Université de versailles
saint-quentin-en-yvelines

Département d'Informatque MASTER-1
Matière
AWS

Réalisation d'une application web de jeu de dames multi-joueurs en ligne

---

Étudiants :
AMIROUCHE Amine
MAIGA Abdalah
Said Mohammed Seghir
DERRA Abdoul Rachid
KHAMARI Nor El Khitem

Enseignants :
PERRIN Léo
YANN Rotella

7 mai 2020

Table des matières

1 Introduction

Un jeu en ligne est un jeu électronique doté d'une interface utilisateur permettant une interaction permanente du client avec un ou plusieurs serveurs, en portant une attention particulière aux processus communicationnels, on constate que les données partagées relative au jeu et aux joueurs peuvent contenir des attaques ou des erreurs, qui mettent l'application hors service. Dans le cadre du module Application Web Sécurisé, nous allons réaliser un jeu de dames en ligne, permettant une interaction client serveur en temps réel tout en veillant à ce qu'elle soit sécurisée.

2 Conditions exigées pour la réalisation du projet

Un jeu en ligne est une application client-serveur :

1. Le serveur doit gérer les connections, les déconnections, les sauvegardes ainsi que le transfert de données entre le client et le serveur, il est en intéraction avec la base de données pour la sauvegarde et le chargement de données reliées au jeu.
2. Dans le cas de notre application en ligne, le client héberge toute la logique du jeu de dames. Le client (le programme joueur dans notre cas) se connecte au serveur pour jouer avec d'autres clients (joueurs). Pour se connecter au serveur, une inscription et/ou une authentification est requise. Le joueur connecté alors peut défier d'autres joueurs connectés, à une partie. Plusieurs clients peuvent se connecter à la fois au serveur initiant plusieurs parties du jeu simultanément.

Le cahier des charges de notre projet impose les spécifications fonctionnelles suivantes :

1. Une page permettant au joueur de s'enregistrer.
2. Une page présentant la liste des utilisateurs en ligne, leur nombre de parties gagnées/perdues, et autres informations éventuelles.
3. Un mécanisme pour se connecter.
4. Un mécanisme permettant de défier un adversaire.
5. Une interface permettant de jouer le jeu.

L'interface du jeu doit être réalisée en JavaScript :

1. Les pions sont disposés au début la partie, et c'est au blanc de jouer.
2. Le joueur sélectionne le pion à jouer en cliquant dessus, puis il clique sur la case où le pion doit se déplacer. Le serveur vérifie que le coup est valide, et prend le pion adverse le cas échéant.
3. Lorsqu'un joueur a perdu tous ses pions, ce joueur est déclaré perdant et le jeu se termine.

3 Définition du jeu de dames

Le jeu de dames est un jeu à deux joueurs qui se joue avec des pions (de deux couleurs différentes, généralement noirs ou blancs), disposés sur les cases noires d'un damier de 10x10 cases, chaque joueur dispose de 20 pions.
Le but du jeu est de capturer les pions de l'adversaire (nous parlons alors de prise de

pions) ou de les mettre dans l'impossibilité de bouger.
Peuvent se résumer comme suit :

1. Les joueurs jouent à tour de rôle, les pions blancs commencent.
2. Conditions de fin de partie :
   Une partie de dames se termine si :
   (a) Un joueur ne peut plus bouger, même s'il lui reste des pions, il perd la partie.
   (b) Un adversaire quitte la partie.
3. Déplacement du pion :
   (a) Un pion ne peut se déplacer que d'une case, en diagonale vers l'avant (deux choix possibles : diagonale avant droite, ou diagonale avant gauche).
   (b) Lorsqu'un pion arrive sur la dernière ligne adverse de l'échiquier, il est promu en *Dame*.
4. Déplacement de la Dame :
   (a) La dame se déplace sur une même diagonale d'autant de cases qu'elle le désire, en avant et en arrière.
5. La prise par la dame :
   (a) La dame doit prendre tout pion situé sur sa diagonale et doit changer de direction à chaque fois qu'une nouvelle prise est possible.

4 Conception du jeu

4.1 Conception du jeu de dames en ligne

Avant de nous lancer dans le développement de notre projet, nous nous sommes réunis et avons réfléchi à la conception ou une perception future de notre application client-serveur.
L'architecture qui s'en est dégagée fait ressortir :

1. *côté client : Le client du jeu s'articule autour des principales classes suivantes :*
   (a) la classe pièce : cette classe définit les pièces du jeu, avec leurs caractéristiques (type de pièce pion, dame, et sa couleur).
   (b) la classe square : définit chaque case du damier.
   (c) La classe move : définit le déplacement des pièces sur le damier.
   (d) La classe damier : les règles du jeu sur un damier. Un damier est composé de 10x10 squares, et comporte 20 pièces du jeu (pions) de deux couleurs différentes.
   (e) La classes dames : cette classe initialise le damier et récupère les sockets des joueurs adverses, et permet l'interaction avec le serveur au moment du jeu.
2. *côté serveur : Le serveur s'occupe principalement des connexions et des interactions inter-joueurs.*
   Le code est réparti en différents répertoires et fichiers comme suit :
   (a) Fichier app.js : fichier d'initialisation de l'application qui permet d'initialiser le socket, de se connnecter à la base de donnée, de créer une session et de rajouter toutes les routes requises.
   (b) Fichier wssocket.js : permet d'interagir avec le client (réception de messages au moment de la connexion du client, transmission des différentes actions au client)

4.2 Conception du site d'accès au jeu

Pour jouer au jeu de dames en lignes, il faut se connecter à la page web du jeu. Pour cela, une inscription et/ou une authentification préalable est requise. L'architecture de la page du jeu s'articule autour de :

— Page enregistrement : qui permet à un utilisateur de s'enregistrer avant qu'il ne puisse accédér aux ressources, cette page gére notamment les différents aspects de sécurités WEB :

1. Interdiction des caractères html dans les champs à remplir .
2. Limitation de la taille du pseudo.
3. 8 caractères minimum requis pour le mot de passe .
4. Les champs à renseigner sont vérifiés avant d'effectuer n'importe quelle requête côté serveur. Nous utilisons une règle REGEX pour vérifier le pseudo que saisit l'utilisateur. Ce contrôle nous permet d'avoir des informations cohérentes dans notre base de donnée et d'éviter éventuellement les attaques par injections sql.

Lors de son enregistrement, l'utilisateur doit renseigner son :

1. Pseudo.
2. Email.
3. Mot de passe.
4. Confirmez son mot de passe.

Les données fournies par l'utilisateur lors de son inscription sont analysées par le serveur.
En effet, le serveur vérifit si l'utilisateur a bien renseigné tout les champs du formulaire, verifie si les mots de passe renseignés sont identiques puis si l'email renseigné n'existe pas dèjà dans la base de données, il verifie également si la taille minimum du mot de passe est de 8 caractères avant l'enregistrement des infomations de l'utilisateur, le mot de passe de l'utilisateur est haché avec le module bcrypt.
Après annalyse, les données fournis seront stockés dans une base de donnée cloud MongoDB (No-SQL) MongoDB Atlas puis l'utilisateur est redirigé vers la page de connexion, afin de se connecter avec ses identifients d'inscription qui seront aussi analysés par le serveur.

— Page de Connection : permet à l'utilisateur de se connecter, il doit renseigner son :

1. Email.
2. Mot de passe.

Lors de la Connexion, les données fournies par l'utilisateur sont analysées par le serveur.
En effet nous utilisons le middleware d'authentification Passport qui dispose d'un ensemble de stratégies pour authenifier la demande de connexion de l'utilisateur.

— Page d'accueil (dashboard) : une fois l'utilisateur connecté, il sera redirigé sur une page d'acceuil où il lui sera rappelé quelques règles pour qu'il commence à jouer, ce dernier pourra avoir aussi accès à la liste des utilisateurs connectés , un bouton défier apparaitra devant chaque utilisateur connecté.

— Dans la page d'accueil le joueur peut voir dans la liste le nombre de parties gagnées de chaque joueur connecté ainsi avoir une idée du niveau de l'adversaire à défier.

— Page du défi(Play) : en cliquant sur défier, l'utilisateur est envoyé vers une page où le jeu lui sera visible après que son adversaire ait accepté le défi.

— Captcha.js : permet de rajouter le captcha au moment de l'enregistrement.

5 Outils de développement utilisés pour la réalisation du projet

Dans cette partie du rapport, nous allons présenter les outils, logiciels utilisés afin de réaliser notre projet.

1. NODE.JS :
   — Node. js est une plateforme basée sur le moteur d'exécution JavaScript de Chrome, il permet de créer facilement des applications réseau rapides et évolutives. Il utilise un modèle d'E/S non bloquant, piloté par les événements, qui le rendent léger et efficace, parfait pour les applications temps réel à forte intensité de données qui s'exécutent sur des appareils distribués.
2. EXPRESS.JS :
   — C'est un framework pour construire des applications web basées sur Node.js. C'est le framework standard pour le développement de serveurs en Node.js. A partir d'Express, nous avons créé diverses routes pour les différentes pages du site. Ensuite, nous avons établi la connexion à notre base de données à travers le client MongoDb (Mongoose).
3. BCRYPT :
   — Pour la gestion des mots de passes, nous avons utilisé bcrypt. En effet, les mots de passes sont hashés avant d'être stockés. Lors de la connexion, pour garantir l'authentification de l'utilisateur, le serveur vérifie le hashé du mot de passe renseigné et celui stocké.
4. PASSPORT :
   — En ce qui concerne la connexion des utilisateurs nous utilisons Passport. Il s'agit d'un middleware d'authentification pour Node.js. Passeport implémente le concept de stratégies pour authentifier les demandes, les stratégies peuvent aller de la vérification des informations d'identification du nom d'utilisateur et du mot de passe, à l'authentification déléguée à l'aide d'OAuth (par exemple, via Facebook ou Twitter) ou l'authentification fédérée à l'aide d' OpenID .
5. BODY-PARSER :
   — Pour gérer la demande POST provenant de l'application front-end, nous devrons être capables d'extraire l'objet JSON de la demande. Nous avons donc utilisé le package body-parser qui extrait la partie entière du corps d'un flux de demande entrant et l'expose sur req.body.
6. MONGOOSE : Mongoose est un module Node.js qui nous a servi de passerelle entre notre serveur Node.js et notre base de données MongoDB.
   Côté serveur : Nous avons utilisé les technologies suivantes :

(a) EJS : Nous l'utilisons au niveau front-end. C'est un moteur de template utilisé par Node.js. Il permet de créer un template HTML avec un minimum de code. Il peut également injecter des donnés au niveau client et produire le HTML final.

(b) WS : (WebSockets) Pour assurer les communications bidirectionelles entre un client et un serveur , nous avions le choix entre les websockets **WS** .
A l'opposé des websockets Ajax a une technique unidirectionnelle basée sur un paradigme à requête –> réponse, elle permet au client et au serveur d'échanger des informations sans recharger la page, mais elle a l'inconvénient que ce soit toujours le client qui demande et le serveur qui répond. Le serveur ne peut donc pas décider de lui-même d'envoyer des informations au client et d'initier des actions pour le client comme par exemple l'exécution de fonction à distance.
Pour pallier ce problème nous nous sommes reportés sur les **web sockets**, qui permettent la communication en temps réel, mais aussi un échange bilatéral synchrone entre le client et le serveur.
En effet, grâce aux **web sockets** chaque utilisateur connecté initie une connexion auprès du serveur, ainsi nous avons la possibilité d'afficher une liste des utilisateurs connectés(en ligne). Les websockets permettent également de communiquer aux joueurs les actions des adversaires.

6 Perspectives d'amélioration

Il existe certaines fonctionnalités que nous avons implémenté, mais pas eu le temps de déboguer, elles seront réglées d'ici la présentation de l'application :

— Le mode simultané (un joueur affrontant plusieurs joueurs sur différents plateaux simultanément) a été implémenté mais présente néanmoins un bug que nous n'avons pas eu le temps de corriger.

— L'outil captcha avec la vérification d'image a été aussi implémenté dans l'application mais présente un bug, que nous éspérons corriger.

7 Conclusion technique

Notre application fonctionne correctement, en effet toutes les règles du jeu ont été respectées comme la prise obligatoire, elle offre aussi au joueur la possibilité de voir les différents déplacements possibles en un simple clique sur la pièce à déplacer, ainsi que de savoir le nombre de parties gagnés et perdues de tous les utilisateurs connectés, ainsi que de lui-même. Tous les champs renseignés par un utilisateur sont soigneusement inspectés par le serveur afin de garantir la sécurité de l'application.

8 Conclusion générale

Notre équipe s'est mobilisée durant 8 semaines afin de réaliser l'application Web jeu de dames, ce projet fut l'occasion pour les membres de l'équipe de mettre à profit

les acquis et les bonnes pratiques acquises dans (l'UE Applications Web Sécurisées), en effet tout les les membres de l'équipe étaient d'accord dès la première réunion pour dire que, l'organisation serait la clé de la réussite de notre projet, pour cela, il nous tenait à cœur de réaliser une application à hauteur des exigences ou même plus, ceux en élaborant des plannings qui ont toujours été respectés, ainsi qu'une répartition équitable des tâches.

Nous tenons à remercier l'ensemble de l'équipe pédagogique, qui nous ont guidé dans la réalisation de cette application.