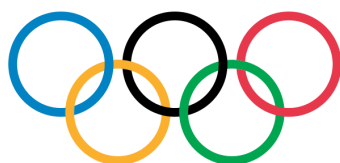


CAHIER DES CHARGES TECHNIQUE

Projet JO 2024



PARIS 2024



Dossier rédigé par Pierre-Alix BOMET

SOMMAIRE

- 1. Contexte du projet (p.3)**
 - 1.1. Présentation du projet (p.3)
 - 1.2. Date de rendu du projet (p.3)
- 2. Besoin fonctionnels (p.3)**
- 3. Ressources nécessaires à la réalisation du projet (p.3 - 4)**
 - 3.1. Ressources matérielles (p.3)
 - 3.2. Ressources logicielles (p.4)
- 4. Gestion du projet (p.4-5)**
- 5. Conception du projet (p.6 - 11)**
 - 5.1. Le Front-end (p.5)**
 - 5.1.1. Wireframes (p.5-6)
 - 5.1.2. Maquettes (p.7-8)
 - 5.1.3. Arborescences (p.8)
 - 5.2. Le Back-end (p.9)**
 - 5.2.1. Diagramme de cas d'utilisation (p.9)
 - 5.2.2. Diagramme d'activités (p.9)
 - 5.2.3. Dictionnaire de Données (p.10)
 - 5.2.4. Modèle Conceptuel de Données (MCD) (p.11)
 - 5.2.5. Modèle Logique de Données (MLD) (p.11)
 - 5.2.6. Modèle Physique de Données (MPD) (p.12)
- 6. Technologies utilisées (p.12)**
 - 6.1. Langages de développement Web (p.12)
 - 6.2. Base de données (p.12)
- 7. Sécurité (p.13 - 15)**
 - 7.1. Login et protection des pages administrateurs (p.13)
 - 7.2. Cryptage des mots de passe avec Bcrypt (p.14)
 - 7.3. Protection contre les attaques XSS (Cross-Site Scripting) (p.14)
 - 7.4. Protection contre les injections SQL (p.15)

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 comprennent l'ordinateur ainsi que tous ses périphériques tels que les écrans, claviers et souris.

3.2. Ressources logicielles

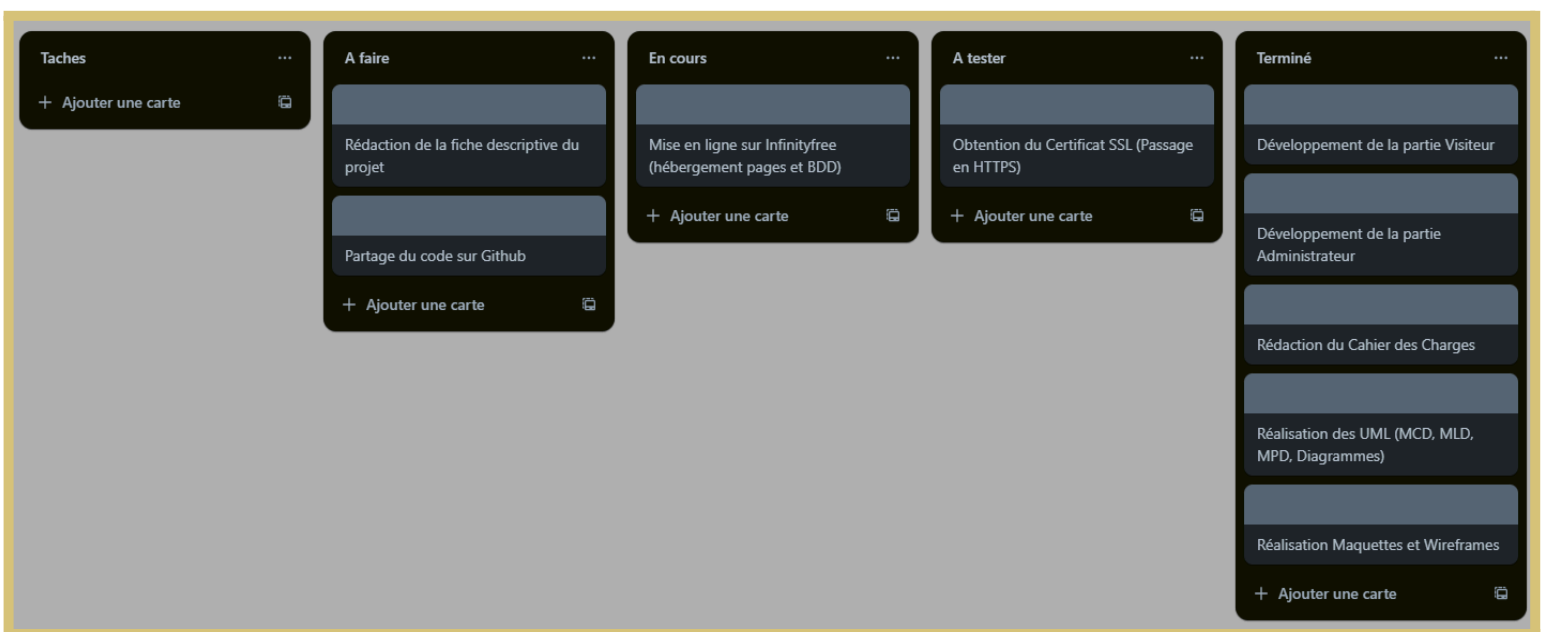
Les ressources logicielles nécessaires comprennent :

- l'IDE à savoir Visual Studio Code nécessaire à la programmation HTML, CSS et PHP de notre application
- MAMP qui contient le système de gestion de base de données **MySQL** et le serveur **Apache** permettant de communiquer avec PHP
- Trello , un outil de gestion de projet
- Github, la plateforme de développement

4. Gestion de projet

Le projet sera cadré à l'aide de la méthode agile Kanban. Le Kanban Board réalisé sur Trello. On utilisera également un diagramme de Gantt.

Ils permettront tous deux de mieux visualiser et organiser l'enchaînement des tâches.



Kanban Board

NUMÉRO	TITRE DE LA TÂCHE	DATE DE DÉBUT	DATE LIMITE	DURÉE	TÂCHE TERMINÉE (EN %)															
						SEMAINE 1					SEMAINE 2					SEMAINE 3				
						L	M	M	J	V	L	M	M	J	V	L	M	M	J	V
1	UML																			
1.1	Diagramme de cas d'utilisation	01/12/23	01/12/23	1	100 %															
1.1.1	Diagramme d'activités	01/12/23	01/12/23	1	100 %															
1.2	MCD	01/12/23	01/12/23	1	90 %															
1.3	MLD	01/12/23	01/12/23	1	100 %															
1.4	MPD	01/12/23	02/12/23	1	100 %															
1.5	Rédaction du cahier des charges	02/12/23	16/12/23	14	100 %															
2	Développement																			
2.1	Partie Visiteur	02/12/23	04/12/23	3	100 %															
2.2	Partie Administrateur	05/12/23	12/12/23	7	100 %															
3	Mise en ligne																			
3.1	Partage de code GitHub	13/12/23	13/12/23	1	100 %															
3.2	Mise en ligne sur Infinityfree	13/12/23	13/12/23	1	50 %															
3.2.1	Obtention du certificat SSL	13/12/23	15/12/23	3	0 %															

Diagramme de Gantt

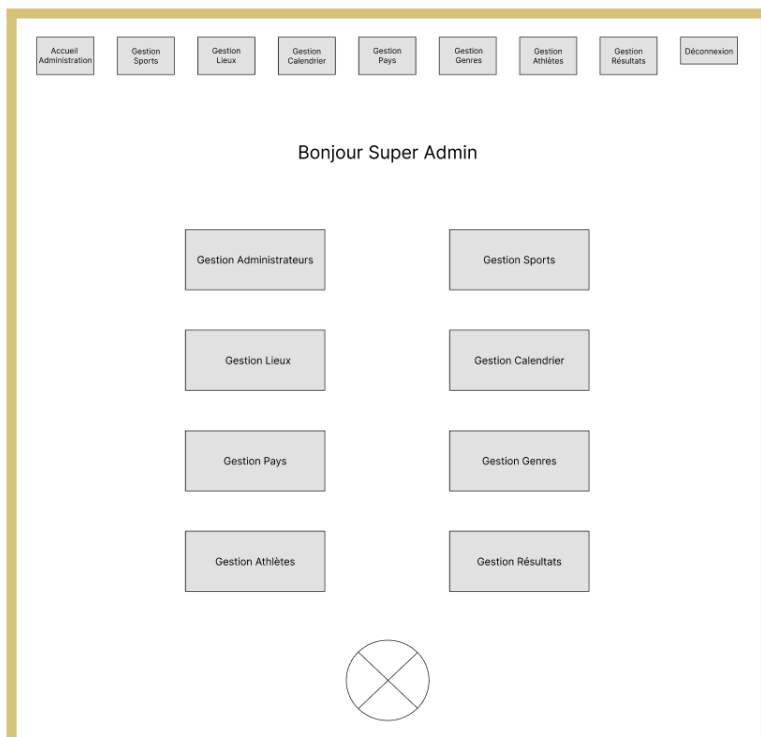
5. Conception du projet

5.1. Le Front-End

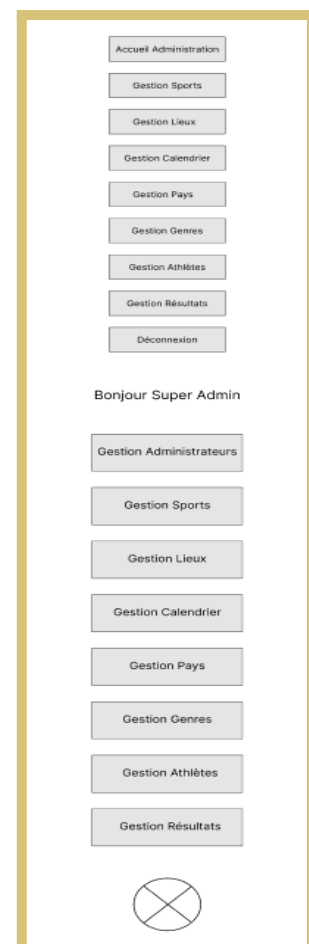
5.1.1. Wireframes

Afin de concevoir notre application, la réalisation de wireframes est essentielle et permettra de déterminer la structure de son interface utilisateur.

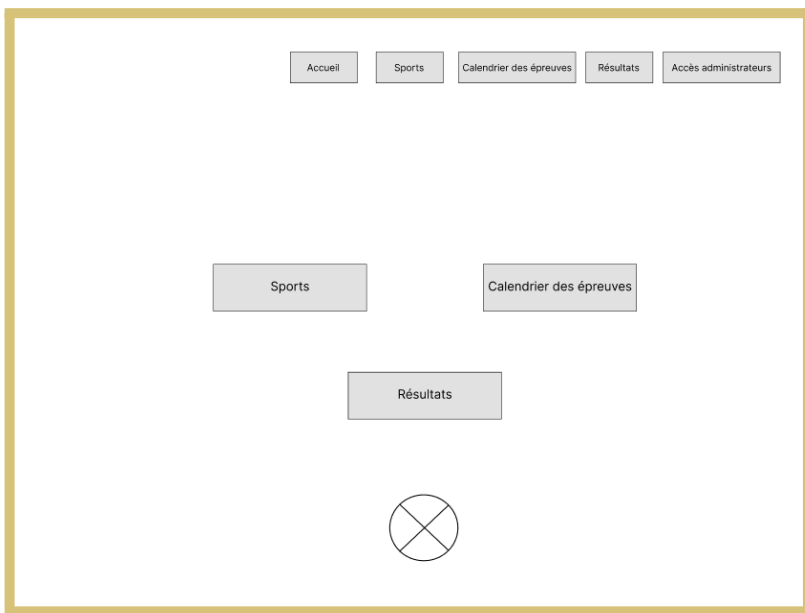
Accueil admin (Ordinateur)



Accueil admin (Responsive)



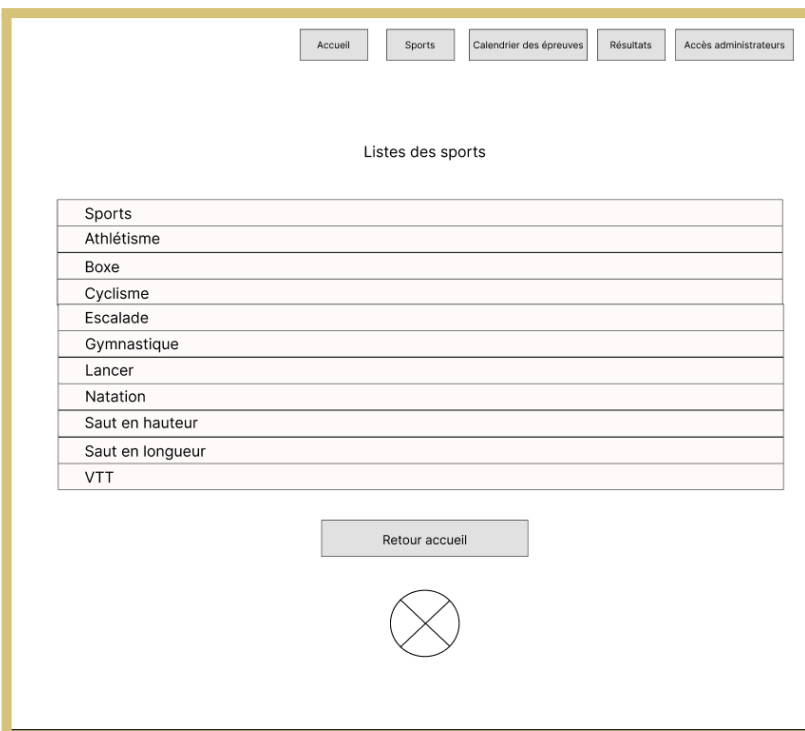
Index (Ordinateur)



Index (Responsive)



Liste Sports (Ordinateur)

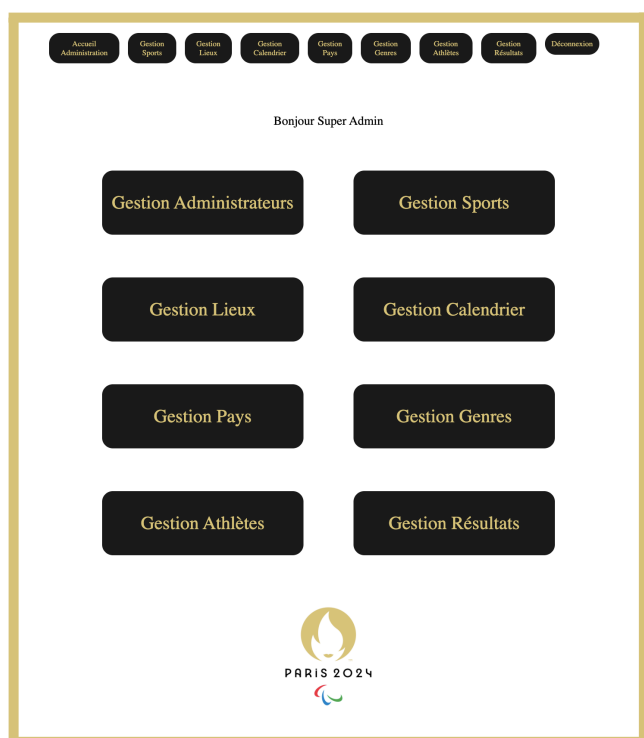


Liste Sports (Responsive)

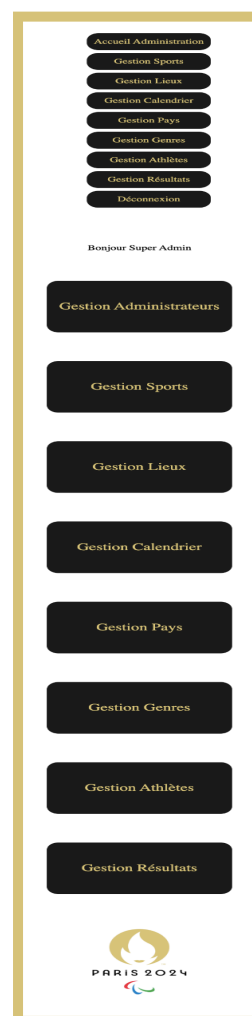


5.1.2. Maquettes

Accueil admin (Ordinateur)



Accueil admin (Responsive)



Index (Ordinateur)



Index (Responsive)




Liste Sports (Ordinateur)

AccueilSportsCalendrier des évènementsRésultatsAccès administrateur

Liste des Sports

Sport
Athlétisme
Boxe
Cyclisme
Escalade
Gymnastique
Lancer
Natation
Saut en hauteur
Saut en longueur
VTT

Retour Accueil



Liste Sports (Responsive)

AccueilSportsCalendrier des évènementsRésultatsAccès administrateur

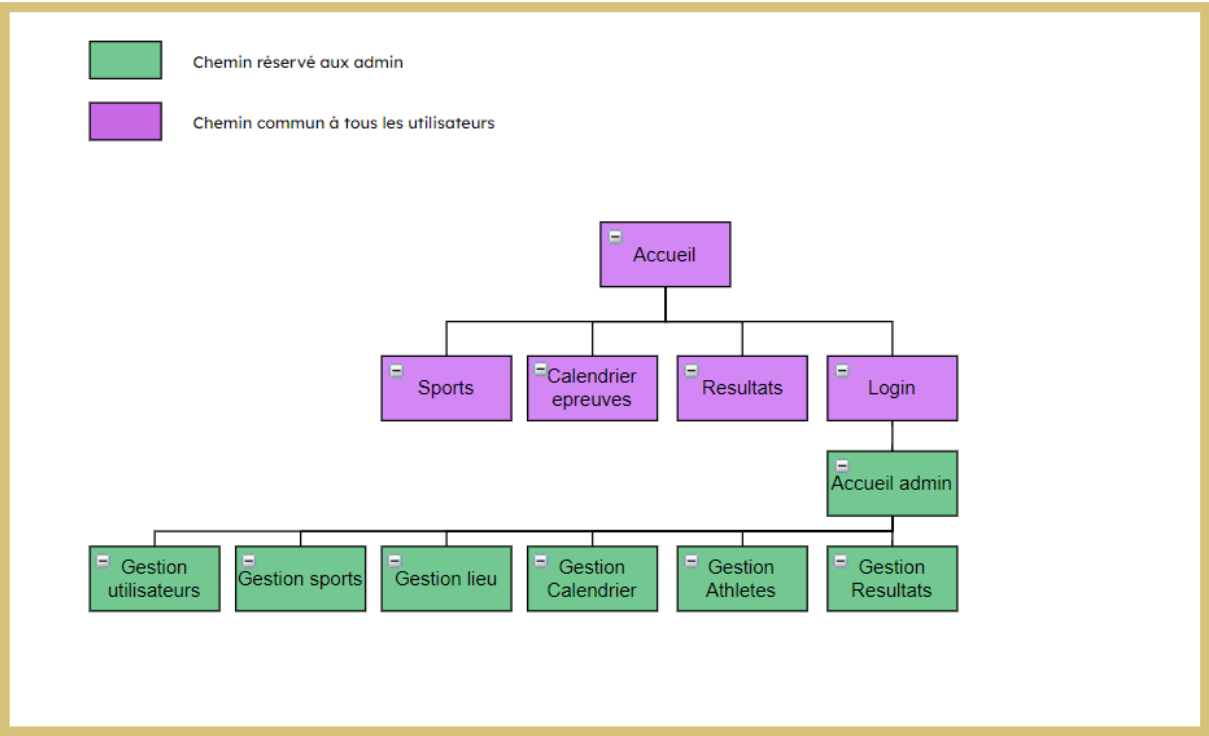
Liste des Sports

Sport
Athlétisme
Boxe
Cyclisme
Escalade
Gymnastique
Lancer
Natation
Saut en hauteur
Saut en longueur
VTT

Retour Accueil

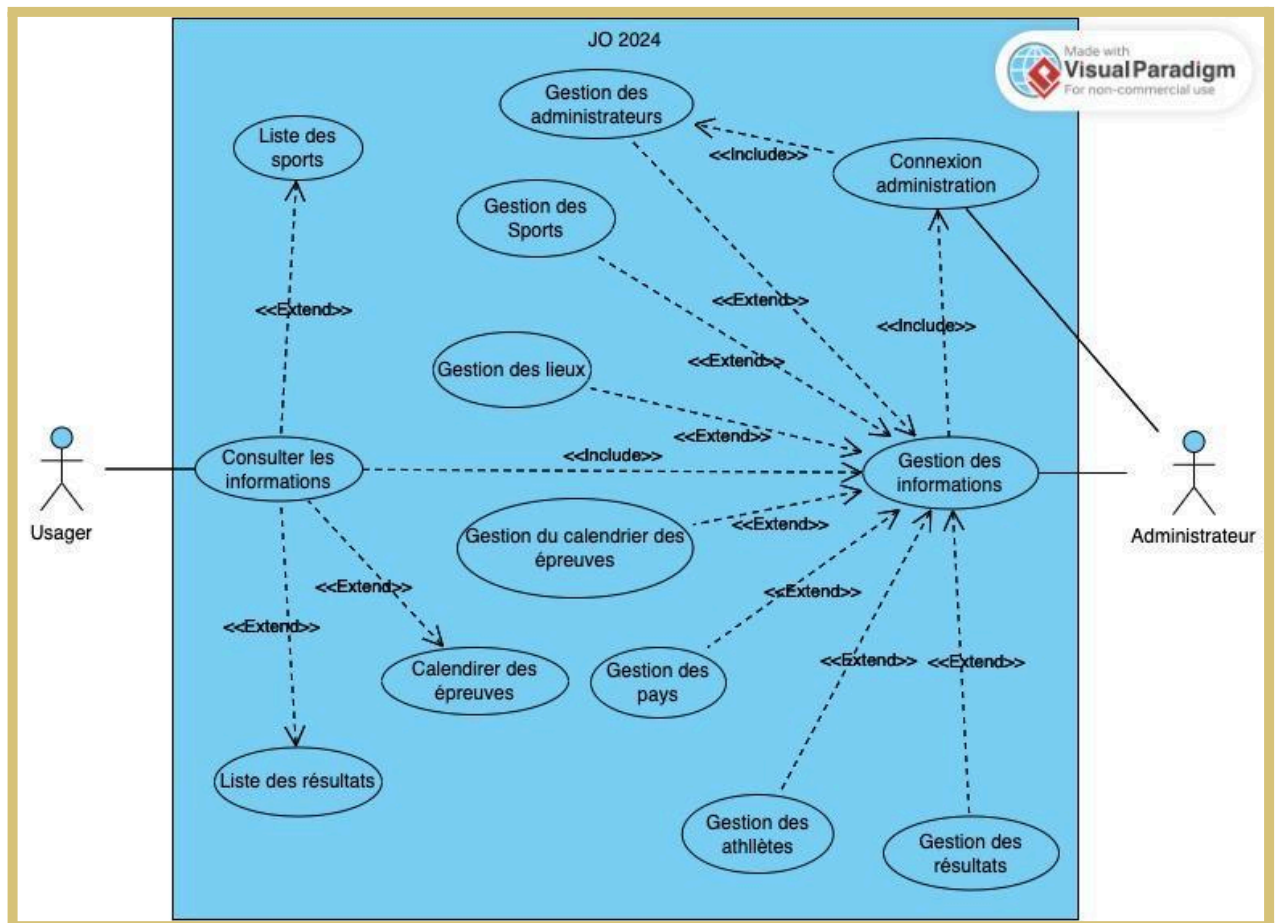


5.1.3. Arborescence

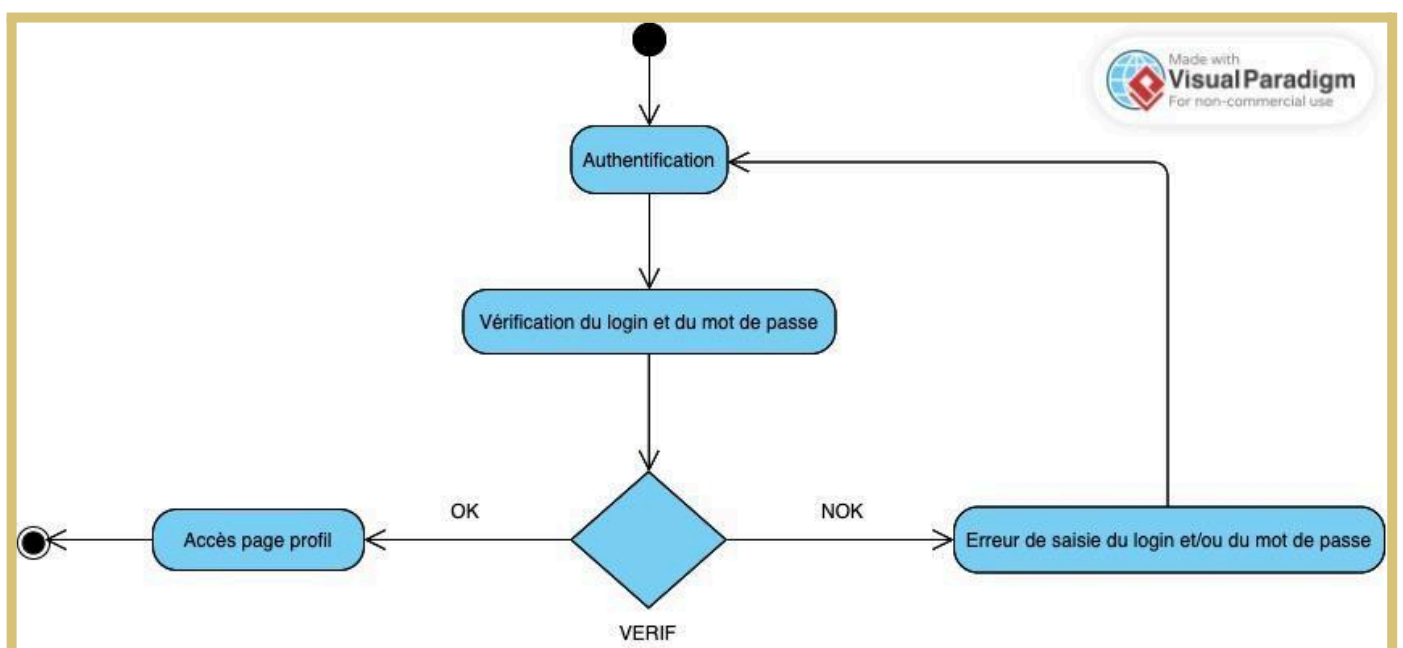


5.2. Le Back-End

5.2.1. Diagramme de cas d'utilisation



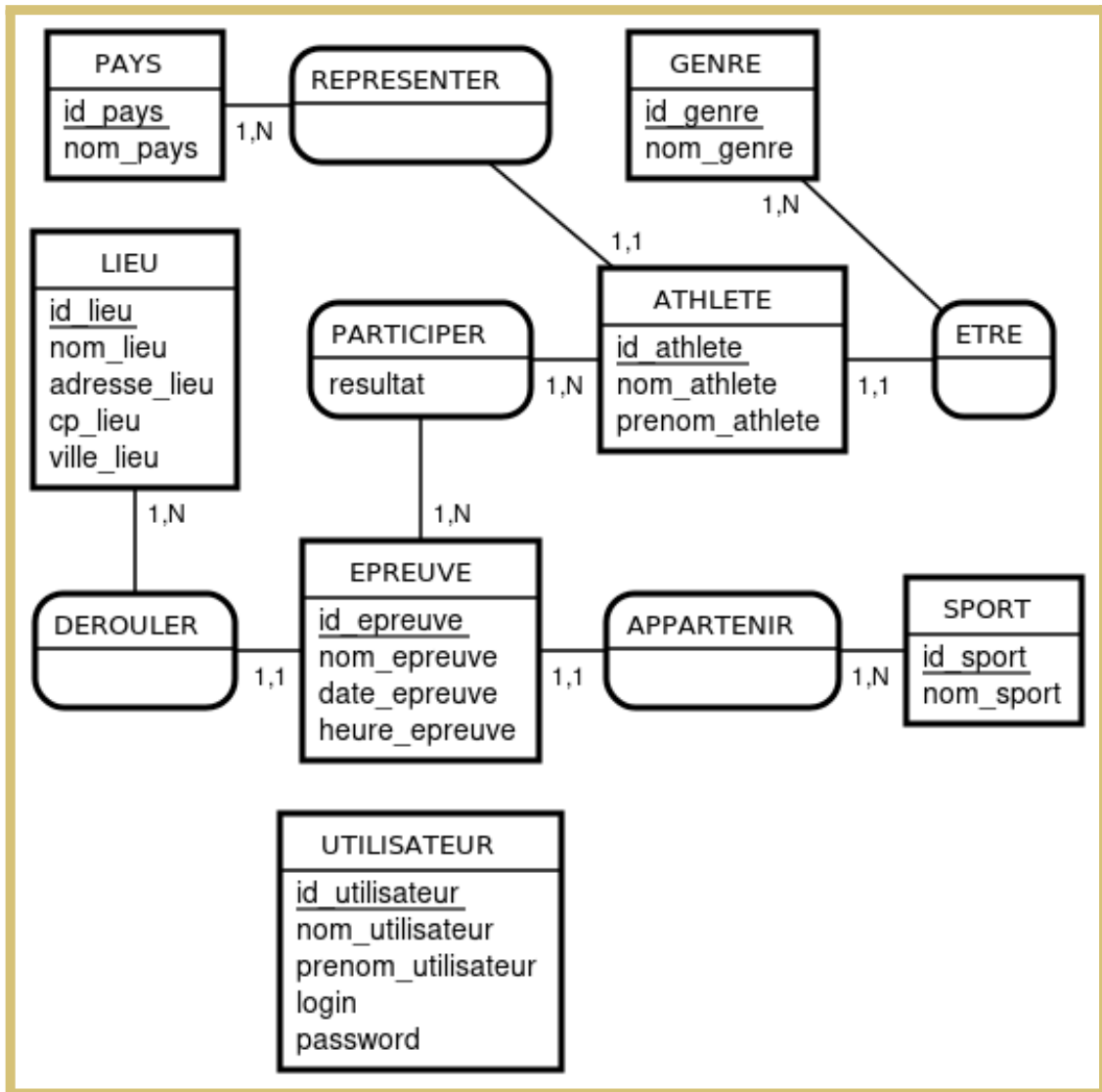
5.2.2. Diagramme d'activités



5.2.3. Dictionnaire de Données

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	INT(4)
/	nom_utilisateur	VARCHAR(255)
/	prenom_utilisateur	VARCHAR(255)
/	login	VARCHAR(255)
/	password	VARCHAR(255)

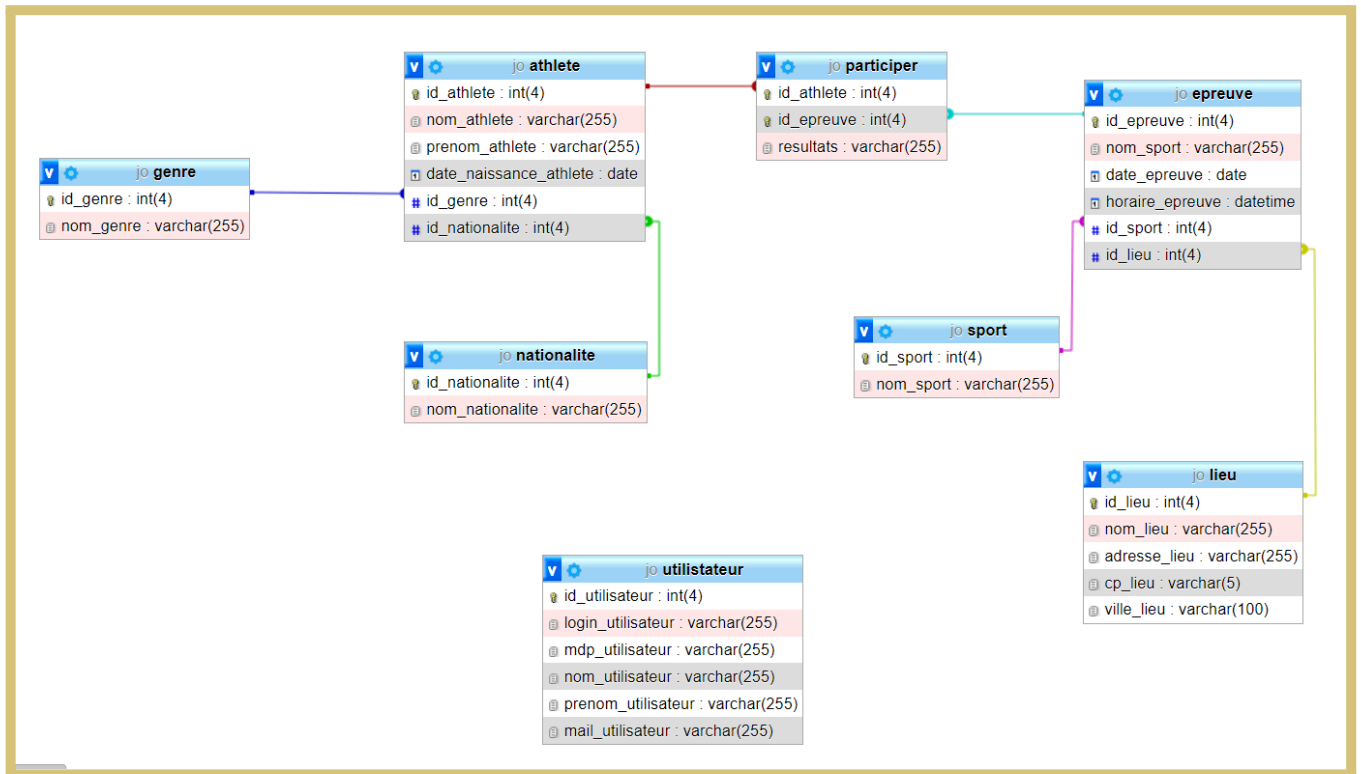
5.2.4. Modèle conceptuel de données (MCD)



5.2.5. Modèle logique de données (MLD)

- **ATHLETE** (*id_athlete, nom_athlete, prenom_athlete, #id_pays, #id_genre*)
- **EPREUVE** (*id_epreuve, nom_epreuve, date_epreuve, heure_epreuve, #id_lieu, #id_sport*)
- **GENRE** (*id_genre, nom_genre*)
- **LIEU** (*id_lieu, nom_lieu, adresse_lieu, cp_lieu, ville_lieu*)
- **PARTICIPER** (*#id_athlete, #id_epreuve, resultat*)
- **PAYS** (*id_pays, nom_pays*)
- **SPORT** (*id_sport, nom_sport*)
- **UTILISATEUR** (*id_utilisateur, nom_utilisateur, prenom_utilisateur, login, password*)

5.2.6. Modèle physique de données (MPD)



6. Technologies utilisées

6.1. Langages de développement Web

Les langages utilisés pour la partie web comprennent :

- HTML 5
- CSS 3
- JavaScript

Les langages utilisés pour les interactions avec la base de données sont :

- PHP 8
- SQL

6.2. Base de données

La base de données utilisée est MySQL. On utilisera le langage SQL.

7. Sécurité

7.1. Login et protection des pages administrateurs

Les pages administrateurs seront protégées et accessibles uniquement à l'aide d'un login et d'un mot de passe.

Voici le formulaire utilisé pour la connexion à la partie administrateur :

```
<h1>Connexion</h1>
<form action="../../database/auth.php" method="post">
  <label for="login">Login :</label>
  <input type="text" name="login" id="login" required><br><br>
  <label for="password">Mot de passe :</label>
  <input type="password" name="password" id="password" required><br><br>
  <input type="submit" value="Se connecter">
</form>
```

Les informations du formulaire sont ensuite récupérées et stockées afin de pouvoir authentifier l'utilisateur et lui octroyer ou non l'accès :

```
// Prépare la requête SQL pour récupérer les informations de l'utilisateur avec le login spécifié.
$query = "SELECT id_utilisateur, nom_utilisateur, prenom_utilisateur, login, password FROM UTILISATEUR WHERE login = :login";
$stmt = $connexion->prepare($query); // Prépare la requête avec PDO.
$stmt->bindParam(":login", $login, PDO::PARAM_STR); // Lie la variable :login à la valeur du login, évitant les injections SQL.

if ($stmt->execute()) { // Exécute la requête préparée.
    $row = $stmt->fetch(PDO::FETCH_ASSOC); // Récupère la première ligne de résultat de la requête.

    if ($row && password_verify($password, $row["password"])) {
        // Si une ligne est récupérée et le mot de passe correspond à celui stocké dans la base de données.
        $_SESSION["id_utilisateur"] = $row["id_utilisateur"]; // Stocke l'ID utilisateur dans la session.
        $_SESSION["nom_utilisateur"] = $row["nom_utilisateur"]; // Stocke le nom de l'utilisateur dans la session.
        $_SESSION["prenom_utilisateur"] = $row["prenom_utilisateur"]; // Stocke le prénom de l'utilisateur dans la session.
        $_SESSION["login"] = $row["login"]; // Stocke le login de l'utilisateur dans la session.

        header("location: ../pages/admin/admin.php"); // Redirige vers la page d'administration.
        exit(); // Termine le script.
    } else {
        $_SESSION['error'] = "Login ou mot de passe incorrect.";
        header("location: ../pages/login.php"); // Redirige vers la page de login avec un message d'erreur.
    }
} else {
    $_SESSION['error'] = "Erreur lors de l'exécution de la requête.";
    header("location: ../pages/login.php"); // Redirige vers la page de login avec un message d'erreur.
}

unset($stmt); // Libère la ressource associée à la requête préparée.
}

unset($connexion); // Ferme la connexion à la base de données.

header("location: ../pages/login.php"); // Redirige vers la page de login par défaut.
```

7.2. Cryptage des mots de passe Bcrypt

Les mots de passe des utilisateurs seront hachés à l'aide de la fonction `password_hash()`. Cet algorithme bcrypt permettra d'éviter l'enregistrement du mot de passe en texte brut dans la base de données.

```
// Vérifiez si le formulaire est soumis
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Assurez-vous d'obtenir des données sécurisées et filtrées
    $nomUser = filter_input(INPUT_POST, 'nomUser', FILTER_SANITIZE_STRING);
    $prenomUser = filter_input(INPUT_POST, 'prenomUser', FILTER_SANITIZE_STRING);
    $loginUser = filter_input(INPUT_POST, 'loginUser', FILTER_SANITIZE_STRING);
    $pwUser = password_hash($_POST['pwUser'], PASSWORD_DEFAULT); // Hache le mot de passe
```

7.3. Protection contre les attaques XSS

Pour se protéger des attaques XSS, les entrées utilisateurs seront filtrées et validées côté serveur à l'aide de `htmlspecialchars` dans le but d'assurer leur correspondances aux attentes.

Les données seront également échappées avant d'être affichées sur le site. Ainsi, les caractères spéciaux (tels que <, >, ", ' ou encore &) ne seront pas interprétés comme du code HTML ou Javascript par le navigateur.

Voici un exemple de formulaire utilisé pour la modification d'un utilisateur :

```
?>
<form action="modify-user.php?id_utilisateur=<?php echo $id_utilisateur; ?>" method="post"
onsubmit="return confirm('Êtes-vous sûr de vouloir modifier cet utilisateur?')">
    <label for="prenomUser">Prénom de l'utilisateur :</label>
    <input type="text" name="prenomUser" id="prenomUser"
        value="<?php echo htmlspecialchars($user['prenom_utilisateur']); ?>" required>
    <label for="nomUser">Nom de l'utilisateur :</label>
    <input type="text" name="nomUser" id="nomUser"
        value="<?php echo htmlspecialchars($user['nom_utilisateur']); ?>" required>
    <input type="submit" value="Modifier l'Utilisateur">
</form>
```

7.4. Protection contre les injections SQL

Pour lutter contre les injections SQL, nous utiliserons des requêtes préparées et éviteront la concaténation directe de chaînes de caractères des entrées utilisateurs dans ces requêtes.

Voici un exemple de requête préparée utilisée pour vérifier si l'athlète que l'on veut ajouter existe déjà dans la base de données et éviter la présence de doublons :

```
// Vérifiez si l'athlete existe déjà
$queryCheck = "SELECT id_athlete FROM ATHLETE WHERE nom_athlete = :nomAthlete AND prenom_athlete = :prenomAthlete AND id_pays = :idPays AND id_genre = :idGenre";
$stmtCheck = $connexion->prepare($queryCheck);
$stmtCheck->bindParam(":nomAthlete", $nomAthlete, PDO::PARAM_STR);
$stmtCheck->bindParam(":prenomAthlete", $prenomAthlete, PDO::PARAM_STR);
$stmtCheck->bindParam(":idPays", $idPays, PDO::PARAM_INT);
$stmtCheck->bindParam(":idGenre", $idGenre, PDO::PARAM_INT);
$stmtCheck->execute();
```