

# Projet Programmation Web : Rami

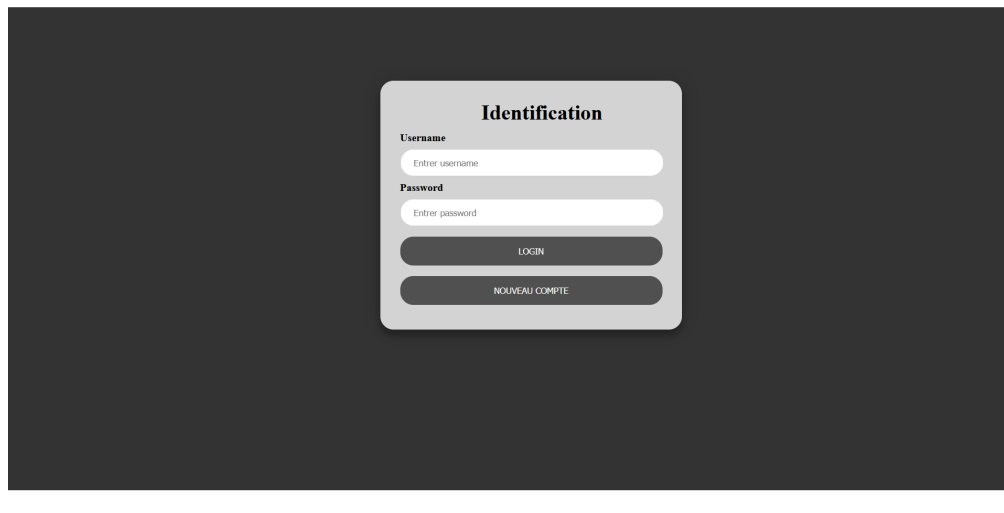
## Membres du groupe

- COMBEAU Thomas
- SIVACOUMAR Arsène

## Présentation du jeu

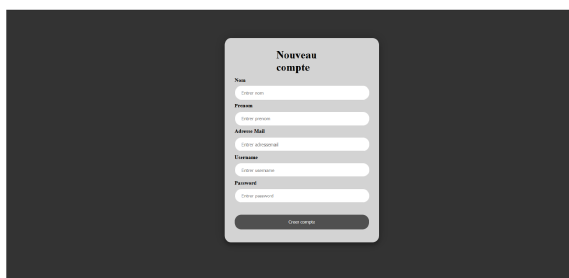
Pour notre projet, nous avons décidé de faire un jeu ressemblant au Rami mais avec nos propres règles et certaines simplifications par rapport au jeu d'origine. Tout d'abord nous avons deux jeux de cartes où on enlève les jokers. Notre jeu se joue à 4 et chaque joueur récupère au début 14 cartes. Le but du jeu est de déposer l'ensemble des cartes et donc de ne plus en avoir dans sa main. Pour pouvoir déposer des cartes, il faut former des combinaisons : des carrés, des brelans ou des suites d'au moins 4 cartes qui se suivent (On ignore les couleurs dans notre jeu), il faut aussi avoir un minimum de point pour déposer la première fois : les valets, dames et rois valent 10 points et les cartes numériques valent leurs propres valeurs. Chaque joueur peut piocher et jeter une carte vers la pioche à chaque tour. Dans le cas où le joueur ne dépasse pas 14 cartes et ne peut pas déposer, il peut, s'il le veut, piocher une carte sans en jeter une.

## Identification, Authentification et interconnexion des joueurs

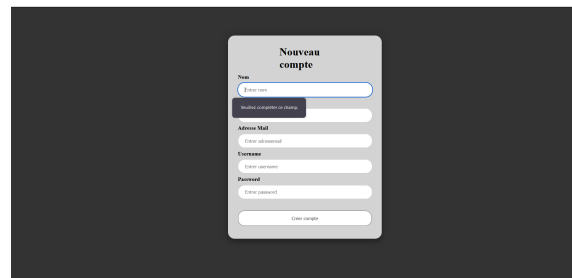


The image shows a login form titled 'Identification' centered on a dark background. The form is a light gray rounded rectangle. It contains two input fields: 'Username' with the placeholder text 'Entrez username' and 'Password' with the placeholder text 'Entrez password'. Below these fields are two buttons: 'LOGIN' and 'NOUVEAU COMPTE'.

Tous les joueurs qui souhaitent jouer à notre jeu doivent s'authentifier afin qu'ils puissent accéder au jeu. La page d'authentification, visible sur la capture ci-dessus, permet aux joueurs d'effectuer 2 tâches distinctes : l'authentification et l'identification. L'authentification du joueur permet à ce dernier d'accéder directement au jeu en indiquant, dans les champs spécifiques, son nom d'utilisateur (Username) et un mot de passe sur cette page d'authentification. Si le joueur ne s'est pas identifié au préalable en créant un compte, les informations le concernant n'existeront pas dans la base de données donc le joueur ne sera pas redirigé vers la page de jeu. Un message d'avertissement sera indiqué afin que le joueur puisse créer un compte. Cette procédure de création de compte utilisateur est possible en appuyant sur le bouton 'NOUVEAU COMPTE' et le joueur non identifié sera redirigé vers une nouvelle page qui est détaillée dans la partie suivante. Lorsque le joueur est bien identifié, un identifiant unique est attribué à ce joueur en accédant au jeu.



The image shows the left side of a 'Nouveau compte' (New account) form. It has a title 'Nouveau compte' and a 'Nom' field. Below it are fields for 'Prénom', 'Adresse postale', 'Adresse Mail', 'Téléphone', 'Pays', and 'Mot de passe'. At the bottom is a 'Créer compte' button.



The image shows the right side of a 'Nouveau compte' (New account) form. It has a title 'Nouveau compte' and a 'Nom' field. Below it are fields for 'Adresse postale', 'Adresse Mail', 'Téléphone', 'Pays', and 'Mot de passe'. At the bottom is a 'Créer compte' button.

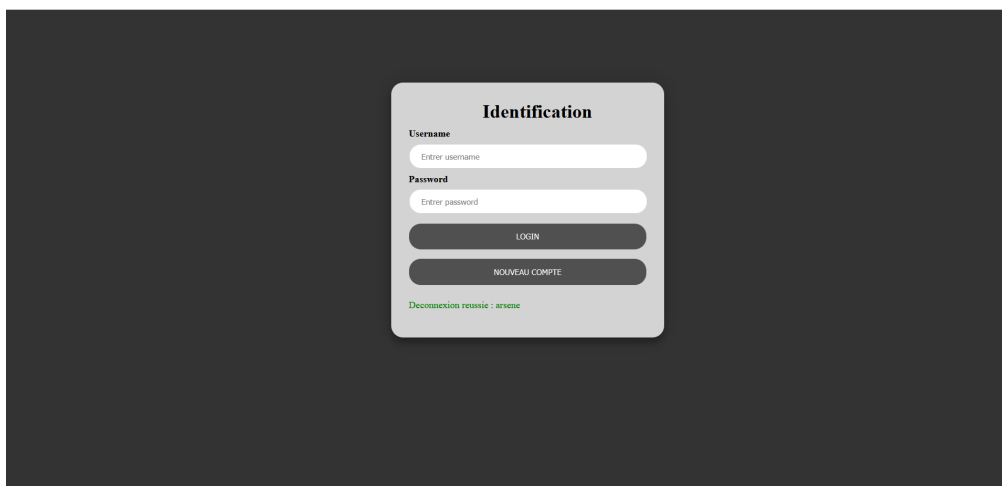
Chaque utilisateur n'ayant pas de compte doit créer un compte en fournissant les informations demandées sur la page d'identification visible sur les captures ci-dessus. Toutes les informations nécessaires doivent obligatoirement être fournies par le joueur. Si au moins un champ est vide lors de la validation de la création de compte en cliquant sur le bouton 'créer compte', alors un message indiquera au

joueur de remplir les champs manquants. Cette fonctionnalité est possible grâce à l'attribut 'required' dans les inputs présents dans le fichier nouveauCompte.php). Sachant que toutes les adresses email et tous les noms d'utilisateur doivent être uniques dans la base de données, un joueur ne pourra pas fournir un Username ou une adresse email déjà existante. Afin de sécuriser le mot de passe fourni, ce dernier est haché puis stocké dans la base de données.



The image contains two side-by-side screenshots of a web application interface. The left screenshot shows a dark-themed background with a light gray rounded rectangle containing the 'Identification' form. This form has two input fields labeled 'Username' and 'Password', each with placeholder text 'Entrez username' and 'Entrez password' respectively. Below these fields are two buttons: 'LOGIN' and 'NOUVEAU COMPTE'. At the bottom of the form, there is a green message: 'Connexion réussie : avertissement'. The right screenshot shows a similar dark-themed background with a light gray rounded rectangle containing the 'Nouveau compte' form. This form has four input fields: 'Pseudo', 'Adresse e-mail', 'Mot de passe', and 'Confirmer le mot de passe', each with placeholder text. Below these fields are two buttons: 'Créer compte' and 'Retour à la page d'identification'.

Lorsqu'un compte utilisateur est créé correctement, l'utilisateur sera redirigé vers la page de connexion avec un message indiquant au joueur que son compte est créé. Ainsi, le joueur peut se connecter et accéder au jeu. La page de connexion avec ce message est visible sur la capture gauche. Lorsque le joueur fournit une adresse email ou un nom d'utilisateur déjà existant, un message d'avertissement indiquera l'échec de la création de son compte. Tant que le joueur ne fournira pas une adresse email et un nom d'utilisateur non existant dans la base de données, la création de son compte ne sera pas possible. Le message indiquant l'échec de création de compte est visible sur la capture de droite.

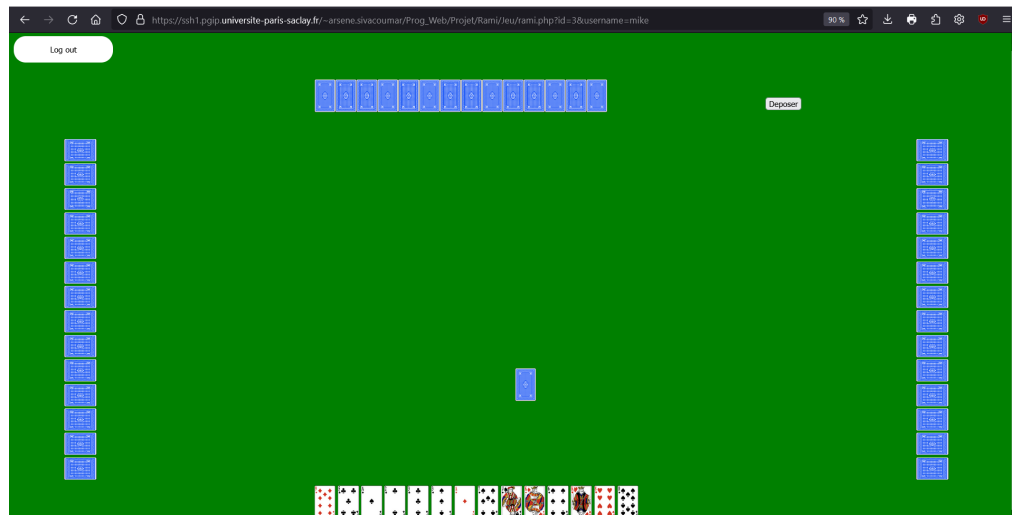


The image shows a single screenshot of the 'Identification' form on a dark background. The form is a light gray rounded rectangle. It has two input fields: 'Username' with placeholder 'Entrez username' and 'Password' with placeholder 'Entrez password'. Below the fields are two buttons: 'LOGIN' and 'NOUVEAU COMPTE'. At the bottom of the form, there is a green message: 'Déconnexion réussie : avertissement'.

Lorsqu'un joueur souhaite quitter le jeu, il doit appuyer sur le bouton 'Logout' présent sur l'interface du jeu. Après cette action, ce joueur sera déconnecté du jeu et sera redirigé vers la page de connexion avec un message indiquant qu'il a bien été déconnecté. L'attribut 'connected' dans la base de données sera mis 0 (signifiant

que le joueur possédant ce nom d'utilisateur est déconnecté du jeu). Ce message est visible sur la capture ci-dessus.

## Moteur du jeu



La première étape de notre moteur de jeu était de réussir à distribuer 14 cartes pour tous les joueurs et considérer le reste comme la pioche. Pour ce faire, on a eu l'objectif de mettre en place un fichier JSON qui permet de mettre sur le serveur la distribution des cartes courantes pour chaque joueur et la pioche, ce fichier se nomme « results.json ». Donc après l'authentification, quand un joueur arrive sur le plateau de jeu, il se voit distribuer son paquet de cartes grâce à l'appel ajax qui envoie l'ID du joueur vers le fichier « début.php ». Dans ce fichier, on regarde si l'ID du joueur est le premier joueur, si c'est le cas alors on va créer remplir le fichier « results.json » qui est donc vide, si le joueur n'est pas le premier alors on renvoie l'ensemble des données qui contient le paquet du joueur en question et on s'occupera d'afficher les cartes correspondantes pour ce joueur. Donc quand c'est le premier joueur qui se connecte, on crée un tableau qui correspond à un jeu de cartes avec toutes les cartes (sauf joker) dont chaque carte est représentée elle-même par un tableau de deux éléments : la valeur de la carte (as, 2, trois, ... dame, roi) et la couleur de la carte. Une fois le jeu créé on le double pour avoir deux jeux de cartes. Puis on mélange le jeu de cartes.

Une fois le jeu de carte mélangé on va distribuer dans 4 tableaux différents 14 cartes, ces tableaux correspondront aux paquets des 4 joueurs et le reste des cartes est la pioche qu'on met encore une fois dans un autre tableau. Ensuite, on met l'ensemble de ces données dans un autre tableau qu'on met lui-même dans le fichier « results.json » puis on envoie aussi ces données vers le client qui a fait l'appel ajax.

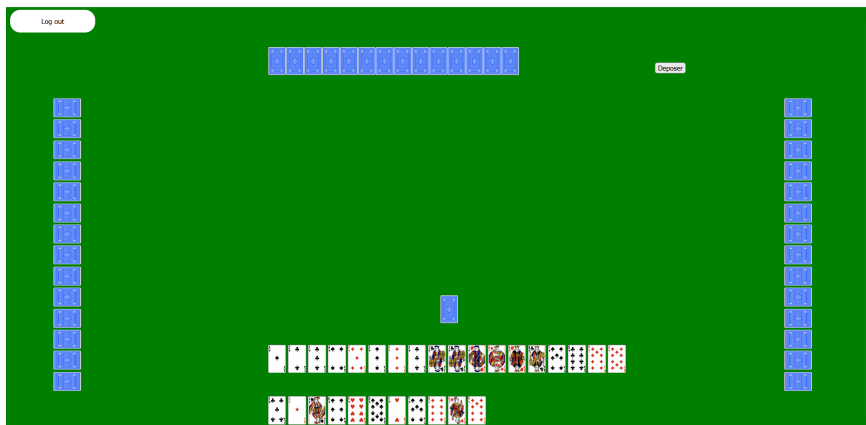
Au niveau, du client on va regarder quel tableau des données envoyées correspond au paquet du joueur, puis pour chaque carte dans ce tableau, on va rajouter un code html qui permet d'afficher cette image (les images dans le serveur sont renommées de façon à ce que les cartes soient facilement trouvables), et sur cette image nous

avons mis un event d'objet JQuery pour lancer une fonction JS quand on clique sur la carte. Chaque paquet de joueur se trouve dans une balise DIV différente et chaque joueur ne voit que ses cartes, les cartes des autres joueurs sont retournées. Aussi, les cartes du joueur possèdent un ID qui permet de savoir leurs emplacements dans le serveur, par exemple la 3ème carte du joueur 2 possède l'ID 12 (décalage de 1 car on commence à 0, dans le serveur la carte se trouve donc dans le tableau 1 et l'emplacement de tableau numéro 2). Enfin, nous ajoutons le nom de la carte (ex : 3coeur) en « name » dans la balise image.

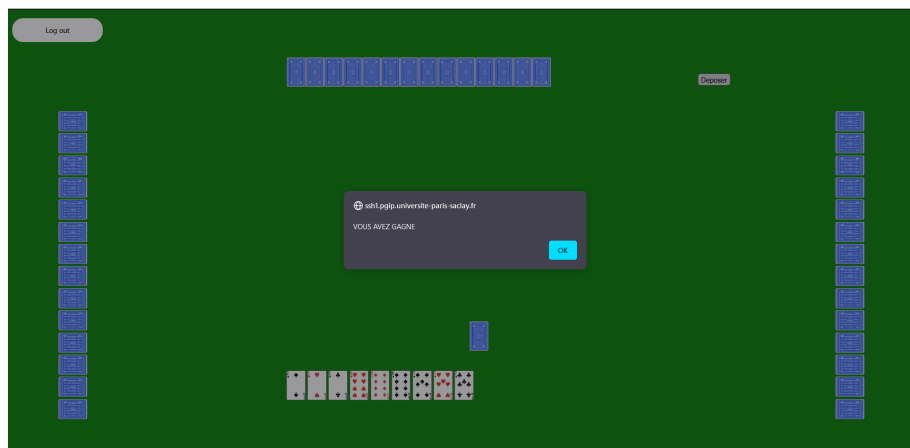
Ensuite, le but du joueur est d'obtenir dans son jeu de cartes des combinaisons (carrés, brelans ou suites de 4 minimum), pour cela à chaque fois que c'est son tour, le joueur peut piocher puis jeter une carte. Chacune de ces actions se font grâce à des appels ajax. Jeter une carte se déclenche quand on clique sur la carte en question et piocher se déclenche quand le joueur clique sur la pioche. Un joueur qui possède moins de 14 cartes peut piocher sans jeter de cartes s'il le souhaite.

Quand un joueur jette une carte en cliquant dessus, un appel ajax envoie l'ID de la carte vers le fichier « versPioche.php », ce fichier permet de mettre à jour dans le serveur que la carte sélectionnée par le joueur est supprimée de ses cartes et se retrouve à la fin de pioche, comme expliqué plus haut, l'ID de la carte nous permet de connaître son emplacement dans le serveur. Au niveau du client, on supprime l'élément lié à l'image de la carte, puis on rééquilibre les ID des autres cartes du joueurs (ex : le joueur 1 jette sa 6<sup>ème</sup> carte alors les cartes au-dessus cette carte perdent 1 sur l'emplacement dans leurs ID, la carte 7 aura donc l'ID -> 05 maintenant), cela nous permet de garder une similarité avec ce qui se passe dans le serveur et graphiquement.

Maintenant quand un joueur pioche une carte en cliquant sur la carte retournée au milieu, il se passe l'action inverse, un appel ajax envoie l'ID du joueur vers le fichier « piocher.php » puis on enlève la première carte du tableau représentant la pioche sur le serveur et on la met en dernier dans le tableau des cartes du joueur (toujours dans le fichier « results.json »), enfin on renvoie cette carte vers le client à la fin de l'appel ajax. Une fois la carte récupérée, on l'affiche dans le paquet de carte du joueur de la même façon qu'au début.



Quand le joueur veut maintenant déposer ses cartes car il a atteint le nombre de points suffisant par ses combinaisons (51), il peut cliquer sur le bouton « Déposer ». Ce bouton, encore une fois, déclenche un appel ajax qui envoie l'ID du joueur vers le fichier « `deposer.php` ». Dans ce fichier, on récupère le tableau de carte du joueur à partir de « `results.json` » et son nombre de points dans le fichier « `points.json` » si il avait déjà posé des cartes avant (si ce n'est pas le cas, les points sont initialisés à 0). Ensuite le fichier contient plusieurs algorithmes pour calculer le nombre de points et récupérer les cartes engendrant des combinaisons. D'abord, la première étape a été de trier les cartes de manière à ce qu'elles suivent l'ordre d'un jeu de carte (1,2,3, ..., dame, roi). Puis on regarde d'abord si les cartes du joueur possèdent des cartes faisant des carrés ou des brelans, si c'est le cas les points du joueur augmentent de respectivement 4 et 3 fois la valeur de la carte. Puis on récupère ces cartes dans un tableau et on les supprime dans le tableau du joueur. Ensuite, on cherche à savoir si le joueur possède des suites dans le reste des cartes, encore une fois si c'est le cas on augmente les points du joueur de la somme des valeurs de la suite, puis on récupère ces cartes dans un tableau et on les supprime du tableau des cartes du joueurs. Enfin si les points du joueurs dépassent 51, on met à jour sur le serveur le tableau du joueurs en enlevant les cartes faisant ces combinaisons puis on envoie vers le client la liste de ces cartes et les points du joueur mis à jour (qu'on a mis à jour dans le fichier « `points.json` aussi »). Si les points ne dépassent pas 51 points, on ne fait aucune mise à jour et on ne renvoie rien. Côté client, on dépose devant les cartes du joueurs, les cartes dont il s'est débarrassé avec les combinaisons, puis on supprime ces cartes dans son paquet. Les cartes déposées possèdent en ID leur nom (ex : `id = '3trèfle'`).



Si un joueur dépose ses dernières cartes alors une fenêtre apparaît pour annoncer sa victoire.

### Conclusion:

Pour conclure, ce projet a été un vrai défi à relever car il demandait d'aller chercher beaucoup de choses et de bien comprendre les notions utilisées pour trouver les problèmes puis les résoudre. Cependant, à cause d'un problème au niveau de l'utilisation des sessions, on n'a pas pu implémenter un système de tour donc on a dû rebrousser chemin et arrêter cette idée. Donc, en perspective d'amélioration, on peut imaginer un système de tour complet et un mode de communication efficace entre les joueurs comme un chat. Aussi, notre jeu peut être amélioré en se rapprochant encore plus du vrai jeu du Rami, par exemple un joueur pourrait déposer ses cartes pour compléter des suites qui sont dans ses cartes déposées.

Ce projet nous a donc permis de développer des connaissances et des compétences dans une branche du monde du web particulièrement en parcourant les notions d'Ajax, JQuery et Sessions.