CAHIER DES CHARGES TECHNIQUE

Projet Jeux Olympique 2024

Dossier rédiger par DISY Théo

2023-2024

Sommaire

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.

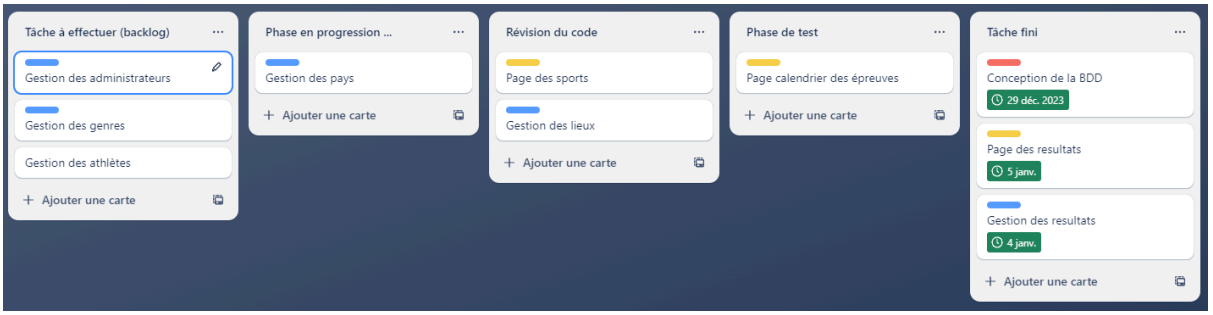
Les données seront stockées dans une base de données relationnelle pour faciliter la gestion et la mise à jour des informations. Ces données peuvent être gérées directement via le site web à travers un espace administrateur.

3. Ressources matérielles nécessaires à la réalisation du projet

Afin de pouvoir réaliser ce projet, des ressources matérielles sont nécessaires, nous aurons notamment besoin d'un ordinateur et des différents périphériques qui le composent (Souris, claviers et écran) et d'un ordinateur portable.

4. Gestion du projet

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



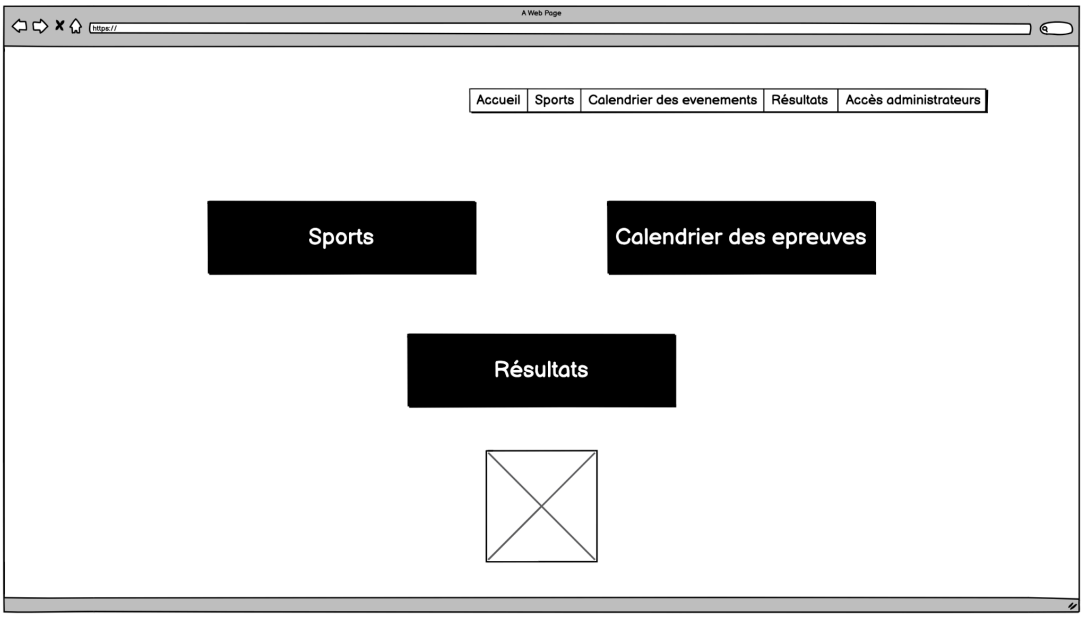
5. Conception du projet

5.1. Le front-end

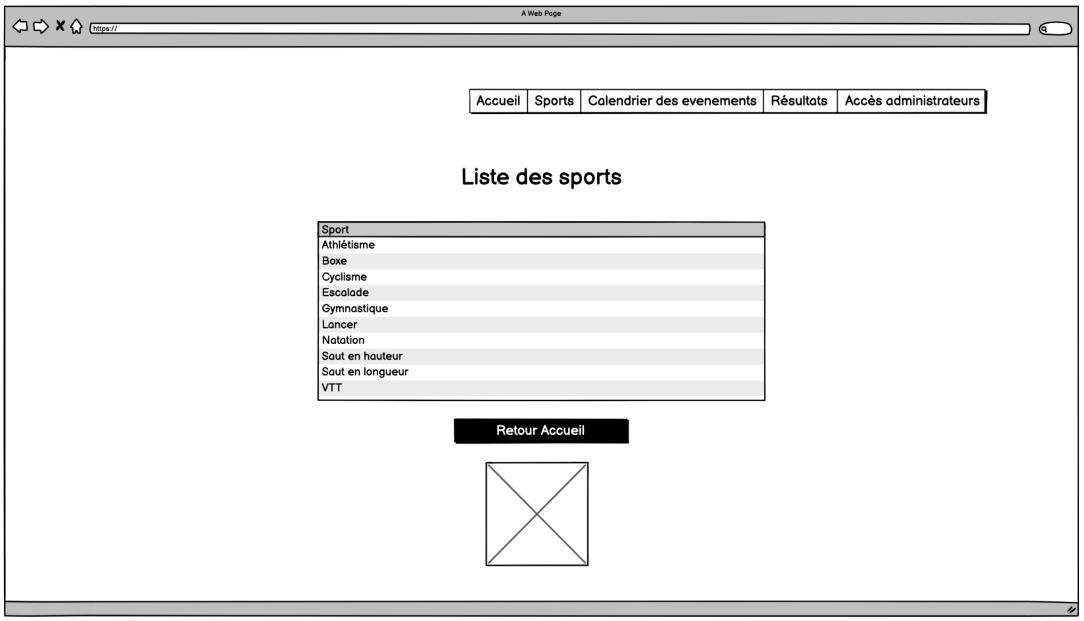
La conception du projet repose sur le développement d'un front-end dynamique et intuitif, élément essentiel dans l'expérience utilisateur globale.

5.1.1. Wireframes

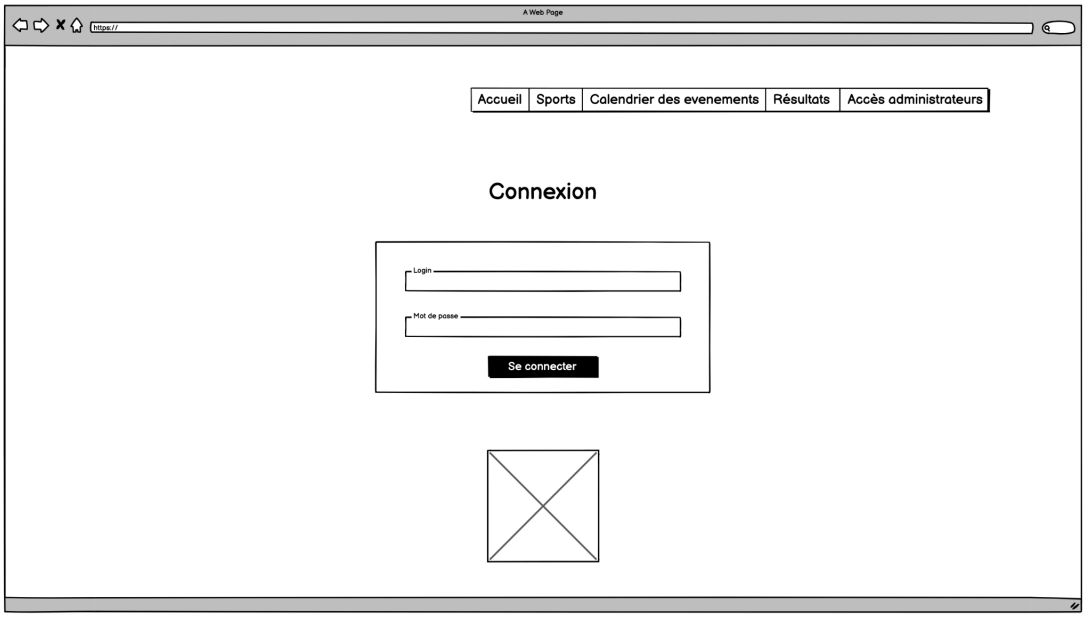
Lors de la conception, nous avons établi différents Wireframes qui sont mis à la disposition du commanditaire. On peut notamment retrouver les wireframe suivants:



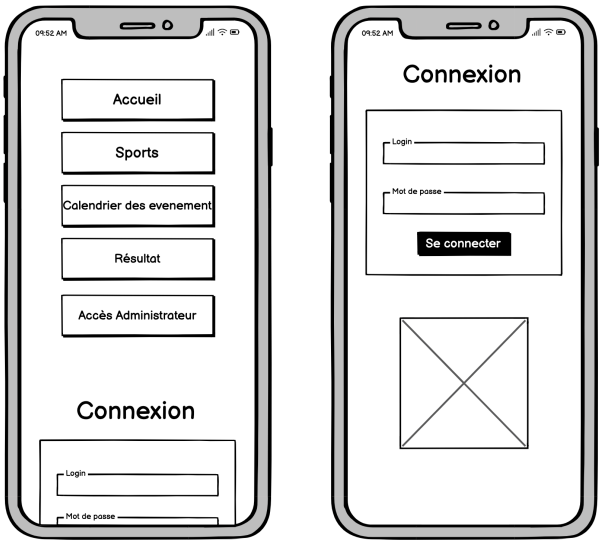
Wireframe de la page index



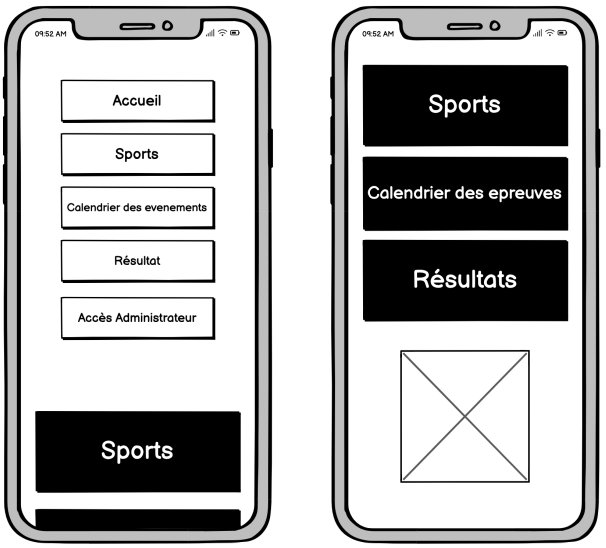
Wireframe de la page des sports



Wireframe de la page connexion



Wireframe responsive de la page connexion

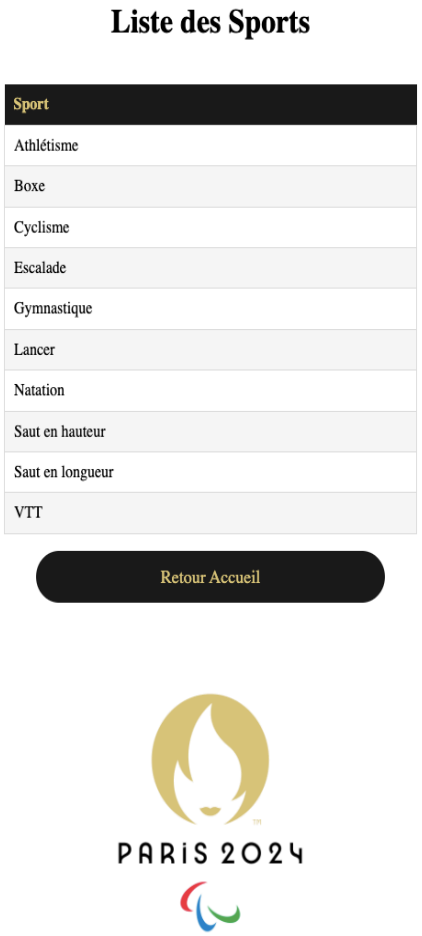


Wireframe responsive de la page index

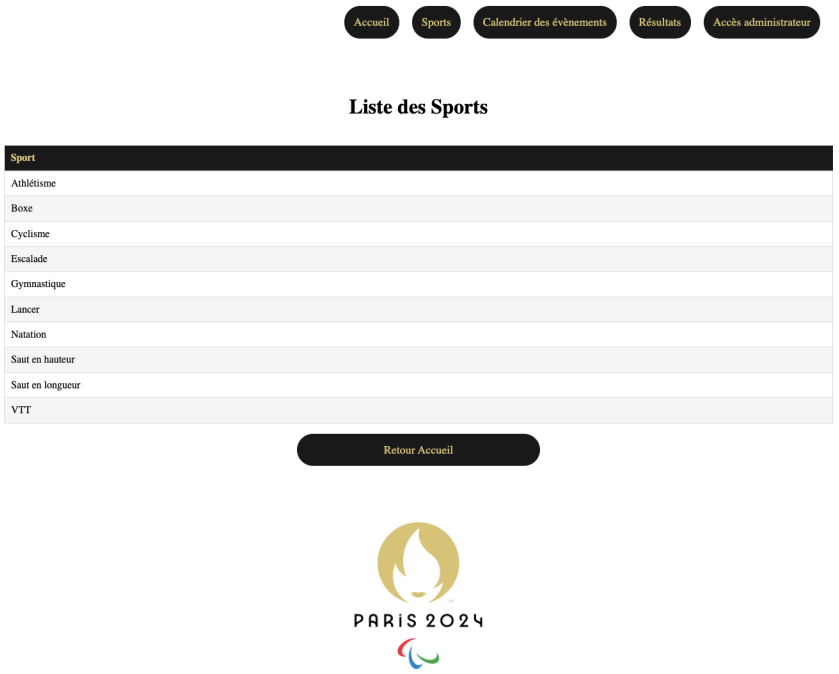
Wireframe responsive de la page sports

5.1.2. Maquettes

Dans le cadre du processus de maquettage, nous avons élaboré divers prototypes visuels qui sont mis à la disposition du client. Cela offre une représentation structurée des interfaces attendues. On retrouve notamment:

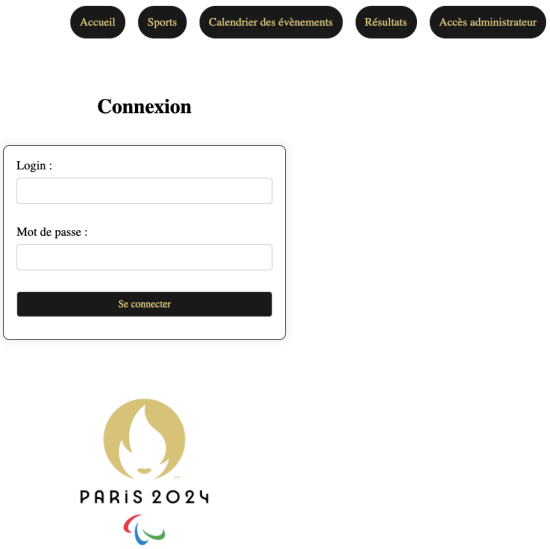


Maquette responsive de la page sports *Maquette responsive de la page index* *Maquette responsive de la page connexion*



Maquette de la page sports

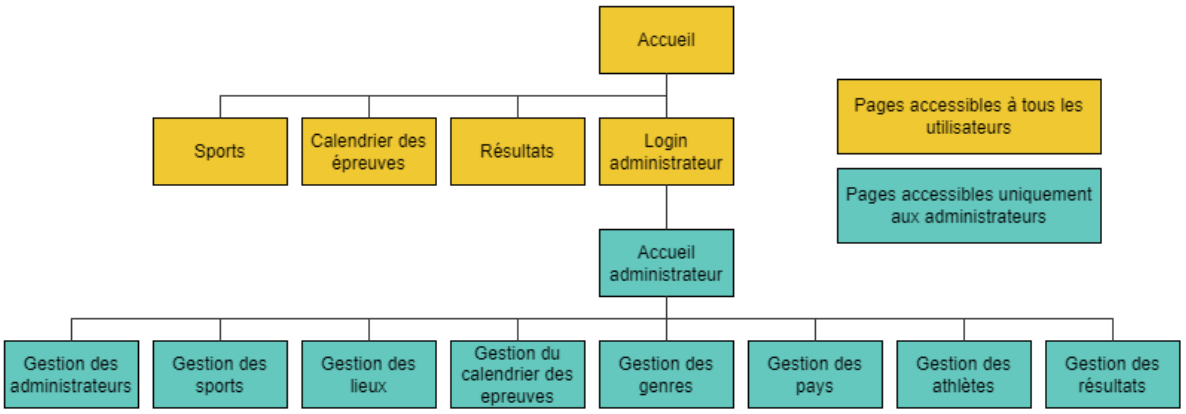
Maquette de la page index



Maquette de la page connexion

5.1.3. Arborescences

L'arborescence d'un site web représente la structure organisationnelle de ses pages et contenus, déterminant la manière dont l'information est hiérarchisée et présentée aux utilisateurs. Pour ce projet la réalisation de l'arborescence correspondra à ceci:

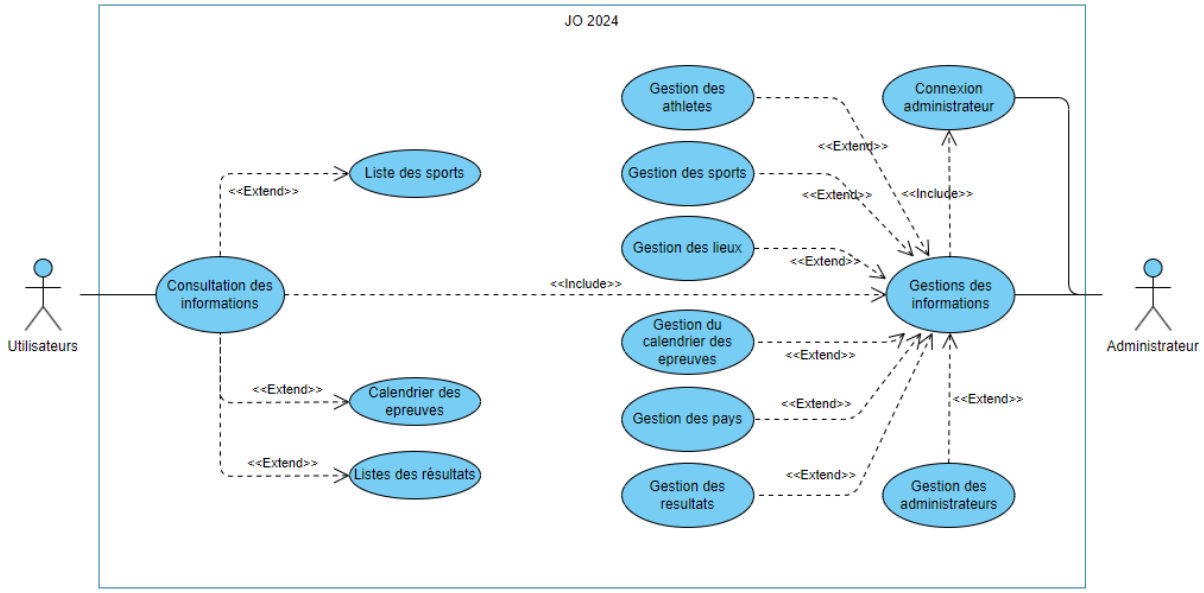


5.2. Le back-end

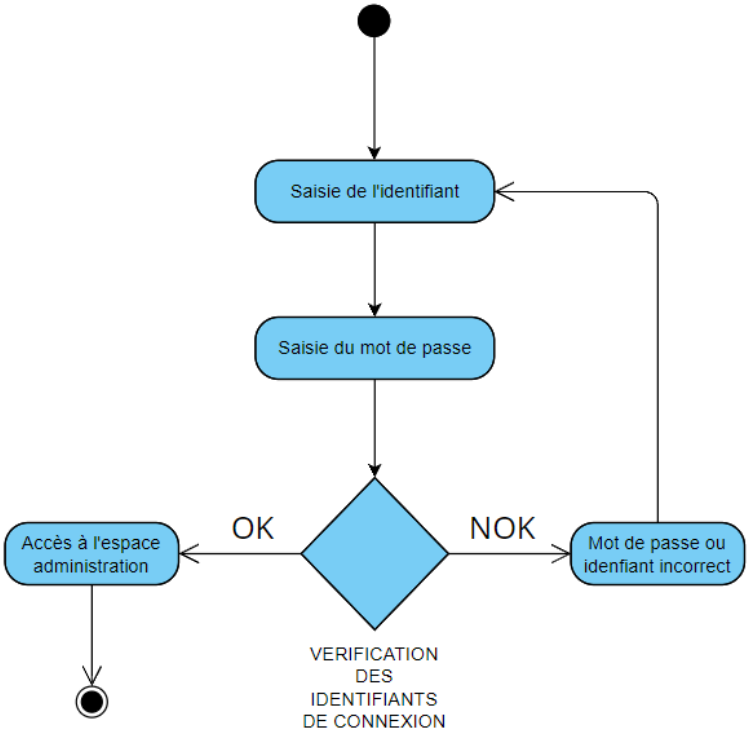
Le Backend, se charge de la communication avec la base de données, l'exécution des opérations côté serveur, et la génération des réponses nécessaires pour les requêtes provenant du front-end.

5.2.1. Diagramme de cas d'utilisation

Pour notre projet, nous avons dû élaborer un diagramme de cas d'utilisation ce qui va nous permettre d'avoir une représentation du comportement fonctionnel de notre projet.



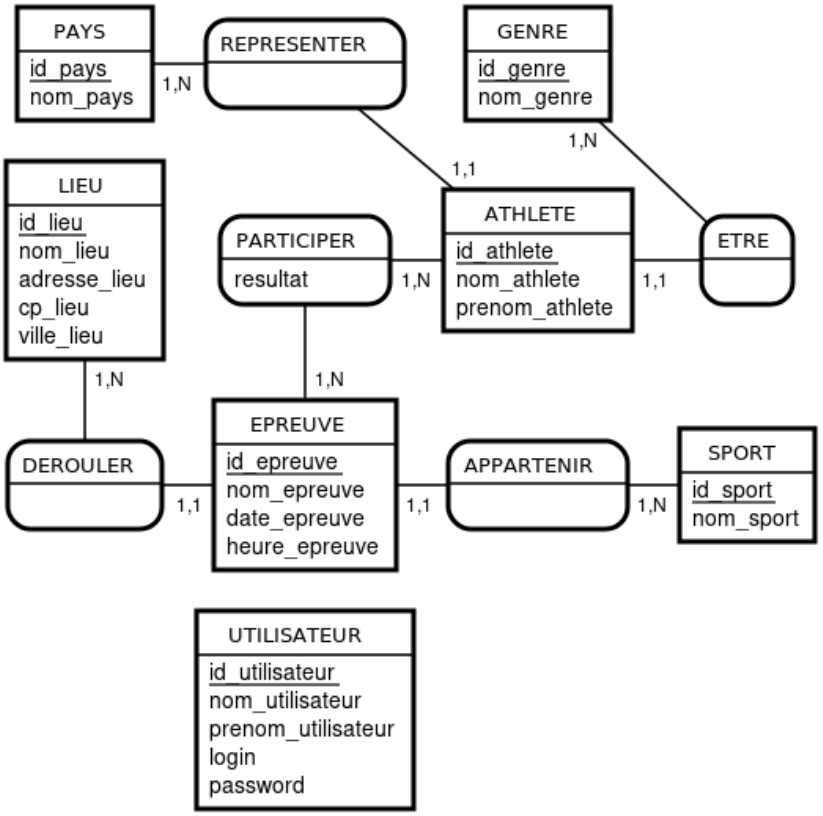
5.2.2. Diagramme d'activités



Grâce à l'élaboration du diagramme d'activité, on peut mettre en évidence les interactions entre les différentes séquences d'actions et les conditions de déclenchement. Offrant ainsi une représentation visuelle compréhensible.

5.2.3. Modèles Conceptuel de Données (MCD)

Nous avons dû créer un modèle conceptuel de données pour représenter de manière abstraite les entités, les relations et les contraintes du système d'information lié à l'événement. Offrant ainsi une vue globale et simplifiée de la structure des informations.



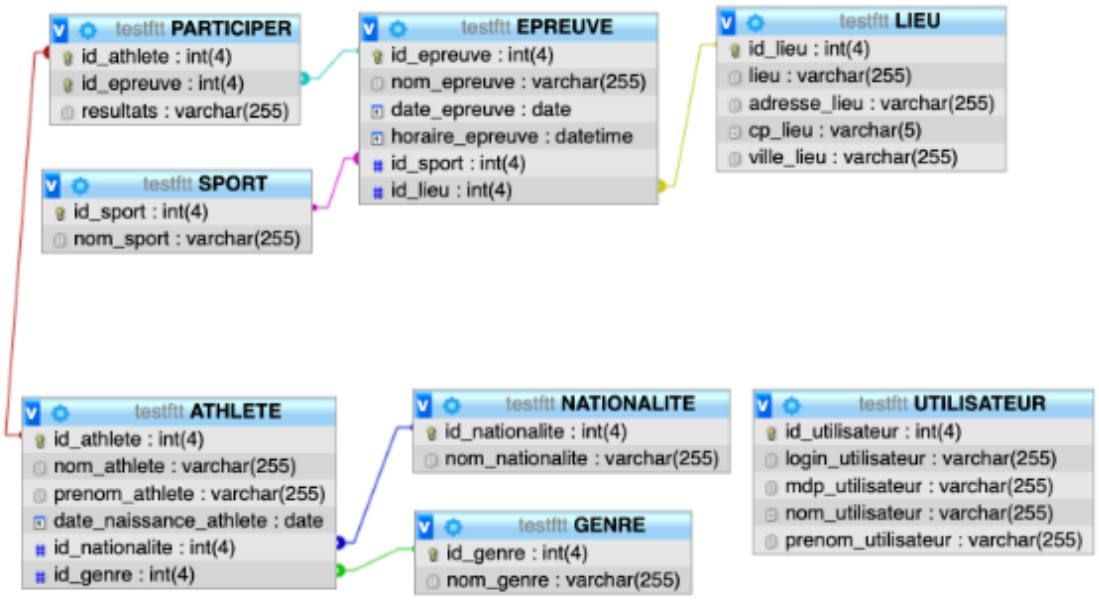
5.2.4. Modèles Logique de Données (MLD)

Ensuite, dans le cadre du projet, nous avons élaboré un modèle logique de données, pour définir plus précisément la structure des données, en utilisant des concepts tels que les tables, les clés primaires et étrangères, et les relations entre les entités.

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

5.2.5. Modèle Physique de Données (MPD)

Enfin, pour concrétiser notre système d'information pour les JO 2024, nous avons développé un modèle physique de données qui spécifie les détails techniques de mise en œuvre du MLD dans un système de gestion de base de données spécifique. Le MPD a inclus des éléments tels que les types de données, les index, les contraintes d'intégrité, et d'autres aspects techniques nécessaires à la création effective de la base de données pour les Jeux Olympiques.



6. Technologies utilisées

Dans le cadre des technologies utilisées pour notre projet, nous avons adopté une approche soigneusement planifiée, couvrant divers aspects du développement web.

6.1. Langages de développement Web

Le choix des langages de développement est crucial, ainsi nous veillerons à ce que le projet intègre HTML5, CSS3, JavaScript, PHP, et SQL. Ces langages sont utilisés par de nombreux sites et permettent de créer des interfaces dynamiques, réactives et fonctionnelles. HTML5 assure la structure de la page, CSS3 gère la présentation et le style, JavaScript permet des interactions côté client, tandis que PHP et SQL sont déployés pour la manipulation des données côté serveur.

6.2. Base de données

Pour la gestion efficace des données, nous avons opté pour une base de données. Mamp est l'outil de choix pour cette tâche, offrant une configuration simple et efficiente du serveur local. Cette solution garantit la persistance des données et permet des opérations de lecture et d'écriture rapides et fiables.

7. Sécurité

7.1. Login et protection des pages administrateurs

Pour créer une page de login administrateur sécurisé, il faut que le code PHP démarre une session pour stocker les variables de session, puis inclure le fichier de connexion à la base de données. Ensuite, il faut vérifier si la requête est une méthode POST, cela récupère les identifiants soumis par l'utilisateur, et prépare une requête SQL pour récupérer les informations de l'utilisateur avec le login spécifié.

En utilisant PDO, cela lie la variable :login à la valeur du login pour éviter les injections SQL, puis il faut exécuter la requête préparée et vérifier si le mot de passe correspond à celui stocké dans la base de données en utilisant la fonction password_verify.

Si les identifiants sont corrects, les informations de l'utilisateur sont stockées dans la session, et l'utilisateur est redirigé vers la page d'administration. Sinon, un message d'erreur est stocké dans la session et l'utilisateur est redirigé vers la page de connexion. Enfin, pour que le code soit optimisé on fait en sorte que le code libère les ressources associées à la requête préparée et ferme la connexion à la base de données.

De plus pour sécuriser les pages administrateur il faut mettre en place une vérification et voir si l'utilisateur est connecté avant de lui permettre d'accéder à une certaine partie du site. Cela empêche un utilisateur non authentifié d'accéder à des pages ou des fonctionnalités qui nécessitent une connexion.

Exemple de code résumant ce texte:

```php
<?php
session_start(); // Démarre la session PHP pour stocker des variables de session.

require_once("database.php"); // Inclut le fichier de connexion à la base de données.

if ($_SERVER["REQUEST_METHOD"] == "POST") { // Vérifie si la requête est une méthode POST (formulaire soumis).
    $login = $_POST["login"]; // Récupère la valeur du champ "login" du formulaire.
    $password = $_POST["password"]; // Récupère la valeur du champ "password" du formulaire.

    // 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.
```

Code permettant la connexion sécurisé de l'espace administration

```php
<?php

// page_admin.php

session_start();

if (!isset($_SESSION['id_utilisateur'])) {
    // Rediriger vers la page de connexion si l'utilisateur n'est pas authentifie
    header('Location: login.php');
    exit();
}

// Le reste du code de la page d'administration...

?>
```

Code permettant de sécuriser l'accès aux pages administrateur

7.2. Cryptage des mots de passe avec Bcrypt

Bcrypt est un algorithme de hachage spécialement conçu pour le stockage sécurisé des mots de passe. Il fonctionne sous le principe d'une itération paramétrable, ce qui permet de le rendre résistant aux attaques par force brute et par dictionnaire.

L'utilisation de BCRYPT doit se faire depuis la fonction PHP "password_hash()", ainsi voici un exemple d'utilisation de l'algorithme BCRYPT .

```php
<?php

// ...

//Variable d'un de passe en clair qui doit etre hasher
$MotDePasse_a_crypter = "motDePasseSecret";
//Variable d'un de passe hasher provenant de la BDD
$MotDePasse_de_la_BDD = "motDePasse_Hasher_Provenant_De_La_BDD";

//Hasher la variable $ par BCRYPT dans la variable $hash
$hash = password_hash($MotDePasse_a_crypter, PASSWORD_BCRYPT);
//Verfier si le mot de passe hasher correspond à celui provenant de la bdd
if (password_verify($MotDePasse_a_crypter, $MotDePasse_de_la_BDD)){
// Si mot de passe correst FAIRE
}else{
//Sinon Mot de passe incorrect
}

?>
```

Utilisation et vérification du mot de passe

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

Les attaques XSS se produisent quand des cybercriminels injectent un script malveillant dans le contenu du site Web ciblé, qui est ensuite inclus dans le contenu dynamique reçu par le navigateur de la victime.
Ces scripts malveillants peuvent accéder aux cookies, aux jetons de session ou à d'autres informations sensibles conservées par le navigateur et utilisées sur le site.

Exemple pour contrer les attaques en échappant les données:

```php
<?php
// Données provenant d'une source non fiable
$nom = "<script>alert('XSS attack');</script>";

// Échapper les données avant de les afficher
$nomEchappe = htmlspecialchars($nom, ENT_QUOTES, 'UTF-8');

// Afficher les données échappées dans la page
echo "<p>Nom : " . $nomEchappe . "</p>";
?>
```

7.4. Protection contre les injections SQL

L'injection SQL est une technique d'injection de code utilisée pour attaquer les applications basées sur les données, dans laquelle des instructions SQL malveillantes sont insérées dans un champ de saisie pour exécution. Il est donc crucial de protéger le code de toutes injections SQL possible, plusieurs méthode doivent donc être appliqué:

L'utilisation de requêtes préparées:

En manipulant la base de données depuis le code, il est préférable d'utiliser PDO au lieu de Mysqli, en effet ce dernier est beaucoup plus vulnérable.

```php
<?php

// Creation d'une requete preparer pour recuperer tout les utilisateurs de la table USERS
$stmt = $pdo->prepare("SELECT * FROM USERS WHERE nom = :nom AND mot_de_passe = :mot_de_passe");
$stmt->bindParam(':nom', $nom);
$stmt->bindParam(':mot_de_passe', $mot_de_passe);
$stmt->execute();

?>
```

utilisation de requêtes préparées

L'échappement des données:

Si l'utilisation de requêtes préparées n'est pas possible, ou que l'utilisation de la bdd est en mysqli, on peut échapper les données en utilisant des fonctions spécifiques.

```php
<?php

// Echapement des données avec la fonction mysqli_real_escape_string
$nom = mysqli_real_escape_string($connexion, $_POST['nom']);

?>
```

utilisation des échappements des données pour une bdd connecté en mysqli

Filtrer les données de saisie:

Enfin, on peut valider les données d'entrée pour assurer qu'elles correspondent au format attendu. On peut également ajouter des filtres ou des expressions régulières pour garantir que les données respectent les critères définis.

```php
<?php

// filter les donnees avec la fonction filter_var et son parametre FILTER_VALIDATE_EMAIL
if (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    // Gerer l'erreur d'email invalide
}
?>
```

utilisation des filtrages des données