# Objectifs

Créer une application permettant de jouer à Chi Fou Mi. Le jeu se joue à deux joueurs en ligne. Une partie se joue en trois manches.

# Authentification

L'authentification se fait par token JWT

# Endpoints

Base de l'URL: à voir en cours Les sources du serveur sont disponbles sur Github

# Ce qui est demandé

## Fonctionnel

- Une page permettant de se connecter/s'enregistrer
- Une page permettant de lister les parties et de créer une partie
- Une page permettant d'afficher une partie et de pouvoir jouer
- Avoir un semblant d'UI (utilisation de packages possibles, ex: mui, tailwind, ...)

## Technique

- Avoir un découpage de composants efficaces
- Mise en place de contexts/custom hooks

## Bârème (15pts)

- 9pts: Fonctionnel
- 2pts: UI
- 4pts: Architecture logiciel (hiérarchie des fichiers, découpage des composants)

# Fonctionnalités supplémentaires (5pts)

- Utilisation du système de notification (SSE) pour les événements (voir plus bas pour les différents types d'événements) (+3pts)
- Ajout d'animations (+1pts)
  - ex: Révéler les coups des jouers via une carte qui se retourne
- Statistiques des parties jouées (+1pts)
  - ex: Nombre de parties gagnées, perdues, nombre de parties jouées, ...
- Dévier le système de jeu pour créer un jeu ChiFouMi-like (+1pts)
  - ex: Ajouter un coup "feu" qui bat tout, mais qui ne peut être joué qu'une fois par partie

- ... (à faire valider)

# Rendu

Le rendu doit se faire via un lien Github **publique**. Les différents membres de l'équipe doivent chacun travailler sur le projet donc s'il n'y a pas de commits l'étudiant aura 0.

## POST /register

- Requête

```
{
  "username": "monpseudo",
  "password": "mypassword"
}
```

- Réponse

```
// Code: 201
{
  "_id": "IDUSER",
  "username":"token",
  "password":"password"
}
```

## POST /login

- Requête

```
{
  "username": "monpseudo",
  "password": "mypassword"
}
```

- Réponse

```
// Code: 200
{
  "token":"token"
}
```

## GET /matches

- Body

```
// Code: 200
[
  {
    "user1": {
      "_id": "24aefbbb-8def-4e2c-b19a-929ff55020c0",
      "username": "player1",
    },
    "user2": null, //{"_id": "24aefbbb-8def-4e2c-b19a-
929ff55020c1","username":     "player2"}
    "turns": [],
    "_id": "61979ce9ff4a0e83e02df260",
  }
  // ,...
]
```

## GET /matches/:id

- Body

```
// Code: 200
{
  "user1": {
    "_id": "24aefbbb-8def-4e2c-b19a-929ff55020c0",
    "username": "player1",
  },
  "user2": null, //{"_id": "24aefbbb-8def-4e2c-b19a-
929ff55020c1","username": "player2"}
  "turns": [],
  "_id": "61979ce9ff4a0e83e02df260",
}
```

## POST /matches

Si un match est en attente (pas de user2), on le modifie pour ajouter le user2user2

- Body Aucun
- Réponse

```
// si pas de match en attente pour l'utilisateur courant
// Code: 201
{
  "user1": {
    "_id": "24aefbbb-8def-4e2c-b19a-929ff55020c0",
    "username": "player1",
  },
```

```
  "user2": null, //{"_id": "24aefbbb-8def-4e2c-b19a-
929ff55020c1","username": "player2"}
  "turns": [],
  "_id": "61979ce9ff4a0e83e02df260",
}
// sinon
// Code: 400
{
  "match": "You already have a match"
}
```

### POST /matches/:id/turns/:idTurn

- Body

```
{
  "move": "rock" // "rock", "paper", "scissors"
}
```

- Réponse
  - Erreur 400
    - si idTurn est invalide `{ turn: "not found" }`
    - si idTurn est déjà terminé `{ turn: "not last" }`
    - si mmatch est déjà terminé `{ match: "Match already finished" }`
    - si le joueur a déjà joué le tour et attend l'adversaire `{ user: "move already given" }`
  - Code 202: Si tout se passe bien

## Notifications du match

A chaque événement lié à un match, une notification est envoyée via le protocole Server-Sent Events (SSE).

L'endpoint pour souscrire aux notifications est /matches/:id/subscribe

L'endpoint est lui aussi protégé par un token JWT (Bearer), utilisé la lib Yaffle EventSourcePolyfill

### Event PLAYER_JOIN

```
{
  "type": "PLAYER1_JOIN", // "PLAYER1_JOIN"|"PLAYER2_JOIN"
  "matchId": "id_match",
  "payload": {
    "user": "player1_username"
  }
}
```

## Event NEW_TURN

```
{
  "type": "NEW_TURN",
  "matchId": "id_match",
  "payload": {
    "turnId": 1,
  }
}
```

## Event TURN_ENDED

```
{
  "type": "TURN_ENDED",
  "matchId": "id_match",
  "payload": {
    "newTurnId": 2,
    "winner": "winner_username", // "winner_username"|"draw",
  }
}
```

## Event PLAYER_MOVED

```
{
  "type": "PLAYER1_MOVED", // "PLAYER1_MOVED"|"PLAYER2_MOVED"
  "matchId": "id_match",
  "payload": {
    "turn": 1,
  },
}
```

## Event MATCH_ENDED

```
{
  "type": "MATCH_ENDED",
  "matchId": "id_match",
  "payload": {
    "winner": "winner_username", // "winner_username"|"draw",
  },
}
```