Dossier de projet

SUDOKU

**Sommaire**

I – Présentation du projet  
II – Contexte et objectifs  
III – Organisation du projet  
IV – Maquettage de l’application

V – Modèle de données / Diagrammes

**Présentation du projet**

Dans le cadre de la formation, et pour valider le module d’Android, notre coordinateur de formation nous a imposé 3 sujets au choix, à concevoir et à développer seul du 7 février au 15 février 2019.  
  
Il a voulu nous faire travailler à travers la création de jeux car cela touche à plusieurs étapes de développement importantes. Les sujets sont les suivants : Un Tower defense, un sudoku ou un sujet libre en rapport avec un jeu choisit et à valider par le coordinateur avant de commencer.

Pour ma part, j’ai pris le sujet « Sudoku ». Pour la simple et bonne raison que ça me parlait et que c’est un jeu auquel je joue énormément durant mes temps libres.

**Contexte et Objectifs**

Le sudoku est un jeu de logique d’origine japonaise consistant à compléter une grille de 9 x 9 cases avec des chiffres allant de 1 à 9.

|  |  |
| --- | --- |
| C:\Users\Public\Documents\eSNP52Y.jpg  Une grille de sudoku vide | C:\Users\Public\Documents\sudoku1.png  Une grille de sudoku inachevé |

Le but de ce jeu est de finir la grille sans qu’un chiffre identique ne soit dans la même grille 3x3, ni sur sa ligne horizontal, ni sur sa ligne verticale.

L’objectif sera de créer une interface utilisateur sous Android. Il nous faudra aussi une api rest pour communiquer avec un classement en ligne et pouvoir enregistrer automatiquement nos utilisateurs en toute transparence. Et pour finir nous devons trouver un algorithme rapide pour résoudre le sudoku.

**Organisation du projet**

J’ai décidé d’organiser mon travail via l’outil de gestion de projet « Trello », qui est rapide et simple à mettre en place.

Pour commencer, je suis parti voir la concurrence sur le market de google play voir les applications proposant du sudoku et ainsi voir si je ne faisais pas fausse route.  
J’en ai testé quelques une et je ne pense pas pouvoir faire la même chose avec le peu de temps que j’ai pour développer l’application. J’ai donc pris les fonctionnalités les plus importantes en tête et qui correspondent avec nos objectifs.

|  |  |  |
| --- | --- | --- |
| C:\Users\Lenovo\Desktop\52051089_1279133562224404_8098280739210002432_n.jpg | C:\Users\Lenovo\Desktop\51612573_419015555507711_7859109111600775168_n.jpg  Des jeux de sudoku sur l'app store | C:\Users\Lenovo\Desktop\51659908_1507221306078102_9086456223284330496_n.jpg |

J’ai ensuite mis sur feuille les fonctionnalités qu’on trouvera dans l’application.

Du coup quand l’utilisateur arrivera sur l’application et on lui proposera un menu d’accueil comportant :

- « Jouer » qui permettra à l’utilisateur de commencer une partie

- « classement » qui permettra de voir son classement et le classement général

- « paramètres » qui permettront de changer son pseudo et la difficulté de la grille

Les étapes d’une partie seront :

- Quand l’utilisateur rentre dans la partie, on démarre un chronomètre et on affiche quelques chiffres dans les cases du tableau pour commencer.  
- Quand l’utilisateur rempli la grille entièrement, on déverrouille le bouton valider.  
- Quand l’utilisateur clique sur valider, on arrête le chrono et on vérifie la grille.  
- Si la solution est valide, on termine la partie et on enregistre le score en ligne.  
- Si la solution n’est pas valide, on relance le chronomètre là où on l’a arrêté et on reprend la partie.

On peut maintenant faire un diagramme de cas d’utilisation pour représenter le fonctionnement nécessaire à l’utilisateur.

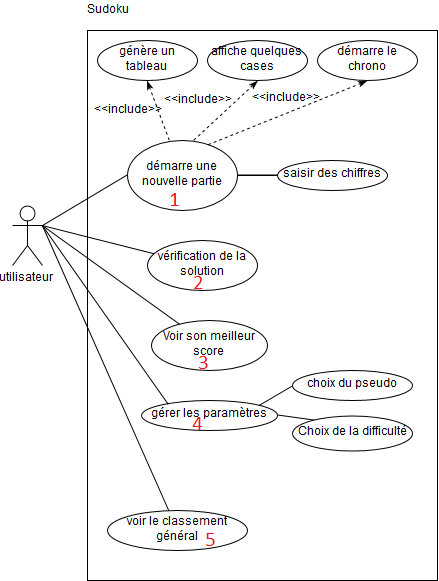


Diagramme de cas d’utilisation

**UC1 – Démarre une nouvelle partie**

**Description** : Permet au joueur de jouer une nouvelle partie de sudoku

**Acteurs** : Joueur

**Préconditions** : Aucune

**Scénario Nominal** :

1. Le joueur clique sur « jouer » dans l’accueil
2. Le système va préparer un tableau 9 x 9 déjà résolu et y sélectionner quelques cellules suivant la difficulté ( 20 pour facile et 10 pour difficile )
3. Le système va générer un tableau 9x9 vide, insérer les cellules précédemment sélectionnées
4. Le système va afficher notre grille et le contenu des cellules de notre tableau précédemment rempli
5. Le système affiche un menu contenant « Valider » désactivé et « Abandonner » activé
6. Le système démarre le chronomètre
7. Le joueur sélectionne une cellule qui n’est pas une cellule initiale ( celle générée au début )
8. Le système invite le joueur à saisir un chiffre entre 1 et 9 ( si 0, vide et autres la cellule sera vide )
9. Le joueur saisit une valeur
10. Le système met à jour la grille avec la valeur saisie
11. Si toutes les cellules sont remplis, le système déverrouille dans le menu l’onglet « valider »
12. Le joueur clique sur « valider »
13. Le système arrête le chronomètre
14. Le système teste si la solution du joueur est valide
15. Si non valide, le système redémarre le chronomètre et le joueur reprend la partie
16. Si Valide, le système fini la partie

**Post-conditions** :

Le système affiche un message de félicitation au joueur

Le système demandera au joueur s’il veut envoyer son score en ligne

**UC2 – Vérification de la solution**

**Description** : Permet au joueur de valider sa grille

**Acteurs** : Joueur

**Préconditions** : Avoir rempli sa grille entièrement

**Scénario Nominal** :

1. Le système va créer un tableau1 vide de taille 3
2. Le système va créer un tableau2 vide de taille 9
3. Le système va additionner la valeur des cellules de chaque ligne
4. Si l’addition de la ligne est égale à 45, le système ajoute true au tableau2
5. Si tout est true dans tableau2, le système ajoute true au tableau1
6. Le système va écraser le tableau2 et garder la taille 9
7. Le système va additionner la valeur des cellules de chaque colonne
8. Si l’addition de la ligne est égale à 45, le système ajoute true au tableau2
9. Si tout est true dans tableau2, le système ajoute true au tableau1
10. Le système va écraser le tableau2 et garder la taille 9
11. Le système va additionner la valeur des cellules du carré 3x3
12. Si l’addition de la ligne est égale à 45, le système ajoute true au tableau2
13. Si tout est true dans tableau2, le système ajoute true au tableau1
14. Le système vérifie que tout est true dans le tableau1
15. Si tout est true, la solution est valide

**Post-conditions** :

On envoie une réponse true ou false

**UC3 – Voir son meilleur score**

**Description** : Permet au joueur de voir son meilleur score

**Acteurs** : Joueur

**Préconditions** : aucune

**Scénario Nominal** :

1. Le joueur clique sur « classement »
2. Le joueur va cliquer sur l’onglet « solo »
3. Le système va chercher sur une base de données distante les scores du joueur
4. Le système va chercher le meilleur temps en facile
5. Le système va chercher le meilleur temps en difficile
6. Le système va afficher le meilleur temps du joueur dans les difficultés facile et difficile

**Post-conditions** : aucune

**UC4 – Gérer les paramètres**

**Description** : Permet au joueur de modifier ses préférences

**Acteurs** : Joueur

**Préconditions** : aucune

**Scénario Nominal** :

1. Le joueur clique sur « paramètres »
2. Le Joueur clique sur « Pseudonyme »
3. Le système ouvrira une boîte de dialogue reprenant le pseudonyme du joueur
4. Le joueur change son pseudonyme et confirme
5. Le système va vérifier si le pseudonyme changé n’existe pas déjà
6. Si oui, le système redemande à l’utilisateur le changement de pseudonyme
7. Si non, fait le changement avec celui de la base de données

**Post-conditions** :

Le système ferme la boite de dialogue

**UC5 – Voir le classement général**

**Description** : Permet au joueur voir le classement général

**Acteurs** : Joueur

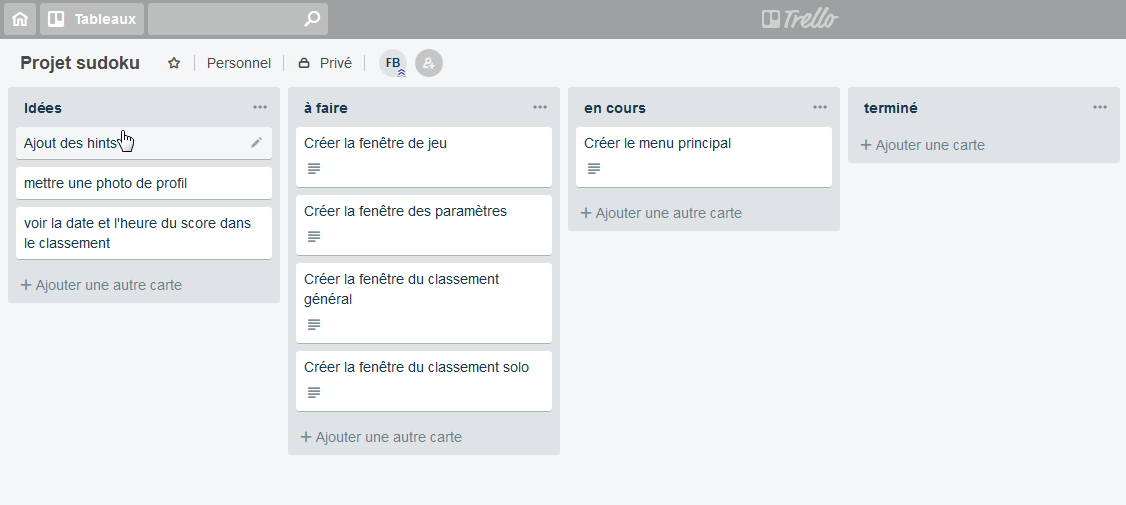
**Préconditions** : aucune

**Scénario Nominal** :

1. Le joueur clique sur « classement »
2. Le joueur clique sur « facile »
3. Le système va dans sa requête chercher les 50 premiers meilleurs scores en facile tout joueur confondu
4. Le système va les afficher dans l’ordre croissant
5. Le joueur clique sur « difficile »
6. Le système va dans sa requête chercher les 50 premiers meilleurs scores en facile tout joueur confondu
7. Le système va les afficher dans l’ordre croissant

**Post-conditions** : aucune

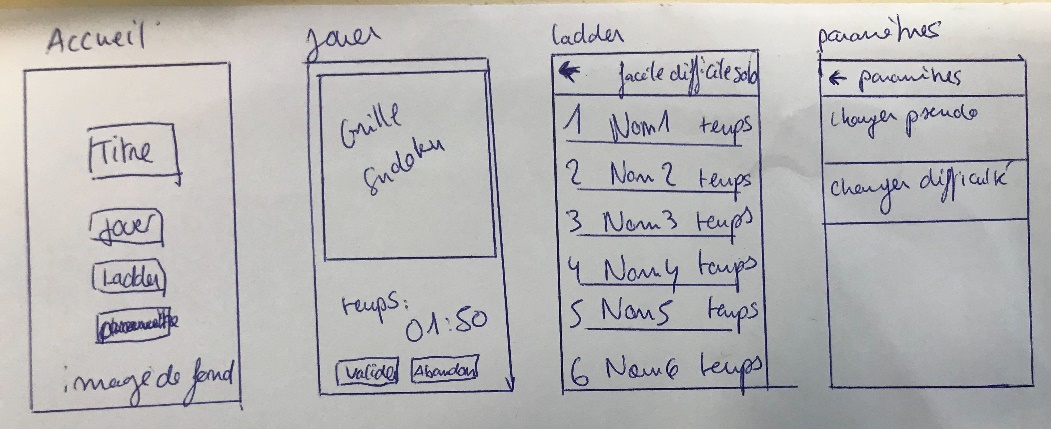
Une fois fini, j’ai ajouté dans le « Trello » les étapes nécessaires qu’on validera au fur et à mesure du développement de l’application et on détaillera dans les cartes de chaque étape ce qu’il sera nécessaire de travailler pour celle-ci.



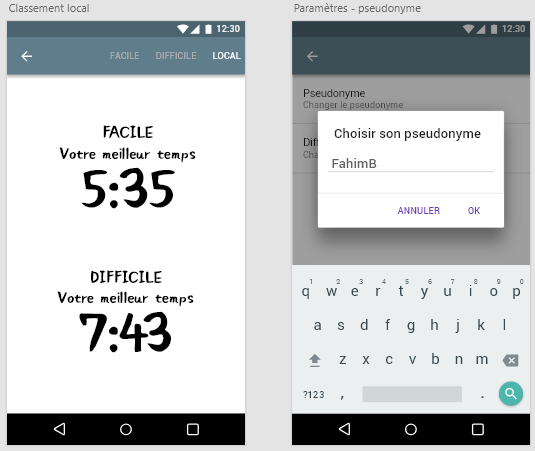
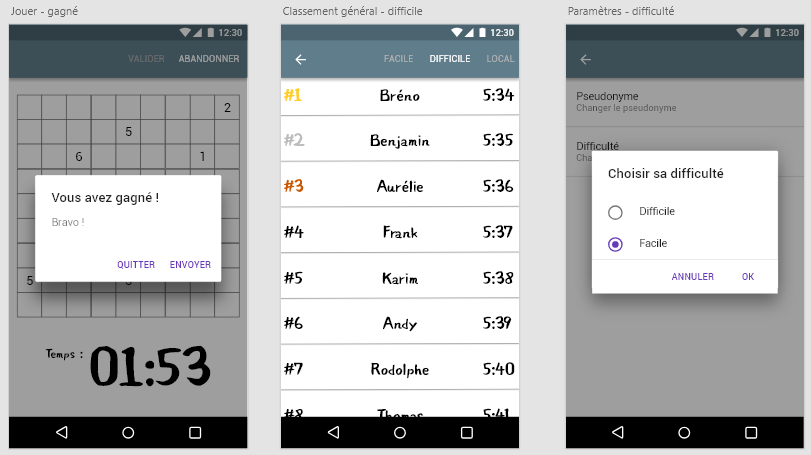
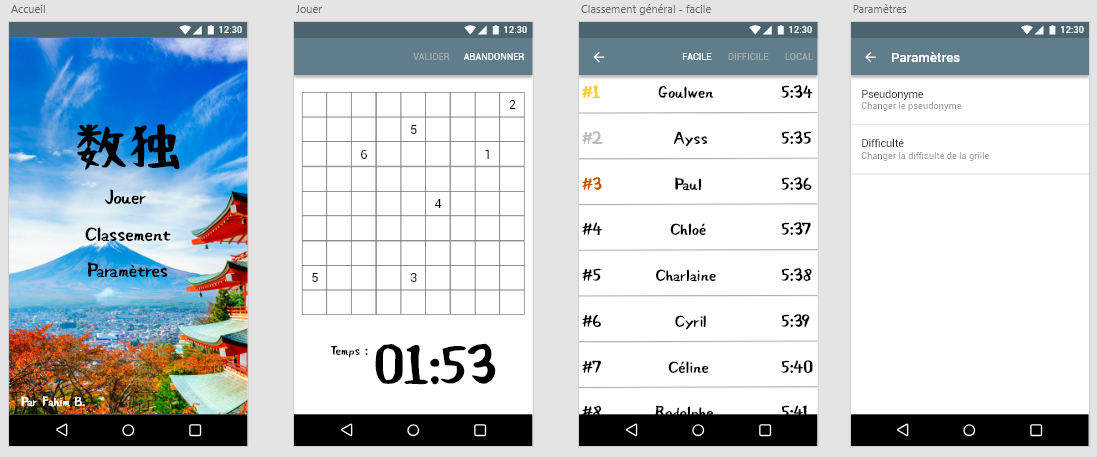
Trello vue d’ensemble du projet

**Maquette**

Tout d’abord, j’ai dessiné sur papier ce qui me venait à l’esprit pour le design de l’application.



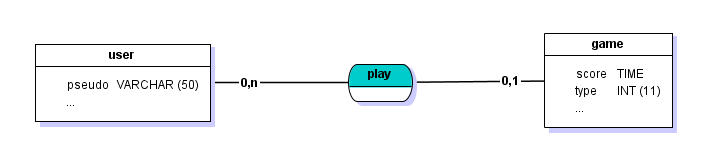
Puis grâce à l’application ADOBE XD on peut approfondir le design des fenêtres. J’ai préféré partir sur tous les cas de fenêtres possible à faire. Cela reste très simpliste et ce n’est peut-être pas notre résultat final sachant que ça n’est que le prototype de l’application.



Maquette de l’application sudoku

**Modèles de données et Diagrammes**

J’utilise JMerise pour faire la modélisation de mes données. On commencera par la création du MCD ( Modèle Conceptuel de Données ). On aura donc une entité user et une entité game, avec une cardinalité 0,N et 0,1 car :

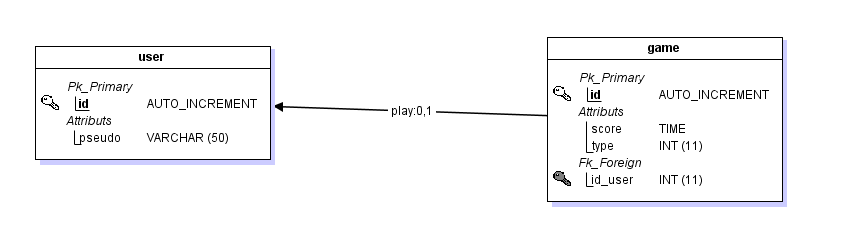


MCD du projet sudoku

- Un user peut ne pas avoir joué ou jouer à plusieurs parties

- Dans une game, sont joué aucun ou 1 user

Ce qui nous génère le MLD ( Modèle logique de données ) suivant :



MLD du projet sudoku

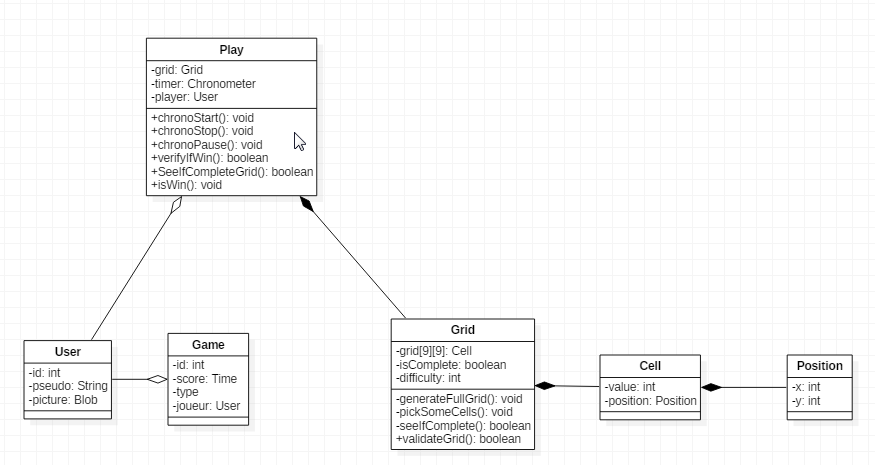
Nous avons aussi un diagramme de classe représentant les relations entre les classes de l’application :

Diagramme de classe du projet sudoku