Mini-Projet GEN

Gamee Librairy

HEIG-VD
Avril 2018

Auteurs : Lionel Nanchen, Olivier Nicoulaz, Mentor Reka, Christophe Joyet
Professeur responsable : M. Eric Lefrançois

Table des matières

Introduction

Nous avons pour but de créer la « Gamee Library ». Le but de notre projet est d'informatiser les collections de jeux de société que les familles emmènent souvent avec eux pour des voyages.

Gamee Library propose une librairie de jeux de société pour deux joueurs. Tel que le tic-tac-toe, puissance 4, le pendu. D'autres jeux peuvent être ajoutés au fur et à mesure.

Fonctionnement Général

Utilisateur

Un utilisateur connecté à un serveur de jeux, peut choisir le jeu auquel il veut jouer. Afin de jouer, il doit soit accepter une demande de jeu, soit qu'il fasse une demande et qu'elle soit acceptée.

Admin

Un administrateur peut héberger des serveurs de jeux, ce dernier se laissera le droit de choisir les jeux qu'il voudra, le nombre de joueur maximum sur le serveur et d'autres propriétés de configuration qui seront définies en temps voulu.

Base de données

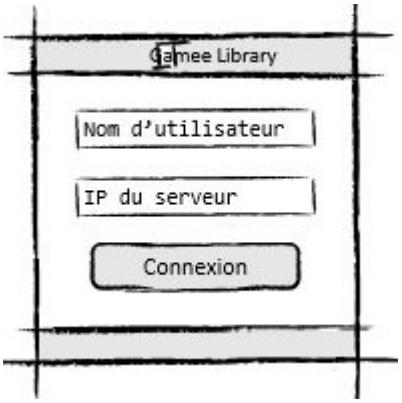
La base de données contiendra les listes des joueurs et leurs scores. Pour ce faire, nous utiliserons un système de gestion de base de données MySQL.

Communication client-serveur

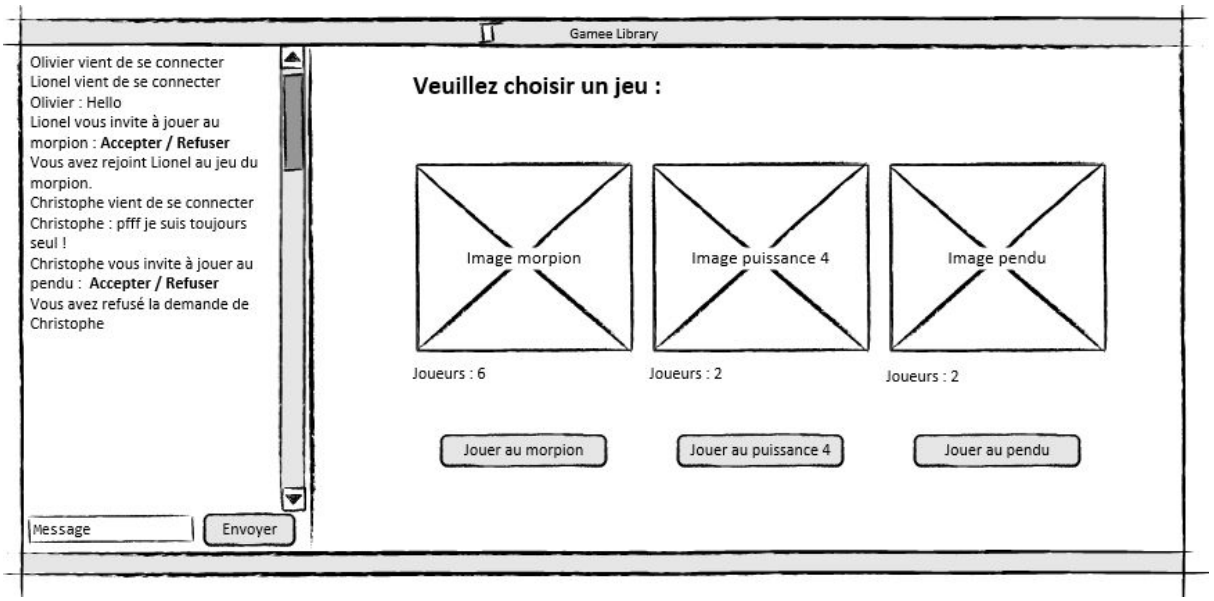
Afin de jouer, un des utilisateurs va se connecter sur un serveur centralisant les jeux et les informations relatives aux parties. Pour ce faire, il renseigne l'adresse IP publique du serveur ainsi que le port utilisé pour la communication.

Interface / Mokups

Lorsque l'utilisateur lance l'application cliente, il est invité à fournir un nom d'utilisateur et l'adresse ip du serveur. L'application tentera de se connecter au serveur à l'adresse IP fournie et lui enverra le nom d'utilisateur qui servira aux interactions entre joueurs.

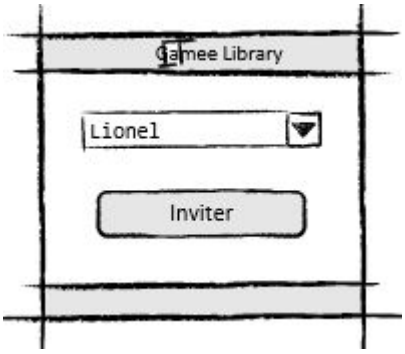


Une fois connecté au serveur, la page suivante est affichée :

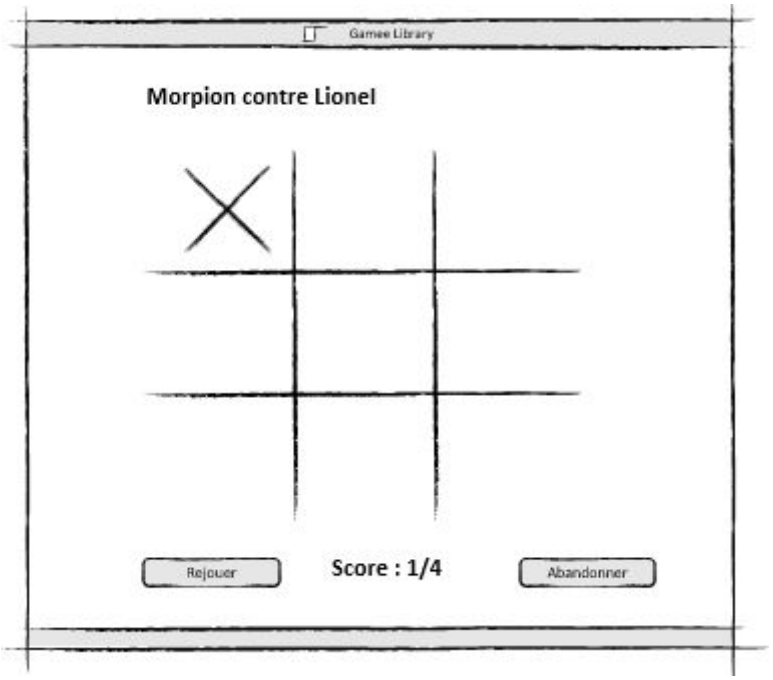


À gauche, nous trouvons un chat qui permet aux joueurs de voir les invitations en attente, les personnes qui se connectent ainsi que les messages des autres joueurs. On peut imaginer avoir des commande commençant par "/" afin d'effectuer différentes actions (exemple "/connected" pour afficher une liste des joueurs connectés au serveur).
Dans la partie de droite, on retrouve l'accès aux différents jeux proposés par le serveur auquel on est connecté, juste en dessous de l'image on peut voir combien de joueurs sont en ce moment même en partie dans ce jeu.

Lors du clic sur un bouton “Jouer au …” la fenêtre suivante s’affiche :

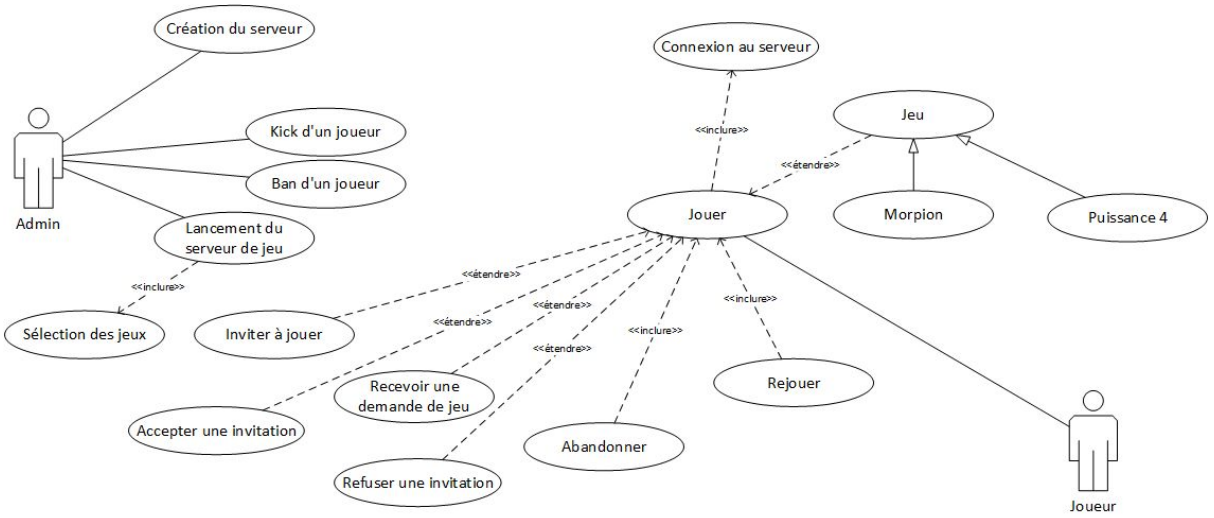


Elle permet d’inviter un joueur pour jouer à un certain jeu. Dès que ce joueur accepte l’invitation, la page du jeu s’ouvre et permet aux deux joueurs de jouer.



*Ces mockups sont des versions préliminaires de l’application et sont susceptibles de changer durant l’analyse des fonctionnalités

Diagramme des cas d'utilisation



1. **Création du serveur :**

L'utilisateur lance l'application serveur et renseigne le port qu'il veut utiliser pour la communication avec les clients.

1. **Sélection des jeux :**

L'administrateur peut choisir quels jeux il veut pouvoir mettre à disposition des clients.
Il peut choisir parmi une liste de jeux prédéfini.

1. **Kick d'un joueur :**

Si un joueur (non-admin) n'est pas appréciable sur le réseau de l'admin, cet admin peut virer ce joueur.

1. **Ban d'un joueur :**

Quand il lui semble nécessaire ou que le besoin s'en fait ressentir de la part des autres utilisateurs, l'administrateur peut bannir un joueur aux comportements étranges.

1. **Lancement du serveur de jeu :**

Une fois les configurations renseignées par l'administrateur, il clic simplement sur un bouton "Démarrer le serveur de jeu" qui le met en ligne à partir de ce moment le serveur est à l'écoute des connexions

1. **Connexion au serveur :**

Ce cas est requis par le joueur afin d'effectuer toutes les autres actions. Nous n'avons pas représenté ces liens sur le diagramme afin qu'il reste lisible.

1. **Inviter à jouer :**

Un joueur peut inviter un autre joueur également connecté au serveur. Sa demande est affichée sur le chat du jeu.

1. **Recevoir une demande de jeu :**

Un joueur peut inviter un autre joueur à rejoindre une partie.

1. Accepter :
Quand un joueur demande à un autre joueur s'il veut faire une partie avec ce dernier et a le choix entre accepter et refuser. Il sera nécessaire d'avoir reçu une invitation afin de pouvoir effectuer cette action.

1. Refuser :
Quand un joueur demande à un autre joueur s'il veut faire une partie avec ce dernier et a le choix entre accepter et refuser. Il sera nécessaire d'avoir reçu une invitation afin de pouvoir effectuer cette action.

1. Jouer :
Lorsqu'un joueur accepte l'invitation d'un autre joueur, une partie de jeu démarre et le joueur commence à "Jouer"

1. Abandonner :
Si un joueur ne veut plus jouer au jeux courant, il peut abandonner la partie ce qui signifiera la perte de la partie pour ce dernier.

1. Rejouer :
Lorsque la partie se termine, les deux joueurs ont la possibilité de demander à l'autre une revanche à l'aide de l'action "Rejouer"

1. Jeu :
Plateforme sur laquelle les joueurs vont s'affronter dans un duel sans merci.

1. Morpion :
Exemple d'un jeu disponible lors du choix de l'utilisateur.

1. Placer un rond ou une croix :
Dans le jeu du Morpion, les joueurs à chaque tour, doivent poser une pièce dans une case. Ces pièces sont des croix (pour le premier joueur) et des ronds (pour le second joueur).

Responsabilités client-serveur

Dans le cadre de ce projet, l'intelligence est concentré autour du serveur. En effet, c'est lui qui va assumer la gestion et la coordination de tous les clients. Le client agit comme une interface qui va se limiter à lire les informations du serveur et envoyer les actions (événements) effectués par l'utilisateur.

Ci-dessous, une description des responsabilités des clients et du serveur :

Client :

- Etablir une connection sur un port et une IP spécifique à un serveur de jeux.
- Communiquer avec le serveur afin de lister les jeux, le chat et pouvoir effectuer les différentes actions à disposition (voir cas d'utilisation).
- Mettre-à-jour l'interface cliente à chaque événement concerné sur le serveur.
- Fermer la connection sur le serveur.

Serveur :

- Mettre à disposition un port sur lequel les clients peuvent se connecter.
- Réceptionner et traiter les connection clientes.
- Traiter et ordonnancer le jeux entre différents clients.
- Mettre à disposition un chat commun.
- Persister les informations.

Un exemple illustrant la philosophie du jeux est le Morpion. Typiquement, lorsque deux joueurs (J1 et J2) jouent au Morpion sur un serveur (S1) voici une vue non-exhaustive de ce qui va se passer :

- S1 va créer une partie entre J1 et J2 et choisit aléatoirement celui qui va commencer. J1 est choisi et informé.
- Le client va afficher à J1 le jeux du morpion et l'invite à effectuer le premier coup.
- J1 clique sur une case. Le client transmet (conformément au protocole) la case coché au serveur.
- Le serveur analyse et transmet au client J2 la mise-à-jour du jeux avec le nouveau coup de J1 et lui donne la main.
- Finalement, lorsqu'un joueur a gagné, le serveur termine le jeux et informe les deux joueurs J1 et J2 du résultat.

On remarque que notre architecture est centralisé autour du serveur et toute la logique métier y est concentré.

Protocole

Le protocole ci-dessous va définir le format d'échanges entre les clients et les serveurs. Nous avons choisit de distinguer deux parties en deux sockets distincts :

1. La première partie illustré dans les MockUps correspond au chat qui se verra attribuer une "ligne" de communication (un socket asynchrone) afin de communiquer avec tous les clients.
2. La deuxième partie concernant une partie correspond à une ligne synchrone entre un joueur J1 et un joueur J2.

Les informations transmises seront
Voici, le protocole d'échange (les commandes) établie pour la partie CHAT :

- **List** : récupère la liste des messages (auteurs, date et le message)
- **Message (Pseudo)** : envoie un message à l'ensemble des clients (avec l'auteur). Exemple:

  Message (Pseudo): Bonjour à tous

Voici, le protocole d'échange (les commandes) établie dans le cadre générale et le cadre d'une partie :

- **List games** : récupère la liste des jeux (id du jeux, nom du jeux, nombre de joueurs)
- **List users** : récupère la liste des utilisateurs
- **Play <game_id> <user_id>** : propose une partie à l'utilisateur <user_id> pour le jeux <game_id>

- Le protocole pour jouer une partie d'un jeu. Chaque jeu respectera un protocole générale avec ses propres spécificités
  - action

Modèles de domaines client & serveur // OLIVIER NICOLE

Prototypes éventuels // OLIVIER NICOLE

Base de données

L'objectif principal de notre base de données sera de conserver les données émises par le client au niveau du serveur ainsi que les diverses configuration de nos jeux.

Pour les clients, nous serons amenés à garder :

- nom et prénom
- adresse mail
- ces droits (Si c'est un administrateur ou non)
- le nombre de bannissement subis

Pour les jeux, les propriétés de chacun seront enregistrées et comprendront notamment :

- nom du jeu
- règles du jeu

Nous aurons également un chat dans lequel seront envoyés plusieurs message afin que les utilisateurs puisse discuter entre eux.

Modèle Conceptuel de la base de données

-