**Documentation**

**Fullstack Web service M2**

**INJIN KIM**

**Description**

**Nom de l’application :** Rock-Scissors-Paper

**Objectif de l’application :** Une application permettant aux utilisateurs de s’enregistrer / se connecter et de créer des salles de discussions. Dans les salles de discussions ils peuvent jouer au jeu « Pierre papier ciseaux »

**Technologies**

**Framework full stack:** AdonisJs V5

**Base de données :** Sqlite

**Dépendance du projet :**

* SocketIo : pour la communication des joueurs (chat)
* Luxon : ORM pour communication avec la BDD
* Axios : permettant dans le front des appels vers le back end

Une image contenant Police, logo, Graphique, symbole

Description générée automatiquementUne image contenant cylindre, conception

Description générée automatiquement

**Une image contenant Police, logo, symbole, Graphique

Description générée automatiquement**

**Objectifs de l’application**

**Rest :** L’application propose une API REST permettant de

* S’enregistrer (login/mot de passe)
* Se connecter
* CRUD sur les salles de jeu
  + Liste les salles
  + Créer une nouvelle salle
  + Supprimer une salle

**Socket :** L’application utilise les sockets permettant d’ouvrir les communications entre les utilisateurs pour

* Envoyer des messages dans un chat classique
* Jouer à un jeu de type pierre/feuille/ciseau en direct la page attend que les deux joueurs sélectionnent leur choix avant de déterminer un gagnant.

**Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement**

**Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement**

**Présentation des pages**

**Accueil**

L’application se lance sur <http://127.0.0.1:3333/> grâce à la commande :

* **node ace serve –watch**

**Prérequis :** Avoir node et npm

**Une image contenant texte, grenouille, clipart, capture d’écran

Description générée automatiquement**

Actions de la page :

* Bouton **Login** redirection vers la page de connexion
* Bouton **Register** redirection vers la page d’inscription
* Bouton **Rooms lobby** redirection vers la page de liste des salles de jeux (demande d’être connecté d’abord)

**Login**

La page de connexion est accessible sur l’url <http://127.0.0.1:3333/login>

Aucun prérequis accéder à cette page

**Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement**

Actions sur la page :

* **Email** : information obligatoire pour se connecter
* **Password** : mot de passe obligatoire pour se connecter
* Bouton **Login** permet d’appeler le backend pour faire la connexion et recevoir une session

La session est gérée dans le back grâce à Adonis :



**Erreurs possibles**

* **Si Email absent:** Email and password are necessary for login.
* **Si email introuvable:** This email is not registered
* **Si mot de passe mauvais**: Invalid credentials

**Register**

La page d’inscription est accessible sur l’url http://127.0.0.1:3333/register

Aucun prérequis accéder à cette page

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Actions sur la page :

* **Email** : information obligatoire pour s’inscrire
* **Password** : mot de passe obligatoire pour s’inscrire
* Bouton **Register** permet d’appeler le backend pour faire l’inscription

Le mot de passe est Hash dans le backend avant d’être dans la BDD :

**Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement**

**Erreurs possibles**

* **Si Email absent:** Email and password are necessary for register.
* **Si email déjà existant:** This email already exist

Redirection vers la page de connexion une fois que l’inscription est faites.

**Lobby**

La page de lobby est accessible sur l’url <http://127.0.0.1:3333/lobby>

Il est **obligatoire** d’être connecté pour accéder à cette page sinon redirection vers la page de **login**

**Une image contenant texte, capture d’écran, Police, Système d’exploitation

Description générée automatiquement**

Actions sur la page :

* **Create a room** : permet la création d’une nouvelle salle de jeu (nom **obligatoire** pour la création. Il doit aussi être **unique**)
* **Bouton bleu**: permet de recharger la liste des salles disponible si quelqu’un d’autre en créé en même temps
* **Join**: permet de rejoindre une salle de jeu
* **Delete**: permet de supprimer la salle de jeu

**Erreurs possibles**

* **Si pas de nom de salle :** A room need a name.
* **Si nom déjà existant:** A room already exists with this name.

**Salle de jeu**

La page de lobby est accessible sur l’url <http://127.0.0.1:3333/room?name=Injin%20room>

Il est **obligatoire** d’être connecté pour accéder à cette page sinon redirection vers la page de **login**

Une image contenant texte, capture d’écran, Système d’exploitation, logiciel

Description générée automatiquement

Le nom de l’utilisateur (email) avec un bouton de retour est disponible en haut à gauche.

Dans cette page :

* On affiche du nom de la salle de jeu qui est unique

Une image contenant texte, Police, vert, capture d’écran

Description générée automatiquement

* Une section pour jouer est disponible (2 joueurs sont nécessaires)

Une image contenant texte, capture d’écran, cercle

Description générée automatiquement

* Un chat permet de discuter avec les autres utilisateurs

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Fonctionnement des sockets dans l’application :

Pour la partie chat lors de l’ouverture de la page, une connexion est faite au socket avec le nom de la salle en paramètre pour identifier comment les messages retours doivent revenir

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

L’envoi d’un message considère à récupérer la string et de faire un **emit** qui permet d’envoyer aux autres le message. On en profite pour afficher aussi notre message dans la liste.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour la partie jeu :

Connexion lors de l’arriver sur la page au socket **game** et attend que le manager de socket répond quand deux joueurs on participés.

L’application affiche en fonction du gagnant si c’est il gagne / perd ou égalité

Une image contenant texte, capture d’écran

Description générée automatiquement

Lorsqu’un joueur veux jouer on fait donc un **emit** avec son choix



Exemples :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Utilisation du template avec Adonis**

Avec adonis dans le projet j’utilise des conditions d’affichage comme

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Et aussi de l’injection de données via les controller exemple :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour montrer le lobby je renvoie l’utilisateur courant ainsi que la liste des rooms du lobby

**Gestion de profil**

Dans l’application il y a deux profils. Cela dépend de la valeur de is\_admin dans la BDD

* Admin
* Simple utilisateur

L’admin peut accéder à une page supplémentaire uniquement pour les admins qui liste les utilisateurs de l’application **http://127.0.0.1:3333/admin/users**

Une image contenant texte, capture d’écran, Police, menu

Description générée automatiquement

Si l’utilisateur n’est pas admin alors il est redirigé vers la page HOME de l’application

**APRES ORAL MODIFICATIONS**

1. Ajout d’un système de Rôle avec Adonis

* Migration base donnée pour enlever la colonne is-admin
* Création d’un model Rôle et d’une migration pour création d’une colonne role\_id dans User foreign key de la table ROLE avec pour valeur 1 (user) ou 2(admin)
* Création d’une seed pour mettre tous les users à 1 sauf l’adresse email [ikim@gmail.com](mailto:ikim@gmail.com) à 2 (admin)

1. Ajout d’un middleware avec Adonis

Création d’un middleware admin pour vérifier si le user connecté est un admin ou pas

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Le middleware permet ici de vérifier directement si l’utilisateur est admin ou pas



Dans les routes on oblige la route de faire auth ET admin middleware avant d’arriver dans le controller.

1. Ajout d’un bouton sur la page home pour aller sur la page admin

Une image contenant texte, capture d’écran, grenouille, clipart

Description générée automatiquement

Une image contenant capture d’écran, texte, Police, ligne

Description générée automatiquement

Rajout d’un controller pour vérifier si le user est admin ou non

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Rajout d’un bouton retour sur la page admin

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Mise en place de la sécurité Shield CSRF

Configuration et ajout d’un middleware

* npm i @adonisjs/shield@7.1.1
* node ace configure @adonisjs/shield
* Server.middleware.register([
* () => import('@ioc:Adonis/Core/BodyParser'),
* () => import('@ioc:Adonis/Addons/Shield')
* ])

Exemple ajout dans le form register

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Exception pour les routes home et admin users qui n’ont pas de formulaire

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement