Équipe : Le Conseil des Sept
Joël Collinet
Félix Erb
Célia Kocher
Vincent Moitry
Adam Rimelé
Carlo Spiga
Maxime Zimmer

Protocole de communication des coups

Projet Acrobatt 2014

La Clémence d'Auguste

Tuteur : Pierre Kraemer

SOMMAIRE

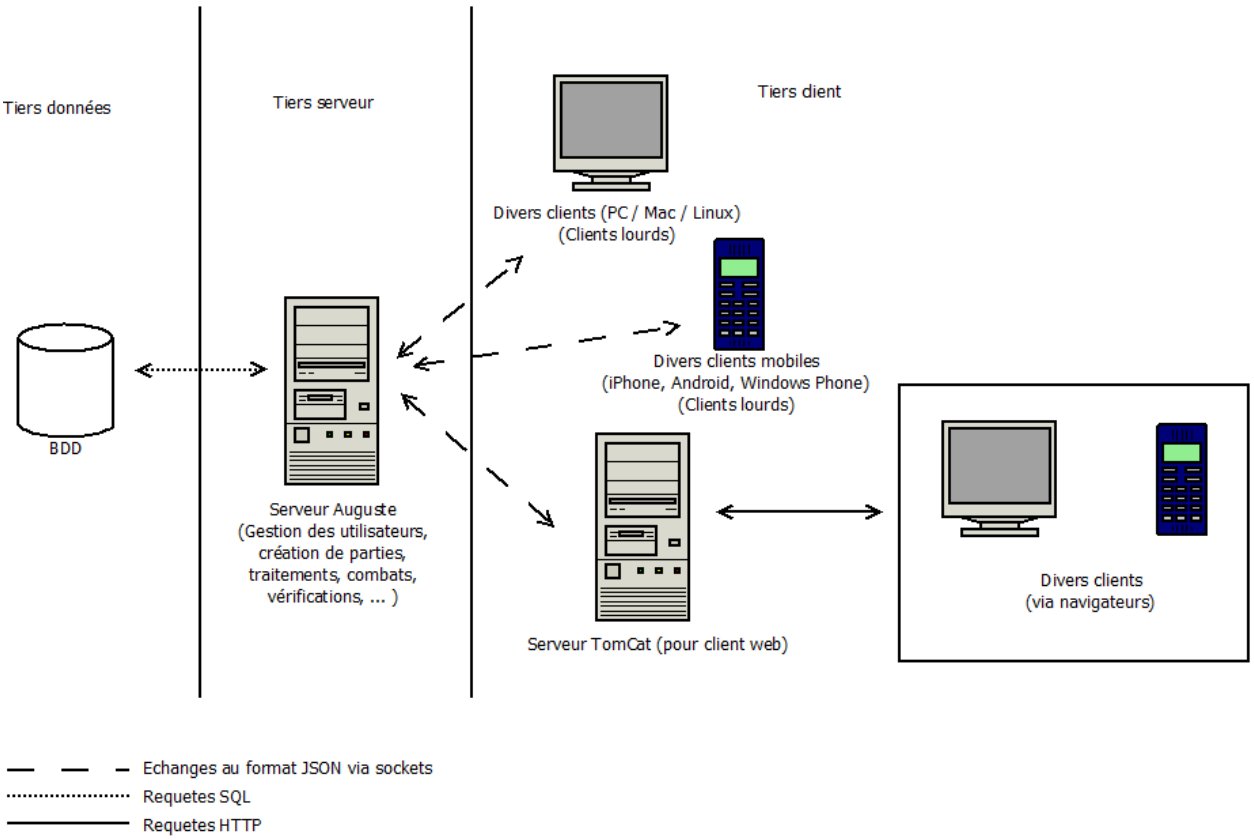
INTRODUCTION

Dans le cadre du Projet Acrobatt 2014 aussi nommé “La Clémence d’Auguste”, il a été demandé de réaliser un protocole de communication des coups.

Ce protocole a pour objectif de schématiser et d’expliquer le processus de communication client-serveur proposé par notre équipe. Ce processus doit répondre efficacement à la problématique posée par le projet (comment utiliser l’application en multijoueur, quel que soit le support (PC, smartphone, tablette, web, …).

De plus, le protocole mis en place sera adopté par l’ensemble des groupes de projet, ce qui permet d'interchanger serveurs et clients des différents groupes sans problème.

EXPLICATIONS



L'architecture proposée est une architecture trois tiers (données, serveur, client).

Le tiers serveur comprend un simple serveur (nommé serveur Auguste ici). Il reçoit les données envoyées par les clients lourds ou par le serveur web TomCat (ou GWT, encore à l'étude). Il réalise ensuite les différentes opérations nécessaires au bon déroulement du jeu (application des changements).

Le tiers données se compose d'une base de données, chargée de recevoir les requêtes SQL envoyées par le serveur Auguste. Le stockage de certaines données

(comptes des joueurs, parties, …) est nécessaire pour assurer leur persistance de manière efficace.

Le tiers client est composé des machines et logiciels utilisés par les utilisateurs finaux (ordinateur, tablette, smartphone, navigateur web). Chaque client dispose d'un identifiant. Il est possible de jouer de deux manières : installer un client lourd / mobile sur son terminal (application pour PC et smartphone), ou utiliser un navigateur web.

Pour cette dernière option, un serveur TomCat est mis en place côté client. Celui-ci se charge d'envoyer les données des navigateurs au serveur Auguste qui les traitera ensuite. Cet envoi se fait avec l'aide d'une unique socket comprenant ces données.

La liaison entre navigateurs et serveur TomCat s'effectue avec l'aide de websockets. Les liaisons établies entre serveur Auguste et clients lourds (ou serveur TomCat) sont permises par l'action des sockets.

La mise en place de ce protocole est nécessaire. Celui qui sera choisi devra être respecté par l'ensemble des groupes Acrobatt.

CHOIX DU FORMAT DE SÉRIALISATION

Il faut définir un mode d'échange entre client et serveur. Les données sont transportées à l'aide de sockets. Ces dernières sont utilisées pour envoyer des chaînes de caractères, mais on passe par une sérialisation en objet JSON pour structurer ces données.

Le JSON a été choisi, car il assure une compatibilité plus simple avec les différents langages de programmation.

D'autres formats ont été envisagés comme par exemple le XML, mais son parcours s'avère moins facile et sa structure arborescente n'est pas la plus adaptée dans le cadre de nos besoins.

RÈGLES DE CRÉATION D'UN JSON

En ce qui concerne les règles de création d'un JSON, nous nous sommes accordés sur quelques principes simples que nous respecterons afin de les rendre plus lisibles et plus clairs.

Les variables seront en anglais, en minuscules. Les différents mots seront séparés par des "_" . Chaque JSON comprend une variable "command" qui permet d'identifier l'action à réaliser. Si l'action est réalisée par le client, son identifiant de session est ajouté dans une variable "session".

Exemple : le déplacement du laurier

Déplacement du laurier

Attributs :

command : la commande décrite, ici "turn_laurel"

session : identifiant de la session du joueur

destination : case de destination du laurier

Exemple :

```
{
    "command": "turn_laurel",
    "session": "dfs65d7s984vsd213v3ds0sdvs"
}
```

LISTE DES JSON

COMMANDES CLIENT

chat - Chat (coté client)

Attributs :

command : la commande décrite, ici "chat"

session : identifiant de la session du joueur

message : message envoyé

Exemple :

```
{
	"command": "chat",
	"session": "fvb3v2nh498gh7k9g8h7sd",
	"message": "Bonjour ;)"
}
```

---

game_addbot - Ajouter un bot

Attributs :

command : la commande décrite, ici "game_addbot"

session : identifiant de la session du joueur

Exemple :

```
{
	"command": "game_addbot",
	"session": "trut4y65n41n3v24bn64654"
}
```

game_create - Création d'une partie

Attributs :

command : la commande décrite, ici "create_game"

session : identifiant de la session du joueur

game_name : nom de la partie tel que vue par les autres joueurs

Exemple :

```
{
    "command": "game_create",
    "session": "dfb4d6f5b4d65f4bd654fb65d4",
    "game_name": "Partie de Maxime"
}
```

---

game_config - Paramétrer une partie avant de la lancer

Attributs :

command : la commande décrite, ici "game_config"

session : identifiant de la session du joueur

game_name : nom de la partie tel que vue par les autres joueurs

turn_timer : durée maximum d'un tour en secondes

board_size : taille du plateau

cards : booléen indiquant si les cartes sont autorisées ou non

teams : nombre d'équipe

players : nombre de joueur

legions_max : nombre légion maximum (utile pour lister les parties disponible)

players : Configurer les équipes, légions…

player_id : l'identifiant du joueur (et de ses légions)

legions : le nombre de groupe d'unité controlé par le joueur

team : l'équipe du joueur

Exemple :


```
{
     "command": "game_config",
     "session": "dfb4d6f5b4d65f4bd654fb65d4",
     "game_name": "Partie de Maxime",
     "turn_timer": 120,
     "board_size": 8,
     "cards": false,
     "teams": 2,
     "players": 2,
     "legions_max": 4,
     "players":
     [
          {
               "player_id": 1,
               "legions": 1,
               "team": 1
          },
          {
               "player_id": 2,
               "legions": 3,
               "team": 2
          }
     ]
```

```
}
```

---

game_join - Rejoindre une partie (lancé depuis le client)

Attributs :

command : la commande décrite, ici "game_join"

session : identifiant de la session du joueur

game_id : identifiant de la partie à rejoindre

Exemple :

```
{
    "command": "game_join",
    "session": "sdv6s54v65sd4v6s5d4vs654",
    "game_id": 47
}
```

---

game_kick - Kick d'un joueur

Attributs :

command : la commande décrite, ici "game_kick"

session : identifiant de la session du joueur

player_id : le numéro du joueur qu'on veux kicker

Exemple :

```
{
    "command": "game_kick",
    "session": "sezgerjty654g65h7kg65h4",
```

```
    "player_id" : 2
}
```

---

game_leave - Déconnexion d'un joueur

Attributs :

command : la commande décrite, ici "game_leave"

session : identifiant de la session du joueur

Exemple :

```
{
    "command": "game_leave",
    "session": "sdg87b9f1d651324sdgz"
}
```

---

game_list - Lister les parties en attente de joueurs.

Attributs :

command : la commande décrite, ici "game_list"

session : identifiant de la session du joueur

Exemple :

```
{
    "command": "game_list",
    "session": "sfb6c7jty84n64f6876"
}
```

game_start - Lancement de la partie

Attributs :

command : la commande décrite, ici "game_start"

session : identifiant de la session du joueur

Exemple :

```
{
    "command": "game_start",
    "session": "dhs6g5n4s6f5gn7s987"
}
```

---

log_in - Identification

Attributs :

command : la commande décrite, ici "log_in"

login : le login du joueur

password : le mot de passe du joueur (peut être haché)

Exemple :

```
{
    "command": "log_in",
    "login": "Joel",
    "password": "Ch0c0l4t"
}
```

log_out - Déconnexion

Attributs :

command : la commande décrite, ici "log_out"

session : identifiant de la session du joueur

Exemple :

```
{
	"command": "log_out",
	"session": "xdghq6srg76sqf4b6sf5g654"
}
```

---

turn_leave - Abandon d'un joueur

Attributs :

command : la commande décrite, ici "turn_leave"

session : identifiant de la session du joueur

Exemple :

```
{
	"command": "turn_leave",
	"session": "47dg6n1fg0fg6nf01sxw65"
}
```

---

turn_move - Déplacement d'une légion

Attributs :

command : la commande décrite, ici "turn_move"

session : identifiant de la session du joueur

source : case de départ de la légion (ou LAU -> laurier) [u,w]

destination : case de destination de la légion

cards : liste des cartes jouées

card_name: nom de la carte

info suplémentaire si besoin

Exemple :

```
{
     "command": "turn_move",
     "session": "sdbdfbvn13vb246h5g4g",
     "source":
     [
          4,
          1
     ],
    "destination":
     [
          2,
          7
     ],
     "cards":
     [
          {
               "card_name":"Cavalier"
          },
          {
               "card_name":"Muraille",
```

```
                "wall1":
                [
                    5,
                    2
                ],
                "wall2":
                [
                    8,
                    10
                ]
            },
            ...
        ]
}
```

COMMANDES SERVEUR

game_list - Liste des parties disponibles

Attributs :

command : la commande décrite, ici "game_list"

games : liste des parties

game_id: le numéro d'identification de la partie

game_name: le nom de la partie

players: le nombre de joueur ayant actuellement rejoins la partie

players_max: le nombre requis de joueur pour commencer la partie

Exemple :

```
{
	"command": "game_list",
	"games":
	[
		{
			"game_id": 2,
			"game_name": "Partie de Joelle",
			"players":1,
			"players_max":2
		},
		...
	]
}
```

game_turn - Lancement d'un nouveau tour

Attributs :

command : la commande décrite, ici "game_turn"

board : le plateau (emplacement de chaque pièce sur le terrain)

moves : la liste des déplacement à effectuer

cards: la liste des cartes jouées

cards_played:la liste des cartes jouées se tour

Exemple :

```
{
    "command": "game_turn",
    "board":
    [
        {
            "u": 11,
            "w": 54,
            "armor": false,
            "laurel": false,
            "tent": false,
            "team": 2,
            "player": 2,
            "death": "none"|"battle"|"surrounding"
        },
        ...
    ],
    "moves":
```

```
[
     {
          "source":
          [
               21,
               45
          ],
          "destination":
          [
               68,
               69
          ]
     },
],
"cards":
[
     {
          "player_id": 1,
          "cards":
          [
               "devin",
               "volcan"
          ]
     },
     ...
],
"cards_played":
```

```
    [
        {
            "player_id": 2,
            "cards":
            [
                [
                    "volcan"
                ]
            ]
        },
        ...
    ]
}
```

---

log_confirm - Identification réussie

Attributs :

command : la commande décrite, ici "log_confirm"

session : identifiant de la session du joueur

Exemple :

```
{
    "command": "log_confirm",
    "session": "srg5e6r54b65f4bd6546"
}
```

log_error - Réponse de l'identification

Attributs :

command : la commande décrite, ici "log_error"

error : code de l'erreur (par ex. 4 = login inconnu)

Exemple :

```
{
	"command": "log_error",
	"error": 4
}
```

---

player_chat - Chat (côté serveur)

Attributs :

command : la commande décrite, ici "player_chat"

author : numéro du joueur l'ayant envoyé (0 à X)

time : temps (timestamp) de réception du message

message : message envoyé

Exemple :

```
{
	"command": "player_chat",
	"author": 2,
	"time" : 6846516651321,
	"message": "Bonjour ;)"
}
```

player_leave - Déconnexion d'un joueur

Attributs :

command : la commande décrite, ici "player_leave"

player : numéro du joueur déconnecté (0 à X)

disconnect : déconnecté ou abandon

Exemple :

```
{
    "command": "player_leave",
    "player": 2,
    "disconnect" : false
}
```

---

player_join - Un joueur à rejoin une partie (avant le lancement pendant la configuration)

Attributs :

command : la commande décrite, ici "player_join"

session : session : identifiant de la session du joueur

player_name : nom du joueur

Exemple :

```
{
    "command": "player_leave",
    "session": srg5e6r54b65f4bd6546,
    "player_name" : "George"
}
```

ANNEXES

STRUCTURES JSON

Données échangées SERVEUR -> CLIENT

Données d'un joueur :

```
{
     "name": (string),
     "pawn": (string représentant un type de pion),
     "color": (string couleur hexadécimale),
     "legions": (int représentant le nombre de légions contrôlées par le joueur),
     "state": ("waiting"|"playing"|"disconnected"|"left"),
     "cards": (cartes du joueur)
     [
     <carte>,
     <carte>,
     ...
     ]
}
```

Données des cartes :

```
{
     "type": (int représentant le type de la carte),
}
```

Données des équipes :

```
{
     "players": (joueurs de l'équipe)
     [
          <joueur>,
          <joueur>,
          …
     ],
     "cards": (cartes de l'équipe)
     [
          <carte>,
          <carte>,
          …
     ]
}
```

Données d'un jeu :

```
{
     "name": (string),
     "turn_timer": (int représentant le temps pour un tour en secondes),
     "board_size": (int représentant la dimension du plateau),
     "cards": (booléen autorisant ou non les cartes),
     "legion_per_group": (int représentant le nombre de légions par groupe),
     "teams": (équipes de la partie)
     [
          <équipe>,
```

```
            <équipe>,
            …
      ]
}
```

Données d'une case :

```
{
      "u": (int),
      "w": (int),
      "armor": (booléen représentant la présence d'une armure),
      "laurel": (booléen représentant la préésence du laurier ou non),
      "tent": (booléen représentant la présence d'une tente ou non),
      "team": (int représentant l'équipe a qui appartient la légion sur la case, -1 si pas de légion)
      "player": (int représentent le membre e l'équipe qui appartient la légion sur la case, -1 si pas de légion)
      "death: ("none"|"battle"|"surrounding" représentant comment la légion de la case va mourrir)
}
```

Données d'un déplacement :

```
{
      “start”:
      [
            u,
            w
      ]
```

```
	"end":
	[
		u,
		w
	]
}
```

Données d'un tour :

```
{
	"board": (liste des cases non vides)
	[
		<case>,
		<case>,
		…
	],
	"moves": (liste des déplacements du tour)
	[
		<déplacement>,
		<déplacement>,
		…
	],
	"teams": (liste des équipes et des joueurs)
	[
		<équipe>,
		<équipe>,
		…
	],
```

```
    "cards": (liste des cartes jouées, si cartes activées)
    [
    {
        "team": (int représentant la team),
        "player": (int représentant le joueur),
        "team_card": (booléen indiquant s'il s'agit d'une carte joueur ou équipe),
        "card": (int représentant la position de la carte dans la liste des cartes du joueur ou de l'équipe)
    },
        (...)
    ]
}
```