

Nathan DUCREY
Arnaud JELMONI
Céphise LOUISON
César MARCHAL
Vincent POULAIN
Xavier RODRIGUEZ



Dossier de réalisation

Groupe IHM Connexion

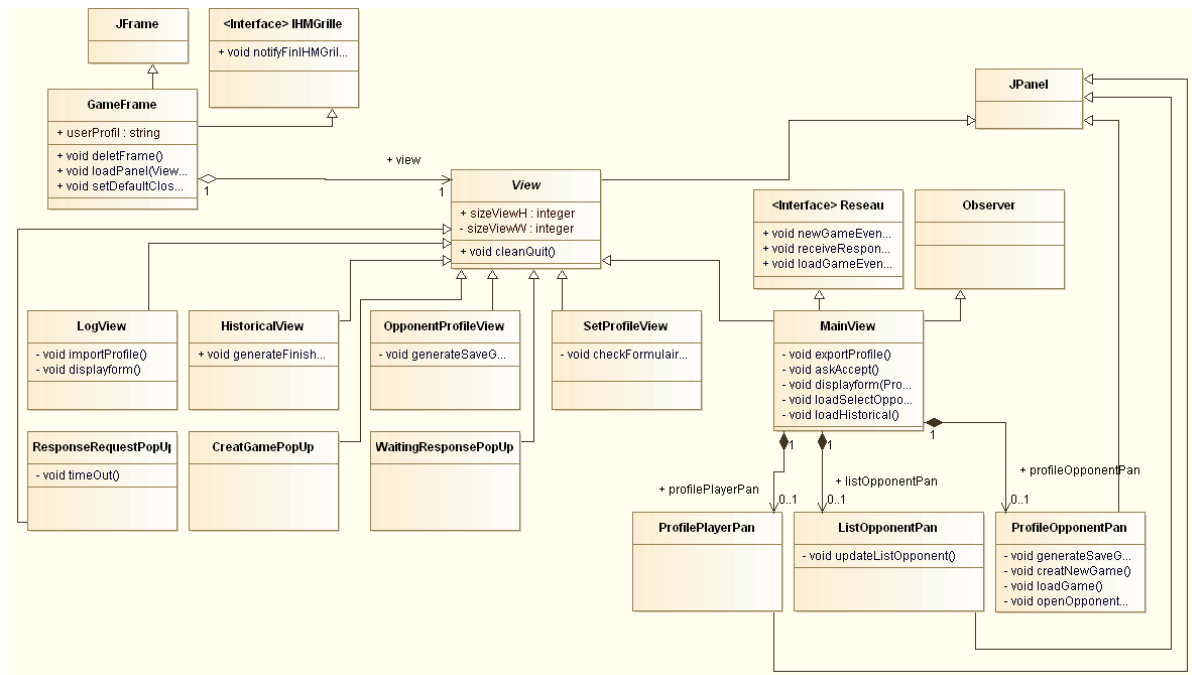
Responsable : M. Benjamin LUSSIER

Sommaire

I. RETOUR SUR LA CONCEPTION.....	4
II. DEVELOPPEMENT	6
1. OUTILS ET COMPONENT.....	6
<i>Classe PasswordFieldCustom et InformationField :</i>	6
<i>LobbyConstant :</i>	7
2. CREATION DES INTERFACES.....	8
<i>LogView :</i>	9
<i>SetProfileView :</i>	11
<i>MainView :</i>	12
<i>OpponentProfileView :</i>	14
<i>HistoricalView :</i>	16
<i>ResponseRequestPopup :</i>	18
<i>WaitingResponsePopup :</i>	19
<i>CreateGamePopup :</i>	19
3. DIFFICULTES RENCONTREES	20

I. Retour sur la conception

La phase de conception à travers le diagramme de classes de notre groupe IHM Connexion est d'un certain côté différente des autres groupes car les définitions de toutes nos classes ne sont pas aussi complètes que dans les autres modules. Par conséquent, à partir des spécificités et des maquettes que nous avons apportées dans le dossier de conception voici le diagramme de classes qui s'est dégagé avant la phase de développement :



Avant d'élaborer ce diagramme de classes, nous avons préalablement fait une réunion avec le groupe d'IHM grille afin de se mettre d'accord sur les technologies propre à JAVA que nous allons utiliser et ainsi mettre en place une première cohérence dans le code de notre projet. Il a donc été décidé que nous utiliserions la bibliothèque SWING pour créer les interfaces graphiques car c'est l'API la plus répandu et plus rapide que AWT.

De plus, d'après la proposition de notre chef de groupe Xavier Rodriguez, nous avons choisi de simplifier la gestion des différentes interfaces graphiques en mettant en place un système de vues qui seront chargées à la volée dans la fenêtre principale. Cela va présenter des avantages, notamment comme le passage d'une vue à une autre sans que l'on ferme la JFrame de la vue pour en ouvrir une autre avec la nouvelle vue. Cela va rendre au final notre IHM beaucoup plus ergonomique et plus convivial pour l'utilisateur, mais surtout cela va faciliter la gestion de la communication réseau. En effet, après l'étape d'authentification sur la machine un pointeur sur la classe principale du module réseau sera instancié une unique fois dans la fenêtre principale pour initier la communication sur le réseau avec tous les autres utilisateurs.

Pour ce faire, nous avons donc créé une classe *GameFrame*, qui en plus de contenir le profil complet de l'utilisateur courant, s'occupera de charger les différentes vues que notre IHM proposera à l'aide de la méthode *LoadView*. Après discussion avec le groupe d'IHM grille, il a également été décidé qu'ils utiliseraient notre gestion des vues et donc notre classe *GameFrame*, ce qui rendra le déroulement du jeu plus cohérent, d'avoir toutes les informations du joueur directement accessible et surtout de disposer de la communication réseau initialisée est prête pour échanger des données pendant le déroulement de la partie. C'est pourquoi cette classe implémente également les interfaces notifiant le lancement de partie à l'IHM grille mais

également le retour à l'IHM connexion après la fin d'une partie. Au final, le choix technique de développer cette classe à un peu pénalisé le début de notre phase de développement car elle a bloqué la création de toutes les autres, mais nous pouvons dire que cela n'a pas été vain car cela nous a grandement facilité le travail pour la gestion des vues et pendant la phase d'intégration avec le module d'IHM grille et le module Réseau.

Ensuite, nous avons décidé de créer une classe Mère *View* dont toutes nos vues principales hériteront, cela nous a ainsi permis de facilement fixer la taille de toutes nos vues de manière indépendante. Les vues héritant de cette classe et qui seront chargées dans la *GameFrame* sont :

- **LogView** : Vue permettant l'authentification de l'utilisateur.
- **SetProfileView** : Vue permettant la création ou la modification d'un profil.
- **MainView** : Vue principale de notre IHM proposant un récapitulatif des informations du joueur, la liste des adversaires et la possibilité de créer ou de rejoindre une partie. Elle hérite également de la classe *Observer* qui va nous permettre d'utiliser l'asynchronisme pour gérer l'affichage de la connexion/déconnexion d'un joueur adverse dans la *JTable* les répertoriant. De plus, elle sera composée de trois panels indépendants qui peuvent communiquer ensemble et être modifiés sans avoir besoin de recharger la vue complète. Par exemple, quand le joueur courant cliquera sur le pseudo d'un adversaire dans la *JTable*, le panel répertoriant ses informations sera automatiquement mis à jour et ce à chaque fois que nous cliquerons sur un nouvel adversaire. Enfin, elle implémente l'interface dont le module Réseau a besoin pour interagir avec nous.
- **HistoricalView** : Vue sensiblement semblable à *MainView* proposant un historique des parties.

Ces vues sont donc chargées à la volée dans la fenêtre principale, mais certaines auront besoin d'être instanciées dans leur propre *JFrame*. Cela va ainsi permettre de mettre en place un système de popup dans notre IHM pour toutes nos vues que nous qualifierons de temporaires. En effet, ces vues ne doivent pas être bloquantes pour le bon déroulement du jeu, c'est à dire la création d'une partie, le chargement d'une partie, la consultation des anciennes parties ou des profils des adversaires. C'est pourquoi nous avons décidé d'instancier ces vues dans leurs propres fenêtres, ainsi même si l'utilisateur est en train de modifier son profil il pourra tout de même recevoir une demande de partie de la part d'un autre joueur sur le réseau.

Néanmoins, ces popups ne sont pas totalement indépendantes de la fenêtre principale et de ses vues. En effet, il a fallu garder en lien entre eux afin que les informations contenues dans la *GameFrame* soient continuellement mises à jour. De plus, certains de ses attributs notamment comme le pointeur sur le Réseau doivent également être accessibles lorsque le joueur souhaite inviter un adversaire à jouer une partie. Ce lien est garanti en passant un pointeur sur la vue parente ayant instanciée la popup, qui lui même permet d'avoir accès aux informations de la *GameFrame*. Les popups de notre IHM sont les suivantes :

- **SetProfilePopup** : Cette popup est identique à *SetProfileView*. Dans *LogView* c'est la vue qui est chargée dans la fenêtre principale car la création d'un profil est nécessaire avant toute authentification. En revanche, une fois connecté il ne faut pas que la modification du profil bloque le bon déroulement du jeu, c'est pourquoi nous dupliquons cette vue dans une *JFrame* indépendante.
- **ResponseRequestPopup** : Popup instanciée quand un joueur reçoit une demande de lancement de partie.
- **WaitingResponsePopup** : Popup instanciée quand un joueur a demandé une nouvelle partie ou le chargement d'une sauvegarde.
- **CreateGamePopup** : Popup instanciée quand un joueur souhaite créer une nouvelle partie.

Une fois que toutes nos classes et le fonctionnement de chacune d'entre elles ainsi que des interfaces aient été approuvés par chacun des membres du groupe, nous avons fait une dernière réunion avec le groupe d'IHM grille afin de leur communiquer nos choix vu qu'ils avaient décidé d'utiliser notre système de gestion de vues à travers notre classe *GameFrame*. Après cette réunion, nous avons établie une ligne directrice de développement qui a d'abord commencé par une mise à niveau de chacun des membres au layout et autres particularités du développement d'interfaces graphiques avec SWING. Ensuite, notre chef de projet a décidé d'une organisation et d'une répartition des tâches équitables et en fonction de l'expérience de développement JAVA de chacun d'entre nous.

Les maquettes de chacune de nos vues et popups ont déjà été préalablement fournies dans le rapport de conception rédigée en début de semestre. Néanmoins, ces dernières seront reprises et commentées en fonction des changements apportés pendant la phase de développement dans la partie Développement de ce dossier de réalisation.

II. Développement

1. Outils et component

Nous avons décidé d'utiliser la bibliothèque graphique de java Swing comme bibliothèque graphique de base (consensus avec l'autre module IHM). Nous l'avons choisie pour des raisons de simplicité d'apprentissage. De plus, un grand nombre de personnes connaissait déjà cette bibliothèque et son utilisation. L'utilisation de Swing n'empêche pas l'utilisation d'autres bibliothèques telles que AWT qui sont complémentaires sur certains points et parfaitement compatibles. La bibliothèque Swing a été utilisée pour créer tous les éléments : fenêtres, vues et composants (bouton, champs...). La bibliothèque AWT a, quant à elle, été utilisée pour la gestion des événements et pour la disposition des composants dans les fenêtres.

Classe *PasswordFieldCustom* et *InformationField* :

Ce sont des champs de saisie de texte et de saisie de mot de passe modifiés qui intègrent en même temps un label permettant de décrire le champ de saisie. Le component *InformationField* hérite de *Jpanel* et comprend un *JLabel* et un *TextField* (Figure 1). Le component *PasswordFieldCustom* hérite de *Jpanel* et comprend un *JLabel* et un *JPasswordField*. Ils offrent une redéfinition de toutes les méthodes utiles comme la récupération du champ de saisie, le choix de la taille, etc. De plus, *PasswordFieldCustom* récupère les événements de type « utilisation de la touche Entrée » grâce à un *ActionListener* (qu'on peut passer en paramètre lors de sa création).

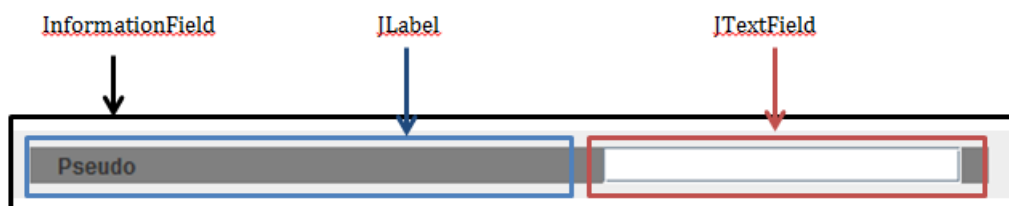


Figure 1 : Détail *InformationField*

Ils ont été utilisés dans plusieurs vues comme par exemple dans LogView, SetProfileView.

LobbyConstant :

Cette classe comporte toutes les variables constantes utiles au groupe IHM Connexion. Cette classe permet d'éviter l'utilisation de valeurs écrites en dur dans le code qui nuisent à la qualité du code, sont difficilement maintenables et peuvent être la source de nombreuses erreurs. Cette classe a notamment été utilisée pour définir les tailles de chaque fenêtre, les chemins d'accès aux fichiers de ressources ou encore pour définir des méthodes statiques qui peuvent être utilisées partout dans notre interface. Elle permet donc de centraliser les variables et les méthodes outil. Les variables sont définies comme des variables constantes et statiques. Ainsi, il n'est pas nécessaire d'instancier un objet pour y avoir accès.

Exemple de définition de variable constante :

```
/**
 * Default size of windows
 */
static final int DEFAULT_SIZE_H = 500;
static final int DEFAULT_SIZE_W = 500;
```

Exemple de définition de méthode :

```
/**
 * Static method to resize an image stored in one of packages
 * 'Resources'
 *
 * @param w The new width
 * @param h The new height
 * @param path The absolute path of the image
 * @return ImageIcon An image resizing
 */
static ImageIcon Resize(int w, int h, String path)
{
    Image tempo = new ImageIcon(path).getImage().getScaledInstance(w, h,
    java.awt.Image.SCALE_SMOOTH);
    return new ImageIcon(tempo);
}
```

Exemples d'utilisation :

```
public View()
{
    this(LobbyConstant.DEFAULT_SIZE_H, LobbyConstant.DEFAULT_SIZE_W, null);
}
```

```
_labelLogo = new JLabel(LobbyConstant.Resize(211, 224, this.getPath() +
LobbyConstant.getPathLogo()));
```

2. Création des interfaces

Dans cette partie, nous allons décrire l'ensemble des interfaces graphiques qui ont été réalisées pour l'IHM Connexion ainsi que leurs fonctions en comparant au travail de conception qui avait été fait.

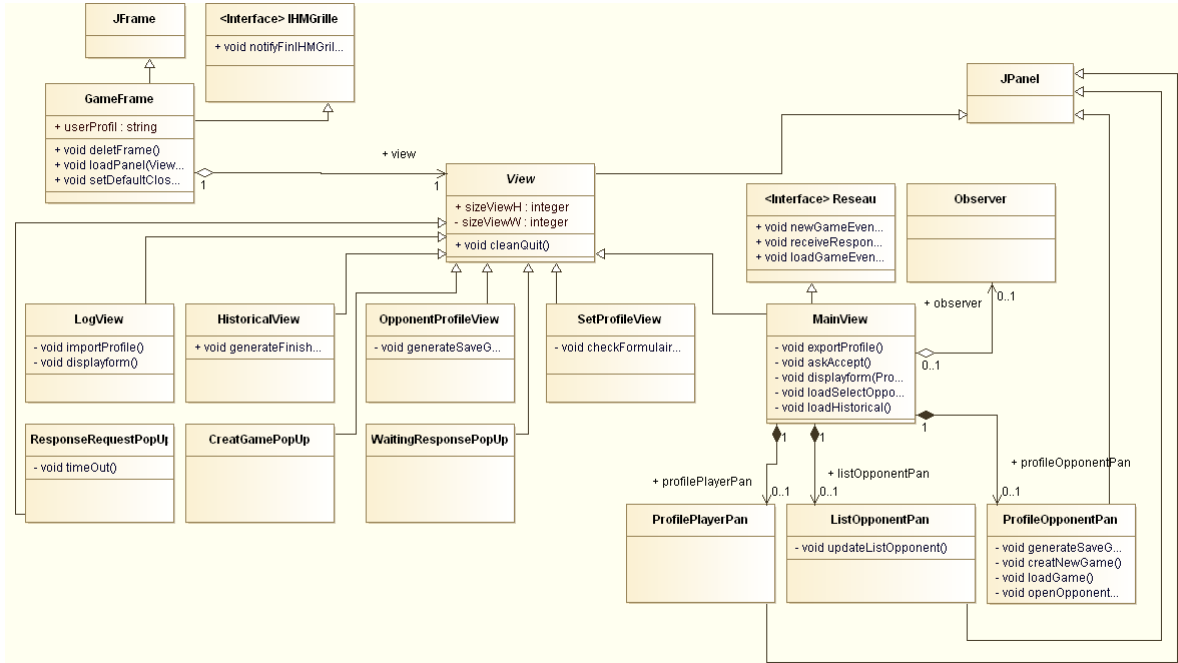


Figure 2 : Diagramme de classe Initial

Par rapport à la conception initiale, quelques modifications ont été apportées. La plupart des *View* est instanciée en tant que pop-up, ceci pour des raisons d'ergonomie. En effet, l'utilisation initiale des *View* devait se faire sous forme de fenêtre remplaçant la précédente, or toutes les fenêtres ne faisant pas la même taille, les fenêtres auraient disparues et réapparues avec des tailles différentes. Nous avons donc décidé d'utiliser des pop-up pour toutes les *View* qui doivent être ouvertes une fois la *MainView* chargée pour que l'effet visuel soit moins agressif. Toujours pour les mêmes raisons : éviter d'ouvrir et fermer les fenêtres ; et parce que les vues *MainView* et *HistoricalView* ont une construction très proche et ont un panel en commun, nous avons supprimé la vue : *HistoricalView* pour la transformer en deux panels qui viennent remplacer les panels de la *MainView* lorsque l'on souhaite afficher l'historique des parties. Les pop-ups seront des fenêtres instanciées en plus de la vue principale (*MainView*) invitant le joueur à effectuer des actions spécifiques.

LogView :

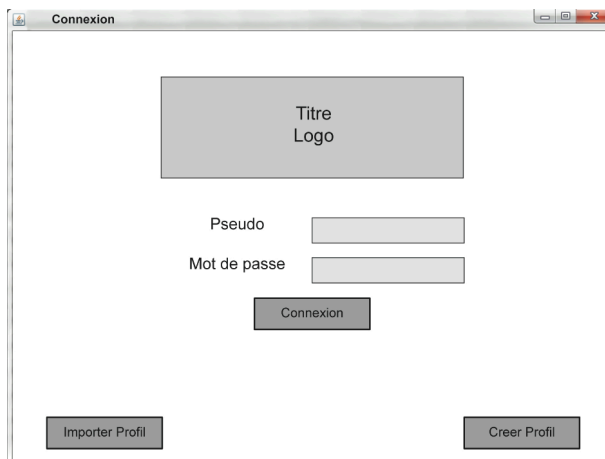


Figure 3 : Conception initiale LogView

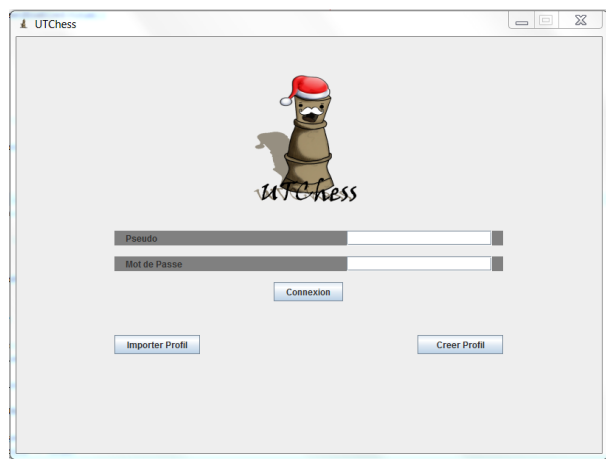


Figure 4 : Réalisation LogView

Cette vue correspond à l'interface de connexion. C'est la fenêtre qui s'ouvre lorsqu'on lance l'application. Elle est constituée de :

- 2 champs à renseigner :
 - Pseudo
 - Mot de passe
- 1 champ de notification (Figure 5) : il informe l'utilisateur d'éventuelles erreurs, comme le renseignement d'un mauvais couple de pseudo/mot de passe, d'erreur d'importation ou la confirmation d'action comme l'importation d'un profil (Figure 7). Ce champ de figurait pas dans la conception initiale. Il a été ajouté pour faciliter la compréhension de l'utilisateur vis-à-vis des actions qui s'exécutent.

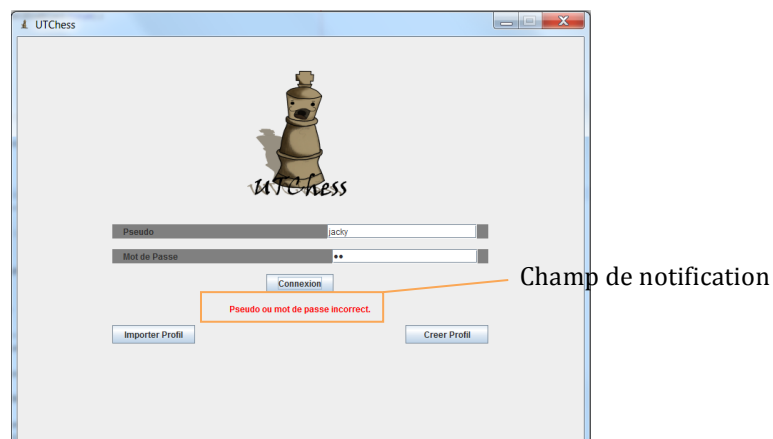


Figure 5 : champ de notification

- 3 boutons :
 - Connexion : l'utilisation du bouton entraîne la vérification des champs *Pseudo* et *Mot de passe* (Interface avec Data). S'ils sont concordants avec un des profils stockés sur l'ordinateur, une connexion réseau sera établie

(Interface avec Réseau), puis l'utilisateur sera redirigé vers la vue principale (MainView).

- Importer Profil : le bouton ouvre une fenêtre pop-up permettant de sélectionner un dossier (Figure 6). L'importation permet d'importer un dossier contenant toutes les données d'un profil. Plusieurs vérifications sont effectuées pour que le fichier contienne bien les données du profil. Puis le dossier est déplacé à l'emplacement où sont les autres profils (Interface avec data). L'utilisateur peut ensuite se connecter avec le profil importé. Toutes erreurs concernant l'importation sont mises dans le champ de notification (Figure 7).

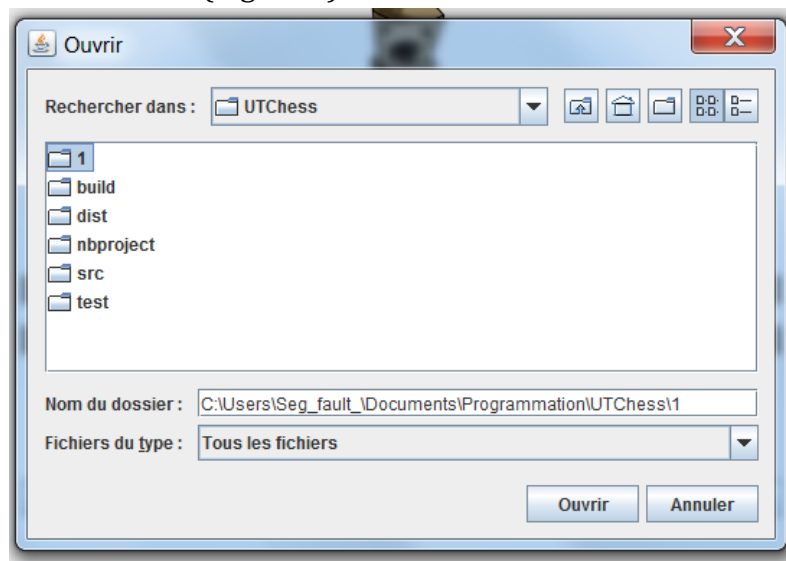


Figure 6 : Fenêtre pop-up d'importation

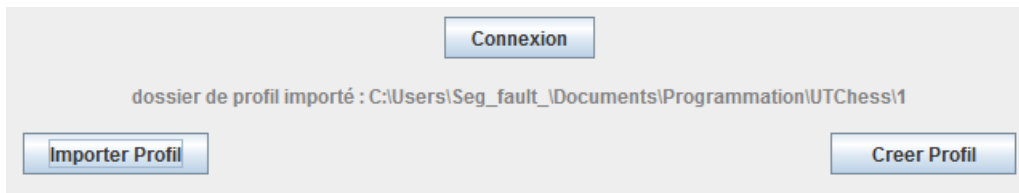


Figure 7 : notification de réussite d'importation

- Créer Profil : Le bouton redirige l'utilisateur vers la vue permettant de créer (et modifier) un profil (SetProfileView)

SetProfileView :

Creer/Modifier Profil

Votre Profil :

Nom


Prénom

Age

Pseudo

Mot de passe

Mot de passe

Avatar 

Avatar

Statistiques

Parties Gagnées : 0

Parties Perdues : 0

Ratio : x

...

Enregistrer Annuler

Figure 8 : Conception initial SetProfileView

UTChess

Votre Profil :

Nom


Prénom

Age

Pseudo

Mot de passe

Confirmation

Avatar 

Avatar

Statistiques

Parties Gagnées : 0

Parties Perdues : 0

Ratio : x

...

Annuler Enregistrer

Figure 9 : Réalisation SetProfileView

La vue est utilisée à la fois pour création et la modification d'un profil. Néanmoins les champs seront pré-remplis pour la modification. De plus la vue est instanciée dans une fenêtre dans le cas de la création (c'est-à-dire qu'elle vient remplacer celle de connexion) alors qu'elle est instanciée dans un pop-up lors de la modification.

Cette vue est composée de :

- 6 champs à renseigner :
 - Nom
 - Prénom
 - Pseudo
 - Nouveau Mot de passe
 - Confirmation du nouveau mot de passe
 - Age
 - Avatar : le choix se fait dans une liste déjà prédéfinie.
- Image représentant l'avatar du joueur
- Un champ de notification, il permet de donner des instructions ou d'informer sur des erreurs dans les champs (par exemple erreur de confirmation de mot de passe, mot de passe trop court, champ *Age* contenant des caractères non numérique...). Ce champ de figurait pas dans la conception initial. Il a été ajouté pour faciliter la compréhension de l'utilisateur vis-à-vis des actions qui exécutent.
- 2 boutons :
 - Enregistrer : Nous vérifions si les champs sont valides, si oui nous modifions/créons le profil de l'utilisateur et nous redirigeons l'utilisateur vers la vue principale (MainView). Lors de la création, une étape création du dossier de profil (Interface avec Data) puis de connexion au réseau sont effectué avant l'ouverture de la fenêtre principal. Sinon un message d'erreur est affiché à l'utilisateur par le biais du champ de notification.

- Annuler : L'utilisateur sera redirigé vers la vue précédente (*LogView* s'il était en train de créer un profil ou *MainView* s'il était en train modifier son profil). Aucune modification n'est enregistrée.
- Les informations sur les statistiques du joueur ont été retirées car elles sont déjà visibles dans le panel du profil de la fenêtre principale.

MainView :

La fenêtre principale à deux apparences suivant elle est utilisée comme accueil ou pour afficher l'historique des parties. Etant donné que la second utilité avait été conçu comme une vue a par entière nous parlerons de ces modification que dans la partie sur l'ancienne fenêtre qui abritait cette fonctionnalité : *HistoricalView*.

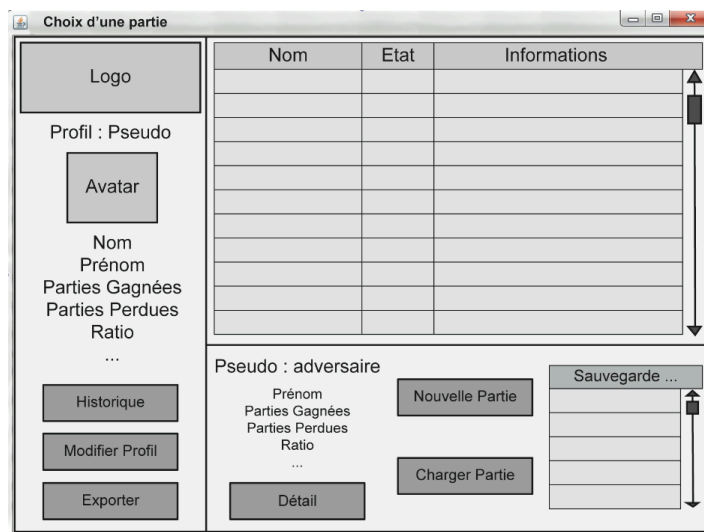


Figure 10 : Conception initial MainView

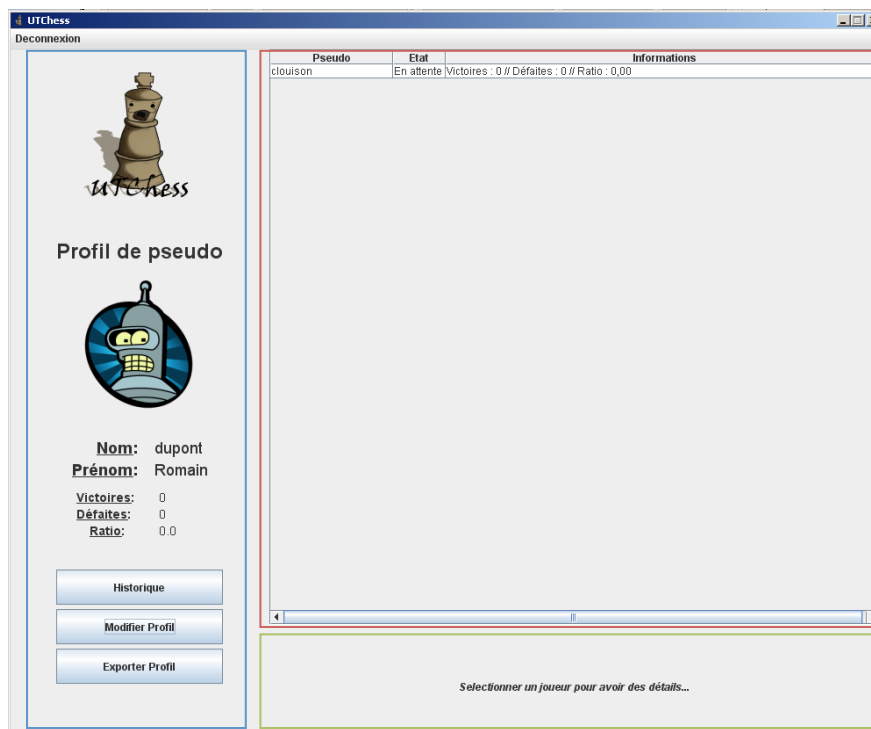


Figure 11 : Réalisation Mainview (1 - Accueil)

Legende :

- Panel
- ProfilePlayerPan
- Panel
- ListOpponentPan
- Panel
- ProfileOpponentPan

Cette vue correspond à la vue principale de l'IHM Connexion elle deux apparence : Accueil et Historique des parties (voir partie HistoricalView). L'apparence servant d'accueil est composé de trois parties : une où les informations du joueur sont récapitulées (ProfilePlayerPan), la liste des adversaires potentiels (les autre joueur connecté) (ListOpponentPanel) et l'affichage des détails sur un adversaire ainsi que la possibilité de créer une partie(ProfilOppnentPan). Le choix de ces trois parties indépendantes nous a permis de nous partager le travail :

1. Profil du joueur courant : ProfilePlayerPan

- Informations personnelles du joueur
- 3 boutons :
 - Historique : permet de passer la MainView en apparence Historique des parties (voir partie HistoricalView pour plus de détail).
 - Modifier Profil : le bouton ouvre une fenêtre pop-up contenant la vue de modification (/création de profil) (SetProfileView).
 - Exporter : le bouton ouvre une fenêtre pop-up permettant de sélectionner le chemin l'emplacement ou le profil doit être exporté. Une fois que l'utilisateur à sélectionné l'emplacement, le dossier du profil est copié a l'emplacement demandé (Interface avec Data).

2. Liste des adversaires connectés (ListOpponentPanel): Sélection de l'adversaire avec le clic gauche de la souris.

- Pseudo
- Etat : En attente ou En jeu
- Informations : Parties gagnées/Parties jouées/Ratio, Le joueur peut ainsi connaitre facilement le niveau de cet adversaire et faciliter son choix.

3. Profil de l'adversaire sélectionné (ProfilOppnentPan). Ce panel contient :

- Informations du joueur adverse
- Liste des sauvegardes
- 3 boutons :
 - Détails : le bouton ouvre une fenêtre pop-up contenant la vue OpponentProfileView. Elle contient toutes les informations du profil du joueur adverse sélectionnée. On peut ainsi accéder à toutes les informations du joueur adverse sans surcharger la pré-vue.
 - Nouvelle Partie : Ouverture d'une fenêtre pop-up (CreateGamePopup) demandant à l'utilisateur les paramètres de partie (couleur, timer). Une fois les informations validées, la demande de partie est envoyée au joueur adverse par le réseau (interface avec Réseau). En attendant la réponse de l'adversaire (positive ou négative), une fenêtre pop-up d'attente

(WaitingResponsePopup) est instanciée. Elle est bloquante et empêche l'utilisateur de revenir sur la vue principale (MainView) tant qu'une réponse n'a pas été reçue.

- **Charger Partie :** Une fois la sauvegarde sélectionnée et validée par l'utilisateur, la demande de reprise de partie est envoyée au joueur adverse par le réseau (Interface avec Réseau). En attendant la réponse de l'adversaire (positive ou négative), une fenêtre pop-up d'attente (WaitingResponsePopup) est instanciée. Elle est bloquante et empêche l'utilisateur de revenir sur la vue principale (MainView) tant qu'une réponse n'a pas été reçue.

De plus, il est intéressant de noter que cette fenêtre est toujours active après la phase de connexion, elle se désactive uniquement lors d'un lancement de partie (Interface avec IHMGrille), d'une fermeture forcée (croix) ou la déconnexion qui entraîne le retour à la fenêtre de connexion. Pour toutes les autres vues sont instancié dans des fenêtres pop-up, afin de ne pas perdre la vue principale (MainView) en cas de demande de nouvelle partie lors de la consultation d'une de ces vues.

OpponentProfileView :

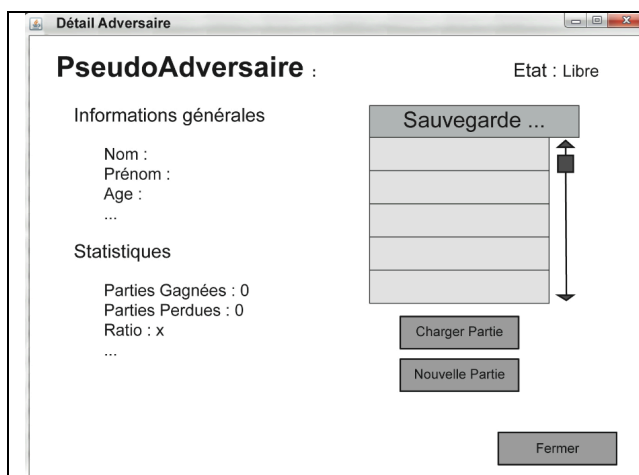


Figure 12 : Conception initial OpponentProfileView

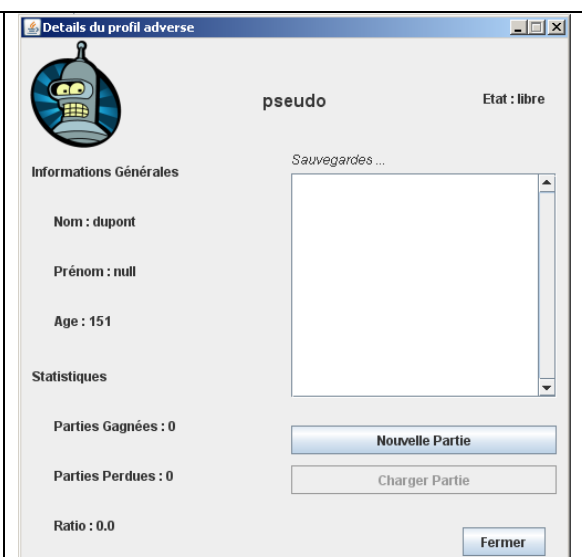


Figure 13 : Réalisation OpponentProfileView

Cette vue s'ouvre dans une fenêtre pop-up. Elle contient l'intégralité des informations de l'adversaire ainsi que la possibilité de créer ou de charger une partie. Elle est composée de :

- L'intégralité des informations publiques de l'adversaire
- Gestion des sauvegardes communes
 - L'Etat du joueur actuellement : En ligne ou Hors ligne
 - Une liste des sauvegardes

- 3 boutons :
 - Charger Partie : Une fois la sauvegarde sélectionnée et validée par l'utilisateur, la demande de reprise de partie est envoyée au joueur adverse par le réseau (Interface avec Réseau). En attendant la réponse de l'adversaire (positive ou négative), une fenêtre pop-up d'attente (WaitingResponsePopup) est instanciée. Elle est bloquante et empêche l'utilisateur de revenir sur la vue principale (MainView) tant qu'une réponse n'a pas été reçue.
 - Nouvelle Partie : Ouverture d'une fenêtre pop-up (CreateGamePopup) demandant à l'utilisateur les paramètres de partie (couleur, timer). Une fois les informations validées, la demande de partie est envoyée au joueur adverse par le réseau (interface avec Réseau). En attendant la réponse de l'adversaire (positive ou négative), une fenêtre pop-up d'attente (WaitingResponsePopup) est instanciée. Elle est bloquante et empêche l'utilisateur de revenir sur la vue principale (MainView) tant qu'une réponse n'a pas été reçue.
 - Fermer : Ce bouton ferme la fenêtre pop-up.

HistoricalView :

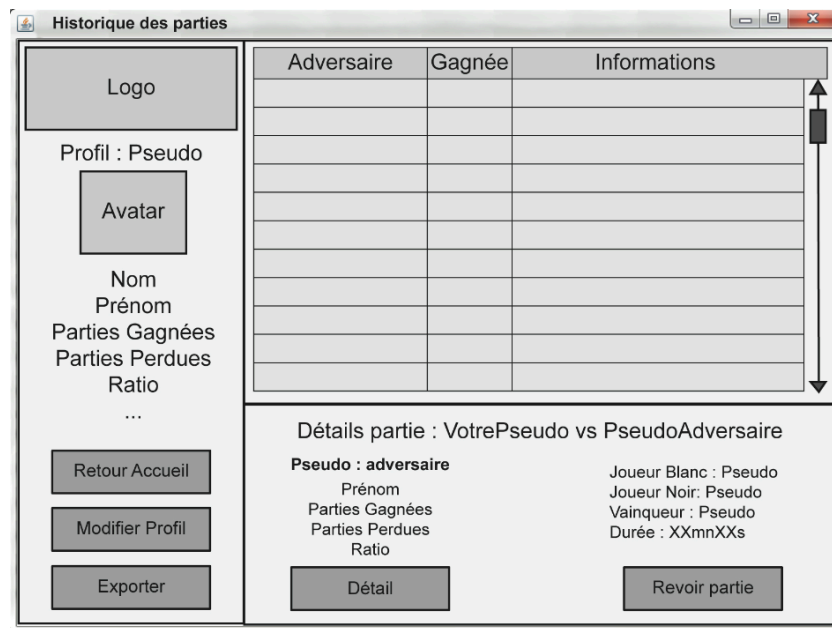


Figure 14 : Conception initial MainView

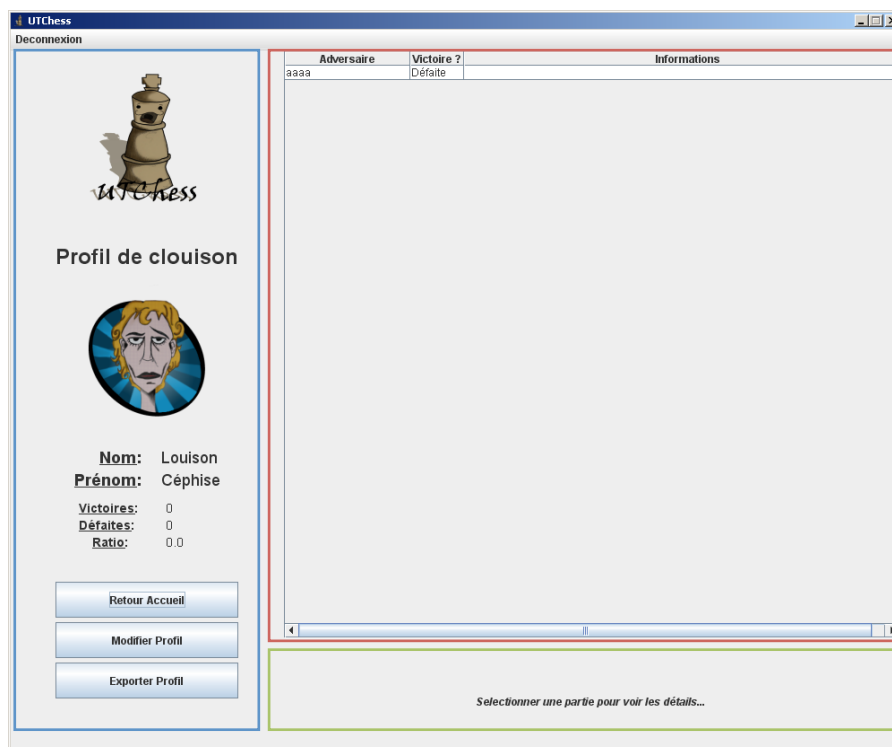


Figure 15 : Réalisation MainView (2 - Historique des parties)

Anciennement (pendant la conception) définie comme une vue a part entière qui venais remplacer la vue principale MainView. Elle a été réalisée pour faire partie dans la Vue principal. En effet, désormais le panel ListOpponentPan (de l'affichage Accueil) est remplacé le panel HistoricalList (en affichage historique) et le panel ProfileOpponentPan par le panel HistoricalDetail lorsque le joueur utilise le bouton Historique de l'affichage Accueil de ma MainView. Le panel ProfilePlayerPan n'ai pas remplacé seul le bouton Historique est modifier pour permettre de retourné à l'affichage de l'Accueil L'objectif

de cette affichage est de contenir un historique à jour de toutes les parties que l'utilisateur a disputé en vue de les revoir. Elle contient trois parties indépendantes:

1. Profil du joueur courant (ProfilePlayerPan)
 - Informations personnelles du joueur
 - 3 boutons :
 - Retour Accueil : permet de passer la MainView en apparence Accueil (voir la partie MainView pour plus de détail).
 - Modifier Profil : le bouton ouvre une fenêtre pop-up contenant la vue de modification (/création de profil) (SetProfileView).
 - Exporter : le bouton ouvre une fenêtre pop-up permettant de sélectionner le chemin l'emplacement ou le profil doit être exporté. Une fois que l'utilisateur a sélectionné l'emplacement, le dossier du profil est copié a l'emplacement demandé (Interface avec Data).
2. Liste des parties (HistoricalList): Sélection de la partie avec le clic gauche de la souris.
 - Adversaire : Pseudo de l'adversaire
 - Gagnées : Victoire ou Défaite selon si le joueur a gagné ou perdu la partie.
 - Informations : Couleur/Timer/Date
3. Profil de l'adversaire sélectionné (HistoricalDetail). Ce panel contient :
 - Informations du joueur adverse avec qui on avait joué
 - Information sur la partie
 - 2 boutons :
 - Détails : le bouton ouvre une fenêtre pop-up contenant la vue OpponentProfileView. Elle contient toutes les informations du profil du joueur adverse. On peut ainsi accéder à toutes les informations du joueur qu'on avait affronté.
 - Revoir Partie : Bouton permettant de revoir la partie jouée entre les deux joueurs. L'utilisation ce se bouton ouvre la fenêtre de revue de partie à la place de la vue principal MainView (interface avec IHMGrille).

ResponseRequestPopup :

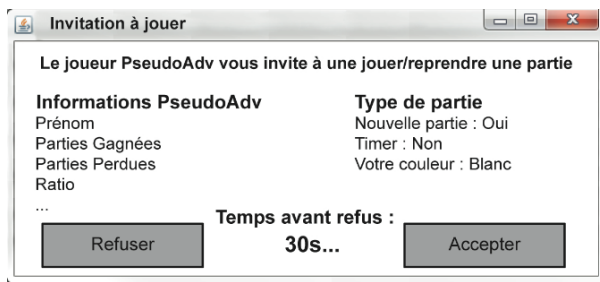


Figure 16 : Conception initial ResponseRequestPopup

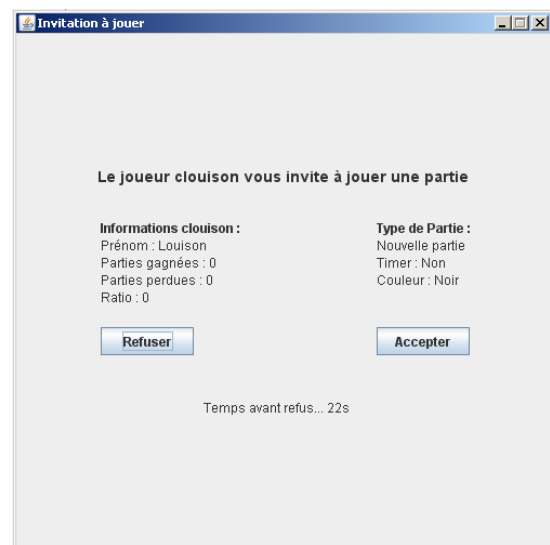


Figure 17 : Réalisation ResponseRequestPopup

Cette fenêtre pop-up est instanciée lorsqu'un joueur reçoit une demande de lancement de partie venant d'un adversaire. Elle contient :

- Informations sur la partie souhaitée par l'adversaire :
 - Informations sur le joueur adverse
 - Informations sur la partie
- Acceptation de la partie :
 - Un champ représentant le temps restant avant le refus automatique de la partie. Un temps maximum de 30 secondes va permettre de laisser suffisamment de temps à l'adversaire pour réagir et ne pas bloquer l'autre joueur trop longtemps.
 - 2 boutons :
 - Accepter : En cliquant sur ce bouton, le joueur accepte la partie ce qui entraîne l'ouverture de la grille à la place de la fenêtre principal (interface avec IHM Grille) et l'envoi de la réponse à l'autre joueur (interface avec Réseau).
 - Refuser : Le joueur refuse la partie la fenêtre pop-up se ferme et un message est envoyé par le réseau pour informer l'autre joueur du refus (Interface avec Réseau)

WaitingResponsePopup :

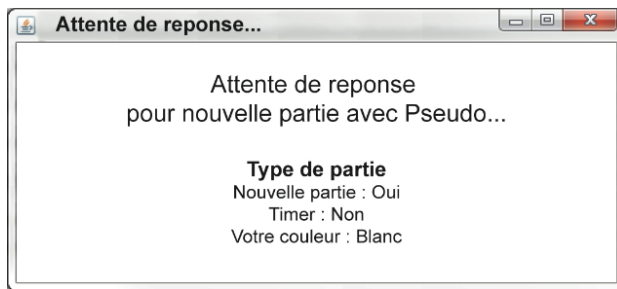


Figure 18 : Conception initial WaitingResponsePopup

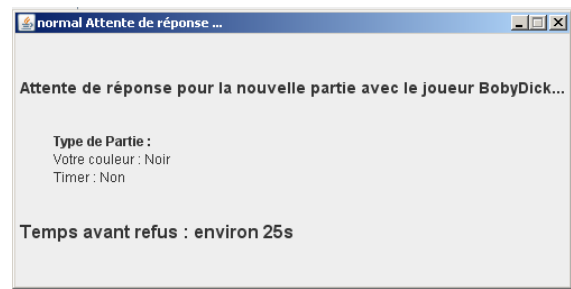


Figure 19 : Réalisation WaitingResponsePopup

Cette fenêtre pop-up est instanciée lorsqu'un utilisateur effectue la demande d'une nouvelle partie ou le chargement d'une sauvegarde. Elle est constituée d'un unique panel informant d'une part l'attente d'une réponse de l'adversaire et d'autre part les détails de la partie souhaitée. Il est également intéressant de noter que lorsque cette fenêtre pop-up est activée, la vue principale (MainView) est bloquée à toute interaction tant qu'une réponse positive ou négative n'est pas reçue.

CreateGamePopup :

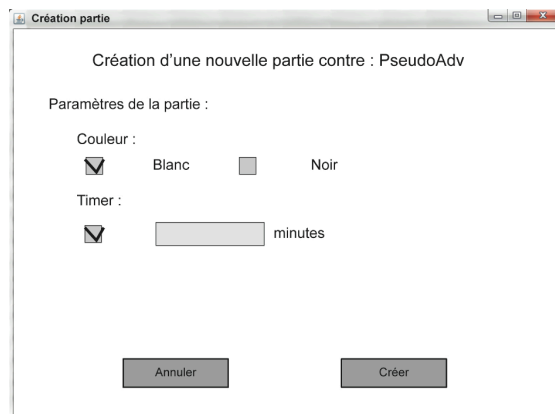


Figure 20 : Conception initial CreateGamePopup

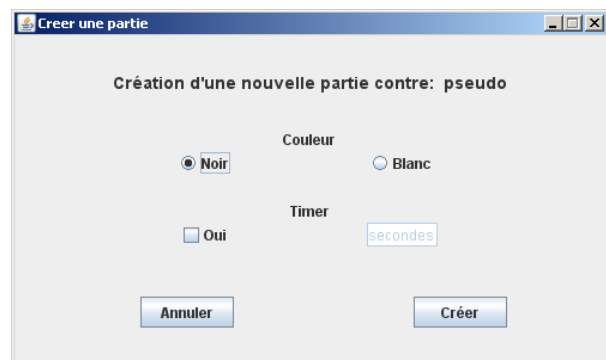


Figure 21 : Réalisation CreateGamePopup

Cette fenêtre pop-up est instanciée quand l'utilisateur souhaite créer une nouvelle partie avec un adversaire préalablement sélectionné dans la liste proposée. Elle contient :

- 2 séries de boutons radio :
 - La couleur des pièces (noir ou blanc)
 - La présence d'un timer durant la partie ainsi que sa durée
- 2 boutons :
 - Créer : le bouton valide la création de la partie. La fenêtre pop-up se fermera et une demande de partie sera envoyée au joueur adverse (interface avec Réseau), puis une fenêtre pop-up WaitingResponsePopup sera instanciée en attendant la réponse.
 - Annuler : le bouton entrainera la fermeture de la fenêtre pop-up. Aucune action vers le joueur adverse ne sera effectuée.

3. Difficultés rencontrées

Nous avons rencontré des difficultés dans l'utilisation de certaines parties des bibliothèques graphiques.

Répartition des éléments dans les fenêtres

Nous avons rencontré des difficultés à faire en sorte que la fenêtre développée respecte les fenêtres définies à la conception. En cause la difficulté d'utilisation des layouts ou leur manque de flexibilité. La solution que nous avons trouvée réside dans l'utilisation de GridBagLayout associé avec GridBagConstraints de la bibliothèque graphique AWT. Ce layout permet de diviser l'espace d'affichage en grille (comme les layouts grille), mais avec possibilité d'associer de nombreuses contraintes (position, taille des cases, marge des cases). Si les possibilités offertes par ce layout nous ont permis d'avoir le rendu souhaité, il a aussi été difficile à prendre en mains et a demandé aux membres du module de se former pour l'utiliser comme il convient.

Mise à jour des fenêtres

Nous avons souhaité faire une interface assez dynamique. Nous avons par exemple souhaité afficher les détails d'un joueur connecté dans un composant plutôt que de directement afficher les informations dans la liste. Cela a entraîné des difficultés à mettre à jour le contenu de certains éléments dynamiquement. En effet, certains composants graphiques ont du mal à se mettre à jour automatiquement. Nous avons donc dû mettre en place un rafraîchissement des composants. Le plus gros problème est venu des listes comme la liste des joueurs connectés qui doit se mettre à jour : lorsqu'un joueur se connecte ou se déconnecte et lorsque son état change. Nous avons finalement réussi à mettre les valeurs contenues dans ces éléments sans avoir à refaire l'intégralité de la vue.