UFR de mathématique et d'informatique
Université de Strasbourg

UNIVERSITÉ DE STRASBOURG

PROJET INTÉGRATEUR - POKERGANG

DOCUMENTATION TECHNIQUE

Hayk ZARIKIAN – Lead Tech Game Engine

Yannick LECAM – Lead Tech Server

Johary RALAMBO et Ahmed ZANED – Lead Tech Front

Ekaterina ZAITCEVA – Lead Tech Base de données

ÉQUIPE 6B - RoundCoders

Table des matières

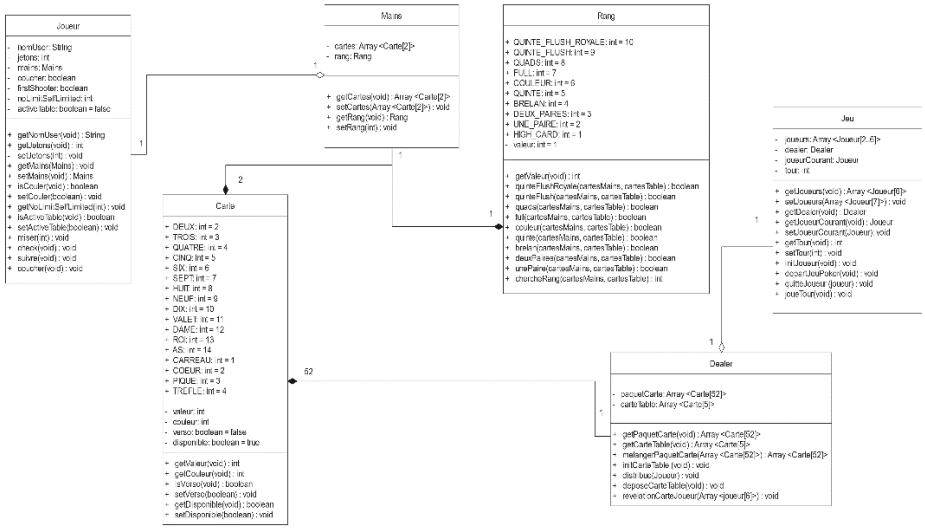
1 Socle technique

Pour que notre projet fonctionne correctement, nous avons utilisés plusieurs dépendances, qui sont les suivants :

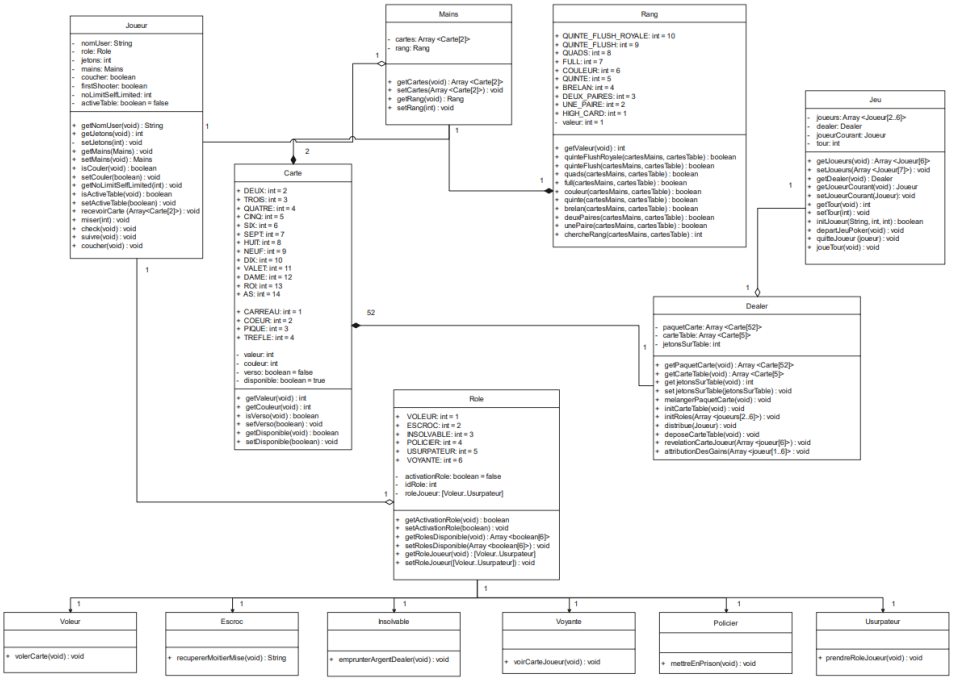
- Framework VueJs : Utilisation pour le Front-End
- Framework ElectronJs : Utilisation de base pour la transformation de l'application web en desktop
- Framework JestJs : Utilisation de jest pour effectuer les tests unitaires
- Plateforme de développement NodeJs : Utilisation pour le serveur
- Langage JavaScript : Langage Back-End principal
- Langage HTML, CSS : Langage Front-End principal
- Langage SQLite : Langage Base de données principal

2 Backend Jeu

2.1 Diagramme de classe du jeu de base



2.2 Diagramme de classe (avec fonctionnalité rôle)



2.3 Classe Carte

2.3.1 Description attributs

VALEUR DES CARTES		
Type	Nom de l'attribut	Description
int	const DEUX = 2	Carte de valeur 2
Int	const TROIS = 3	Carte de valeur 3
Int	const QUATRE = 4	Carte de valeur 4
Int	const CINQ = 5	Carte de valeur 5
int	const SIX = 6	Carte de valeur 6
Int	const SEPT = 7	Carte de valeur 7
int	const HUIT = 8	Carte de valeur 8
int	const NEUF = 9	Carte de valeur 9
int	const DIX = 10	Carte de valeur 10
int	const VALET = 11	Carte de valeur valet
Int	onst DAME = 12	Carte de valeur dame
Int	const ROI = 13	Carte de valeur roi
int	const AS = 14	Carte de valeur as

VALEUR DES COULEURS		
Type	Nom de l'attribut	Description
int	const CARREAU = 1	Valeur de la couleur CARREAU
Int	const CŒUR = 2	Valeur de la couleur CŒUR

int	const PIQUE = 3	Valeur de la couleur PIQUE
int	const TREFLE = 4	Valeur de la couleur TREFLE

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
int	valeur	Valeur de la carte	La valeur est comprise entre 2-14
int	couleur	Valeur de la couleur	La valeur est comprise entre 1-4
boolean	disponible	Vrai si la carte est disponible, faux sinon	La valeur par défaut est « true »

2.4 Classe Dealer

2.4.1 Description Attributs

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
Carte	paquetCarte = new Array(52)	Tableau de 52 cartes	
Carte	carteTable = new Array(5)	Tableau des 5 cartes sur la table	
int	jetonsSurTable = 0	Jetons présents sur la table	
boolean	rolesDisponible = new Array(6)	Tableau de boolean représentant la disponibilité des rôles	

2.4.2 melangerPaquetCarte ()

Paramètres	Aucun
Description	Mélanger les cartes de manières aléatoires
Retour	Modification de l'attribut « `paquetCarte[52]` » par l'adresse

2.4.3 initCarteTable ()

Paramètres	Aucun
Dépendances	melangerPaquetCarte ()
Description	Initialise les 5 cartes de la table aléatoirement dans l'attribut
Retour	Modification de l'attribut « `carteTable[5]` » par l'adresse

2.4.4 initRoles ()

Paramètres	Joueurs [2..6] : Tableau de joueurs ayant au minimum 2 joueurs initialisés et 6 joueurs au maximum initialisés
Dépendances	Constructeur Role
Description	Initialise le rôle des joueurs
Retour	Initialisation de l'attribut « `role` » dans la classe « `Joueur.js` ». Modification de la disponibilité du rôle dans l'attribut « `rolesDisponible[2..6]` »

2.4.5 distribue ()

Paramètres	joueur : Objet de type Joueur
Dépendances	❖ melangerPaquetCarte () ❖ joueur.recevoirCarte ()
Description	Distribution de deux cartes à un joueur aléatoirement
Retour	Initialisation de l'attribut « `mains` » dans la classe « `Joueur.js` » et l'attribut « `cartes` » dans la classe « `Mains.js` » par l'adresse.

2.4.6 revelationCarteJoueur ()

Paramètres	Joueurs [2..6] : Tableau de joueurs ayant au minimum 2 joueurs initialisés et 6 joueurs au maximum initialisés
Dépendances	attributionDesGains ()
Description	Détermination du / des vainqueur(s) et attribution des gains
Retour	Retourne les indices des joueurs gagnants

2.4.7 attributionDesGains ()

Paramètres	❖ joueurs [2..6] : Tableau de joueurs ayant au minimum 2 joueurs initialisés et 6 joueurs au maximum initialisés.

	❖ indicesJoueursGagnants[1..6] : Tableau des indices des joueurs gagnants
Dépendances	BDD.add_winner ()
Description	Permet de mettre à jour la base de données pour les joueurs gagnants
Retour	Met à jour les jetons des joueurs gagnants depuis la base données par l'adresse

2.5 Classe Mains

2.5.1 Description Attributs

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
Carte	Cartes = new Array(2)	Tableau de deux cartes dans la main	
Rang	Rang	Objet de type rang	

2.5.2 creeRang ()

Paramètres	carteTable[5] : Tableau qui représente les 5 cartes de la table
Dépendances	Constructeur Rang
Description	Initialisation du rang du joueur
Retour	Initialise l'attribut « rang » dans la classe « Mains.js » par l'adresse

2.6 Classe Rang

2.6.1 Description Attributs

VALEURS DES RANGS DE MAIN TRADITIONNELLE		
Type	Nom de l'attribut	Description
int	const QUINTE_FLUSH_ROYALE = 10	Main QUINTE_FLUSH_ROYALE
int	const QUINTE_FLUSH = 9	Main QUINTE_FLUSH
int	const QUADS = 8	Main QUADS
int	const FULL = 7	Main FULL

int	const COULEUR = 6	Main COULEUR
int	const QUINTE = 5	Main QUINTE
int	const BRELAN = 4	Main BRELAN
int	const DEUX_PAIRES = 3	Main DEUX_PAIRES
int	const UNE_PAIRE = 2	Main UNE_PAIRE
int	const HIGH_CARD = 1	Main HIGH_CARD

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
int	valeur	Valeur du rang	La valeur est comprise entre 1-10

2.6.2 quinteFlushRoyale ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Dépendances	❖ couleur () ❖ quinteFlush ()
Description	Détermine si le joueur possède un rang de niveau 10 représentant la main « `quinteFlushRoyale` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 10.

2.6.3 quinteFlush ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 9 représentant la main « `quinteFlush` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 9.

2.6.4 quads ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 8 représentant la main « `quads` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 8.

2.6.5 full ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Dépendances	❖ brelan () ❖ unePaire ()
Description	Détermine si le joueur possède un rang de niveau 7 représentant la main « full »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 7.

2.6.6 couleur ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 6 représentant la main « `couleur` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 6.

2.6.7 quinte ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 5 représentant la main « `quinte` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 5.

2.6.8 brelan ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 4 représentant la main « `brelan` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 4.

2.6.9 deuxPaires ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur

Description	Détermine si le joueur possède un rang de niveau 3 représentant la main « `deuxPaires` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 3.

2.6.10 unePaire ()

Paramètres	cartesMainsTable[7] : Tableau de 7 cartes qui représente les 5 cartes de la table et les 2 cartes de la main du joueur
Description	Détermine si le joueur possède un rang de niveau 2 représentant la main « `unePaire` »
Retour	Retourne vrai ou faux si le joueur possède un rang de niveau 2.

2.6.11 chercheRang ()

Paramètres	❖ cartesMains[2] : Tableau des 2 cartes du joueur ❖ cartesTable[5] : Tableau des 5 cartes de la table
Dépendances	❖ quinteFlushRoyale () ❖ quinteFlush () ❖ quads () ❖ full () ❖ couleur () ❖ quinte () ❖ brelan () ❖ deuxPaires () ❖ unePaire ()
Description	Permet de chercher le rang du joueur
Retour	Retourne le niveau du rang

2.7 Classe Joueur

2.7.1 Description Attributs

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
int	idJoueur	L'identifiant du joueur	L'identifiant doit être cohérent avec celui de la base de données
Role	Role	Objet de type Role	
Mains	Mains	Objet de type Mains	

int	Jetons	Nombre de jeton	
int	deposeJetons	Nombre de jeton déposé dans la partie courante	
boolean	Coucher = false	Vrai si le joueur est coucher, faux sinon	La valeur par défaut est « false »
boolean	firstShooter = false	Vrai si le joueur est le premier joueur à jouer, faux sinon	La valeur par défaut est « false »
int	noLimitSelfLimited	La limite du joueur en jeton	
boolean	jaiDejaJouerMonTour = false	Vrai si le joueur a déjà joué, faux sinon	La valeur par défaut est « false »

2.7.2 recevoirCarte ()

Paramètres	cartes[2] : Tableau des 2 cartes du joueur
Dépendances	Constructeur Mains
Description	Initialise l'attribut « `mains` »
Retour	Initialisation de mains par l'adresse

2.7.3 miser ()

Paramètres	❖ dealer : Un objet de type Dealer ❖ jetons : Entier représentant le nombre de jeton
Dépendances	BDD.bet ()
Description	Permet de miser des jetons
Retour	Met à jour la base de données

2.7.4 check ()

Paramètres	Aucun
Description	Le joueur passe son tour
Retour	Modification de la variable « `jaiDejaJouerMonTour` » à true

2.7.5 seCoucher ()

Paramètres	Aucun

Description	Permet au joueur de se coucher
Retour	Modification de la variable « `jaiDejaJouerMonTour` » et « `coucher` » à true

2.8 Classe Jeu

2.8.1 Description Attributs

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
Joueur	Joueurs = new Array(6)	Tableau des joueurs	Le nombre de joueur est compris entre 1-6
Dealer	Dealer = new Dealer	Objet de type Dealer	
Joueur	joueurCourant	Joueur courant	
int	Tour	Tour courant	La valeur est comprise entre 1-5
Joueur	joueursEnCours	Tableau des joueurs en cours de partie	Le nombre de joueur en cours et compris entre 2-6
int	indiceFirstShooter	Indice du premier joueur	
int	nbJoueur	Nombre de joueur restant à jouer	La valeur est comprise entre 0-6
boolean	finPartie = false	Vrai si la partie est terminée, faux sinon	La valeur par défaut est « false »

2.8.2 initJoueur ()

Paramètres	❖ idJoueur : Identifiant du joueur ❖ jetons : Jetons mis en place sur la table (modifiable) ❖ noLimitSelfLimited : Jetons mis en place sur la table (non modifiable)
Dépendances	Constructeur Joueur
Description	Initialise le joueur si une place est disponible
Retour	Retourne vrai si l'initialisation du joueur c'est bien passée, faux sinon. Modification du tableau de « `joueurs[]` » par l'adresse.

2.8.3 departJeuPoker ()

Paramètres	Aucun
Dépendances	❖ Dealer.initCarteTable () ❖ Dealer.distribue () ❖ Dealer.initRoles () ❖ Joueur.miser () ❖ jouerTour ()
Description	Lance le jeu de poker tout en initialisation les prérequis
Retour	Aucun

2.8.4 quitterJoueur ()

Paramètres	❖ joueurs : Tableau des joueurs[2..6] ❖ index : Indice du joueur qui souhaite quitter
Description	Quitter joueur, libération de la place dans le tableau
Retour	Libère une place dans le tableau par l'adresse

2.8.5 jouerTour ()

Paramètres	
Description	
Retour	

2.9 Classe Role

2.9.1 Description attributs

VALEURS DES ROLES		
Type	Nom de l'attribut	Description
int	const VOLEUR = 0	Role VOLEUR
int	const ESCROC = 1	Role ESCROC
int	const INSOLVABLE = 2	Role INSOLVABLE
int	const POLICIER = 3	Role POLICIER
int	const USURPATEUR = 4	Role USURPATEUR
int	const VOYANTE = 5	Role VOYANTE

ATTRIBUTS A INITIALISER			
Type	Nom de l'attribut	Description	Contraintes
boolean	activationRole	Vrai si le pouvoir a été	

		activé, faux sinon	
int	idRole	Identifiant du role	La valeur est comprise entre 0-6
[VOLEUR..VOYANTE]	roleJoueur	Objet de type Voleur, Escorc, Insolvable, Policier, Usurpateur ou Voyante	

2.10 Classe Escroc

2.10.1 recupererMoitierMise ()

Paramètres	Joueur : le joueur escroc
Description	Permet de récupérer la moitié de sa mise en se couchant
Retour	Modification de l'attribut « `coucher` » pour le joueur en question

2.11 Classe Usurpateur

2.11.1 copierRoleJoueur ()

Paramètres	❖ joueur : le joueur voleur ❖ joueurs : Le tableau des joueurs
Description	Copie le rôle d'un des joueurs aléatoirement
Retour	Modification de l'attribut « role » dans la classe « Joueur.js »

2.12 Classe Policier

2.12.1 mettreEnPrison ()

Paramètres	Joueurs : Le tableau des joueurs
Description	Permet de faire coucher un joueur
Retour	Modification de l'attribut « `coucher` » pour le joueur en question

2.13 Classe Insolvable

2.13.1 emprunterArgentDealer ()

Paramètres	
Description	Permet d'emprunter des jetons au dealer
Retour	

2.14 Classe Indicateur

2.14.1 indiquerCarteJoueur ()

Paramètres	
Description	Permet de voir l'une des cartes d'un des joueurs
Retour	

2.15 Classe Voleur

2.15.1 volerCarte ()

Paramètres	❖ joueur : le joueur voleur ❖ joueurs : Le tableau des joueurs
Description	Échange la carte avec un autre joueur
Retour	Changement de l'attribut « cartes[0 ou 1] » dans la classe « mains.js »

3 Scripts Python : Mail automatique

3.1 Introduction

La création du script python « mail.py <email> » permet d'envoyer un mail via SMTP du compte « round.coders@gmail.com » afin de notifier l'inscription du joueur sur le jeu PokerGang.

3.2 python_launch_mail.js

L'exécution du script python est réalisé par un processus fils qui est créé en langage JS.

3.3 mail.py

Configuration du SMTP :

```
# Configuration SMTP

smtp_server = "smtp.gmail.com"
smtp_port = 587

smtp_login = "round.coders@gmail.com"
to_email = sys.argv[1]
auth_code = "ecujnyocilfwujtk" # Code chiffré

msg = MIMEMultipart ()
msg['From'] = smtp_login
msg['To'] = to_email
msg['Subject'] = "Sujet du message"
body = "Message"

msg.attach(MIMEText(body, 'plain'))

# Fin configuration SMTP
```

Attachement du logo d'équipe :

```
# Attachement du logo équipe

with open('logo.png', 'rb') as f:
    img_data = f.read()
    f.seek(0)  # Remettre le curseur à la position de départ
pour éviter l'erreur 'ValueError: seek of closed file'

    # Resize the image to a maximum width of 200 pixels
    img = Image.open(f)
    max_width = 200
    width, height = img.size
    if width > max_width:
        ratio = max_width / width
        img = img.resize((max_width, int(height * ratio)),
Image.ANTIALIAS)

    # Add the image to the email as an attachment
    img_byte_array = io.BytesIO()
    img.save(img_byte_array, format='PNG')
    img_byte_array = img_byte_array.getvalue()
    img_mime = MIMEImage(img_byte_array, name='logo.png')
    img_mime.add_header('Content-ID', '<logo>')
    img_mime.add_header('Content-Disposition', 'inline',
filename='logo.png')
    msg.attach(img_mime)

# Fin attachement du logo équipe
```

Signature du mail générique :

```
# Ajout de la signature
```

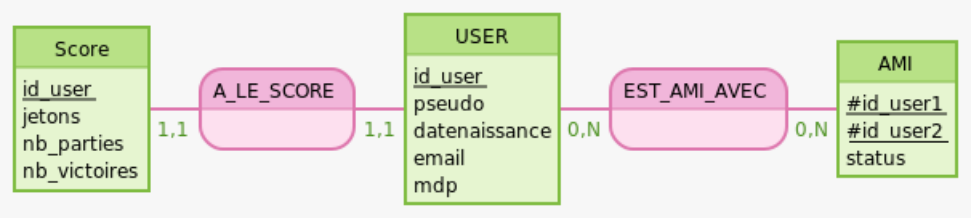
```
signature = """<br>
-- <br>
Bien à vous,<br>
<br>
Equipe RoundCoders<br>
round.coders@gmail.com<br>
Université de Strasbourg<br>
UFR Mathématique et Informatique<br>
Licence Informatique<br>
L3-S6 Equipe 6B"""
signature += f'<br><img src="cid:logo" alt="logo"
style="display:block; max-width:{max_width}px;">'
msg.attach(MIMEText(signature, 'html'))

#Fin ajout de la signature
```

Connexion SMTP et envoi du mail :

```
# Connexion SMTP et envoi du mail

with smtplib.SMTP(smtp_server, smtp_port) as server:
    server.starttls()
    server.login(smtp_login, auth_code,
initial_response_ok=False)
    server.docmd("AUTH", "XOAUTH2" + auth_code)

    server.sendmail(smtp_login, to_email, msg.as_string())
    print ("Message envoyé à " + to_email)

# Fin connexion SMTP et envoi du mail
```

4 Base de données

4.1 Schéma EA



4.2 Fonctions

4.2.1 check_pseudo_exists

Paramètres	pseudo : nom de l'utilisateur
Description	Vérifie si le pseudo existe ou non
Retour	Retourne vrai si le pseudo existe sinon faux

4.2.2 update_pseudo

Paramètres	❖ Id_user : identifiant d'un utilisateur ❖ pseudo : nom de l'utilisateur
Description	Met à jour le pseudo de l'utilisateur
Retour	Met à jour la base de données

4.2.3 db_open ()

Paramètres	Aucun
Description	Connexion à la base de données
Retour	Accessibilité à la base ouverte

4.2.4 create_tables ()

Paramètres	Aucun
Description	Permet de créer des tables de la base de données et des déclencheurs nécessaires s'ils n'existent pas encore.
Retour	Création des tables

4.2.5 db_close ()

Paramètres	Aucun
Description	Fermeture de la connexion à la base de données
Retour	Accessibilité à la base fermée

4.2.6 inscription ()

Paramètres	❖ pseudo : pseudonyme d'un utilisateur ❖ datenaissance : sa date de naissance ❖ email : son adresse email ❖ mdp : son mot de passe
Dépendances	❖ genSsalt () ❖ hash () de la bibliothèque bcryptjs
Description	Création d'un utilisateur dans la base de données, table User
Retour	Retourne vrai si l'inscription se passe bien, faux sinon

4.2.7 connexion ()

Paramètres	❖ email : adresse email d'un utilisateur

	❖ mdp : son mot de passe
Dépendances	❖ compare () de la bibliothèque bcryptjs
Description	Vérifie si l'email et le mot de passe sont corrects
Retour	Retourne l'identifiant de l'utilisateur ou faux si la connexion si la connexion se passe bien

4.2.8 change_pwd ()

Paramètres	❖ Id_user : identifiant d'un utilisateur ❖ nouveauMdp : son nouveau mot de passe
Description	Met à jour le mot de passe de l'utilisateur
Retour	Met à jour la base de données

4.2.9 update_user_jetons ()

Paramètres	❖ id_user : identifiant d'un utilisateur ❖ nb_jetons : le nombre de jetons qu'il faut ajouter au utilisateur
Description	Ajoute nb_jetons au compte de l'utilisateur
Retour	Met à jour la base de données

4.2.10 bet ()

Paramètres	Id_user : identifiant d'un utilisateur
Dépendances	❖ update_user_jetons () ❖ reload_jetons ()
Description	Vérifie que l'utilisateur n'essaie pas de miser plus de jeton qu'il a sur son compte
Retour	Met à jour la base de données

4.2.11 reload_jetons ()

Paramètres	Id_user : identifiant d'un utilisateur
Dépendances	❖ update_user_jetons () ❖ reload_jetons ()
Description	Vérifie que le nombre de jetons sur le compte de l'utilisateur est inférieur à 5000 et lui ajoute 25000 jetons si c'est le cas
Retour	Met à jour la base de données

4.2.12 take_back_half_bet_money_escroc ()

Paramètres	Id_user : identifiant d'un utilisateur
Dépendances	❖ bet_money : le nombre de jetons misés par un utilisateur au cours de la partie
Description	update_user_jetons ()
Retour	Met à jour la base de données

4.2.13 update_nb_games ()

Paramètres	Id_user : identifiant d'un utilisateur
Description	Augmente le compteur de parties jouées de l'utilisateur par 1
Retour	Met à jour la base de données

4.2.14 update_nb_victories ()

Paramètres	Id_user : identifiant d'un utilisateur
Description	Augmente le compteur de victoires de l'utilisateur par 1
Retour	Met à jour la base de données

4.2.15 add_winner ()

Paramètres	❖ id_user : identifiant d'un utilisateur ❖ nb_jetons : le nombre de jetons qu'il a gagné
Dépendances	❖ update_user_jetons () ❖ update_nb_victories ()
Description	Fonction_wrapper pour mettre à jour le joueur gagnant, augmenter son nombre de victoires et ajouter les jetons gagnés sur son compte
Retour	Met à jour la base de données

4.2.16 get_nb_jetons ()

Paramètres	Id_user : identifiant d'un utilisateur
Description	Retourne le nombre de jetons de l'utilisateur extrait de la base de données
Retour	Le nombre de jetons sur le compte de l'utilisateur

4.2.17 get_user_info ()

Paramètres	Id_user : identifiant d'un utilisateur
Description	Retourne les informations personnelles de l'utilisateur issues de la table User de la base de données
Retour	Un objet JSON contenant les informations personnelles non-sensibles de l'utilisateur, ses ids, pseudos, date de naissance et email

4.2.18 get_user_stats ()

Paramètres	Id_user : identifiant d'un utilisateur
Description	Retourne les statistiques de l'utilisateur issues de la table Score de la base de données
Retour	Un objet JSON contenant les statistiques de l'utilisateur, son id, nombre de jetons, nombre de paries jouées et nombre de parties gagnées

4.2.19 classement_jetons ()

Paramètres	Aucun
Description	Trie tous les utilisateurs en fonction du nombre de jetons et du nombre de victoires (en cas d'égalité de jetons sur leurs comptes) et retourne un array contenant des ids des utilisateurs dans le bon ordre
Retour	Un array de id_users dans l'ordre de classement par le nombre de jetons

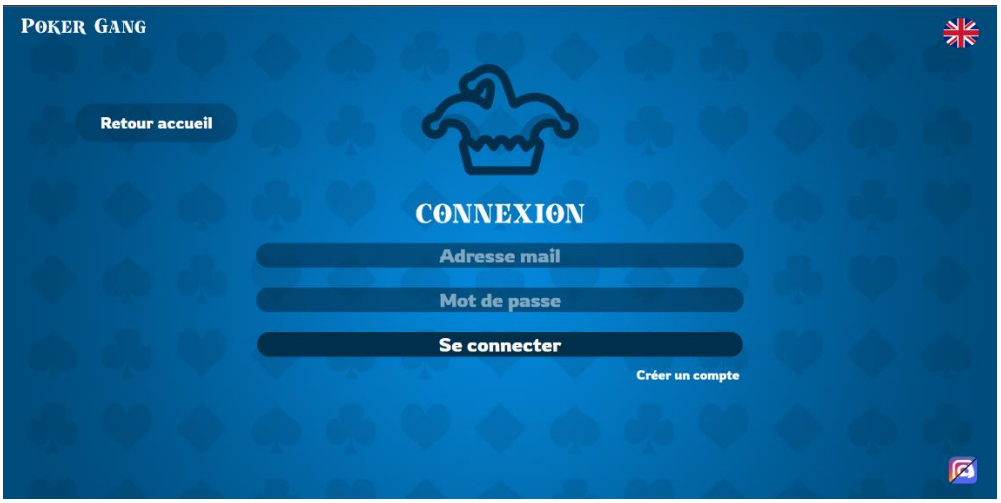
4.2.20 classement_victoires ()

Paramètres	Aucun
Description	Trie tous les utilisateurs en fonction du nombre de victoires et du nombre de jetons (en cas d'égalité de victoires) et retourne un array contenant des ids des utilisateurs dans le bon ordre
Retour	Un array de id_users dans l'ordre de classement par le nombre de victoires

5 Frontend

5.1 Accueil

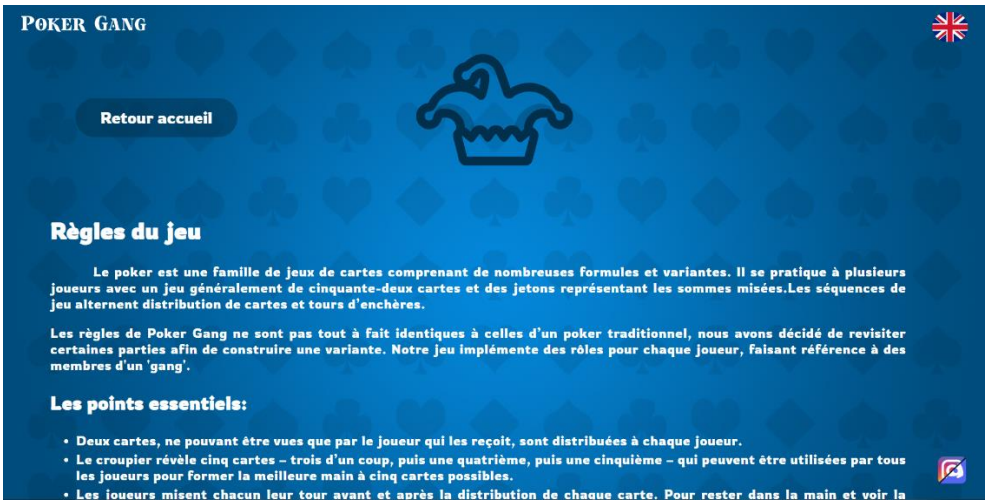
5.2 Connexion



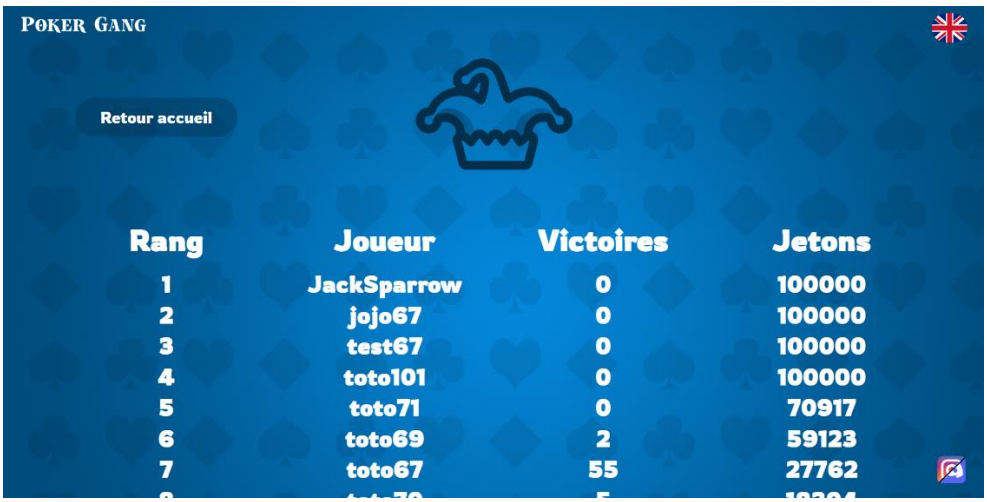
5.3 Inscription

5.4 Menu principal

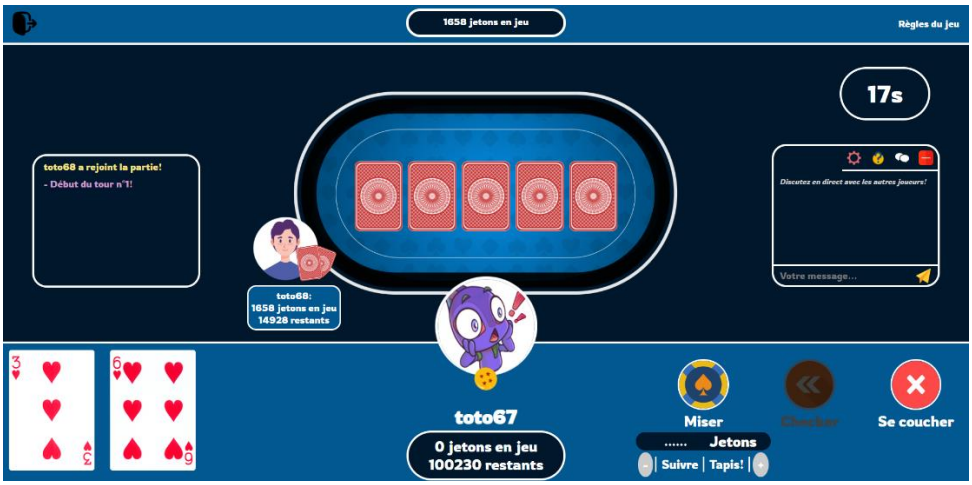
5.5 Règles du jeu



5.6 Classement



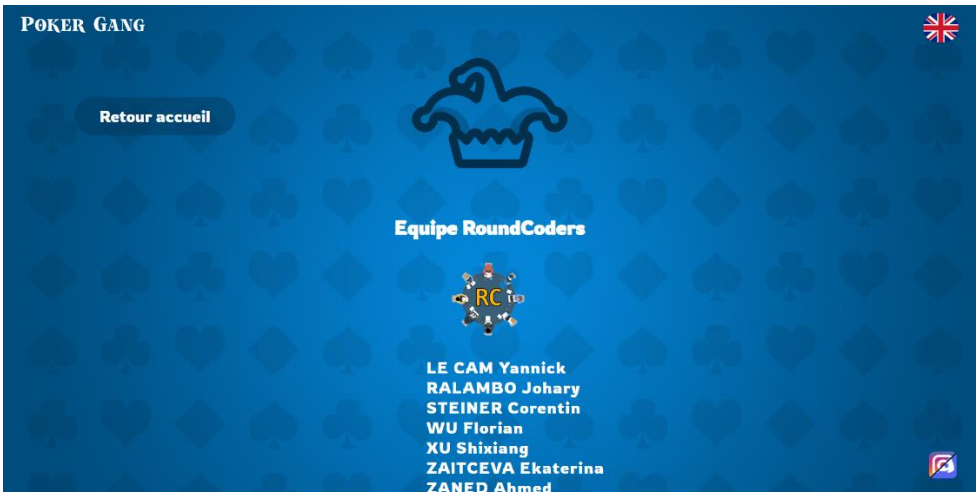
5.7 Interface



5.8 Changer mot de passe

5.9 Contact

5.10 Credits



6 Serveur

6.1.1 Les objets

Table
this.Mise_total=Mise_total
this.Mise_maximum=0
this.Mise_minimum=Mise_minimum
this.Nb_Joueurs=0
this.CartesSurTable=CartesSurTable
this.blind=0;
this.tour=0;
Cet objet permet de gérer tous les arguments présents dans une table de poker

Joueur
this.pseudo=pseudo
this.nb_jetons=nb_jetons
this.Mise=0
this.Coucher=false
this.isPlaying=false
this.isInGame=false
this.isPremierJoueur=false
Cet objet permet de gérer tous les arguments des joueurs sans les données sensible du jeu (tel que la main et le rôle)

JoueurDetail
this.IdJoueur=IdJoueur
this.pseudo=pseudo
this.nb_jetons= nb_jetons
this.Mise=0

this.Mise_Tour=0
this.Coucher=false //Bool
this.Main=0
this.Role=-1//wip
this.Cd_Pouvoir=-1
this.isPlaying=false
this.isInGame=false
this.IdSocket=IdSocket
this.isPremierJoueur=false
Cet objet permet de gérer tous les arguments des joueurs avec toutes les informations essentielles pour la partie.

Update_Data
this.Table=Table//table
this.MonJoueur=MonJoueur//Joueur Detail
this.Joueur1=Joueur1//Joueur
this.Joueur2=Joueur2
this.Joueur3=Joueur3
this.Joueur4=Joueur4
this.Joueur5=Joueur5
Cet objet est fait pour que le front puisse afficher sans se compliquer la tâche, présent dans un tableau et envoyer chaque update_data de façon personnalisée à chacun. L'algorithme permettant de mettre en place le tableau d'Update_Data permet un affichage avec le bon emplacement du joueur parmi l'ordre de jeu.

Main
this.carte1={carte1Valeur,carte1Couleur}
this.carte2={carte2Valeur,carte2Couleur}
Cet objet permet de stocker les mains des joueurs de façon fluide pour le front, un élément de Joueur_Detail.

Spectateur_Update_Date
this.Table=Table
this.MonJoueur=Joueur1
this.Joueur1=Joueur2
this.Joueur2=Joueur3
this.Joueur3=Joueur4
this.Joueur4=Joueur5
this.Joueur5=Joueur6
Cet objet est semblable à Update_Data (mais pour les spectateurs) cependant il n'est composé que de Joueur et n'a pas de JoueurDetail pour ne pas que le spectateur n'est accès au données sensible de jeu tel (la Main et le Rôle)

6.1.2 Les routes

Routes ‘get’
Route ‘/accueil’ : Est une sécurité à une connexion par l’url a la page accueil sans s'être connecté, si l’utilisateur n’est pas connecté il est donc redirigé sur la page de connexion.
Route ‘/jeu’ : Est une sécurité à une connexion par l’url a la page accueil sans s'être connecté, si l’utilisateur n’est pas connecté il est donc redirigé sur la page de connexion.
Route ‘classement’: Est une sécurité à une connexion par l’url a la page accueil sans s'être connecté, si l’utilisateur n’est pas connecté il est donc redirigé sur la page de connexion.
Route ‘/lobby’ :Est une sécurité à une connexion par l’url a la page accueil sans s'être connecté, si l’utilisateur n’est pas connecté il est donc redirigé sur la page de connexion.
Route ‘/connexion’: Si l’utilisateur est déjà connecté alors il est directement envoyé sur la page de l’accueil.
Route ‘/inscription’: Si l’utilisateur est déjà connecté alors il est directement envoyé sur la page de l’accueil.
Route ‘logout’ : Permet à l'utilisateur de se déconnecter (Par exemple pour changer de compte.
Route ‘getMoreToken’ : permet à l'utilisateur de récupérer des jetons une fois qu’il passe sous la barre des 5000 jetons.
Route ‘/getUserStat’ : Permet au front d’afficher les statistiques de l’utilisateur.
Route ‘/getUserInfo’ : Permet au front de récupérer les informations élémentaires de l’utilisateur (Pseudo , nombre de jetons , id_user , …).
Route ‘/getClassementInfo’ : Permet au front d’afficher le classement total des joueurs en fonction des jetons, cependant on peut voir le nombre de parties gagnées par le joueur.

Routes ‘post’
Route ‘/connexion’ : Demande à l'utilisateur d’envoyer un mail et un mot de passe et vérifie que le compte est juste, si juste alors l’utilisateur se fait rediriger sur la page d’accueil et le serveur enregistre les informations de jeu dans une session, Si l’utilisateur transmet des informations erronées alors il reste sur la page d’accueil.
Route ‘/inscription’ : Demande à l'utilisateur de fournir un pseudo, un mail, un mot de passe, sa date de naissance. Après la véracité des informations le site transmet les informations a la BDD, or si il y a un refus de la BDD alors un message est envoyé pour annoncer à l'utilisateur qu’il n’a pas réussi à se connecter lui indiquant pourquoi (Sans trop de détail).
Route ‘/changeusername’: Demande à l'utilisateur son id et le nouveau username, cette requête permet de changer le pseudo de l’utilisateur.
Route ‘/verifierusername’ : Demande à l'utilisateur un username, cette requête vérifie si l’username existe déjà dans la BDD.

6.1.3 Parcours des tableaux

findIndicePlayerParID (idJoueur)

Retourne l'indice du joueur dans les tableaux Joueur et JoueurDetail en prenant comme argument l'idjoueur , renvoie -1 si l'id n'existe pas

updateDesDonnees()
Actualise les tableaux Joueur et JoueurDetail avec les données présentes dans le jeu.

findIndicePlayer(sock)
Retourne l'indice du joueur dans les tableaux Joueur et JoueurDetail en prenant comme argument l'id_socket, renvoie -1 si le socket n'existe pas dans les tableaux.

trouveUnePlaceLibre()
Retourne l'indice dans les tableaux Joueur et JoueurDetail, l'indice qui contient un emplacement vide, s'il n'y a pas d'emplacement vide la fonction renvoie -1.

6.1.4 Gestion des réceptions

Socket connection
Ce type de socket permet au serveur de réceptionner les joueurs entrant dans le socket.

Socket disconnect
Ce type de socket permet au serveur de savoir quand un utilisateur se déconnecte de la partie jeu, par défaut il enclenche le mode déconnexion sauvage.

Socket initConnection
Ce type de socket permet au serveur, d'initialiser avec les données envoyées par le front (id_user , nb_jeton , pseudo) d'initialiser les tableaux joueur et joueurDetail et initialiser ou non en spectateur selon le nombre de joueurs déjà présent autour de la table. Autre condition c'est que si à l'arrivée du joueur il n'y a que 2 joueurs et qu'une partie n'est pas en train de se relancer alors la partie va être lancée.

Socket miser
Ce type de socket permet au serveur quand le front utilise la fonction "miser" en amont. La fonction émit fait un filtrage des informations transmises, dans le côté serveur il y a une lecture et une transmission des données au côté Jeu. Puis renvoie les données de la table avec le prochain joueur qui joue.

Socket message

Ce type de socket permet au serveur de récupérer un message venant d'un utilisateur afin de le transmettre à tous les autres utilisateurs autour de la table transformée en forme d'objet (Pseudo + message).

Socket se coucher
Ce type de socket permet au serveur de récupérer le joueur qui se couche envoyé par le front et le serveur le transmet à la partie jeu.Puis renvoie les données de la table avec le prochain joueur qui joue.

Socket passer
Ce type de socket permet au serveur de récupérer le joueur qui passe (checker) envoyé par le front et le serveur le transmet à la partie jeu.Puis renvoie les données de la table avec le prochain joueur qui joue.

Socket RejoindrePartie
Ce type de socket permet au serveur de détecter les joueurs qui rejouent en les mettant dans un tableau temporaire (réinitialiser dans la fonction RecuperationDesJoueurs).

Socket deconnexion
Ce type de socket est un message du front quand l'utilisateur se déconnecte de façon "propre" , fait appel à la fonction quitter joueur(les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket voyanteVisuCarte
Ce type de socket permet au serveur de récupérer les informations pour utiliser le pouvoir de la "voyante", renommé indicateur (les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket voleurAction
Ce type de socket permet au serveur de récupérer les informations nécessaires pour l'utilisation du pouvoir du voleur (les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket policeAction
Ce type de socket permet au serveur de récupérer les informations nécessaires pour l'utilisation du pouvoir du policier (les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket usurpateurAction
Ce type de socket permet au serveur de récupérer les informations nécessaires pour l'utilisation du pouvoir de l'usurpateur (les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket insolvableAction

Ce type de socket permet au serveur de récupérer les informations nécessaires pour l'utilisation du pouvoir de l'insolvable (les informations nécessaires sont détaillées dans la fonction l'utilisant).

Socket escrocAction
Ce type de socket permet au serveur de récupérer les informations nécessaires pour l'utilisation du pouvoir de l'escroc (les informations nécessaires sont détaillées dans la fonction l'utilisant).

6.1.5 Gestion des émissions

Socket escrocAction
Fonction permettant d'envoyer à chaque utilisateur les informations du(des) gagnant(s) (Pseudo avec sa main). Les infosGagnant sont pour la pluspart des cas en lien avec findPseudoGagnant()

Socket escrocAction
Fonction permettant d'envoyer à chaque joueur autour de la table la version personnalisée qui lui est destinée une version de l'objet Update_Data. Prend également en compte les joueurs spectateur en envoyant un objet Update_Data_Spec, contenant pas les infos sensibles pour le jeu (Les Mains des joueurs et le role).

Socket escrocAction
Fonction permettant d'envoyer l'action du joueur au front afin de faire l'animation adaptée.

Socket escrocAction
Fonction permettant de signaler qu'un joueur vient de se connecter avec son pseudo.

Socket escrocAction
Fonction permettant de signaler au front a quel la partie en cours est actuellement

6.1.6 Gestion de la partie jeu

prochainJoueurAJouer(indiceJoueurCourant)
Renvoie l'indice du prochain joueur à jouer dans la partie en prenant comme argument l'indice du joueur courant

initJoueurPresent()

Initialise dans les tableaux Joueur et JoueurDetail tous les joueurs qui vont commencer la partie.

resetStatutJoueur()
Remet à zéro tous les statuts des joueurs.

recuperationMainsJoueurs()
Récupération de la main de chaque joueur dans l'objet “serveur” MAIN (qui est dans “./Objet.js”), en sélectionnant chaque attribue

recuperationCartesTable(tour)
Permet de récupérer les cartes présentes sur la table afin de l'envoyer le nombre exact de carte à afficher pour le front pour ne pas que les joueurs puissent l'intercepter des informations pouvant faciliter la win en récupérant le tour actuel comme argument.

initNewTour(tour)
Quand un changement de tour s'opère dans la partie alors cette fonction récupère le tour en argument elle est déclenchée afin d'actualiser les actions correspondante

envoieCompteur()
Fonction qui émet un compteur au front entre 2 parties afin qu'il soit synchro

recuperationLeNbDeJetonLePlusBas()
Parcours de la liste entière afin de retourner la valeur de jetons le plus bas autour de la table actuellement

initPremierJoueur()
Se sert de l'attribut dans le jeu "First Shooter" afin de retrouver l'id du joueur pour le comparer à toute la table afin de trouver le bon premier joueur

isAlreadyHere(idJoueur)
Cette fonction regarde si le joueur est autour de la table grâce à son id(en argument), marche avec la feature "déconnexion sauvage"

findPseudoGagnant()
Renvoie les informations du gagnant (Pseudo + Mains)

resetMiseTour()
Reset les Mise_Tour de chaque joueur (est appelé entre chaque dans la fonction InitNewTour)

initBlind(blind,indiceJoueurPremier)
Initialisation des blinds et actualisation pour le front et dans le jeu en récupérant en argument la blind et indice du premier joueur

SuiteDeFinDePartie()
Est la suite de la détection de fin de partie, est utile de la séparer en 2 car la fonction SetTimeout permet de lancer un fonction après un temps donné

recuperationDesJoueurs()
Cette fonction laisse dans le tableau de joueur et joueursDetail tous les joueurs ayant cliqué sur rejouer, fait quitter ceux qui n'ont pas cliqué , et ajoute les spectateurs dans l'ordre d'arrivée de ces dernier

6.1.7 Gestion de la déconnexion des joueurs

Paramètres	idJoueurQuiQuitte : l'id du joueur qui quitte
Description	Supprime l'existence d'un joueur qui part en prenant comme argument l'id du joueur qui quitte, à la fois dans le jeu ainsi que dans les données faites pour transmettre au front.et Prends en considération toutes les situations possibles de chaque joueur avant de quitter.
Retour	Aucun

Paramètres	idJoueurQuiQuitte : l'id du joueur qui quitte
Description	Cette fonction détecte si un joueur a déjà été passer dans quitter joueur en prenant comme argument l'id du socket du joueur qui quitte, mais se fait appeler dans déco joueur.
Retour	Aucun

6.1.8 Gestion des rôles

envoiVisuVoyante(data)	
Paramètres	Data - data[0] : pseudo du joueur ciblé - data[1] : socketid du joueur provenant
Description	Envoyer une carte de la main du joueur souhaité par la voyante (renommé indicateur par la suite).
Retour	Aucun

broadcastJoueurPrison(data)	
Paramètres	Data - data[0] : pseudo du joueur ciblé

	- data[1] : socketid du joueur provenant
Description	Récupère le joueur ciblé et l'interdit d'utiliser son pouvoir jusqu'à la fin de la partie.
Retour	Aucun

broadcastJoueurUsurpateur(data)	
Paramètres	Data - data[0] : pseudo du joueur ciblé - data[1] : socketid du joueur provenant
Description	Récupère le joueur ciblé et clone son rôle jusqu'à la fin de la partie.

broadcastJoueurVoleur(data)	
Paramètres	Data - data[0] : pseudo du joueur ciblé - data[1] : socketid du joueur provenant - data[2] : l'indice dans la main de la carte du voleur qu'il veut se débarasser
Description	Récupère le joueur ciblé et l'indice de la carte du voleur dont il veut se séparer. Ensuite la fonction échange cette dernière avec l'une du joueur ciblé
Retour	Aucun

broadcastJoueurEscroc(data)	
Paramètres	Data - data[0] : socket.id du joueur courant
Description	Ajoute au joueur la moitié de ses mises depuis le début de la partie et le front se charge de le coucher
Retour	Aucun

broadcastJoueurInsolvable(data)	
Paramètres	Data - data[0] : socket.id de l'insolvable - data[1] : somme pour suivre
Description	Ajoute la moitié des jetons de la mise lorsque ce dernier suit avec son pouvoir
Retour	Aucun

recuperationRole(data)	
Paramètres	Aucun
Description	Initialise aléatoirement un rôle à chaque joueur en début de partie en faisant appel à la fonction du jeu
Retour	Aucun

7 Tests unitaires

7.1 Test unitaire Carte

Dans ce test unitaire, nous testons la bonne création des cartes et la bonne initialisation des valeurs.

Création de carte :

- ✓ Valeur() = undefined, couleur() = undefined
- ✓ Valeur(2) = 2, couleur(2) = 2
- ✓ Valeur(0) = undefined, couleur(0) = undefined
- ✓ Valeur(5) = undefined, couleur(0) = undefined
- ✓ Valeur(0) = undefined, couleur(2) = undefined

7.2 Test unitaire Dealer

Dans ce test unitaire, nous avons plusieurs groupes de test :

Création du paquet de carte :

- ✓ Le paquet contient exactement 52 cartes
- ✓ Pas de doublons, les cartes sont uniques
- ✓ La valeur des cartes sont comprise entre 0-52 et la couleur entre 1-4
- ✓ Production de paquet identique à chaque fois

Mélanger les cartes :

- ✓ Le paquet mélangé contient exactement 52 cartes
- ✓ Le paquet mélangé est différent du paquet original
- ✓ Toutes les cartes du paquet original se trouvent dans le paquet mélangé
- ✓ Pas de doublons
- ✓ Produit des paquets différents à chaque fois

Distribuer les cartes :

- ✓ Distribution des cartes aux joueurs
- ✓ Carte indisponible après la distribution

Initialisation des cartes sur la table :

- ✓ Initialise les cartes de la table
- ✓ Carte indisponible après l'initialisation des cartes sur la table

Révélation des cartes du joueur :

- ✓ Création rang et récupération des joueurs gagnants correcte

7.3 Test unitaire Jeu

Création joueur :

- ✓ Initialisation joueur
- ✓ Plus de place disponible

- ✓ Quitter joueur

Départ du jeu :

- ✓ Initialisation des cartes sur la table
- ✓ Initialisation des cartes du joueur

7.4 Test unitaire Mains

Création rang :

- ✓ Le rang est défini
- ✓ Le rang est valide

7.5 Test unitaire Rang

Création bonne valeur du rang :

- ✓ Une paire
- ✓ Deux paires
- ✓ Brelan
- ✓ Quinte
- ✓ Couleur
- ✓ Full
- ✓ Quads
- ✓ Quinte Flush
- ✓ Quinte Flush Royale

7.6 Test unitaire Role

Création du rôle joueur :

- ✓ Assignation du rôle joueur aléatoire

7.7 Test unitaire Escroc

Pouvoir Escroc :

- ✓ Récupération de la moitié de la mise

7.8 Test unitaire Usurpateur

Pouvoir Usurpateur :

- ✓ Copier rôle

7.9 Test unitaire Policier

Pouvoir Policier :

- ✓ Mettre en prison

7.10 Test unitaire Insolvable

Pouvoir Insolvable :

- ✓ Emprunter argent

7.11 Test unitaire Indicateur

Pouvoir Indicateur :

- ✓ Indiquer une info

7.12 Test unitaire Voleur

Pouvoir Voleur :

- ✓ Voler une carte