



Dossier rédigé par ALMABOUADA Abdennour



Sommaire

1. Contexte du projet

1.1. Présentation du projet

1.2. Date de rendu du projet

2. Besoins fonctionnels

3. Ressources nécessaires à la réalisation du projet

3.1. Ressources matérielles

3.2. Ressources logicielles

4. Gestion du projet

5. Conception du projet

5.1. Le front-end

5.1.1. Wireframes

5.1.2. Maquettes

5.1.3. Arborescences

5.2. Le back-end

5.2.1. Diagramme de cas d'utilisation

5.2.2. Diagramme d'activités

5.2.3. Modèles Conceptuel de Données (MCD)

5.2.4. Modèle Logique de Données (MLD)

5.2.5. Modèle Physique de Données (MPD)

6. Technologies utilisées

6.1. Langages de développement Web

6.2. Base de données

7. Sécurité

7.1. Login et protection des pages administrateurs

7.2. Cryptage des mots de passe avec Bcrypt

7.3. Protection contre les attaques XSS (Cross-Site Scripting)

7.4. Protection contre les injections SQL

1. Contexte du projet

1.1. Présentation du projet

Votre agence web a été sélectionnée par le comité d'organisation des jeux olympiques de Paris 2024 pour développer une application web permettant aux organisateurs, aux médias et aux spectateurs de consulter des informations sur les sports, les calendriers des épreuves et les résultats des JO 2024.

Votre équipe et vous-même avez pour mission de proposer une solution qui répondra à la demande du client.

1.2. Date de rendu du projet

Le projet doit être rendu au plus tard le 22 mars 2024.

2. Besoins fonctionnels

Le site web devra avoir une partie accessible au public et une partie privée permettant de gérer les données.

Les données seront stockées dans une base de données relationnelle pour faciliter la gestion et la mise à jour des informations. Ces données peuvent être gérées directement via le site web à travers un espace administrateur.

3. Ressources nécessaires à la réalisation du projet

3.1. Ressources matérielles

Les ressources matérielles nécessaires à la réalisation du projet sont :

- Un ordinateur portable.

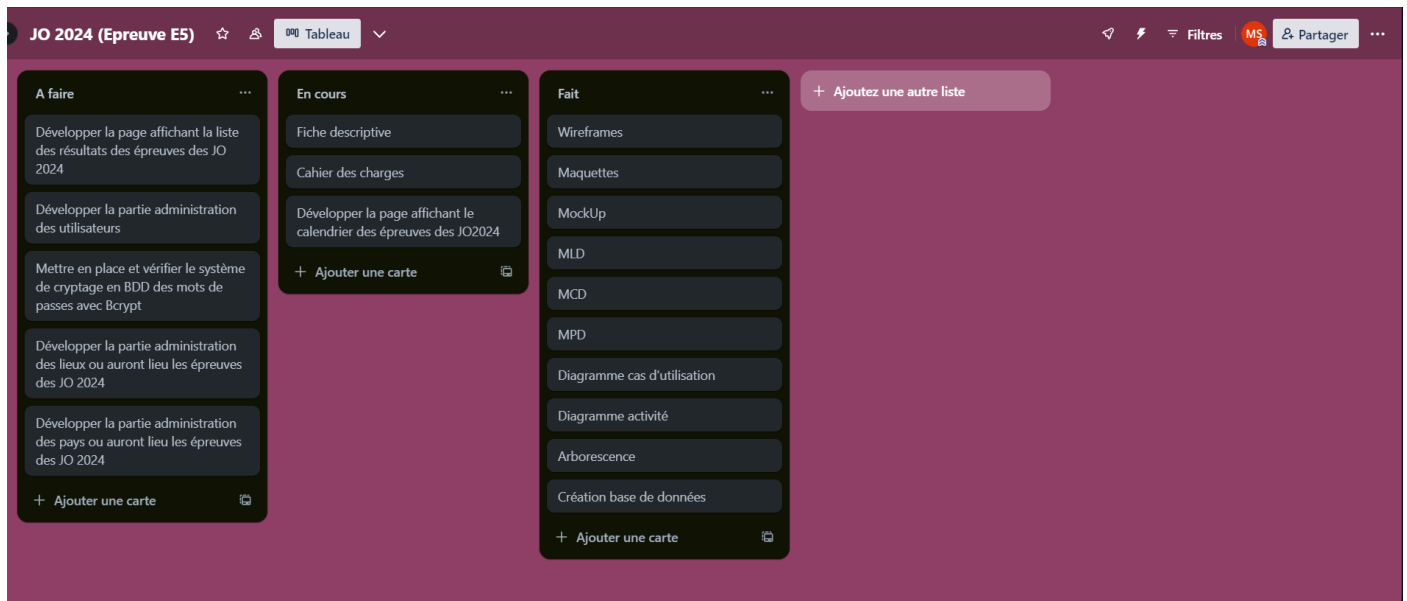
3.2. Ressources logicielles

Les ressources logicielles nécessaires à la réalisation du projet sont :

- Visual Studio Code
- MAMP
- GitHub
- Trello.

4. Gestion du projet

Pour réaliser le projet, nous utiliserons la méthode Agile Kanban. Nous utiliserons également l'outil de gestion de projet en ligne Trello.



Nous travaillons également sur GitHub, plateforme de développement collaboratif.

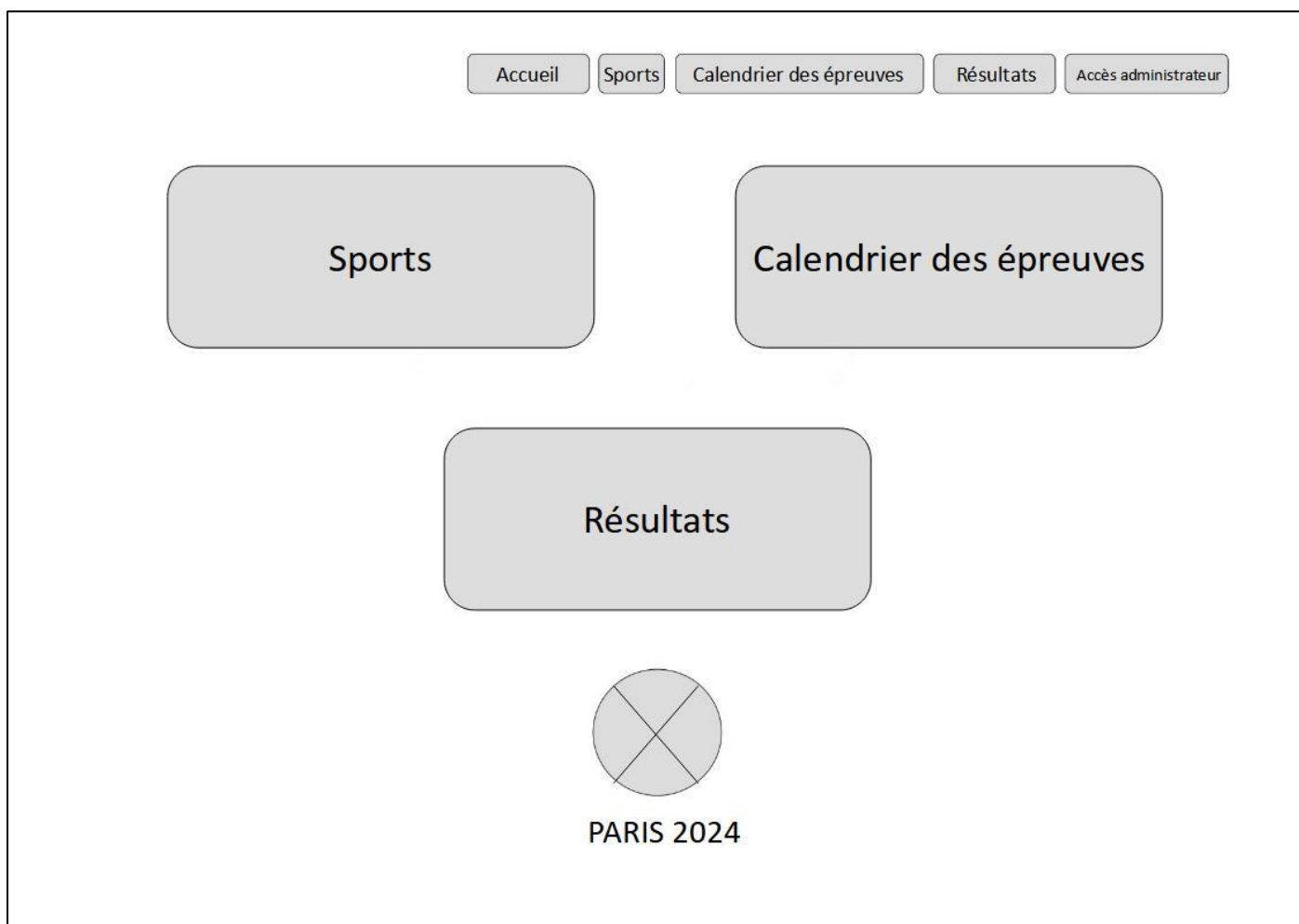
5. Conception du projet

5.1. Le front-end

5.1.1. Wireframes

Wireframes version ordinateur :

Page index :



Page connexion :

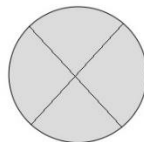
[Accueil](#)[Sports](#)[Calendrier des événements](#)[Résultats](#)[Accès administrateur](#)

Connexion

Login :

Mot de passe :

Se connecter



PARIS 2024

Page accueil administrateur :

[Accueil Administration](#)[Gestion Sports](#)[Gestion Lieux](#)[Gestion Lieux](#)[Gestion Pays](#)[Gestion Genres](#)[Gestion Athlètes](#)[Gestion Résultats](#)[Déconnexion](#)

Bonjour Super Admin

Gestion Administrateurs

Gestion Sports

Gestion Lieux

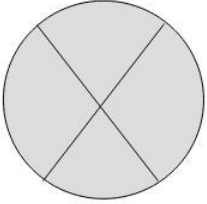
Gestion Calendrier

Gestion Pays

Gestion Genres

Gestion Athlètes

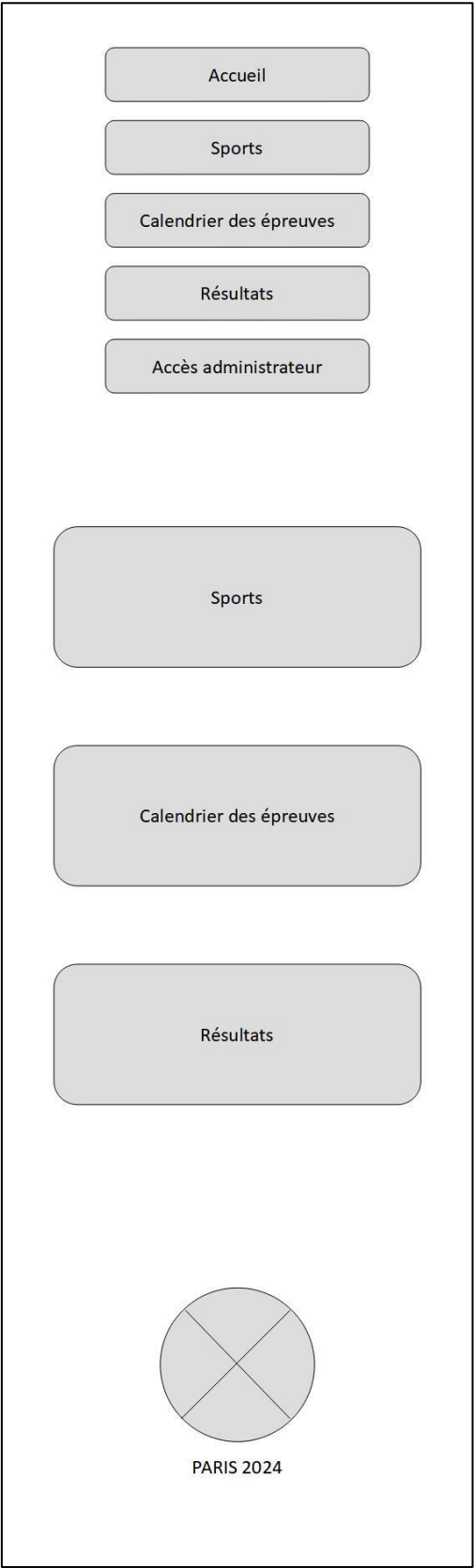
Gestion Résultats



PARIS 2024

Wireframes version responsive :

Page index :



Accueil

Sports

Calendrier des évènements

Résultats

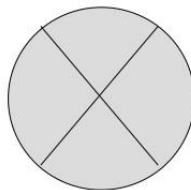
Accès administrateur

Connexion

Login :

Mot de passe :

Se connecter



PARIS 2024

Page accueil administrateur :

Accueil Administration

Gestion Sports

Gestion Lieux

Gestion Calendrier

Gestion Pays

Gestion Genres

Gestion Athlètes

Gestion Résultats

Déconnexion

Bonjour Super Admin

Gestion Administrateurs

Gestion Sports

Gestion Lieux

Gestion Calendrier

Gestion Pays

Gestion Genres

Gestion Athlètes

Gestion Résultats



PARIS 2024

5.1.2. Maquettes

Maquette version ordinateur :

Page index :



[Accueil](#)[Sports](#)[Calendrier des évènements](#)[Résultats](#)[Accès administrateur](#)

Connexion

Login :

Mot de passe :

Se connecter



Accueil
Administration

Gestion
Sports

Gestion
Lieux

Gestion
Calendrier

Gestion
Pays

Gestion
Genres

Gestion
Athlètes

Gestion
Résultats

Déconnexion

Bonjour Super Admin

Gestion Administrateurs

Gestion Sports

Gestion Lieux

Gestion Calendrier

Gestion Pays

Gestion Genres

Gestion Athlètes

Gestion Résultats



Maquette version responsive :

Page index :



Accueil

Sports

Calendrier des évènements

Résultats

Accès administrateur

Connexion

Login :

Mot de passe :

Se connecter



PARIS 2024



Page accueil administrateur :

[Accueil Administration](#)
[Gestion Sports](#)
[Gestion Lieux](#)
[Gestion Calendrier](#)
[Gestion Pays](#)
[Gestion Genres](#)
[Gestion Athlètes](#)
[Gestion Résultats](#)
[Déconnexion](#)

Bonjour Super Admin

[Gestion Administrateurs](#)

[Gestion Sports](#)

[Gestion Lieux](#)

[Gestion Calendrier](#)

[Gestion Pays](#)

[Gestion Genres](#)

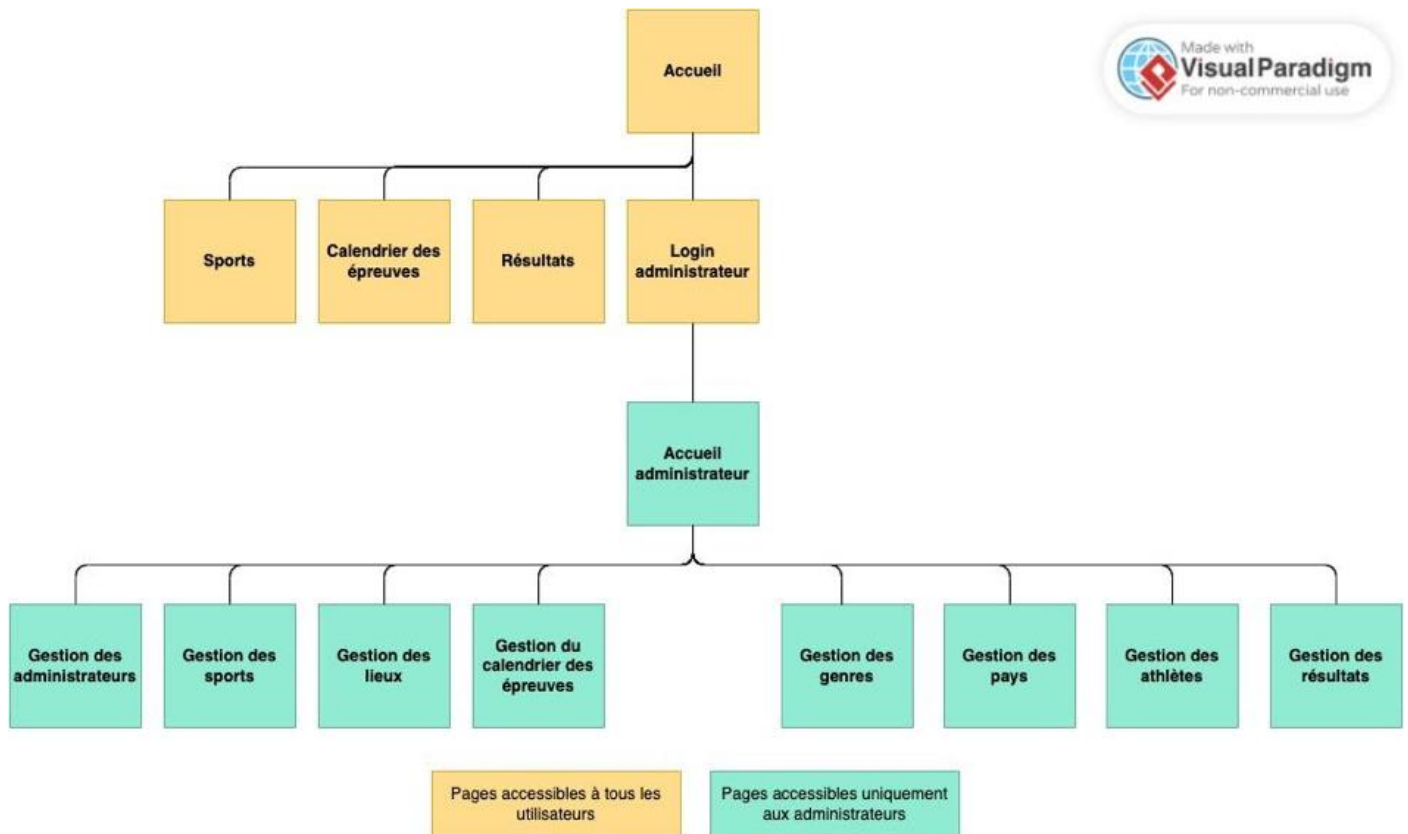
[Gestion Athlètes](#)

[Gestion Résultats](#)



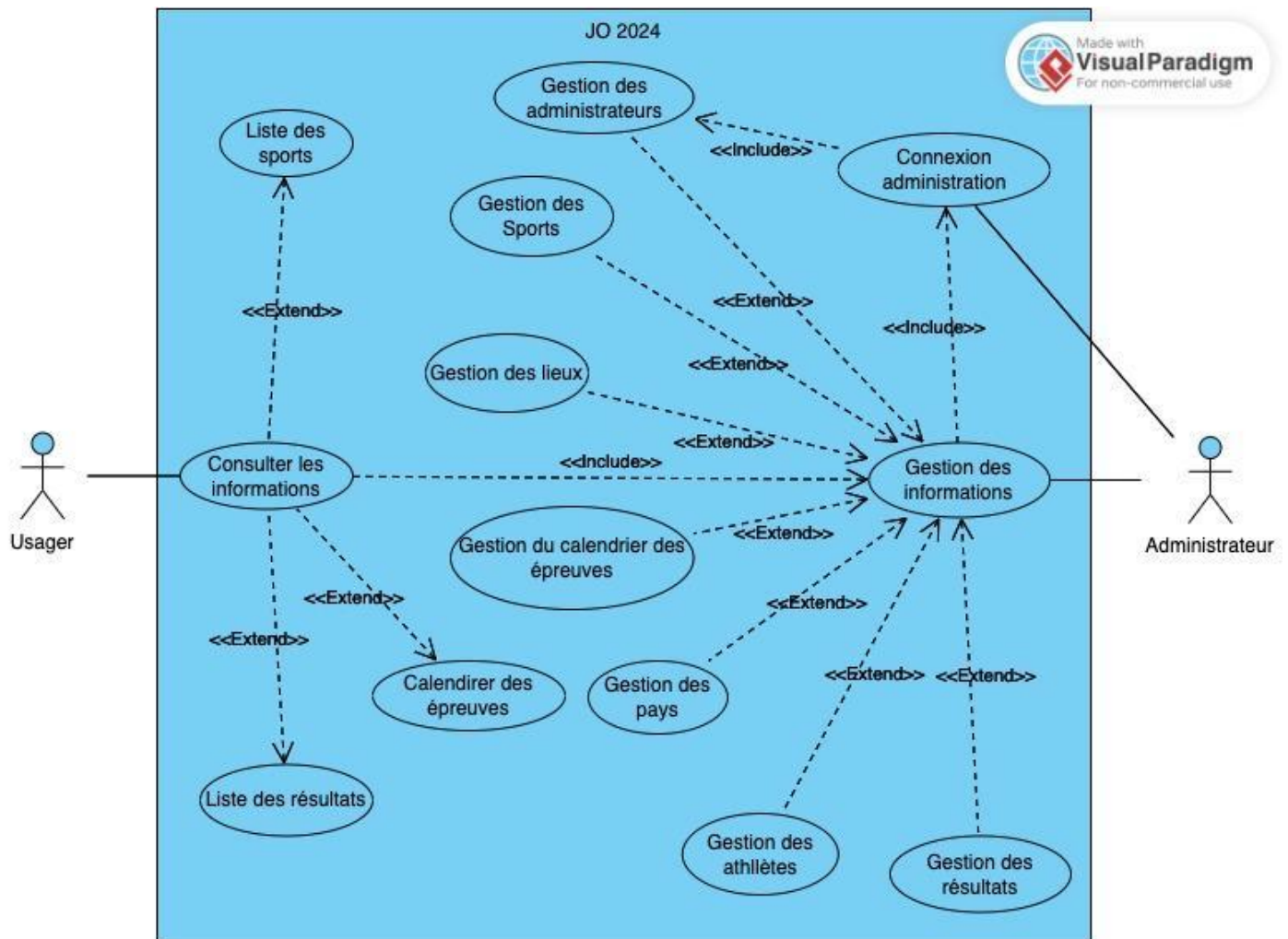
PARIS 2024

5.1.3. Arborescences

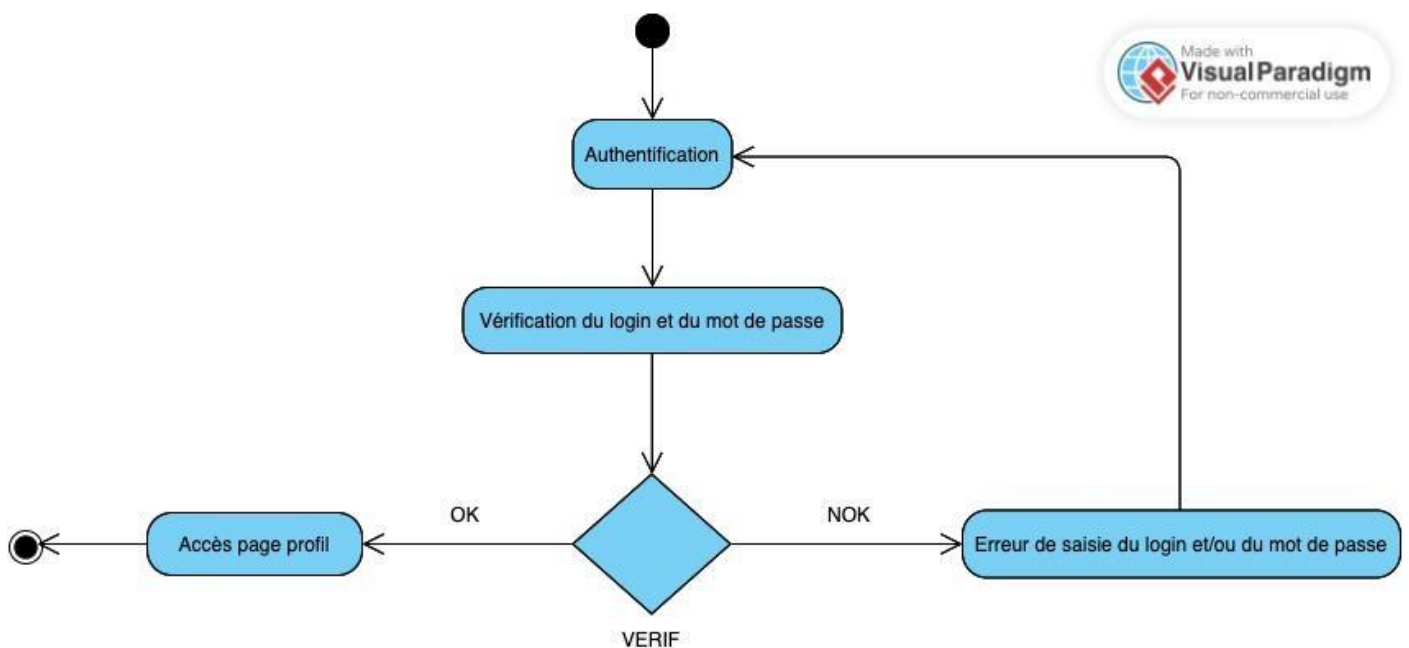


5.2. Le back-end

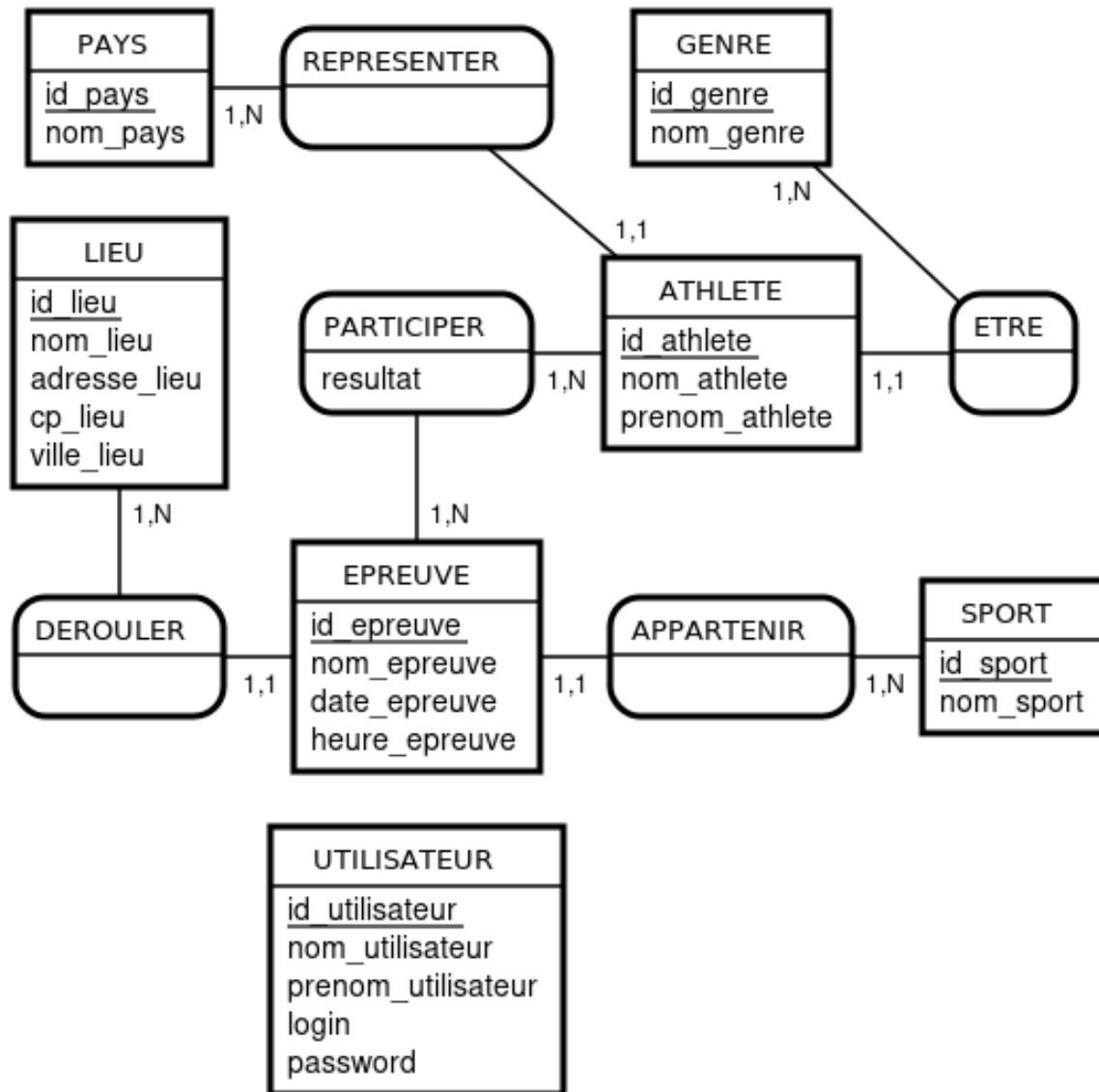
5.2.1. Diagramme de cas d'utilisation



5.2.2. Diagramme d'activités



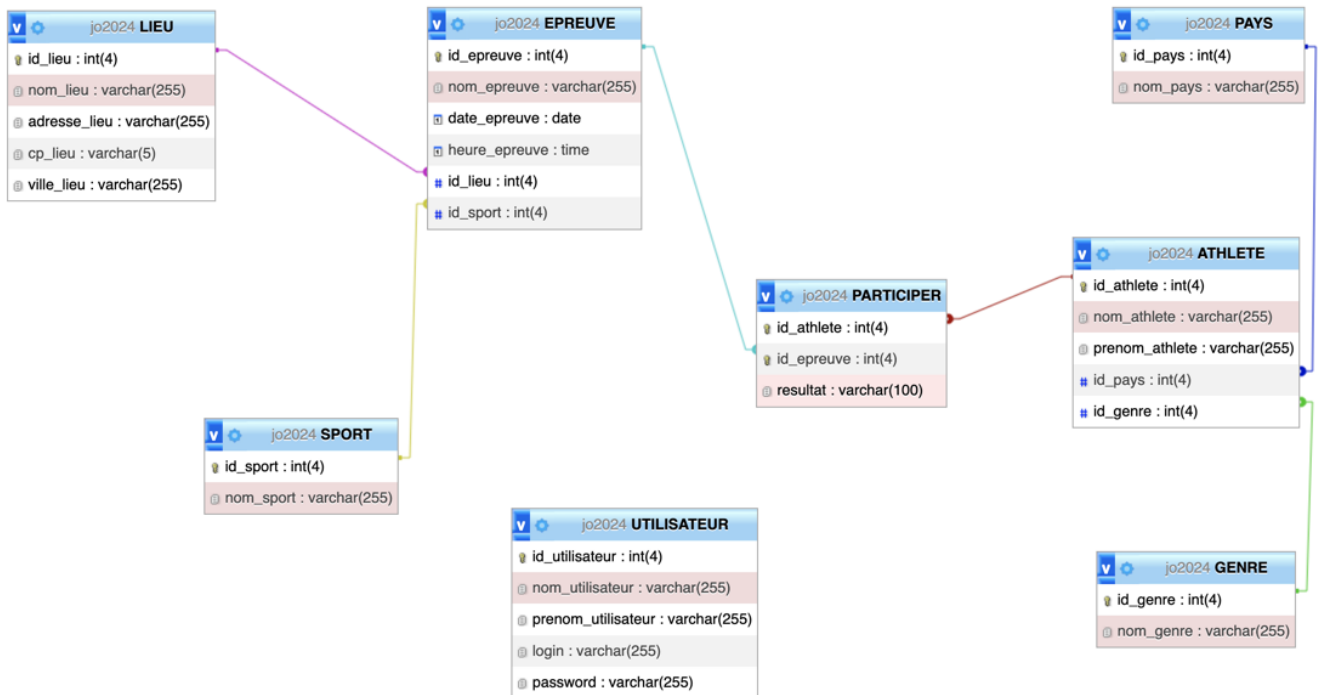
5.2.3. Modèles Conceptuel de Données (MCD)



5.2.4. Modèle Logique de Données (MLD)

Entité ou association	Libellé de l'attribut	Type
ATHLETE	id_athlete	INT(4)
/	nom_athlete	VARCHAR(255)
/	prenom_athlete	VARCHAR(255)
EPREUVE	id_epreuve	INT(4)
/	nom_epreuve	VARCHAR(255)
/	date_epreuve	DATE
/	heure_epreuve	TIME
GENRE	id_genre	INT(4)
/	nom_genre	VARCHAR(255)
LIEU	id_lieu	INT(4)
/	nom_lieu	VARCHAR(255)
/	adresse_lieu	VARCHAR(255)
/	cp_lieu	VARCHAR(5)
/	ville_lieu	VARCHAR(255)
PARTICIPER	resultat	VARCHAR(100)
PAYS	id_pays	INT(4)
/	nom_pays	VARCHAR(255)
SPORT	id_sport	INT(4)
/	nom_sport	VARCHAR(255)
UTILISATEUR	id_utilisateur	
	nom_utilisateur	
	prenom_utilisateur	
	login	
	mdp	

5.2.5. Modèle Physique de Données (MPD)



6. Technologies utilisées

6.1. Langages de développement Web

Les langages de développement web utilisé pour le projet sont :

- HTML
- CSS
- PHP
- JavaScript

6.2. Base de données

La base de données utilisée est MySQL.

7. Sécurité

7.1. Login et protection des pages administrateurs

La méthode POST permet de récupérer nos identifiants et mots de passes, par la suite il faut vérifier l'identifiant et le mot de passe, puis s'ils sont corrects une redirection vers la page administrateur est faite puis la session est lancée. En revanche si les identifiants et mots de passes sont erronés alors une redirection vers la page index est faite avec un message invitant à réessayer.

```
1 <?php
2 session_start(); // Démarre la session PHP pour stocker des variables de session.
3
4 require_once("database.php"); // Inclut le fichier de connexion à la base de données.
5
6 if ($_SERVER["REQUEST_METHOD"] == "POST") { // Vérifie si la requête est une méthode POST (formulaire soumis).
7     $login = $_POST["login"]; // Récupère la valeur du champ "login" du formulaire.
8     $password = $_POST["password"]; // Récupère la valeur du champ "password" du formulaire.
9
10    // Prépare la requête SQL pour récupérer les informations de l'utilisateur avec le login spécifié.
11    $query = "SELECT id_utilisateur, nom_utilisateur, prenom_utilisateur, login, password FROM UTILISATEUR WHERE login = :login";
12    $stmt = $connexion->prepare($query); // Prépare la requête avec PDO.
13    $stmt->bindParam(":login", $login, PDO::PARAM_STR); // Lie la variable :login à la valeur du login, évitant les injections SQL.
14
15    if ($stmt->execute()) { // Exécute la requête préparée.
16        $row = $stmt->fetch(PDO::FETCH_ASSOC); // Récupère la première ligne de résultat de la requête.
17
18        if ($row && password_verify($password, $row["password"])) {
19            // Si une ligne est récupérée et le mot de passe correspond à celui stocké dans la base de données.
20            $_SESSION["id_utilisateur"] = $row["id_utilisateur"]; // Stocke l'ID utilisateur dans la session.
21            $_SESSION["nom_utilisateur"] = $row["nom_utilisateur"]; // Stocke le nom de l'utilisateur dans la session.
22            $_SESSION["prenom_utilisateur"] = $row["prenom_utilisateur"]; // Stocke le prénom de l'utilisateur dans la session.
23            $_SESSION["login"] = $row["login"]; // Stocke le login de l'utilisateur dans la session.
24
25            header("location: ../pages/admin/admin.php"); // Redirige vers la page d'administration.
26            exit(); // Termine le script.
27        } else {
28            $_SESSION['error'] = "Login ou mot de passe incorrect.";
29            header("location: ../pages/login.php"); // Redirige vers la page de login avec un message d'erreur.
30        }
31    } else {
32        $_SESSION['error'] = "Erreur lors de l'exécution de la requête.";
33        header("location: ../pages/login.php"); // Redirige vers la page de login avec un message d'erreur.
34    }
35
36    unset($stmt); // Libère la ressource associée à la requête préparée.
37 }
```

7.2. Cryptage des mots de passe avec Bcrypt

Bcrypt est un algorithme de hachage de mots de passe conçu pour être lent et résistant aux attaques par force brute. Il est particulièrement utilisé pour stocker de manière sécurisée les mots de passe dans les bases de données.

```
1 <?php
2 session_start();
3 require_once("../../database/database.php");
4
5 // Vérifiez si l'utilisateur est connecté
6 if (!isset($_SESSION['login'])) {
7     header('Location: ../../index.php');
8     exit();
9 }
10
11 // Vérifiez si le formulaire est soumis
12 if ($_SERVER["REQUEST_METHOD"] == "POST") {
13     // Assurez-vous d'obtenir des données sécurisées et filtrées
14     $nom_utilisateur = filter_input(INPUT_POST, 'nomUtilisateur', FILTER_SANITIZE_STRING);
15     $prenom_utilisateur = filter_input(INPUT_POST, 'prenomUtilisateur', FILTER_SANITIZE_STRING);
16     $login_utilisateur = filter_input(INPUT_POST, 'loginUtilisateur', FILTER_SANITIZE_STRING);
17     $mdp_utilisateur = filter_input(INPUT_POST, 'mdpUtilisateur', FILTER_SANITIZE_STRING);
18
19     // Vérifiez si le nom d'utilisateur est vide
20     if (empty($nom_utilisateur)) {
21         $_SESSION['error'] = "Le nom de l'utilisateur ne peut pas être vide.";
22         header("Location: add-users.php");
23         exit();
24     }
25
26     try {
27         // Vérifiez si le nom d'utilisateur existe déjà
28         $queryCheck = "SELECT id_utilisateur FROM UTILISATEUR WHERE nom_utilisateur = :nomUtilisateur";
29         $statementCheck = $connexion->prepare($queryCheck);
30         $statementCheck->bindParam(":nomUtilisateur", $nom_utilisateur, PDO::PARAM_STR);
31         $statementCheck->execute();
32
33         if ($statementCheck->rowCount() > 0) {
34             $_SESSION['error'] = "Le nom d'utilisateur existe déjà.";
35             header("Location: add-users.php");
36             exit();
37         } else {
```

7.3. Protection contre les attaques XSS (Cross-Site Scripting)

Les attaques XSS surviennent lorsqu'un attaquant injecte du code JavaScript malveillant dans les pages web consultées par d'autres utilisateurs. On utilise la fonction `htmlspecialchars()` de cette manière : même si l'utilisateur entre du code JavaScript malveillant dans le champ de saisie, celui-ci sera affiché comme du texte brut dans la page, sans être interprété comme du code exécutable par le navigateur.

7.4. Protection contre les injections SQL

Une injection SQL est une façon d'accéder et d'effectuer sans autorisation des opérations sur une base de données en injectant une requête SQL, la plupart du temps cette injection se fait via un formulaire. Afin de pallier ce problème, on utilise PDO. Ce sont des requêtes préparées qui permettent d'éviter les injections SQL.