

Spécifications Projet DAC

Guilhem Stevant, Hicham Al Jaabari, Paul Gauthier, Nils Depuille,
Ayman Azeroual, Isla Gubbay, Yun-fé Girault, Carles Cerqueda

I) Introduction.....	2
Front-end.....	3
Backend.....	3
II) Design Visuel.....	3
Palette de couleurs :.....	3
Typographie :.....	3
Graphiques et icônes :.....	3
Layout Général et Positionnement :.....	4
III) Navigation.....	4
Page d'Accueil :.....	4
Page de connexion :.....	5
Concernant la base de données utilisateur:.....	6
Page d'Accueil Connectée :.....	6
Page de Profil :.....	7
Page de Gestion d'Argent (Ajouter/Retirer) :.....	9
Concernant la gestion de l'argent:.....	10
Gestion des conversions en backend :.....	11
Base de données des dépôts:.....	13
Modal d'instruction de jeu :.....	13
IV) Déroulement des Jeux.....	13
Page de Jeu Roulette :.....	13
Algorithme et règle de la roulette:.....	16
Ensemble des paris.....	16
Roulette en backend :.....	18
Page de Jeu Blackjack :.....	20
BlackEnd BackJack :.....	21
Instructions et Options de Mise :.....	25
Configuration de la Partie :.....	25
Distribution des Cartes :.....	25
Options de Jeu :.....	25
Résultats et mise à jour du solde :.....	25
Option pour jouer :.....	26
Les règles et l'algorithme du black jack:.....	27

1. Objectif du Jeu :.....	27
2. Déroulement du Jeu :.....	28
3. Fin du Jeu :.....	28
4. Algorithme:.....	28
V) Intéractions entre les différentes entités techniques.....	29
Ports :.....	30
Endpoints :.....	31
VII) Diagramme de cas d'utilisation.....	33
VIII) Diagramme de séquence.....	34
1) Diagramme de Séquence : Roulette russe.....	34
2) Diagramme de Séquence : BlackJack.....	35

I) Introduction

Front-end

- Il représente l'interface à travers laquelle les utilisateurs interagissent avec les fonctionnalités et les contenus du site. L'interface utilisateur est développée de telle manière à ce qu'elle donne une première impression positive du site, une navigation fluide et intuitive.
- Il est implémenté en React.

Backend

- Il est implémenté en SpringBoot.
- Il représente l'ensemble des fonctionnalités appelées par le frontend.

Le front-end communique avec le backend pour obtenir les informations nécessaires via une APIRest.

II) Design Visuel

Le design de EasyBet est principalement basé sur la marque EasyJet.

Palette de couleurs :

- Inspiration EasyJet : des teintes vives et accrocheuses, couleur principale étant du orange, pour créer une interface dynamique.
- Les couleurs complémentaires sont le blanc et le noir, le contrast permettant une meilleure visibilité.
- Sur les pages de jeux (roulette, blackjack), le fond sera vert pour rappeler un tapis de jeu.

Typographie :

- Même style que EasyJet
- Différentes tailles de police sont utilisées pour établir une hiérarchie claire.

Graphiques et icônes :

- Sur la plupart des pages, des jetons (légèrement transparents) seront en arrière-plan, pour maintenir le thème du jeu.

Layout Général et Positionnement :

- Tous les écrans suivront un layout de base avec une barre de navigation en haut (hauteur: 60px), un contenu principal au centre, et un pied de page en bas (hauteur: 50px).
- La largeur du contenu principal sera de 1200px centré pour les écrans larges, avec un ajustement responsive pour les écrans plus petits.

Navigation Bar :

- Logo 'easyBet' positionné à 30px du bord supérieur gauche de la page et à 30px du bord gauche.
- Barre de navigation comprenant 'BLACKJACK', 'ROULETTE', et d'autres liens alignés horizontalement avec une marge de 20px entre chaque élément.
- Bouton 'LOG OUT' situé à 30px du bord supérieur droit et à 20px du bord droit de la page.
- Icône du profil et solde du compte affichés à droite, alignés avec le bouton 'LOG OUT'.

FOOTER:

- Hauteur de la zone de pied de page de 50px.
- Police du pied de page à 14px.

Ces choix visuels, utilisés de manière cohérente, créent une expérience homogène du site, et renforcent la reconnaissance de la marque.

III) Navigation

L'utilisateur naviguera facilement entre les différentes sections du site grâce à une arborescence de menus intuitive. Voici un aperçu de la navigation :

Page d'Accueil :

- Accueil chaleureux avec le logo du site.
- Le logo doit être placé à 20px du bord supérieur gauche et mesurer 150x80px.

- Bouton de connexion ou d'inscription, redirigeant vers des formulaires distincts.(doit être aligné à droite avec le logo, à 20px du bord supérieur droit et avoir des dimensions de 150x50px.)
- Une fois connecté, redirection vers la nouvelle page d'accueil connectée.



Page de connexion :

- Une option de connexion : demande l'adresse mail et le mot de passe de l'utilisateur (mot de passe non visible).
- Les champs de formulaire auront une largeur de 350px et une hauteur de 40px.
- La distance entre les champs de formulaire sera de 15px.
- Le bouton de soumission aura des dimensions de 150x50px et sera centré en dessous des champs de formulaire avec un espacement vertical de 20px.
- Une option d'inscription : demande l'adresse mail (vérification du format), un mot de passe (doit contenir majuscule minuscule chiffre), le nom, le prénom, un nom utilisateur (unique) et la date de naissance (création de compte impossible pour les mineurs). Une case pour accepter les conditions générales d'utilisation sera placée en-dessous du formulaire.
- s'enregistrer ou se connecter donne un token de connexion à l'utilisateur permettant de l'identifier dans ses futurs actions sur le site

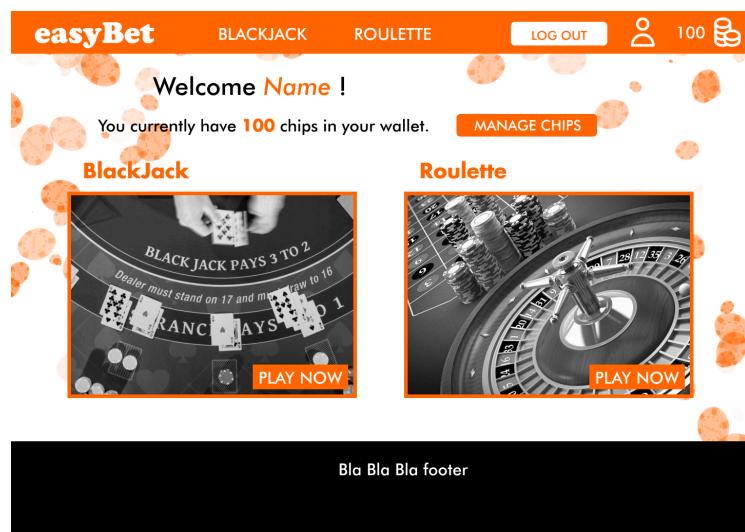


Concernant la base de données utilisateur:

- surnom (unique)
- nom
- prénom
- adresse mail unique (clé primaire)
- mot de passe : 1 majuscule, 1 minuscule et 8 caractères minimum, stocké crypté avec SHA256
- balance de jeton (nombre entier)

Page d'Accueil Connectée :

- Met en avant les jeux proposés et permet d'y accéder facilement en cliquant dessus
- Les images représentant les jeux auront des dimensions de 300x200px et seront espacées de 20px.
- Permet également d'accéder à son compte en cliquant l'icône profil, de se déconnecter en cliquant sur 'log out', ou d'accéder la page permettant d'acheter les jetons permettant de jouer.



Page d'Accueil Connectée - Spécifications de Positionnement

- Vignettes de Jeux :
 - Vignettes de 'BlackJack' et 'Roulette' centrées sur la page, sous la zone de bienvenue.
 - Dimensions des vignettes : 300x200px, espacées de 20px verticalement et horizontalement.
 - Boutons 'PLAY NOW' sur chaque vignette positionnés à 20px du bord inférieur de chaque vignette.
- Bouton 'MANAGE CHIPS' :
 - Situé à droite du solde, aligné verticalement avec le texte du solde.
 - Dimensions de 150x40px, avec un padding interne de 10px pour le texte.

Page de Profil :

- Informations du compte (nom utilisateur, nom, prénom, date de naissance) avec des options pour modifier.
- Possibilité de modifier ses informations personnelles.
- Possibilité de se déconnecter ou de supprimer le compte.
- Accès rapide à la page de gestion des fonds.

Page de Profil - Spécifications de Positionnement:

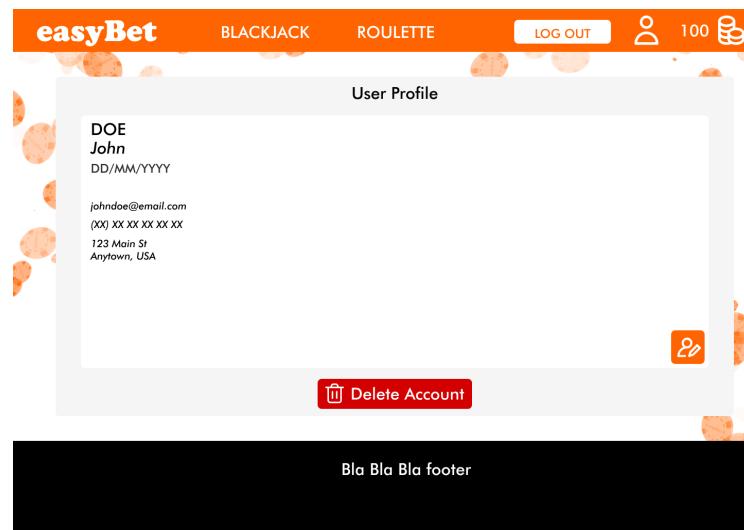
Bloc d'Informations du Profil :

- Centré horizontalement sur la page.
- Largeur de 500px avec un padding interne de 20px.

- Le bloc commence à 150px du bord supérieur de la page.
- Chaque ligne d'information (nom d'utilisateur, email, adresse, etc.) est espacée de 10px verticalement.

Bouton 'Delete Account' :

- Placé en bas au milieu à l'intérieur du bloc d'informations du profil.
- Dimensions du bouton : 150px de largeur par 40px de hauteur.
- Marge de 20px du bord droit et du bas du bloc d'informations du profil.



Page d'Accueil : L'utilisateur accède à la page d'accueil où il est accueilli et voit le logo du site.

Bouton de connexion ou d'inscription : Deux options distinctes sont fournies, permettant aux utilisateurs de se connecter s'ils ont déjà un compte ou de s'inscrire s'ils sont nouveaux.

Formulaire d'inscription : En cliquant sur le bouton d'inscription, l'utilisateur est redirigé vers un formulaire où il fournit une adresse e-mail, un mot de passe (qui doit respecter les critères spécifiés), son nom, prénom et sa date de naissance.

Vérification de la majorité : Le système vérifie que l'utilisateur est majeur en fonction de sa date de naissance. Si l'utilisateur est mineur, son inscription est refusée : un popup s'affichera.

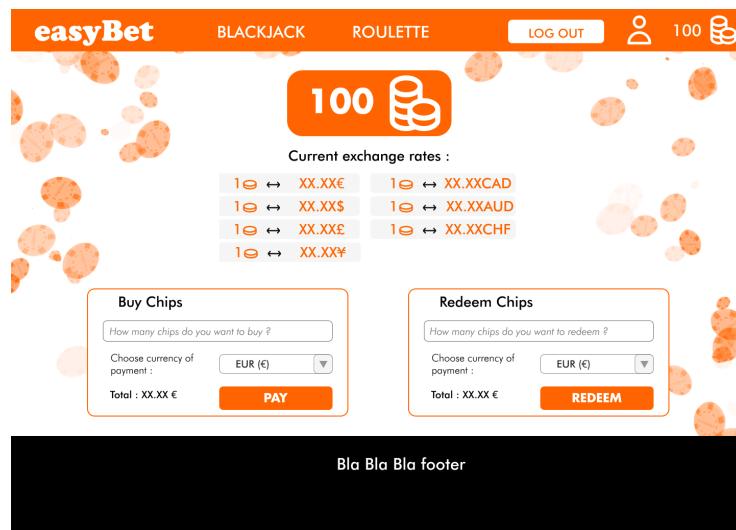
Processus de connexion : Pour les utilisateurs existants, un formulaire de connexion demande l'adresse e-mail et le mot de passe.

Profil utilisateur :

- Nouvelle page d'accueil connectée : Après la connexion, l'utilisateur est redirigé vers une nouvelle page d'accueil.
- Section profil : Une section "Profil" affiche les informations du compte telles que le nom d'utilisateur, la date de naissance, avec des options pour modifier ces détails si nécessaire.
- Déconnexion : Une option de déconnexion est fournie pour permettre aux utilisateurs de se déconnecter en toute sécurité après avoir utilisé le site.

Page de Gestion d'Argent (Ajouter/Retirer) :

- Affichage du nombre de jetons possédés.
- Options pour acheter des jetons dans différentes devises ou convertir ses jetons dans ces mêmes devises.
- Méthodes de paiement disponibles avec redirection vers des pages d'erreur (simulées).



Page de Gestion d'Argent- Spécifications de Positionnement

- Section Solde et Conversion :

- L'affichage du solde (l'icône et le nombre de jetons) centré en haut de la section principale de la page, environ à 100px du bord supérieur.
- Tableau des taux de change directement sous l'affichage du solde, avec un espacement vertical de 20px.
- Chaque ligne du tableau de taux de change espacée de 10px.
- Options d'Achat et de Conversion de Jetons :
 - Deux sections principales : 'Buy Chips' et 'Redeem Chips', alignées côté à côté et centrées horizontalement sur la page.
 - Chaque section a une largeur de 300px et est séparée par un espacement de 40px.
 - Boutons 'PAY' et 'REDEEM' centrés en dessous des options de sélection de devise, avec un espacement vertical de 20px par rapport aux sélecteurs.

Concernant la gestion de l'argent:

- on ne stocke pas de monnaie sur le site, l'utilisateur conserve des jetons du jeu
- les achats se font en direct et la base de donnée est immédiatement mise à jour au moment de l'achat
- on ne peut acheter qu'un nombre entier de jeton dans la limite de 10000 jetons par jour et par dépôt
- on se base sur l'euro puis conversion par rapport à l'euro pour les autres monnaies
- 1 jeton = 1 euro
- convertisseur : on peut utiliser l'api yahoo finance pour récupérer les données en direct". Version gratuite 2000 requêtes/h.
- les 7 plus grosses monnaies les plus utilisés:
- Le dollar américain (USD)
- L'euro (EUR)
- Le yen japonais (JPY)
- La livre sterling (GBP)
- Le dollar australien (AUD)

- Le dollar canadien (CAD)
- Le franc suisse (CHF)
- actualisation toute les minutes en back-end puis données transmises aux clients
- si un client doit valider son achat pendant une actualisation de taux on garde le taux de la minute précédente (conservation du taux de changes 15 sec à partir de la demande de conversion par l'utilisateur pour qu'il puisse valider sa transaction avant paiement), puis 5 min pour payer avec sa CB
- une fois que l'utilisateur à valider, la transaction a lieu
- l'arrondi pour le taux de change se fait au centime supérieur (2 chiffres après la virgule)
- même chose pour retirer les fonds en monnaie
- dépôt maximal journalier de 10 000 jetons

Gestion des conversions en backend :

Le backend sera composé de :

- ChipsMain
- ChipsContrôleur
- ChipsService
- ChipsConversion
- CurrencyConversion

Un ChipsConversion est une classe contenant un type correspondant à un String représentant la monnaie dans laquelle on veut acheter ou vendre le jeton, un entier correspond au nombre de jetons que l'utilisateur souhaite acheter.

Au niveau des types de monnaie, on a :

- EUR : correspond à l'euro, monnaie sur laquelle est adossé le prix d'un chip (1 chip égal à 1 euro)
- USD : dollar US
- CHF : franc suisse
- JPY : yen japonais
- GBP : Livre sterling
- AUD : Dollar australien
- CAD : dollar canadien

Le fichier ChipsControlleur expose les endpoints qui seront appelés par le frontend. Lors de l'appel d'un endpoint, ce fichier renverra vers le Service qui implémentera la logique.

Le fichier ChipsService implémente les méthodes pour les conversions :

- une méthode *getConversion*, qui renvoie un double, qui correspond au prix de n chips dans la monnaie souhaitée

Depuis le frontend, en confirmant les paris l'utilisateur fait un post sur "/Currency" avec un ChipsConversion, sous cette forme :

```
{
  "type": "CHF",
  "nbChip": 3
}
```

On récupère les données dans le backend que l'on déserialize, une API permet de récupérer les différents taux de change par rapport à l'euro. Il suffit juste de multiplier les deux données pour obtenir le prix souhaité.

Exemple de raw à mettre dans le body pour jouer récupérer les prix:

```
{
  "type": "CHF",
  "nbChip": 3
}
```

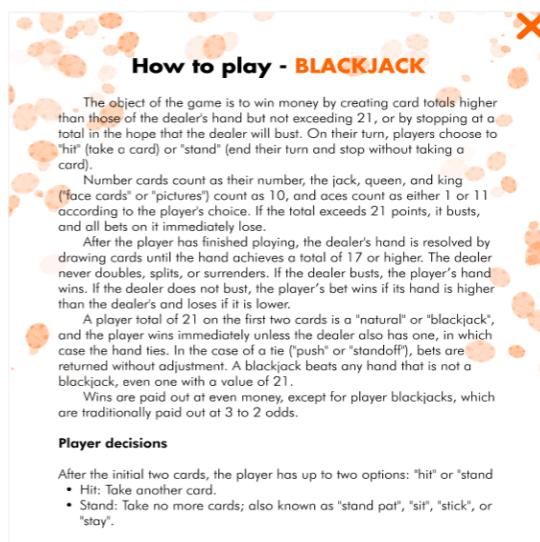
Pour récupérer tous les prix d'un chip dans toutes les monnaies disponibles on peut faire un GET sur "/currencies". Le backend va alors renvoyer un dictionnaire string, double qui associe les différentes monnaies avec leur taux de change par rapport à l'euro soit le prix d'un chip dans toutes les monnaies.

Base de données des dépôts:

- Les dépôts sont stockés dans un autre table contenant pour chaque utilisateur la somme de ses dépôts
- la table est supprimée et recréée tous les jours à 00H00.

Modal d'instruction de jeu :

- avant d'accéder à un jeu
- contient les instruction du jeu en question
- Quand la page de jeu est accédée depuis la page d'accueil, le modal d'instruction est automatiquement ouvert par dessus, et il faut accepter les règles pour accéder au jouer
- Mise minimale: 1 jeton
- Mise maximale: 10 000 jetons
- Mise vérifiée côté frontend



IV) Déroulement des Jeux

Page de Jeu Roulette :

- Options pour définir la mise en jetons et placer les paris.
- Bouton "Spin" pour lancer la roue.
- Popup d'avertissement si l'utilisateur quitte la page pendant le jeu.

- Résultat affiché en tant que modal avec mise à jour du solde et possibilité de rejouer, sinon retour à l'accueil

Page de Jeu Roulette - Spécifications de Positionnement

- Zone de Jeu :
 - La table de roulette est centrée sur la page avec un espace de 50px du haut de l'interface utilisateur du jeu.
 - Dimensions de la table de roulette : environ 60% de la largeur de la page pour une visualisation optimale sur les écrans de bureau.
 - Les boutons pour la sélection de jetons ('Place your bets') sont situés en dessous de la table de jeu, avec un espacement de 10px entre les boutons.
- Bouton 'SPIN' :
 - Situé en dessous de la zone de sélection de jetons, centré par rapport à la table de roulette.
 - Dimensions du bouton : 150x50px pour une interactivité claire.
- Popup d'Avertissement :
 - En cas de sortie de la page, un popup d'avertissement apparaîtra, recouvrant environ 80% de la largeur de la page et centré verticalement et horizontalement.
 - Le popup aura un padding interne de 20px pour le texte d'avertissement.
- Résultat du Jeu :
 - Lors de l'affichage des résultats, un modal apparaîtra au centre de l'écran avec un espace suffisant pour afficher le nouveau solde et les options de jeu suivantes.
 - Le modal des résultats aura une largeur de 400px et une hauteur suffisante pour contenir tout le texte et les boutons nécessaires, avec un padding interne de 20px.



Instructions et Options de Mise :

- How to play : Des instructions claires sur le fonctionnement de la roulette.
- Sélection de la mise : L'utilisateur choisit les jetons de certaines valeurs et les placent sur la table.

Placement des Paris :

- Paris intérieurs : Interface permettant de sélectionner des paris sur des numéros individuels ou des combinaisons spécifiques.
- Paris extérieurs : Options pour les paris sur des groupes de numéros (manque/passe, rouge/noir, pair/impair, douzaines, colonnes).
- Graphiques visuels : Représentation graphique de la table de roulette pour une sélection facile.

Activation du Bouton "Spin" :

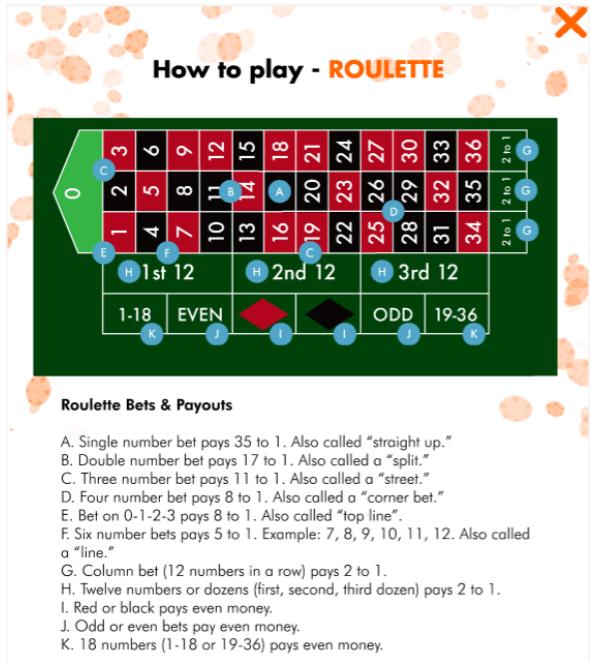
- Dégrisement automatique : Le bouton "Spin" se dégrise dès que l'utilisateur a placé ses paris.
- Avertissement Popup : Un popup d'avertissement persistant prévient l'utilisateur de ne pas quitter la page pendant la rotation de la roue.

Animation de la Roulette :

- Visuel attrayant : Animation fluide et visuellement attrayante de la roue qui tourne.
- Attention : Pendant la rotation, un message indique que l'utilisateur peut perdre sa mise s'il quitte la page.

Algorithme et règle de la roulette:

- le nombre tiré est un tirage aléatoire réalisé dans le backend



Ensemble des paris

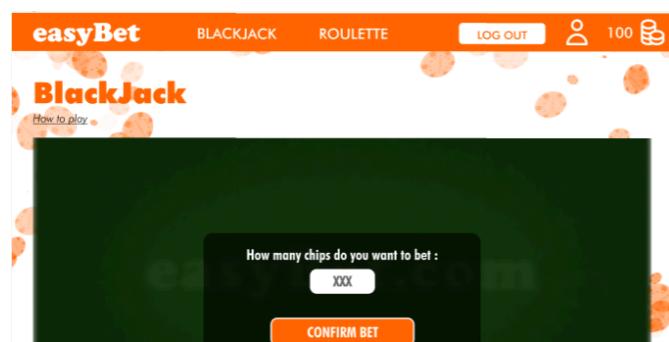
Paris intérieurs

Les paris intérieurs sont appelés comme tels, car ils sont placés sur une, plusieurs ou des sections entières de numéros sur la table.

1. **Le pari plein** : Le pari se fait sur un ou plusieurs numéros séparés. Le paiement est de 35 contre 1 — le rendement le plus élevé du jeu.
2. **Le pari à cheval** : Vous misez sur deux numéros proches du tracé (vertical ou horizontal). Le paiement est de 17 contre 1.
4. **Le pari carré** : Vous pariez sur quatre numéros adjacents qui convergent sur un bord de la table, formant un motif carré. Le paiement est de 8 contre 1.

Paris extérieurs

1. **Manque/passe** : Vous pariez sur n'importe quel numéro en fonction de la série qui vous semble appropriée : manque (1 à 18) ou passe (19 à 36). Le paiement est de 2 contre 1.
2. **Rouge ou noir** : Vous pariez sur n'importe quel numéro en fonction de la couleur qui vous arrange : rouge ou noir. Le paiement est de 2 contre 1.
3. **Pair ou impair** : Vous pariez sur n'importe quel nombre pair ou impair. Le paiement est de 2 contre 1.
4. **Douzaines** : Vous pariez sur n'importe quel numéro inclus dans l'un des trois types de douzaines. Les groupes et leurs numéros respectifs sont les suivants : Première douzaine : 1 à 12. Deuxième douzaine : 13 à 24. Troisième douzaine : 25 à 36. Le paiement est de 3 contre 1.
5. **Colonnes** : Vous pariez sur n'importe quel numéro en fonction de son placement dans l'une des trois colonnes verticales de la table de paris, qui sont : 1er groupe : 3 à 36. 2e groupe : 2 à 35. 3e groupe : 1 à 34. Le paiement est de 3 contre 1.



La roue tourne et pointe vers un numéro. (Un numéro possède une couleur et une position sur le tapis unique).

Le joueur gagne la somme des gains associées à chacune des mises.

Roulette en backend :

Le backend sera composé de :

- RouletteMain
- RouletteControlleur
- RouletteService
- RouletteBet

Un **RouletteBet** est une classe contenant un type correspondant à un String représentant le type de pari posé que l'on va détailler juste après, une value correspond à une liste de integer que l'on va détailler juste après et un amount correspond à un Integer qui représente la mise sur ce pari.

Au niveau des types de pari, on a :

- inside : correspond au pari Single (sur une case), Split (sur deux cote à cote sur le tapis), Street (sur 1 ligne du tapis), Corner (0-1-2 ou 0-2-3), Sixline (à cheval sur deux lignes)
→ Pour les paris typés “Inside”, les values correspondent aux numéros du tapis qui satisfassent ce pari. Ainsi le payOff de cette mise correspond à 36 / nombre de values dans la liste.
- evenOdd : Correspond au pari Pair/Impair, les values sont 0 si pair (Even) et 1 si impair (Odd)
- column : correspond à une des colonnes du tapis : les valeurs correspondent au reste de la division euclidienne du numéro par 3, donc la colonne contenant le 1 a une valeur de 1, la colonne contenant le 2 a une valeur 2, la colonne contenant le 3 a une valeur de 0.
- dozen : correspond au 1er, 2ème et 3ème douzaine. La première douzaine (de 1 à 12) a une valeur de 0, la 2eme (de 13 à 24) a une valeur de 1, la 3eme (de 25 à 36) a une valeur de 2.
- highLow : Low correspond aux cases entre 1 et 18 inclus et a pour valeur 0 et High correspond aux cases entre 19 et 36 et a pour valeur 1.
- blackRed : correspond à la couleur des cases, les valeurs : 0 pour Red et 1 pour Black. Les numéros de chaque couleur sont stockés en dur car ils ne correspondent pas à une logique mathématique.

Le fichier **RouletteController** expose le endpoint `/roulette/bet` qui sera appelé par le frontend.

Puis on se connecte à la base de données et on fait uniquement un seul appel à la base de données dans lequel on enlève la mise de joueur et on ajoute le gain qu'il vient de faire. Avec cette méthode, si quelque chose ne se passe pas comme prévu, soit on retire la mise et on ajoute les gains soit on ne fait rien. Comme ça, pas de problème de bug.

Lors de l'appel de ce endpoint, ce fichier renverra vers le Service qui implémente la logique dans la méthode *playRoulette*.

Le fichier **RouletteService** implémente les méthodes pour le déroulement du jeu :

- *getResults*, qui prend en paramètres une liste de rouletteBet, et renvoie une liste de 2 éléments : la position (où la bille va atterrir sur la roulette, déterminée de manière aléatoire avec un random entre 0 et 36), et le payoff du joueur, selon s'il a parié ou non sur la position de la bille finale. Cette méthode fait appel à la suivante : *getPayOff* pour chaque pari placé par le joueur.
- *getPayOff* : prend en paramètres une rouletteBet (un pari placé par le joueur) et la position finale de la bille. Renvoie le gain du joueur, qui est calculé selon le type de pari placé. Selon le type, elle appelle une des méthodes suivantes :
 - *getPayOffInside*
 - *getPayOffColumn*
 - *getPayOffDozen*
 - *getPayOffHighLow*
 - *getPayOffOddEven*
 - *getPayOffColors*

Ces méthodes sont implémentées pour suivre la logique de calcul des gains, selon les différents types de paris, comme expliqués plus haut.

- *playRoulette*. C'est cette méthode qui est appelée par le endpoint du contrôleur. Elle va d'abord appeler *getResults*, pour faire tourner la roulette et obtenir le résultat du joueur. Ensuite, elle appelle *updatePlayerChips*, pour modifier la quantité de jetons du player, selon son gain (et donc sa mise).
- *updatePlayerChips* : cette méthode fait un appel à la table Player de la base de données, pour (en une seule requête vers la base de données) lui retirer sa mise et lui ajouter son gain. Le fait de ne réaliser qu'une seule transaction sur la base de données permet d'éviter des problèmes si la mise du joueur est retirée avant que son gain ne soit ajouté (le service pourrait rencontrer une erreur entre ces deux transactions, et le joueur ne toucherait donc pas son gain).

Voici un exemple de requête qui peut être envoyée par le front sur le endpoint */roulette/bet* :

```
[  
 {  
   "type": "Single",  
   "value": [5],  
   "amount": 2  
 }
```

On récupère l'ensemble des paris dans le backend que l'on deserialize, le frontend s'assure que le joueur possède assez de jetons.

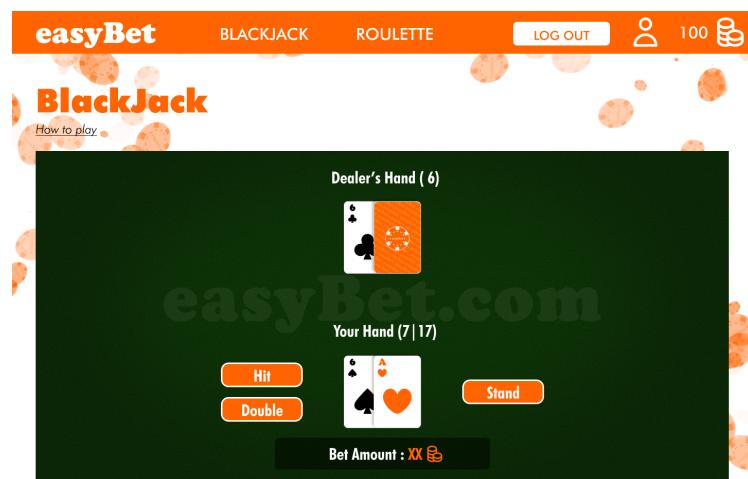
Le token du joueur sera situé dans le header du endpoint.

Exemple de raw à mettre dans le body pour jouer à la roulette :

```
[  
{  
  "type": "blackRed",  
  "value": [0],  
  "amount": 10  
}, {  
  "type": "oddEven",  
  "value": [0],  
  "amount": 10  
}  
]
```

Page de Jeu Blackjack :

- Options pour définir la mise en jetons.
- Bouton "Jouer" pour commencer une nouvelle partie.
- Distribution des cartes pour le joueur et la maison.
- Possibilité de doubler la mise.
- Résultat affiché avec mise à jour du solde et option pour rejouer, sinon retour à l'accueil.



BlackEnd BackJack :

Le backend sera donc organisé comme suit :

- Card
- Deck
- Game
- EndGame
- BlackJackMain
- BlackJackController
- BlackJackService
- BlackJackConfig

Mise en place :

Une classe *Card* :

- un objet *Card* est défini, qui possède 2 attributs : *Rank* et *Suit*. Un jeu de cartes (entier, 52 cartes), est défini par un objet *Deck*, qui est construit en énumérant sur les *Rank* et *Suit*.
- La classe *Card* contient une méthode *getValue()* permettant d'obtenir la valeur de la carte pour le jeu (valeurs allant de 1 à 10, le as étant fixé à 1 initialement).

Une classe *Deck* :

- contient les méthodes pour construire un jeu de cartes, pour le mélanger, et pour distribuer une carte.

Une classe *Game*

- contient toutes les données pour l'initialisation du jeu pour un joueur.

C'est-à-dire 2 attributs :

- *List<Card> cards* : contient une liste d'au plus 7 cartes (qui est le nombre de cartes maximale que l'on peut tenir sans dépasser 21), qui seront les cartes du joueur pendant la partie.
- *List<Integer> scores* : cette liste calcul le score associé à la pile de cartes *cards* au fur et à mesure.

Le fichier *BlackJackConfig* contient les constantes nécessaires au déroulement du jeu (valeur maximale d'une main, nombre de cartes à distribuer, ...)

Le fichier *BlackJackController* expose les endpoints qui seront appelés par le frontend. Lors de l'appel d'un endpoint, ce fichier renverra vers le Service qui implémentera la logique.

- */blackjack/start-game* : initialise la partie : renvoie 2 *Game*:
 - le jeu "maximal" du joueur, c'est-à-dire toutes les cartes qu'il est possible pour lui de tirer (s'arrête lorsqu'il dépasse 21). Le score est calculé après chaque tirage de carte.
 - le jeu du dealer : donne le score initial (la première carte), et le score final (le dealer retire toujours une carte tant qu'il ne dépasse pas 17).
- */blackjack/getPayoff* : prend en paramètre la mise du joueur et sa main finale, et renvoie son gain, selon les règles du BlackJack.

Le fichier *BlackJackService* implémente les méthodes pour le déroulement du jeu :

- une méthode *start-game*, qui renvoie une liste de 2 *Game*, contenant les parties pour le joueur et pour le dealer. Cela permet d'initialiser complètement le jeu en un seul appel par le frontend. Cette méthode appelle les méthodes suivantes (pour une meilleure lisibilité du code) :
 - une méthode *getCardDeal*, qui renvoie les 2 tas de cartes totales : 11 cartes pour le joueur et 11 pour le dealer. Cette méthode est appelée avant d'initialiser les jeux
 - une méthode *initPlayerGame*, qui initialise le jeu du joueur, qui contient le "jeu maximal" (11 cartes max sont possibles : 4 as, 4 deux et 3 trois dépassent 21). Cette méthode renvoie la liste des cartes à tirer, ainsi que les scores correspondants, i.e. tous les scores possibles jusqu'à dépasser 21).
 - une méthode *initDealerGame*, qui fait jouer le dealer. Le dealer continue à tirer une carte tant que son score est en dessous de 17. Cette méthode renvoie donc également la liste des cartes que le dealer va tirer, mais ne à la différence de *initPlayerGame*, il ne renvoie que le score initial (donc la

value de la première carte) et le score final (soit il a dépassé 17 et il s'est arrêté, soit il a dépassé 21).

- une méthode *getPayoff*, qui prend en paramètres le pari du joueur ainsi que ses cartes (ou son score) et renvoie son gain, qui est de 0 s'il a perdu.
- une méthode *getScore* : calcule le score d'une main donnée en paramètre. Prend en compte le calcul de la valeur d'un as : un as sera toujours de valeur 11, sauf si le score total dépasse 21 et dans ce cas l'as vaut 1.

La partie démarre lorsque le frontend appelle le endpoint */blackjack/start-game*.

Lorsque le joueur a fini de tirer ses cartes, le frontend fait appel au endpoint */blackjack/getPayoff* pour déterminer les gains du joueur.

Exemple de résultat pour start-game :

```
[  
 {  
   "cards": [  
     {  
       "rank": "SIX",  
       "suit": "SPADES",  
       "value": 6  
     },  
     {  
       "rank": "THREE",  
       "suit": "DIAMONDS",  
       "value": 3  
     },  
     {  
       "rank": "KING",  
       "suit": "HEARTS",  
       "value": 10  
     },  
     {  
   ]  
 }]
```

```
        "rank": "FIVE",
        "suit": "HEARTS",
        "value": 5
    },
],
"scores": [
    9,
    19,
    24
]
},
{
    "cards": [
        {
            "rank": "EIGHT",
            "suit": "HEARTS",
            "value": 8
        },
        {
            "rank": "QUEEN",
            "suit": "HEARTS",
            "value": 10
        }
    ],
    "scores": [
        8,
        18
    ]
}
```

Exemple de requete pour getPayoff :

```
{  
    "playerHand": [  
        {  
            "rank": "TEN",  
            "suit": "HEARTS",  
            "value": 10  
        },  
        {  
            "rank": "SIX",  
            "suit": "SPADES",  
            "value": 6  
        },  
        {  
            "rank": "TWO",  
            "suit": "HEARTS",  
            "value": 2  
        }  
    "dealerHand": [  
        {  
            "rank": "FOUR",  
            "suit": "DIAMONDS",  
            "value": 4  
        },  
        {  
            "rank": "THREE",  
            "suit": "CLUBS",  
            "value": 3  
        }  
    ]}
```

```
"suit": "SPADES",
"value": 3
},
{
"rank": "NINE",
"suit": "SPADES",
"value": 9
},
{
"rank": "EIGHT",
"suit": "HEARTS",
"value": 8
}
],
"bet":10000000
}
```

Instructions et Options de Mise :

- How to play : Des instructions claires sur le fonctionnement du BlackJack.

Configuration de la Partie :

- Sélection des jetons : L'utilisateur choisit le montant de sa mise en jetons.
- Bouton "Jouer" : Démarrer une nouvelle partie de Blackjack.

Distribution des Cartes :

- Cartes du joueur : Deux cartes initiales du joueur sont distribuées face visible.
- Cartes de la maison : Une carte de la maison est visible, l'autre face cachée.

Options de Jeu :

- Doublement de la mise : Possibilité pour l'utilisateur de doubler sa mise en retirant automatiquement une carte.
- Hit and Stand : Options pour demander des "cartes supplémentaires" (hit) ou choisir de "rester" (stand) avec la main actuelle.

Résultats et mise à jour du solde :

- Affichage des résultats : Les résultats s'affichent après que le joueur et la maison aient joué.
- Mise à jour du solde : Si l'utilisateur gagne, son solde est mis à jour avec une indication claire de l'ancien et du nouveau solde.

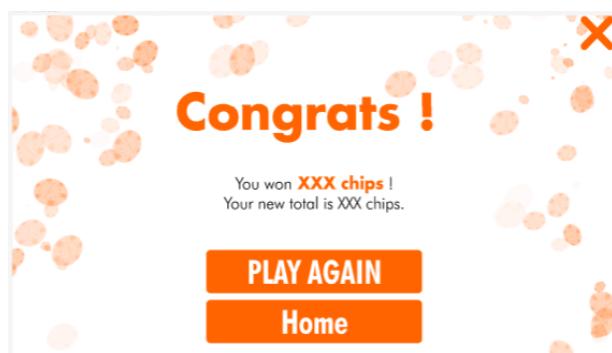
Option pour jouer :

- Bouton "Play Again" : Possibilité de rejouer avec la même mise en un clic.
- Retour à la page d'accueil : Option pour revenir à la page d'accueil après le jeu.

Page de Jeu Blackjack - Spécifications de Positionnement

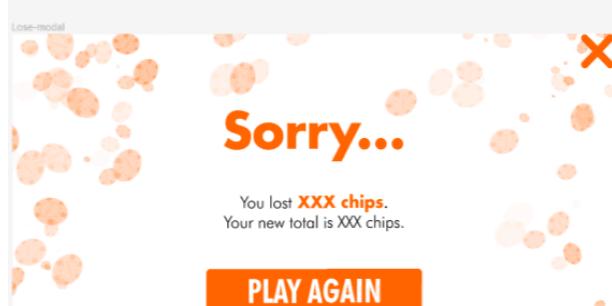
- Zone de Jeu :
 - La zone de jeu est centrée sur la page avec un espace de 80px du haut de la page pour l'entête.
 - La table de jeu devrait occuper une largeur de 70% de la page pour une visualisation claire sur les écrans de bureau.
- Bouton 'Jouer' :
 - Positionné sous la section de sélection de jetons, centré horizontalement par rapport à la table de jeu.
 - Dimensions du bouton : 120x50px, pour une interaction facile.
- Distribution des Cartes :

- Les cartes du joueur sont affichées face visible, espacées de 5px entre elles.
- La première carte de la maison est affichée face visible, la seconde carte est masquée.
- Options de Jeu :
 - Les boutons pour 'Hit' et 'Stand' sont placés sous les cartes, avec des dimensions de 100x40px chacun.
 - En cas de doublement de mise, un bouton 'Double' apparaît à côté de ces options avec la même dimension.
- Résultats et Mise à Jour du Solde :
 - Lors de l'affichage des résultats, un modal apparaît au centre de l'écran.
 - Le modal des résultats a une largeur de 400px et contient le solde mis à jour et un message indiquant le résultat de la main.
- Option pour jouer à nouveau :
 - Un bouton 'Play Again' est présent dans le modal de résultats avec une dimension de 120x50px.
 - Une option de 'Retour à l'accueil' est également disponible dans le modal.



Résultats et Mise à Jour du Solde :

- Affichage des résultats : Les résultats s'affichent après l'arrêt de la roue.
 - Mise à jour du solde : Si l'utilisateur gagne, son solde est mis à jour avec un affichage clair de l'ancien et du nouveau solde.



Option pour Rejouer :

- Bouton "Play Again" : Option pour rejouer avec la même mise en un clic.
- Retour à la Page d'Accueil : Possibilité de retourner à la page d'accueil après le jeu.

Les règles et l'algorithme du black jack:

1. Objectif du Jeu :

- Le but est de battre le croupier en ayant un total de points plus élevé sans dépasser 21.
- Les As peuvent valoir 1 ou 11 points, les cartes de figures (Roi, Dame, Valet) valent 10 points et les autres cartes valent leur valeur nominale.

2. Déroulement du Jeu :

- Chaque joueur reçoit deux cartes initiales face visible, tandis que le croupier en reçoit une face visible et une face cachée.
- Si le joueur obtient un blackJack en 2 cartes et que le croupier ne l'a pas obtenu, alors le joueur gagne 2,5 fois sa mise.
- Si le croupier obtient un blackJack, et que le joueur ne l'a pas obtenu alors le croupier gagne et récupère la mise.
- Le joueur peut demander des "cartes supplémentaires" (hit) pour améliorer sa main. Il peut demander autant de cartes qu'il le souhaite, mais s'il dépasse 21 points, il perd automatiquement (buste) et le croupier récupère la mise.
- Le joueur peut également choisir de "rester" (stand) avec sa main actuelle.
- Lorsque le joueur a pris sa décision, le croupier révèle sa carte face cachée.
- Le croupier doit tirer des cartes supplémentaires jusqu'à ce que sa main atteigne au moins 17 points.
- Le joueur gagne si sa main est plus proche de 21 que celle du croupier sans dépasser 21. Si le croupier dépasse 21, le joueur gagne sa mise.

3. Fin du Jeu :

- Après que tous les joueurs et le croupier aient joué, les mains sont comparées.
- Les joueurs avec un total de points supérieur à celui du croupier sans dépasser 21 gagnent.
- Si le croupier dépasse 21, tous les joueurs restants gagnent.
- En cas d'égalité (push), la mise du joueur est retournée.
- à la fin de chaque tour, le paquet de carte est remis à 0, pour éviter tout comptage

4. Algorithme:

- tirage aléatoire de l'entièreté du jeux de carte (donc création d'un ordre aléatoire des entiers de 1 à 52)
- les cartes sont alors distribuées au joueur dans l'ordre du tirage
- représentation des cartes:
 - i. 0 à 12 -> as à roi de pique
 - ii. 13 à 25 -> as à roi de trèfle
 - iii. 26 à 38 -> as à roi de carreau
 - iv. 39 à 51 -> as à roi de coeur
- le déroulement de la partie est alors traiter côté frontend et le résultat est ensuite envoyé au backend

V) Intéractions entre les différentes entités techniques

Le frontend est réalisé en React.

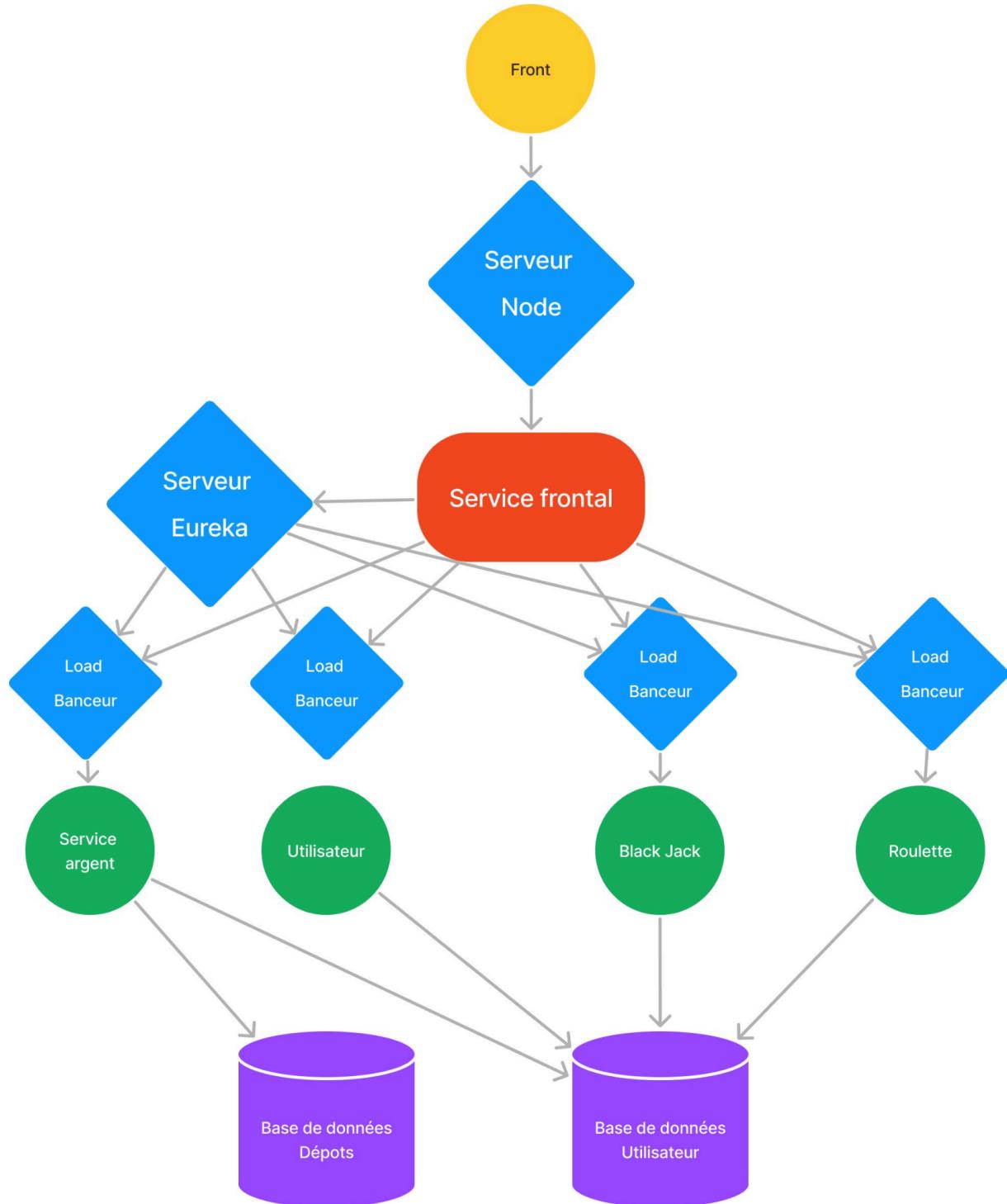
Le backend utilise maven et spring-boot (Java).

Décomposition des services:

- serveur Eureka pour rediriger les requêtes
- un loadbalancer: on veut que toutes nos instances serveurs soient capables de traiter toutes les requêtes, donc il nous faut répartir équitablement toutes celles ci entre les différents serveurs

- des instances serveurs capables de traiter différents services (utilisateur, gestion d'argent, roulette et black jack), et donc d'accéder aux bases de données

Les bases de données sont en PostgreSQL.



VI) API REST

Ports :

- EurekaServer : 8761
- Roulette : 8081
- Chips : 8082
- BlackJack : 8083
- Player : 8084
- Security (Service frontal): 8085

Endpoints :

Tous les appels sont fait sur le service Security, donc sur le port 8085.

Player:

- GET `/allplayers`
 - header : aucun
 - body : aucun
 - return : List de tous les players
- GET `/players`
 - header : token du joueur
 - body : aucun
 - return : le player cherché
- GET `/player/{email}`
 - header : email
 - body : aucun
 - return : le player cherché
- GET `/players/chips`
 - header : token
 - body : aucun
 - return : nombre de chips du player
- POST `/players`
 - header : aucun
 - body : Player
 - return : le player créé
- POST `/players/login`
 - header : aucun
 - body : LoginRequest
 - return : le token
- PUT `/players`
 - header : aucun
 - body : nouveau player (i.e. le player avec les données voulues)
 - return : le player updaté
- PUT `/players/chips`
 - header : token
 - body : nombre de chips

- return : le player dont les chips ont été modifiés
- **DELETE /players**
 - header : token
 - body : aucun
 - return : void

Roulette :

- **POST /roulette/bet**
 - header : token
 - body : List<RouletteBet>
 - return : List<Integer> : le résultat de la roulette et le gain du joueur

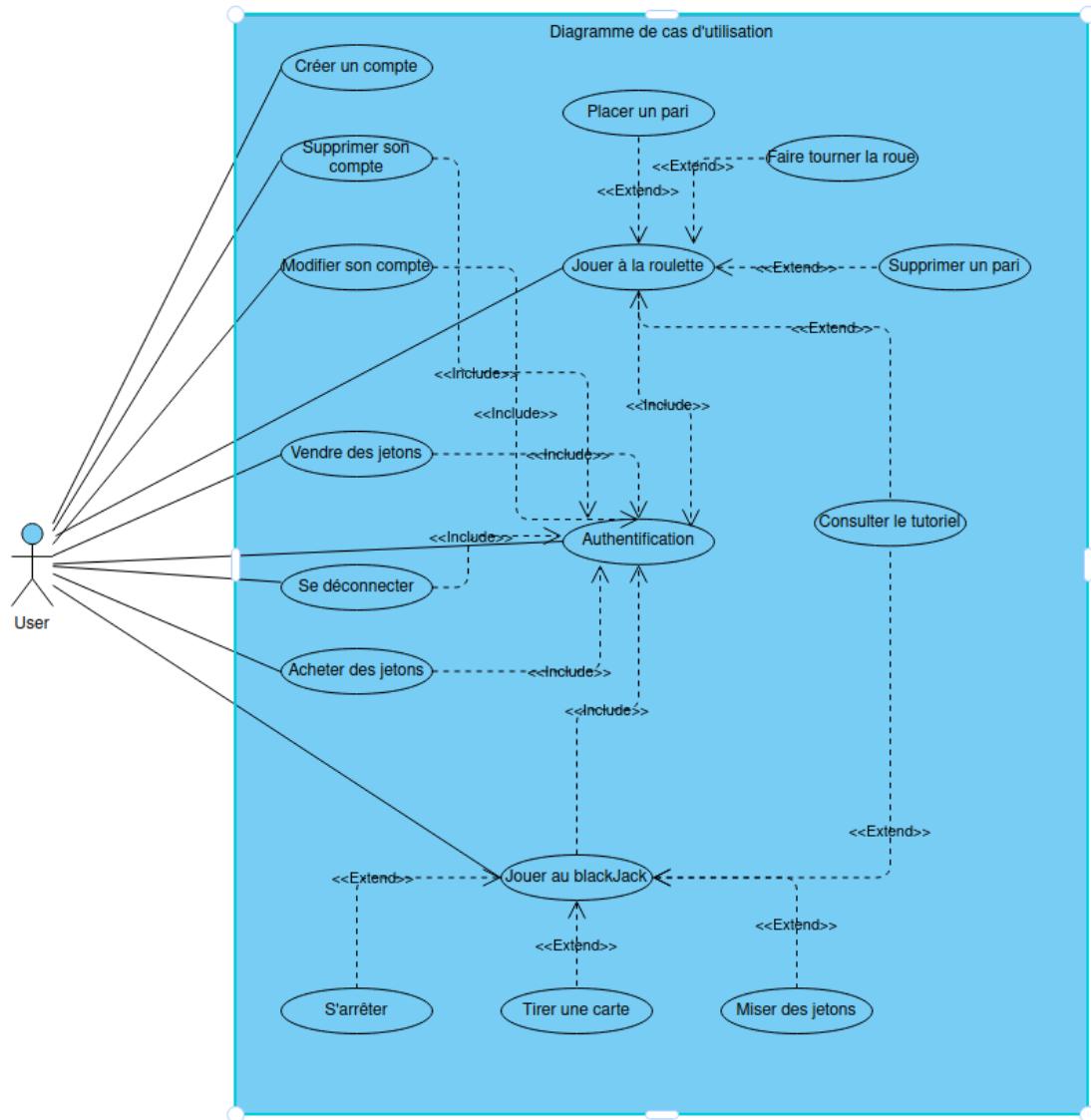
BlackJack :

- **GET /blackjack/start-game**
 - header : aucun
 - body : aucun
 - return : List<Game> : le jeu du joueur et le jeu du dealer
- **POST /blackjack/getPayoff**
 - header : token
 - body : endGame
 - return : double : le payoff du joueur

Chips :

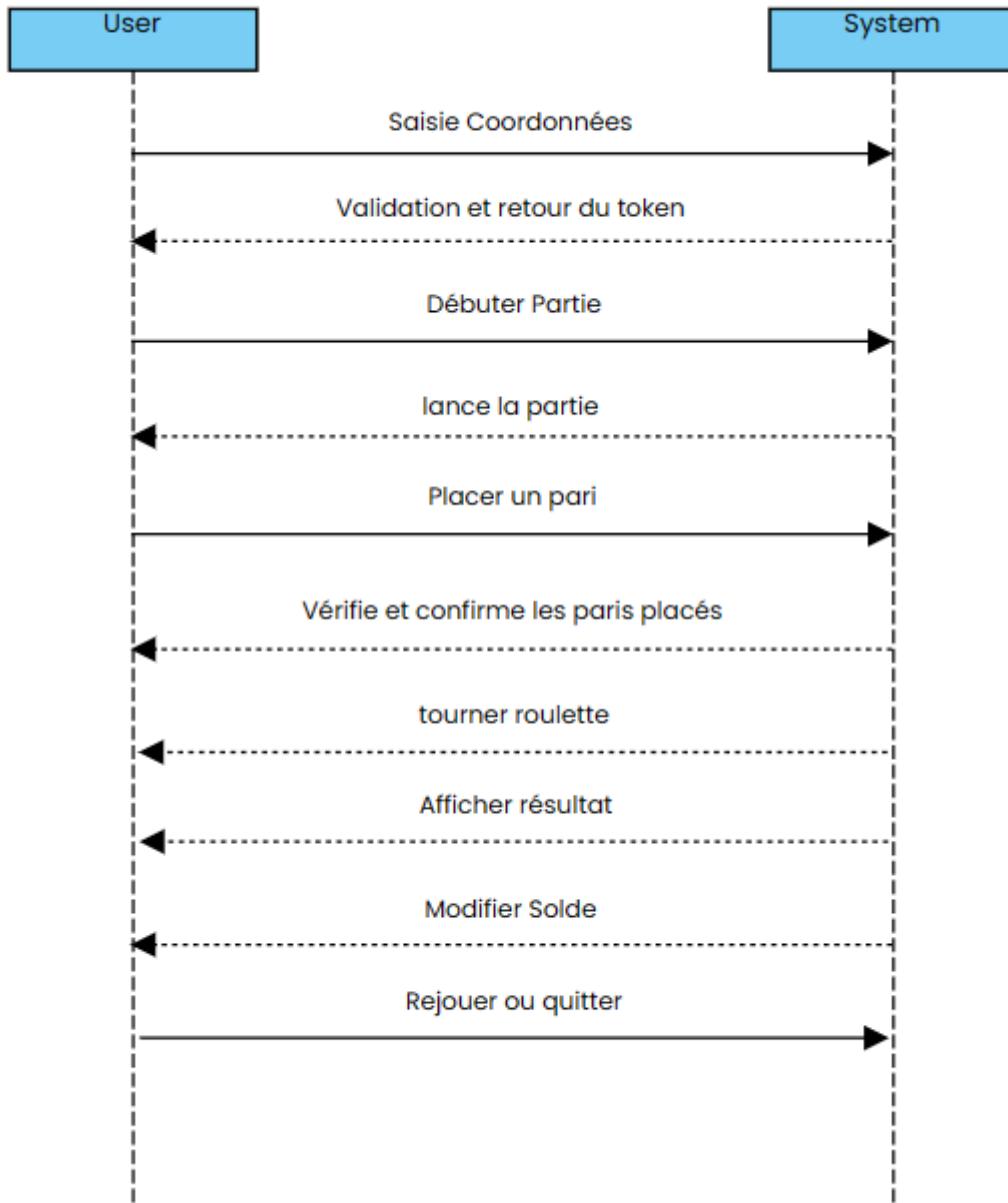
- **POST /getPrice**
 - header : aucun
 - body : chipsConversion
 - return : double
- **GET /getAllPrices**
 - header : aucun
 - body : aucun
 - return : Map<String, Double> : tous les prix associés aux monnaies

VII) Diagramme de cas d'utilisation



VIII) Diagramme de séquence

1) Diagramme de Séquence : Roulette russe



2) Diagramme de Séquence : BlackJack

