APPLICATION JO2028

Cahier des charges techniques

Fousseynou Sidibe
BTS 2 SIO

Cahier des charges techniques

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 Los Angeles 2028 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 2028.

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 8 Novembre 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

REPRENDRE RÉPONSES MISSION 1

3.1. Ressources matérielles

Pour la réalisation de ce projet, les ressources matérielles suivantes sont nécessaires :

- **Ordinateurs portables avec Wi-Fi** : Chaque membre de l'équipe aura besoin d'un ordinateur portable équipé d'une connexion Wi-Fi pour travailler en mobilité ou à domicile.
- **Postes de travail fixes en classe (PC avec écrans et connexion Ethernet)** : Des ordinateurs fixes, équipés d'une connexion Ethernet, seront disponibles dans les salles de classe pour garantir une connexion stable pendant les séances de travail en présentiel.

Ces postes de travail permettront d'accéder aux outils de développement et de collaboration nécessaires à la bonne exécution du projet.

3.2. Ressources logicielles

Les ressources logicielles nécessaires pour la réalisation du projet comprennent :

- **Environnement de développement intégré (IDE)** :
    - **Visual Studio Code (VSCode)** : Un éditeur de code léger et performant, utilisé pour écrire et gérer le code du projet (HTML, JavaScript, PHP, etc.).
- **Gestion de code et collaboration** :
    - **GitHub** : Utilisé pour le partage et la gestion du code source du projet en équipe, permettant ainsi de collaborer efficacement, de gérer les versions et de suivre les modifications via Git.
- **Serveur Web et gestion de base de données (SGBDR)** :
    - **Apache** (inclus dans MAMP) : Utilisé pour héberger le site web localement durant le développement.
    - **MySQL** (inclus dans MAMP) : Un système de gestion de base de données relationnel (SGBDR) utilisé pour stocker et interroger les données du projet.
    - **MAMP** : Un environnement de développement local qui inclut Apache, MySQL et PHP pour faciliter le développement web en local.
- **Outil de gestion de projet** :
    - **Trello** : Une application de gestion de projet en ligne qui permet d'organiser les tâches, assigner les responsabilités et suivre l'avancement des différentes étapes du projet.
- **Conception UML et arborescence** :
    - **Visual Paradigm Online** : Un outil en ligne pour créer des diagrammes UML (modélisation de cas d'utilisation, diagramme de classes, etc.) et concevoir l'arborescence du projet.
- **Maquettage** :
    - **Figma** : Un outil de conception d'interface utilisateur (UI) utilisé pour le prototypage et la création des maquettes du site web.
- **Développement web** :
    - **Frontend** :
        - **HTML5** : Langage de balisage utilisé pour structurer les pages web.
        - **JavaScript (Vanilla)** : Langage de programmation utilisé pour rendre les pages web interactives.
        - **CSS** : Langage de style utilisé pour la mise en forme des pages web.
    - **Backend** :
        - **PHP8** : Langage de programmation côté serveur utilisé pour gérer la logique métier et les interactions avec la base de données.
        - **SQL** : Langage d'interrogation utilisé pour interagir avec la base de données MySQL (requêtes de sélection, insertion, mise à jour et suppression de données).

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.

INSÉREZ UNE CAPTURE D'ÉCRAN DE VOTRE TRELLO

Nous travaillons également sur GitHub, plateforme de développement collaboratif.

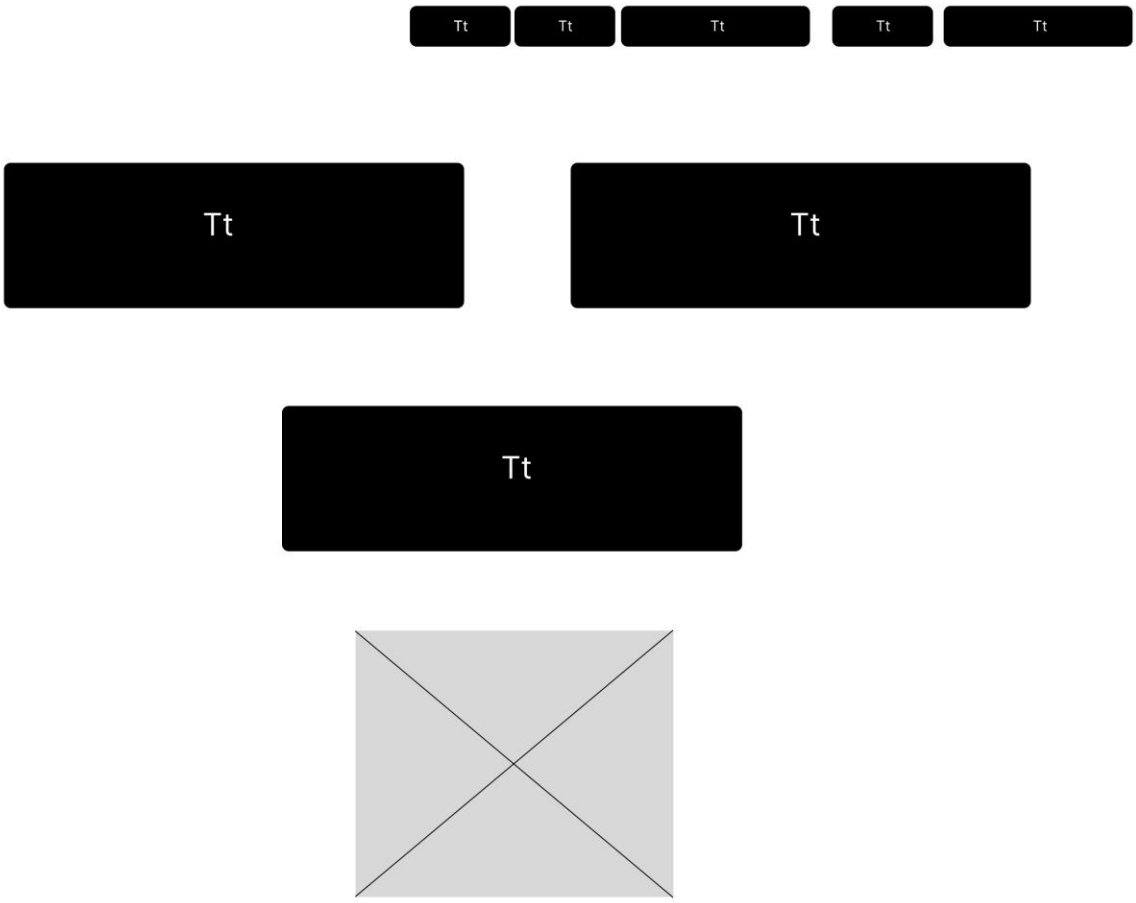
5. Conception du projet

5.1. Le front-end

REPRENDRE RÉPONSES MISSION 4

5.1.1. Wireframes

Version Ordinateur :

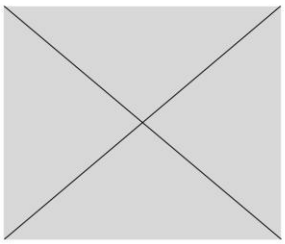


Tt Tt Tt Tt Tt

Login :

Mot de passe :

Se connecter

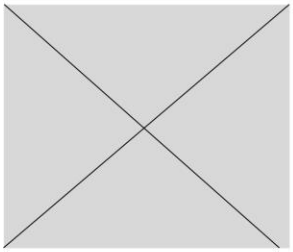


Tt Tt Tt Tt Tt Tt Tt Tt Tt

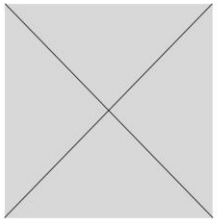
Nom du Sport :

Tt

Tt



Version responsive :



Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Tt

Nom du Sport :

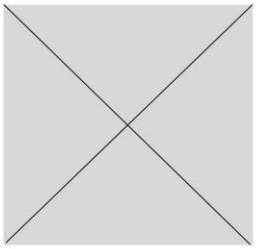
Tt

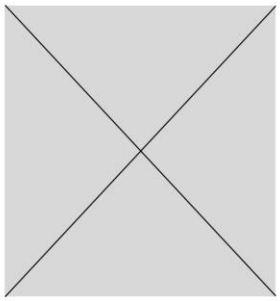
Login :

Mot de passee :

Se connecter

Tt





5.1.2. Maquettes

Accueil | Sports | Calendrier des épreuves | Résultats | Accès administrateur

Sports

Calendrier des épreuves

Résultats

Accueil | Sports | Calendrier des évènements | Résultats | Accès administrateur

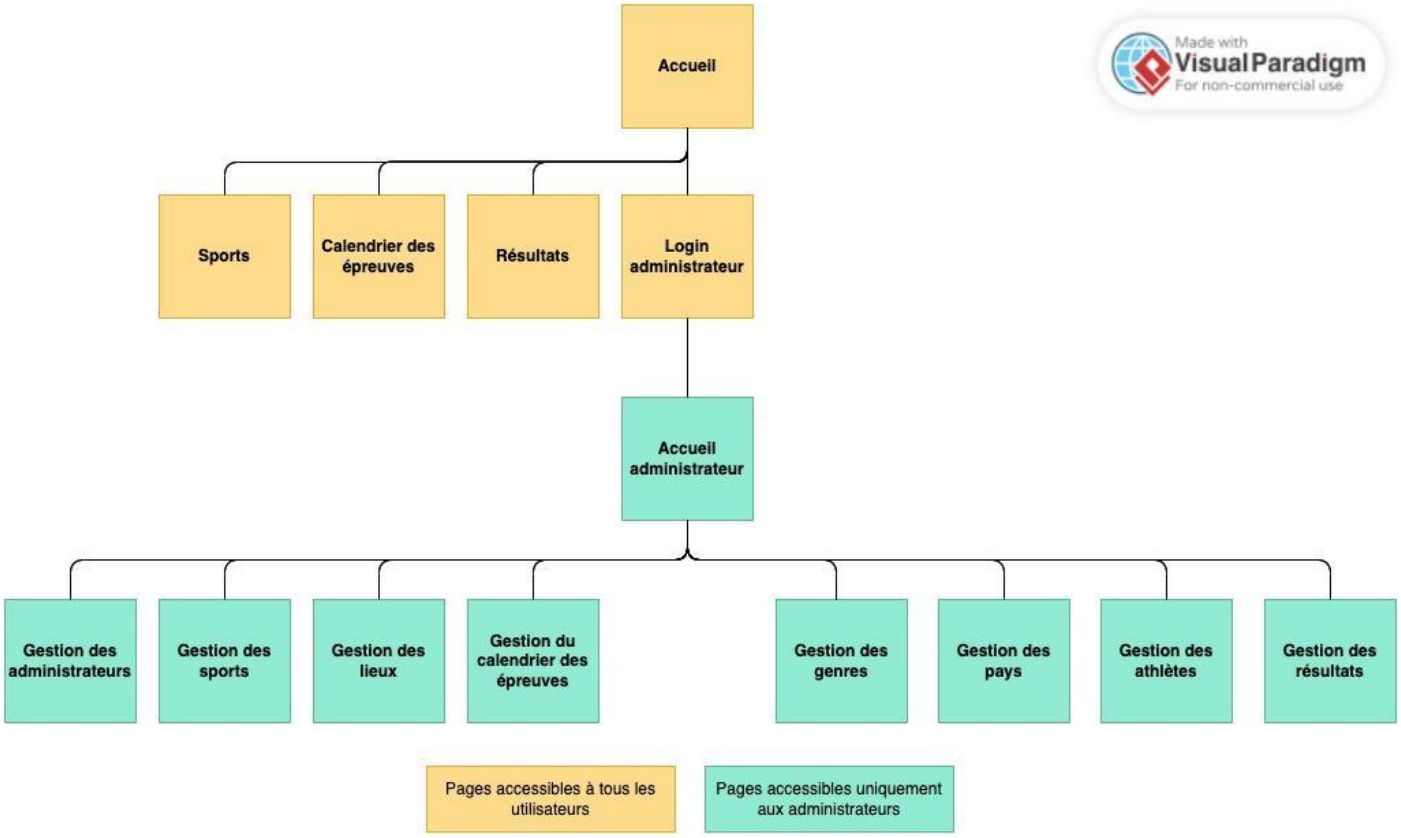
Connexion

Login :

Mot de passe :

Se connecter

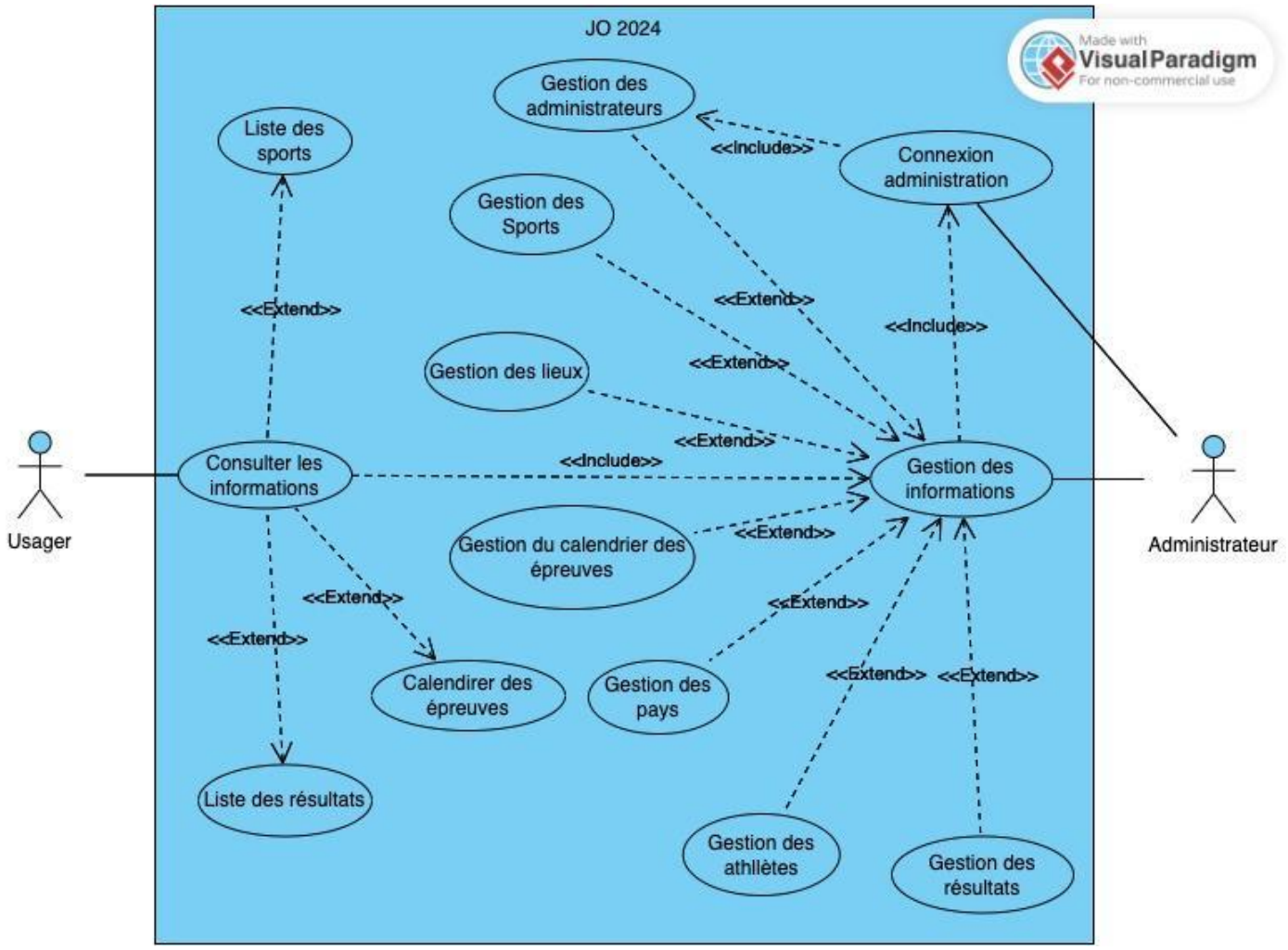
5.1.3. Arborescences



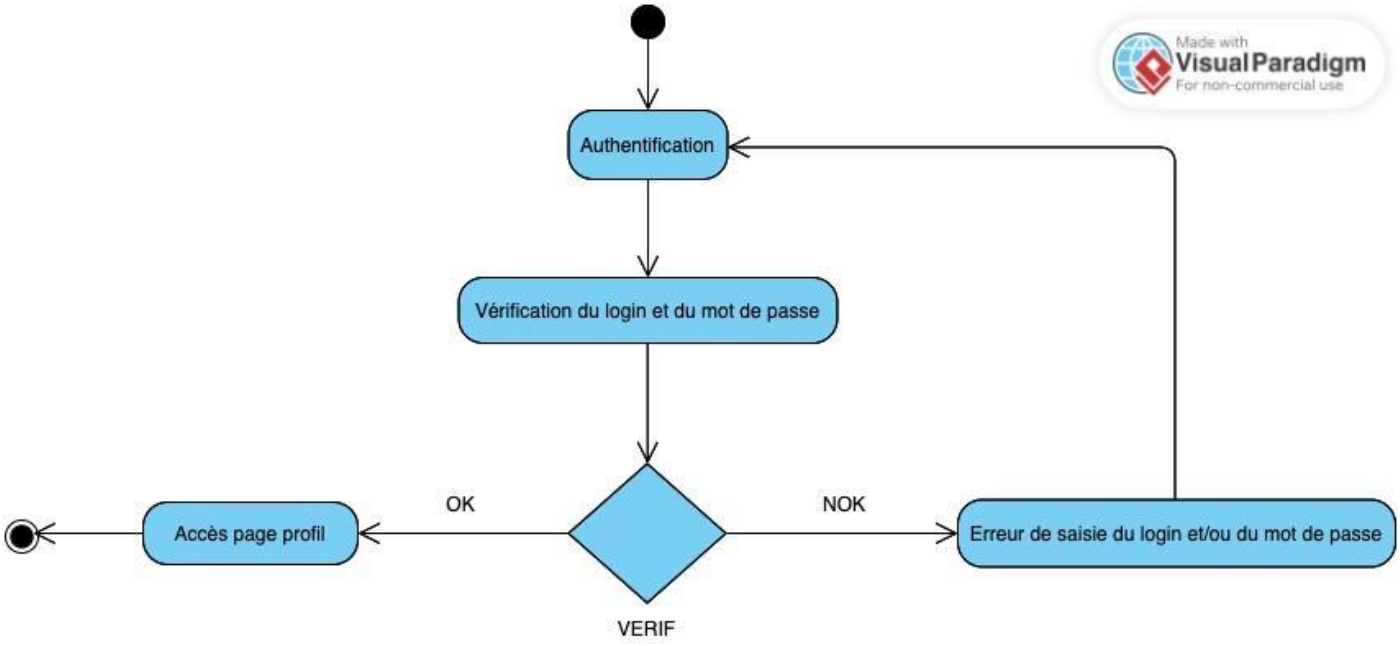
5.2. Le back-end

REPRENDRE RÉPONSES MISSIONS 2 ET 3

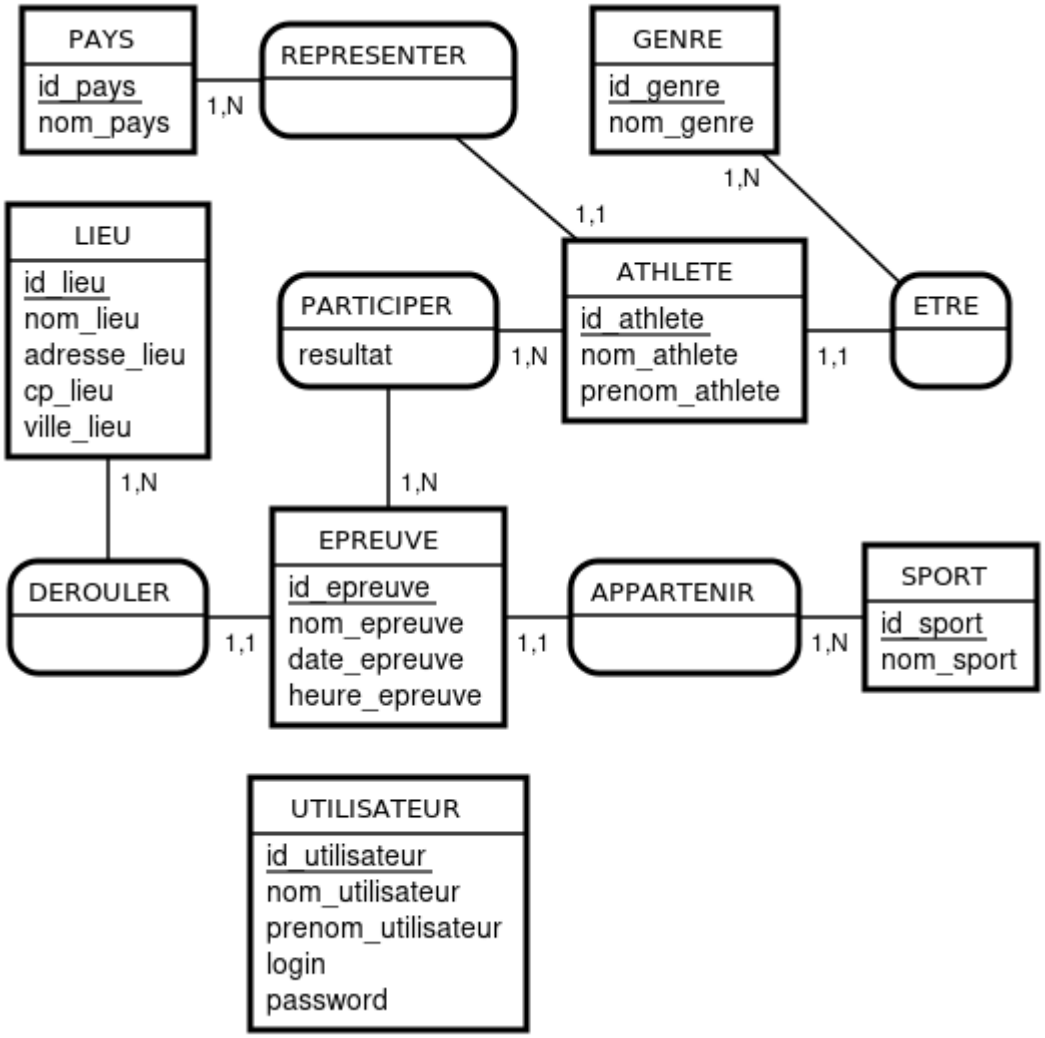
5.2.1. Diagramme de cas d'utilisation



5.2.2. Diagramme d'activités



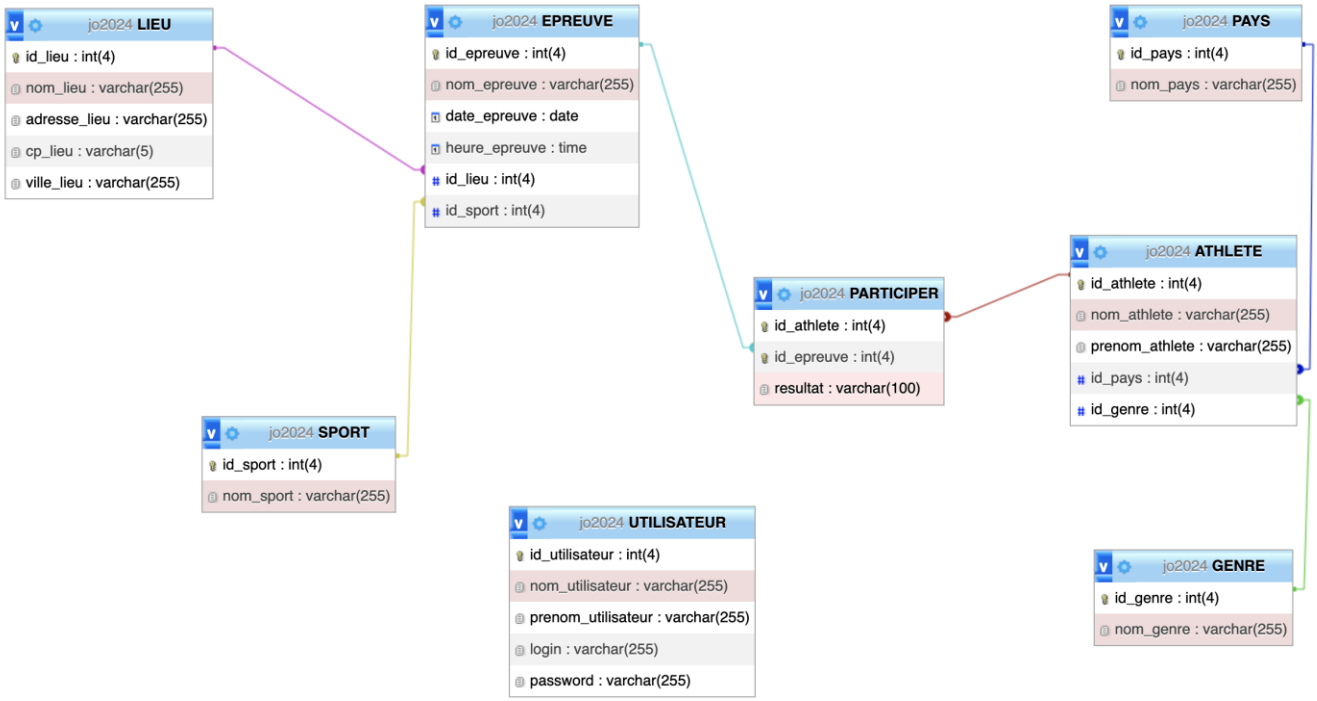
5.2.3. Modèles Conceptuel de Données (MCD)



5.2.4. Modèle Logique de Données (MLD)

- 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)

5.2.5. Modèle Physique de Données (MPD)



6. Technologies utilisées

REPRENDRE RÉPONSES MISSION 1

6.1. Langages de développement Web

Les technologies utilisées pour le développement web couvrent à la fois le **front-end** (côté client) et le **back-end** (côté serveur). Voici les langages utilisés dans le cadre de ce projet :

- **HTML5** : Langage de balisage utilisé pour structurer le contenu des pages web. Il permet de définir les éléments tels que les titres, paragraphes, formulaires, images et liens.
- **CSS (Cascading Style Sheets)** : Langage de style utilisé pour la mise en forme des éléments HTML. Il est utilisé pour définir l'apparence des pages web, notamment les couleurs, les polices, les marges, les positions et les animations.
- **JavaScript (Vanilla)** : Langage de programmation côté client utilisé pour ajouter de l'interactivité aux pages web. JavaScript est essentiel pour gérer les événements, les validations de formulaires et les animations sur le front-end.
- **PHP 8** : Langage de programmation côté serveur, utilisé pour la création de la logique métier et la gestion des interactions avec la base de données. PHP permet de générer dynamiquement du contenu HTML et de traiter les requêtes des utilisateurs.

6.2. Base de données

Pour la gestion des données du projet, un **SGBDR** (Système de Gestion de Base de Données Relationnel) est utilisé :

- **MySQL** : Base de données relationnelle incluse dans MAMP. Elle est utilisée pour stocker et gérer les données du projet de manière structurée. MySQL permet l'exécution de requêtes SQL (Structured Query Language) pour effectuer des opérations de sélection, insertion, mise à jour et suppression de données.

- **SQL** : Langage d'interrogation de base de données utilisé pour manipuler les données dans MySQL. Il permet de créer des tables, d'exécuter des requêtes complexes, de gérer les relations entre les données et d'assurer l'intégrité des informations stockées.

7. Sécurité

DÉFINISSEZ (SI POSSIBLE) ET EXPLIQUEZ BRIÈVEMENT VOTRE SOLUTION.
AJOUTEZ SI VOUS LE SOUHAITEZ DES COURS EXTRAITS DE CODE.

7.1. Login et protection des pages administrateurs

Pour sécuriser l'accès aux pages administratives d'une application PHP, il est crucial de mettre en place un système de login. Cela inclut la création d'un formulaire de connexion où les utilisateurs saisissent leur nom d'utilisateur et mot de passe. Une fois les informations soumises, elles sont vérifiées par rapport à celles stockées dans la base de données.

Les fonctions essentielles impliquées comprennent :

- **session_start()** : Pour gérer les sessions utilisateurs.
- **password_verify()** : Pour vérifier le mot de passe saisi par rapport à son hachage.
- **header('Location: page.php')** : Pour rediriger l'utilisateur vers les pages protégées après authentification.
- **session_destroy()** : Pour détruire la session lors de la déconnexion.

Ces étapes garantissent que seules les personnes autorisées peuvent accéder aux fonctionnalités sensibles de l'application.

7.2. Cryptage des mots de passe avec Bcrypt

Pour le cryptage des mots de passe, nous utilisons la fonction **password_hash()** en PHP. **password_hash()** utilise Bcrypt pour convertir un mot de passe en une chaîne de caractères qui est difficilement décryptable. Pour contrer les attaques par dictionnaire, la fonction génère un sel, ce qui rend le hachage unique.

Extrait de code :

```php
<?php
// Mot de passe à hacher
$motDePasse = 'votreMotDePasse';

// Hachage du mot de passe
$hachage = password_hash(password: $motDePasse, algo: PASSWORD_BCRYPT);

// Affichage du hachage
echo "Hachage du mot de passe : " . $hachage;

// Vérification du mot de passe
$motDePasseSaisi = 'motDePasseSaisi';

if (password_verify(password: $motDePasseSaisi, hash: $hachage)) {
    echo 'Mot de passe correct !';
} else {
    echo 'Mot de passe incorrect.';
}
?>

```

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

Pour se protéger contre les attaques XSS, nous utilisons la fonction **htmlspecialchars()** ce qui permet de remplacer les doubles guillemets en **&quot,** les simples guillemets en **&#039**, < (inférieur à) en **&lt** , > (supérieur à) en **&gt**, et le & (ET commercial) devient **&amp** . Ce qui nous permet de contrer les attaques XSS car la plupart des attaques XSS utilisent le JavaScript.

Extrait de code :

```php
<?php
// Vérification si le formulaire a été soumis
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Récupération et échappement des données
    $nom = htmlspecialchars(string: $_POST['nom'], flags: ENT_QUOTES, encoding: 'UTF-8');
    $message = htmlspecialchars(string: $_POST['message'], flags: ENT_QUOTES, encoding: 'UTF-8');

    // Affichage des données sécurisées
    echo "<h2>Données soumises :</h2>";
    echo "<p><strong>Nom :</strong> $nom</p>";
    echo "<p><strong>Message :</strong> $message</p>";
}
?>
```

7.4. Protection contre les injections SQL

Pour se protéger contre les injection SQL, nous utilisons l'extension PHP Data Objects (PDO) ce qui nous permet de séparer les instructions SQL et les donnes. À l'aide de $pdo = new PDO('…..') et $stmt = $pdo->prepare(requêtesql).

Extrait de code :

```php
<?php
// Connexion à la base de données
$pdo = new PDO(dsn: 'mysql:host=localhost;dbname=base_de_donnees', username: 'utilisateur', password: 'motdepasse');

// Requête préparée
$stmt = $pdo->prepare(query: 'SELECT * FROM utilisateurs WHERE email = :email');
$email = $_POST['email'];
$stmt->execute(params: ['email' => $email]);
$resultats = $stmt->fetchAll();

foreach ($resultats as $utilisateur) {
    echo "Nom : " . htmlspecialchars(string: $utilisateur['nom'], flags: ENT_QUOTES, encoding: 'UTF-8') . "<br>";
}
?>
```