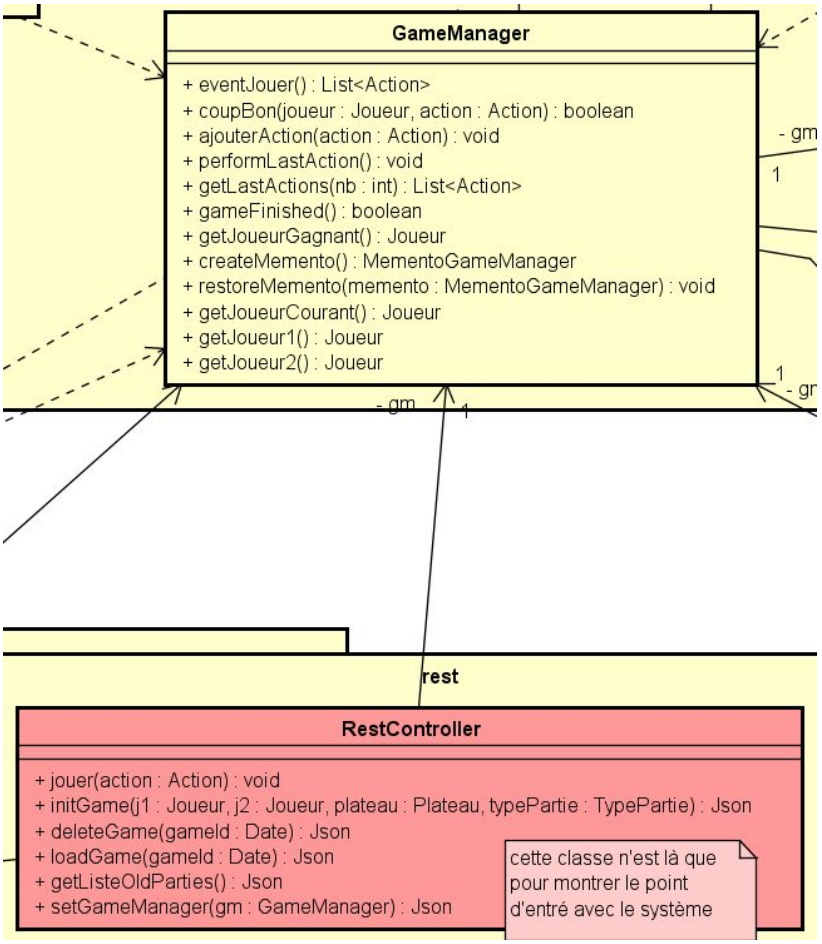
Tsiry Mayet
William Mocaër

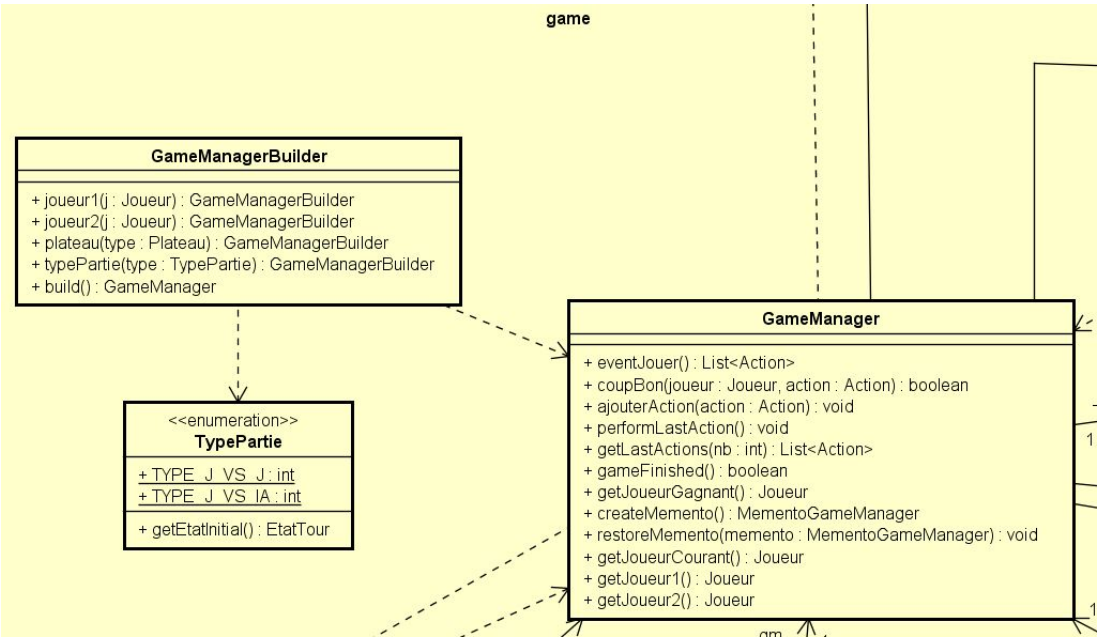
Rapport de modélisation et conception - MOO

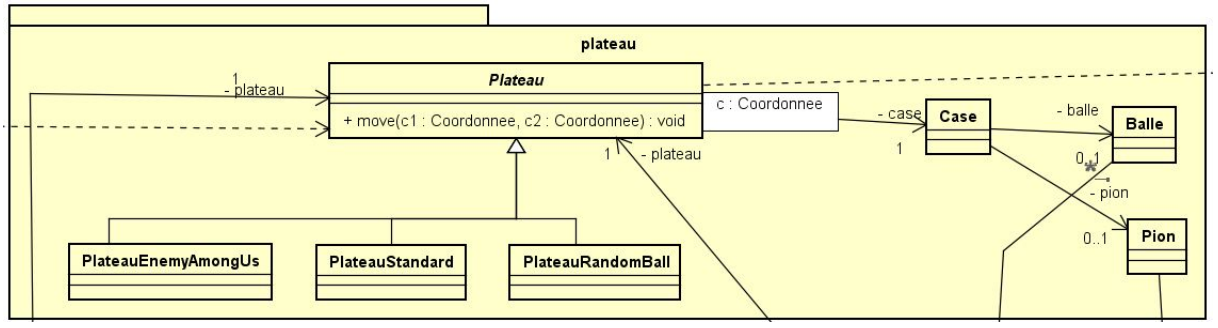
1.Modélisation

La classe RestController est le point d'entré de notre système côté back-end. Elle se chargera de faire l'interface entre le client, et le modèle sur le serveur.

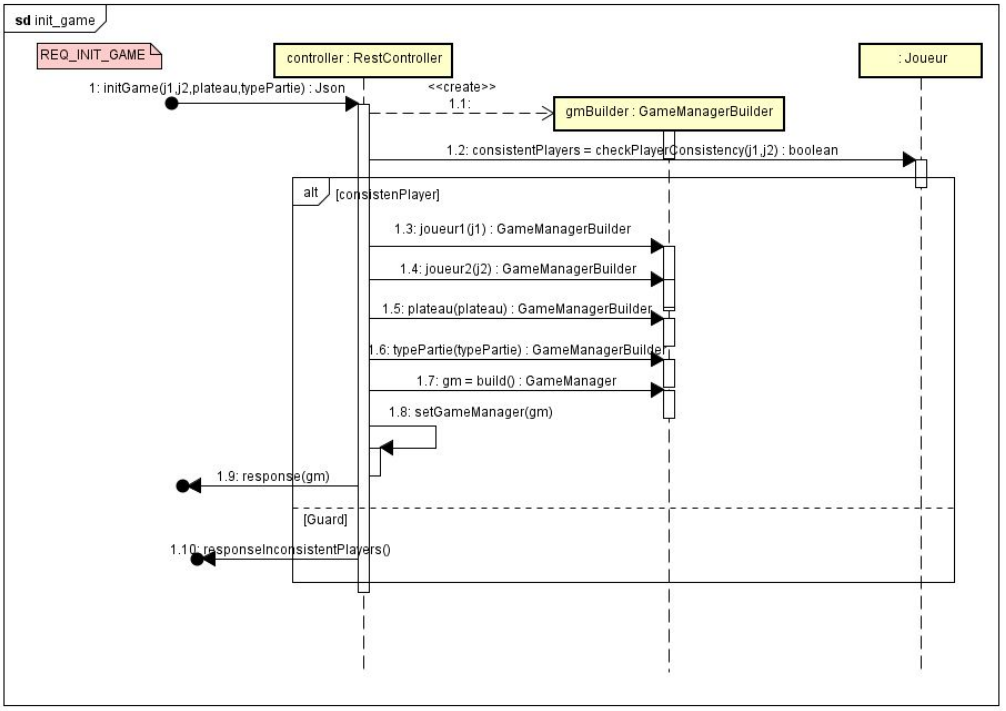


Initialisation

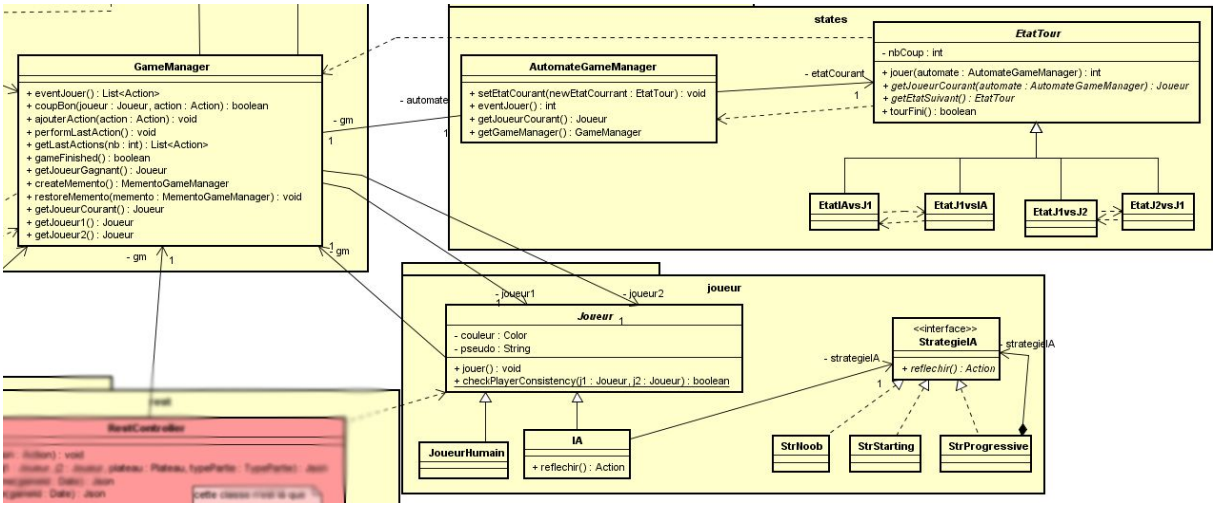




Le pion et la balle ont une référence vers le joueur qui le possède.

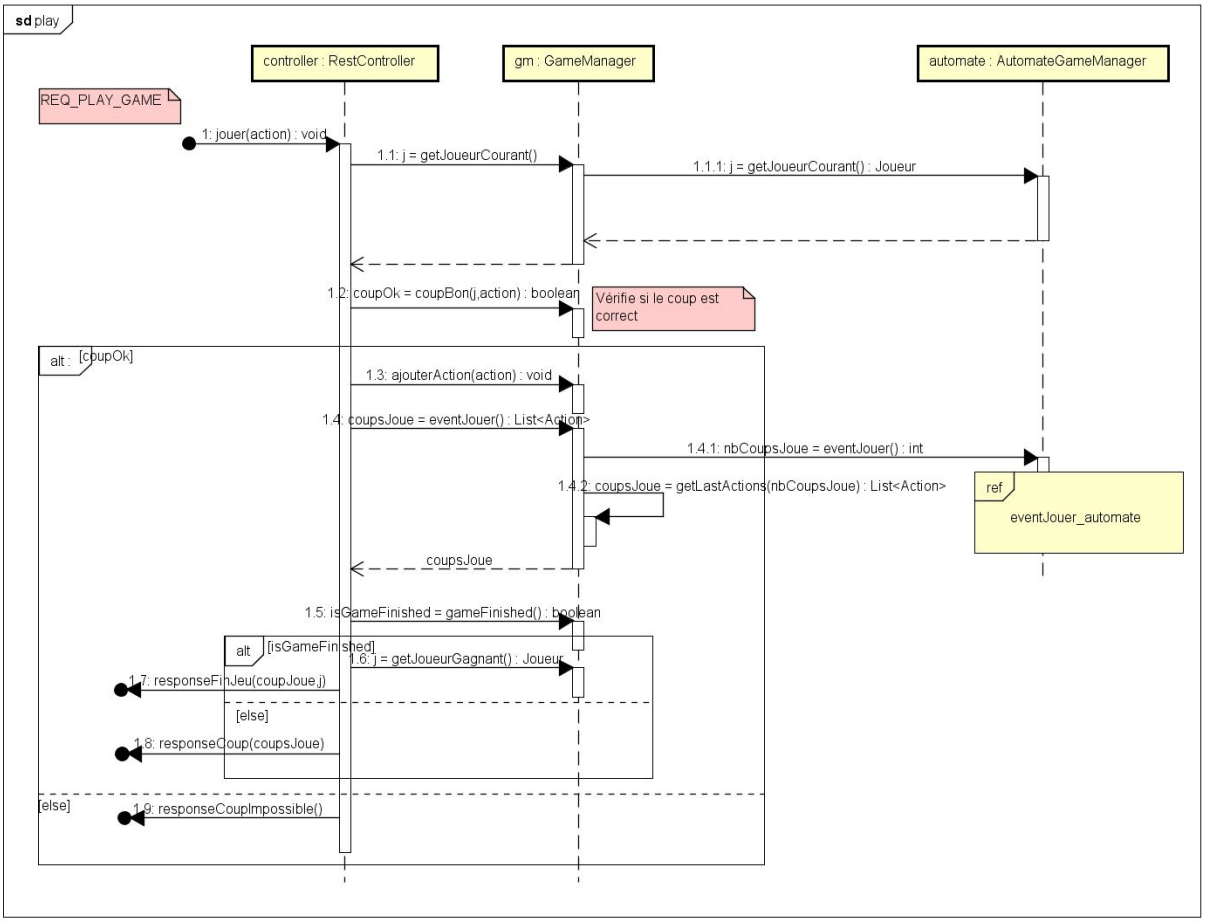


Jouer



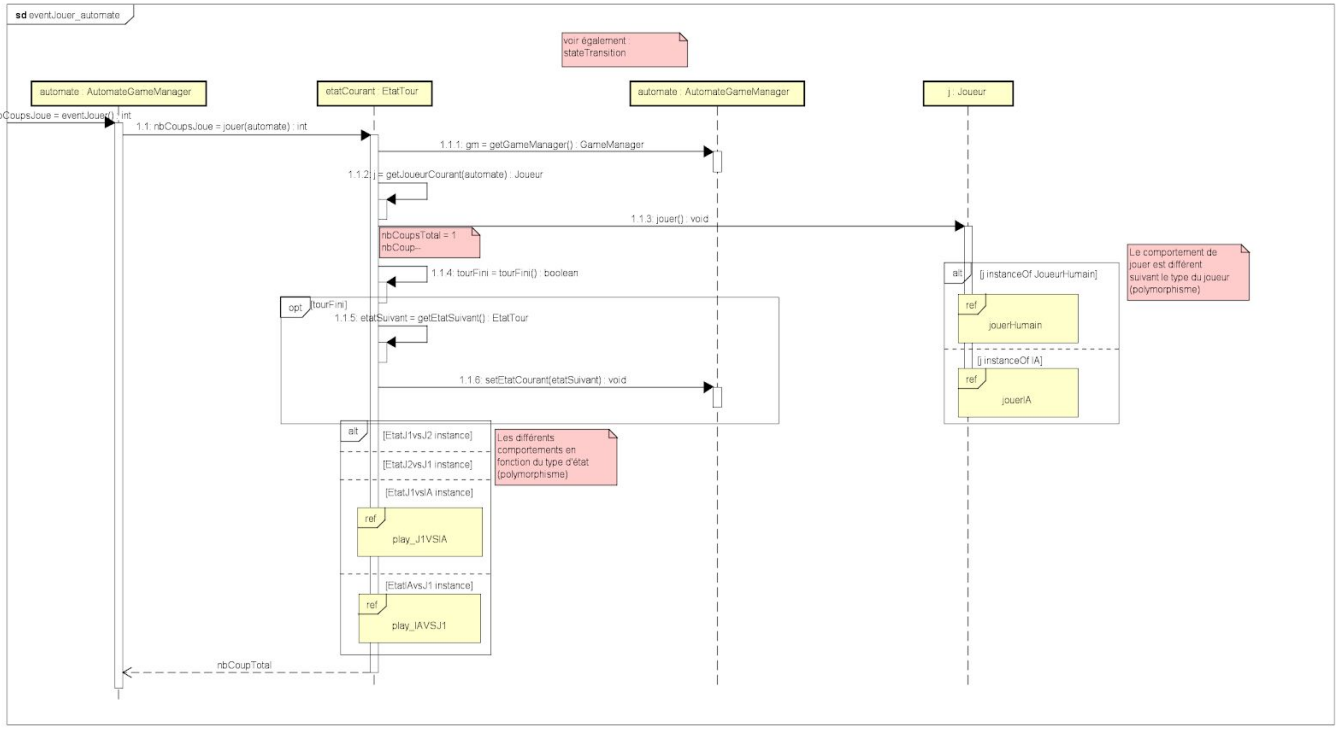
À la réception d'une requête pour jouer un coup.

sd_play



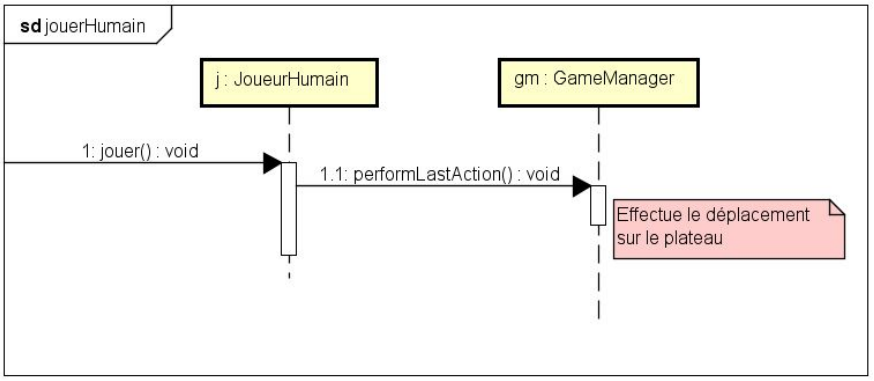
Suite « *sd_play* »

sd_eventJouer_automate



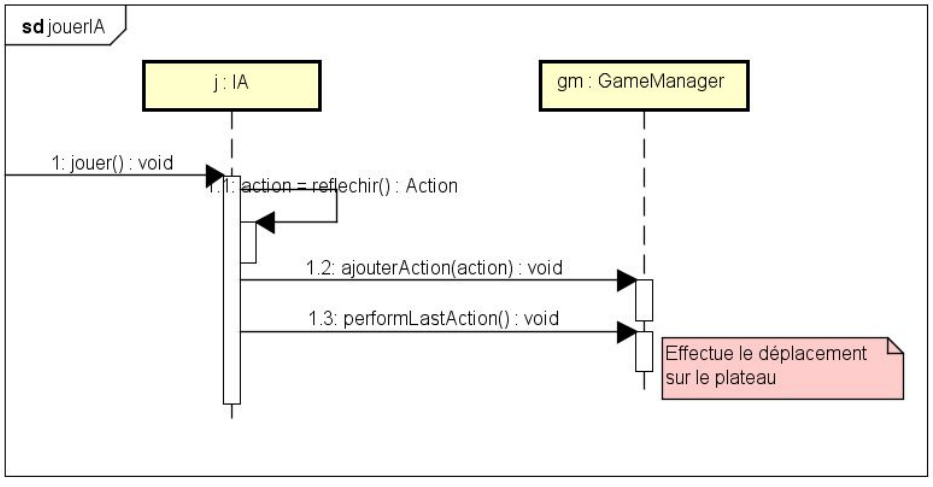
Suite « *sd_eventJouer_automate* », lorsque le joueur humain joue.

sd_jouerHumain



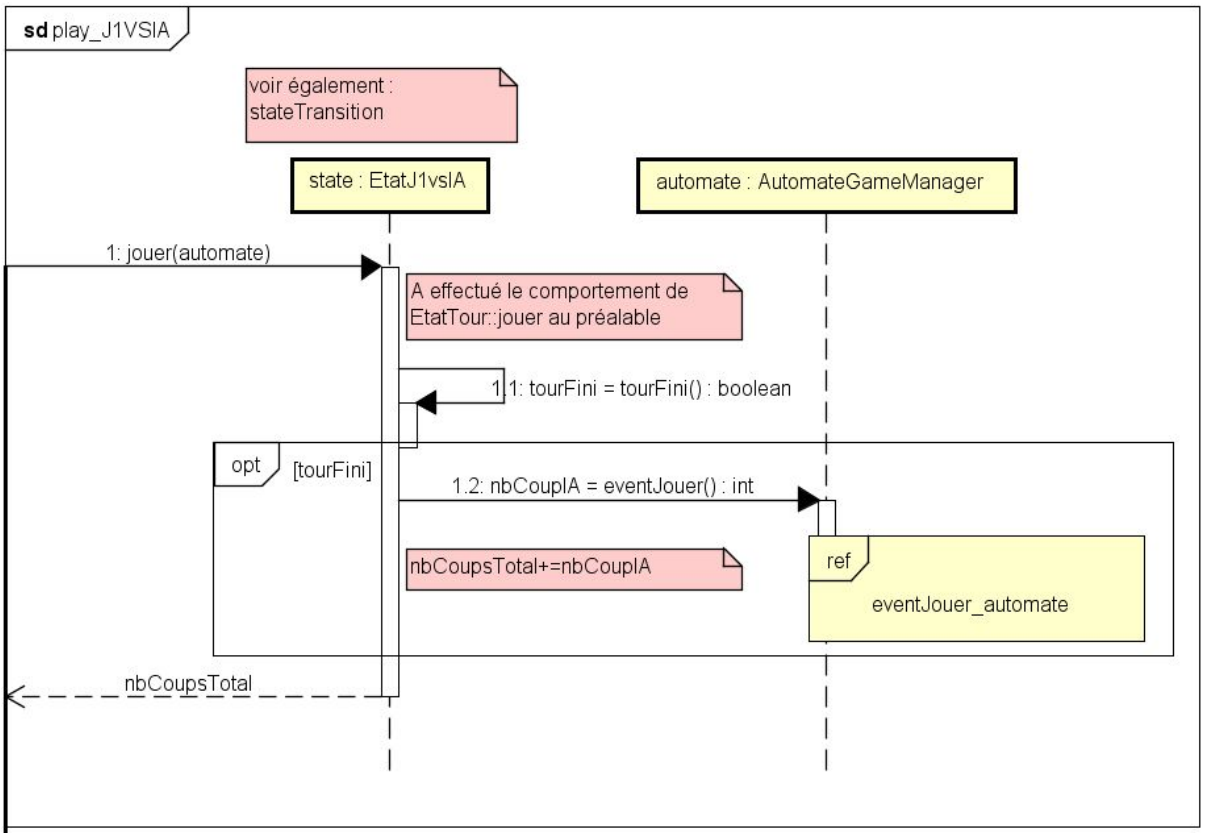
Suite « *sd_eventJouer_automate* », lorsque l'Intelligence Artificielle joue.

sd_jouerIA



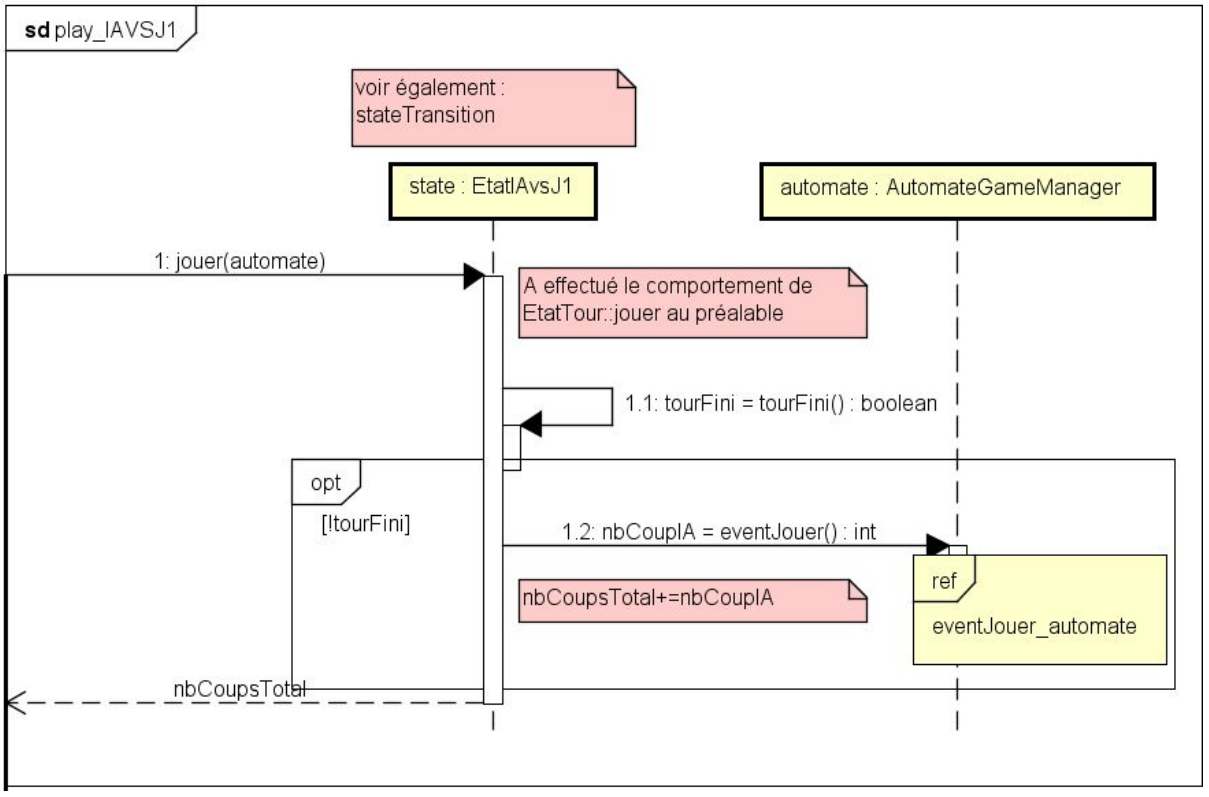
Suite « *sd_eventJouer_automate* », lorsque le joueur joue contre l'IA, c'est au tour du joueur.

sd_playJ1VSIA

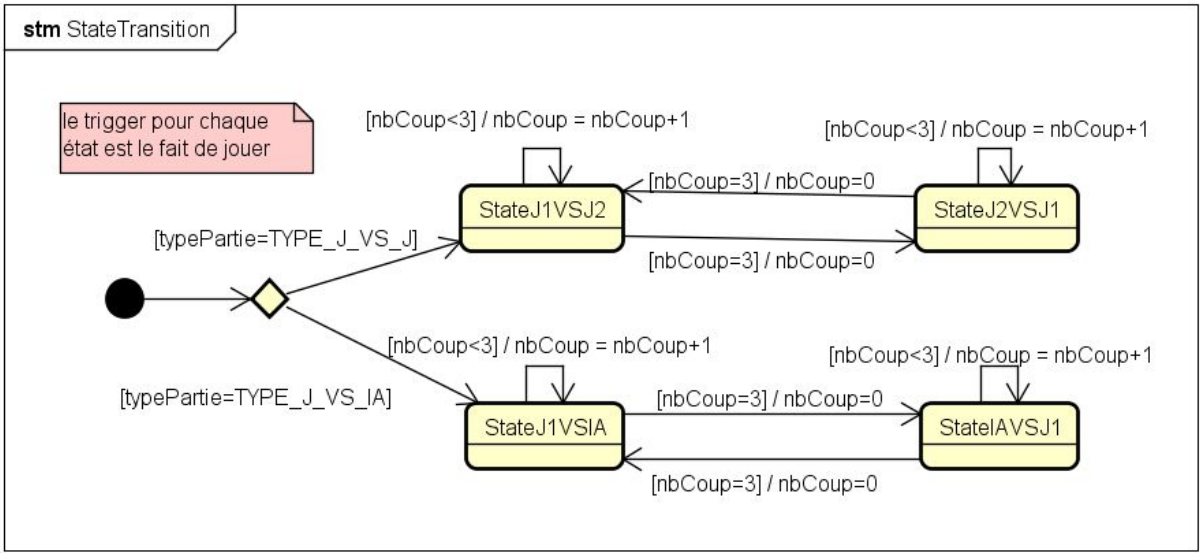


Suite « *sd_eventJouer_automate* », lorsque le joueur joue contre l'IA, c'est au tour de l'IA.

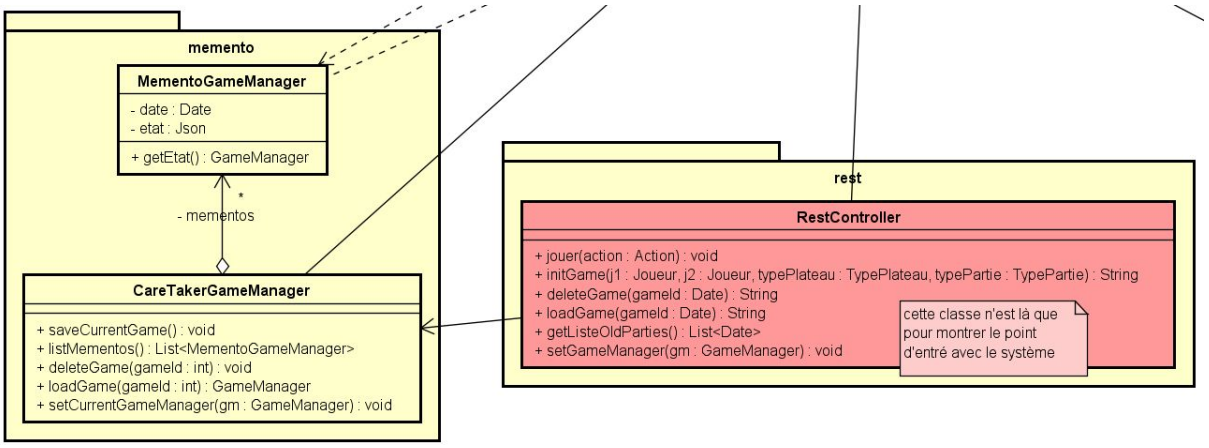
sd_playJ1VSIA



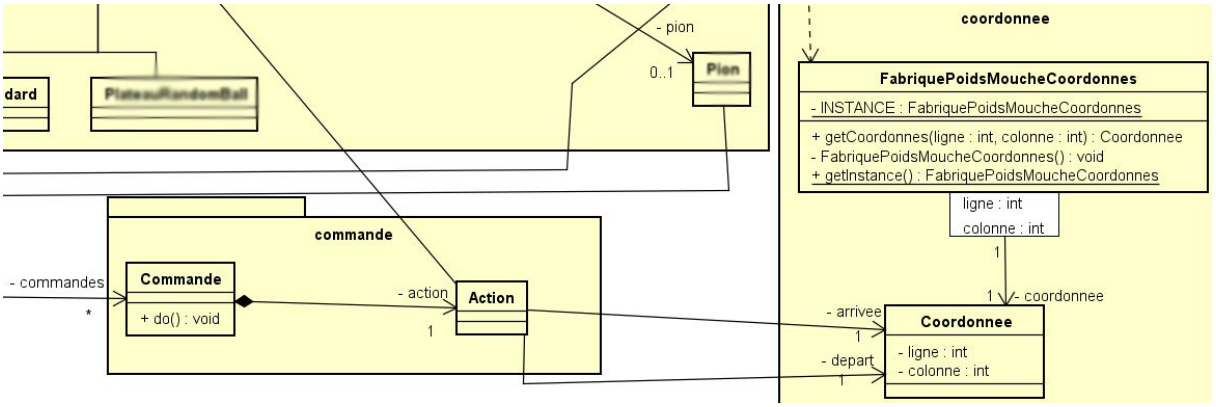
Récapitulatif des états et des transitions entre les états



Lister, sauvegarder, charger et supprimer les parties



Replay la partie



La partie sera envoyé au client, elle contiendra tous les coups effectués. Les coups sont stockées dans des commandes.

2.Requêtes REST

REQ_INIT_GAME :
Créer et initialiser une nouvelle partie.
PUT /init

```
{
   "joueur1" : {
     "pseudo" :"pseudoJ1",
     "couleur":{ "red": 45, "green" : 78, "blue" : 150},
     "type" : "JoueurHumain"
   },
   "joueur2" : {
     "pseudo" :"pseudoJ2",
     "couleur":{ "red": 245, "green" : 78, "blue" : 10},
     "type" : "IA",
     "strategieIA" : { "type" : "StrNoob" }
   },
   "plateau" : {"type" : "PlateauStandard"},
   "type_partie"  : 1
}
```

REQ_PLAY_GAME :
Joue un coup.
PUT /play

```
{
 "action" : {
            "depart" : { "ligne" : 0, "colonne":1},
            "arrive" : { "ligne" : 1, "colonne":1},
        }
}
```

Réponse du serveur dans le cas où on joue contre une IA et que nous venons de jouer le dernier coup de notre tour. On renvoie le coup du joueur ainsi que les trois coups joués par l'IA :

```
{
   "actions" : [
       {
           "depart" : { "ligne" : 0, "colonne":1},      //le coup que le joueur a joué pour lui
           "arrive" : { "ligne" : 1, "colonne":1},      //dire que le coup est bon
        },
        {
           "depart" : { "ligne" : 8, "colonne":1},      //suivit des 3 coups de l'IA
           "arrive" : { "ligne" : 7, "colonne":1},
        },
        { "depart" : { "ligne" : 8, "colonne":2}, "arrive" : { "ligne" : 7, "colonne":2}},
        { "depart" : { "ligne" : 8, "colonne":3},  "arrive" : { "ligne" : 7, "colonne":3}}
         ],
   "gamewinned":false,
   "winningplayer":null
}
```

REQ_DELETE_GAME
Supprimer une partie enregistrée.
DELETE /game/{idGame}

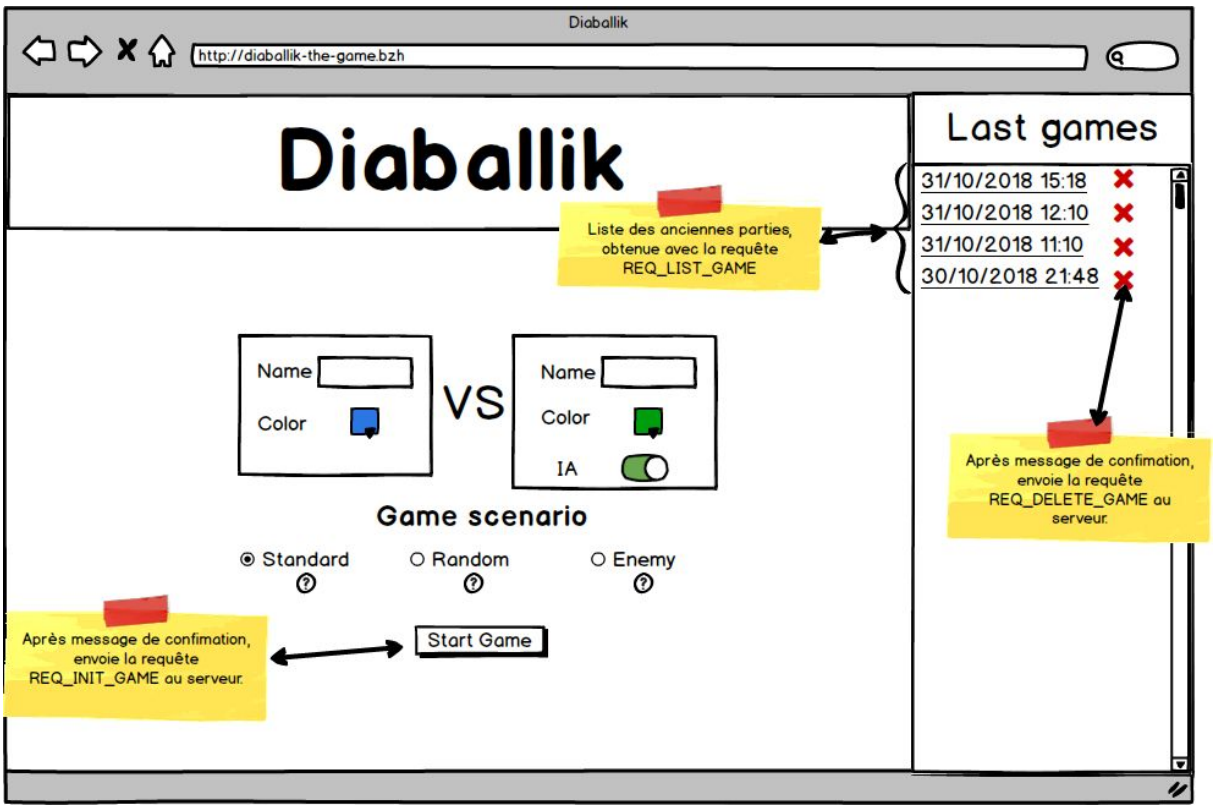
REQ_LIST_GAME
Lister les anciennes parties.
GET /game/games
Exemple de retour du serveur.

```
{
     “games” :
     [ {
          “joueur1”:{..},
          “joueur2”:{..},
           “type_partie”:”0”,   //contre l’ia ou contre l’humain
           “date”: 2018-12-08
      }...]
}
```

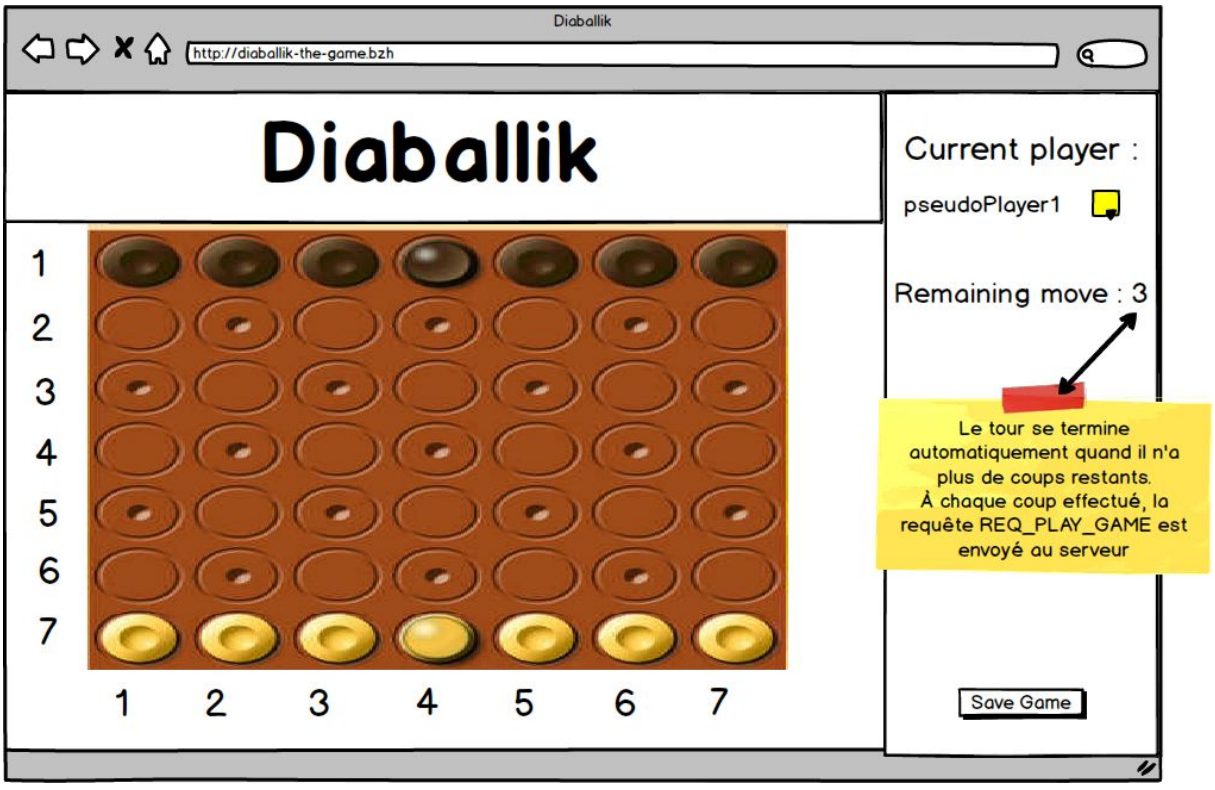
REQ_GET_GAME
Récupère une partie sauvegardée, pour soit la jouer, soit la regarder en replay.
GET /game/{idGame}
Le serveur renvoie le gameManager.

3.Mockups

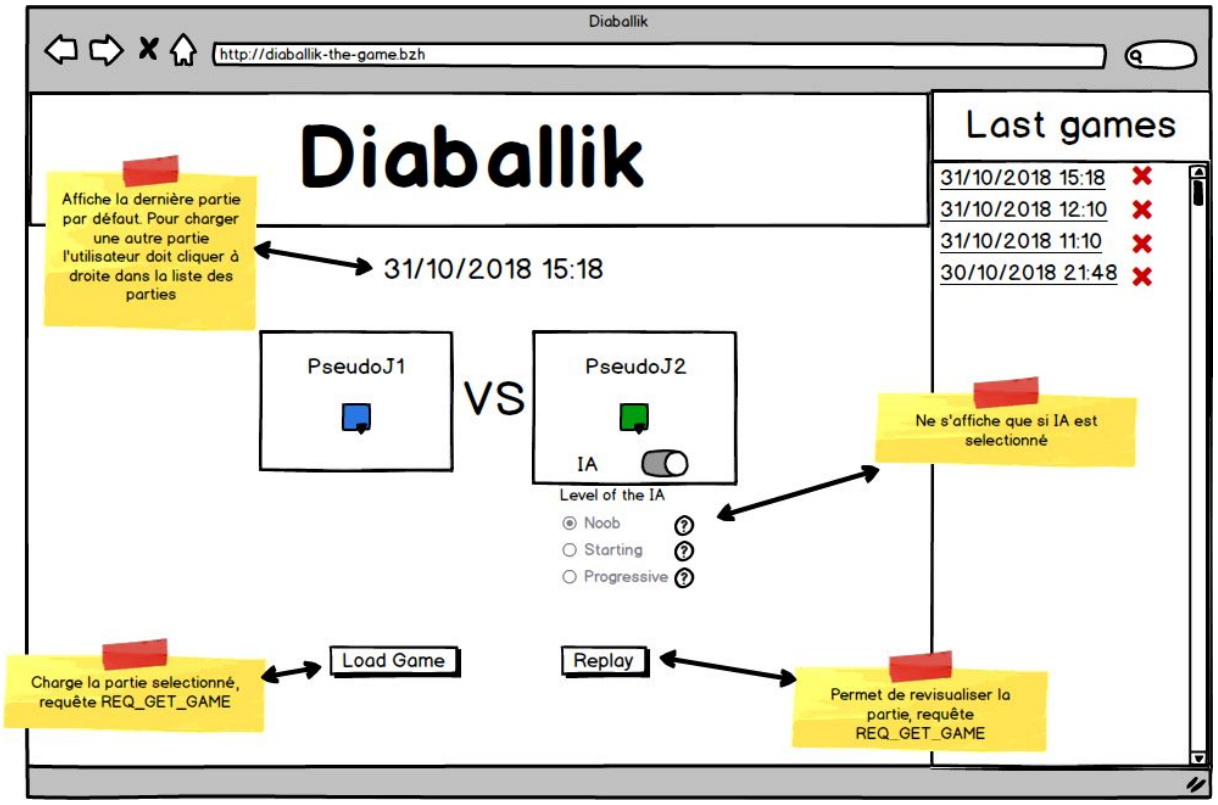
Accueil



Start game



Charger une partie



Replay d'une partie