Cahier des charges techniques

Fait par

KHABALI NAWAF

Date

05 - 12 - 2023

Version Blanc ici

-=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.

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

1.2. Réponses aux attentes

- Lien vers GitHub du Projet :

Github | https://github.com/K02Nawaf/jo-2024-khabali-nawaf

- Hebergement :

InfinityFree | http://jo-2024-khabali-nawaf.free.nf/

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

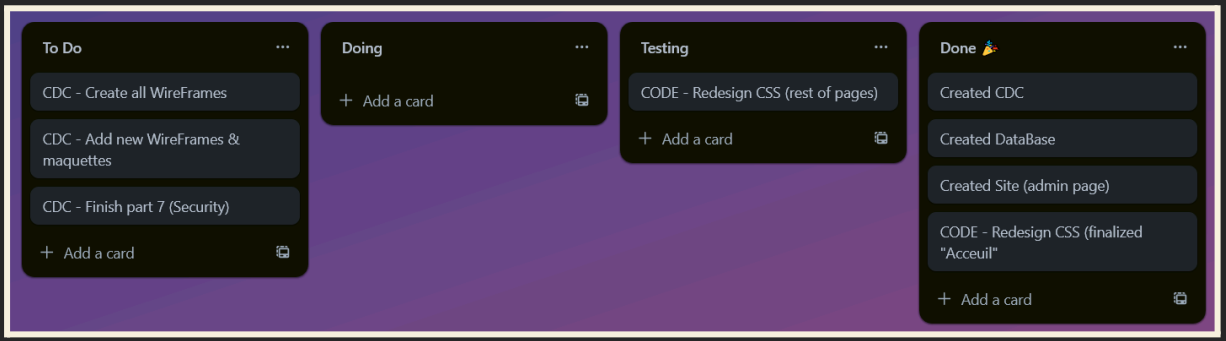
- Ordinateur
- Connexion internet

3.2. Ressources logicielles

- Visual studio code (l'IDE choisi)
- Github (enregistrement du projet dans le cloud)
- Mamp (Apache server, SQL database)
- Trello (organisation du projet)
- Mocodo (Creation du MCD et MLD)
- Monsta (File Transfer Protocol)

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.

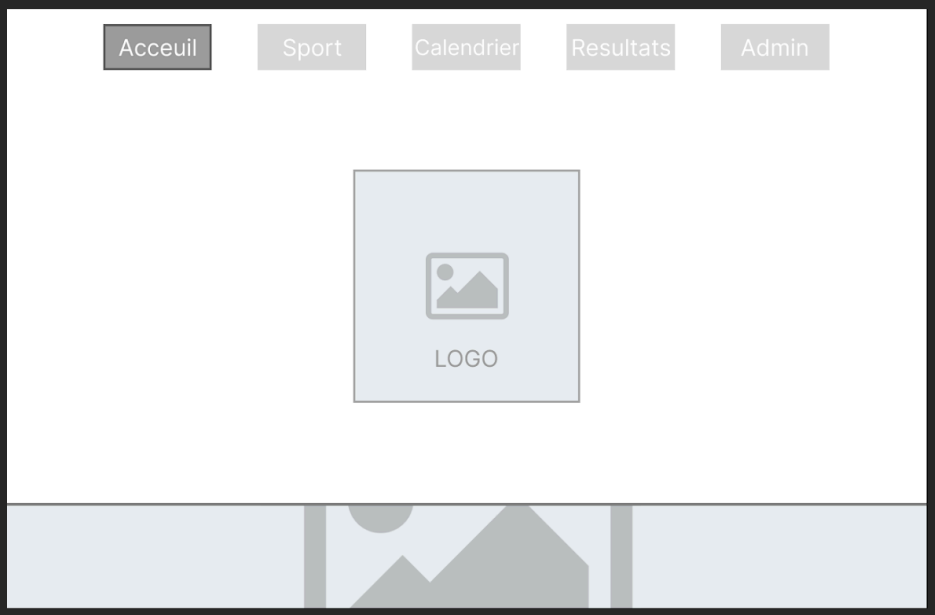
5 Conception du projet

5.1. Le front-end

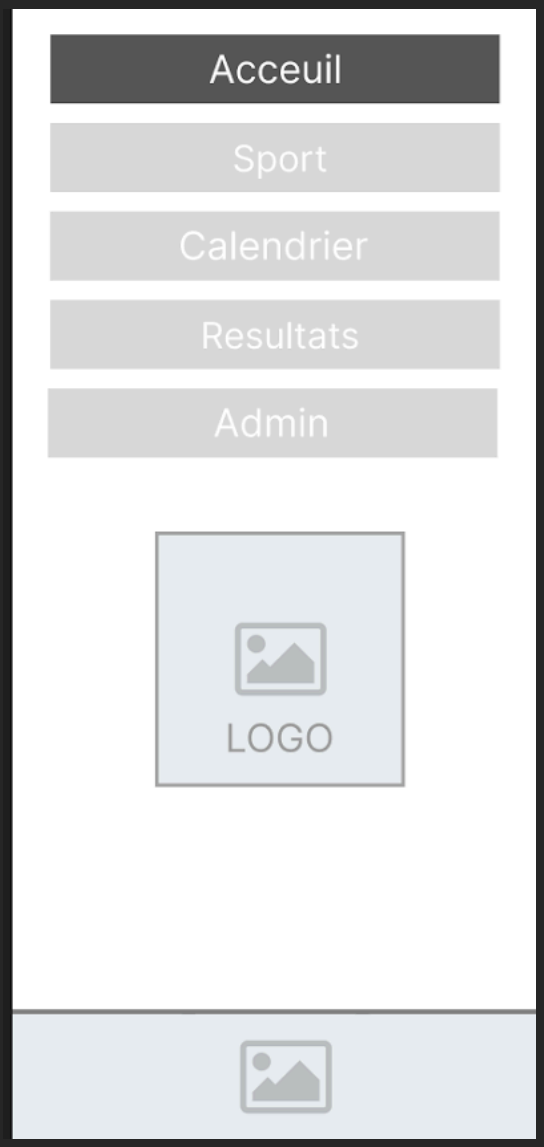
5.1.1. Wireframes

WireFrame.Index

PC



Responsive



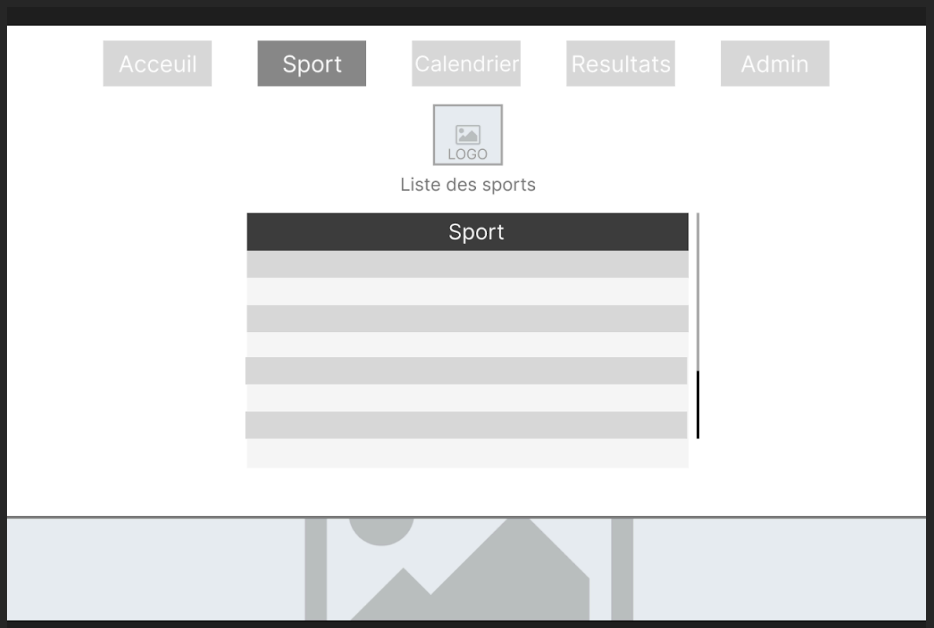
WireFrame.Ajout-sports

PC

Responsive

WireFrame.Liste-Sport

PC



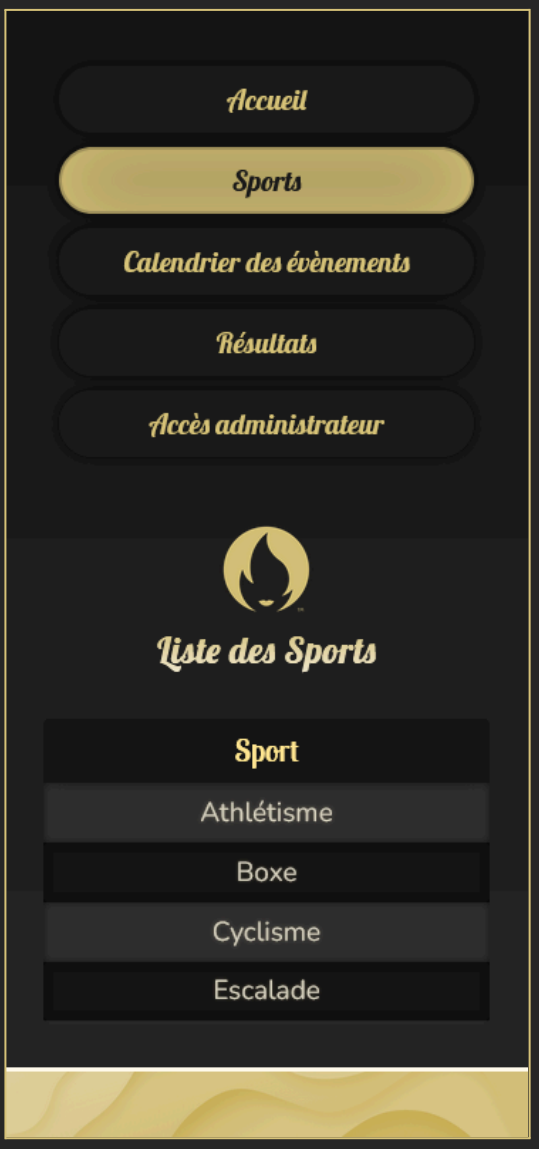
Responsive

5.1.2. Maquettes

Maquette.Index

PC

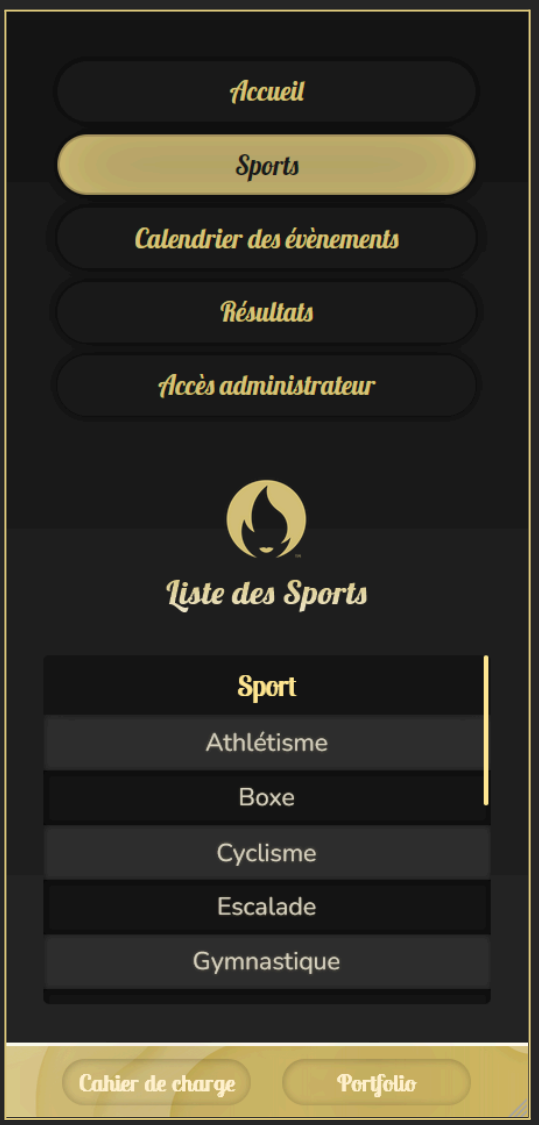
Responsive



Maquette.Liste-sport

PC

Responsive

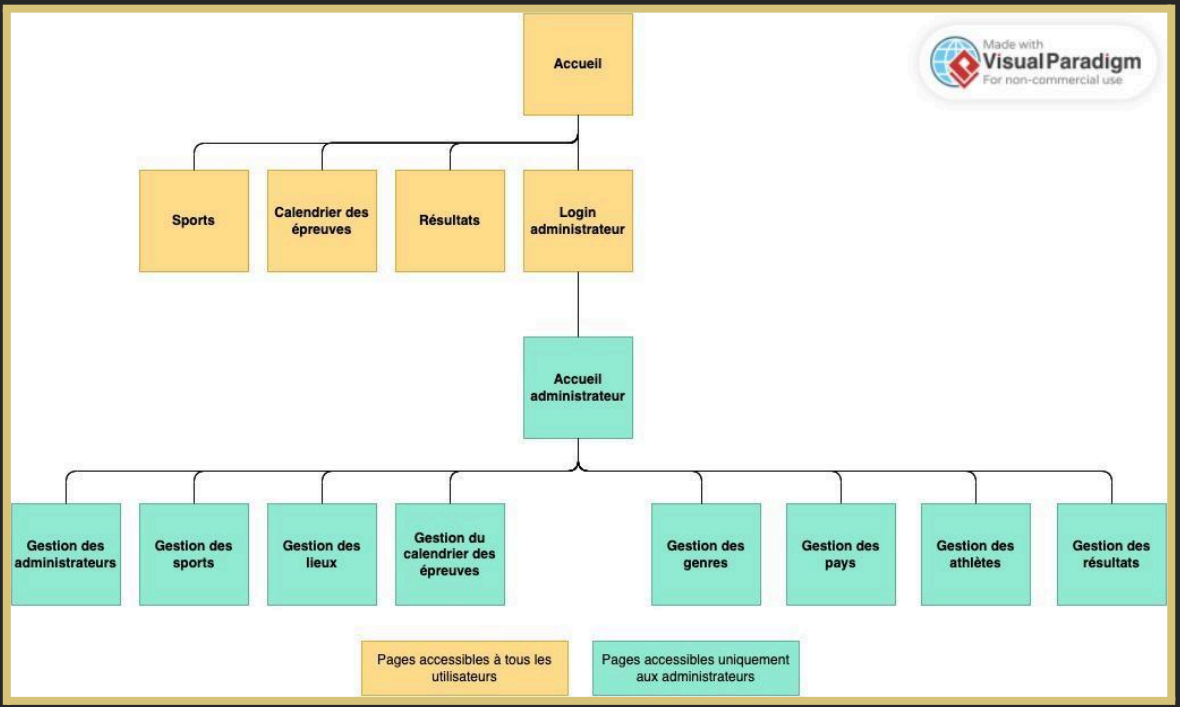


Maquette.Ajoute-sport

PC

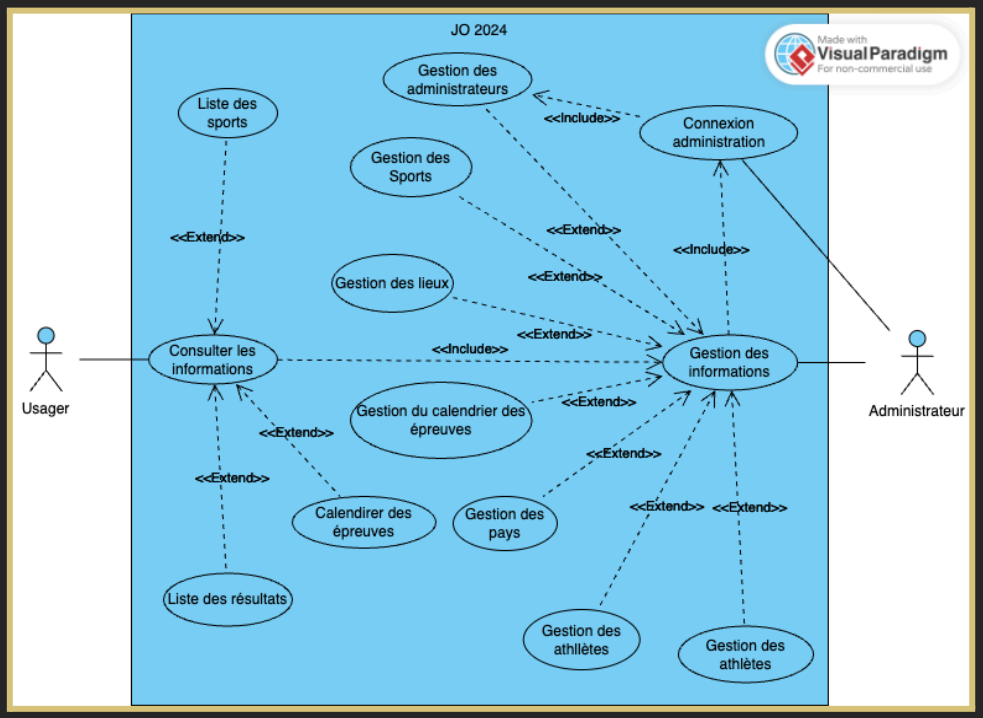
Responsive

5.1.3. Arborescences



5.2. Le back-end

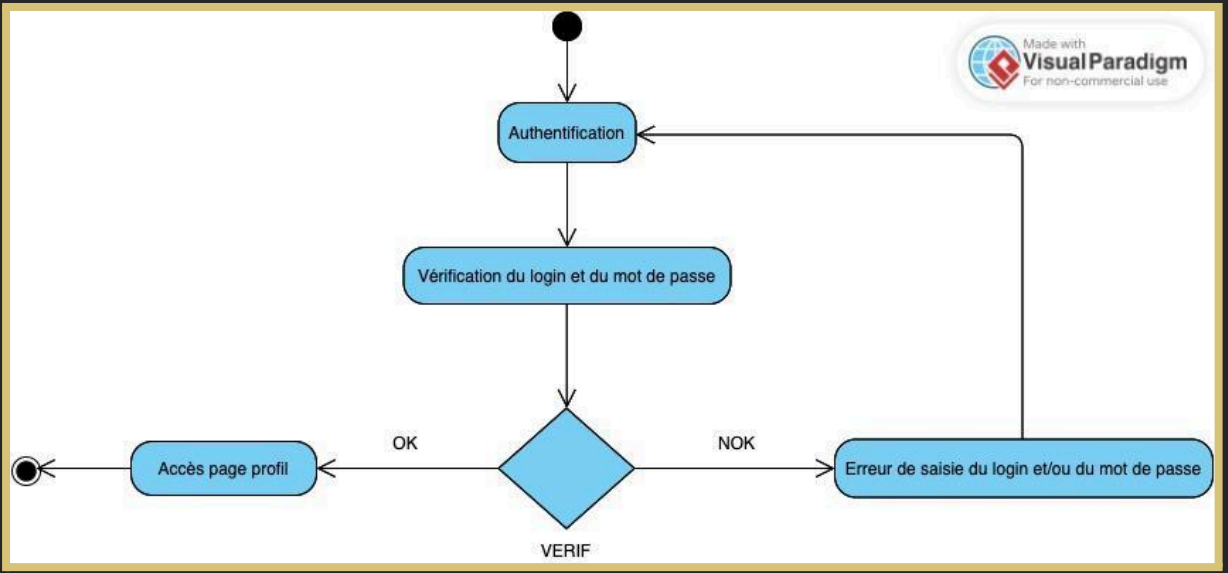
5.2.1. Diagramme de cas d'utilisation



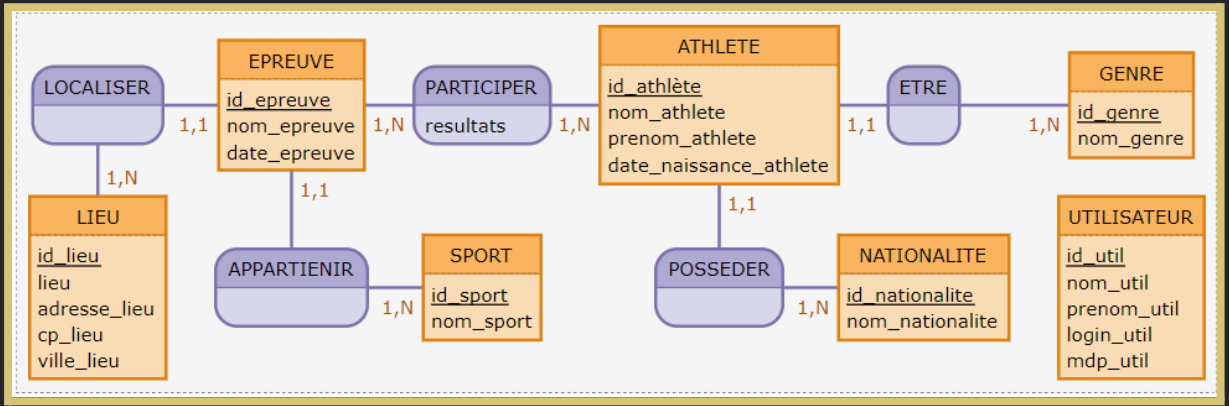
5.2.2. Dictionnaire de Données

Entities	Libelle	Type
Sport	id_sport	int(4)
	nom_sport	varchar(255)
Participer	id_athlete	int(4)
	id_epreuve	int(4)
	resultat	varchar(255)
Pays	id_pays	int(4)
	nom_pays	varchar(255)
Genre	id_genre	int(4)
	nom_genre	varchar(255)
Utilisateur	id_utilisateur	int(4)
	nom_utilisateur	varchar(255)
	prenom_utilisateur	varchar(255)
	login	varchar(255)
	password	varchar(255)
Athlete	id_athlete	int(4)
	nom_athlete	varchar(255)
	prenom_athlete	varchar(255)
	id_pays	int(4)
	id_genre	int(4)
Lieu	id_lieu	int(4)
	nom_lieu	varchar(255)
	adresse_lieu	varchar(255)
	cp_lieu	varchar(255)
	ville_lieu	varchar(255)

5.2.3. Diagramme d'activités



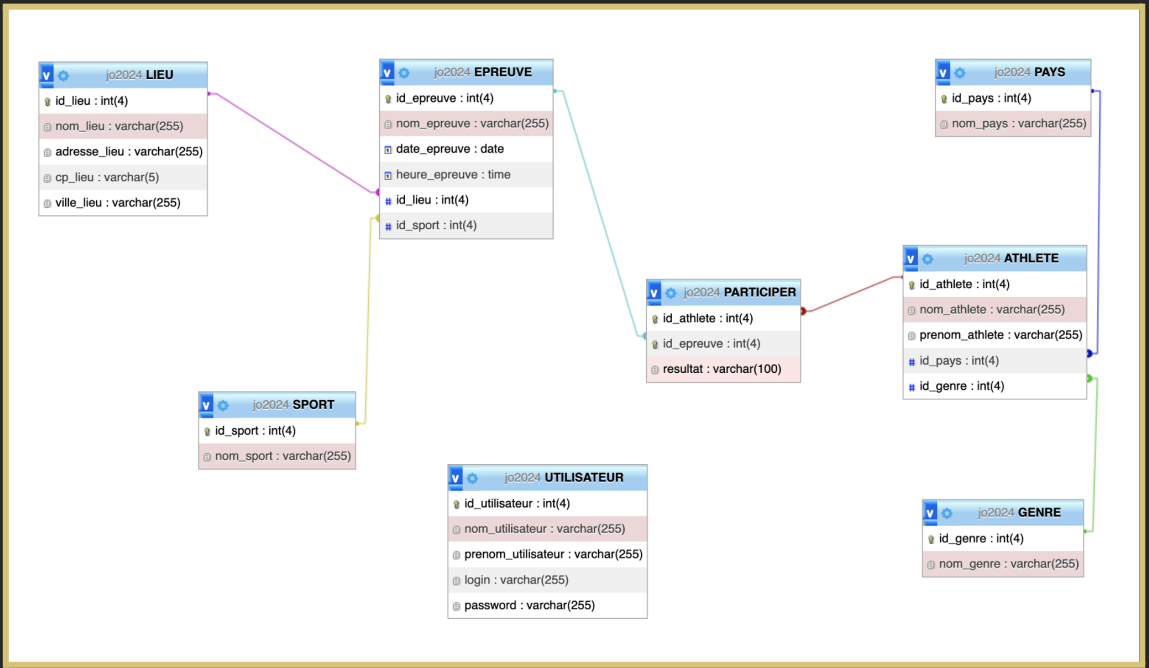
5.2.4. Modèles Conceptuel de Données (MCD)



5.2.5. Modèle Logique de Données (MLD)

- ATHLETE (id_athlète, nom_athlete, prenom_athlete, date_naissance_athlete, #id_genre, #id_nationalite)
- EPREUVE (id_epreuve, nom_epreuve, date_epreuve, #id_lieu, #id_sport)
- GENRE (id_genre, nom_genre)
- LIEU (id_lieu, lieu, adresse_lieu, cp_lieu, ville_lieu)
- NATIONALITE (id_nationalite, nom_nationalite)
- PARTICIPER (#id_athlète, #id_epreuve, resultats)
- SPORT (id_sport, nom_sport)
- UTILISATEUR (id_util, nom_util, prenom_util, login_util, mdp_util)

5.2.6. Modèle Physique de Données (MPD)



6 Technologies utilisées

6.1. Langages de développement Web

- HTML
- CSS
- JS
- php
- SQL

6.2. Base de données

- MySQL

MySQL est utilisé pour stocker des données dans des tables qui correspondent à des objets. Chaque table possède un schéma définissant les colonnes de chaque ligne de la table.

7 Sécurité

Assurer la sécurité d'un site web est l'un des piliers les plus importants lorsqu'il s'agit de créer des sites web dynamiques. Surtout lorsqu'ils contiennent des informations personnelles sur les utilisateurs, des données sensibles ou des informations de connexion.

C'est pourquoi nous avons mis en place les mesures de sécurité suivantes.

7.1. Login

Le site web contient une page de connexion avec un forum de connexion spécialement conçu pour les administrateurs du site web afin de pouvoir gérer les données affichées dans les pages principales.

7.2. Cryptage des mots de passe

Pour la création d'un compte admin, cette action ne peut être effectuée que par le Super Admin, qui peut consulter tous les autres comptes administrateur et peut les créer, les modifier ou les supprimer.

Les comptes admin créés par le Super Admin sont enregistrés dans la base de données et les mots de passe sont hachés à l'aide de la méthode Bcrypt.

- Voici la ligne de code responsable du hachage :

```php
// Hachage du mot de passe
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

// Requête pour ajouter un utilisateur
$query = "INSERT INTO utilisateur (nom_utilisateur, prenom_utilisateur, login, password)
VALUES (:nomUser, :prenomUser, :login, :hashedPassword)";
```

- et voici à quoi ressemble un mot de passe haché :

nom_utilisateur	prenom_utilisateur	login	password
Admin	Super	admin	$2y$10$WFxymbZ/gV2XfGy1We2bB.NZ9owdEU5QKUFWAicOY7q...
User	John	john_doe	$2y$10$VSdvPWt4OQnuQdT2vrP1z.5PzBJ5FuJc/bJhIFL8TB2...

7.3. Protection des pages administrateurs

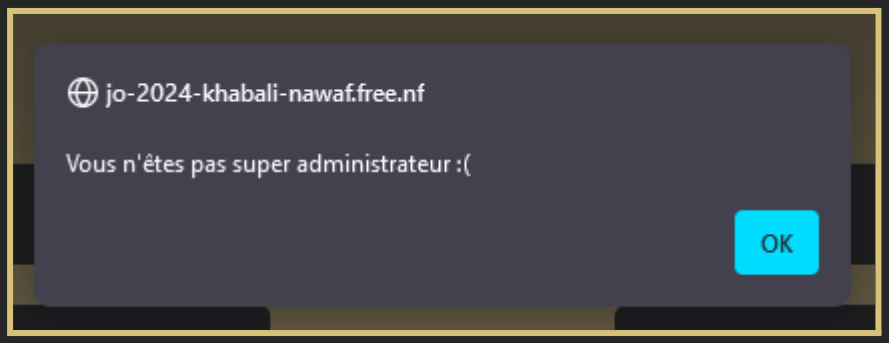
Les pages d'administration sont sécurisées de plusieurs manières, dont les suivantes :

-L'accès aux pages d'administration est limité au public et n'est accessible qu'aux administrateurs utilisant leurs identifiants de connexion, qui sont stockés en toute sécurité dans la base de données.

-Une fois qu'un administrateur s'est connecté, une session est ouverte, lui permettant de gérer les données. Lorsque l'administrateur se déconnecte, toutes les sessions actives sont rapidement détruites, ce qui permet d'empêcher tout accès non autorisé.

-La page "Gestion Utilisateurs" est conçue de manière à ce que seul le Super Admin ait accès à toutes les données des utilisateurs "administrateurs", ce qui inclut la modification, l'ajout et la suppression.

Pour les utilisateurs qui ne sont pas des super-administrateurs, l'accès à cette page entraînera l'affichage d'une fenêtre d'erreur et la redirection vers la page principale.



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

La page de connexion est protégée contre les attaques XSS par la validation des entrées. L'entrée de l'utilisateur est assainie à l'aide de htmlspecialchars dans le backend PHP, ce qui garantit que toute injection de script potentielle est neutralisée avant le rendu.

- Exemple de auth.php :

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") { // Vérifiez si la méthode de demande est POST (formulaire soumis).
    $loginRetrieve = $_POST["login"]; // Récupérez la valeur du champ "login" du formulaire.
    $login = htmlspecialchars($loginRetrieve); // Assainissez l'entrée du login
    $password = $_POST["password"]; // Récupérez la valeur du champ "password" du formulaire.
```

7.5. Protection contre les injections SQL

L'utilisation d'instructions préparées dans les requêtes SQL garantit que les entrées de l'utilisateur sont traitées comme des données et non comme du code exécutable. Une autre méthode, "bindParam", est une méthode PDO permettant de lier des paramètres à une instruction préparée. Elle aide à prévenir les injections SQL en garantissant que les valeurs fournies sont traitées comme des paramètres et non comme une partie de la requête SQL elle-même, ce qui rend difficile l'injection de code SQL malveillant par les attaquants.

- Exemple de auth.php :

```php
// Préparer 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); // Préparer la requête avec PDO.
$stmt->bindParam(":login", $login, PDO::PARAM_STR);
// Lier la variable :login à l'identifiant assaini, empêchant ainsi les injections SQL.
```