DE MARCO Léopold

ARCHIDOIT Lydie

MOULIN Elouan

# Compte-rendu du projet d'informatique du S2

## I – Objectifs & contraintes

L'objectif principal que nous nous sommes fixés pour rendre un projet qui s'approche le plus possible de ce qui était demandé était le suivant :

Avoir un jeu de sudoku jouable en multijoueur (non simultané), avec l'appui d'une base de donné pour pouvoir stocker les différentes données (grilles, statistiques, classement, informations joueurs…) pour pouvoir ainsi permettre à cette fonctionnalité de multijoueur d'exister et permettre au joueur connecté sur sa session de voir ses statistiques. Bien évidemment, il fallait ajouter d'autres fonctionnalités comme les défis du jour par exemple, mais nous nous sommes avant tout concentrés sur un objectif principal pour aller au plus vite et faire au plus simple pour ensuite développer car nous n'avions que peu de temps à notre disposition pour ce qui était demandé.

Le temps était une des premières contraintes. Nous nous sommes donc organisés en conséquence et avons codé sur chaque heure libre disponible.

La deuxième était d'apprendre à gérer le lien entre python et base de données. Il fallait créer une connexion avec la base de données, que tout le monde ait les mêmes données (problème réglé grâce à Github), faire fonctionner les requêtes SQL dans le code python, et vérifier que les changements étaient effectués sur la BDD.

La troisième était de gérer le lien entre l'interface et la base de données. La récupération et l'envoi des données n'étaient pas évidents, nous avons lu beaucoup de documentation et regardé beaucoup de vidéos pour chercher à comprendre comment cela fonctionnait.

## II – Organisation, conception

Nous nous sommes organisés de la façon suivante dans la division des tâches :

Léopold : Création de la base de données, des fonctions pour récupérer et envoyer des données dans la base de données, gestion du lien entre la base de données et les informations dictées par le joueur.

Lydie : Création de l'interface graphique en Tkinter, gestions des grilles et affichage, renvoie des informations joueur proprement.

Elouan : Traitement des informations et faire le lien entre le travail de Léopold et Lydie, code Python « pur » (pas de SQL ni de Tkinter)

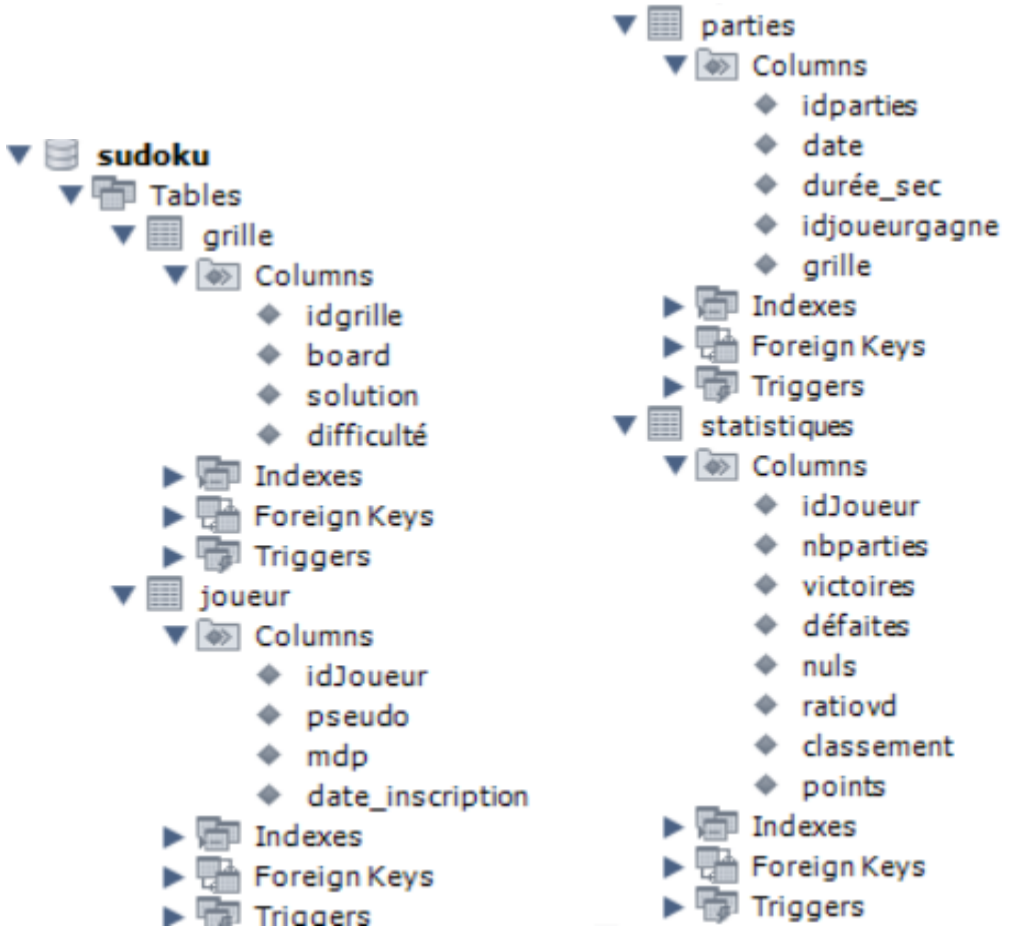
Afin de mener le projet à bien, nous nous sommes servi de Github pour centraliser le code et avoir tous la même version du code.

Cet outil nous a été très utile et a permis d'éviter les incohérences entre les versions et les conflits. Si nous avions échangé les codes sur Discord nous n'aurions peut-être pas eu les mêmes versions à chaque fois.

Ensuite, nous avons fait de la Programmation Orientée Objet. Cela a permis d'avoir un code plus propre et mieux divisé en plusieurs méthodes afin d'utiliser celle dont nous avons besoin. Nous avons créé des classes pour les joueurs, les requêtes SQL, ect.

Nous avons choisi d'utiliser mySQL en tant SGBDR pour stocker toutes nos données multijoueurs, nous avons créé 4 tables :

- Joueur : stock toutes les données du joueur (mdp, identifiant… )
- Grille : stock les grilles qui feront jouer les joueurs ( stock le board de la grille, son id, sa correction …)
- Statistiques : profil statistique du joueur qui stock ses statistiques , par exemple, nombre de victoires, de nuls et défaites.
- Parties : stock les données relatives à la partie en elle-même.



Par manque de temps, nous n'avons finalement pas utilisé les tables grille et parties, nous avons décidé de les laisser apparaitre car elles auraient dû faire partie de notre projet et auraient été importantes.

Pour chiffrer le mot de passe nous avons utilisé le module "cryptography" qui permet avec une clé de chiffrer/déchiffrer un mot de passe. Vous êtes la seule personne à la posséder.

Pour la difficulté, nous avons choisi des intervalles, sur une échelle de 1 à 9 : si le joueur choisit une grille de difficulté 1 à 3, cela va correspondre à facile, si il choisit une difficulté entre 4 et 6 cela sera considéré comme moyen et entre 7 et 9 comme difficile.

Pour le classement, nous avons procédé de la manière suivante :

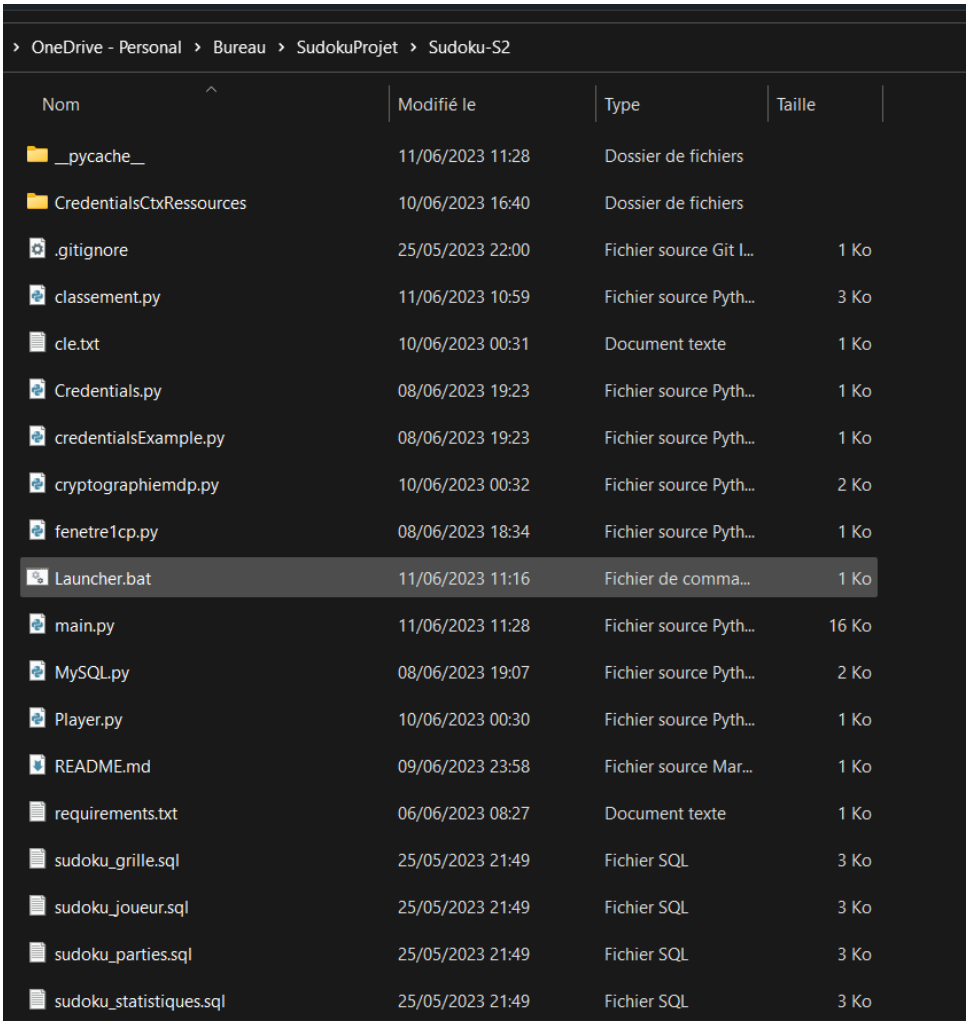
- Une victoire apporte un gain fixe selon la difficulté. Pour une grille facile, +1 point, pour une grille moyenne +3 points, pour une grille difficile +5 points.
- Une défaite fait perdre un nombre de points variables selon le ratio du joueur. Pour une grille facile -(ratio*5) en considérant qu'une défaite sur une grille facile arrive le plus souvent avec des joueurs ayant un faible ratio, ou au contraire un gros ratio mais donc une défaite fera perdre beaucoup. Pour une grille moyenne -(ratio*2) considérant que les joueurs à ratio moyen perdent une quantité raisonnable de points pour ne pas être pistonné trop facilement vers le niveau difficile. Pour une grille difficile -(ratio*1) car les joueurs à ce stade auront un plus gros ratio et donc cela ne les découragent pas. Imaginons un joueur à faible ratio gagnant beaucoup de parties difficiles, alors il aura de plus en plus de chance de perdre gros car son ratio augmente, et imaginons un même joueur perdant souvent à ce stade, il va être découragé et va redescendre logiquement de catégorie.

La différence de difficulté entre 7 et 9 ne reportera pas plus de points ou ne fera pas perdre plus de points. C’est plus pour challenger les joueurs, ils n’ont rien à perdre ou gagner de plus, ils restent dans la même difficulté malgré que celle de la grille augmente. C’était également plus simple pour nous de procéder ainsi.

## III – Description du rendu final

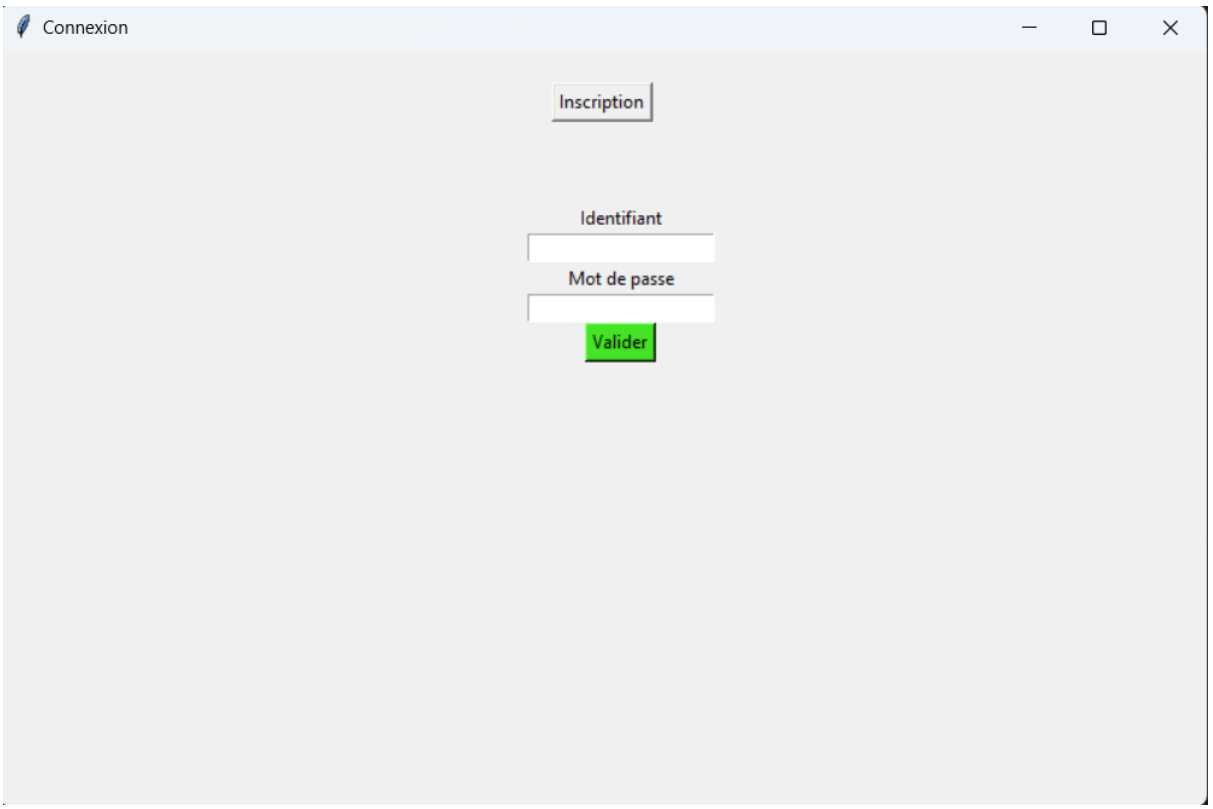
Tout d’abord, avant d'aborder ce que nous avons réussi, échoué, voici des captures d’écran montrant le résultat final :

D’abord à l’ouverture du fichier :

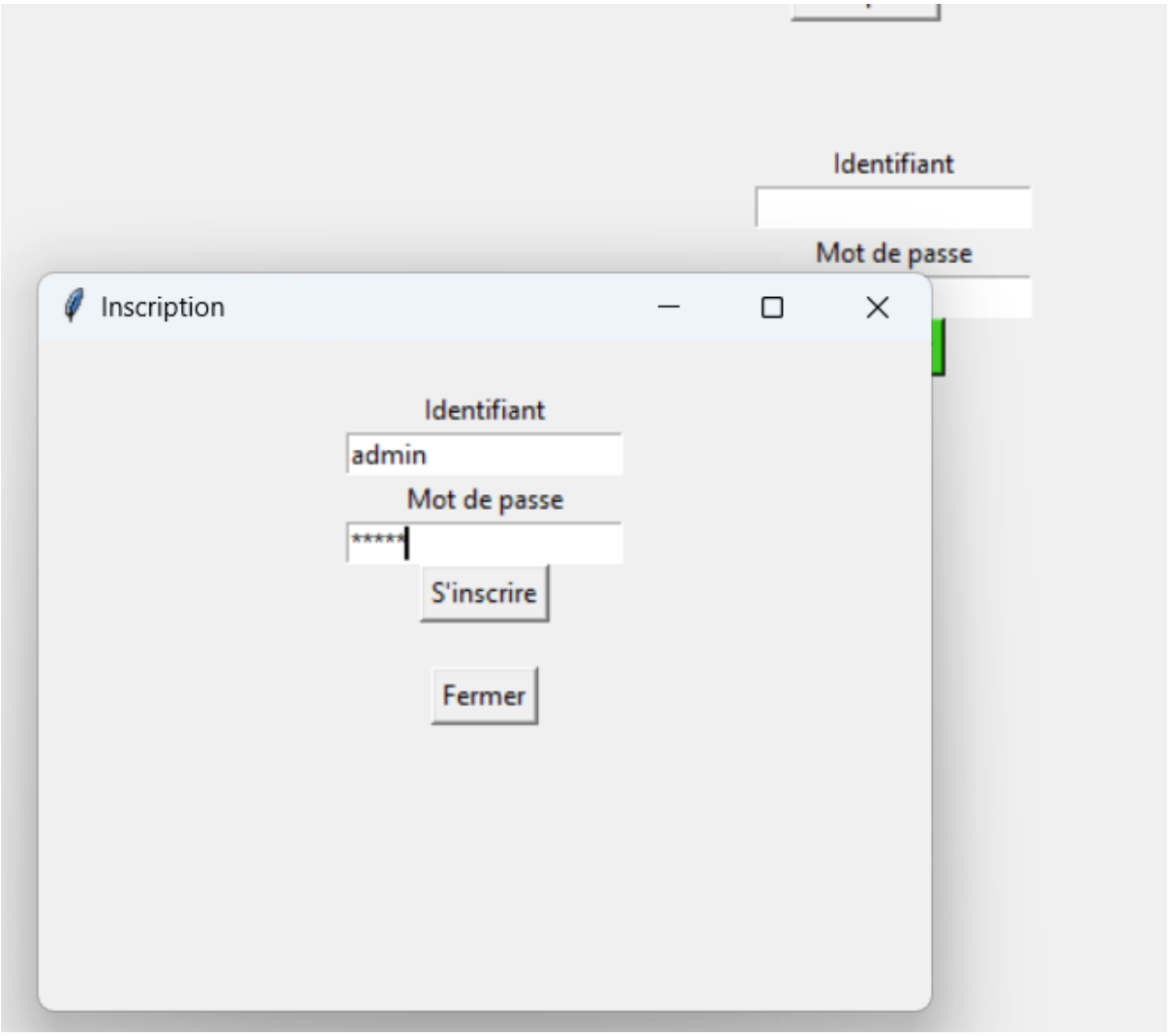



Ouvrir ‘Launcher.bat’

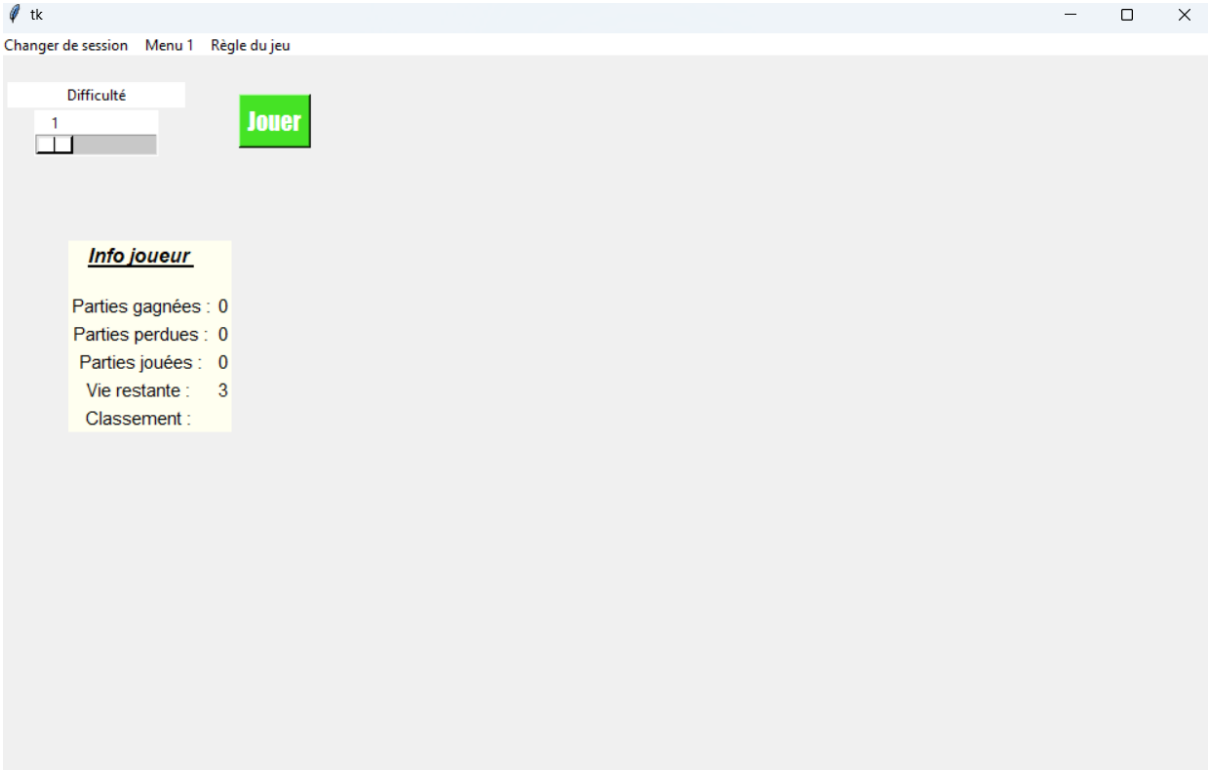
Ensuite :



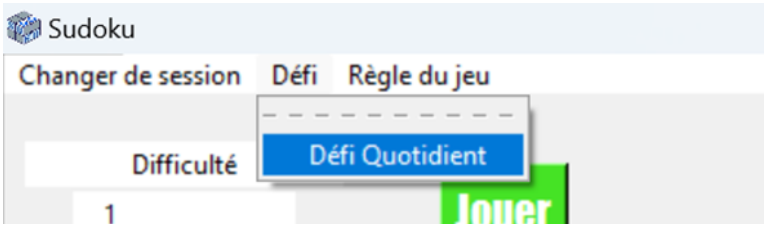

Si vous n’avez pas de compte, il est l’heure de vous en créer un.



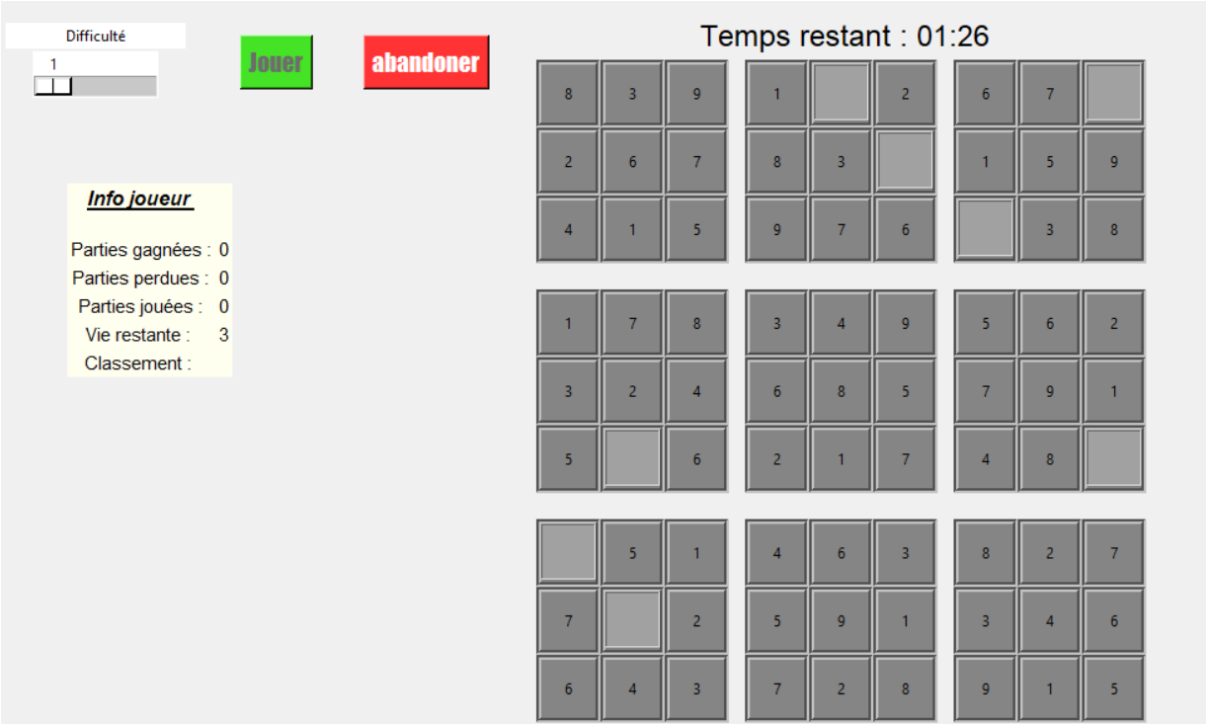

Je me crée une session admin et je clique sur m’inscrire.



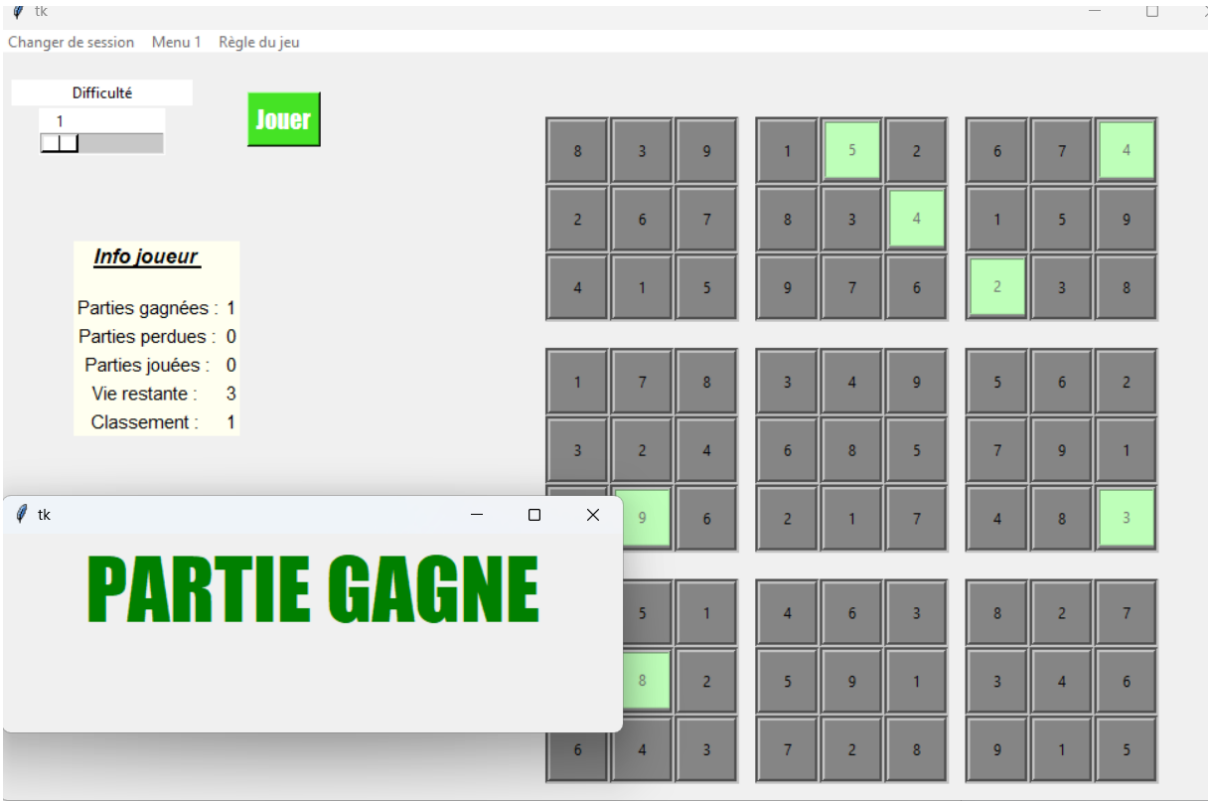

Youpi ! Je suis inscrit ! Si j'ai déjà un compte, je me connecte avec mes identifiants stockés sur la base de données. On peut voir que j'ai accès à plein de statistiques, mais pas encore de classement car je n'ai pas joué. Je peux également régler la difficulté.



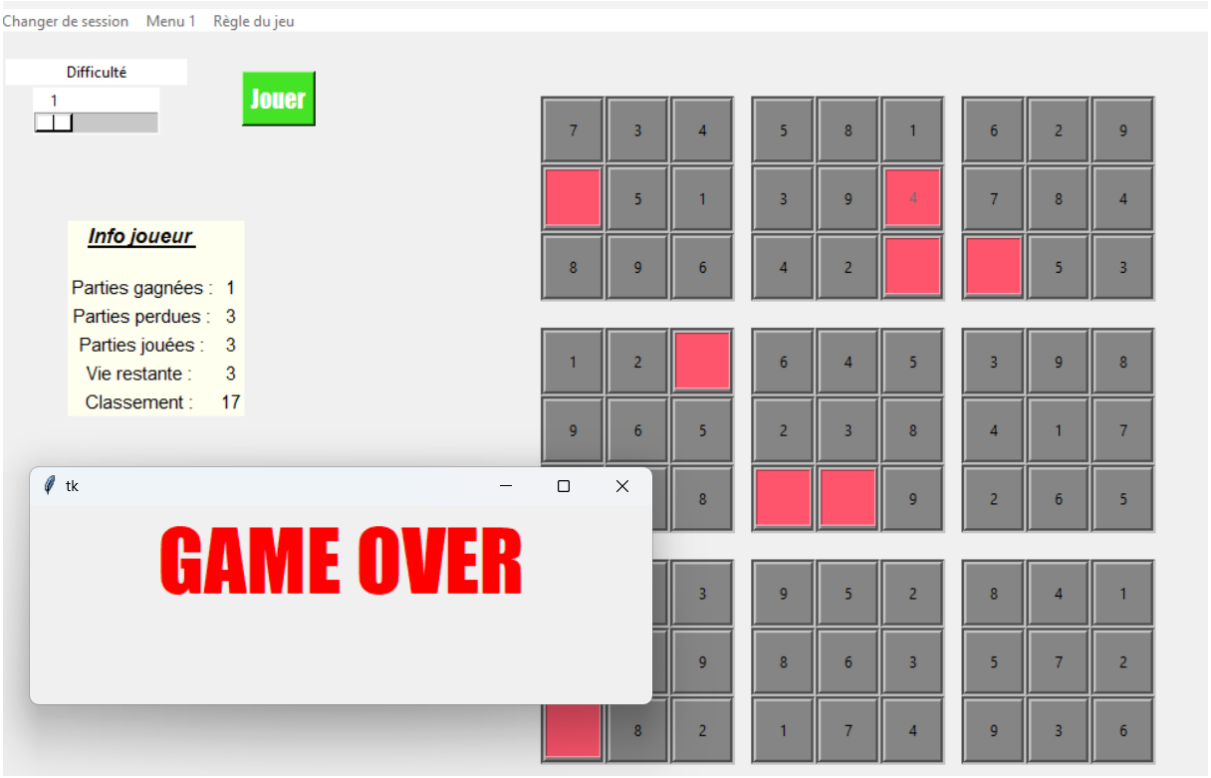

J'ai accès à plusieurs options avec les onglets disponibles en haut à droite, comme les règles du jeu, la grille défi quotidien ou encore quitter le jeu.




Let's play ! Je joue une grille facile et j’appuie sur jouer.




J’ai gagné super !




Mais après plusieurs parties, j'ai perdu. Je suis bien redescendu dans le classement.


J'ai également perdu au temps.

Je vous laisse bien entendu explorer toutes les autres possibilités à disposition.

Nous allons commencer par notre plus grand regret qui est de ne pas avoir réussi à coder correctement les "battles" correctement. En effet, il y a bien un système compétitif entre les joueurs, un classement, en

fonction des victoires/défaites et de la difficulté des grilles. Cependant, tous les joueurs ne s'affrontent pas sur des grilles prédéfinies. Nous n'avons pas réussi à développer cette fonctionnalité car cela mettrait encore plus de temps (par rapport à nos connaissances et capacité actuelles) que nous n'en avons déjà passé. Nous avons donc simplifié le système compétitif du sudoku.

Ensuite, nous avons développé tout un système de statistiques, on ne peut cependant pas voir en fonction du temps nos résultats par exemple combien de matchs gagnés de la mois, c'est justement dans cette partie que les tables "grille" et "partie" devaient nous être utile, ceci n'a malheureusement pas abouti encore une fois par un manque de temps, nous avons ici aussi simplifié.

Cependant nous avons réussi à mettre en place beaucoup de fonctionnalité dont les défis du jour qui sont des grilles 4*4 de difficulté moyenne (5) à résoudre en moins de 5min, mais également la gestion des utilisateurs et du classement, ou encore le chronomètre qui calcul le temps de référence.

Pour l'interface graphique, nous avons réussi à créer un visuel simple et pratique. Nous aurions aimé apprendre plus en profondeur les fonctions de Tkinter pour pouvoir créer une interface plus développée et dans le style des jeux plus récent. Le choix de Tkinter nous a paru dans un premier temps le plus adapté car peut-être plus abordable et facile à prendre en main pour des novices. Cependant, Pygame nous aurait peut-être permis d'avoir plus de liberté et nous aurait permis d'avoir une interface plus personnelle et aboutie. Cependant, une fois lancé, reprendre à zéro l'interface n'aurait pas été réalisable et prendre en main une nouvelle bibliothèque aurait été trop compliqué.

La création d'une grille nous aura posé de nombreux problèmes comme avoir des carrés bien alignés, mettre les zones de saisie de texte au bon endroit (grâce à la méthode grid) et centré sur la case, ou encore séparer la grille en sous carré en les délimitant avec des lignes

blanches. Après de nombreux tests et plusieurs pistes réfléchies sur papier nous avons réussi à créer le visuel de grille que nous voulions.

```
if taille==3:
    for j in range(9):
        nbrow=0
        for i in range(9):
            l=Label(frameprincipale,bg='#858585',width=6,height=3,relief="groove",borderwidth=4)
            l.grid(row=nbrow,column=nbcolumn)
            lab.append([l,(j,i)])
            blanc+=1
            if blanc%3==0:
                nbrow+=1
                lablanc=Label(frameprincipale,  text='', width=0,height=1)
                lablanc.grid(row=nbrow,column=nbcolumn)
            nbrow+=1
        if blanc==27 or blanc==54:
            nbcolumn+=1
            lablanc2=Label(frameprincipale, text='',width=0,height=1)
            lablanc.grid(row=nbrow,column=nbcolumn)
        nbcolumn+=1
```

Par la suite, nous avons enrichi notre code pour pouvoir coder une grille 4x4 pour les défis quotidiens.

## IV- Retours et autres

Le projet était plutôt long à réaliser, en moyenne nous avons chacun passé environ 30 à 35h à coder et nous avons trouvé cela plutôt éprouvant. Cependant, c'était une expérience enrichissante, nous avons beaucoup appris, en gestion de projet et en code, que cela soit en Tkinter, SQL, interface Client/Serveur, Client/Interface graphique.