**CAHIER DES CHARGES**

Projet TPT

Par

Rakotohanitriaina Maharo Rivomahefa

Ranarivelo Voahangy Antsa

Lala Thong Sang Irchad

**Table des matières**

[1. Description générale des besoins 2](#_Toc79937325)

[2. Définition des applications 3](#_Toc79937326)

[2.1. Spécification des exigences 3](#_Toc79937327)

[2.1.1. Les exigences fonctionnelles 3](#_Toc79937328)

[2.1.1.1. Les acteurs de niveau fonctionnel 3](#_Toc79937329)

[3. Conception 33](#_Toc79937330)

[3.1. Contraintes techniques 33](#_Toc79937331)

[4. Tests du système logiciel 42](#_Toc79937332)

# Description générale des besoins

Le projet consiste à créer un site Web et une application mobile de pari en ligne, permettant donc aux utilisateurs d’effectuer des paris en ligne. On développera aussi une application lourde ( client lourd ) qui permettra aux administrateurs du site d’en gérer le contenu.

Le projet se compose de :

* Une interface client, ou le parieur pourra :
* Parier
* Acheter des jetons
* Voir l’historique de ses transactions
* Voir les matchs à venir, terminés et en cours

Dans ce projet, l’utilisateur utilisera des jetons en guise de monnaie pour effectuer ses paris. Il devra donc acheter des jetons sur le site et nous supposerons pour la suite du projet que l’achat se fera via carte bancaire ou paiement par Mobile money (moyen de paiement le plus courant à Madagascar). Nous n’avons pas intégré de fonctionnalité de paiement en ligne mais nous avons néanmoins simulé le solde (argent) de l’utilisateur, que nous verrons plus loin. Lorsque l’utilisateur aura gagné un pari, il sera notifié. Le site permet aussi de voir sur une carte , les points de vente / échange de jetons.

Une application lourde, qui sera l’interface principale de gestion du site web et de la majeure partie de son contenu.

Les administrateurs du site pourront y gérer les contenus concernant les matchs, les équipes, les paris etc... Nous y avons intégré une fonctionnalité pour démarrer les matchs, car effectivement, dans un cas réel, les matchs réels doivent être récupérés via une api dédiée à cet effet, mais l’inconvénient est de devoir attendre qu’un match commence en temps réel. Une fonctionnalité de distribution de gains permet ensuite de trouver et récompenser les utilisateurs ayant misé juste.

Une interface administrateur web, qui sera dédiée aux administrateurs qui assureront la gestion partielle du site.

Cette interface web permettra à l’administrateur de voir globalement sous forme de dashboard, certaines statistiques générées par l’interface client (nombre de pari des trois derniers mois, nombre d’achat de jetons, etc.…). Il offre aussi la possibilité de gérer la localisation des points de vente/ échange de jetons.

Une application mobile qui reprendra les fonctionnalités de l’interface client web.

Les cibles du site et de l’application mobile sont les personnes majeures de Madagascar souhaitant parier sur un ou plusieurs matchs de foot.

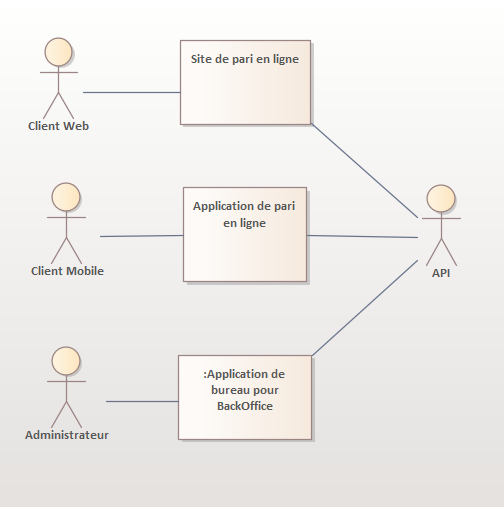
# Définition des applications

# Spécification des exigences

# Les exigences fonctionnelles

# Les acteurs de niveau fonctionnel

**Les acteurs humains**

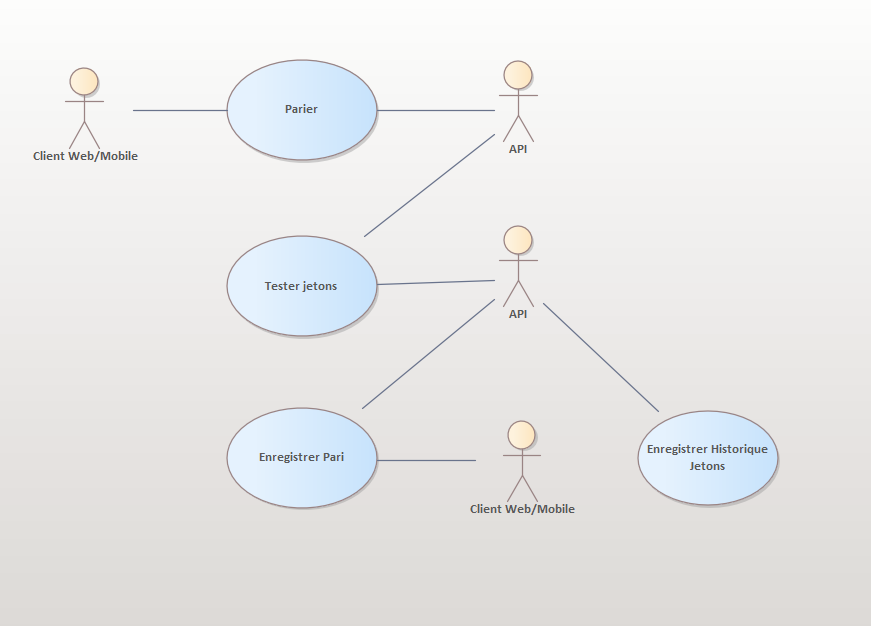


* + - 1. **Les cas d’utilisation**

|  |  |
| --- | --- |
| Cas d’utilisation | Application |
| Parier | Application web / mobile |
| Historique paris | Application web / mobile |
| Historique transactions | Application web / mobile |
| Achat jetons | Application web / mobile |
| Liste matchs | Application web / mobile |
| Voir match | Application web / mobile |
| Top 5 Matchs | Application web |
| Historique matchs d’une équipe | Application web |
| Modification d’informations utilisateur | Application web |
| Récupération de mot passe | Application web |
| Vente de jetons | Mobile |
| Lancer un match | BackOffice |
| Connexion Administrateur | BackOffice |
| Affichage Liste Equipe | BackOffice |
| Ajouter nouvelle Equipe | BackOffice |
| Modifier Equipe | BackOffice |
| Affichage Liste Match | BackOffice |
| Ajouter Nouveau Match | BackOffice |
| Affichage Liste Regles | BackOffice |
| Affichage Liste Paris | BackOffice |
| Recherche Paris par IdUtilisateur | BackOffice |
| Recherche Paris par nom Utilisateur | BackOffice |
| Affichage Liste Utilisateur | BackOffice |
| Recherche Historique d'un Utilisateur par id d’un utilisateur | BackOffice |
| Affichage Liste Match à venir | BackOffice |
| Recherche Historique d'un Utilisateur par nom d'un utilisateur | BackOffice |
| Ajout point de vente | BackOffice Web |
| Liste Point de vente | BackOffice Web |
| Dashboard | BackOffice Web |

* + - * 1. **Pari sur site web**

Ce cas d’utilisation décrit comment un client fait un pari, depuis la connexion sur le site jusqu’à la confirmation de son pari.



**Acteurs** : Client Web, Système de paiement (banque fictive), Administration

**Déroulement normal :**

Le cas d’utilisation débute quand le client se connecte. En effet un utilisateur non connecté a accès à toutes les fonctionnalités sauf valider un pari, voir son profil, voir les historiques transactions et les paris effectués.

L’utilisateur arrive donc sur l’écran d’accueil.

***Mise***

L’utilisateur sélectionne le match sur lequel il veut parier puis mise sur le/les évènements de son choix. Il entre le montant de sa mise dans l’écran qui apparaitra et pourra voir son pari dans son panier après validation du montant. Tant qu’il n’a pas validé ses mises, l’utilisateur pourra retirer un « coupon » (mise) de son panier.

***Validation pari***

Pour engager sa mise et donc valider son pari, le client doit être connecté. En cliquant sur le bouton « valider », l’écran de connexion apparaitra. Il se connectera avec son email et son mot de passe, et pourra alors valider son pari. Après validation, il n’y aura plus moyen d’annuler une mise. Une fois le pari validé, l’API va vérifié que le paiement a été accepté après vérification du solde de l’utilisateur (banque fictive) , puis va incrémenter son solde en jetons de la quantité de jetons achetée et décrémenter son solde « bancaire » du montant total de la mise.

**Déroulements alternatifs :**

Erreur de paiement

Si le solde en banque de l’utilisateur est insuffisant, un message d’erreur apparaitra à l’écran.

La validation ne se fera donc pas.

* + - * 1. **Pari sur application mobile**

Ce cas d’utilisation décrit comment un client fait un pari, depuis la connexion sur l’application jusqu’à la confirmation de son pari.

**Acteurs** : Client Mobile, Système de paiement (banque fictive), Administration

**Déroulement normal :**

Le cas d’utilisation débute quand le client se connecte ensuite il est redirigé vers la page d’accueil.

***Mise et validation pari***

L’utilisateur clique sur le bouton « Liste des matchs » pour pouvoir voir les matchs après il sélectionne le match sur lequel il veut parier puis mise sur le/les évènements de son choix.

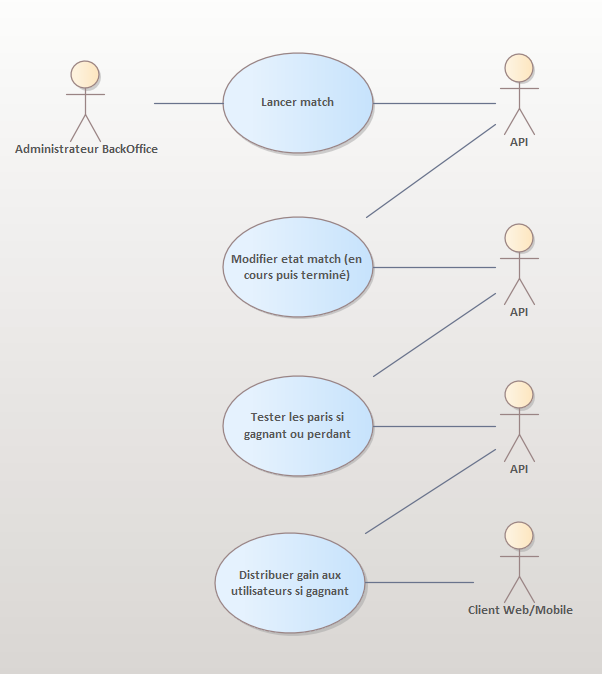
Un petit affichage s’affichera pour qu’il puisse entrer la mise de son pari et pour valider le pari l’utilisateur n’a qu’à cliquer sur le bouton « Valider » et son pari sera ajouter à sa liste de pari.

**Déroulements alternatifs :**

Si le solde de l’utilisateur est insuffisant pour la transaction alors un message d’erreur s’affichera sur l’écran

* + - * 1. **Lancer matchs**

Ce cas d’utilisation décrit comment un administrateur lance un match, depuis la connexion sur le logiciel jusqu’à la répartition des gains de parieurs.



**Acteurs** : Administrateur

**Déroulement normal :**

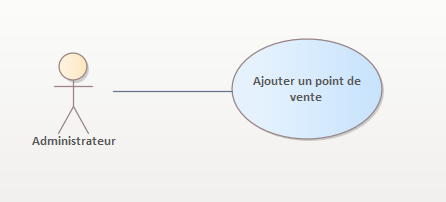
On commence par la connexion de l’administrateur, il faut une connexion car les informations ainsi que les actions dans le logiciel sont seulement destinées aux administrateurs. L’administrateur entrera alors, son pseudo ainsi que son mot de passe pour arriver à l’écran d’accueil du logiciel qui sera la liste de toutes les équipes dans le site.

Il lui faudra donc aller dans la liste des matchs à venir et sélectionné un des matchs pour que ce dernier commence. Le match en question sera donc lancé, c’est à dire, son état va passer de “à” a “terminé” et les statistiques du match seront remplies aléatoirement mais logiquementt vu qu’on veut accomplir c’est la simulation. Un match aura 6 statistiques en tout à savoir : les scores, le nombre de corners et la possession des 2 équipes. Le logiciel va alors récupérer tous les paris sur ce match s’il y en a, et vérifier un par un si les paris correspondent aux statistiques du match. Si un pari correspond, cela veut dire que le parieur a bien deviné et il recevra son gain, ce gain équivaut à la mise du parieur multiplier par la cote sur laquelle le parieur à miser.

Une notification sera alors envoyée à l’utilisateur pour lui informer de son gain et le logiciel enregistrera cette information dans l’historique des jetons. L’administrateur aura alors une annonce sur son écran indiquant que le match a bien été lancé et la liste de départ s’actualisera.

* + - * 1. **Ajout point de vente**

Ce cas d’utilisation décrit comment un client fait un pari, depuis la connexion sur le site jusqu’à la confirmation de son pari.



**Acteurs** : Administrateur

**Déroulement normal :**

Le cas d’utilisation débute quand l’utilisateur se connecte. Après avoir entré le pseudo et le mot de passe, l’utilisateur arrive sur la page d’accueil qui est le dashboard. Dans le menu latéral de gauche, il clique sur « mapping » puis le sous menu «  ajout point de vente », faisant apparaître la page montrant une carte et un formulaire à remplir.

***Ajout***

En cliquant sur le lieu voulu situé sur la carte, un marqueur apparaîtra et les coordonnées du lieu s’inscriront dans les champs correspondants du formulaire. L’utilisateur n’entrera qu’un point de vente à la fois. Il modifiera ensuite l’adresse de ce point de vente dans le formulaire et validera son ajout. Si l’ajout a bien été effectué, un message de succès apparaitra. Il pourra ensuite vérifier cette nouvelle entrée dans la liste des points de vente en cliquant sur « liste points de vente » dans le menu latéral.

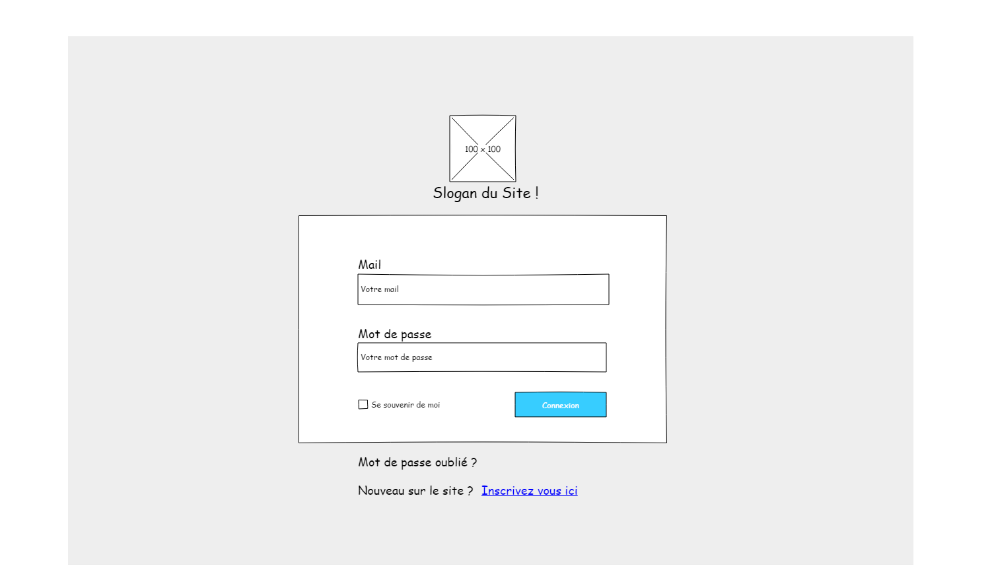
**Déroulements alternatifs :**

Si une erreur se produit, un message d’échec apparaîtra et l’utilisateur devra recommencer sa saisie.

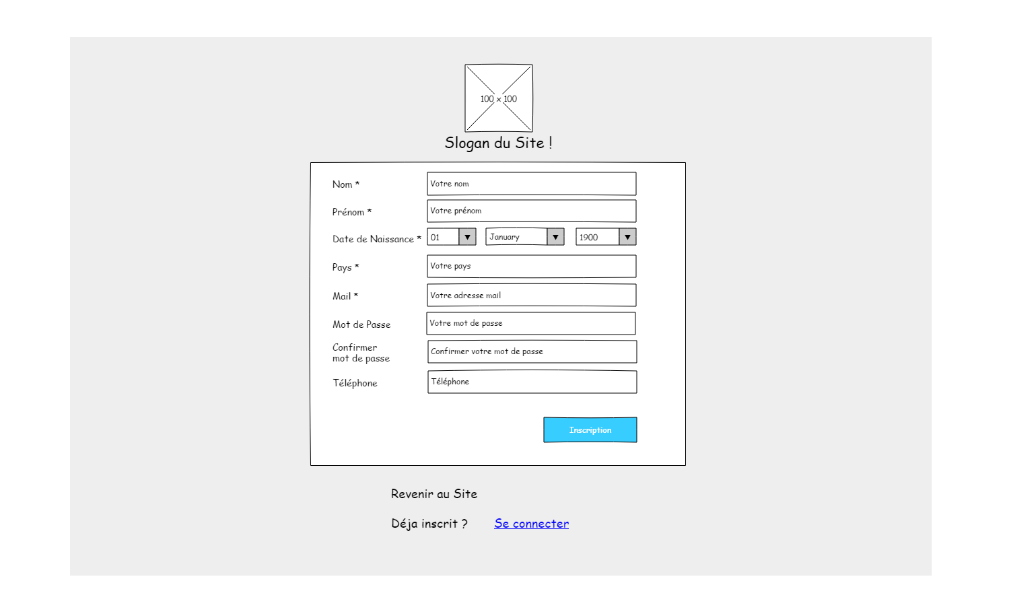
* + 1. **Définition détaillée des interfaces de l’application avec ses acteurs**
       1. **Définition détaillée de l’IHM**

**Client Web :**

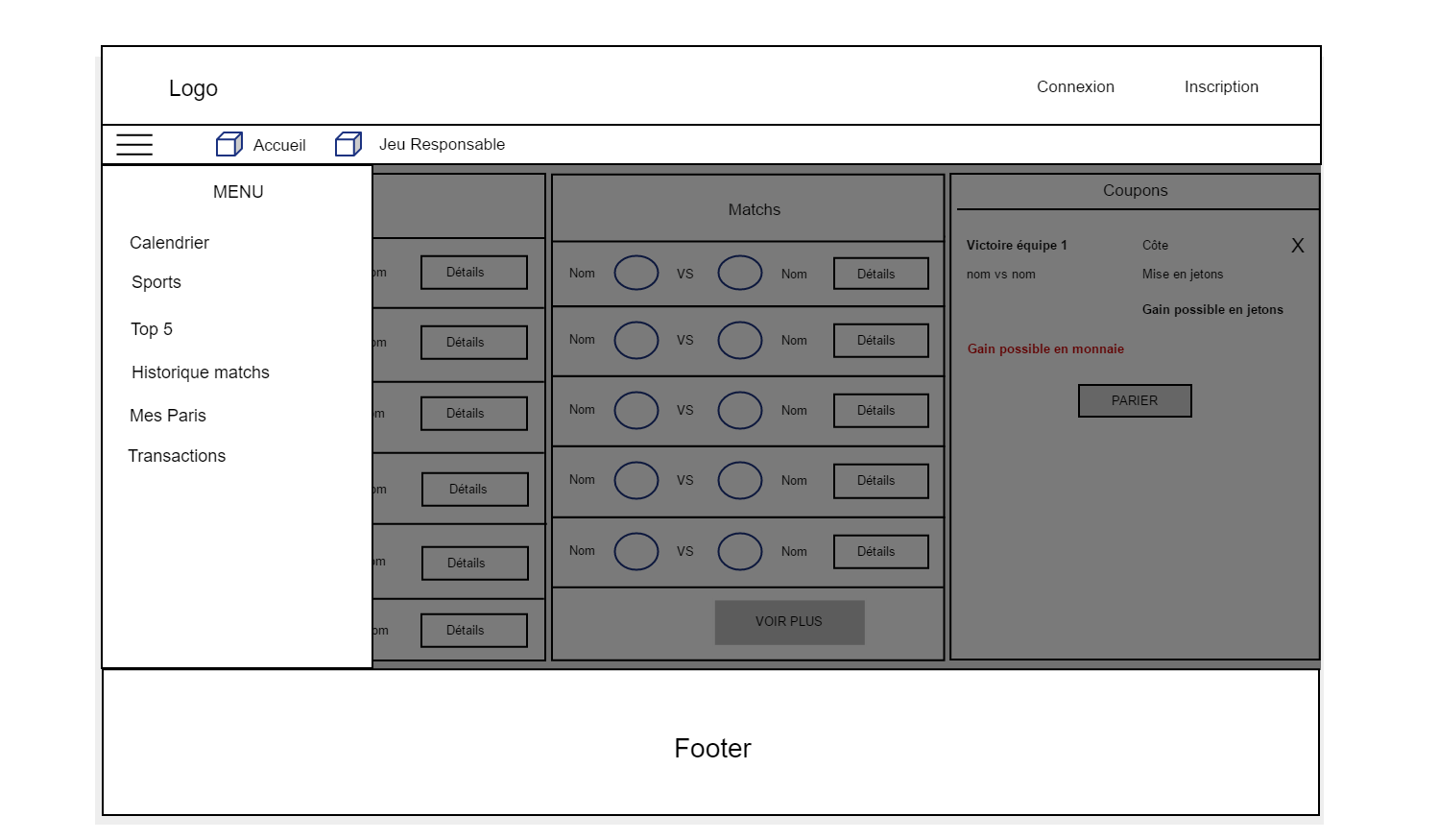
**Login Utilisateur :**

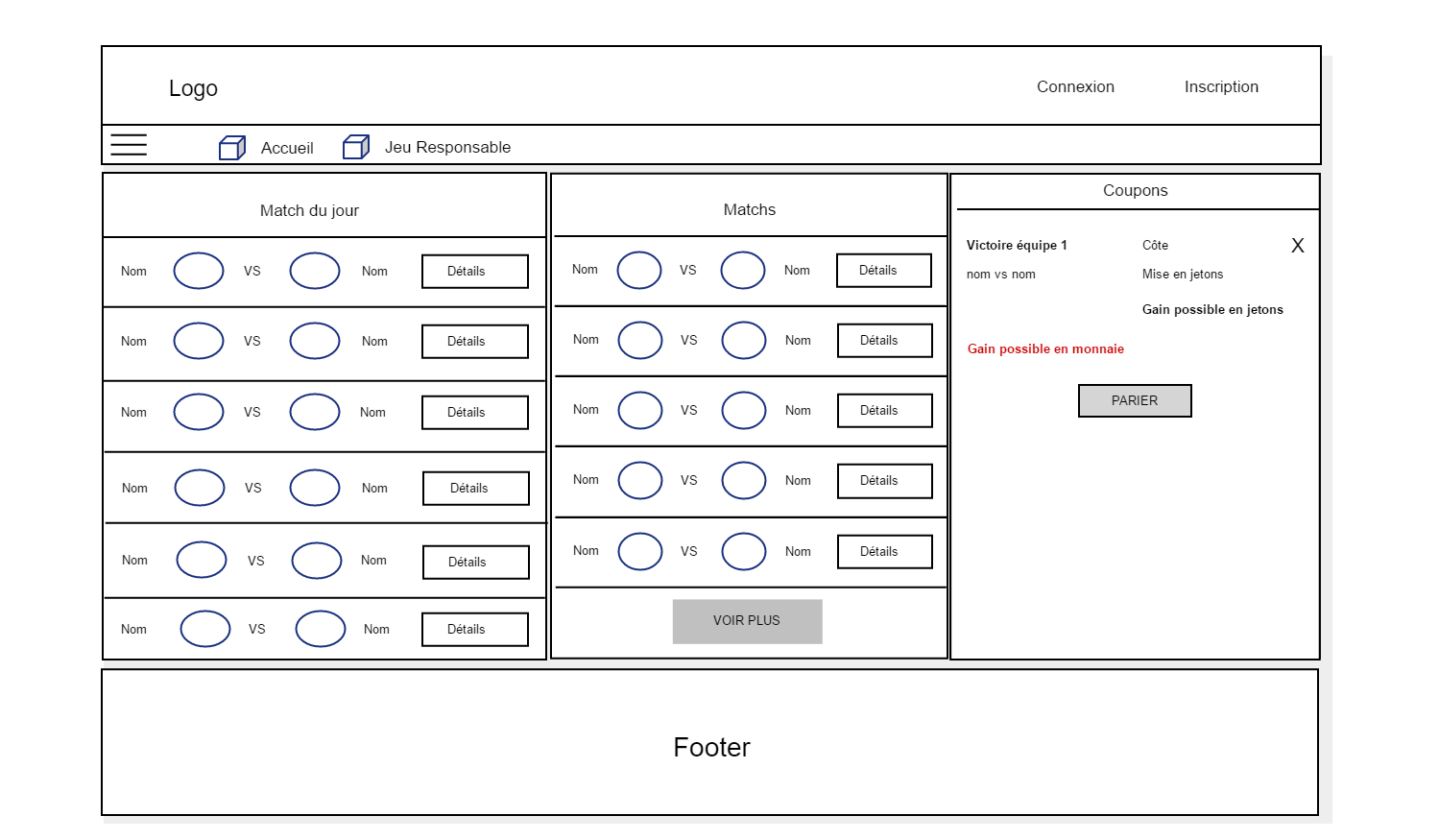


**Inscription utilisateur :**



**Page accueil , sans connexion de l’utilisateur.**

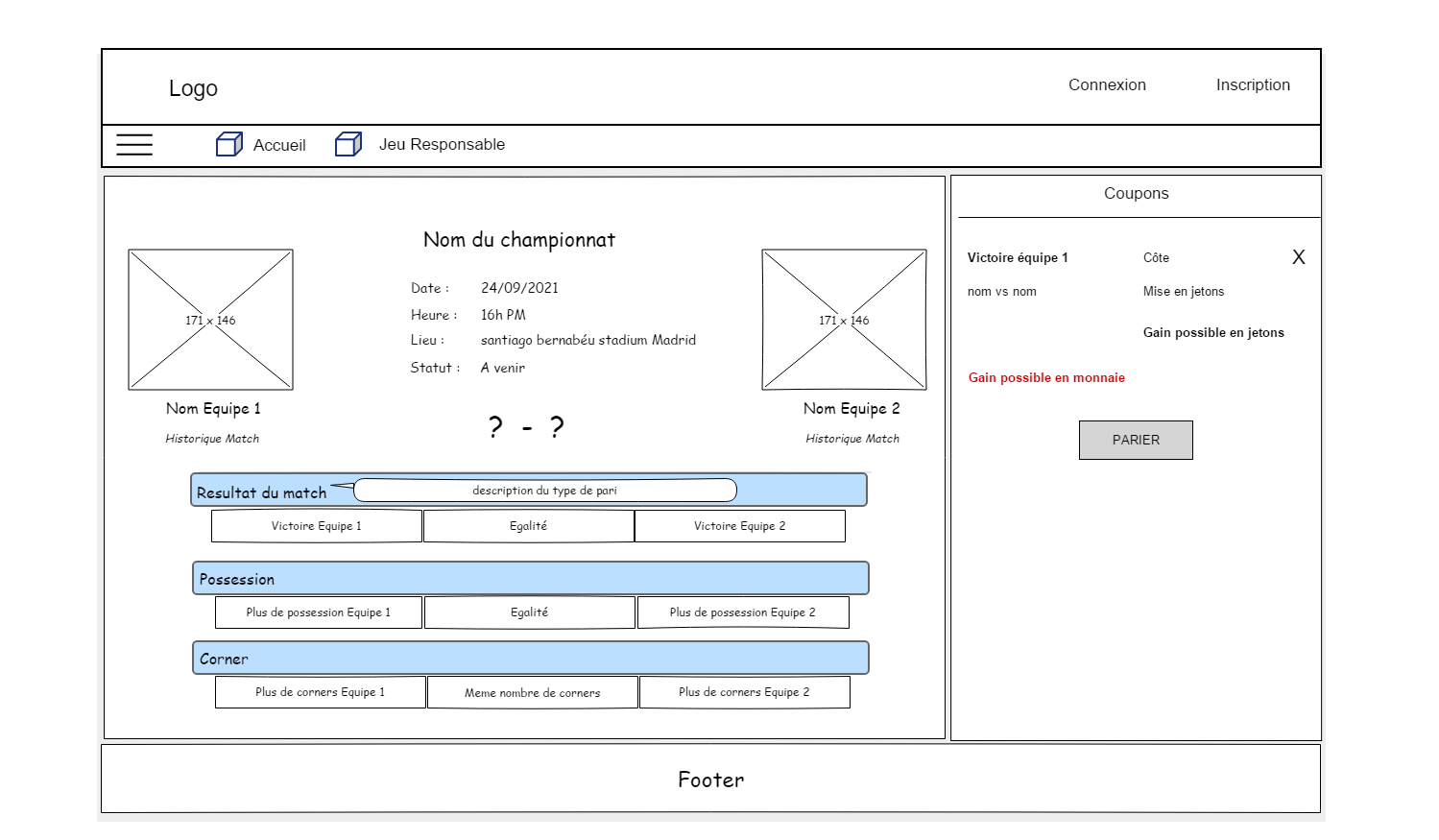




En cliquant sur « détails » nous aboutissons à la page de détails du match sélectionné :

**Page de détails match ( Fiche match ) :**

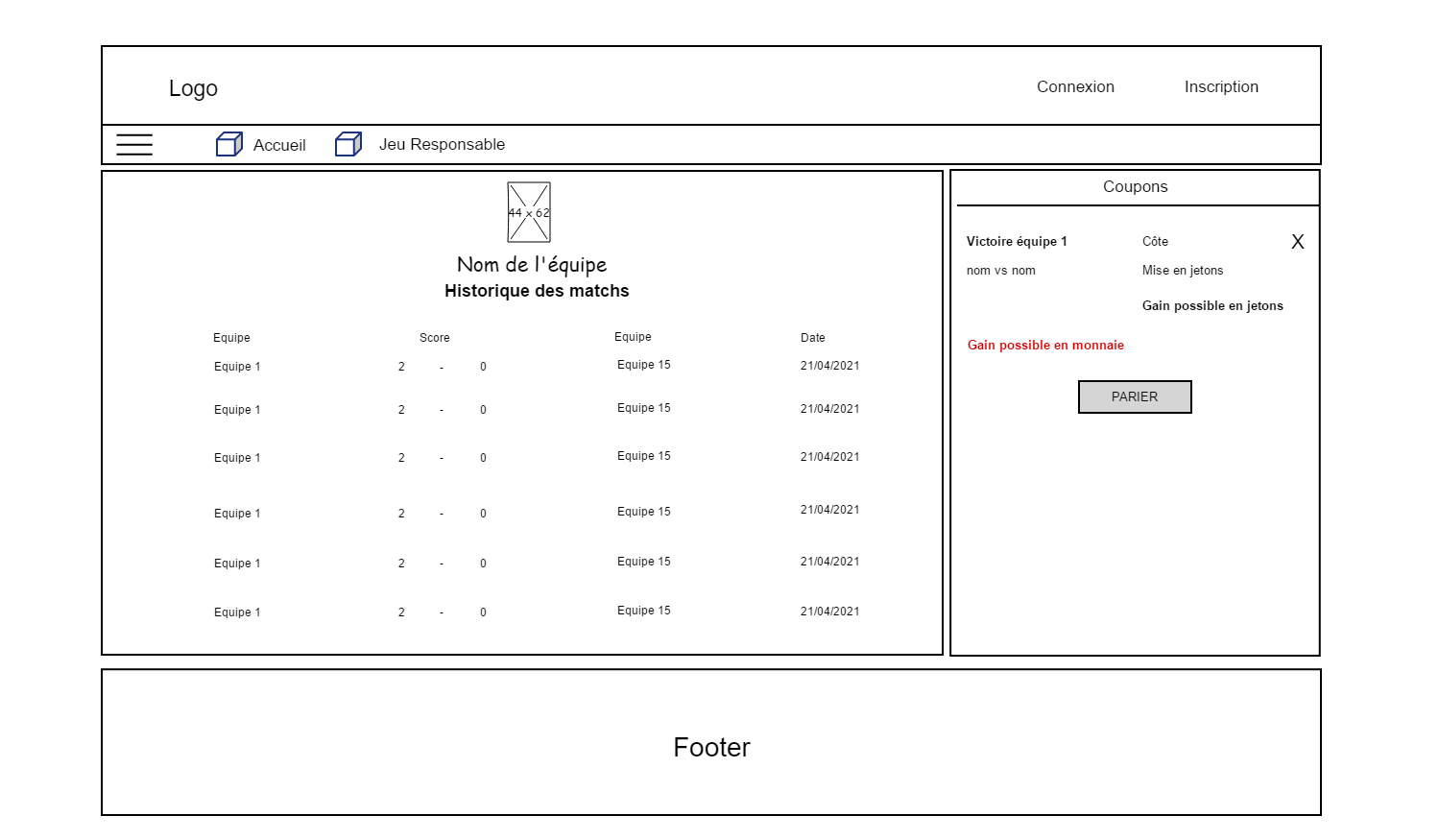
Ici nous pouvons voir les informations sur le match ainsi que les paris disponibles ( ex : corners, score,etc .. )



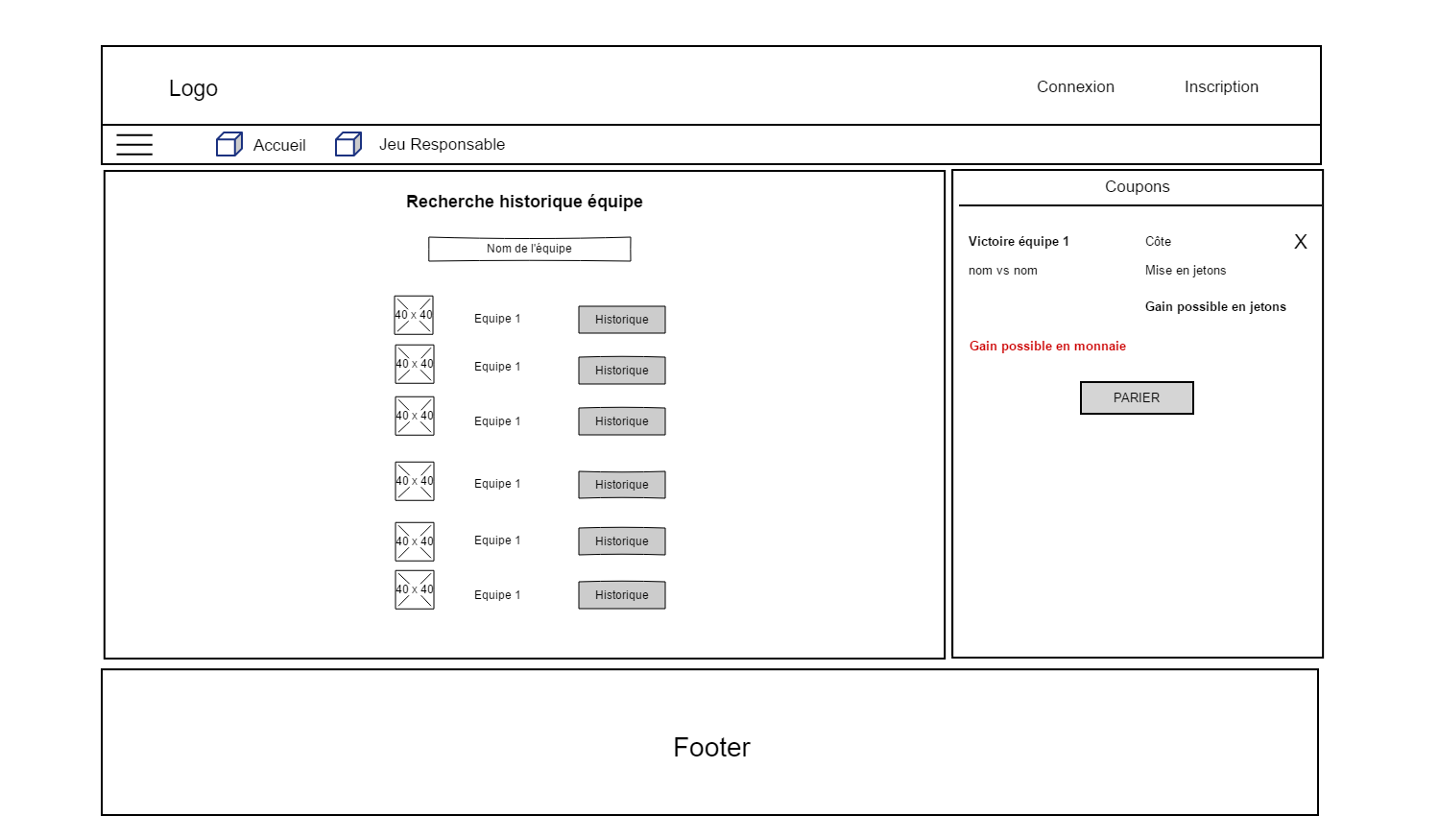
Pour pouvoir miser sur un évènement, l’utilisateur doit être connecté.

**Historique des matchs d’une équipe  :**

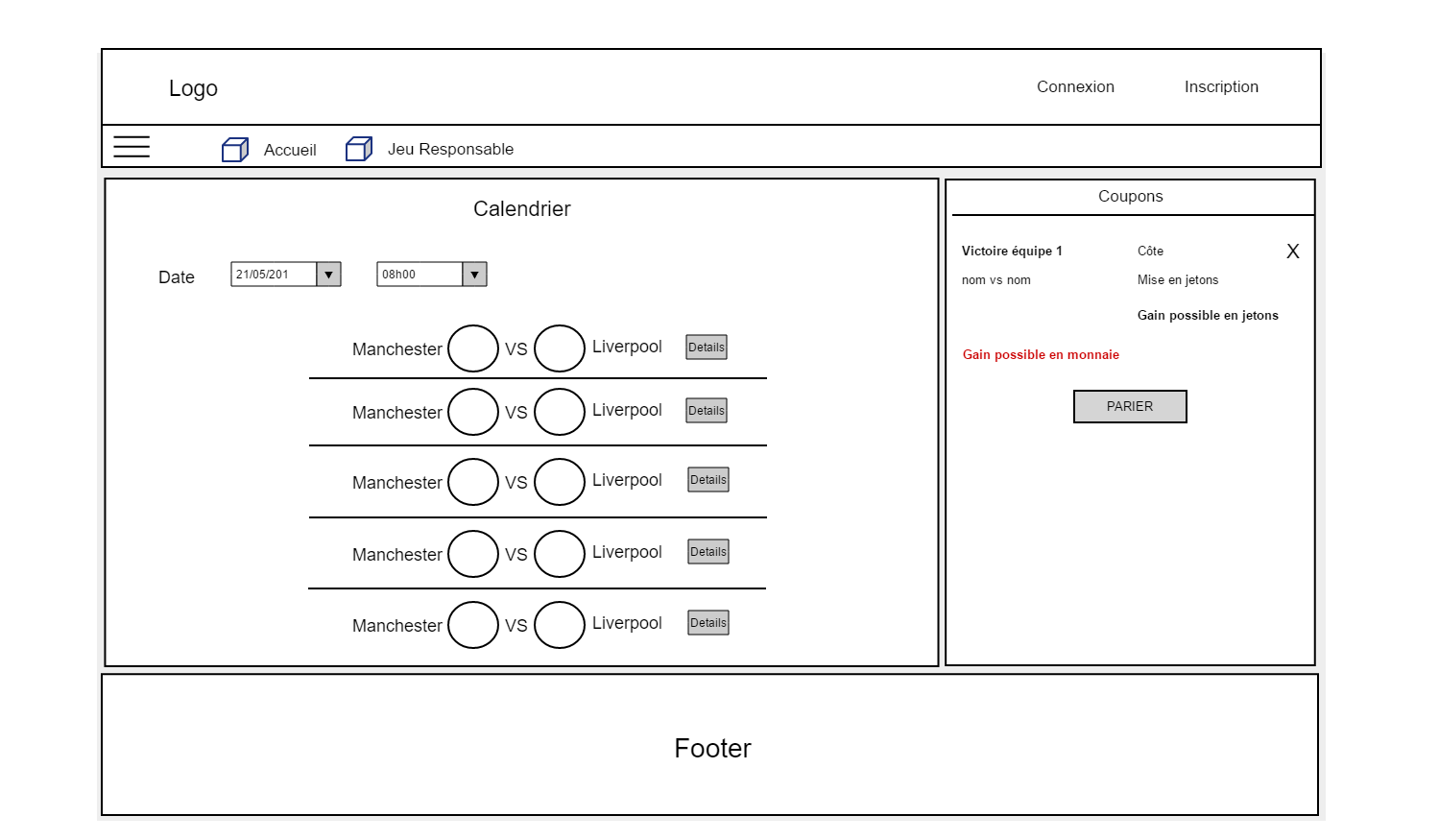
Sur la page précédente on peut accéder à l’historique des matchs d’une des équipes en cliquant sur historique match.



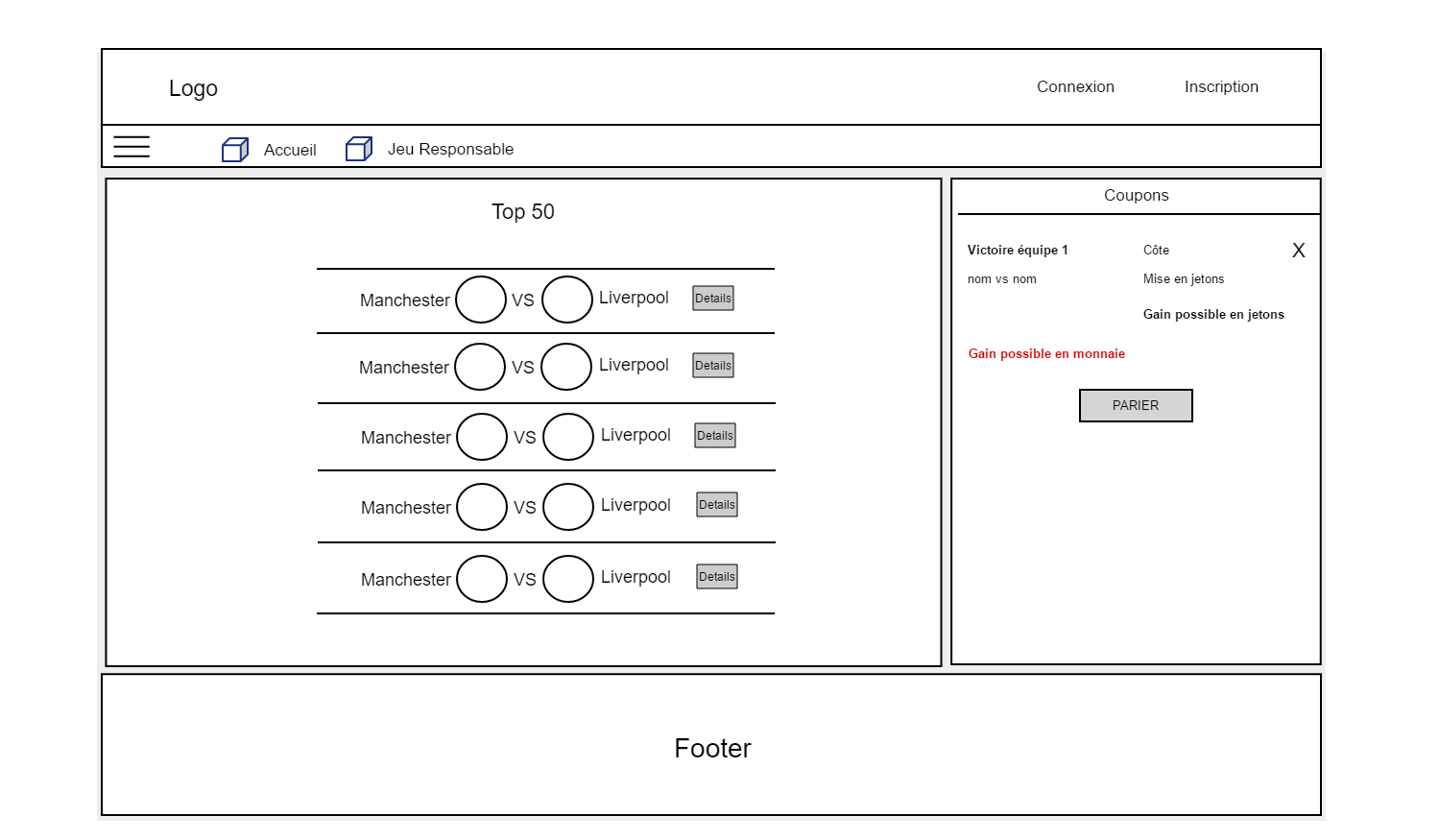
On peut aussi aller rechercher l’historique d’une équipe en cliquant sur « Historique » dans le menu latéral.



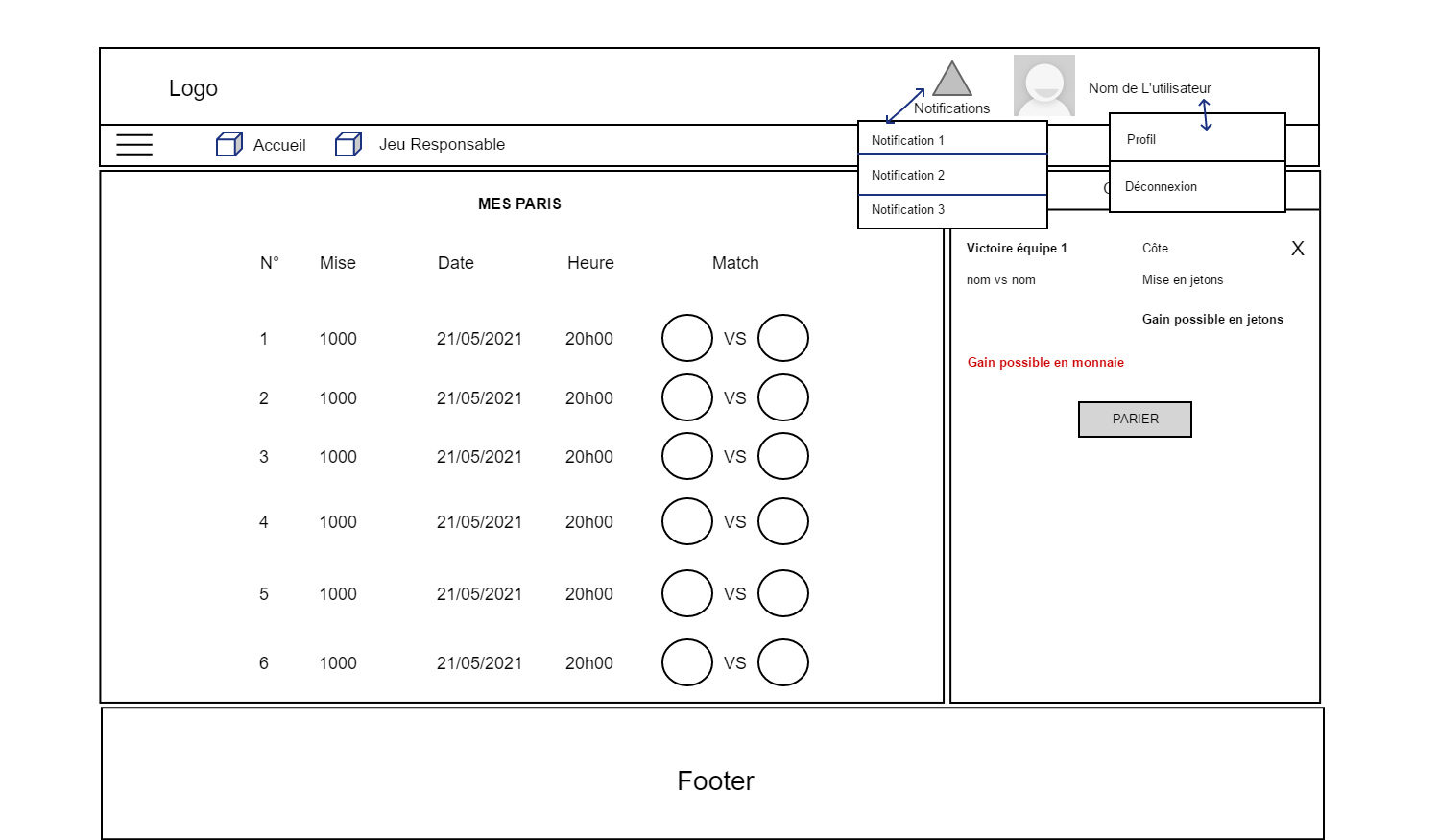
**Page Calendrier :**



**Page des 50 matchs les plus suivis ( avec le plus de parieurs)**

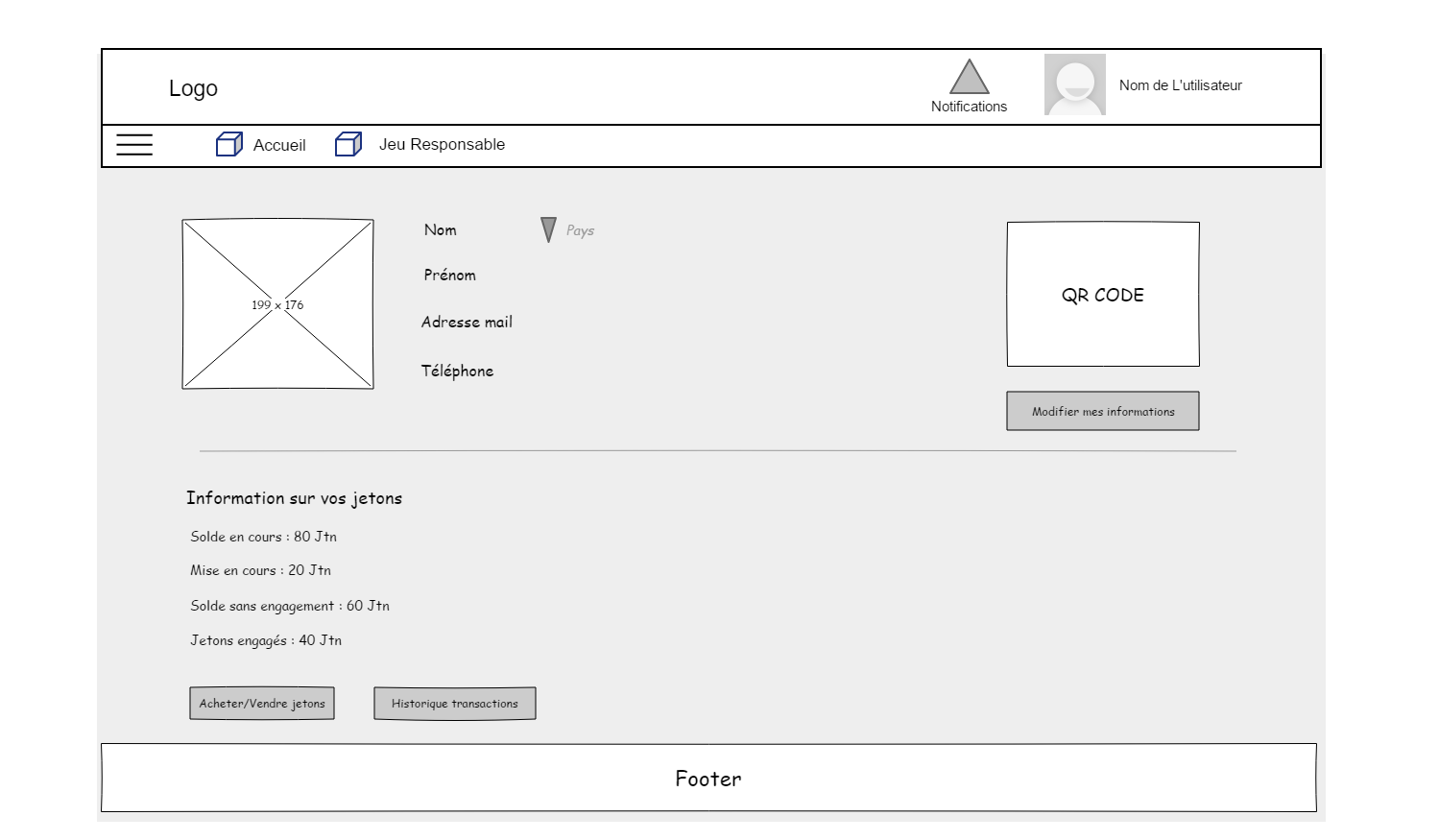


**Page Mes paris**

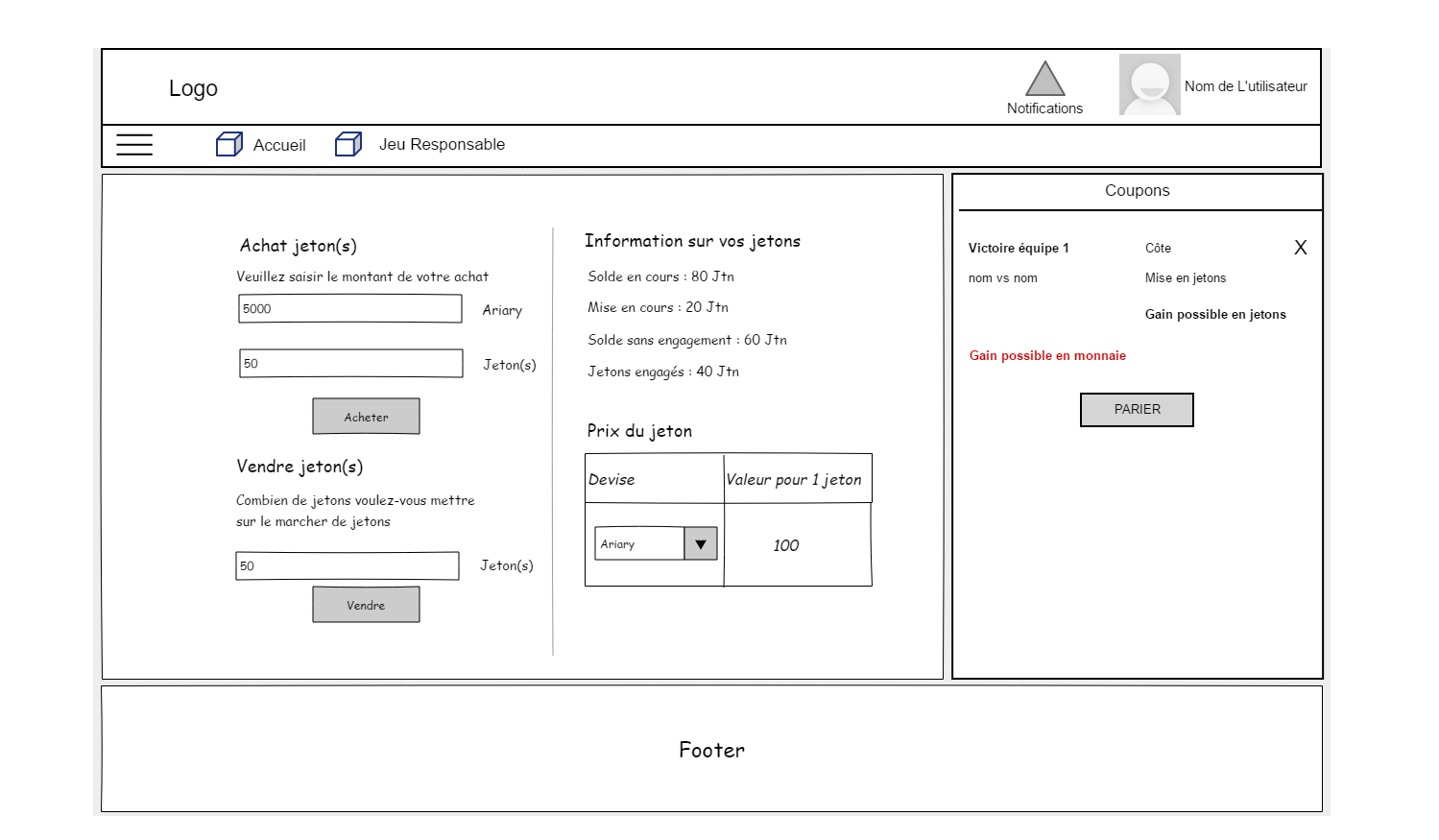


**Page Profil**

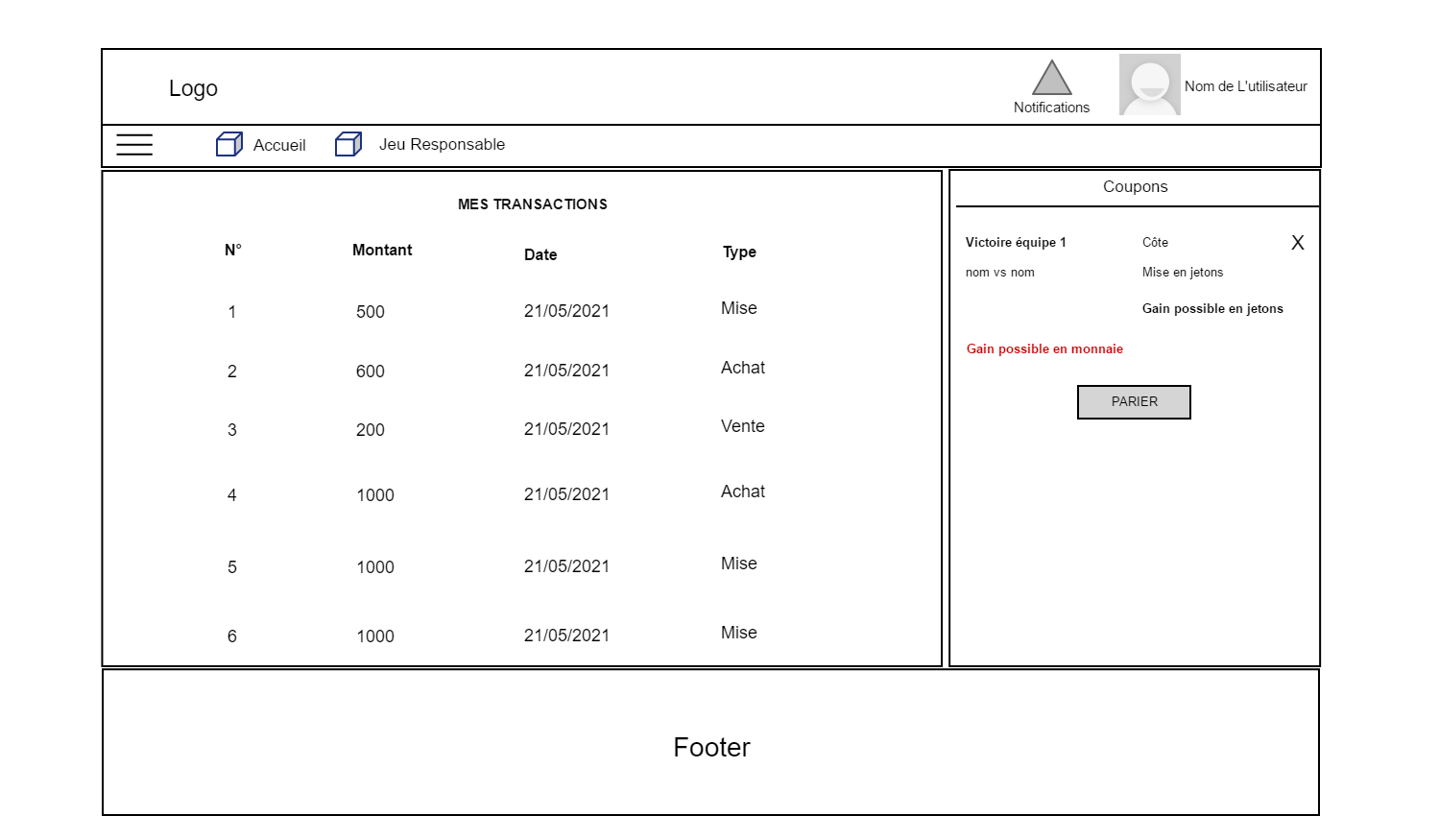
L’utilisateur doit se connecter avant de pouvoir accéder à son profil.



**Pages Achat et Vente de jetons**



**Page historique transactions**



* + - 1. **Wireframes pour la partie mobile :**

**Page de connexion :**

L’utilisateur doit insérer son email ainsi que son mot de passe pour avoir accès à l’intégralité des fonctionnalités de l’application.

****

**Page d’inscription :**

****

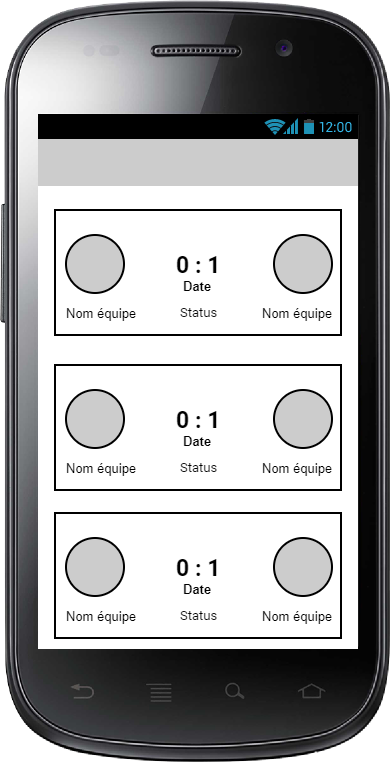
**Page d’accueil :**

****

**Page liste des équipes :**

****

**Page liste des matchs :**

****

**Page détails d’un match :**

Ici on peut voir les détails d’un match, cet page comporte les différentes statistiques du match ainsi que les règles pour parier.

****

Ici l’utilisateur doit insérer une mise pour son pari

****

**Page Calendrier des matchs :**

****

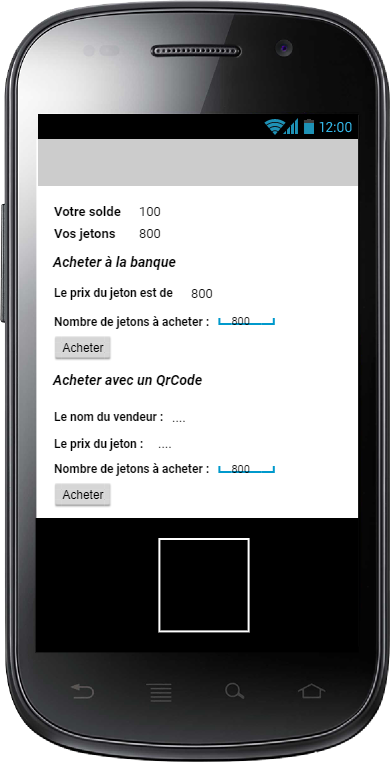
**Page du profil de l’utilisateur :**

****

**Page d’historique des paris :**

****

**Page d’achat de jetons :**

****

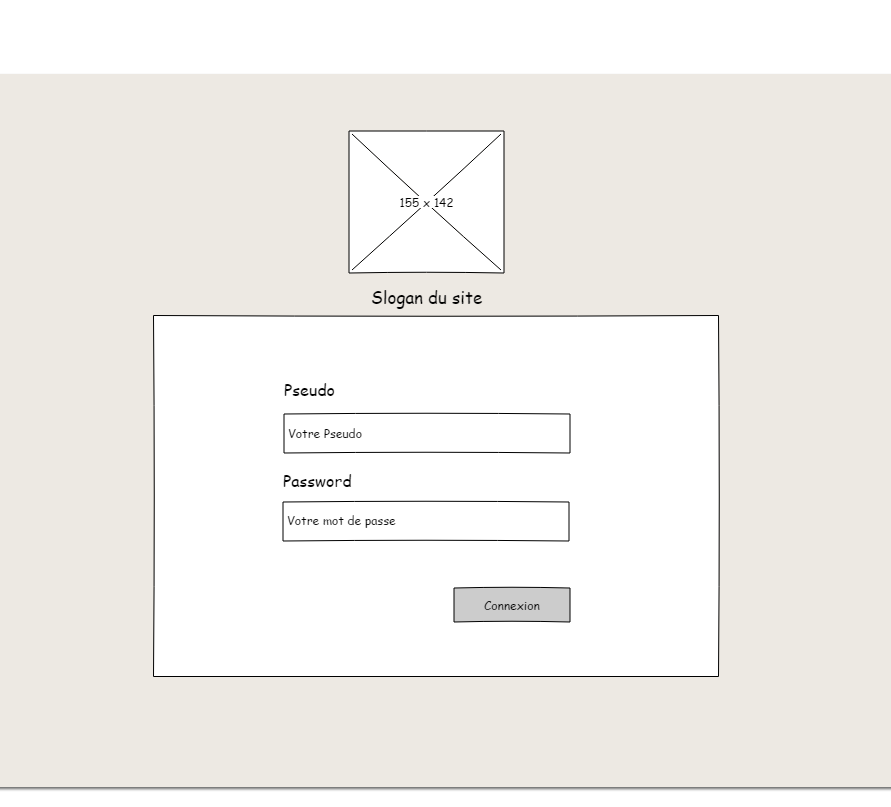
**Page de vente de jetons :**

****

* + - 1. **Wireframes pour le client lourd :**

**Interface de connexion :**

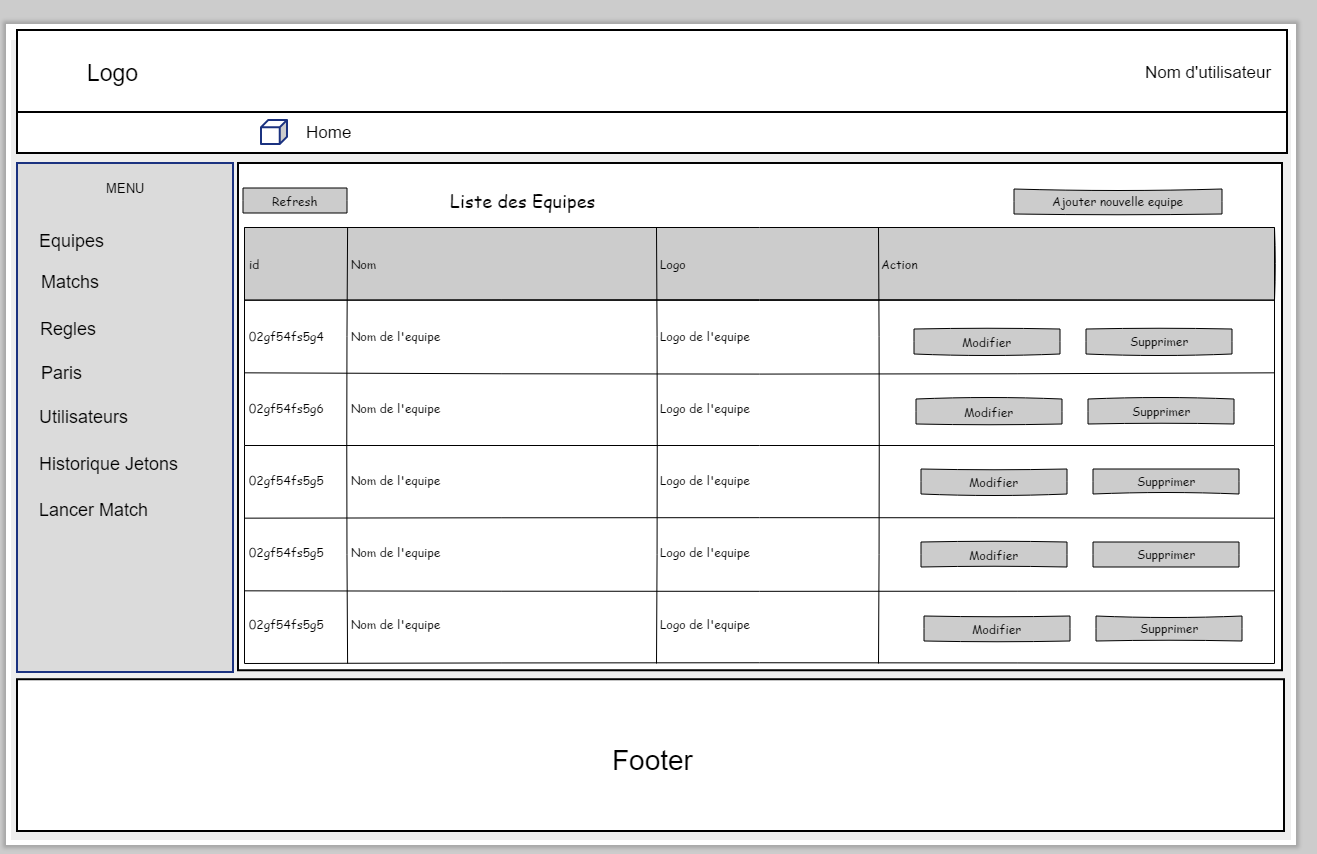
L’Administrateur doit se connecter avec son pseudo et mot de passe pour accéder au Back office.



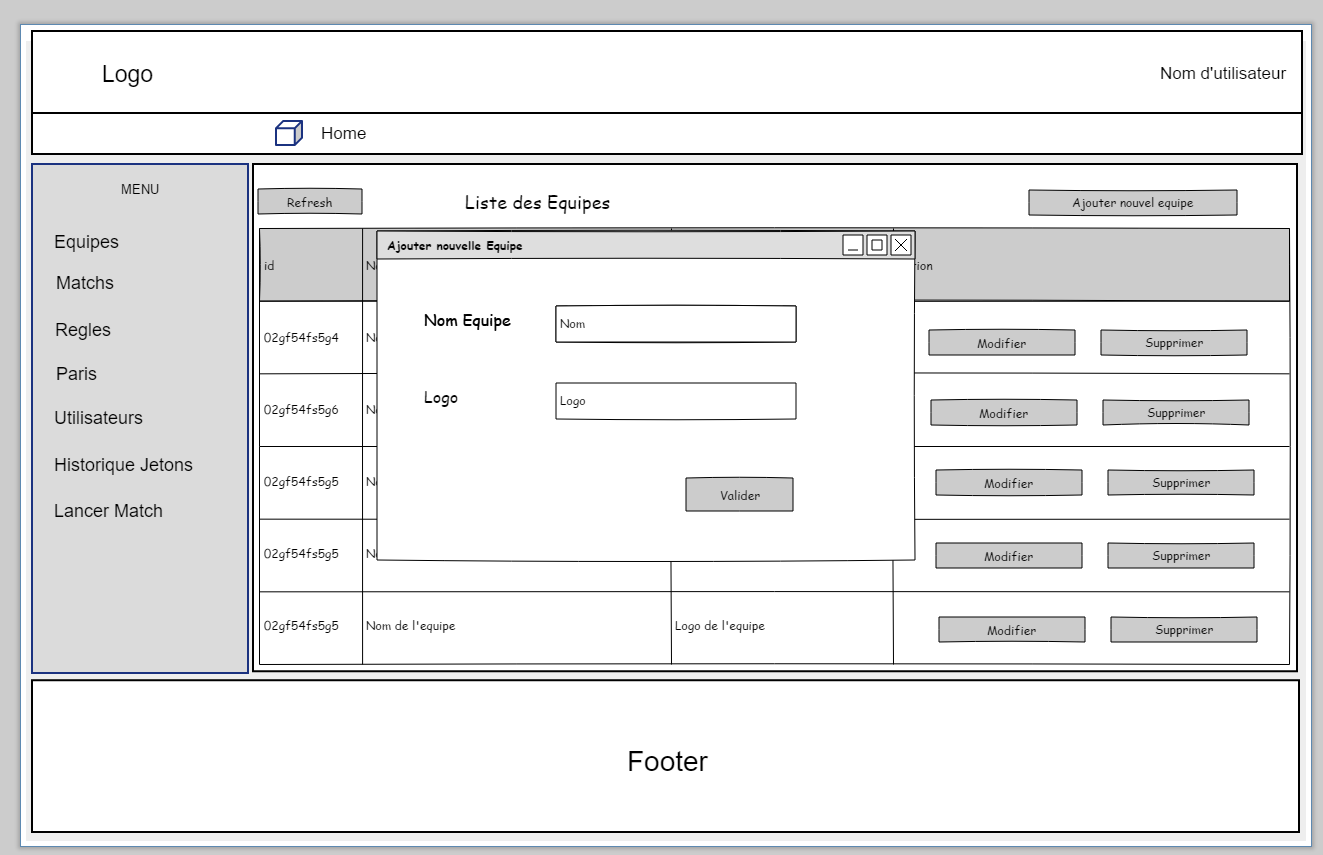
**Affichage Equipe :**

Apres la connexion on aboutit à l’affichage concernant les équipes.

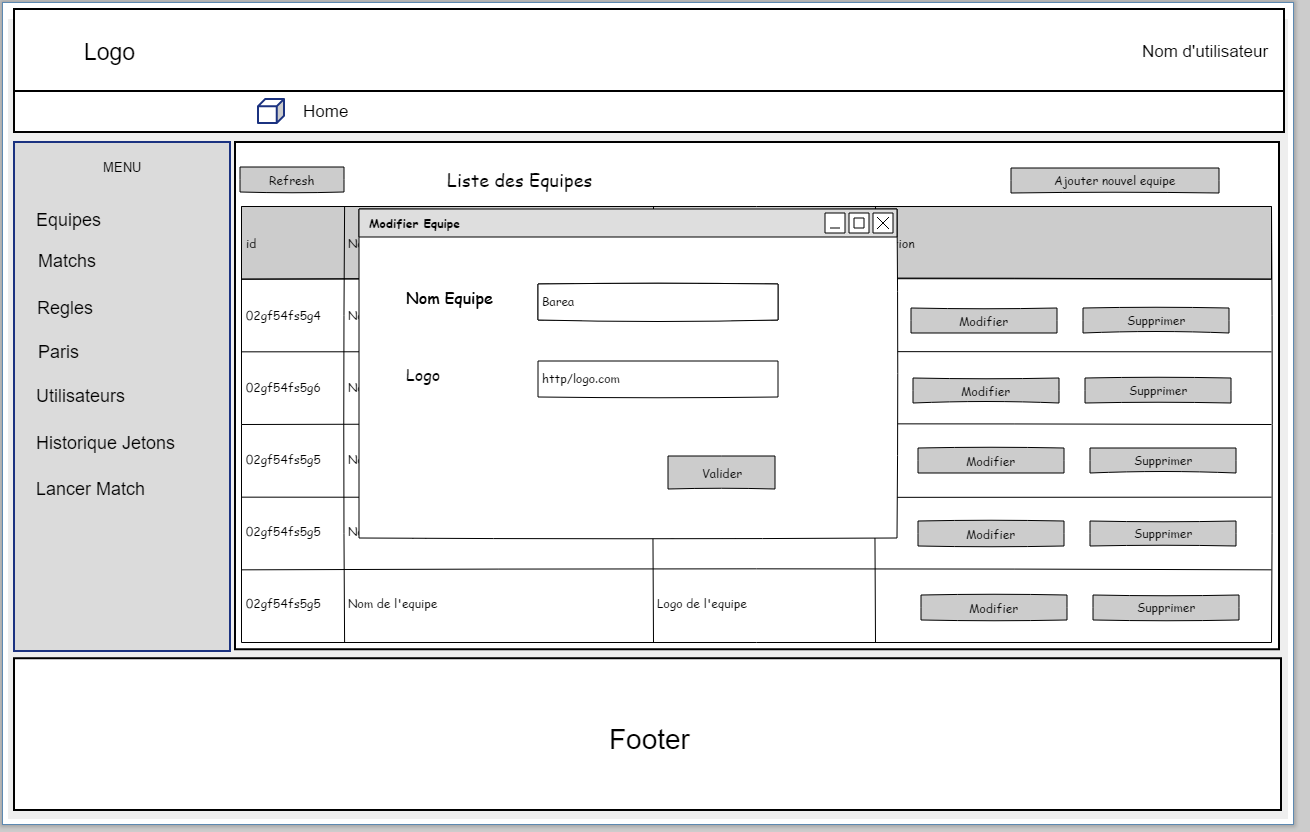
**Liste des Equipes :**



**Ajout d’une Equipe :**

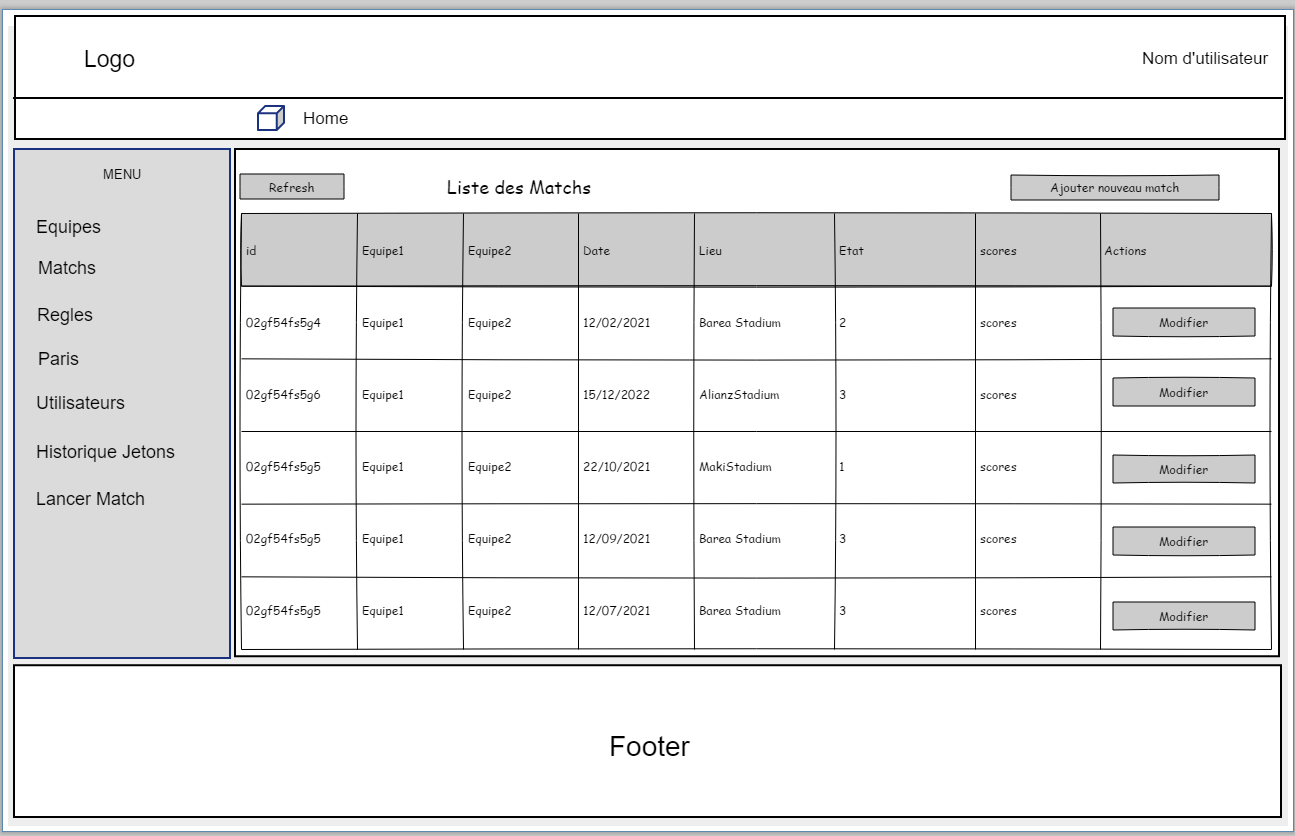


**Modification d’une Equipe :**

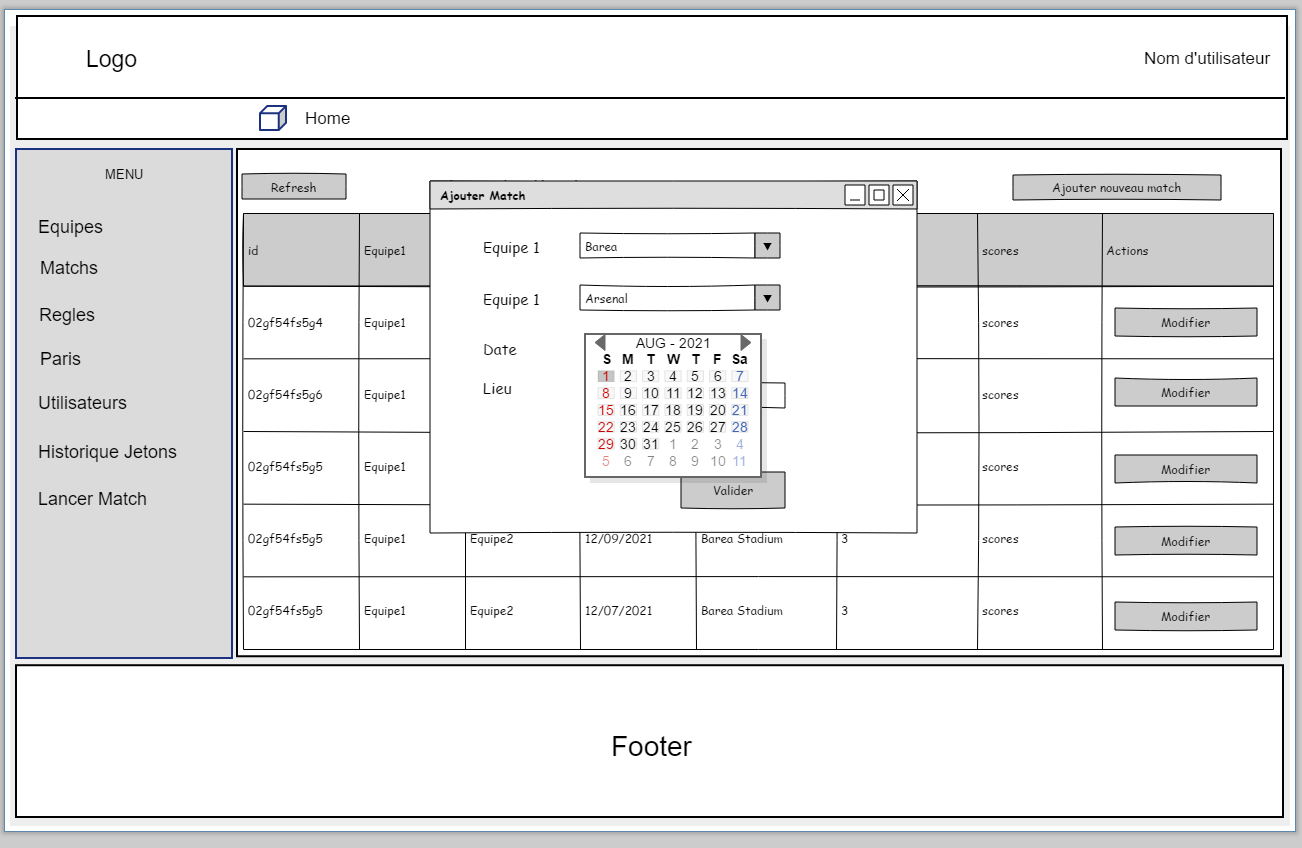


**Interface Match :**

**Liste des Matchs :**

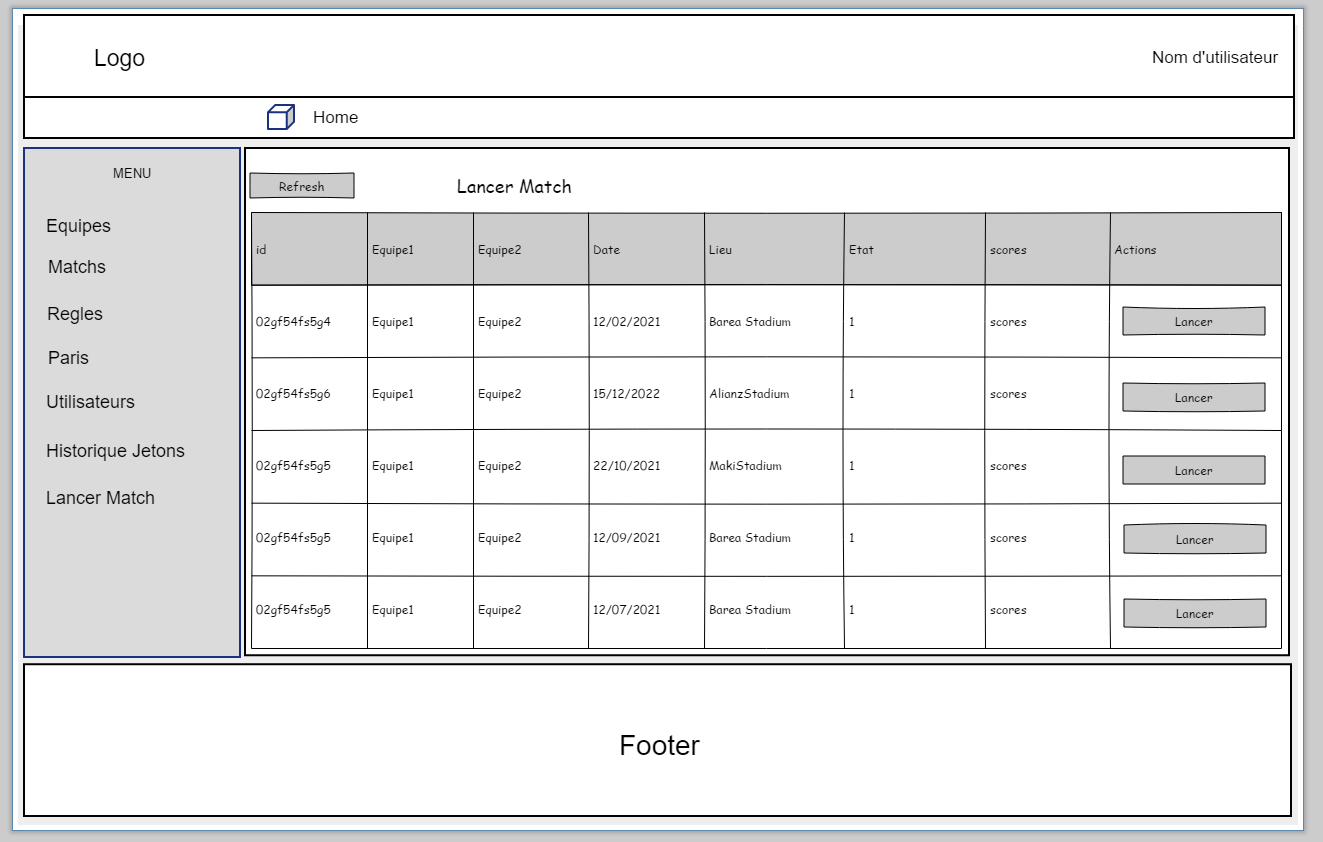


**Ajout d’un nouveau match :**



**Lancer match :**

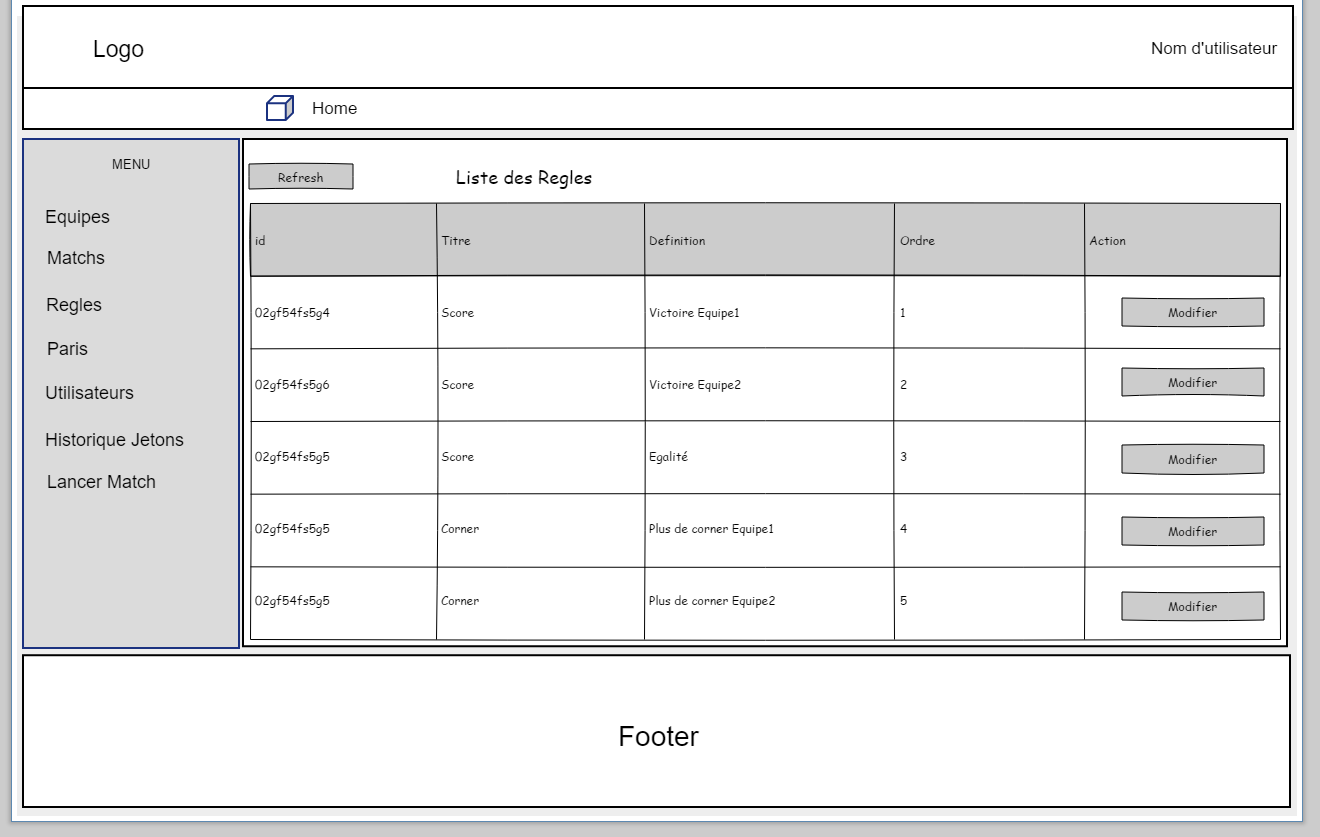
Ici c’est la liste des matchs qui n’ont pas encore été lancés , et que l’administrateur pourra lancer pour que le match ainsi que les paris concernés puissent commencer.



**Interface Règles :**

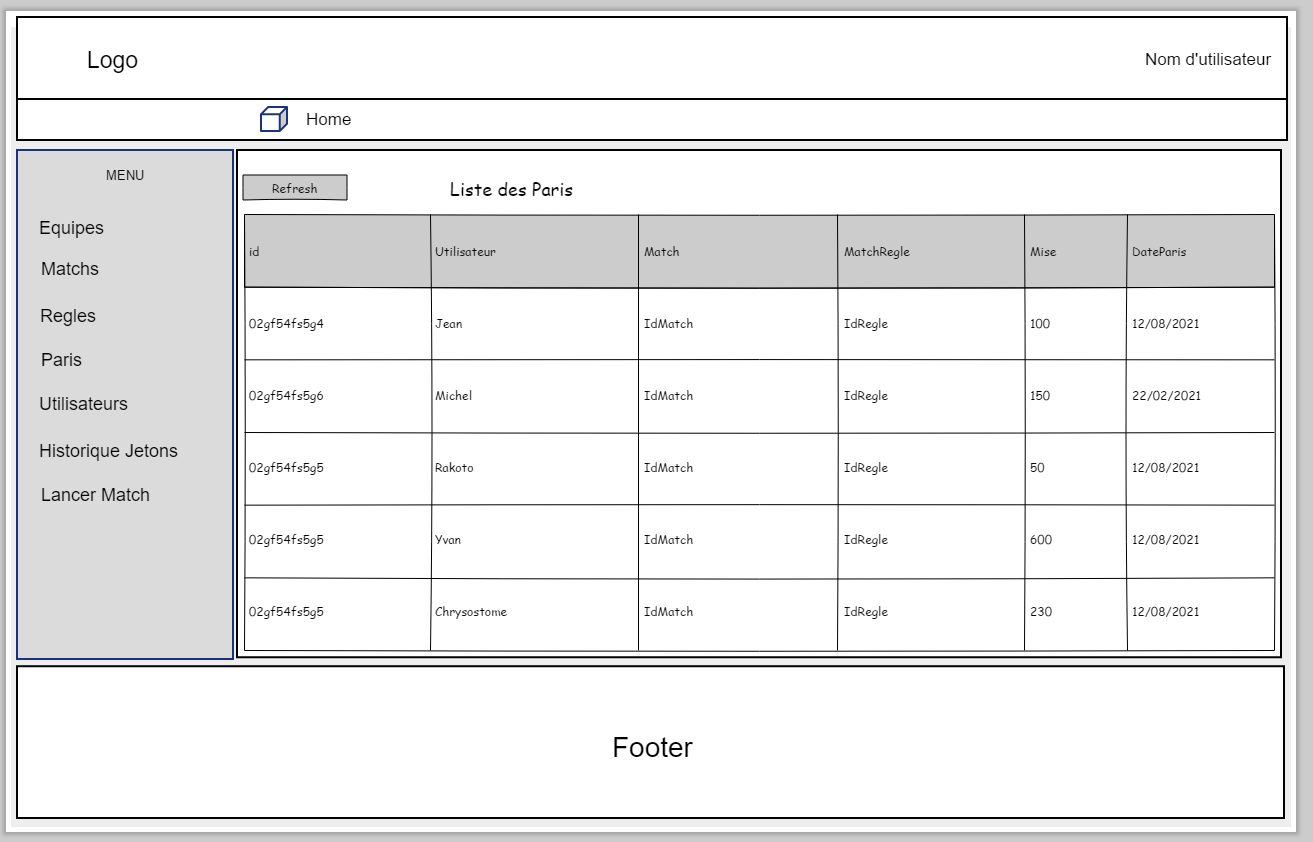
Les règles sont les conditions qui régissent un match et qui permettent les paris.

**Liste des Règles :**



**Interface des paris :**

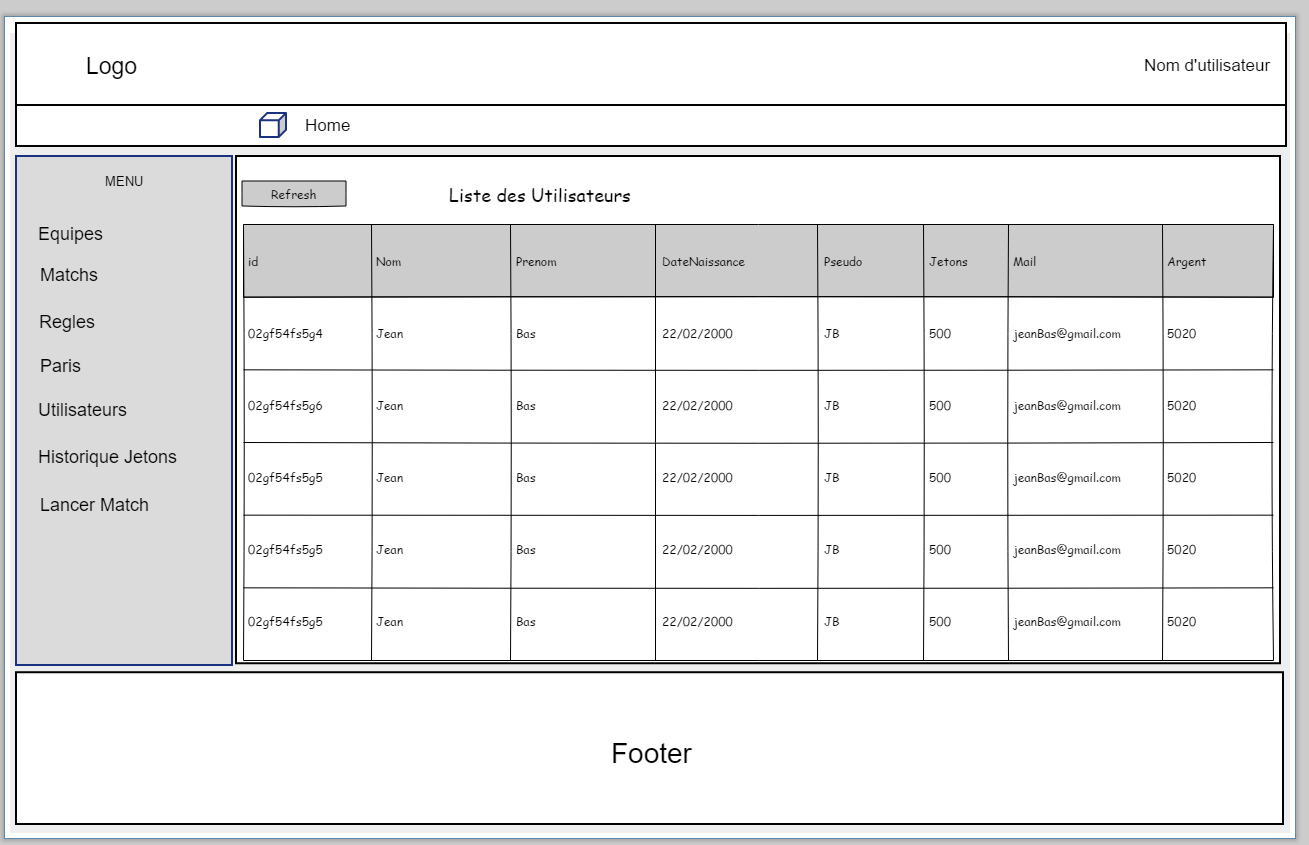
**Liste des paris :**



**Interface des utilisateurs :**

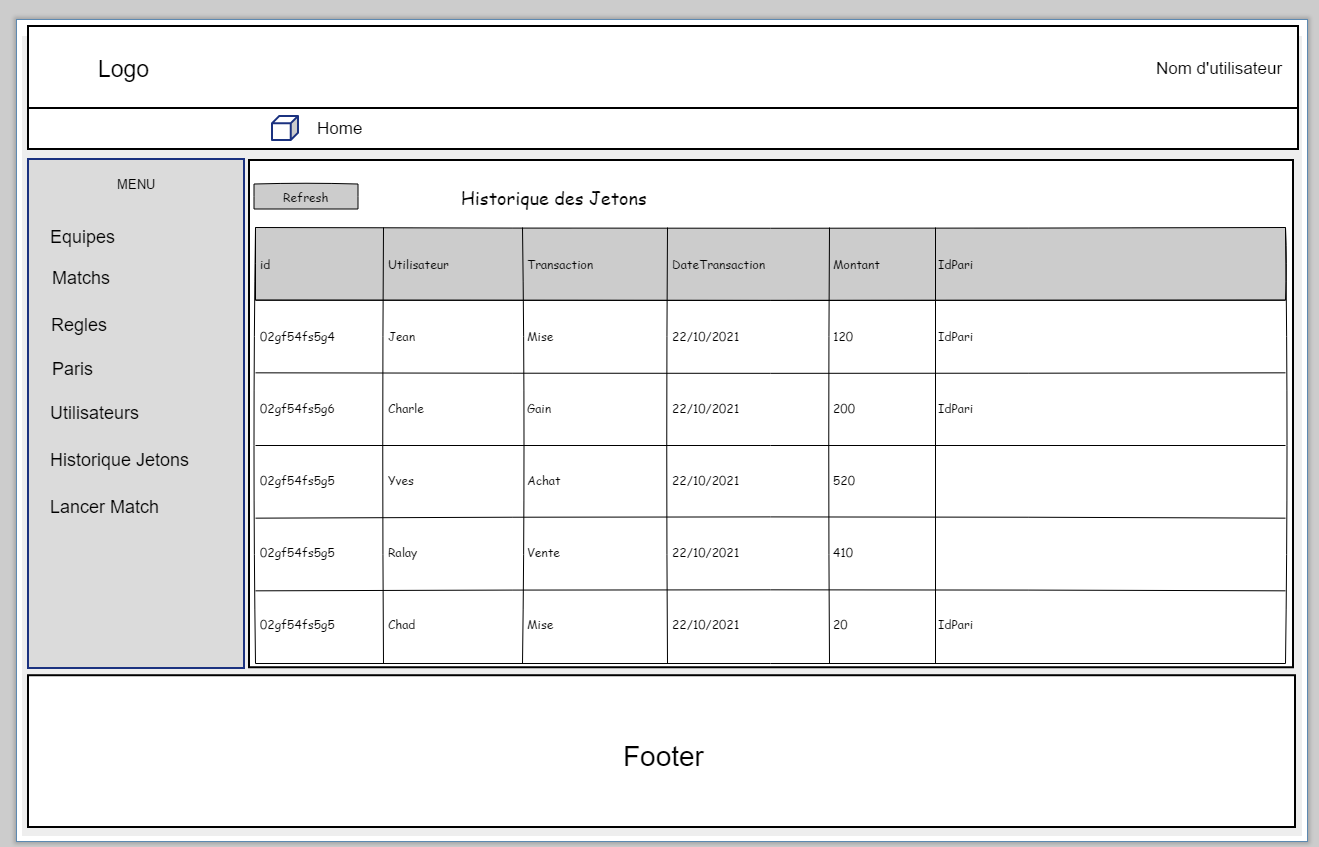
Ce sont les parieurs inscrit dans le site.

**Liste des utilisateurs :**



**Interface des historiques :**

On peut voir ici l’historique des mouvements de jetons des utilisateurs.



* + 1. **Les exigences non fonctionnelles transverses**
       1. **Utilisabilité**

Les fonctionnalités sont accessibles au client Web et mobile moyennant une connexion Internet.

Le design et la structure simple de l’application web lui profèrent une bonne ergonomie, en respectant la règle des trois clics pour faire gagner du temps à l’utilisateur.

Les fonctionnalités de gestion sont accessibles aux administrateurs du site localement à tout moment via une application lourde (donc installée sur le post).

* + - 1. **Robustesse/fiabilité**

L’application repose sur deux bases de données séparées permettant de dispatcher les données de manière en faciliter l’accès et le traitement. L’architecture de l’application lui profère aussi des niveaux de sécurité, diminuant les risques de fautes et de corruption de données.

* + - 1. **Supportabilité**

La conception de l’application a été pensée de façon à bien répartir les rôles de chaque composant de manière hiérarchique. En effet, nous avons un composant central (API) chargé uniquement de traiter, récupérer et renvoyer les données vers les trois applications. Ainsi tout le business logique est centralisé. Les applications ont été développé de manière à faciliter leur maintenance et leur permettre une évolutivité facile.

# Conception

# Contraintes techniques

**Langage de développement**

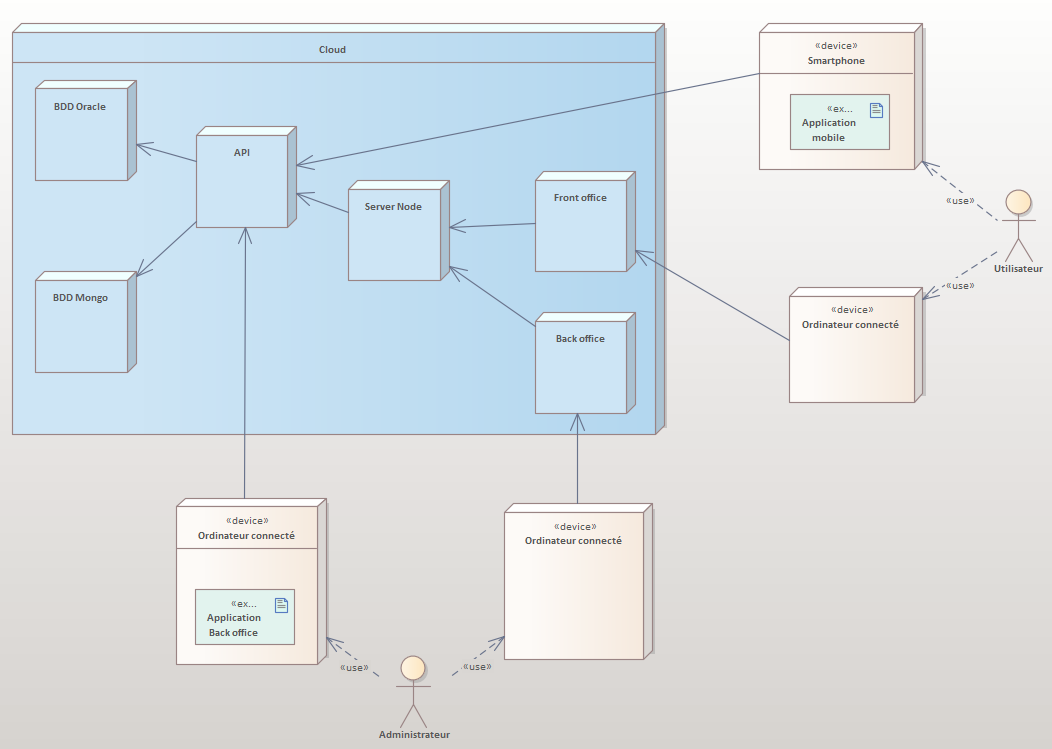
* Langage Java pour le serveur d’application
* Utilisation de Javascript côté serveur
* Utilisation de composant Angular pour la partie front
* Client lourd Winform

**Bases de données**

* Utilisation d’une Base Oracle
* Utilisation d’une base MONGODB
* Base de données locale sur le téléphone

**Modules**

* Utilisation des WebServices REST
* Implémentation d’un module mapping
* QRCode sur l’application mobile
  1. **Plateforme technique**
     1. **Architecture logicielle**

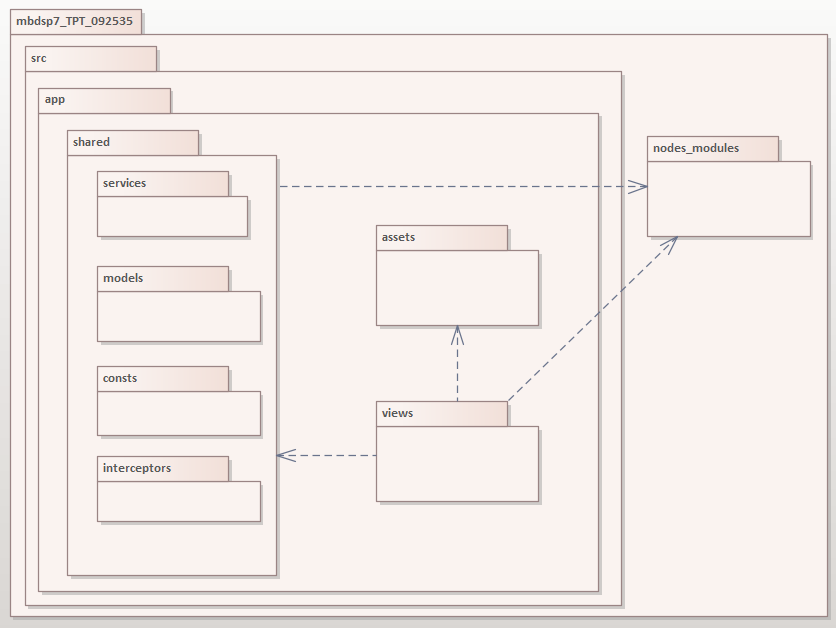


La base de données Orcale est hébergée sur le serveur Oracle Cloud. La base Mongo est une base en ligne appelée MongoDBAtlas. Le front-office, l’application back office web ,l’API et le serveur node sont hébergés sur la plateforme Heroku. Ainsi le front office et le back office communiquent tous deux avec le Server Node qui appellera l’API , intermédiaire entre les applications et les bases de données. Le back office client lourd et l’application mobile communiqueront directement avec l’API.

* 1. **Conception du logiciel développé**
     1. **Conception du code source**
        1. **Le code source – vue statique -web**

L’architecture d’Angular se base sur la construction de l’application comme un ensemble de blocs fonctionnels appelés des composants Angular. Ces composants définissent des ***vues*** qui sont des ensembles d’éléments d’écran.

Les composants utilisent des services qui fournissent des fonctionnalités spécifiques. Les fournisseurs de services peuvent être injectés dans les composants en tant que dépendances, rendant ainsi le code modulaire, réutilisable et efficient.



* + - 1. **Le code source – vue statique – mobile**

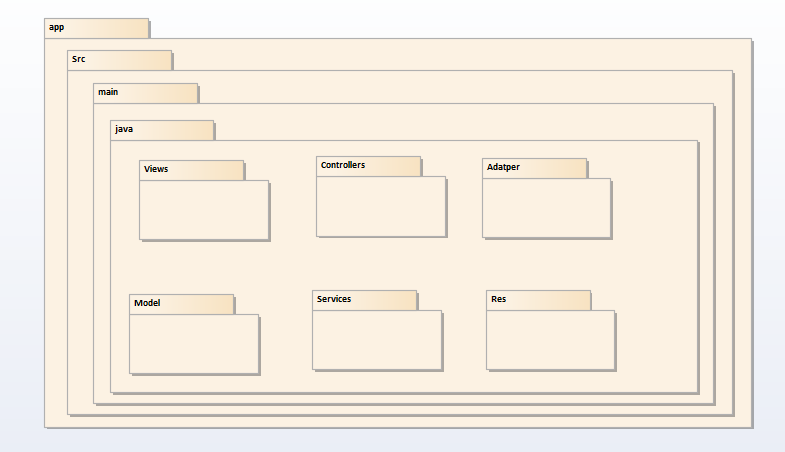
L’architecture de notre application mobile avec java suit le type MVC (Mode-View-Controller).

Dans la partie la partie Views il y a les différents pages qui seront utilisés sur le projet, il est composé des fragments et des activités.

Ensuite la partie Services est celle qui fait l’interprète entre l’API et l’application mobile, elle contient toutes les fonctions qui seront utilisés pour envoyer les requêtes et récupérer les données vers l’API.

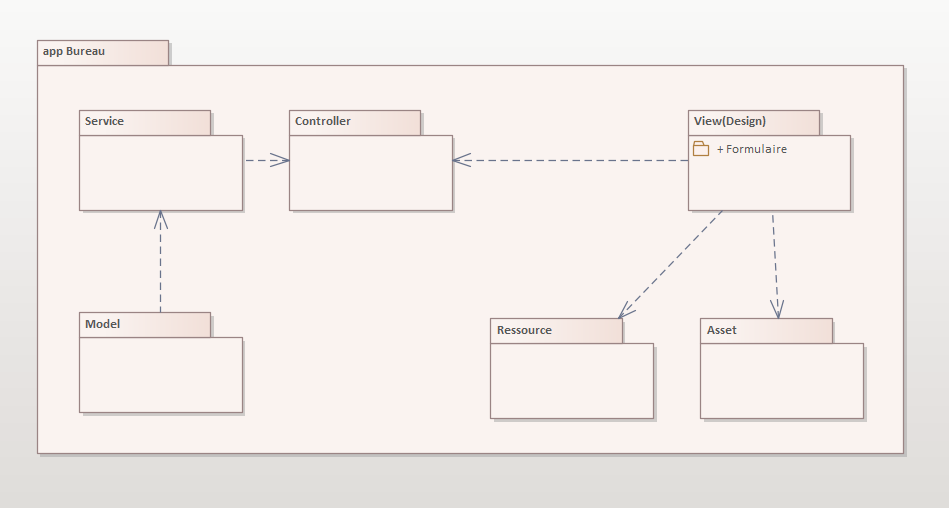
La partie Model est utilisé pour adapter les données venant de l’API en objets utilisables dans l’application.

C’est dans la partie Res qu’est contenu tous les ressources nécessaire et utilisés par l’application.



* + - 1. **L****e code source – vue statique – Application de bureau**

L’architecture de notre application bureau avec C# est du type MVC (Model – View – Controller). Nous pouvons complètement découpler nos couches applicatives métier et de présentation. De plus, nous aurons un objet complètement indépendant pour contrôler la couche de vue(view). L'indépendance entre les objets/couches de notre projet fourni par le MVC rendra la maintenance un peu plus facile et la réutilisation du code plus facile. Les vues sont contrôlées par les contrôleurs qui utilisent des API pour échanger avec la base. Les données recues sont traités par les modèles avant d’être renvoyées aux contrôleurs qui retourneront les vues pour l’affichage.



* + - 1. **Le code source – vue statique – API**

L’API développé avec Spring boot suit une architecture en couches dans laquelle chaque couche communique avec la couche directement en dessous ou au-dessus (structure hiérarchique).

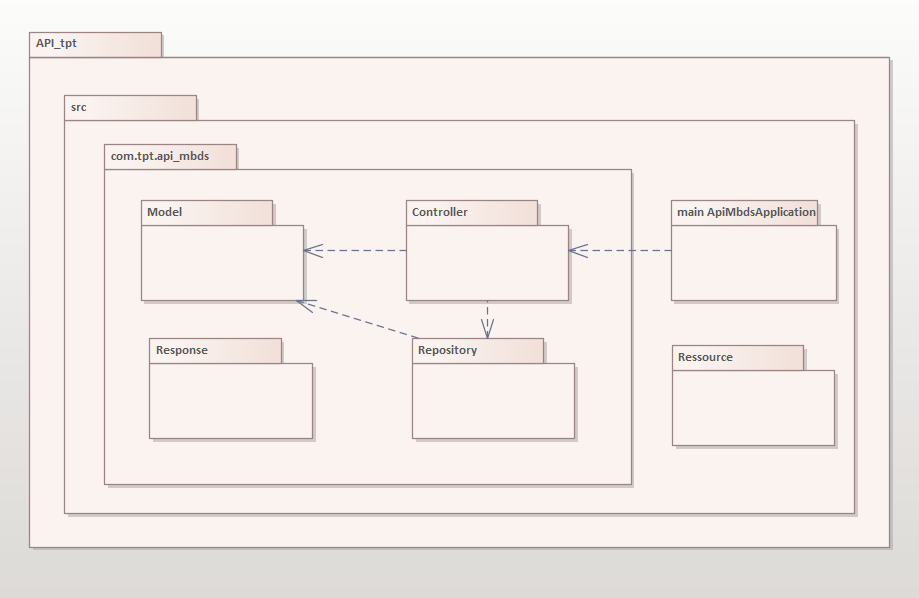
Ces couches sont (Couche de présentation, contrôleur, model et de base de données).

La couche de présentation gère les requêtes HTTP, traduit le paramètre JSON en objet, authentifie la requête et la transfère à la couche contrôleur.

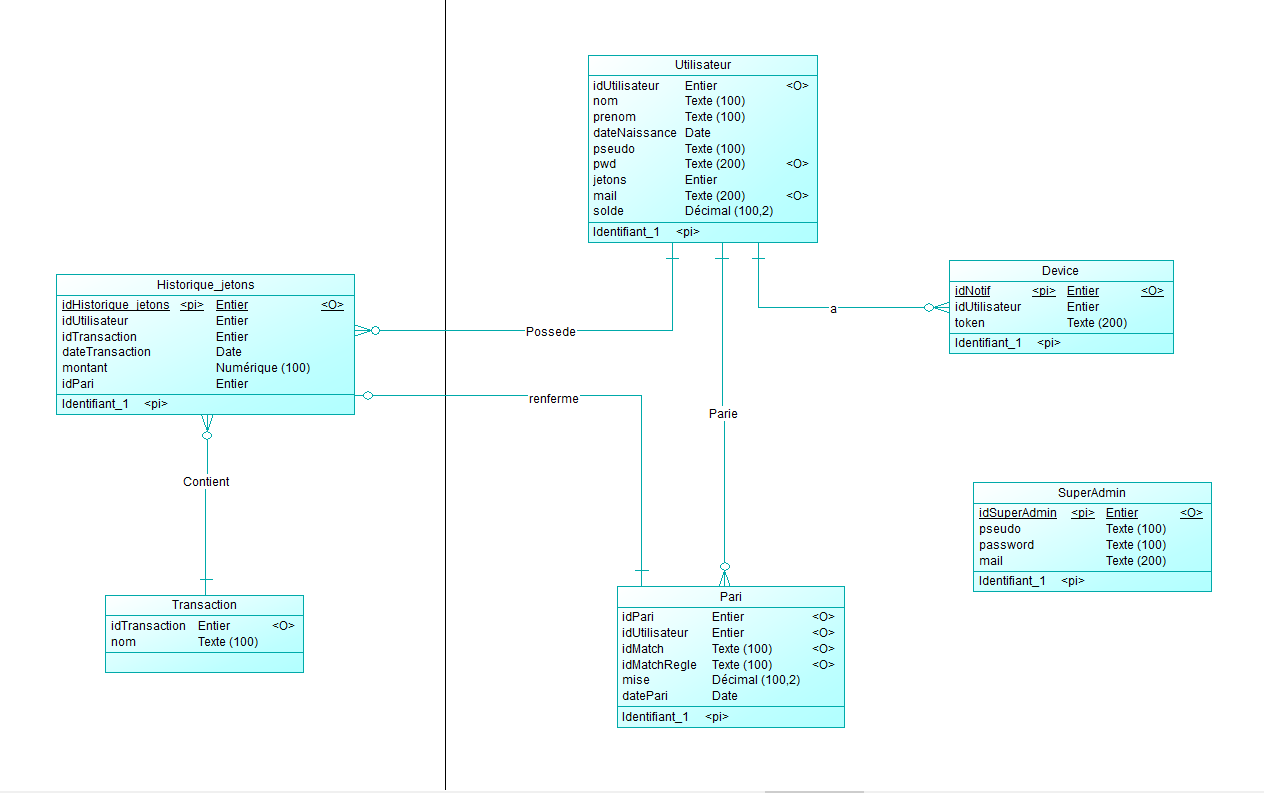
La couche contrôleur : gère toute la logique métier. Il se compose de classes de services et utilise des services fournis par des couches d'accès aux données. Il effectue également l’autorisation et la validation.

La couche model : traduit les objets venant du contrôleur vers les lignes de la base de données.

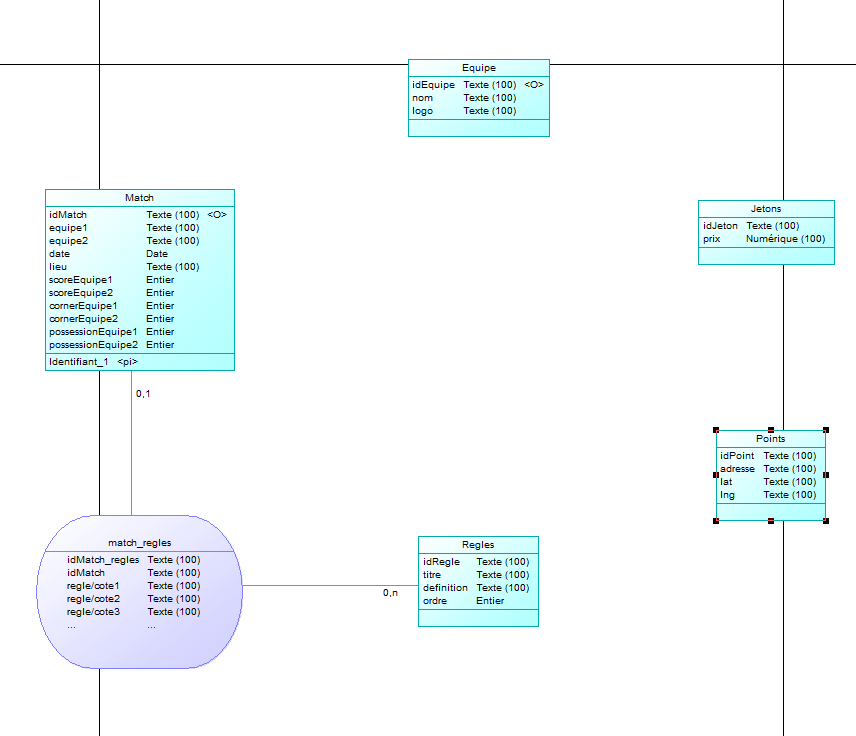
La couche de base de données : avec les « repositories » qui s’occupent des opérations CRUD (créer, récupérer, mettre à jour, supprimer) sont effectuées.



* + 1. **Modélisation des données**
       1. **Base Oracle**



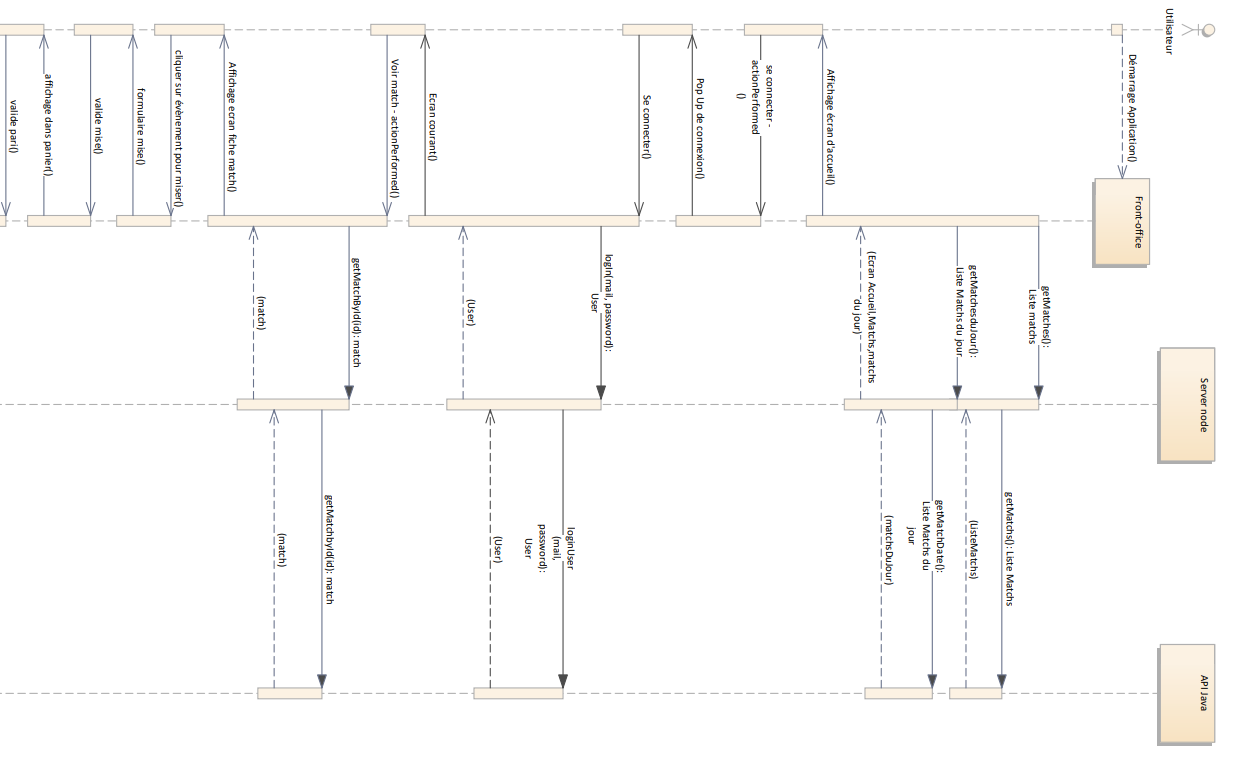
* + - 1. **Base Mongo**

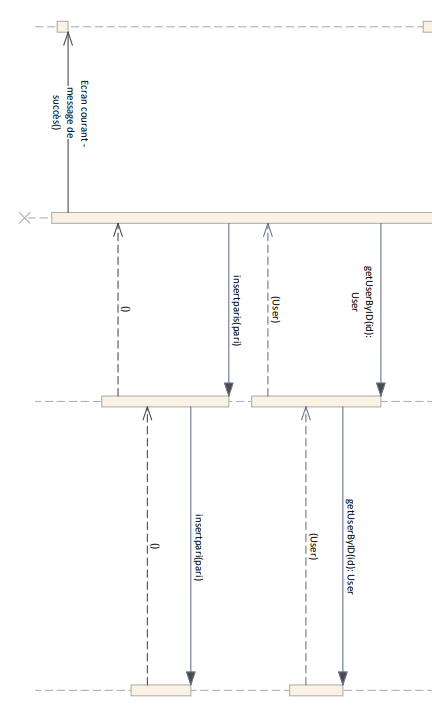


* + 1. **Réalisation des cas d’utilisation**

**Cas d’utilisation : Faire un pari sur Web**

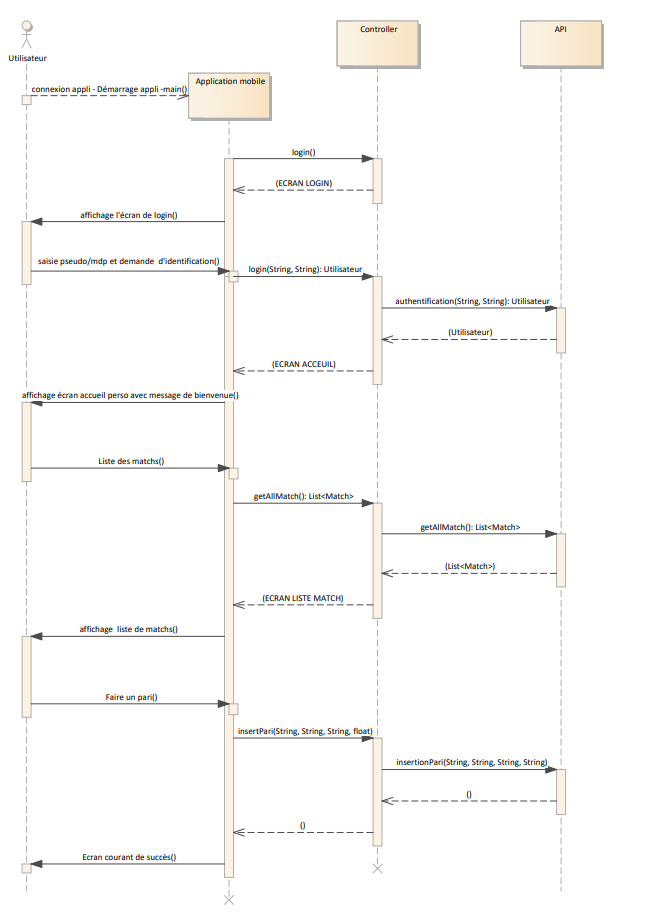
**Scénario normal**





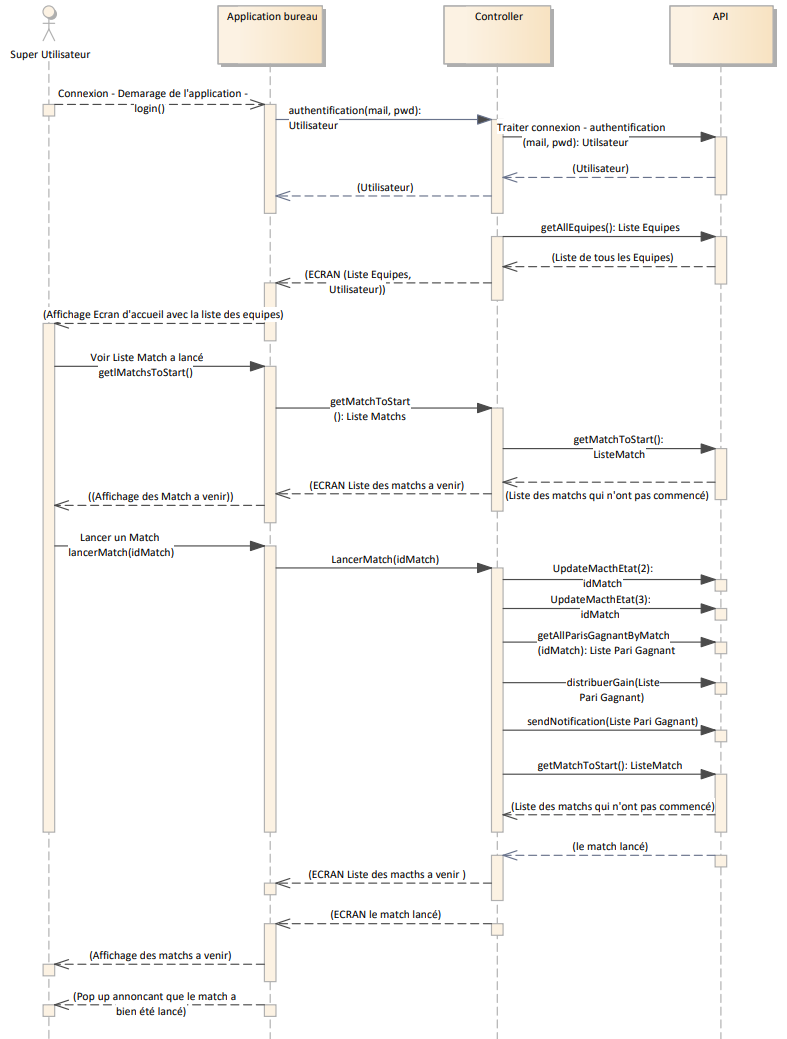
**Cas d’utilisation : Faire un pari sur mobile**

**Scénario normal :**



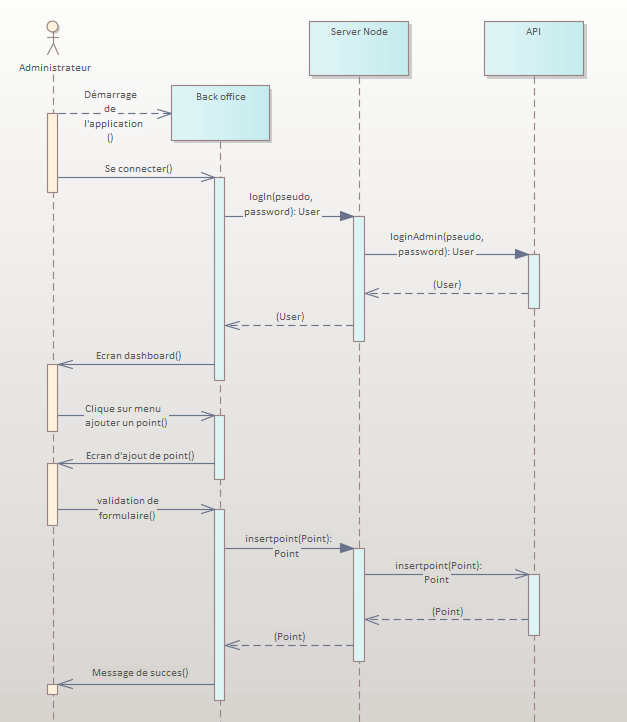
**Cas d’utilisation : Lancer un match sur l’application de bureau (backOffice)**

**Scénario normal :**



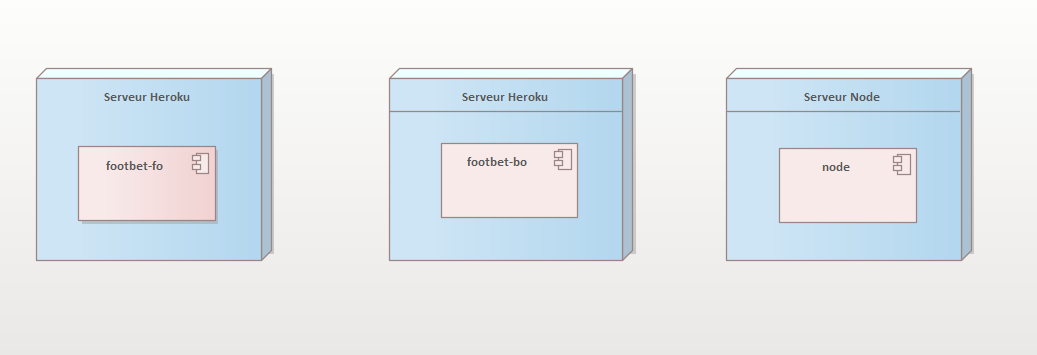
**Cas d’utilisation : Ajout point de vente sur web (backOffice)**

**Scénario normal :**



* + 1. **Les composants et leur déploiement**
       1. **Application Web**

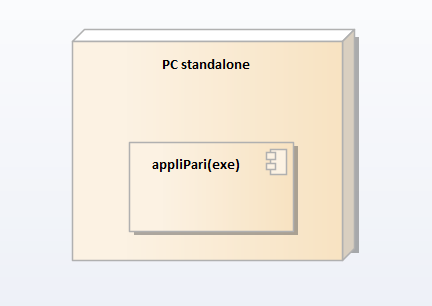
Le front office, le back office secondaire et le server node sont déployés sur la plateforme Heroku.



* + - 1. **Client lourd**

Le système est produit sous la forme d'un seul composant exécutable : un module ".exe" appelé « appliPari » :

Il est déployé sur l’unique nœud, le PC standalone :



* + - 1. **Application Mobile**

Le système est produit sous la forme d'une application mobile exécutable" appelée « appliPari » :

Il est déployé sur le téléphone mobile :

