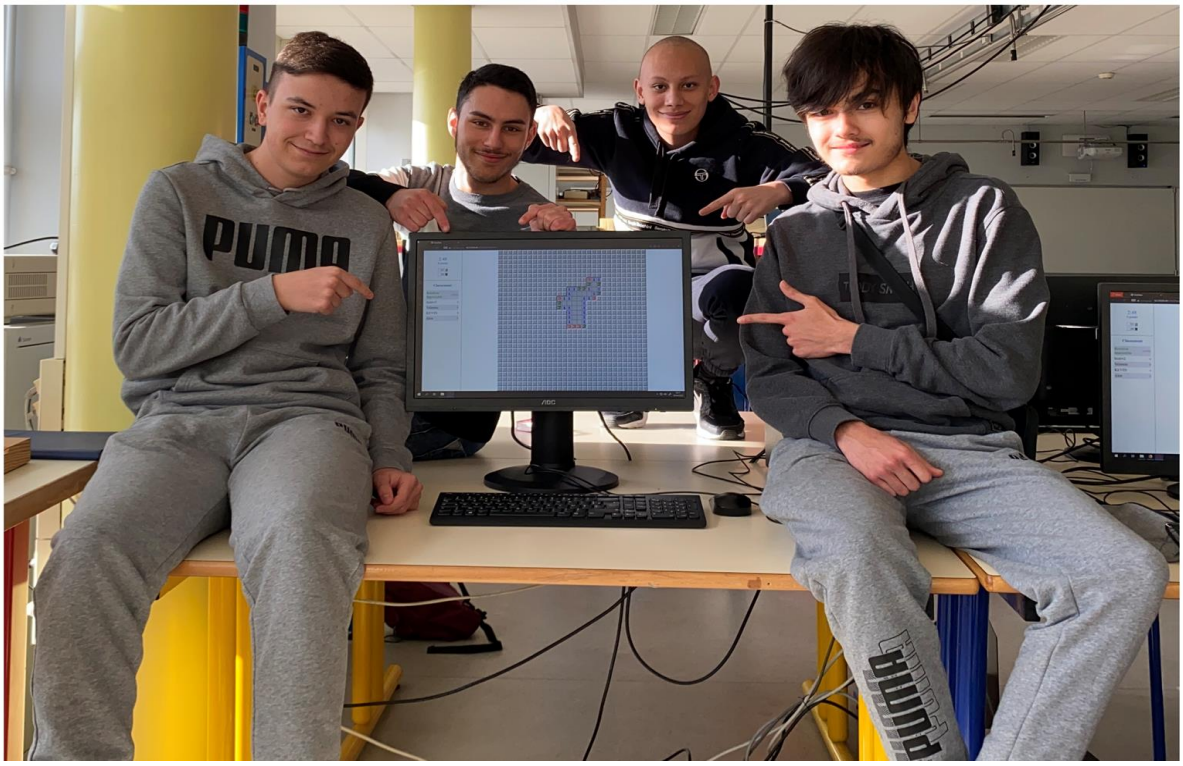
LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

GridWars



NOM DU PROJET : GridWars

> PRÉSENTATION GÉNÉRALE :

*• Notre projet s'appelle **GridWars** et vise à repenser et adapter le célèbre jeu de démineur en version multijoueurs. Notre objectif principal est de proposer une expérience de jeu interactive, engageante et compétitive, où les joueurs peuvent s'affronter en temps réel sur un champ de mines virtuel.*

*• Le projet est né d'une recherche concernant des jeux de stratégie et de réflexion qui peuvent être améliorés tout en gardant le côté historique de celui-ci. L'intérêt de **GridWars** réside dans la combinaison de l'aspect ludique du jeu original avec un aspect compétitif propre aux jeux multijoueurs. En outre, cela nous a permis de travailler sur un projet qui demande une réflexion approfondie en matière de design, d'architecture logicielle et d'expérience utilisateur.*

*• Pour développer **GridWars**, nous avons choisi d'utiliser le langage de programmation Python afin de gérer le serveur ainsi que le jeu en lui-même. Nous avons également utilisé HTML, CSS et JavaScript pour concevoir et mettre en œuvre l'interface utilisateur du jeu et le site web associé. L'utilisation d'une base de données et du langage SQL a également été nécessaire afin de gérer les utilisateurs et les statistiques.*

*• **GridWars** reprend les mécanismes de base du démineur classique, où les joueurs doivent identifier et marquer les mines cachées sur une grille sans les déclencher. Cependant, dans notre version multijoueurs, les joueurs s'affrontent pour marquer le plus grand nombre de points en un temps limité tout en essayant d'aller le plus loin possible dans la partie. Pour ajouter de la profondeur au jeu, nous avons également intégré plusieurs options telles que la taille des grilles, la possibilité de partir chacun de la même grille, ou encore le choix d'un robot qui simulera un joueur (plusieurs difficultés sont disponibles).*

*• Tout au long du développement de **GridWars**, nous avons été confrontés à divers défis techniques et conceptuels, notamment en ce qui concerne la gestion de la synchronisation et de la communication entre les clients et le serveur, ainsi que la conception d'une interface utilisateur intuitive et réactive. En surmontant ces défis, nous avons acquis de précieuses compétences en matière de travail en équipe, de gestion de projet et de développement logiciel, qui nous serviront dans nos futures carrières.*

> ORGANISATION DU TRAVAIL :

*• Notre équipe se compose de 4 membres, chacun ayant un rôle spécifique dans le projet **GridWars**. Afin de travailler de manière efficace et de tirer le meilleur parti des compétences de chacun, nous avons réparti les tâches en fonction des rôles et des domaines d'expertise :*

- *Kevin : Création du robot qui simule un joueur en python*
- *Valentin : Gestion du serveur en python, du stockage des données en SQL*
- *Alexandre : Création du jeu en python*
- *Hizir Ali : Création du site web (HTML, CSS, Js)*

• Notre équipe a travaillé ensemble à la fois pendant les heures de cours et en dehors de l'établissement scolaire. Pour cela, nous avons utilisé une extension de Visual Studio Code nommée Live Share (par Microsoft) qui nous a permis de travailler simultanément sur un ou plusieurs fichiers.

• L'équipe était séparée en deux sous-équipes, d'abord Kevin et Alexandre qui, par le lien fort entre le jeu et le robot, ont eu besoin de mettre en relation leurs deux rôles. Ensuite, Hizir Ali et Valentin ont travaillé en collaboration afin de mettre en place une relation stable entre le serveur et le client tout en prévoyant le maximum de situations possibles.

> LES ÉTAPES DU PROJET :

Le développement de **GridWars** a été divisé en plusieurs étapes clés, allant de l'idée initiale jusqu'à la finalisation du projet. Voici les différentes étapes du projet :

1. Idée et planification : Au cours de cette première phase, nous avons discuté de nos idées et de nos intérêts, et avons décidé de travailler sur un démineur multijoueurs. Nous avons établi les objectifs généraux du projet, déterminé les fonctionnalités et les mécanismes de jeu souhaités, et planifié la répartition des tâches et des responsabilités au sein de l'équipe.
2. Développement et intégration : Une fois l'idée en tête et approuvée par tous, nous avons entamé le développement du projet. Alexandre et Kevin ont développé la logique de jeu et du robot, tandis que Hizir Ali a mis en œuvre l'interface utilisateur en utilisant HTML, CSS et JavaScript en étroite collaboration avec Valentin qui réalise le back-end. À mesure que les différentes parties du projet prenaient forme, nous avons travaillé sur l'assemblage de tous les éléments.
3. Tests et debug : Lorsque les différentes parties du projet ont été intégrées, nous avons procédé à une série de tests pour identifier et corriger les bugs. Après de longues sessions de correction de bugs nous avons pu tester la fiabilité du serveur en simulant des centaines de joueurs lançant des parties simultanément.
4. Tests extérieurs : Afin de pousser la recherche de problèmes nous avons proposé à plusieurs personnes de notre établissement de tester le jeu. Cette recherche a porté ses fruits puisque quelques problèmes qui ne nuisent pas au fonctionnement général mais à l'expérience utilisateur ont pu être identifiés et corrigés.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Au moment de la présentation au jury, notre projet **GridWars** se trouve à un stade avancé de développement. Voici un aperçu de l'état d'avancement des différentes parties du projet :

1. Terminé :
    - Conception et développement de l'architecture du serveur et de la logique de jeu.
    - Développement et intégration de l'interface utilisateur (HTML, CSS, JavaScript).
    - Implémentation des mécanismes de jeu, du multijoueurs et du robot.
    - Gestion des statistiques et des classements
    - Tests de base, corrections de bugs et optimisation des performances.
2. En cours de réalisation :
    - Amélioration et peaufinage de l'expérience utilisateur en fonction des retours des utilisateurs.
    - Finalisation du système de groupes : Ajouts d'options pour le créateur (expulser, bannir, changer les paramètres de la partie sans devoir recréer un groupe...)
3. Reste à faire :
    - Implémentation de nouvelles fonctionnalités et modes de jeu.
    - Développement de fonctionnalités sociales et options de personnalisation.
    - Rendre le site accessible en ligne.
    - Protection contre attaques / spam, ajouts de logs plus précis sur le serveur.

Afin de garantir la qualité et la fiabilité de notre projet, nous avons mis en place plusieurs approches pour identifier et corriger les bugs et nous assurer que GridWars est facile à utiliser :

1. Tests unitaires et tests d'intégration : Durant le développement, nous avons effectué des tests unitaires pour vérifier le bon fonctionnement des composants individuels et des tests d'intégration pour nous assurer que ces composants fonctionnent correctement ensemble.
2. Tests de performance : Nous avons effectué des tests de performance pour détecter et résoudre les problèmes de latence, de temps de chargement et de bande passante, afin de garantir une expérience de jeu fluide et réactive.
3. Tests utilisateurs : Nous avons sollicité des retours de la part de nos camarades et enseignants pour évaluer la facilité d'utilisation de GridWars et recueillir des suggestions d'amélioration. Cela nous a permis d'identifier les problèmes d'ergonomie et de les résoudre au fur et à mesure.

Au cours du développement de **GridWars**, nous avons rencontré plusieurs défis et difficultés, notamment :

1. Synchronisation et communication entre le serveur et les clients : Pour assurer une expérience de jeu fluide et réactive, il était crucial de gérer correctement la synchronisation et la communication entre le serveur et les clients. Pour surmonter ce défi, nous avons utilisé des techniques de programmation asynchrone et des protocoles de communication tels que WebSocket.
2. Conception d'une interface utilisateur intuitive : Concevoir une interface utilisateur à la fois attrayante et facile à utiliser pour un jeu multijoueurs compétitif a été un défi. Les aides extérieures au projet ont été d'une grande utilité.
3. Adaptation aux changements et aux imprévus : Au cours du développement, nous avons dû faire face à des changements de priorités, des imprévus et des ajustements de nos objectifs. C'est par exemple le cas de la gestion des groupes, face a certains imprévu portant sur la résolution des problèmes nous avons décidés de développer à nouveau le système à l'aide de classes.

> OUVERTURE :

Nous avons déjà identifié plusieurs idées d'améliorations et de nouvelles fonctionnalités pour **GridWars** afin de le rendre encore plus attractif et captivant pour les joueurs :

1. Ajout de nouveaux modes de jeu : Proposer des modes de jeu supplémentaires, tels que des défis, des modes coopératifs et des tournois.
2. Intégration des fonctionnalités sociales : Permettre aux joueurs de discuter via un chat intégré..
3. Personnalisation : Offrir aux joueurs la possibilité de personnaliser leur apparence, leur avatar et le design du plateau de jeu, ainsi que de débloquer des objets et des succès en relevant des objectifs de plus en plus hauts.
4. Tutoriels et niveaux d'apprentissage : Proposer des tutoriels interactifs pour aider les nouveaux joueurs à comprendre les mécanismes de jeu et les stratégies de **GridWars**.

Afin de toucher un large public et promouvoir **GridWars**, nous pouvons envisager d'adopter les stratégies de diffusion tel que la création de tournois compétitifs avec des prix pour les gagnants qui peuvent attirer de nouveau utilisateurs par la possibilité d’être le grand vainqueur. C'est en effet un jeu où même les meilleurs font des erreurs dans la précipitation, ainsi tout le monde a ses chances.

En résumé, voici quelques aspects que nous pourrions améliorer si nous devions recommencer le projet **GridWars** :

Organisation : Nous aurions pu planifier notre projet de manière plus structurée, en définissant des jalons clairs et en anticipant les problèmes potentiels pour éviter les retards et les imprévus. L'utilisation d'outils de gestion de projet comme Trello ou Asana aurait pu aider à mieux organiser nos tâches et à suivre nos progrès.

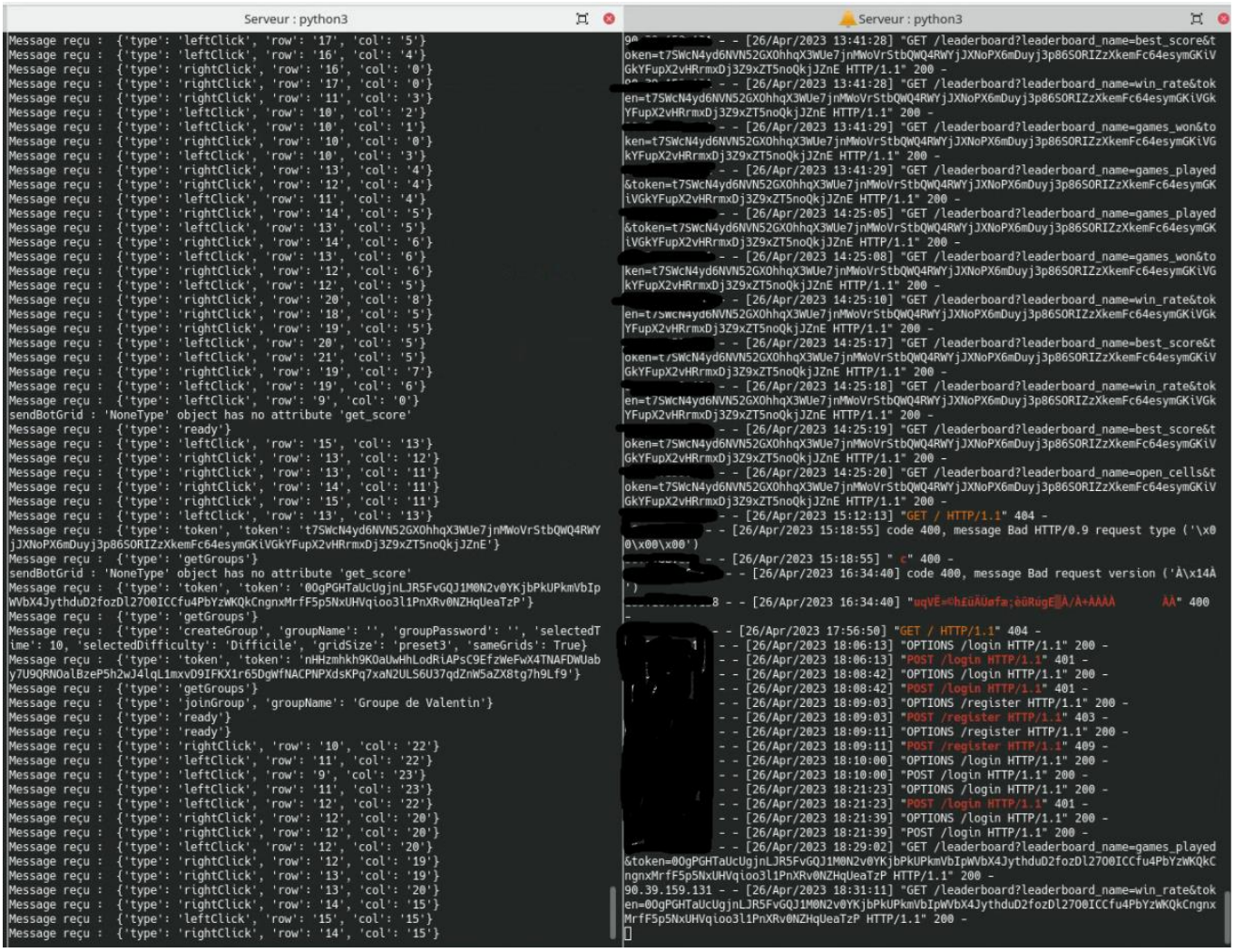
1. Choix techniques : Une réflexion plus approfondie sur les choix techniques et les outils utilisés aurait pu faciliter le développement, notamment en ce qui concerne la gestion des serveurs et la communication client-serveur. Par exemple, nous aurions pu explorer différentes technologies pour le développement du serveur et comparer leurs avantages et inconvénients avant de prendre une décision.
2. Gestion du temps : Nous pourrions améliorer notre gestion du temps en établissant des échéances plus réalistes et en consacrant du temps supplémentaire à la résolution des problèmes et à l'optimisation du code. Il est important de prendre en compte le temps nécessaire pour la recherche, l'apprentissage et les tests lors de la planification du projet, afin d'éviter la pression et le stress liés aux délais serrés.
3. Tests et qualité du code : Bien que nous ayons effectué des tests et des vérifications tout au long du développement, nous aurions pu mettre en place des procédures de test plus rigoureuses et systématiques pour nous assurer que **GridWars** est exempt de bugs et fonctionne correctement sur différents navigateurs et plateformes. De plus, l'adoption de bonnes pratiques de programmation et de normes de codage aurait facilité la maintenance et l'évolutivité du code.

*En conclusion, même si **GridWars** a été un projet réussi, il y a toujours des domaines dans lesquels nous pouvons nous améliorer pour rendre notre travail plus efficace et optimiser notre résultat. En tirant les leçons de cette expérience, nous serons mieux préparés pour relever les défis des futurs projets et continuer à apprendre et à grandir en tant que développeurs et créateurs.*

DOCUMENTATION

Informations générales :

Le projet est composé de deux parties, client et serveur. Le client comprend des fichiers HTML, CSS et Javascript, cette partie ne nécessite pas d'installation particulière si ce n'est un navigateur à jour et **un réseau qui accepte les requêtes non sécurisés HTTP et WS.** Le serveur fonctionne en permanence sur un serveur dédié (hébergé chez dedigo IP : 62.210.85.49), l'utilisateur ne doit pas s'occuper de ces fichiers. Sur le serveur, python 3.11 est requis avec les librairies suivantes : *flask*, *sqlite3*, *websockets* (en plus de celles intégrées à python)

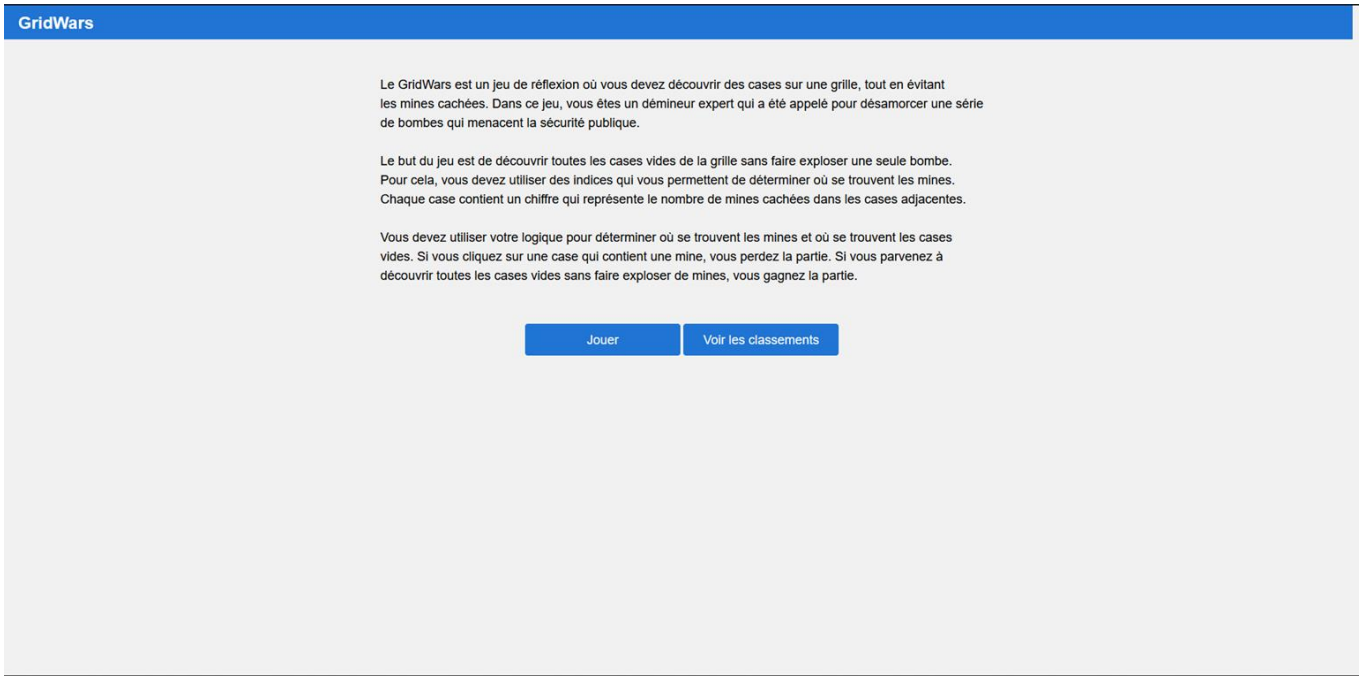


Client

Le lancement du jeu se fait par le fichier *index.html* présent dans le dossier "*Client*".
La partie client est composée de trois parties :

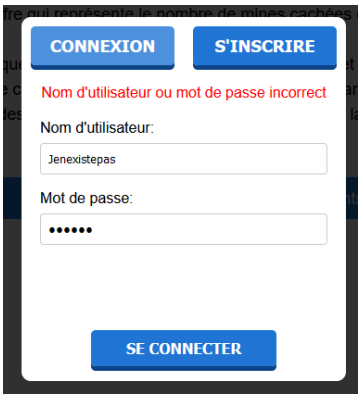
1. Home

Page d’accueil de l'utilisateur, il est possible d'y lire une courte présentation du jeu.



Le bouton jouer permet de créer un compte ou de se connecter (il redirige vers la page de jeu si l'utilisateur est déjà connecté avec un token dans le local storage).

Un message d'erreur est affiché en cas de problème, voici quelques exemples :



Une fois connecté (ou inscrit), l'utilisateur est redirigé vers la page de jeu et a maintenant un token attribué par le serveur.

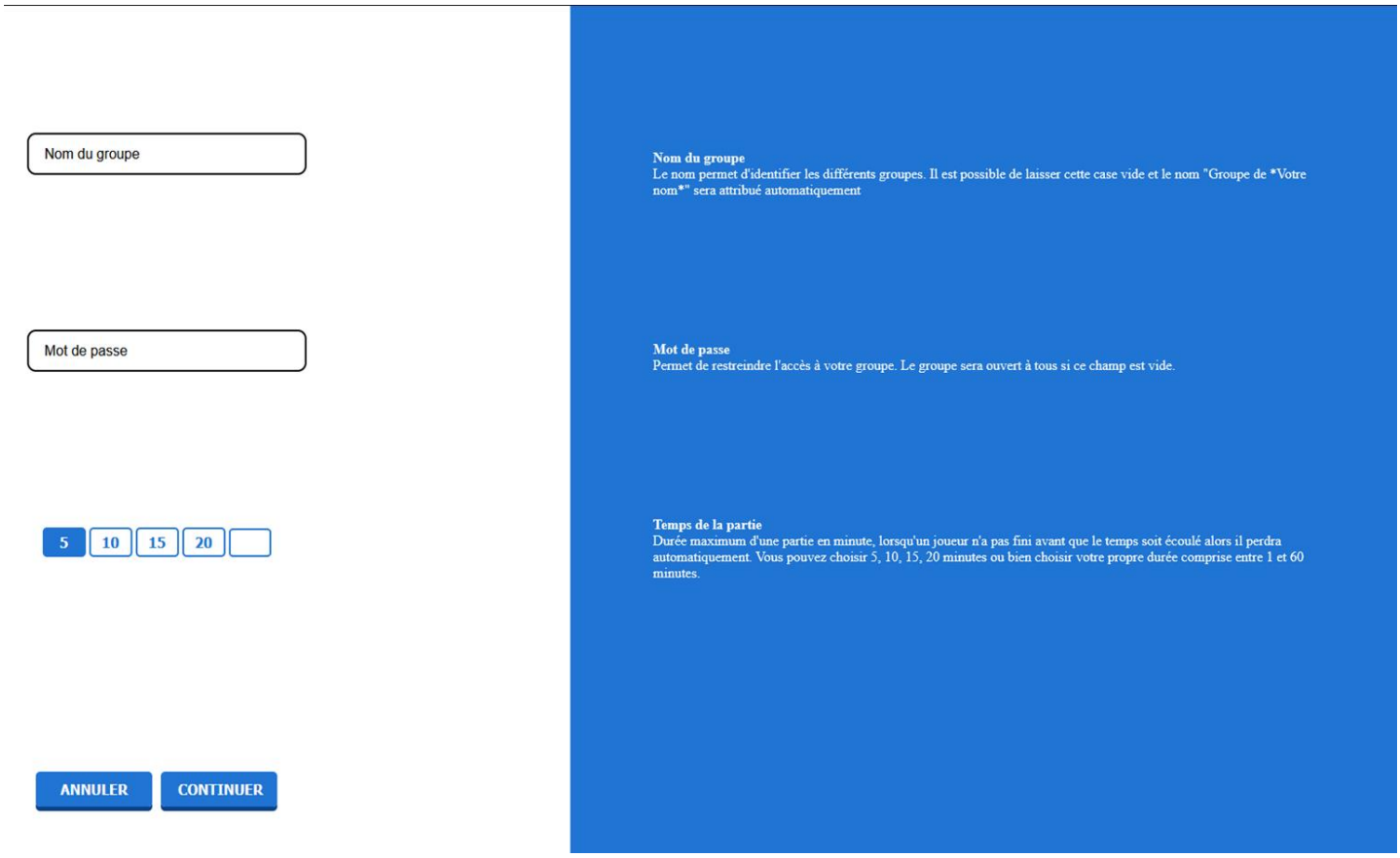
2. Play

Page principale, lorsque l'utilisateur se trouve sur cette page il est connecté au serveur par un webSocket.

Il est possible de retourner au menu (Home) ou de créer une partie.
La création de partie est séparée en deux étapes :

1) Informations générales du groupe

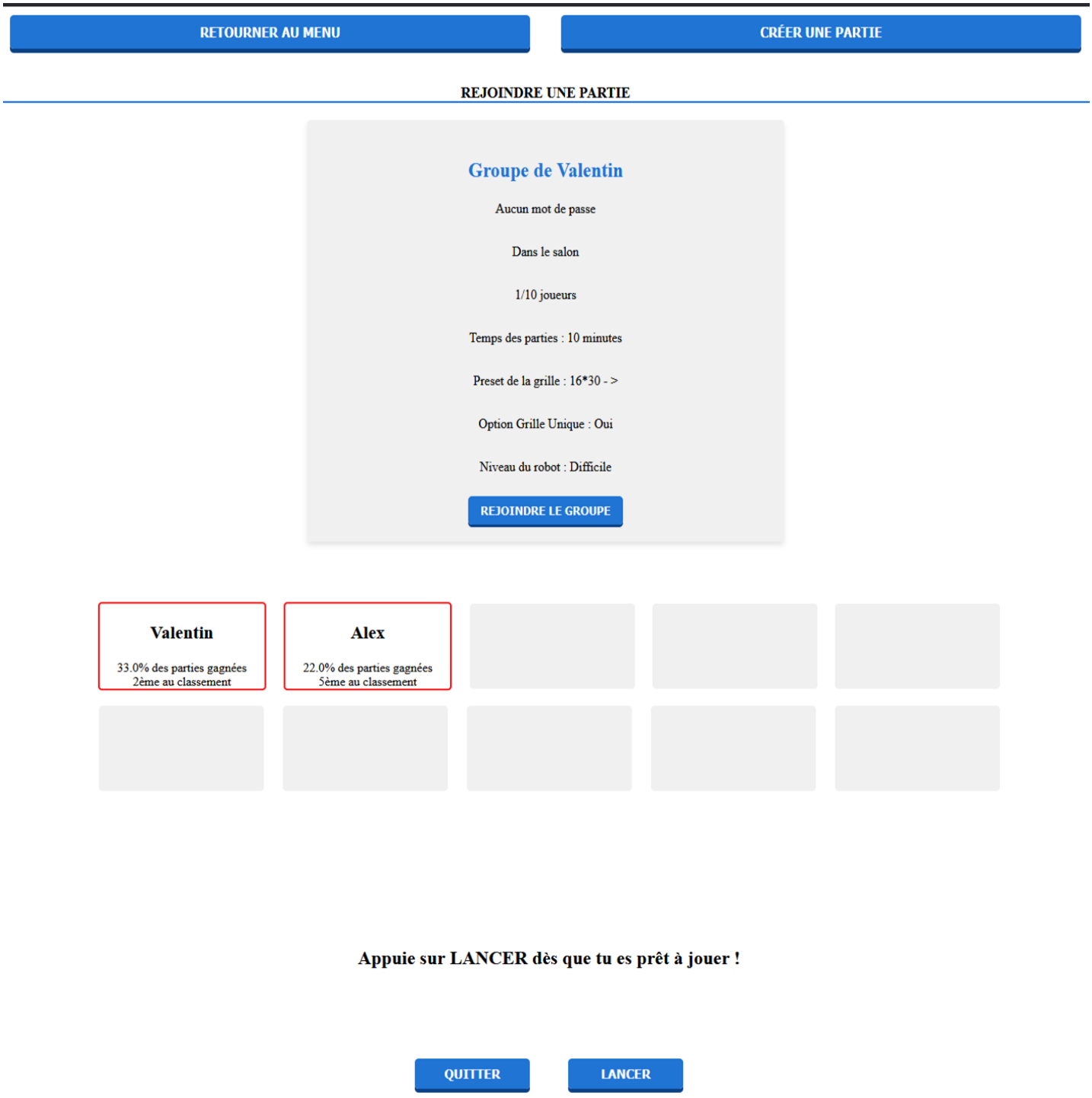
Chaque input est accompagné par sa description à sa droite, il y a le nom du groupe, le mot de passe et la durée maximum de la partie.



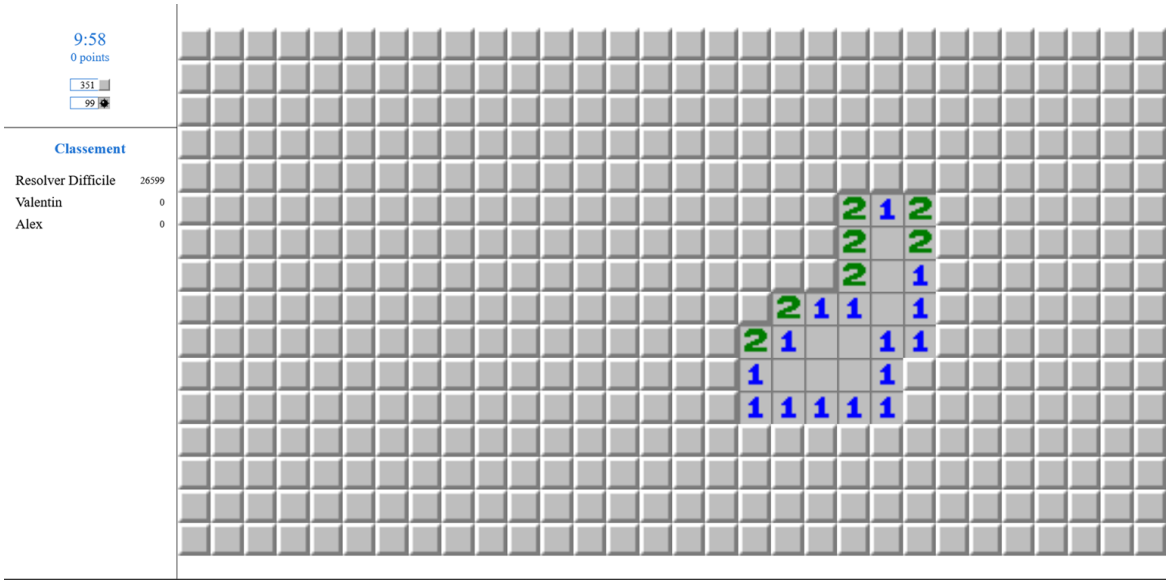
2) La seconde partie se concentre sur les paramètres de la partie.
Il est possible de choisir la taille de la grille, la difficulté du robot et l'option grille unique.
Une grille unique signifie que tous les joueurs partent sur la même grille (Seul le départ est commun, ensuite c'est chacun pour soi !), sinon chaque joueur à une grille générée aléatoirement. Cela permet d'avoir des parties plus équilibrées.

Une fois le groupe créé, l'utilisateur se retrouve dans la salle d'attente où il peut attendre ses amis ou bien lancer seul contre le robot.
Il y a un maximum de 10 joueurs par partie, chacun a un emplacement avec son nom ainsi que ses statistiques qui défilent. Le bouton lancer permet de démarrer un compte à rebours de 4 secondes, si personne ne rejoint ou quitte le groupe durant ces 4 secondes alors la partie se lance.

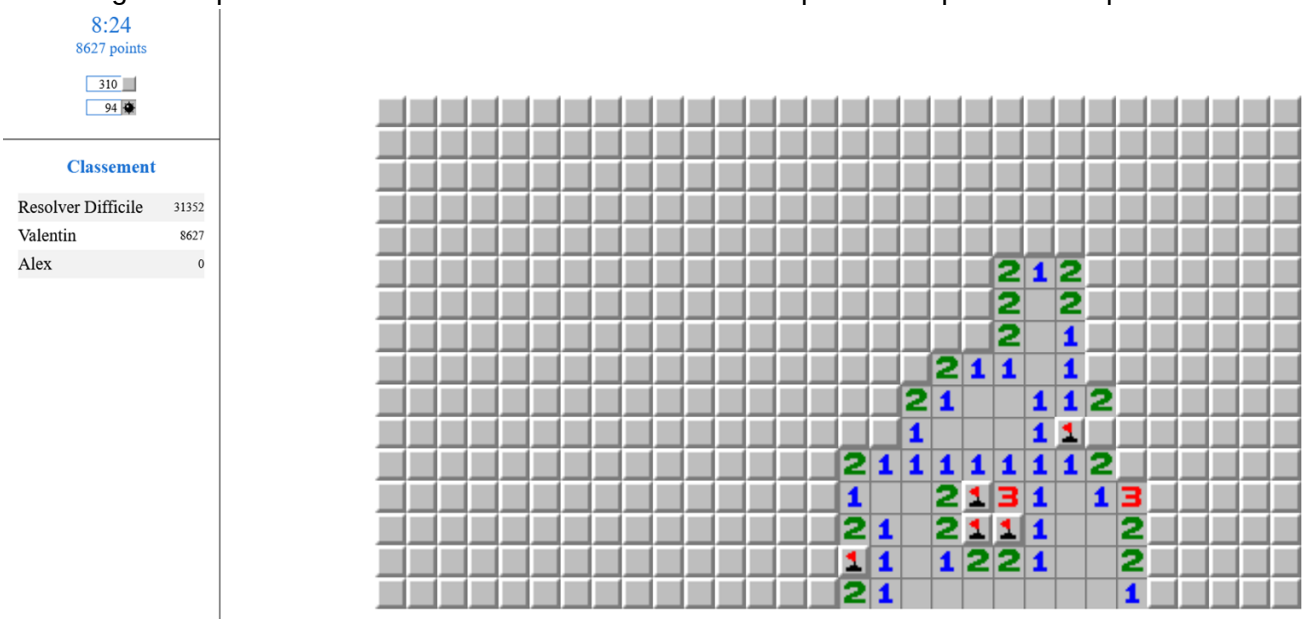
D'un autre point de vue, le groupe est disponible sur la page de jeu, il peut le rejoindre (avec le mot de passe s'il a été défini par le créateur) ou choisir de créer une autre partie.



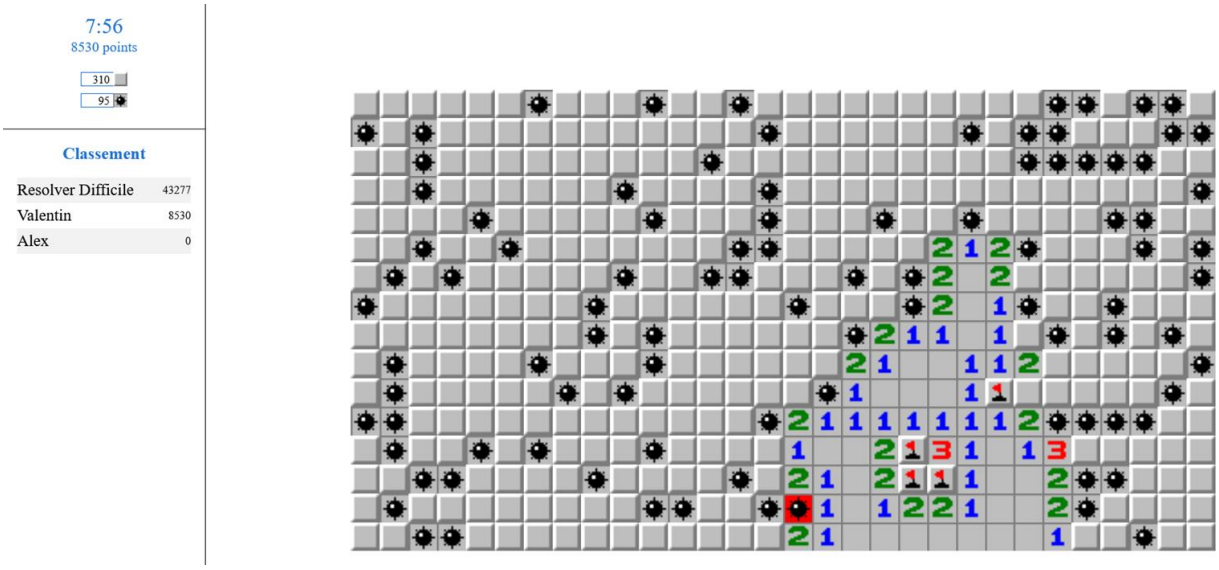
Dès que la partie est lancée, les joueurs ont chacun leur grille et un classement en temps réel basé sur le nombre de cases découvertes et le temps écoulé est disponible à gauche de l'écran.



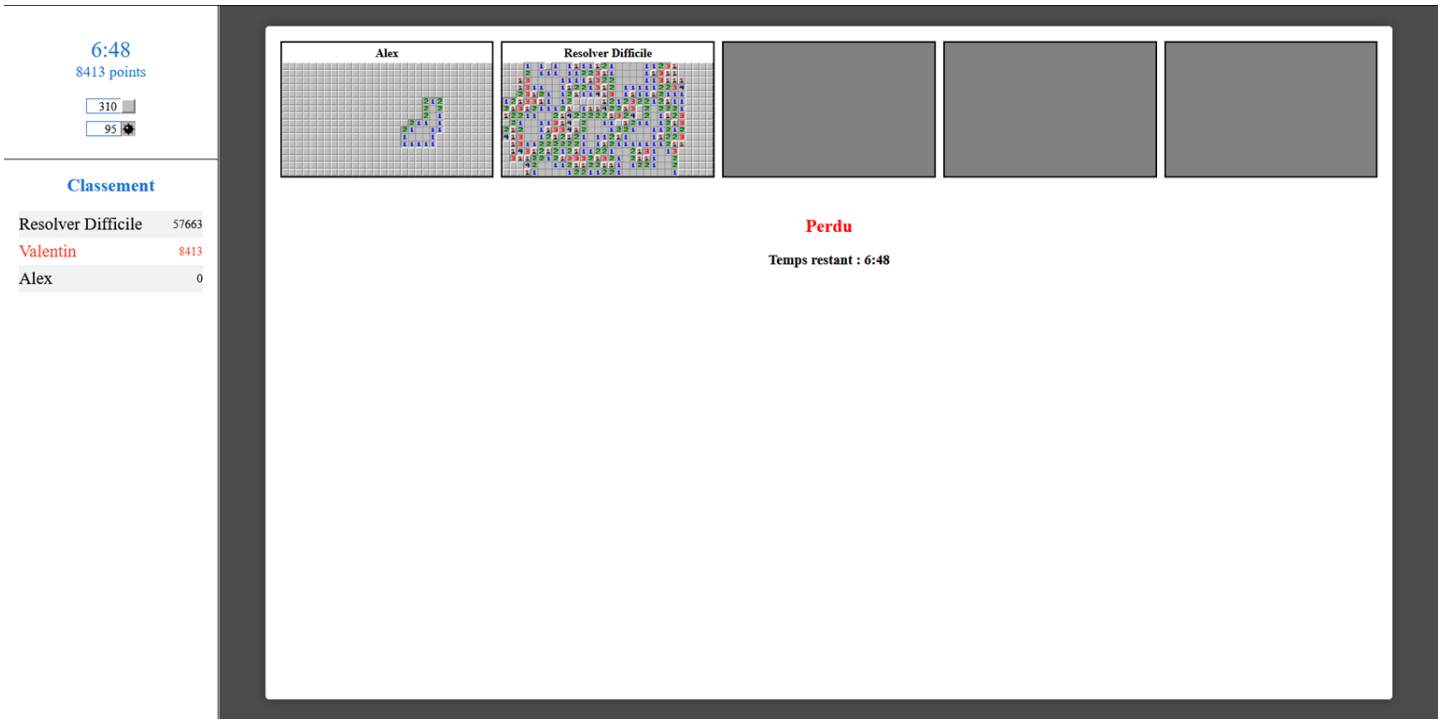
Un clic gauche permet de découvrir une case et un clic droit permet de poser un drapeau.



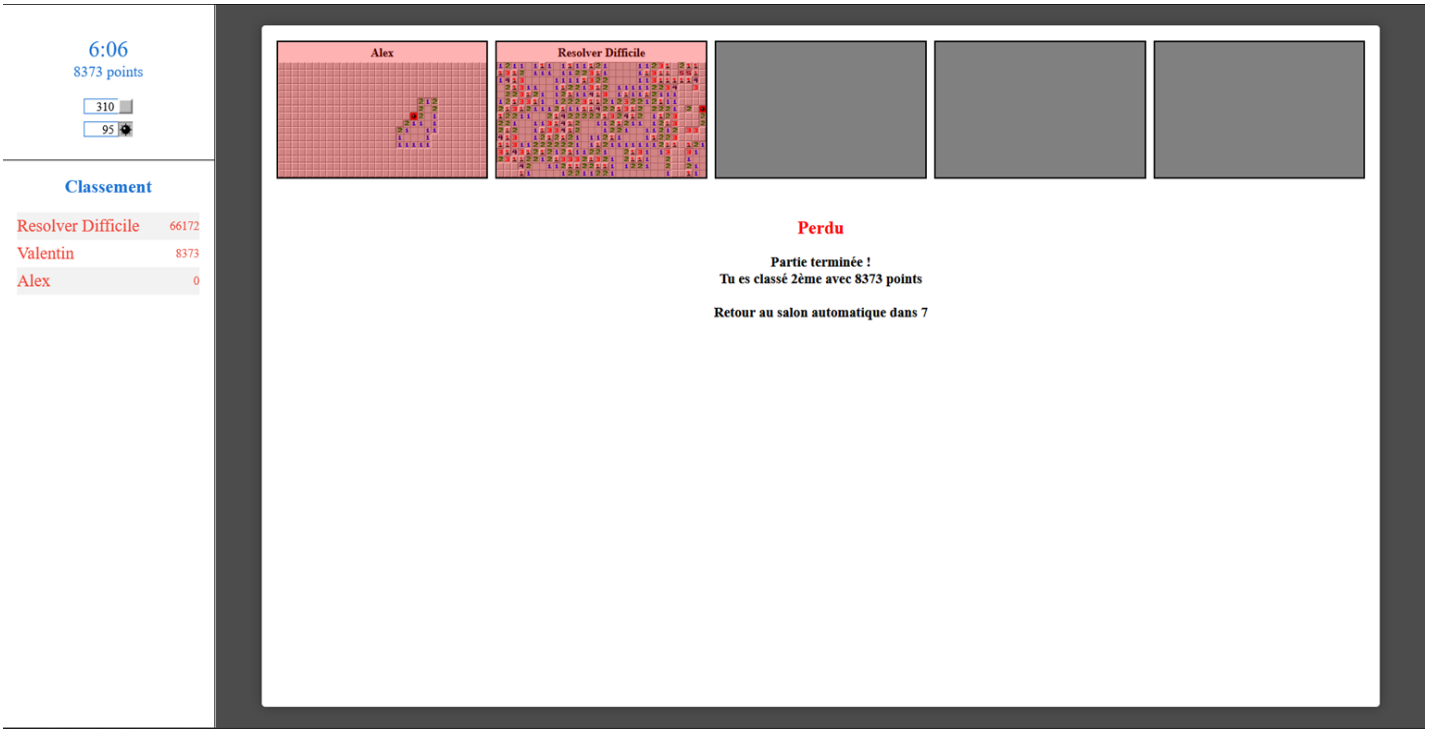
Si vous tombez sur une bombe ou que vous découvrez toutes les cases de la grille, la partie est finie pour vous.



Par chance, il est possible, en attendant que tout le monde finisse, de voir l'avancée en temps réel de chaque joueur, y compris celle du robot.

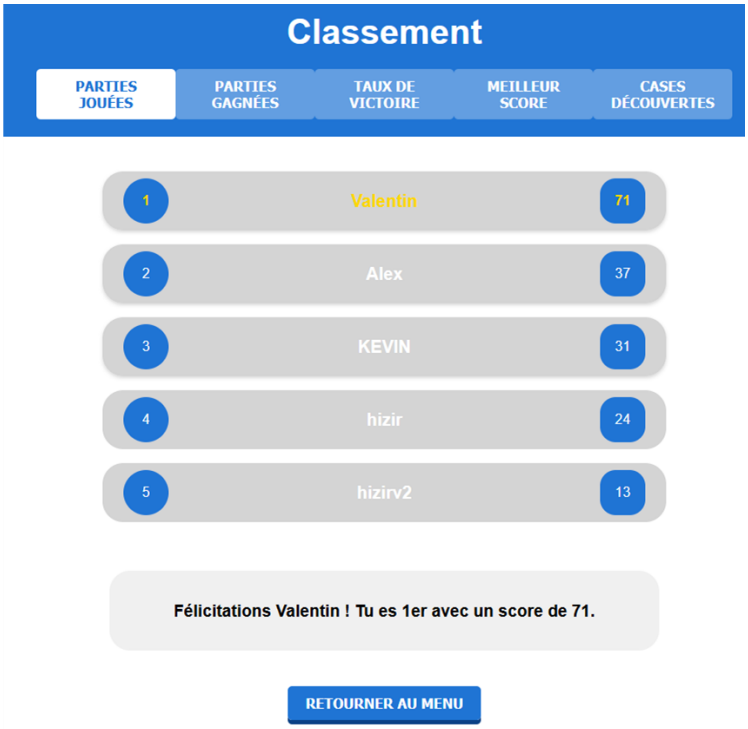


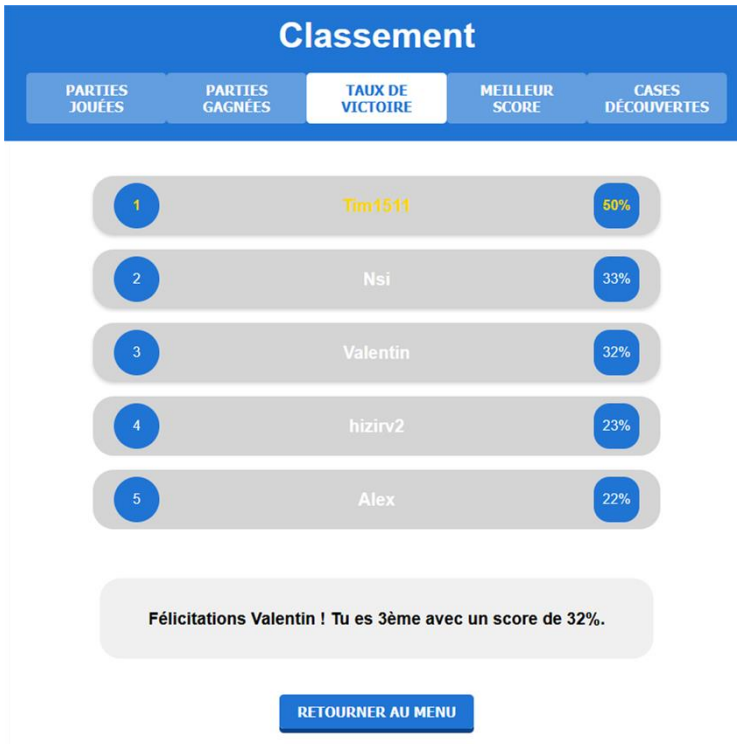
Une fois que tous les joueurs ont gagnés ou perdus, vous connaissez désormais votre classement final et vous allez rejoindre de nouveau la salle d'attente au bout de 10 secondes.



3. Leaderboard

La dernière page est celle du classement. Accessible depuis la page d'accueil, il est possible de consulter 5 classements différents : le nombre de parties jouées, de parties gagnées, le taux de victoire, le meilleur score, et le nombre de cases découvertes. Si l'utilisateur est connecté, il peut consulter son propre classement même s'il ne se situe pas dans le top 5.


Classement
PARTIES JOUÉES
PARTIES GAGNÉES
TAUX DE VICTOIRE
MEILLEUR SCORE
CASES DÉCOUVERTES
1 Valentin 71
2 Alex 37
3 KEVIN 31
4 hizir 24
5 hizirv2 13
Félicitations Valentin ! Tu es 1er avec un score de 71.
RETOURNER AU MENU


Classement
PARTIES JOUÉES
PARTIES GAGNÉES
TAUX DE VICTOIRE
MEILLEUR SCORE
CASES DÉCOUVERTES
1 Tim1511 50%
2 Nsi 33%
3 Valentin 32%
4 hizirv2 23%
5 Alex 22%
Félicitations Valentin ! Tu es 3ème avec un score de 32%.
RETOURNER AU MENU

Serveur (62.210.85.49)

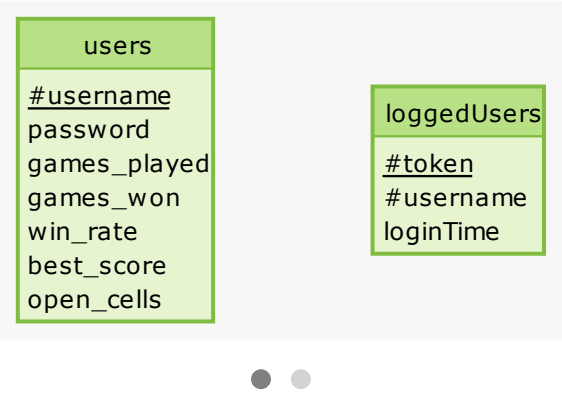
Composé de 6 fichiers, le serveur est essentiel au fonctionnement du jeu. Ces fichiers ont été déposés par ftp et les sources sont présentes dans le Github des trophées de NSI.

1) data.db

La base de données stocke les comptes des utilisateurs (avec leurs statistiques) ainsi que les tokens de connexion. Le token est régénéré à chaque connexion, ainsi il n'est pas possible d'être connecté depuis 2 appareils ou navigateurs simultanément.

Table users :
username TEXT NOT NULL (Primary key)
password TEXT NOT NULL
games_played INT
games_won INT
win_rate FLOAT
best_score INT
open_cells INT

Table loggedUsers :
token VARCHAR(128) (primary key)
username TEXT (foreign key de users)
loginTime NUMERIC



2) game.py

Contient toute la logique du jeu, une classe Case et une classe Grille

3) resolver.py

Contient une classe Bot qui prend en argument une grille de la classe Grille de game.py et un temps d'attente entre chaque action (difficulty). Le bot résous la grille par la logique, lorsqu'une situation impose un choix aléatoire le bot a plus ou moins de chance de tomber sur une bombe suivant la difficulté.

4) server.py

En utilisant Flask, 3 actions sont possibles : *login*, *register*, *leaderboard*. Il stocke et lit des informations dans la base de données.

5) socketServer.py

A l'aide de la bibliothèque websockets ce fichier gère les connexions des différents utilisateurs sur la page *Play*. Il utilise les fichiers *game*, *resolver* et *parties*.

6) parties.py

Ce fichier permet à socketServer.py de gérer les utilisateurs et les parties à l'aide d'instances d'objets *Group* et *Member*.