Projet Picross
Dossier de conception

Groupe C

30 avril 2016

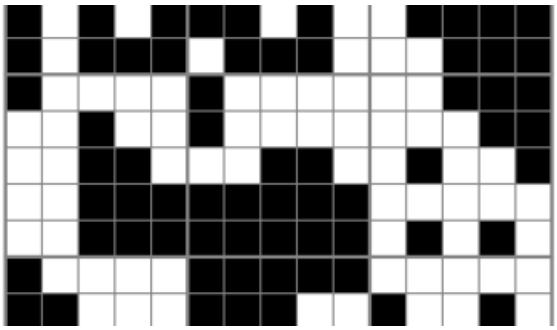

Table des matières

I Présentation générale

I.1 Rappel contexte général

Dans le cadre de la Licence Sciences Pour l'Ingénieur, les étudiants de troisième année se sont vu confié le développement d'un logiciel permettant de jouer au Picross (aussi connu sous le nom de nonogramme ou logigramme).

Le groupe est composé de Yannick ESSIA, Wajdi GUEDOUAR, Yann GUENVER, Mohammed Amine HAJIR, Youssef LAMNIY, Choukri SOULEIMAN IMAN et Luc FONTAINE élu chef de projet par le groupe.

I.2 Objectif du dossier de conception

L'objectif de ce document est de présenter la partie conception orienté objet, l'interface homme-machine ainsi que la conception de la base de donnée.

I.3 Description de l'application

Il s'agit d'un jeu solitaire de logique et de réflexion dont le but est de résoudre une grille à partir de listes de nombres placées en indices en haut et à gauche de la grille. Un Picross se présente sous la forme d'une grille de cases qu'il faudra noîrcires ou laisser blanches pour le résoudre.

Chaque indice représente le nombre de cases noires contiguës qui constituent chaque bloc, sachant qu'il faut au moins une case blanche entre chaque bloc. L'application permet d'être accompagné lors de la réalisation du puzzle par des astuces et d'indices.

II Analyse fonctionnelle

II.1 Spécifications IHM

Dans le cadre de ce projet, il était convenu que l'IHM utilise le gem GTK2 pour Ruby.La figure ci-dessous représente les liens entre les différentes fenêtres principales de l'application.

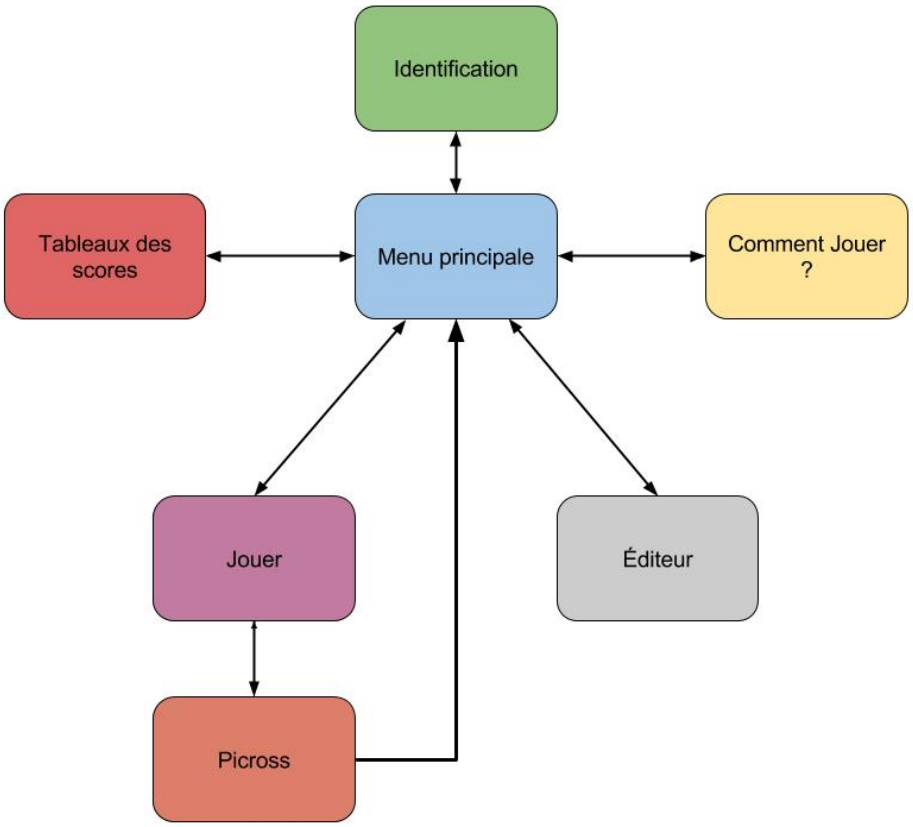

FIGURE 1 – Schéma de navigation

II.2 Spécifications du domaine métier

Dans la figure suivante, on dispose du cas d'utilisation de l'application. On compte un utilisateur pour lequel l'identification avec un pseudonyme est nécessaire pour jouer.

Puis à partir de l'identification ce dernier aura accès au jeu dont il a la capacité de sauvegarder, consulter des aides, charger une partie ou créer une grille de Picross en l'éditant personnellement.

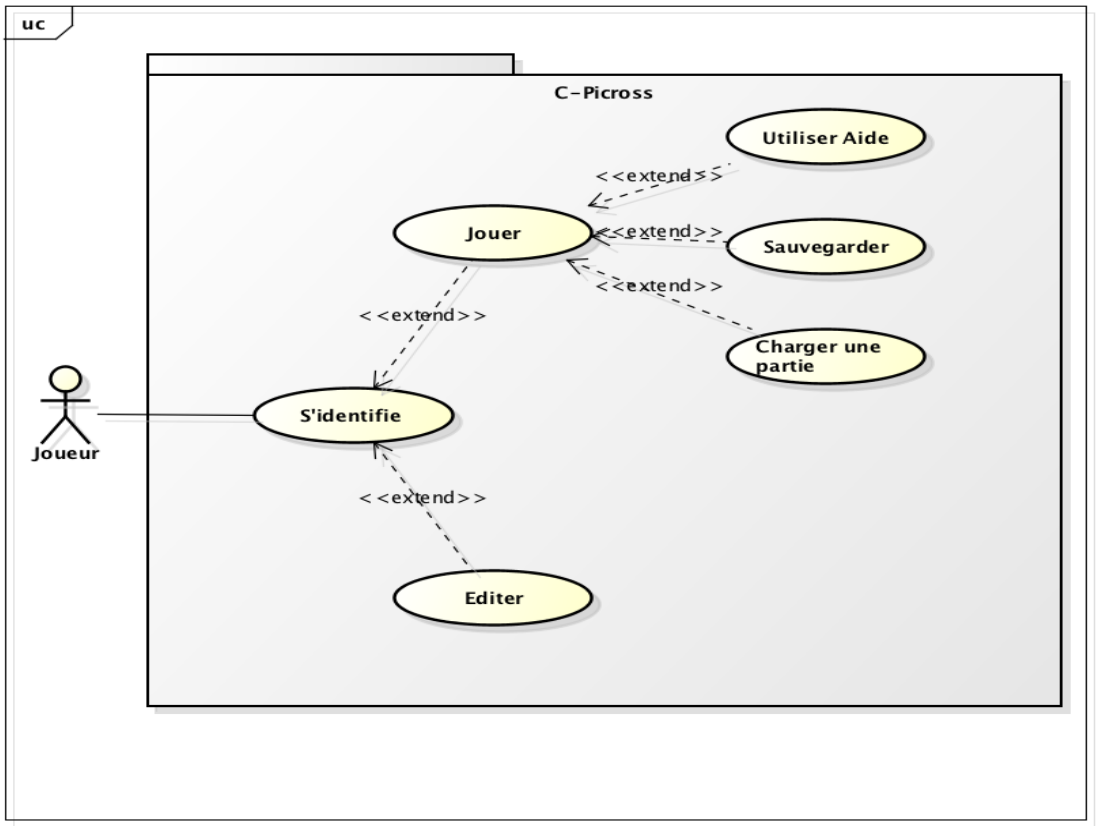

FIGURE 2 – Diagramme des cas d'utilisations

II.3 Spécifications des données et persistance

Nous avons eu le choix pour la persistance des données entre la sauvegarde en xml, fichier binaire, fichier texte et SQLite3.

Cependant nous avons opté pour le choix d'utilisation de SQLite3 qui est une base de donnée locale ,portable et facile à mettre en place. Ci-dessous le modèle entité association décrivant les relations entre les objets sauvegardés dans la base de donnée.

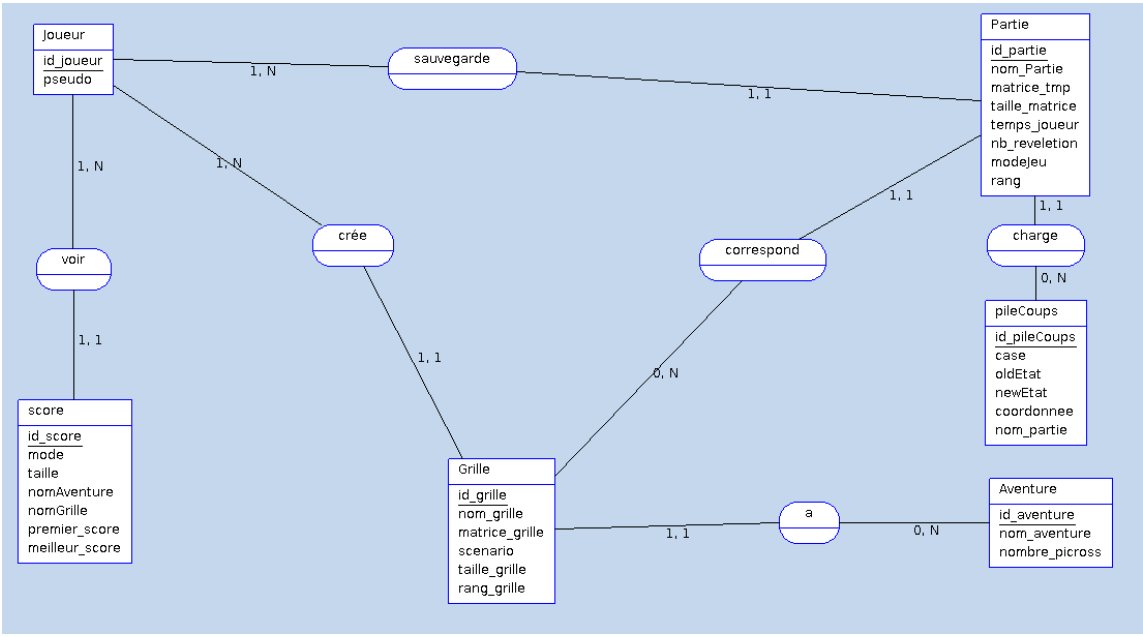

FIGURE 3 – Modèle entité association de la base de donnée

Modèle logique des données :

- Joueur(id_joueur, pseudo)
- Score(id_score, taille, nomAventure, nomGrille, mode, premier_score, meilleur, meilleur_score, #id_joueur)
- Aventure(id_aventure, nom_aventure, nombre_picross)
- Grille(id_grile, nom_grille, matrice_grille, taille_grille, rang_grille, #id_joueur, #id_aventure)

- PileCoups(id_pileCoups, case_coups, oldEtat, newEtat, coordonnee, nom_partie)
- Partie(id_partie,nom_partie, matrice_tmp, taille_matrice,#id_pileCoups, temps_joueur, nb_revalation, #id_grille, #id_joueur,mode)

III Conception technique

III.1 Architecture technique de l'application

L'application a été organisée selon le modèle MVC (Modèle Vue Contrôleur).

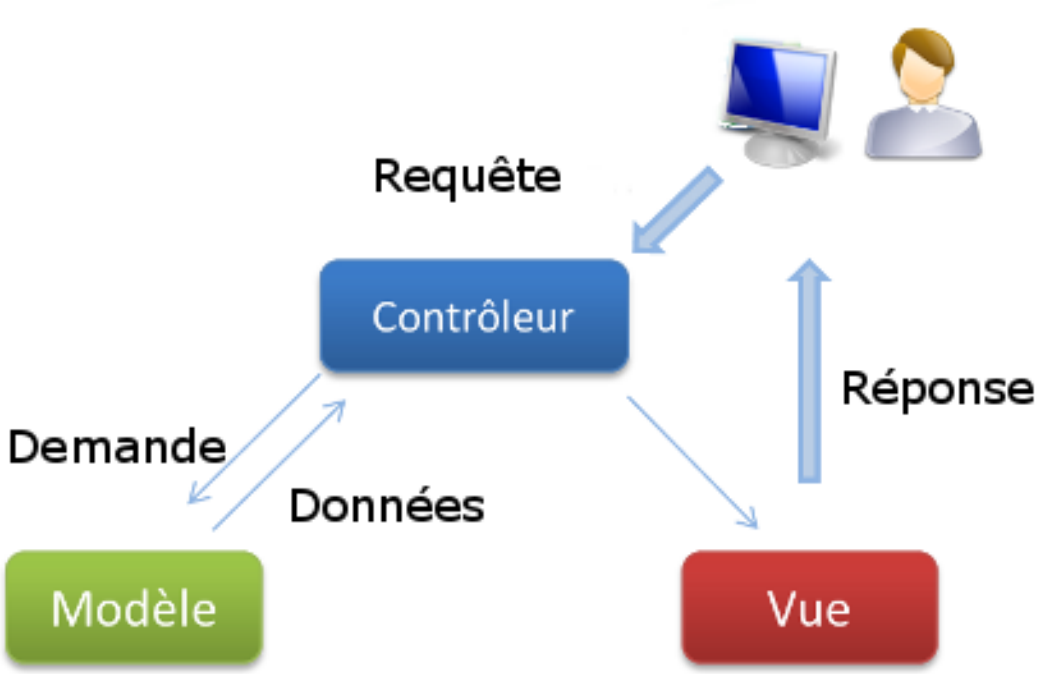

FIGURE 4 – Modèle de l'architecture de l'aaplication

III.2 Moteur du jeu

Ci-dessous le diagramme de classe reprenant la totalité des méthodes utilisé pour lancer le moteur du jeu.

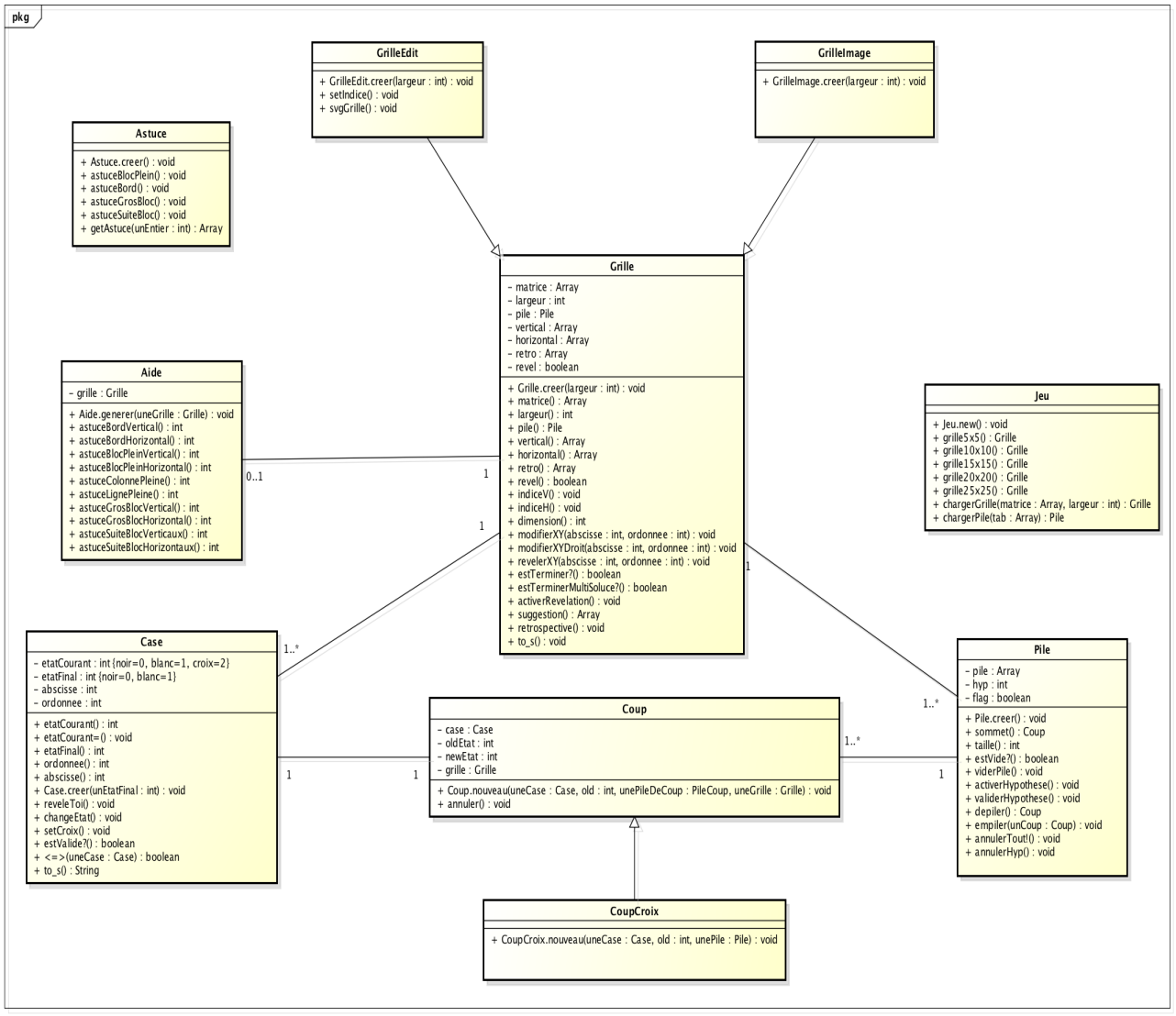

FIGURE 5 – Diagramme de classe du moteur du jeu

Le moteur du jeu est la partie Modèle du MVC, nous allons décrire dans cette partie l'organisation des entités ainsi que leurs fonctionnalités.

L'élément centrale du moteur de notre jeu est la **Grille**, cet objet permet d'accéder ou de générer tous les objets nécessaires à la création d'une partie.

La **Grille** permet de générer une matrice, des indices, de jouer des coups, de tester si la partie est terminée, d'avoir accès à des aides et à tout les éléments qu'elle contient. Nous étudierons donc la conception de cette **Grille** couche par couche.

Lors de sa création, la matrice contenu dans la **Grille** est remplie de **Case** dont l'état courant est vide. Chaque **Case** connaît son état final ce qui nous permet de savoir à quoi doit ressembler la matrice une fois terminée.

Une **Case** connaît également ses coordonnées dans la **Grille**. Deux variables d'instances de la **Grille** : vertical et horizontal, sont les matrices contenant les indices présents en bordure de la **Grille** lors d'une partie de picross. Ces indices sont calculés et placés dans les variables par les méthodes indiceV et indiceH.

Une **Pile** de **Coup** est crée et placée dans la variable d'instance pile.

Un **Coup** est une action réalisée par le joueur, il en existe deux types : un **Coup** de base qui change l'état de la **Case** en blanc ou en noir, un **CoupCroix** qui permet au joueur de placé une croix sur la matrice.

Nous avons choisit de différencier ces deux types de Coups afin de faciliter les contrôles du jeu. Un **Coup** connaît donc la **Case** modifiée par celui-ci, l'état de la **Case** avant qu'elle ne soit modifiée, son futur état suite à la modification et la **Grille** afin de pouvoir retrouver la **Case** via ses coordonnées ; ce qui est utile lorsque l'on charge une partie et que les objets on dût être recréé.

Il existe que deux fonctionnalités : une pour la création du **Coup** et une pour l'annulation. Lors de la création, le **Coup** initialise les valeurs de ses variables d'instances et s'empile automatiquement dans la **Pile** passée en paramètre. L'annulation du **Coup** restaure l'ancien état de la Case.

La **Pile** de **Coup** est une **Pile** qui va contenir tout les Coups joués par le joueur depuis le début de la partie. La particularité de cette **Pile** réside dans le fait que dépiler un **Coup** entraîne l'annulation de celui-ci.

L'objet **Grille** a une méthode permettant de rechercher des Coups jouables pour le joueur, les aides générées indiquent au joueur la ligne/colonne où le **Coup** peut être joué ainsi qu'un texte/GIF expliquant comment le réalisé.

Il existe plusieurs types de **Grille** :

- la **Grille** de base nous permettant de jouer une partie dont la matrice a été générée aléatoirement,
- la **GrilleEdit** est une simple grille dont toutes les Cases sont vierges (état courant et état final sont à vide) afin que le joueur puisse les modifiées à sa guise avant de sauvegarder la **Grille** et de la jouer ultérieurement,
- la **GrilleImage** permettant de passer une matrice en paramètre afin que la matrice de l'objet **Grille** y soit identique ce qui permet de charger une **Grille** depuis la base de donnée ou depuis une image.

Enfin, un objet Jeu contiendra des méthodes encapsulant la création des Grilles, ce qui permet à la partie Contrôle de n'avoir à interagir qu'avec un objet lors des créations. Le Jeu permet de créer des Grilles mais aussi de les charger.

III.3 Conception de l'interface

L'interface du jeu est la partie Vue et Contrôleur du MVC. Chaque vue est liée à un contrôleur, ce dernier prend en charge une fonctionnalité donnée.

Les contrôleurs héritent d'une classe Contrôleur, de même pour les vues (héritent de Vue), ainsi cela nous permet de changer facilement le contenu d'une vue ou d'un contrôleur c'est à dire leur attribuer différentes fonctionnalité sans pour autant modifier la structure du jeu.

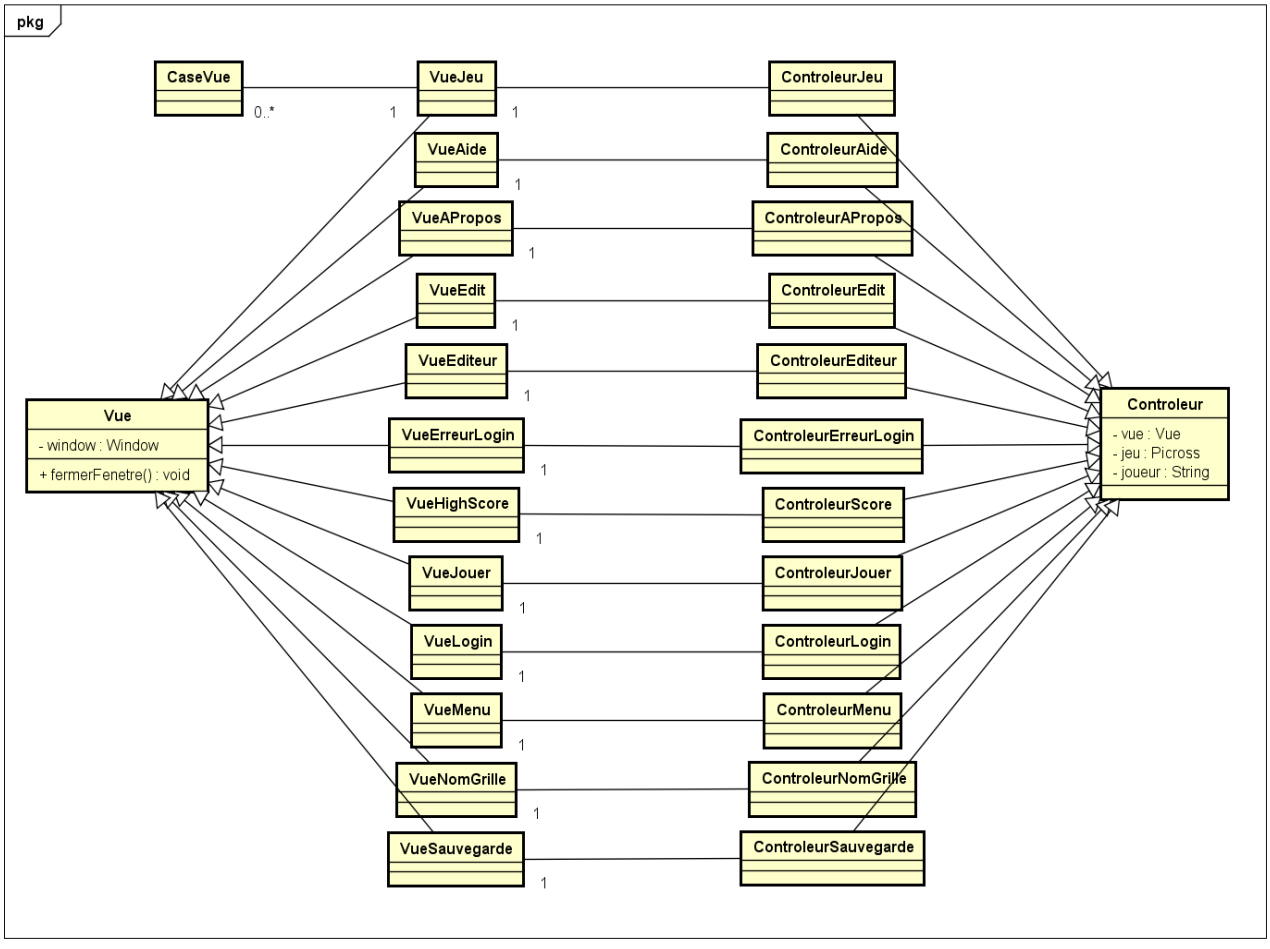

FIGURE 6 – Diagramme de classe de la réalisation de l'interface

III.4 Conception des données

Ci-dessous le diagramme de classe recensant toutes les méthodes de la base de donnée et la classe nécessaire à la création des aventures, cette partie représente le modèle dans le MVC.

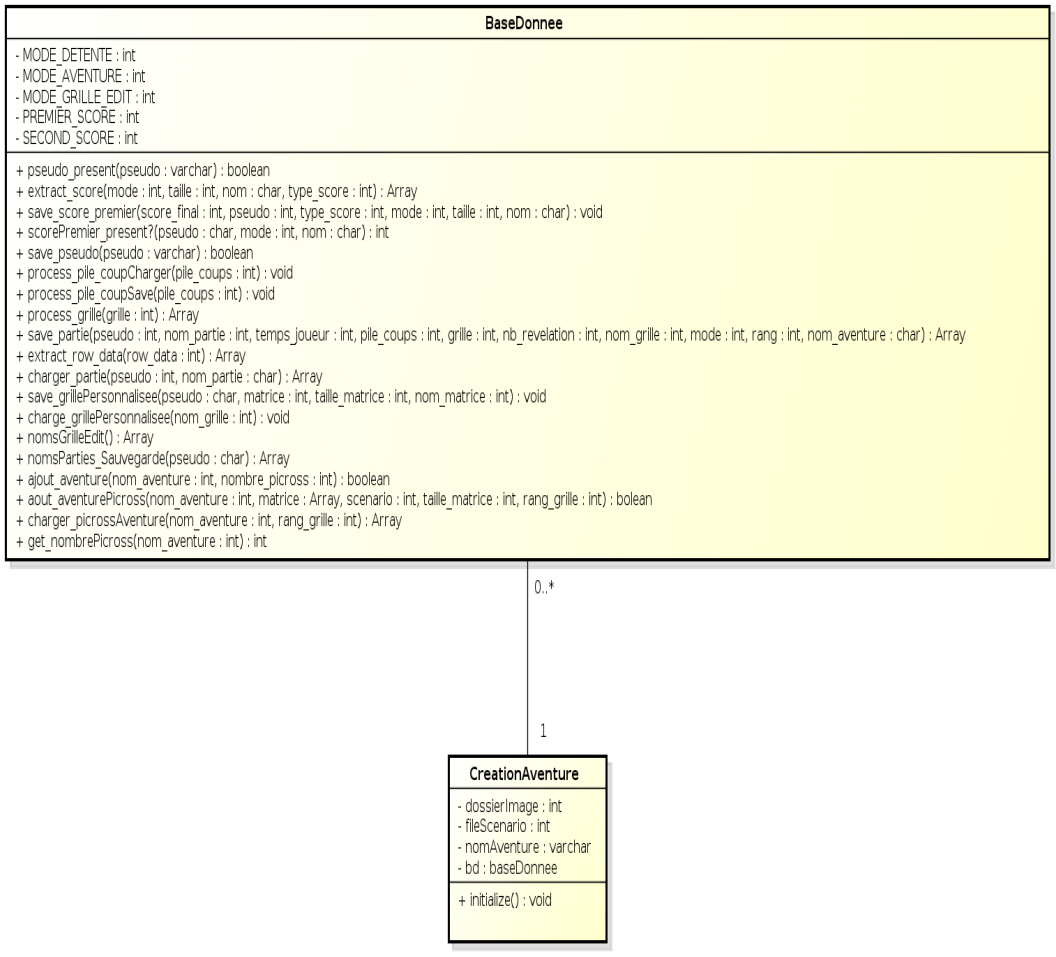

FIGURE 7 – Diagramme de classe de la base de donnée

Pour la réalisation de la base de donnée, la première piste fût de lister les éléments à sauvegarder, qui sont donc :

- Pseudonyme des joueurs
- Scores des joueurs
- Partie jouée
- Picross des aventures
- Grille personnalisée

Nous avons réparti la sauvegarde des données en deux parties :

— sauvegarde d'une partie en utilisant des fichiers xml (grâce au gem nokogiri)

— sauvegarde du reste en utilisant une base de donnée portable (sqlite3)

Cependant on a vite fait la remarque que la manipulation des donnée serait moins facile d'accès avec les fichiers xml (sauvegarde de plusieurs parties pour un même utilisateur). Donc on a décidé d'intégrer de nouvelles tables implémentant la sauvegarde des partie pour pouvoir utiliser sqlite3, ce qui nous ramène au modèle entité association en figure 3.

III.5 Aides

Lors d'une partie, le joueur est capable de consulter des aides ; on compte les différentes aides suivantes :

- Cas 1 :

  Sur une grille de picross quelconque, lorsque l'indice indiqué est égale à la largeur de la grille, alors toutes les cases doivent être noircies car celles-ci appartiennent à un seul bloc. Exemple :

10										

TABLE 1 – Ligne de picross avec indice 10

- Cas 2 :

  Lorsque l'indice indiqué est supérieur à la moitié de la largeur de la grille, vous pouvez déterminé les cases "sûr" avec l'opération suivante :

— (indice - (largeur de la grille/2))*2 => donne le nombre de cases à noircir symétriquement par rapport au milieu de votre ligne/colonne.

Exemple :

TABLE 2 – Ligne de picross avec indice 7

Dans le cas d'une grille de largeur 10 avec un bloc de 7 cases noires, l'opération nous donne :

— (7 - (10/2))*2 = (7-5)*2 = 4

Vous pouvez donc noircir 4 cases noires par rapport au centre de la ligne/colonne.

Une autre méthode consiste à noircir les cases depuis le début de votre ligne/colonne jusqu'à atteindre la taille du bloc. Vous devez alors compter le nombre de cases blanches restantes et retirer autant de cases toujours en partant du début de votre bloc.

Exemple : Dans le cas d'une grille de largeur 10 et d'un bloc de 7 cases, vous remplirez les 7 premières cases de votre ligne/colonne. Puis vous compterez le nombre de cases restantes (ou ferez la différence taille du bloc - largeur, ici 10-7 donne 3). Vous retirerez enfin ce nombre au début du bloc, soit les 3 premières cases du bloc.

- Cas 3 :

  Lorsqu'une case est noircie en bordure de votre Picross, en fonction de sa position (première case ou dernière case), vous pouvez déduire que cette case est la première du premier bloc ou la dernière du dernier bloc. Exemple :

TABLE 3 – Ligne de picross avec indice 2 avant la complétion

TABLE 4 – Ligne de picross après indice 2 avant la complétion

Lorsque la première case est noircie, si le premier bloc est de taille 3, vous pouvez noircir les deux cases suivantes. Si la dernière case est noircie et que le dernier bloc est de taille 3, vous pouvez logiquement noircir les deux cases précédentes.

- Cas 4 :

  Considérons que chaque bloc est séparé d'une case blanche, si :

  — somme des indices des blocs + nombre de bloc - 1 = largeur de la grille

  Alors il existe qu'une seule manière de remplir votre ligne/colonne. Vous n'avez qu'à remplir votre ligne/colonne en ne mettant qu'un espace entre chaque bloc. Si la valeur calculée est inférieur à la largeur de la grille, cela signifie qu'il y a plus d'une case blanche entre deux blocs (pas nécessairement entre chacun d'eux mais au moins entre deux d'entre eux).

III.6 Capture d'écrans

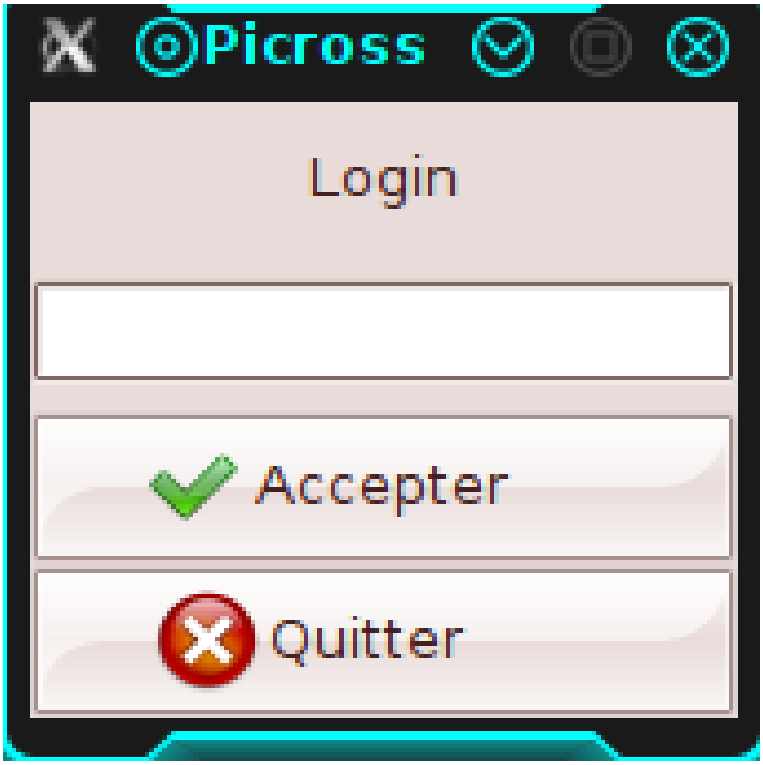

FIGURE 8 – Fenêtre d'identification

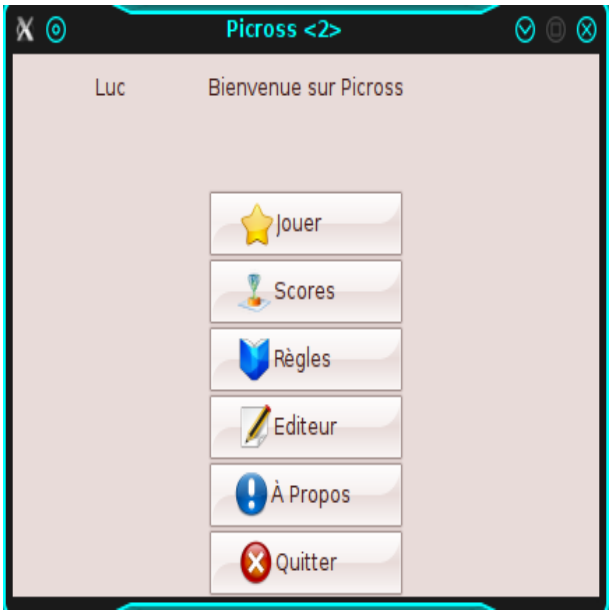

FIGURE 9 – Menu principal

FIGURE 10 – Menu du jeu

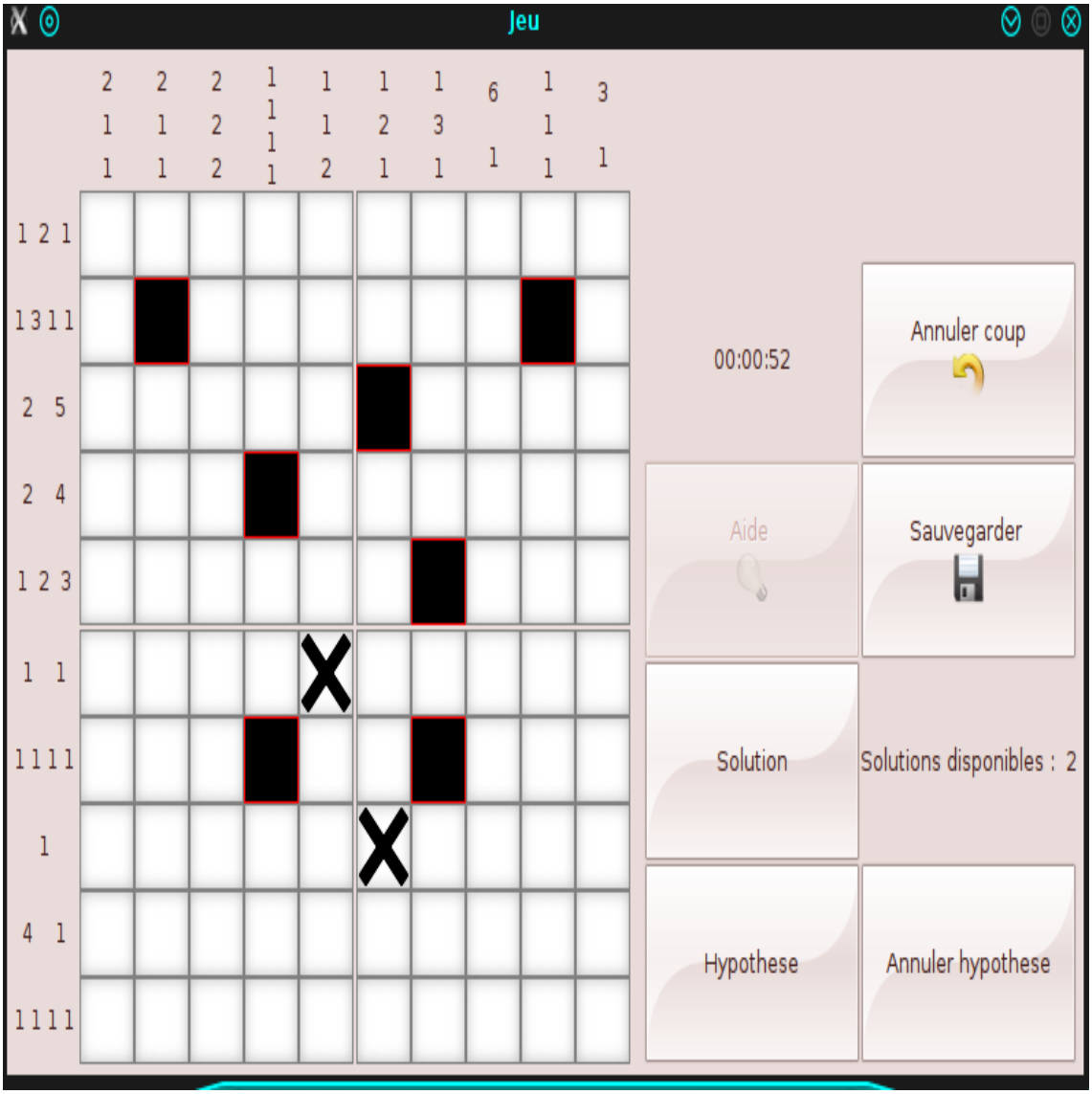

FIGURE 11 – Partie en 10x10

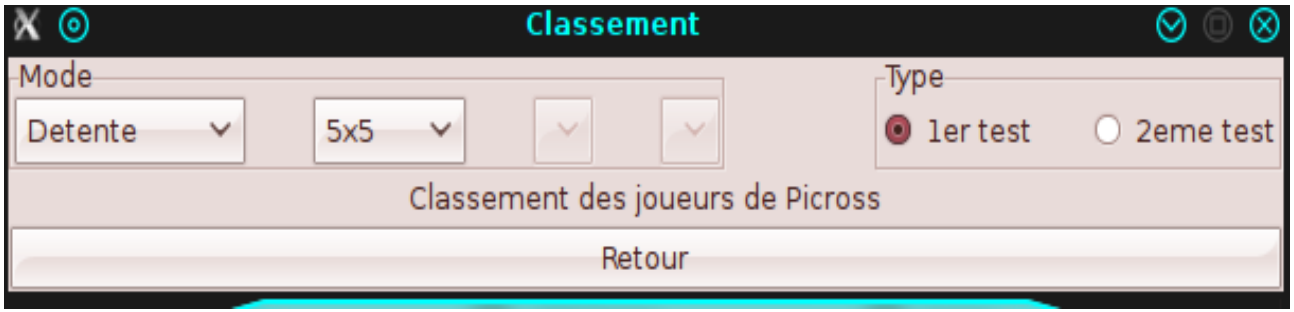

FIGURE 12 – Fenêtre affichant la table des scores (actuellement vide)