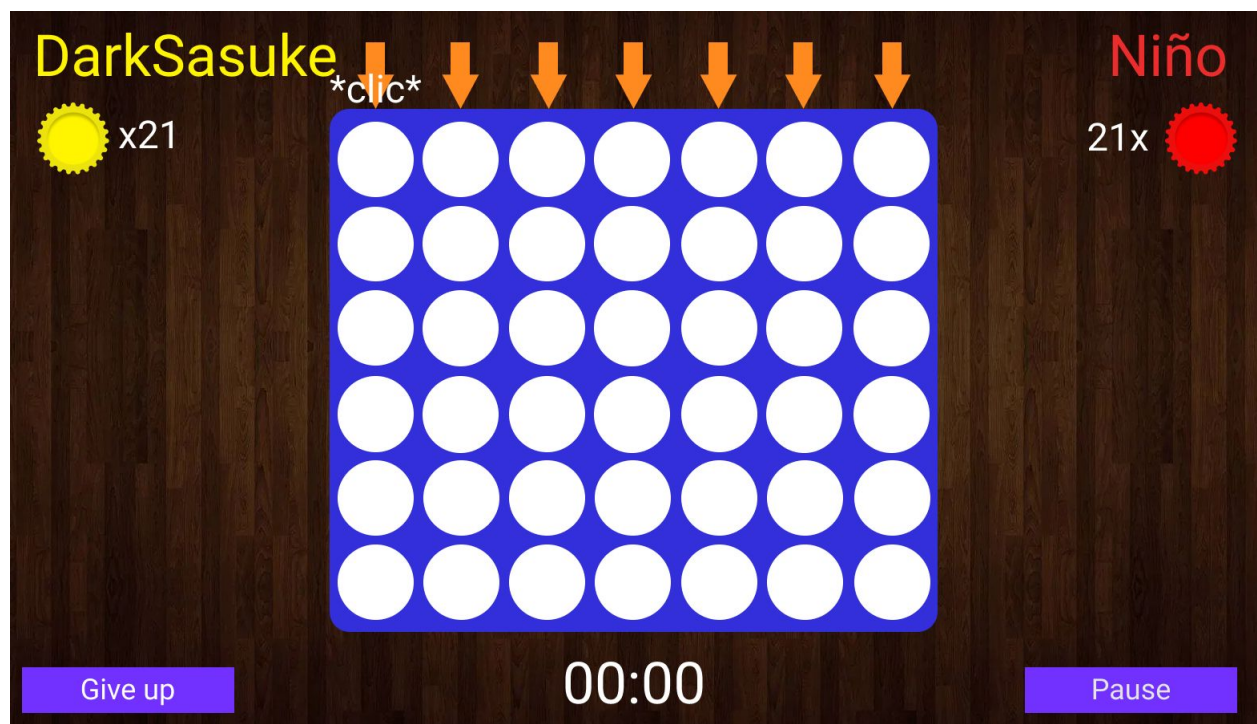


Figure 12 : Maquette Puissance 4, début de la partie

VI. Réalisation

Pour la réalisation du projet, nous devons travailler en commun. Nous avons donc utilisé *GitLab* qui nous permettait de tous y contribuer en parallèle grâce à un système de versionnage. Nous avons répartis le travail en deux parties principales : le module Joueur et Administrateur et le module Puissance 4. Pour chacune d'elle, nous devons réaliser des vues, des modèles, des contrôleurs et des classes qui interagissent avec la base de données.

1. Réalisation du module Joueur et Administrateur

La partie Joueur et Administrateur demandait beaucoup de travail sur les vues et la connection avec la BD. Nous avons donc commencé par là en travaillant en même temps sur les différentes pages de l'application et les classes utilisant la Java DataBase Connectivity (JDBC), une librairie java permettant d'accéder à une BD.

Pour les vues, nous avons créé une classe différente pour chaque page de la plateforme en utilisant la bibliothèque d'interface utilisateur JavaFX. Nous avons fait en sortes que les vues similaires entre elles héritent d'une même classe parent afin d'éviter la redondance de code. Ces classes parents héritent elles même d'un "Pane", c'est-à-dire un élément de JavaFX permettant de contenir des "Control", des éléments visuels, pour les afficher sur une fenêtre d'application.

De cette façon, nous avons deux classes parents : *VueVBoxParent* et *VueBorderPaneParent*. *VueVBoxParent* est la classe parent de *VueConnexion* et *VueInscription* qui gèrent les pages de connexion et d'inscription à l'application. Tandis que *VueBorderPaneParent* est parent de *VueJeux*, *VueSocial*, *VueMatchHistory*, *VueInvitations*, *VueProfil* et *VueAdmin*. Ces classes sont les différents onglets de la plateforme permettant à l'utilisateur d'y naviguer pour voir ses jeux, lancer des parties, avoir une liste d'amis pour jouer et communiquer, voir son l'historique de ses parties, ses invitations à jouer, son profil et ses statistiques.

L'onglet Admin, quant à lui, n'est visible et accessible que par les utilisateurs ayant le rôle administrateur (on rappelle que l'administrateur hérite des méthodes d'utilisateur et que par conséquent il a accès aux mêmes fonctionnalités que les utilisateurs) et il permet de consulter les statistiques globales ainsi que de gérer les joueurs et les parties de la plateforme.

Toutes ces pages sont accessibles via un menu se trouvant en haut de la fenêtre (figure 11) avec la classe *MenuVapeur*. Nous pensions à l'origine créer une vue différente pour le profil et les statistiques mais ces deux pages auraient été bien trop vides. Nous les avons donc rassemblé dans le profil de l'utilisateur et l'onglet "Admin" à pris la place initialement prévue pour l'onglet "Statistics".

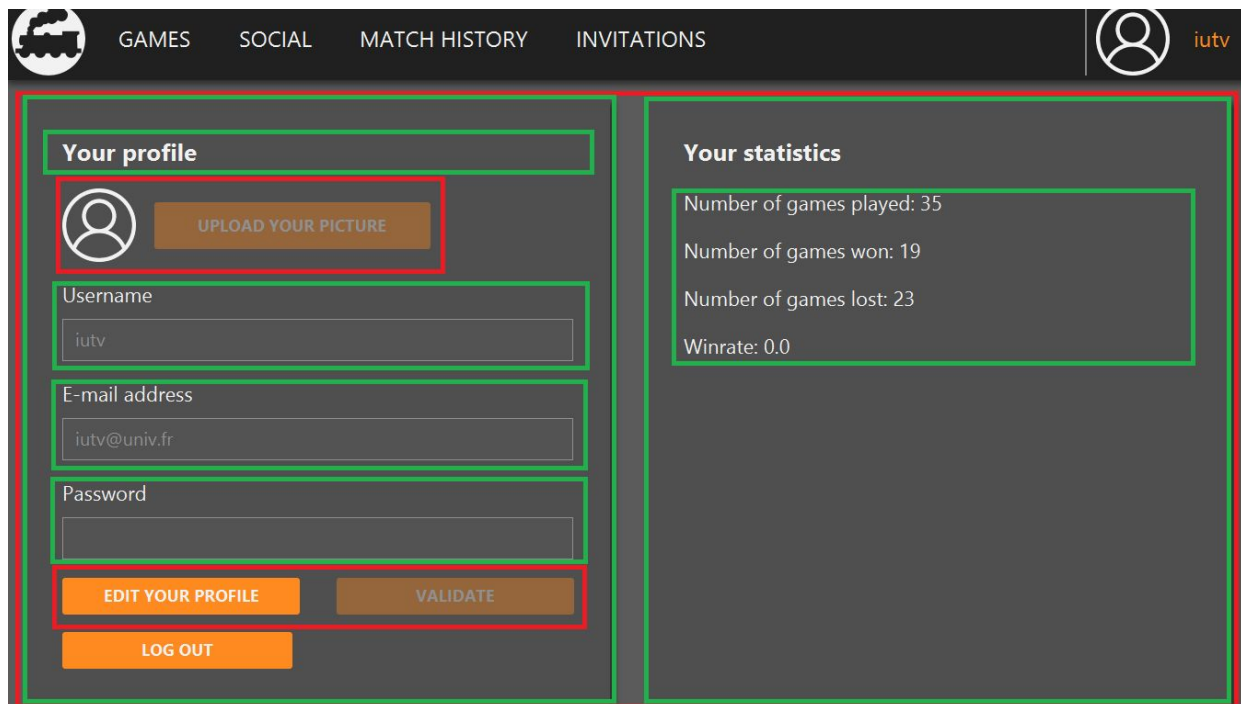


Figure 13 : Capture de la page profil de l'utilisateur "iutv"

Sur la figure 11, on peut remarquer deux types de rectangles. Ceux en vert qui représentent les VBox et ceux en rouge qui représentent les HBox. JavaFX utilise des formes, des conteneurs, etc, afin de disposer les éléments graphiques. Pour mieux comprendre la différence entre le HBox et le VBox vous avez ci-dessous une image expliquant le fonctionnement de ses "boîtes". Le VBox va disposer les éléments en lignes alors que le HBox le fera en colonne comme sur la figure ci-dessus où il y a un grand rectangle rouge avec deux rectangles verts, ces derniers sont disposés en colonne car le rectangle rouge auquel ils appartiennent est un HBox.

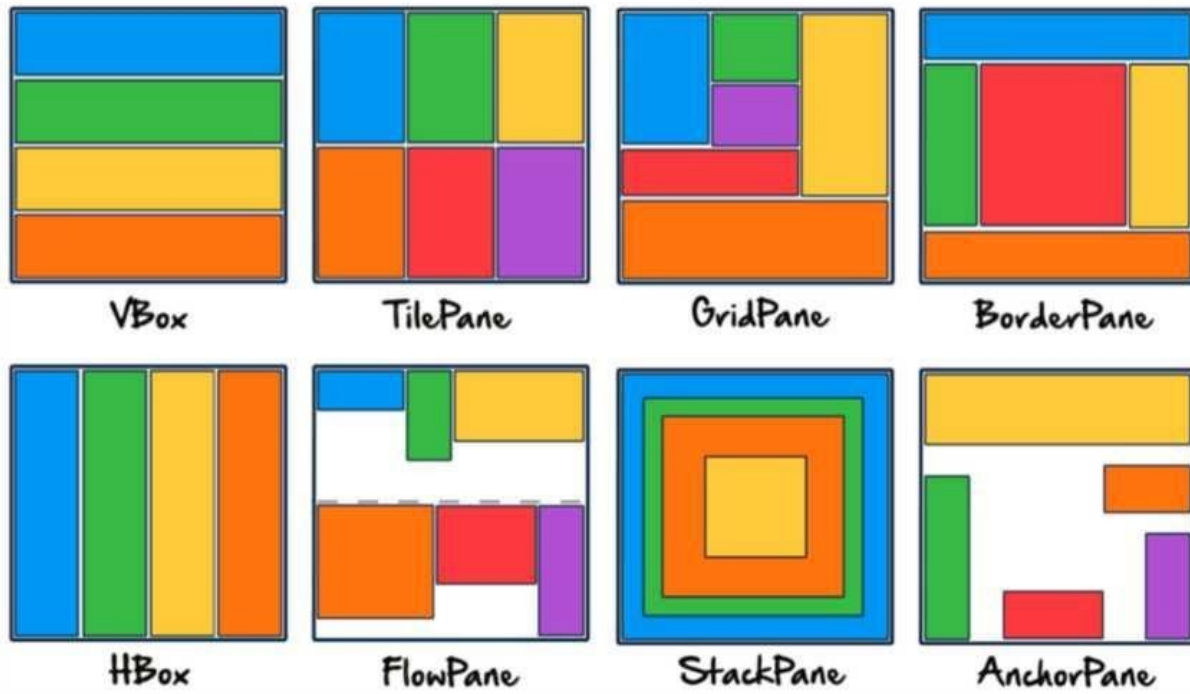


Figure 14 : Schéma des différentes dispositions JavaFX


Pour la JDBC, nous avons décidé de créer des bibliothèques, c'est-à-dire des classes contenant uniquement des méthodes statiques. Cela nous permet de les appeler à n'importe quel endroit de notre application sans pour autant instancier la classe. Pour rendre ces méthodes statiques, il a cependant fallu qu'elle puissent accéder à la connexion SQL, nous avons donc rendu cette connexion statique.

Pour toutes les tables de la BD, nous avons créé des getters, des setters, des requêtes de suppression, ainsi que des méthodes custom dans le but de recueillir des informations spéciales comme par exemple la liste des messages entre deux utilisateurs. Pour pouvoir réaliser ces requêtes, nous avons utilisé les "Statement" et plus particulièrement les "PreparedStatement" qui ont permis de créer des requêtes avec des variables comme sur l'exemple ci-dessous :

```
PreparedStatement ps = laConnexion.prepareStatement("delete from  
ETREAMI where idut_1=? and idut_2=?");  
  
ps.setInt(1, id1);  
ps.setInt(2, id2);  
ps.executeUpdate();
```

Figure 15 : Exemple d'une méthode JDBC

Dans la figure 12, ce sont les points d'interrogation ("?") qui permettent de définir des variables, puis on définit ces dernières avec la méthode "set" suivi du type de variable. On peut donc passer des paramètres aux méthodes JDBC tels que des Id d'utilisateurs comme illustré dans cette requête.



Une fois les classes utilisant la JDBC terminées, nous avons créé les modèles. Ces modèles ont pour objectif de représenter de manière structurée les des objets pertinent afin de pouvoir les manipuler dans les vues ainsi que dans les contrôleurs. Par exemple, le modèle utilisateur permet de stocker toutes les informations d'un utilisateur de la base de donnée précis. On peut donc, soit récupérer un objet utilisateur à partir d'un Id, afin d'avoir ses informations, ou même instancier un nouvel utilisateur nous même puis le fournir à une méthode qui pourra générer un utilisateur. En clair, les modèles servent de stockage d'informations et de méthodes pouvant les modifier, tout en étant pratique à transporter dans le code car il s'agit d'instances.

Concernant les contrôleurs, ils implémentent les actions que l'utilisateur peut effectuer à partir de la vue. Grâce à ces derniers, appuyer sur un bouton provoque l'appel d'un "ActionEvent" menant à une invocation d'une méthode qui effectue différentes procédure en fonction du bouton pressé.

Chaque vue possède ainsi un contrôleur afin d'implémenter chaque bouton de cette même vue. Nous allons voir par la suite que les contrôleur vont également acquérir une tâche clé qui nous as permis de rendre l'application bien plus ergonomique.

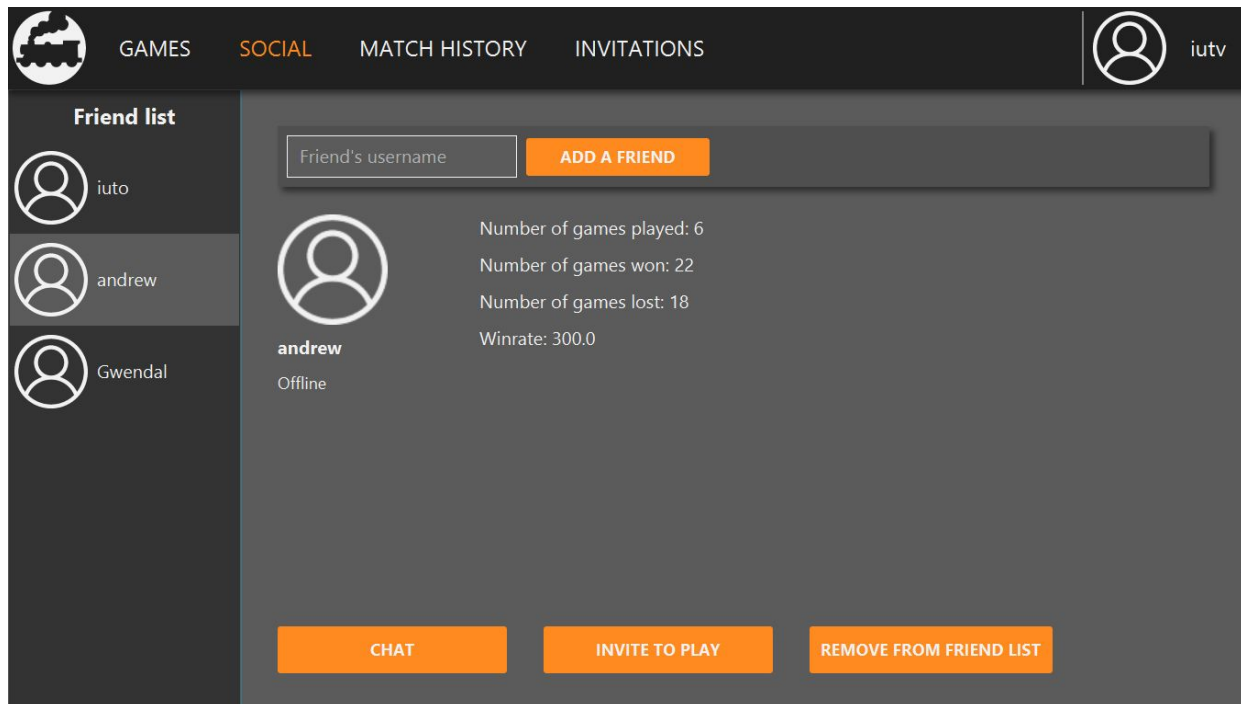


Figure 16 : Exemple d'une sélection dans la liste d'amis

Sur la figure 13, on remarque une liste d'amis sur la gauche lorsque vous sélectionnez un ami, la fenêtre du centre change et vous affiche de nouvelles informations concernant l'ami sélectionné. Ceci est réalisé grâce au contrôleur qui va détecter un événement puis exécuter du code pour détecter quel ami à été sélectionné et enfin pouvoir afficher la fiche d'ami correspondante.

Nous voilà maintenant à la seconde utilisation des contrôleurs. Le problème de l'utilisation précédente étant que pour exécuter du code, il faut que l'utilisateur ait une interaction avec le logiciel. Cela remplit totalement le travail des boutons, mais nous avons en outre besoin d'exécuter du code sous forme de routine afin d'actualiser la page automatiquement pour ne pas que l'utilisateur ait besoin de cliquer sur un bouton refresh manuellement rendant l'application beaucoup moins ergonomique.

Pour pallier à ce problème, nous avons utilisé des “timelines”. Il suffit de les paramétrer (temps entre les routines, code à exécuter, type de cycle...), puis de les lancer pour qu’elles commencent leur routine. Cela nous permet de faire une vérification et de voir si il y a eu des changements dans la BD suivant une intervalle de temps, et dans ce cas de mettre à jour l’affichage de l’utilisateur.

```
private Timeline refresher(ContrôleurMatchHistory cmh){
    Timeline timeline = new Timeline(
        new KeyFrame(Duration.seconds(0),
            actionEvent -> {
                // la vue est mise à jour
                cmh.vueMatchHistory.miseAJourAffichage();
            }
        ),
        // on laisse un délai de 5 secondes entre les mises à jour
        new KeyFrame(Duration.seconds(5))
    );
    timeline.setCycleCount(Animation.INDEFINITE);
    // on retourne la timeline pour la démarrer au bon moment
    return timeline;
}
```

Figure 17 : Exemple d’une méthode refresher

Les timelines ont donc rendu l’application plus facile, rapide et ergonomique à utiliser. Les utilisateurs peuvent communiquer instantanément entre eux et voir les invitations, l’historique des parties et leurs statistiques se mettre à jour en temps réel. De plus, afin d’éviter de surcharger le processeur, les requêtes à la bd et d’éviter d’avoir plusieurs timelines qui fonctionnent en même temps, toutes les timelines sont stoppées quand on quitte un onglet, et celles du nouvel onglet se mettent en route.

2. Réalisation du module Puissance 4

La partie du développement sur le Puissance 4 était beaucoup plus axée sur le modèle et les calculs en arrière plan de l'application. En effet il fallait avoir une représentation virtuelle de la grille d'un Puissance 4 et modifier l'interface utilisateur en conséquence.

Pour la vue de ce module, nous avons choisi de rester simple. La vue est composé d'une seule classe *VuePuissance4* héritant d'un *BorderPane*, c'est-à-dire un affichage qui nous permet d'ajouter des éléments visuels au niveau des quatre bords et du centre de la fenêtre comme sur la figure ci-dessous.

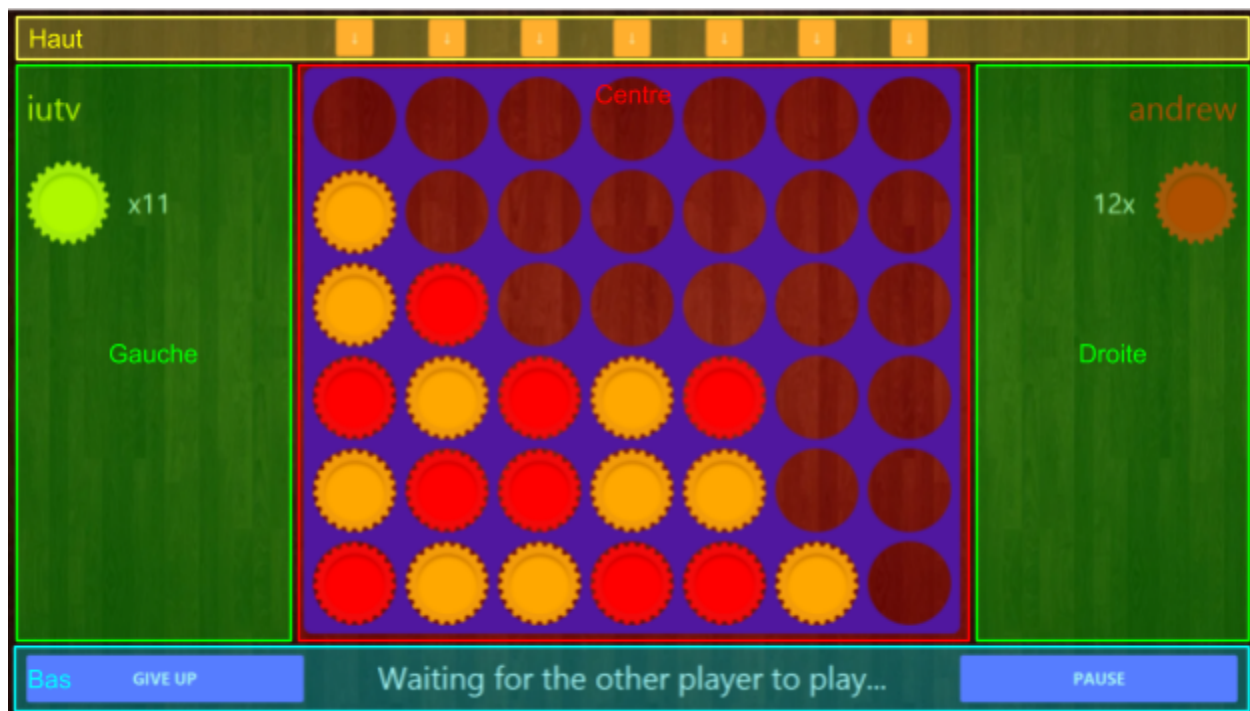


Figure 18 : Schéma d'une partie de Puissance 4 en cours

En haut de la fenêtre, des boutons indiquant les colonnes où les joueurs peuvent placer leurs jetons. En bas, on peut voir des boutons pour déclarer forfait ou mettre le jeu en pause et un texte indiquant quand c'est à votre tour de jouer. À droite et à gauche, il y a les pseudonymes des deux joueurs et le nombre de jetons qu'il leur reste. Finalement, au centre de la fenêtre, une grille à 7 colonnes et 6 lignes permet de placer les jetons grâce au contrôleur du Puissance 4.

Pour avoir un puissance 4 fonctionnel, le plus important c'est le modèle avec le code java qui va derrière. Pour le réaliser, nous avons choisi de faire 3 classes : Grille, Colonne et Pion. la Grille étant une liste le 7 Colonnes étant elles même des listes de 6 Pions. Pour définir ces classes, nous nous sommes aidé d'un diagramme des classes que nous avons réalisé. Chaque joueur à une instance propre qui est défini par sa couleur.

Pour la conception du contrôleur du puissance 4, nous avons établi un attribut de type booléen afin de déterminer si c'est au tour du joueur ou non, ainsi qu'un autre attribut booléen indiquant s'il s'agit du joueur 1 ou du joueur 2. Par exemple une fois que le joueur clique sur un bouton pour ajouter un jeton, cet attribut passe à false l'empêchant d'ajouter des jetons supplémentaire et indiquant que c'est au tour du joueur adverse. De plus le numéro d'étape de la partie est augmenté de 1 afin de pouvoir communiquer un changement de tour à l'autre joueur.

Après avoir joué son tour une Timeline va appeler des méthodes de la JDBC Partie afin de voir si le numéro de l'étape de la partie est pair ou impair, en effet, le joueur 1 va toujours jouer à une étape paire car le numéro d'étape commence à zéro. Donc admettons que nous sommes le joueur 1, dès que la Timeline va voir que l'étape de la partie va passer à un nombre pair, l'attribut *isYourTurn* va passer à True, la grille va se refresh pour acquérir le coup de l'autre joueur, et vous allez pouvoir à nouveau jouer.

```
if (numEtape % 2 == 0) { //si le num étape est pair, alors c'est au tour
du joueur1
    try {
        cp.grille.updateGrille(PartieJDBC.getEtatPartie(cp.partyId));
        cp.vuePuissance4.updateGrille(cp.grille.getGrille());
        this.vuePuissance4.changerNombreJeton2(cp.grille.getNbPionJ());
    } catch (SQLException e) {
        e.printStackTrace();
    }
    cp.isYourTurn = true;
}
```

Figure 19 : Détection de tour du joueur 1

3. Manuel utilisateur

La réalisation du projet s'accompagne de la rédaction d'un manuel utilisateur. En effet, cela permet d'y inscrire toutes les fonctionnalités de l'application pour qu'un utilisateur qui ne connaît pas notre plateforme puisse s'informer et comprendre toutes les possibilités qui s'offrent à lui.

Comme nous ne savions pas si nous aurions le temps d'implémenter toutes fonctionnalités que nous avions prévus avant de commencer la réalisation, nous avons décidé de le rédiger après que l'application soit terminée. Cela nous a permis de le faire sans perdre de temps car nous connaissions déjà toutes les actions que les utilisateurs peuvent effectuer sur l'application.

Nous avons séparé chaque page de Vapeur en une partie du manuel. Ensuite, le travail a été réparti entre les membres du groupes selon la partie sur laquelle il avait travaillé. De cette façon, nous avons tous pu expliciter les fonctionnalités disponibles du mieux possibles.

VII. Gestion du projet

1. Matrice RACI

Suite à l'analyse des modules **Joueurs et Administrateurs** et **Puissance 4**, nous avons créé une **matrice RACI** (Responsable, Acteur, Consulté et Informé). Cela nous permettra de savoir qui travaille sur quelle partie du projet ainsi que de mieux répartir les responsabilités de chaque personnes au sein du groupe.

Dans ce but nous avons utilisé un tableur, plus précisément *LibreOffice Calc*, auquel nous avons programmé des macros permettant d'ajouter des tâches et des groupes de tâches en appuyant simplement sur un bouton. Nous avons aussi modifié le visuel afin d'aisément repérer les différents rôles attribués aux membres du groupe.

Nous avons donc abouti à la matrice ci-dessous le 27 avril où, pour chaque tâche, les membres du groupe se voient attribuer un rôle.

		Simon	Maxime	Gwendal	Andrew	Nino	Landry
Base de données							
	Dictionnaire de données	A1	A2	R3	A3	A1	R2
	MCD / MLD	C2	C2	R3	C3	C2	A3
	Réalisation de la BD	C2	I2	A2	C1	I2	R2
	Test unitaire (requêtes)	I1	R2	C1	I1	R2	I1
Analyse de l'application et des modules							
	Analyse module Joueur & Administrateur	R2	A1	R3	I1	I1	I1
	Analyse module Puissance 4	I1	I1	I1	A1	R2	A
	Planning prévisionnel	R2	C1	A1	C1	C1	C1
	Répartition des tâches	C1	C1	A2	R2	C1	C1
Maquettage / IHM							
	Analyse UML	I	R	I	C	R	A
	Modèle des tâches	A	I	R	A	I	I
	IHM (maquettes)	I	I	A	R	A	I
	État d'avancement du projet	C	C	A	R	C	C
Développement							
	Finalisation analyse	A	R	A	A	R	A
	Développement module Joueur & Administrateur	A	A	R	I	I	I
	Développement module Puissance 4	I	I	I	R	A	A
	Intégration	R	A	C	C	A	R
	Tests	C	A	C	R	A	C
	Corrections erreurs / bug	R	I	A	A	I	R
	Manuel utilisateur	I	R	C	C	A	I

Tableau 4 : Matrice RACI initiale du projet (26/04)

Les **responsables** ("R") sont les personnes qui doivent s'assurer que leur tâches soit fonctionnelle. Les **acteurs** ("A") sont les membres qui réalisent la tâche. Les **consultés** ("C") sont sollicités afin d'avoir leur avis sur la tâche à réaliser. Les **informés** ("I") sont uniquement notifiés de l'avancé de la tâche mais n'ont aucune influence dessus.

Le rôle **responsables** étant celui le plus haut, il hérite de tous les autres rôles en dessous, de même pour les autres rôles à l'exception de **informés**.

2. Diagramme de GANTT

L'outil de planification **GANTT** initialement utilisé pour planifier notre projet nous a permis de voir en grand. En effet, après que la **matrice RACI** est été réalisé, nous avons effectué le GANTT avec le logiciel *GANTT Project*, cherchant à estimer les tâches planifiées en jours/hommes. Toujours dans le but de planifier, l'outil de GANTT nous a permis de répartir les tâches dans le temps. On voit qu'il est important dans l'organisation du travail car cela permet d'optimiser le temps ainsi que d'exprimer clairement les délais à ne pas dépasser.

Il faut remarquer qu'on a divisé les tâches en 4 sections représentant les phases principales du projet qui sont :

1. **Base de données**
2. **Analyse de l'application et des modules**
3. **Maquettage / IHM**
4. **Développement**

On peut également noter que la section **Base de données** et **Analyse de l'application et des modules** commencent en même temps. De plus certaines tâches apparaissent dans la matrice RACI mais pas dans le diagramme de GANTT ci-dessous, car les tâches en question ne concernent pas réellement des tâches planifiées vu qu'il s'agit du déroulement du projet en général, comme **Etat d'avancement** du projet ou **Planning prévisionnel**.

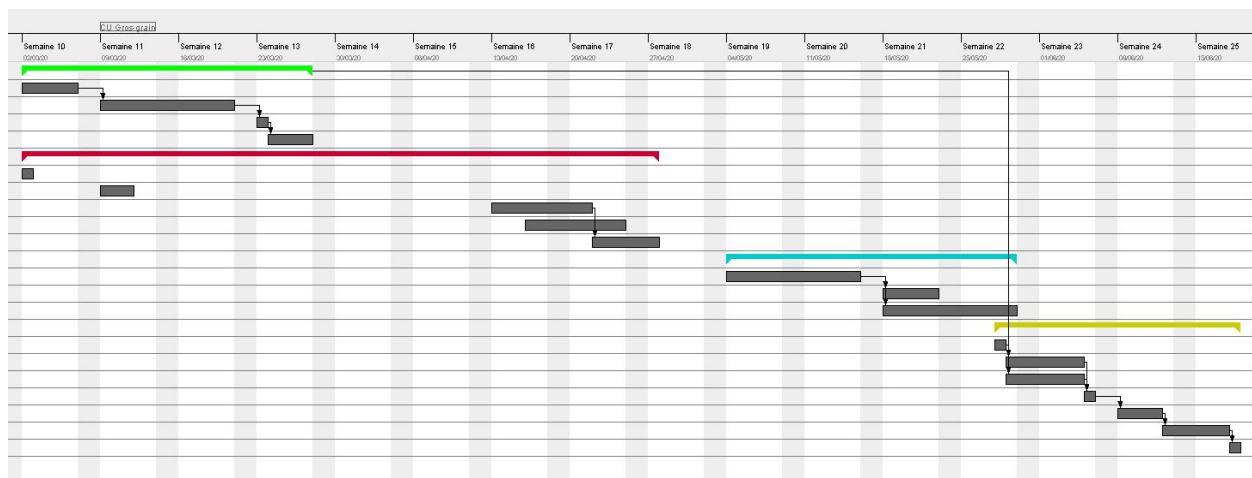


Figure 20 : Diagramme de GANTT du projet

3. Etat d'avancement du projet

a. Etat d'avancement au 01/06

Lors de la phase 3 du projet, nous devons travailler sur l'analyse UML et le modèle des tâches des modules ainsi que les maquettes IHM. Nous avons prévu séparer le groupe en deux pour réaliser l'analyse UML et le modèle des tâches en parallèle. Puis réaliser les maquettes pour ceux qui aurait fini.

Mais, le 1^{er} juin en faisant l'état d'avancement, nous avons remarqué que l'analyse UML et le modèle des tâches ont avancé au même rythme. L'analyse était moins complexe à réaliser que prévu. Ces deux parties étant terminées aujourd'hui, nous avons revu la répartition des tâches de la matrice RACI, comme on peut le voir sur la figure ci-dessous. Tous les membres du groupe seront acteurs de la réalisation de l'IHM. Cela nous permettra de travailler en même temps dessus et se mettre d'accord sur la direction graphique.

		Simon	Maxime	Gwendal	Andrew	Nino	Landry
Maquettage / IHM							
	Analyse UML	I1	R3	I1	C1	R3	A4
	Modèle des tâches	A3	I1	R3	A3	I1	I1
	IHM (maquettes)	A4	A4	A4	R5	A4	A3
	Etat d'avancement du projet	C1	C1	A1	R1	C1	C1

Tableau 5 : Mise à jour de la matrice RACI

Nous avons mis en place l'outil que nous allons utiliser pour réaliser les maquettes, *figma.com*. Grâce à ça nous pourront les finir à temps avant le troisième dépôt du 06/06.

b. Etat d'avancement au 19/06

Lors de la dernière du projet nous avons dû travailler sur le développement de l'application mais également le conception du Manuel d'utilisateur, les tests et correction de bug, intégration, etc. Pour cela nous avons utilisé ce que le modèle RACI nous avait donner avec les groupes qui avait créer les modules :

- Module Joueur / Administrateur : Simon, Gwendal, Andrew
- Module Puissance 4 : Nino, Maxime et Landry

Développement							
Finalisation analyse	A	R	A	A	R	A	
Développement module Joueur & Administrateur	A	A	R	I	I	I	
Développement module Puissance 4	I	I	I	R	A	A	
Intégration	R	A	C	C	A	R	
Tests	C	A	C	R	A	C	
Corrections erreurs / bug	R	I	A	A	I	R	
Manuel utilisateur	I	R	C	C	A	I	

Tableau 6 : Extrait de la matrice RACI

Mais finalement pendant le conception du code, nous avons remarqué que le Module Joueur/Administrateur demandait plus de temps et de ressource, nous avons donc décidé que Landry venait nous aider, après quoi la première semaine nous avons réussi à faire quelques vue et la globalité des modules, chacun devait réaliser une tâche qu'on avait défini grâce au Trello mais également au discord qu'a créé à l'occasion :

```

Phoque éberlué 17/06/2020
@everyone voilà ce qu'on doit faire aujourd'hui, le contrôleur de chaque classe qui sera relié à ce dont on à besoin pour faire fonctionner l'appli (jdbc, modele...). ~Par exemple pour Connexion, on fait un contrôleur qui prend les actions des boutons, le texte dans les champs de connexion et utilise les méthodes des jdbc pour pouvoir se vérifier les identifiants.

private VueConnexion vueConnexion; //GWENDAL
private VueInscription vueInscription; //GWENDAL
private VueJeux vueJeux; //LANDRY
private VueSocial vueSocial; //SIMON
private VueMatchHistory vueMatchHistory; //MAXIME
private VueInvitations vueInvitations; //SIMON
private VueProfil vueProfil; //MAXIME
private VueAdmin vueAdmin;
private VueMessage vueMessage; //LANDRY
private MenuVapeur menuVapeur;

et @warsoso on va faire le puissance 4

```

Figure 21 : Message du chef de projet sur *Discord*

Pendant la deuxième semaine, nous avons commencé à relier nous différents code afin d'avoir une structuration commune, en effet avec le Chef de projet nous avons décidé de créer un arbre sémantique pour mieux se repérer dans le projet et éviter de se perdre, cela nous a permis de gagner beaucoup de temps :

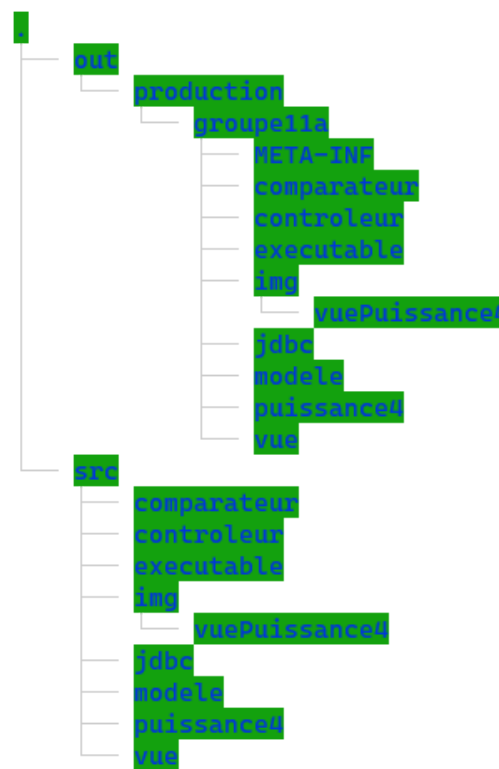


Figure 22 : Arbre sémantique du projet

On remarquera 2 dossiers, **out** qui représente les fichiers java créés par le IDE qui nous permettra après de lancer l'application (La Compilation) et le dossier **src** où nous stockons notre code. Vous pourrez reconnaître certains mots qu'on a énoncés plus haut comme la **vue**, **jdbc**, **modèle**, etc.

C'est à la fin de semaine qu'on a divisé le groupe en 2 parties, certains devaient aider l'autre groupe pour finaliser le développement du module **Puissance 4** et les autres devaient corriger ou trouver les éventuels bugs. Nous avons au final pu avoir une application qui marche avec pratiquement toutes les fonctionnalités énoncées dans le cahier des charges.

4. Bilan collectif

En réalisant ce projet, nous avons pu voir toutes les étapes nécessaires au développement d'un projet informatique. Ainsi que l'importance de la partie analytique et de conception afin de mener à bien la réalisation sans difficulté.

Ce projet nous a initié au travail en groupe et à la collaboration. De plus, à cause de l'épidémie du COVID-19, nous avons également appris à mener un projet à bien malgré la distance, ce qui est une façon de travailler très répandue dans le milieu de l'informatique. Nous avons appris à utiliser Git (même si nous avons tous de solides bases) dans un cas pertinent. Régler des problèmes de conflits, revenir sur une version antérieure du projet, s'assurer que les modifications apportées dans son commit ne soient pas buggées, tout cela nous a fait prendre de l'expérience sur ce logiciel de versionnage.

Lors de ce projet, nous avons rencontré certaines difficultés. La principale difficulté était dû au fait que le projet était uniquement relié à une base de donnée. Nous avons donc dû penser les choses différemment que si nous avions travaillé avec un serveur.

Pour nous ce projet était très intéressant. Il nous a appris une multitude de choses. Cependant on a trouvé que la phase de programmation était trop courte par rapport à la phase d'analyse, et nous n'avons pas eu le temps de faire tout ce que nous voulions et comme nous le souhaitions.

5. Fiches individuelles

a. Simon CAVILLON


Pour la première phase, nous avons réparti les tâches c'est alors que j'ai été attribué à la réalisation du diagramme des cas utilisations mais avant cela j'ai participé à l'élaboration du dictionnaire de données car il était important que tout le monde aide pour la structure de nos données dans le projet (30min), par la suite j'ai réalisé pendant cette séance de **1h30** le début du diagramme grâce à StarUML.

Ensuite pour la seconde phase, j'ai continué donc mon diagramme des cas d'utilisations afin d'apporter plus de précision dans les cas d'utilisation de nos 2 acteurs principaux. J'ai également pu résoudre un problème grâce au TP UML du matin qui concernait l'include entre Inscription et Connexion qui n'était pas bonne, je l'ai donc remplacer par un extend qui part de connexion et qui pointe Inscription.

J'ai par la suite poster différentes versions au groupe en leurs demandant leurs avis, c'est grâce à un groupe discord que nous avons pu nous réunir et apporter nos travaux afin d'argumenter tous ensemble et d'améliorer les travaux de chacun comme par exemple j'ai aider Gwendal pour la cardinalité dans le MCD (**environ 2h / 2h30**).

Nous avons mis en commun le travail que nous avons réalisé pendant le mois d'Avril, j'ai eu comme chose à faire la restructuration du diagramme des **Cas d'utilisation** et particulièrement pour le **module Joueur et Administrateur**, pour cela j'ai analysé une nouvelle fois le cahier des charges et les fonctionnalités pour avoir les cas d'utilisations, je me suis penché sur le fait que je devais détaillé un maximum afin de représenté le plus clairement et explicitement les interactions que l'utilisateur/administrateur peut avoir avec notre système et le Serveur.

J'ai poster différente version jusqu'à avoir un résultat satisfaisant, on m'a confié également la réalisation du planning à l'aide de l'outil *Gantt Project*, j'ai donc attendu que le modèle RACI soit terminé afin de pouvoir noter les tâches clés et pouvoir estimer les heures en jours.



Pour la troisième phase, nous avons mis en commun le travail que nous avons réalisé pendant le mois Mai. Durant celui-ci nous avons travaillé les **IHM** avec notamment la partie **diagramme des tâches** ou l'objectif était d'expliquer graphiquement l'approfondissement chaque cas d'utilisation grâce à la description textuel. Pour ma part je me suis chargé de réaliser les Cas d'utilisations 6, 2, 4, 1 (Module Joueur/Administrateur) et le Cas d'utilisation 5 (Module Puissance 4)

Après avoir effectué les diagrammes des tâches, nous avons commencé à imaginer la partie graphique de notre plateforme, les **maquettes** qui ont été effectués par tous le monde. Nous avons utilisé la plateforme *figma.com* qui nous a permis de réaliser tous ensemble les maquettes en ligne. Après avoir choisie nos couleurs et la forme qu'aurait l'accueil nous nous sommes tous basés dessus pour réaliser nos maquettes. j'ai réalisé la maquette **Social, Chat et Invitations**. Pour pouvoir les réaliser je me suis inspirée de l'organisation de Discord pour le Social et le Chat, j'ai trouvé leur manière de présenter intéressante. Notamment le système de notification avec le 1 en rouge. J'ai essayé également de rendre la manipulation bien plus facile d'accès avec le moins de redondance possible.

Pour la dernière phase, nous nous sommes mis en commun pour savoir qui devait faire quoi et comment, pour ma part j'ai été chargé de réaliser les interactions entre la base de donnée et le module Joueur/Administrateur donc la JDBC du module Joueur/Administrateur, ça été une étape importante et enrichissante de pouvoir utiliser des données de la base donnée directement dans la vue. Pour cela j'ai dû communiquer avec Gwendal qui réalisait les vues mais également Landry qui réalisait les modèles. J'ai dû créer des méthodes spéciales pour faciliter le travail de mes collègues afin qu'ils ont toutes les informations qui ont besoin pour afficher (Vue) ou créer l'objet (Modèle).

Après avoir réalisé la JDBC, je suis venu en aide à Gwendal pour réaliser les contrôleurs, ils sont importants car ils représentent les événements d'un bouton, ou textfields, etc. J'ai dû utiliser mes méthodes de JDBC pour recueillir mes informations afin de les afficher selon un événement comme un clique. Pendant le développement j'ai également dû créer des nouvelles méthodes alors que je faisais les contrôleurs car je n'avais pensé à faire certaines fonctionnalités, parfois c'était long et pour dire celle qui m'a coûté le plus de temps a été la liste des messages. En effet la méthode qu'on avait réalisé avec Landry nous donnait pas ce qu'on voulait j'ai donc dû revoir ce que j'avais auparavant (JDBC) afin d'avoir un meilleur résultat de ma requête SQL.




b. Maxime FEVRIER

Tout d'abord, lors de la **première phase** de ce projet, nous avons effectué la conception du dictionnaire de données. Nous avons tout d'abord fait un premier jet où chacun envoyait les données qu'il trouvait utiles puis dans un deuxième temps nous avons débattu pour décider de quel données étaient les plus intéressantes à garder. J'ai ensuite, avec l'aide de Nino, conçu les requêtes qui nous ont été demandées par le professeur. Cela nous a pris environ 6h.

Lors de la **deuxième phase** du projet, le travail était concentré sur l'analyse des modules de notre futur plateforme de jeu. Je me suis chargé de l'analyse du module Joueur net Administrateur avec Simon Cavillon et Gwendal Héricourt. Nous nous sommes réparti le travail en trois parties distinctes et j'ai donc réalisé le diagramme de séquence du scénario nominal de ce module. J'ai passé environs 3h sur cette partie. Ensuite j'ai été consulté pour la création de la matrice RACI que Andrew et Gwendal ont effectué.

Lors de la **troisième phase** du projet, nous avions différentes choses à faire. Avec Landry et Nino, nous avons débuté par l'analyse UML avec le diagramme des tâches ce qui nous a pris 3h. Ensuite nous devions faire les maquettes de l'interface graphique, nous nous étions chacun répartis des pages à faire. personnellement je me suis occupé des IHM concernant les statistiques et l'historique des parties ce qui m'a pris 3h.



Avant de commencer la partie développement de la **quatrième phase** du projet, j'ai dû choisir avec quel environnement de travail j'allais travailler. J'ai tout d'abord pensé que j'allais utiliser Visual Studio Code. Cependant, ayant rencontré plusieurs problèmes j'ai finalement décidé d'utiliser l'IDE IntelliJ que j'avais pu utiliser auparavant lors des différents TP.

Au début de cette phase nous avons tout d'abord organisé les tâches à faire sur Trello ce qui nous a permis de vite être plus organisés. Nous reportons aussi tout le travail à faire sur un salon de discussion discord.

Pour cette partie développement je me suis donc chargé principalement des vues notamment la vue intitulée « VuePuissance4 » sur laquelle j'ai passé une bonne partie de mon temps. Une fois la vue créée, il a fallu relier les codes concernant le jeu puissance 4 ainsi que cette même vue. Nous avons rencontrés plusieurs problèmes lors de cette liaison, notamment des problèmes d'affichages sur lesquels nous sommes restés bloqués un bon moment. Pendant mon travail j'ai principalement travaillé en collaboration avec Andrew et Nino, même si Gwendal nous a rejoint à la fin pour nous aider à corriger les bugs de liaison.




c. Gwendal HERICOURT

Lors de la **première phase** du projet, nous avons tous participé à la conception du dictionnaire de données. Chacun y ajoutait les données qu'il pensait utile, puis nous en débattions afin de l'optimiser. Je me suis ensuite chargé de faire le modèle conceptuel des données (MCD) avec l'aide des autres membres qui corrigeaient mes erreurs. Cela sur un total d'environ 6 heures avec les améliorations et corrections que l'on apportait sans cesse. Après le script de création de la base de données fait, je me suis occupé d'ajouter les valeurs nécessaires au lancement des requêtes demandées pour pouvoir vérifier qu'elles fonctionnent.

Ensuite, pour la **deuxième phase**, le travail était concentré sur l'analyse des modules de notre futur plateforme de jeu. Je me suis chargé de l'analyse du module Joueur et Administrateur avec Simon Cavillon et Maxime Février. Nous nous sommes réparti le travail en trois et j'ai donc réalisé la description textuelle du diagramme des cas d'utilisation de ce module. J'y est passé près de 3 heures mais j'ai aussi été consulté 1 heure pour le diagramme des cas d'utilisation du module. Ensuite, j'ai mis en place la matrice RACI pour le projet que Andrew et moi avons rempli pendant 3 heures en consultant les membres du groupe.

Pour la **troisième phase** du projet, je me suis tout d'abord chargé du modèle des tâches du projet avec Simon Cavillon et Andrew Mary Huet de Barochez. Nous nous sommes réparti le travail et réaliser le travail en se basant sur les descriptions textuelles de la phase précédente. J'ai passé près de 5 heures pour les diagrammes. J'ai ensuite déterminé l'avancement du projet avec les autres membres. Après cela, j'ai mis en place un groupe sur figma.com, pour réaliser nos IHM. Nous avons débattu sur la direction et la charte graphique de l'application en faisant l'IHM de la page "Jeux" ensemble. Puis les différentes maquettes ont été réparti et je me suis chargé des pages "Connexion" et "Inscription" et aider pour les autres sur environ 4 heures.



Enfin, à la **dernière phase**, nous devons développer le projet en lui-même. Pour cela, le travail a été séparé en paquets (packages) : vue, modele, controleur, jdbc et puissance4. Cela nous a permis de bien organiser le travail. Je me suis chargé en grande majorité de la vue de presque toutes les pages. En effet, chaque page de l'application correspond à une classe java que j'ai codé presque entièrement avec l'aide de Simon pour certaines telles que les pages connexion, inscription, jeux, social, historique des parties, administrateur et profil, ou seulement participé pour ajouter le style graphique comme les vues de message, invitation et puissance 4. J'y ai passé une grande partie de mon temps pour environ 35 heures sur les 50 prévues par l'emploi du temps.

Lorsque les vues étaient terminées, j'ai réalisé plusieurs contrôleurs permettant de naviguer parmi et sur les vues de l'application. Ainsi que des "timelines" permettant de vérifier toutes les 5 secondes si un changement à été effectué dans la base de donnée et dans ce cas, rafraîchir la page sur laquelle se trouve l'utilisateur. Cela ma pris le reste de mon temps en plus de la correction de certains bugs ou fautes dans le programmes.



d. Andrew MARY HUET DE BAROCHEZ


En tant que chef de projet, lors de la première phase du projet, j'ai dû organiser mon groupe afin de pouvoir collaborer de manière efficace.

Nous avons créés un serveur discord afin de centraliser toutes les communications ainsi que les documents internes au projet.

Lors de la première phase nous avons tous travaillé ensemble sur la base de données afin d'être d'accord sur la façon dont nous allons construire notre projet.

J'ai du également répartir le travail dans le groupe afin de pouvoir effectuer un rendu dans les temps. Nous avons utilisé énormément d'outils différent dont beaucoup qui nous ont été présenté en cours tel que : StarUML, Mocodo online, Google sheet, Draw.io et évidemment mySql pour tester nos requêtes.

Premièrement il m'a fallu répartir le travail de la deuxième phase en deux groupes, 3 personnes sur le module Joueur et Administrateur et 3 personnes sur le module puissance 4 (dont moi). J'ai donc aidé à la réalisation de l'analyse du module puissance 4 en faisant le diagramme de séquence du scénario nominal. Secondement j'ai du, avec mon groupe, planifier, organiser et répartir les différentes tâches du projet concernant la phase 3 et 4 du projet. Gwendal et moi avons utilisé la matrice RACI Excel que nous avons programmé en cours de gestion de projet afin de répartir les différentes tâches et également pour définir des rôles de responsables, acteurs, consultés et informés. Nous avons évidemment demandé l'avis au membres du groupe pour qu'ils choisissent les tâches qui leur conviendrait au mieux. Simon a pu réaliser le Gantt en parallèle que nous faisons la matrice RACI.




Pour la dernière phase avant le code, il y a eu énormément de travail à réaliser. Tout d'abord Gwendal Simon et moi devions faire environ 17 diagrammes de tâches différents. Pour cela nous nous sommes répartis équitablement les diagrammes et nous avons tous respectés la même forme en utilisant le site Draw.io. Ces diagrammes furent rapide à faire étant donné qu'il s'agissait d'adapter les descriptions textuelles en diagramme. Puis nous nous sommes rendu compte que l'équipe qui se chargeait de l'analyse UML avait quasiment fini alors nous avons modifié notre matrice RACI afin de nous répartir la création des IHM entre tout le groupe. Nous avons dus établir une charte de couleur, un style et une base de notre application afin que les IHM soient cohérentes. Avec nino nous avons réalisé l'ihm du puissance 4 qui était plutôt simple et rapide à faire puis je suis allé aider les autres sur les ihm relatif à la plateforme.

Cette phase fût sans aucun doute la plus stressante. Nous avons commencé par revoir toute l'analyse pour être sûr de comment nous allions réaliser l'application. Ce ne fût absolument pas du temps perdu car en effet dans notre analyse nous avons omis le fait que cette application était seulement connectée à une base de donnée et qu'il n'y allait pas avoir de serveur. Heureusement cette phase d'analyse fût plutôt rapide et nous avons pus remodeler l'application afin qu'il y est qu'une connexion entre les joueurs avec la base de données.

J'ai donc distribué les tâches aux membres du groupes en fonction de leur compétences : Maxime et Gwendal sont très à l'aise avec JavaFX, ils se sont donc occupés des IHM; Landry et Nino ont une bonne compréhension de l'algorithme et de Java, ils ont donc réalisé les modèles de l'application; quant à Simon et moi, nous avons réalisé les JDBC car nous sommes tous deux apte à créer des requêtes SQL fiables ainsi que d'utiliser l'interface de programmation java data base connectivity.

Un fois ces tâches réalisées (excepté les Vues qui étaient très longues à faire), j'ai refactorisé le classes touchant à la JDBC et j'ai réorganiser les fichiers du projet. Premièrement pour la refactorisation, j'ai modifié les méthodes JDBC afin qu'elles soient statiques, c'est à dire qu'elles pourraient être appelées dans n'importe quel fichier du projet sans avoir à être instanciées. Secondement avec IntelliJ, j'ai réorganisé les fichiers classe qui étaient tous rangés dans le même dossier à l'aide des packages. Le principe était très simple, il fallait rassembler toutes les classes de même "type" : les modèles, les JDBC, les Vues, les Contrôleurs et les classes du puissance 4.



Finale­ment j'ai aussi partici­pé au con­trô­leur du rela­tif au puis­sance 4 (Accep­ter une invi­ta­tion au jeu, recher­cher une partie, jouer au jeu). J'ai aussi essayé de faire la vue du puis­sance 4, mais je crois avoir un problème avec JavaFX et le frontend en général donc je n'ai pas réussi. J'en conclu que je suis bien meilleur en backend qu'en frontend.

Pour conclure nous avons rédigé chacunes de nos parties dans la rapport et finalisé le projet.



e. Nino SPINOLA

Pour ce **début de projet**, nous avons tous fait l'analyse du projet et le dictionnaire des données. Je me suis également occupé des requêtes demandé par le professeur. Ce travail nous a en tout pris 6h.

Durant la **deuxième phase** du projet je me suis occupé de faire de diagramme des Cas d'Utilisation pour le puissance 4 pendant 2h et j'ai été informé pour l'analyse du module Joueur et Administration.

Lors de cette **troisième phase**, nous avons plusieurs chose à faire. Avec Landry et Maxime, nous avons commencé par faire l'analyse UML avec le diagramme des tâches ce qui nous a pris 3h. Ensuite nous devons faire les maquettes de l'interface graphique, nous nous étions chacun répartis des pages à faire. Avec Andrew, nous nous sommes occupé des page du puissance 4 durant une partie ce qui nous a pris 3h également.

Lors de cette **quatrième** et dernière phase de ce projet, nous devons faire le développement de notre application. Je me suis occupé de tout l'algorithme en java du puissance 4, ce travail m'a prit une grosse partie du temps total pour cette phase. J'ai aussi fait une parti du diaporama et du manuel utilisateur.



f. Landry VALETOUX

J'ai proposé une première modélisation de la base de données en coordination avec l'équipe sur Draw.io reprise par la suite par Gwendal. Puis, j'ai également participé à la création du dictionnaire de données sur son organisation comme le contenu. Enfin, j'ai proposé un premier script de création de la base de données en MySQL, validé par l'équipe.

Ensuite, j'ai participé à la réalisation du diagramme RACI et GANTT avec l'ensemble de l'équipe. L'objectif étant de planifier les tâches à suivre pour nous organiser au mieux dans le déroulement de notre travail. On m'a mis responsable des parties Test et Corrections des erreurs pour le développement du projet.

J'ai travaillé en collaboration avec Nino et Maxime sur le diagramme UML afin de modéliser l'intégralité du projet. Celui-ci inclut également le jeu du puissance 4 intégré dans la plateforme. Puis, nous avons créé avec l'ensemble de l'équipe une charte graphique afin d'établir une première maquette de l'application.

Avant de commencer à développer, il me fallait configurer mon environnement de travail. En pensant pouvoir utiliser avec aisance VSCode avec JavaFX je me suis confronté à plusieurs désagréments. Une multitude d'erreur de compilation qui m'a pris plusieurs jours à résoudre jusqu'à ce que je décide de passer à IntelliJ, l'IDE utilisé pendant nos TPs.

Durant la phase de développement, nous avons dans les premiers temps mis en place des tableaux de tâche sur Trello afin de mieux nous les répartir. Je me suis chargé de la partie des Modèles soit : l'utilisateur, les messages, les amis et le modérateur. Je me suis également occupé de créer des contrôleurs et vues pour les pages « Sociale » et « Chat ». Je me suis beaucoup concerté avec Simon pour synchroniser la base de données à la Vue de la messagerie.

Finalement, le projet présente les fonctionnalités de base demandés dans notre cahier des charges. Nous sommes satisfaits de ce que nous avons accomplis ensemble. Ce projet nous a apporté beaucoup d'expérience en gestion de projet en équipe et en connaissances techniques pour ce qui est du langage JAVA.

VIII. Conclusion

Le but de ce projet informatique de fin de semestre était de réaliser une plateforme en ligne. Elle doit permettre à des utilisateurs de s'y connecter ou s'inscrire, communiquer et faire des duels sur des jeux en ligne. Dans le cas présent, un jeu de Puissance 4 devait être implémenté et accessible depuis l'application.

Nous avons donc mener à bien le projet en passant par quatre phases. Tout d'abord la mise en place d'une base de données correspondant aux données que nous aurons à stocker pour la plateforme. L'analyse du projet, la réalisation de diagrammes et la séparation en deux modules : Puissance 4 et Joueur et Administrateur. Le maquettage pour une prévisualisation de l'application. Finalement, la réalisation de la plateforme et du Puissance 4 en Java.

La dernière étape était la plus intense mais nous a permis de produire une application qui fonctionne. Des utilisateurs peuvent se connecter, s'inscrire, avoir une liste d'amis et lancer des parties de Puissance 4 dont ils peuvent voir l'historique. De plus, les administrateurs peuvent gérer les joueurs et les parties.

Le projet à été complété dans le temps qui nous a été permis même si certaines fonctionnalités peuvent être améliorées ou ajouté pour le perfectionner. En effet, avec du temps supplémentaire nous aurions pu envisager d'améliorer certains points. Tels que pouvoir mettre pause à une partie et la reprendre plus tard, demander l'avis de l'utilisateur qui a envoyé une invitation avant de lancer le jeu, rendre le chargement plus rapide lorsqu'un administrateur se connecte ou éviter des problèmes lors de la définition de certaines invitations. Ou même, ajouter de nouvelles fonctionnalités comme trier les amis et parties selon des critères, pouvoir faire des recherches par nom, utiliser sa propre image de profil ou bien implémenter d'autres jeux pour la plateforme.

IX. Annexe

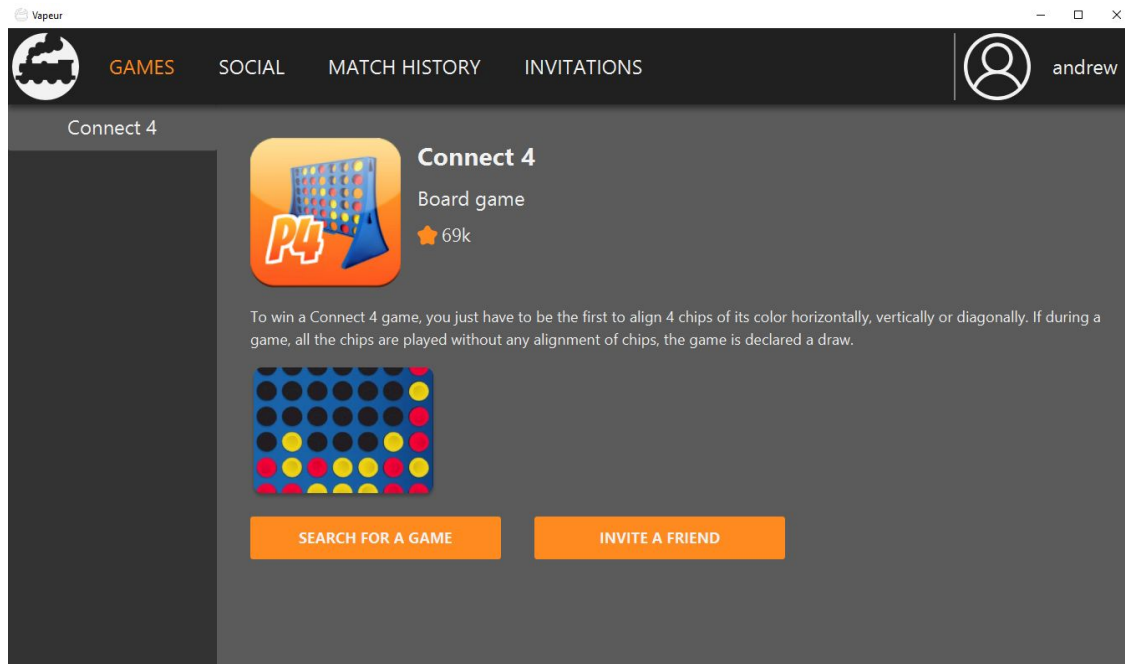


Figure 23 : Page “Games” de notre application Vapeur

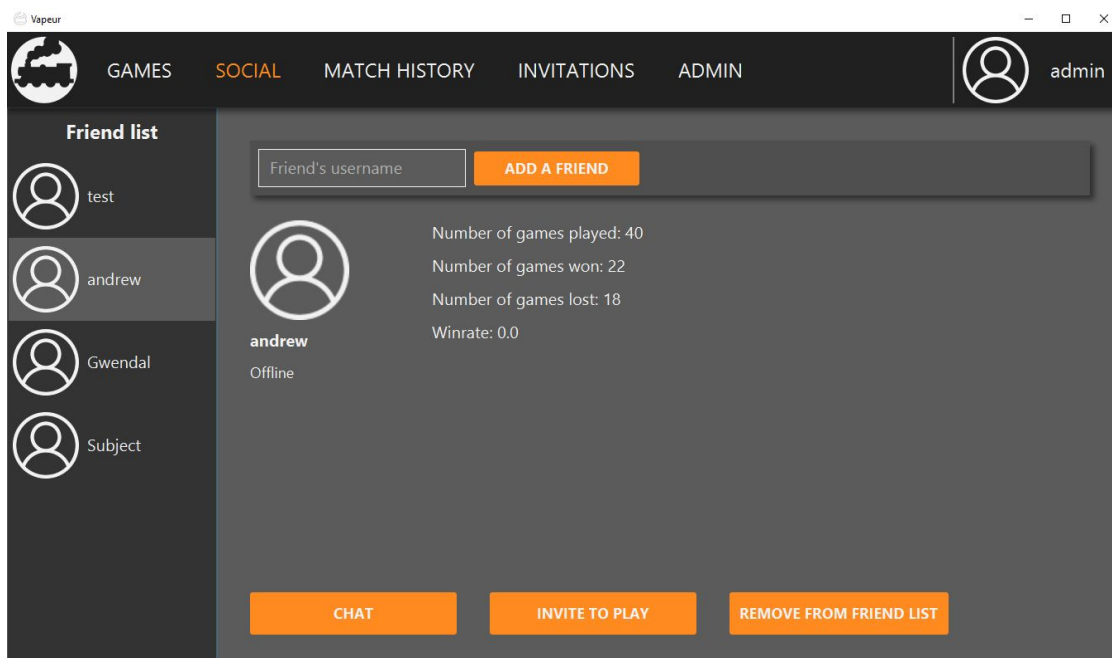
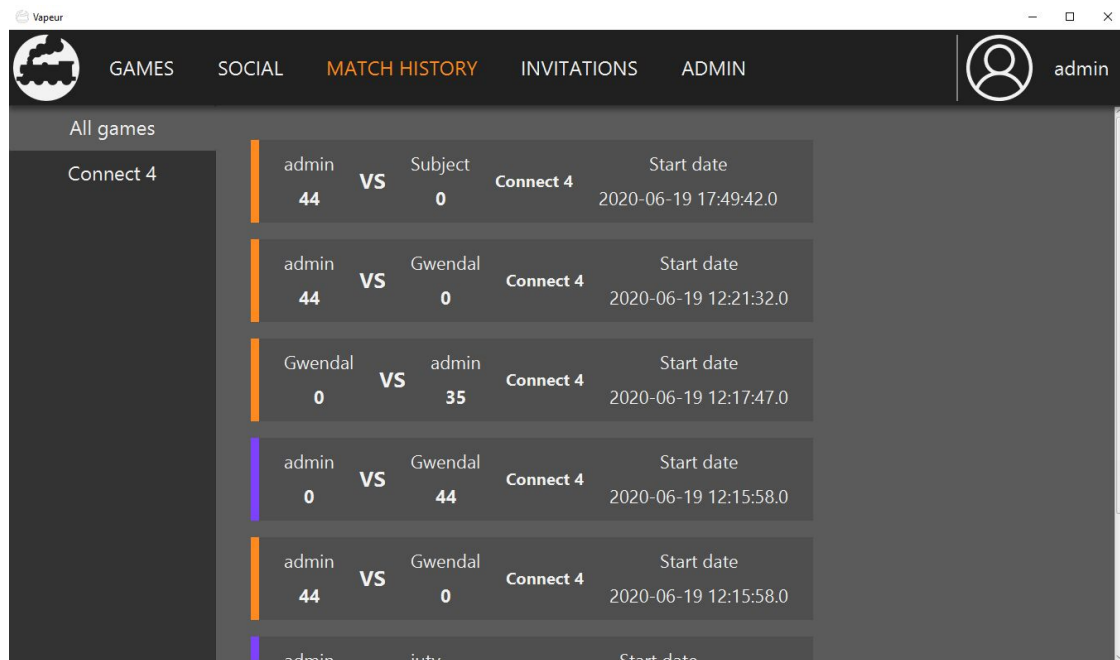


Figure 24 : Page “Social” de notre application Vapeur



admin	VS	Subject	Connect 4	Start date
44		0		2020-06-19 17:49:42.0
admin	VS	Gwendal	Connect 4	Start date
44		0		2020-06-19 12:21:32.0
Gwendal	VS	admin	Connect 4	Start date
0		35		2020-06-19 12:17:47.0
admin	VS	Gwendal	Connect 4	Start date
0		44		2020-06-19 12:15:58.0
admin	VS	Gwendal	Connect 4	Start date
44		0		2020-06-19 12:15:58.0
admin				Start date

Figure 25 : Page “Match History” de notre application Vapeur

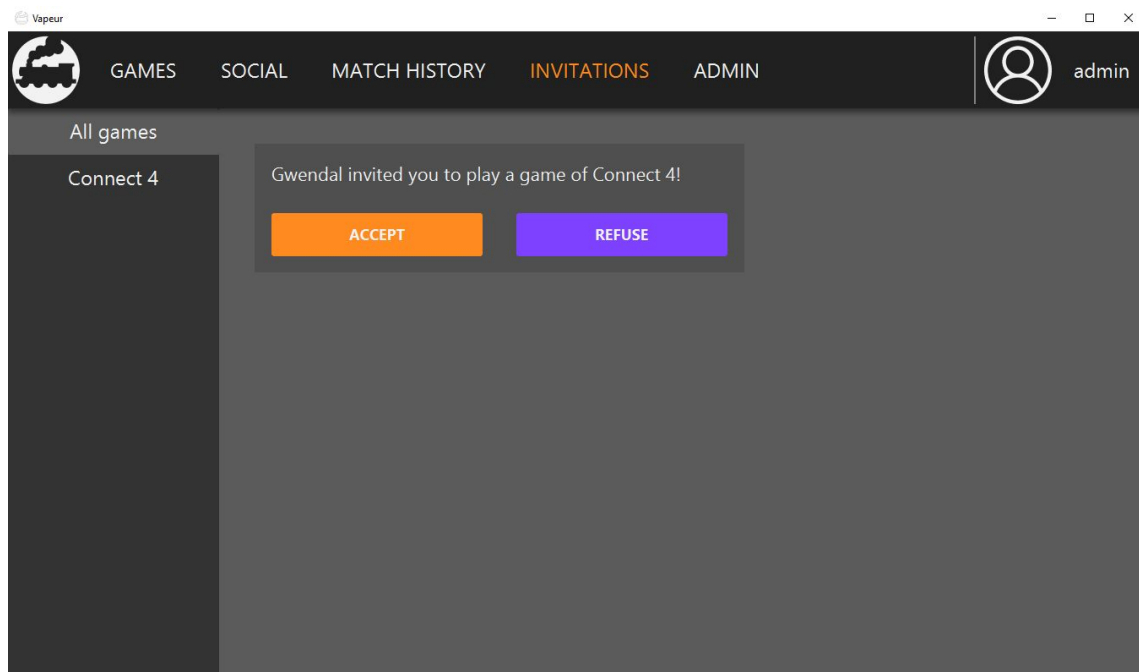


Figure 26 : Page “Invitations” de notre application Vapeur

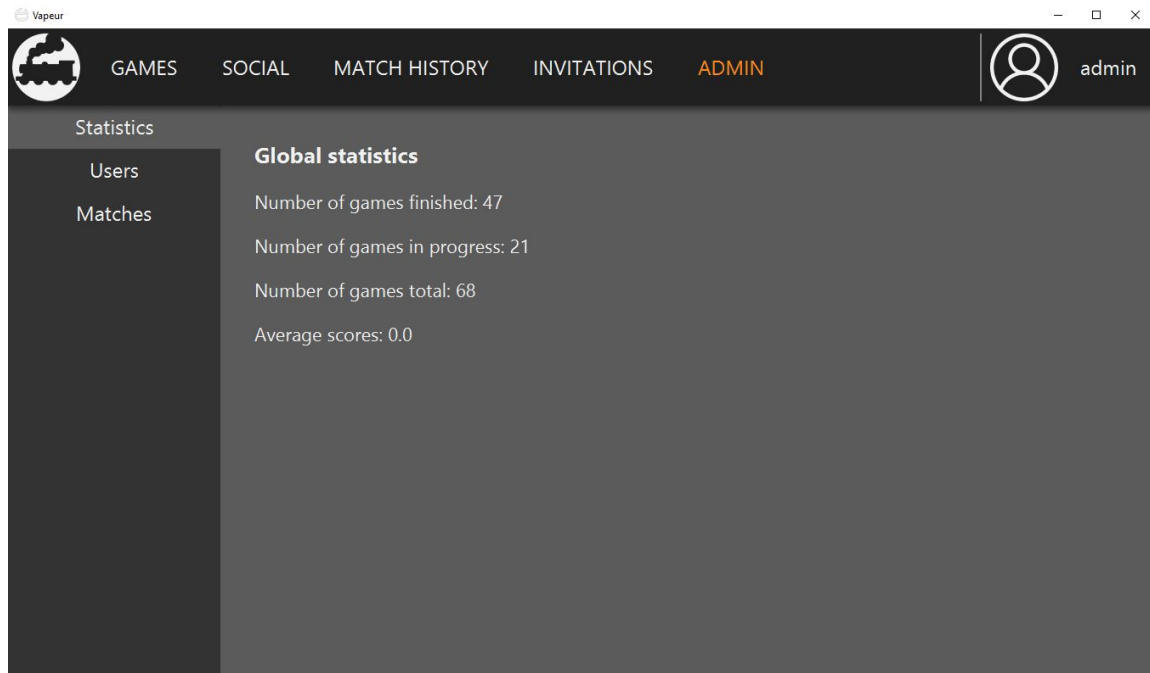


Figure 27 : Page “Admin / Statistics” de notre application Vapeur

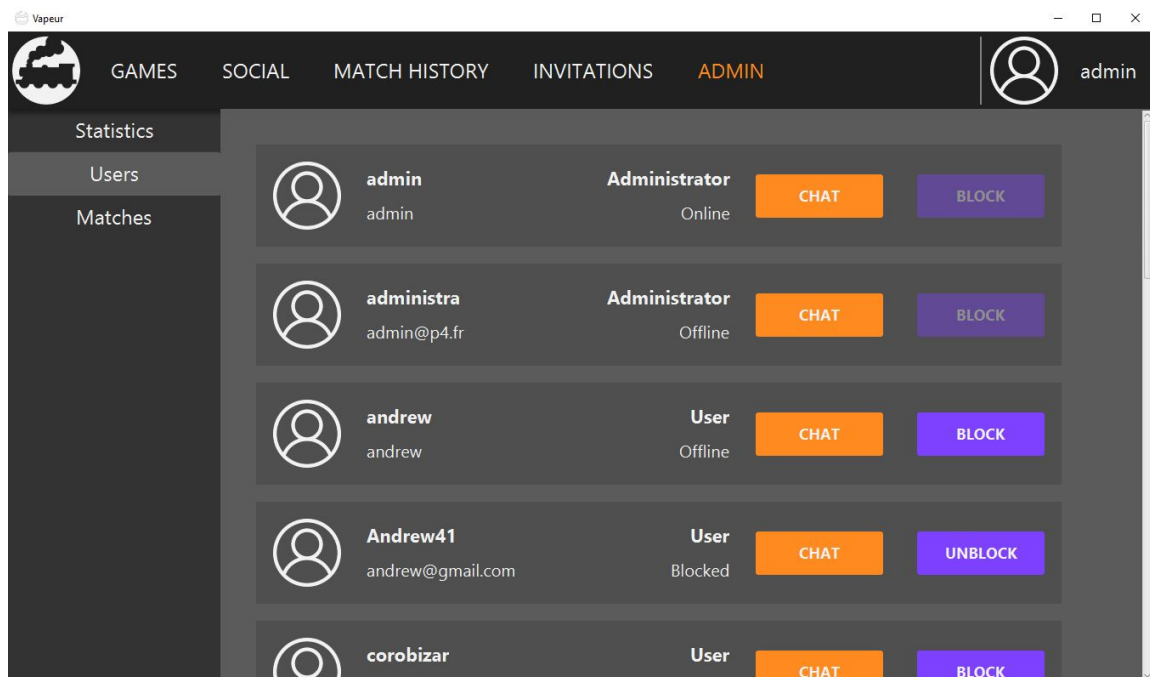


Figure 28 : Page “Admin / Users” de notre application Vapeur

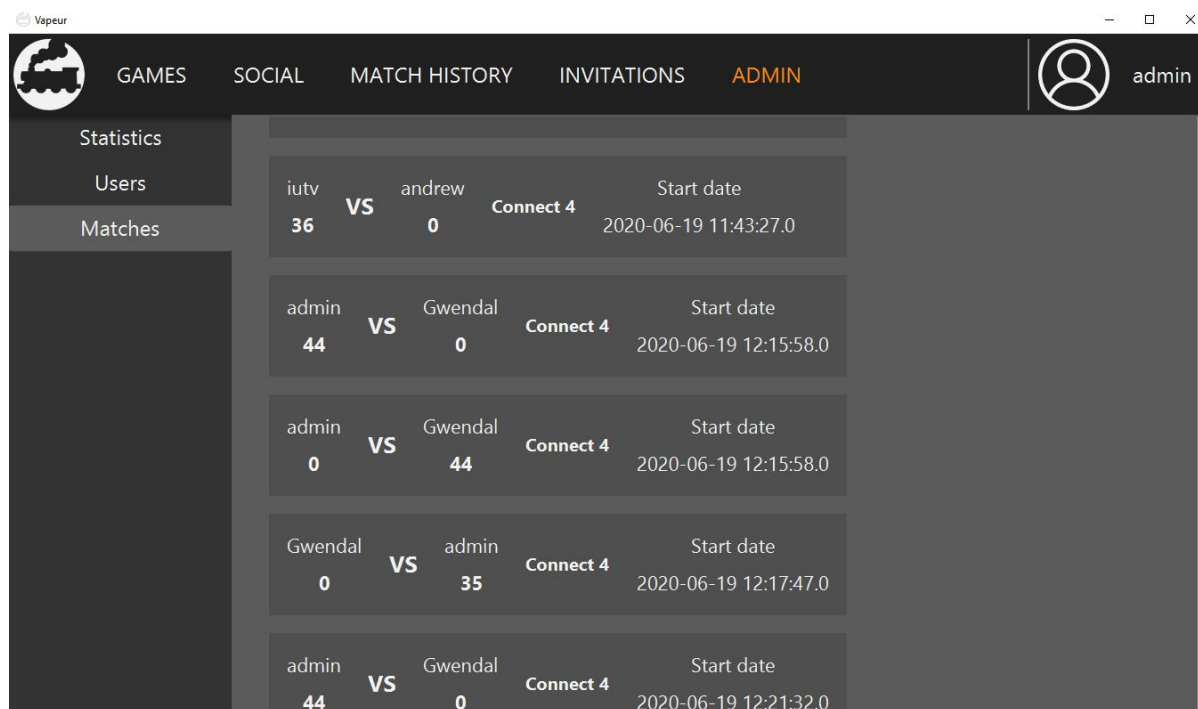


Figure 29 : Page “Admin / Matches” de notre application Vapeur

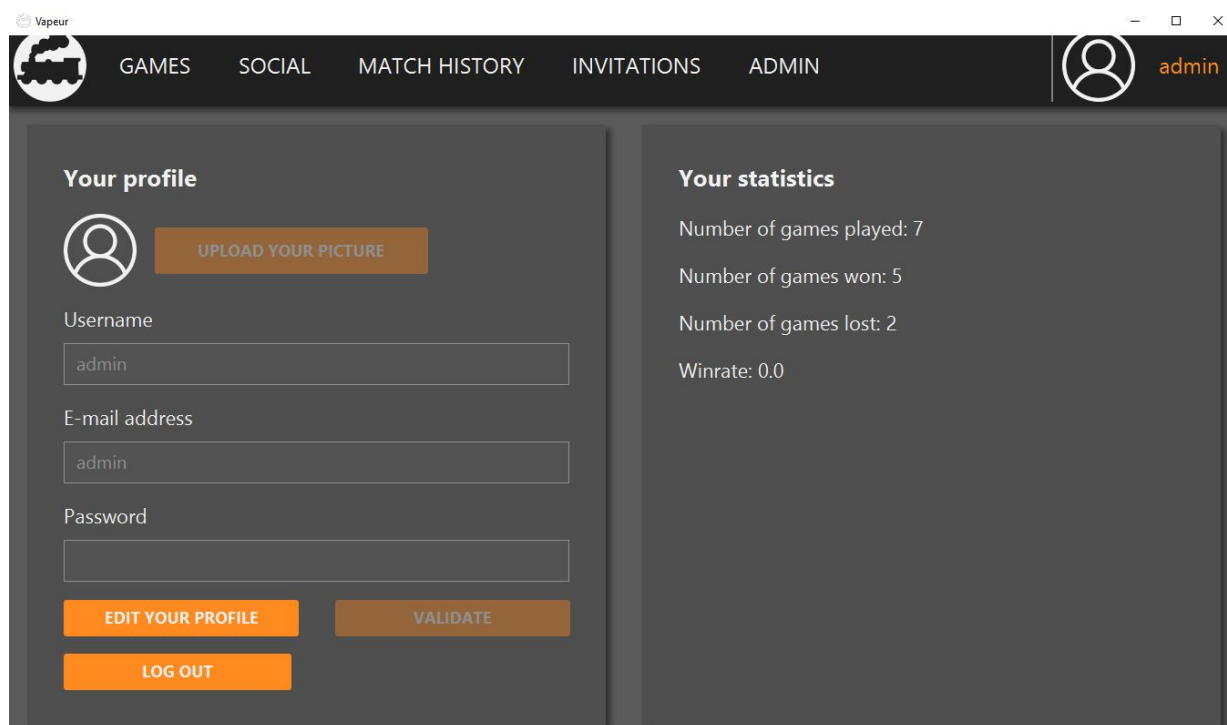


Figure 30 : Page “Profile” de notre application Vapeur