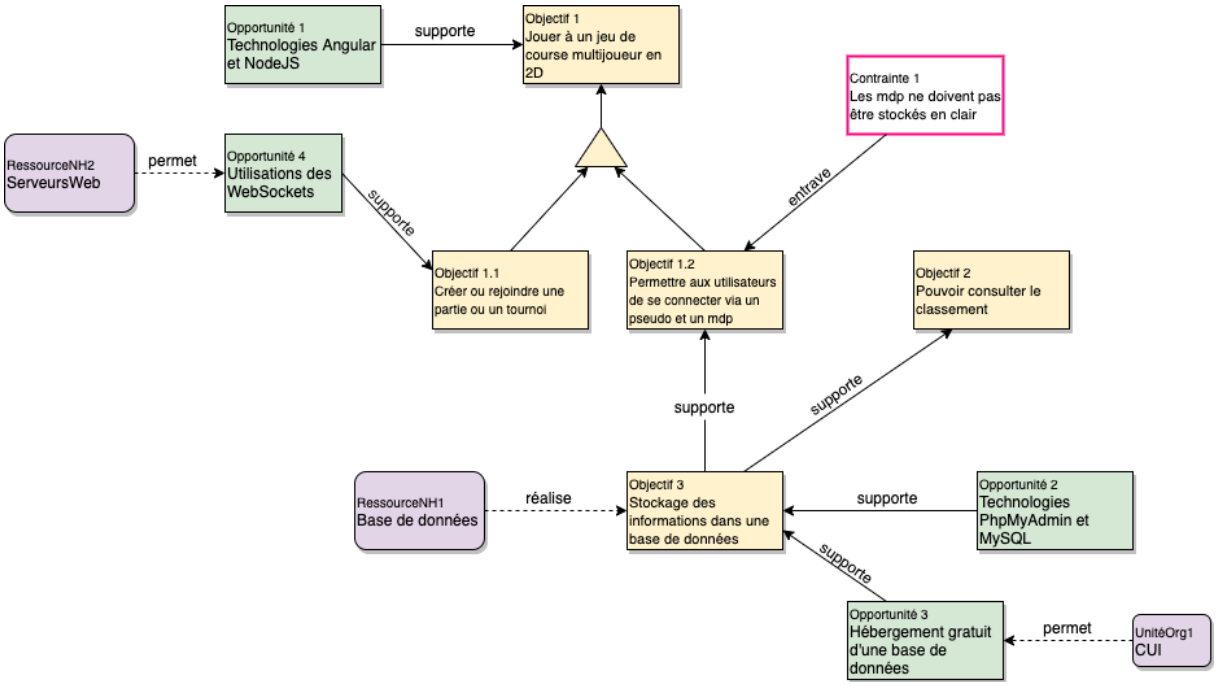
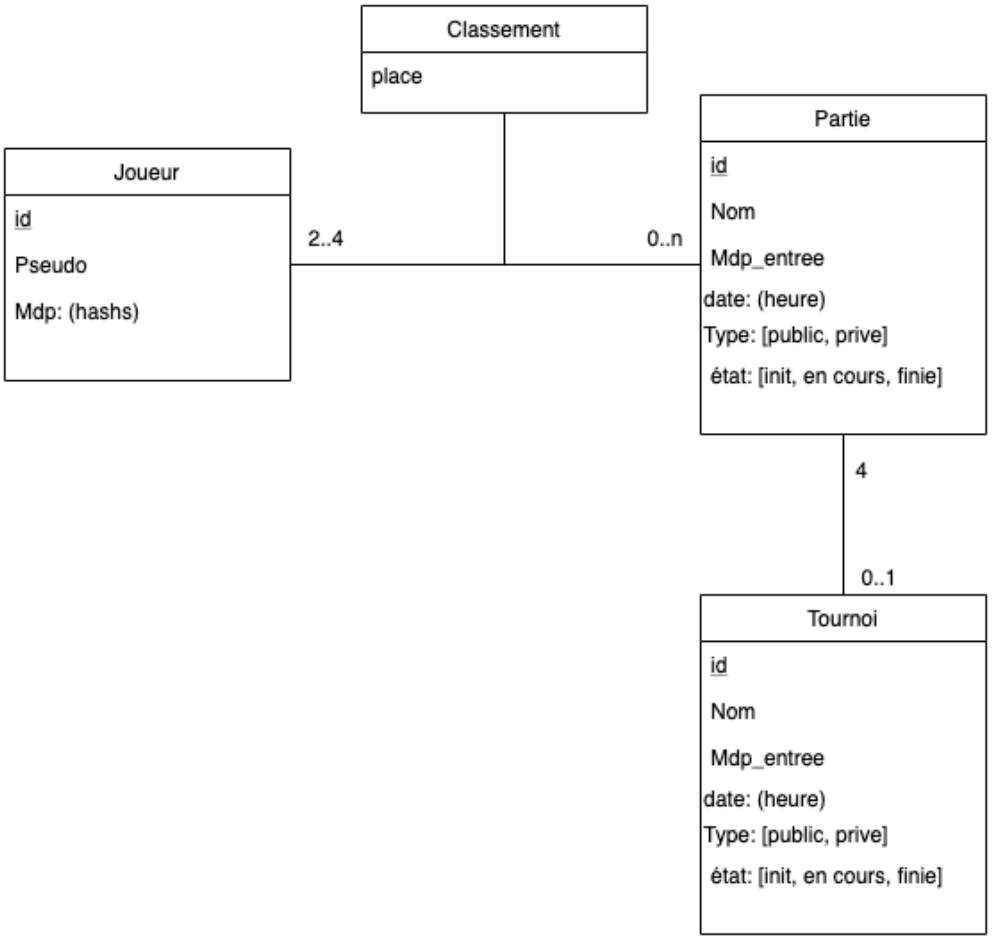
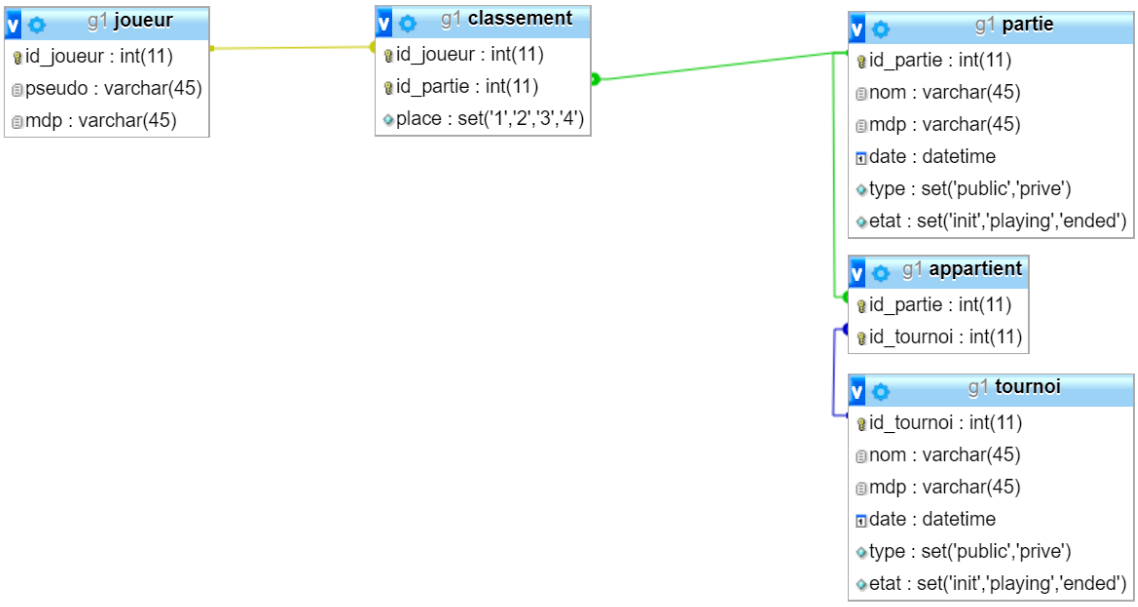
G1 – Juan Kerber & Stéphane Pacheco
Rapport intermédiaire Checkpoint 2

Analyse des objectifs

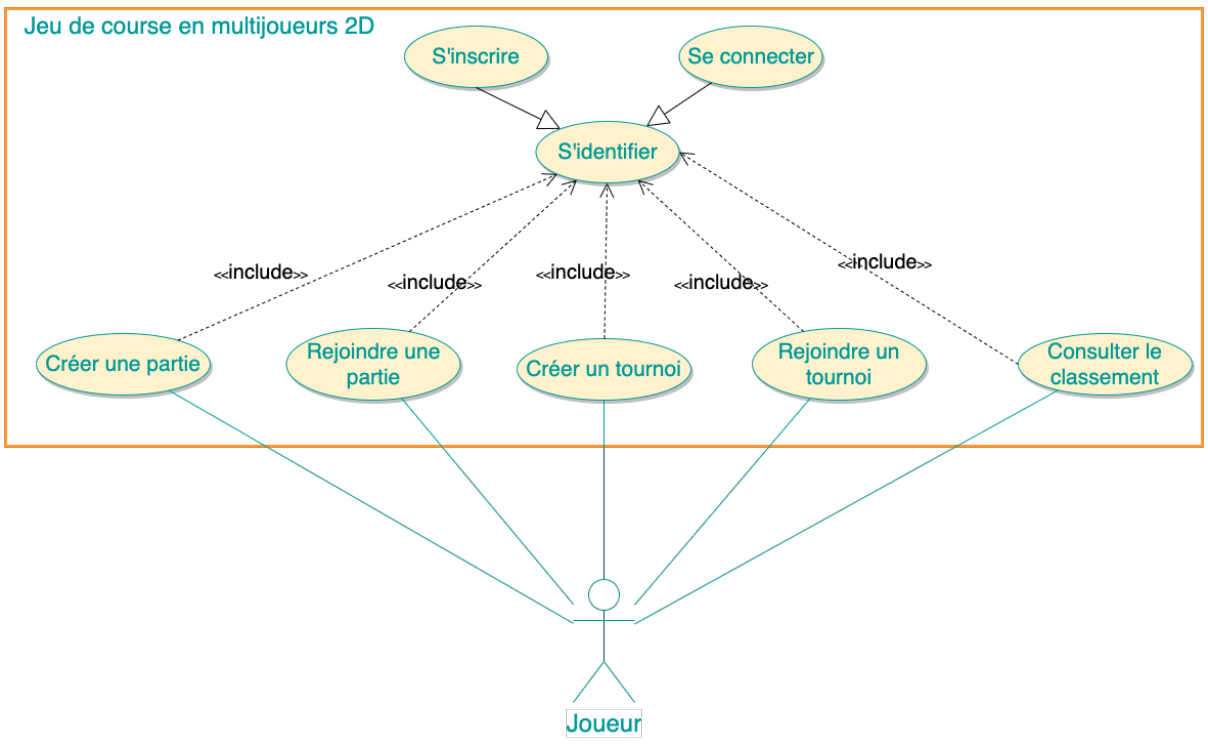


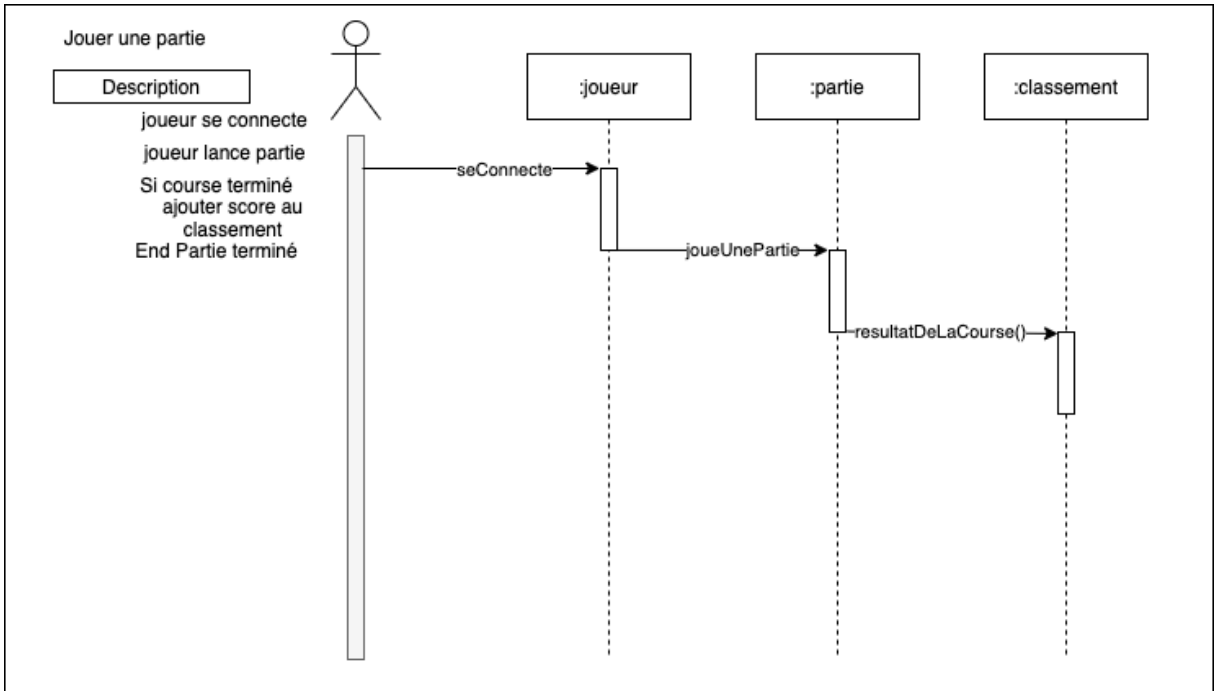
Modèles base de données

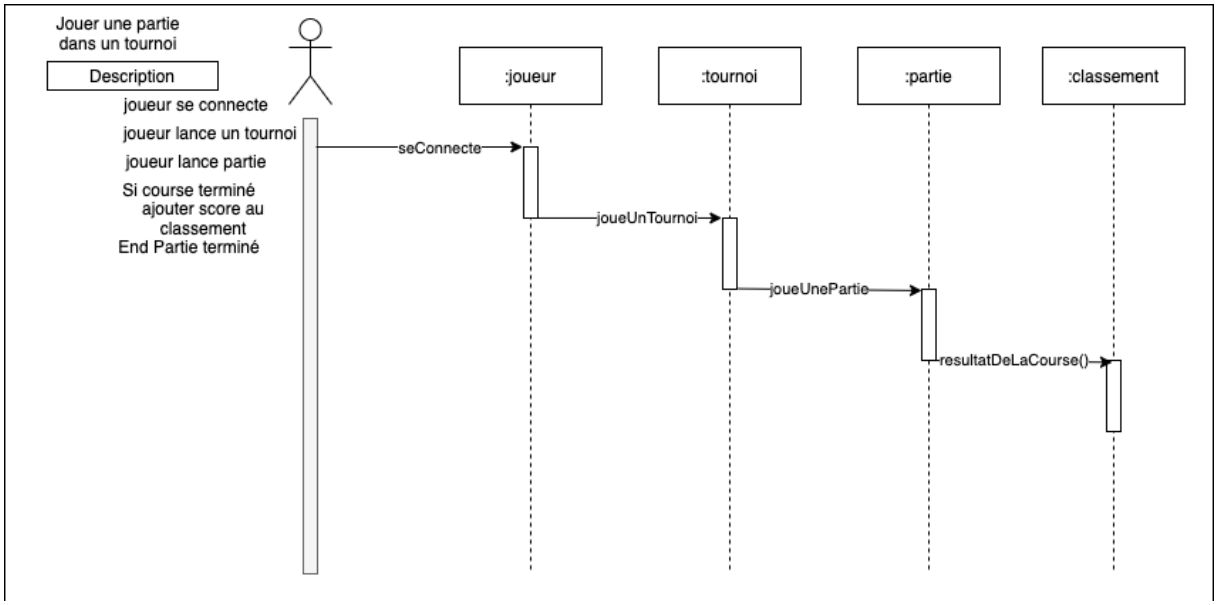


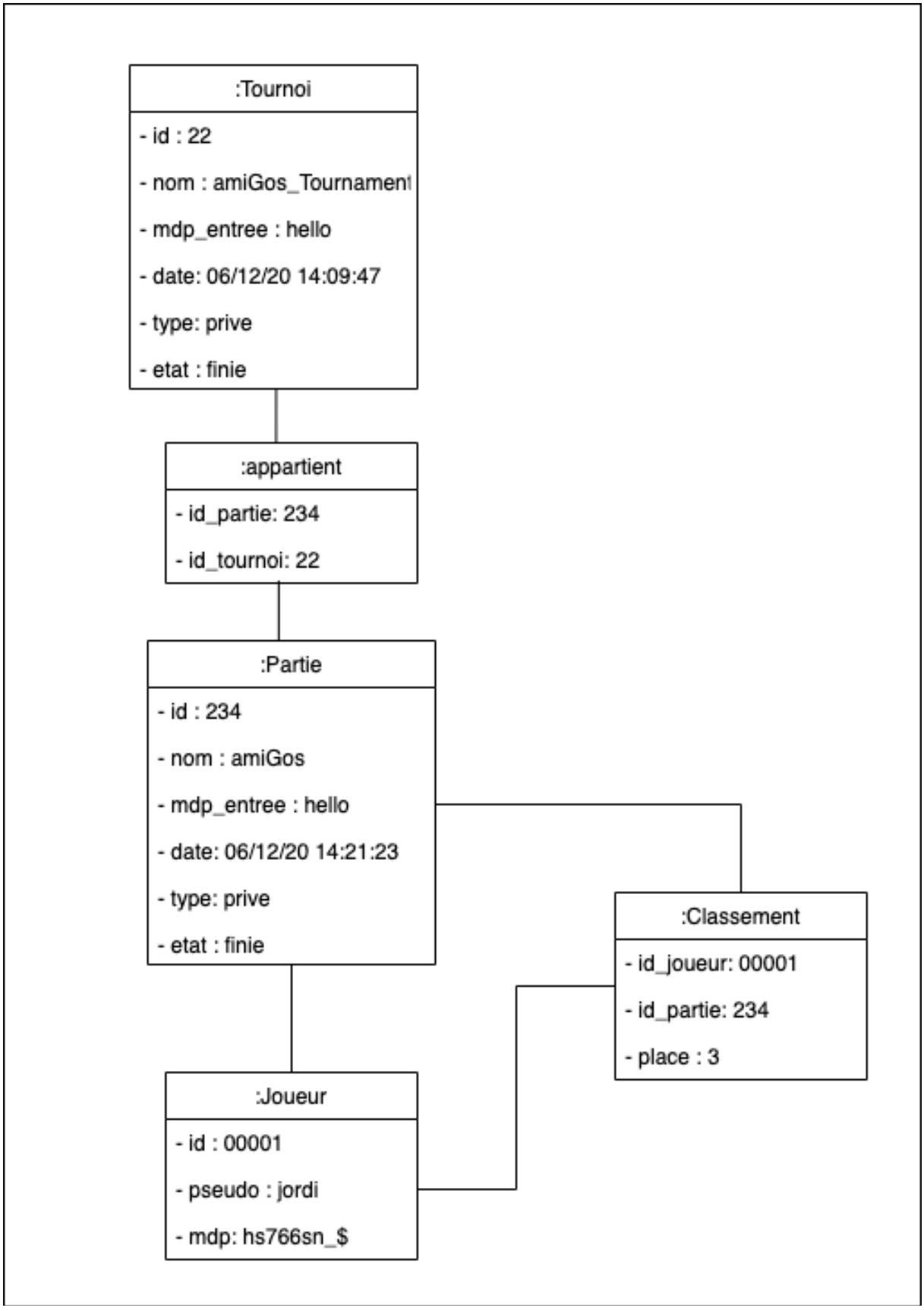


Modèles Système

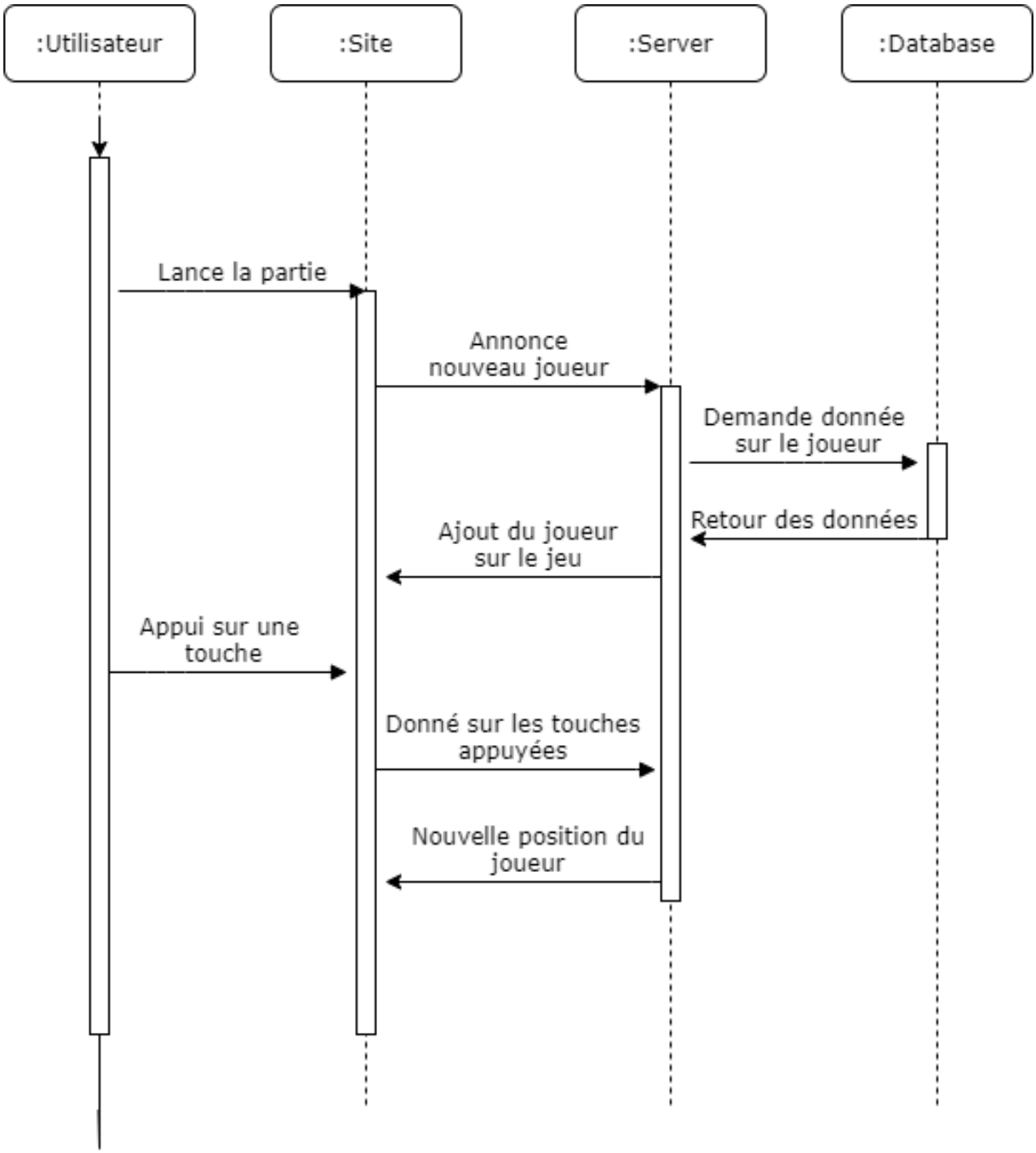



Jouer une partie
Description
joueur se connecte
joueur lance partie
Si course terminé
ajouter score au
classement
End Partie terminé
:joueur
:partie
:classement
seConnecte
joueUnePartie
resultatDeLaCourse()


Jouer une partie
dans un tournoi
Description
joueur se connecte
joueur lance un tournoi
joueur lance partie
Si course terminé
ajouter score au
classement
End Partie terminé
:joueur
:tournoi
:partie
:classement
seConnecte
joueUnTournoi
joueUnePartie
resultatDeLaCourse()


:Tournoi
- id : 22
- nom : amiGos_Tournament
- mdp_entree : hello
- date: 06/12/20 14:09:47
- type: prive
- etat : finie
:appartient
- id_partie: 234
- id_tournoi: 22
:Partie
- id : 234
- nom : amiGos
- mdp_entree : hello
- date: 06/12/20 14:21:23
- type: prive
- etat : finie
:Classement
- id_joueur: 00001
- id_partie: 234
- place : 3
:Joueur
- id : 00001
- pseudo : jordi
- mdp: hs766sn_$

Modèle Algorithme



Dès qu'un joueur se connecte, la connexion est donnée par socket à tous les autres clients qui affiche sur leur client le nouveau personnage. Dès qu'un mouvement est fait, de la même manière., les donnes sont diffusées à tous les autres clients pour qu'ils changent les différentes positions.

Implémentation client/serveur

L'entièreté du code est disponible sur le répertoire GitHub suivant :
https://github.com/JuanKerber/projet-transverse1

```
// Server config
const express = require('express');
const app = express();
var server = require('http').Server(app);
const io = require('socket.io').listen(server);
const path = require('path');

app.use('/js',express.static(__dirname + '/src/app/game/js'));
app.use('/assets',express.static(__dirname + '/src/assets'));


// Serve only the static files form the dist directory
app.use(express.static(__dirname + '/dist/demo'));

app.get('/*', function(req, res) {
    res.sendFile('index.html', {root: 'dist/demo/'}
  );
  });


// Player config
const players = {};
io.on('connection', function (socket) {
    console.log('a player connected: ', socket.id);

    // create a new player and add it to our players object
    players[socket.id] = {
        flipX: false,
        x: Math.floor(50),
        y: Math.floor(100),
        playerId: socket.id
    };

    // send all current players
    socket.emit('CURRENT_PLAYERS', players);

    // update all other players of the new player
    socket.broadcast.emit('NEW_PLAYER', players[socket.id]);

    socket.on('PLAYER_CONNECTED', function () {
        socket.broadcast.emit('PLAYER_CONNECTED', players[socket.id]);
        console.log('A player connected!')
    });

    // update movements of player
    socket.on('PLAYER_MOVED', function (movementData) {
        players[socket.id].x = movementData.x;
        players[socket.id].y = movementData.y;

        socket.broadcast.emit('PLAYER_MOVED', players[socket.id]);
    });

    // when a player disconnects, remove them from our players object
    socket.on('disconnect', function () {
        console.log('user disconnected: ', socket.id);
        delete players[socket.id];

        // emit a message to all players to remove this player
        io.emit('PLAYER_DISCONNECT', socket.id);
    });
});

// Start the server
const port = 8080;
server.listen(port,function(){
    console.log('Listening on '+server.address().port);
});
```

Démo préliminaire

Une vidéo de démonstration du projet à ce stade du développement est disponible sur le lien suivant :
https://www.dropbox.com/s/trk4kqito4g8pej/checkpoint2_videoJeu_JuanStephane.mov?dl=0