Cahier des charges

E5 - Jeux Olympiques

Fait par Alexandre Ossant

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 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 7 novembres 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

Pour assurer la mise en œuvre efficace du projet, les ressources matérielles indispensables comprennent :

- **Ordinateur portable**
- **Cable Ethernet**
- **Internet**
- **Écran d'ordinateur**
- **Clavier**
- **Souris**

3.2. Ressources logicielles

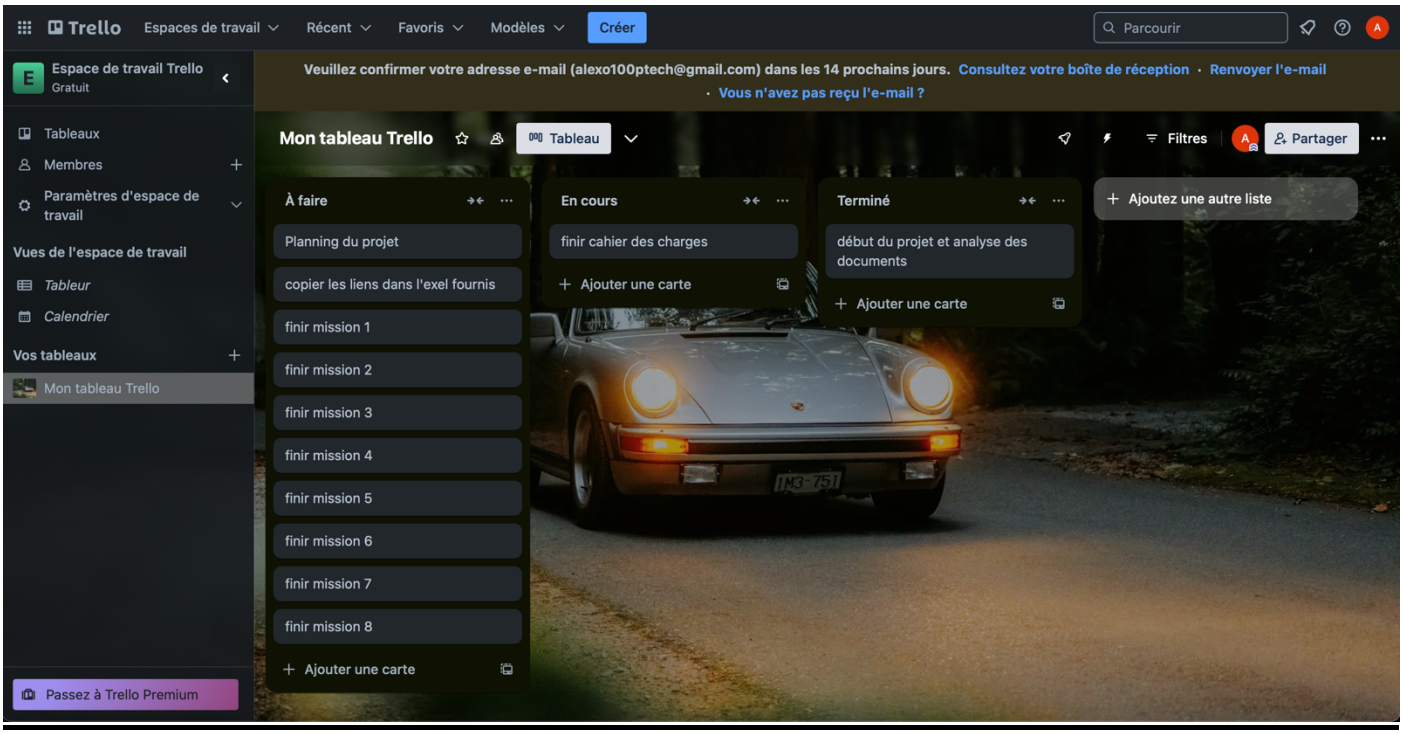
Les outils logiciels essentiels pour mener à bien le projet sont :

- **MAMP** : Environnement de développement local utilisé pour gérer les bases de données MySQL et configurer des serveurs web Apache.
- **GitHub** : Plateforme collaborative permettant d'héberger le code source, de suivre les versions et de gérer les modifications apportées au projet.

- **Visual Studio Code (IDE)** : Outil utilisé pour la programmation du projet.
- **Trello** : Outil de gestion de projet en ligne, facilitant l'organisation des tâches, leur suivi et la collaboration au sein de l'équipe.
- **Figma** : Utilisé pour la création des wireframes et maquettes, afin de visualiser et concevoir l'interface utilisateur.
- **Moccodo** : Permet de simplifier la création du modèle conceptuel de données, facilitant ainsi la structuration et la compréhension des informations.
- **Visual Paradigm** : Employé pour modéliser et concevoir les processus, tout en établissant une arborescence structurée du projet.

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.

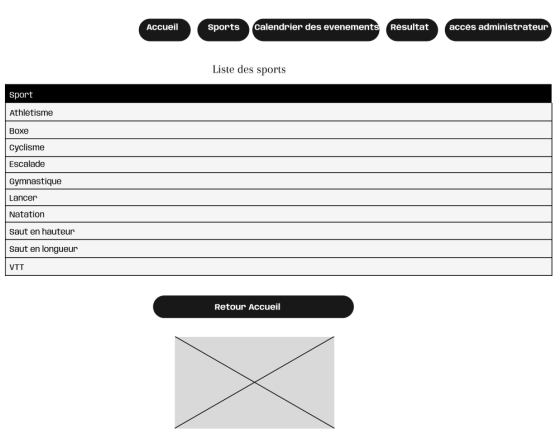


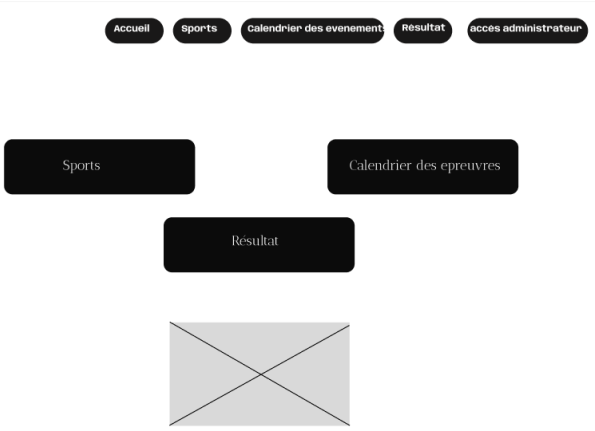
Nous travaillons également sur GitHub, plateforme de développement collaboratif.

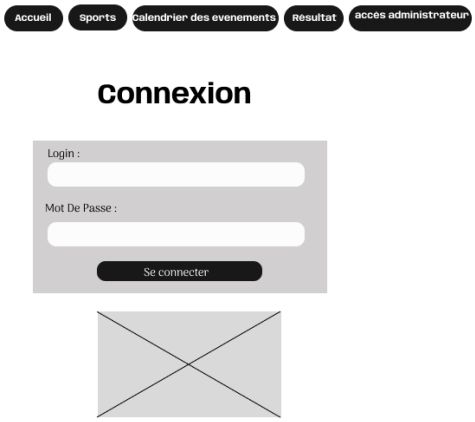
5. Conception du projet

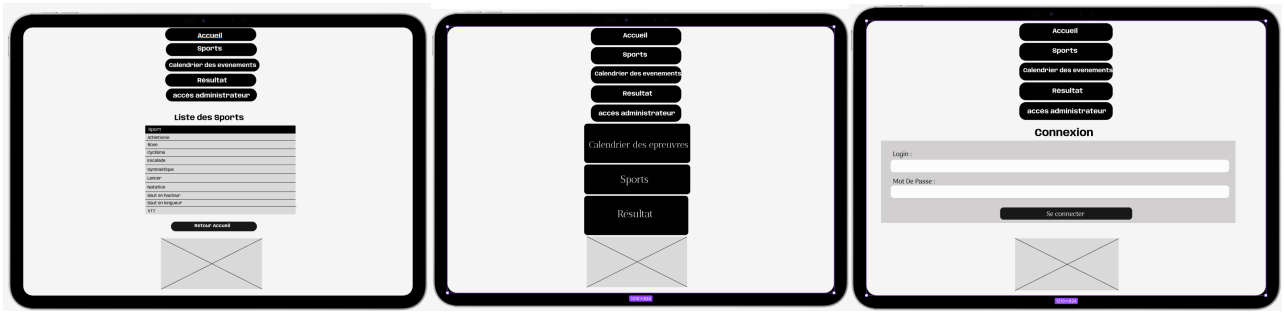
5.1. Le front-end

5.1.1. Wireframes

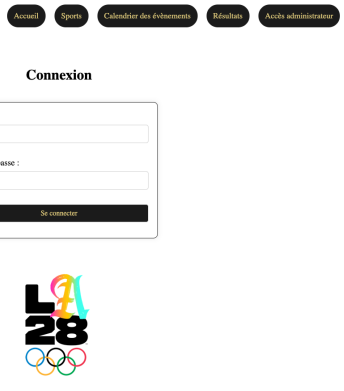


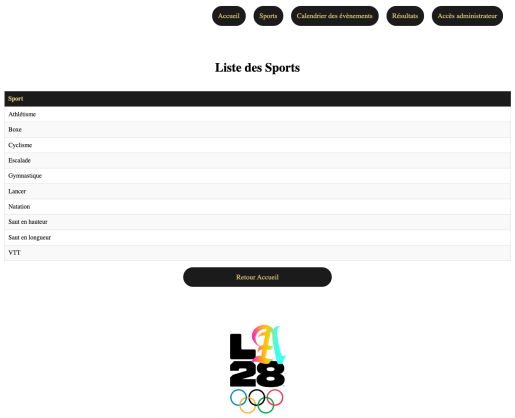


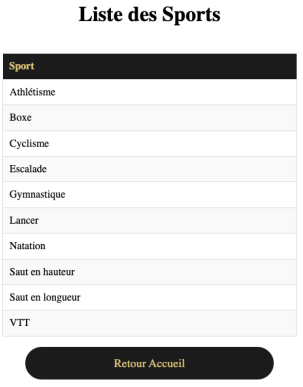




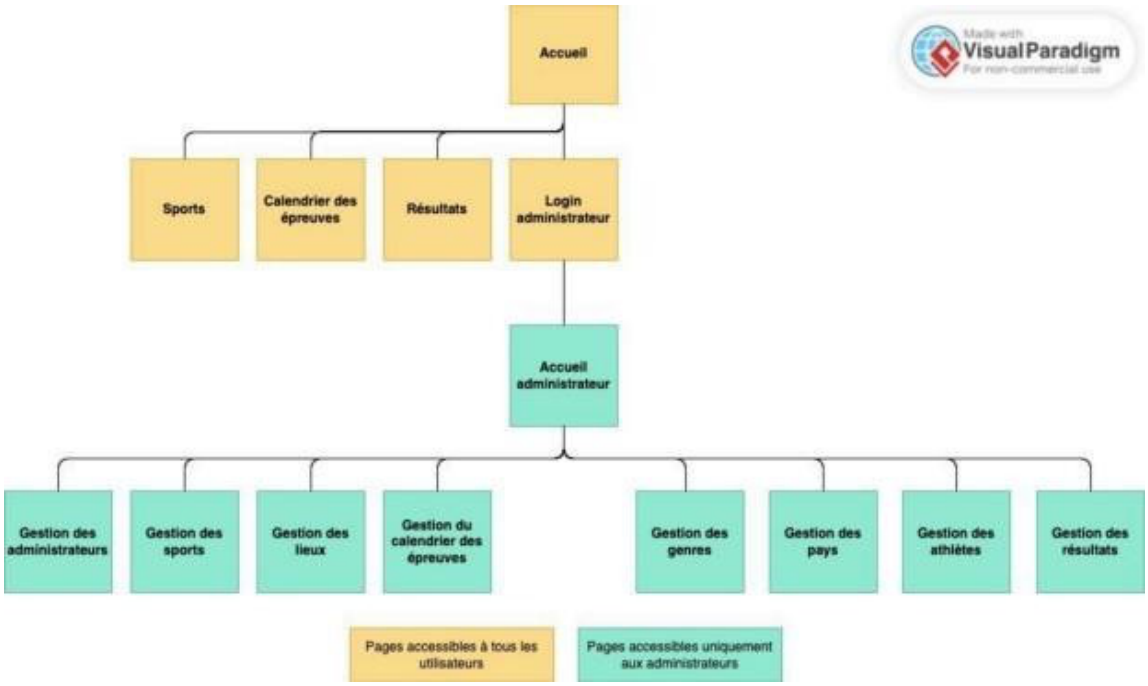
5.1.2. Maquettes





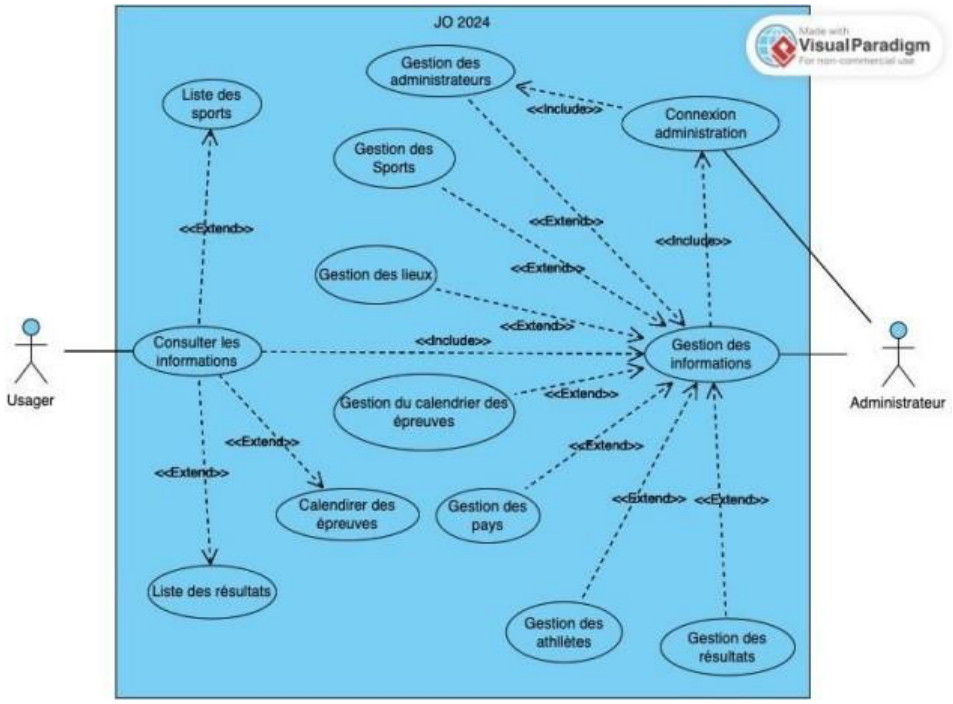


5.1.3. Arborescences

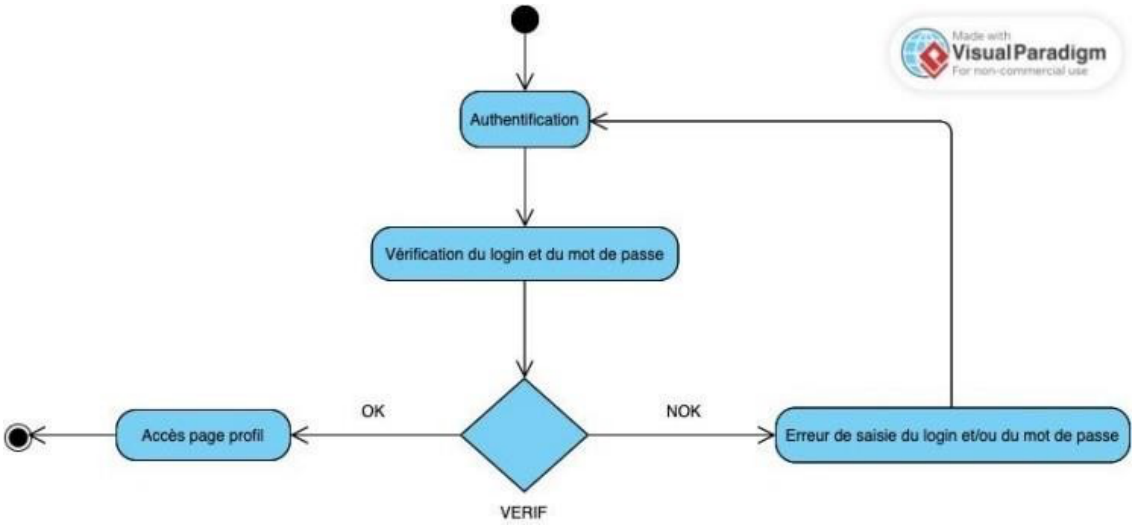


5.2. Le back-end

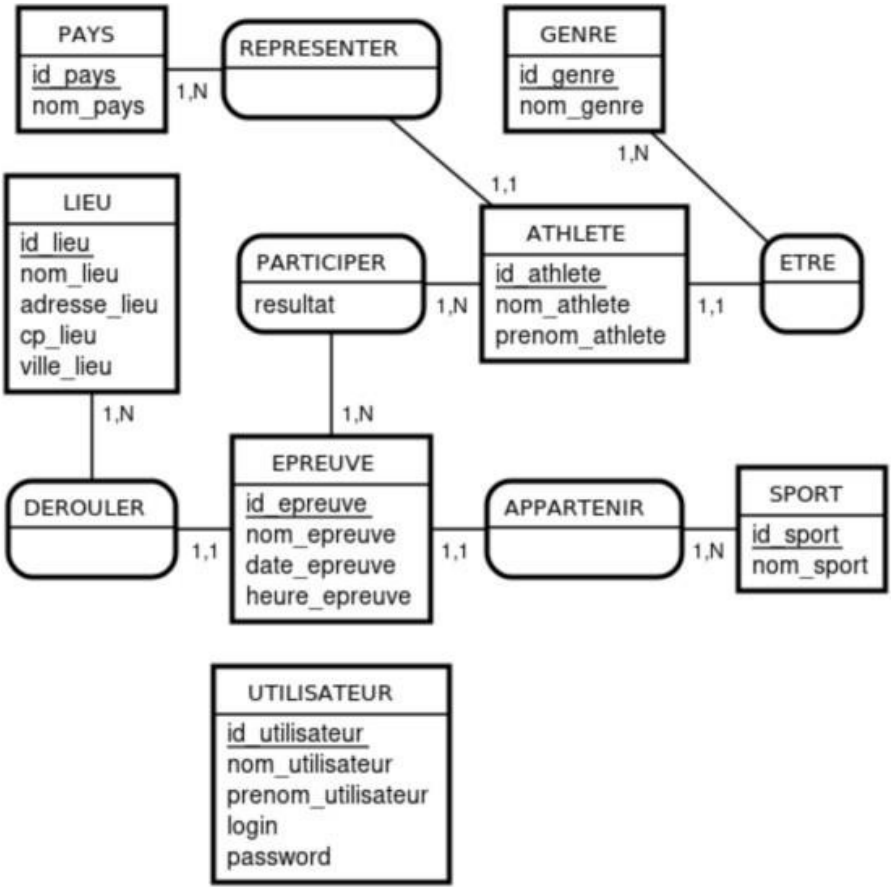
5.2.1. Diagramme de cas d'utilisation



5.2.2. Diagramme d'activités



5.2.3. Modèles Conceptuel de Données (MCD)

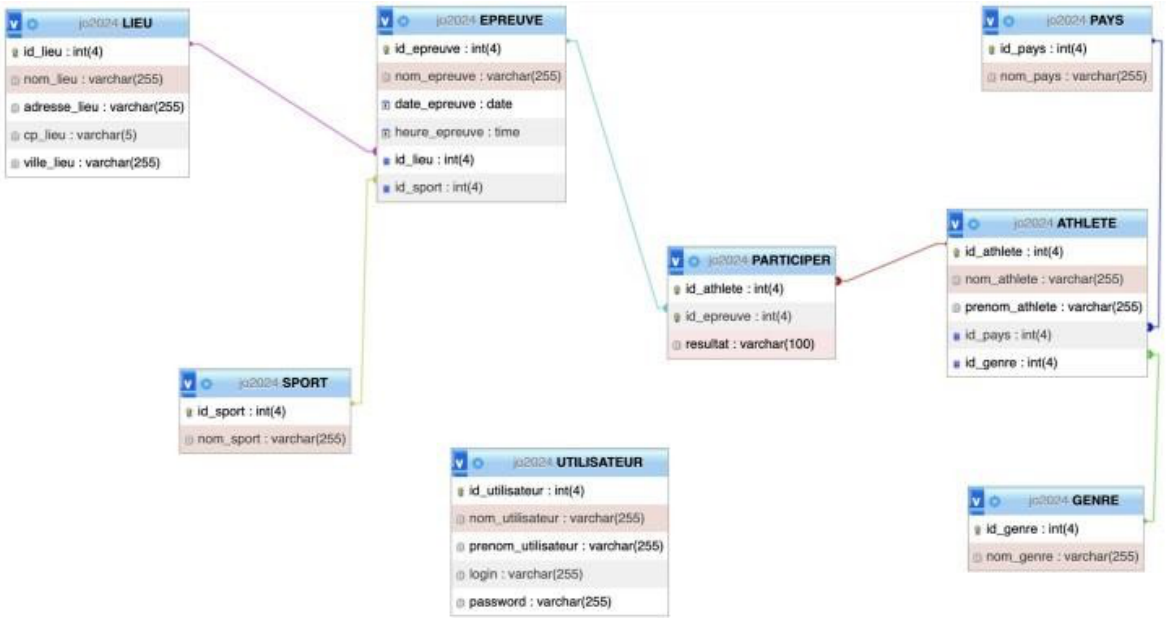


5.2.4. Modèle Logique de Données (MLD)

Modèle Logique de données (MLD) en utilisant mocodo

- 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)

5.2.5. Modèle Physique de Données (MPD)



6. Technologies utilisées

6.1. Langages de développement Web

Dans le cadre de la réalisation du projet, nous avons prévu d'utiliser les langages de développement web suivants, chacun jouant un rôle spécifique dans la création et l'interactivité des pages web :

- **HTML5**
- **CSS3**
- **JavaScript / JavaSript vanilla**
- **PHP8**
- **JS**
- **SQL (Pour la base de données)**

6.2. Base de données

Pour la gestion de la base de données, nous avons prévu d'utiliser les technologies suivantes, alliant un langage de programmation serveur à un langage de requête structuré :

- **PHP8**: Ce langage de programmation côté serveur sera utilisé pour dynamiser les interactions entre la base de données et l'interface de l'application web, facilitant ainsi le traitement des données et la génération de contenu dynamique.
- **SQL** : Langage de requête structuré indispensable pour la manipulation des données dans la base de données, SQL sera utilisé pour exécuter des requêtes, gérer les mises à jour et effectuer diverses opérations, garantissant la cohérence et l'intégrité des informations stockées.

7. Sécurité

7.1. Login et protection des pages administrateurs

Pour gérer l'authentification des utilisateurs, nous mettrons en place un système de connexion qui aura une authentification forte (par exemple, 2FA) et qui collectera les identifiants, tels que le nom d'utilisateur et le mot de passe.
Ces informations seront vérifiées en les comparant à celles enregistrées dans la base de données, afin de confirmer l'existence d'un compte valide (Vérification des rôles et des permissions avant d'accorder l'accès).
Avoir une base de donne qui enregistrera tous les tentative de connexion pour savoir qui c'est connecté.

7.2. Cryptage des mots de passe avec Bcrypt

Bcrypt est un algorithme de hachage conçu pour sécuriser les mots de passe en résistant aux attaques par force brute et dictionnaire. Il ajoute un "sels" (valeurs aleatoires uniques) unique à chaque mot de passe, garantissant que même des mots de passe identiques auront des hachages différents. De plus, il permet d'ajuster la complexité du calcul, ralentissant le processus de hachage et augmentant la sécurité avec le temps. En PHP, Bcrypt est facile à utiliser via la fonction `password_hash()` pour créer des hachages sécurisés et `password_verify()` pour vérifier les mots de passe lors de la connexion. Il est également possible d'ajuster le coût pour renforcer encore la sécurité. Code en Js :

```
const bcrypt = require('bcrypt');
const password = 'mySecretPassword';
const saltRounds = 10;

// Hacher le mot de passe
bcrypt.hash(password, saltRounds, (err, hash) => {
  if (err) throw err;
  console.log(`Mot de passe haché : ${hash}`);

  // Simuler la vérification avec un mot de passe utilisateur
  const userInput = 'mySecretPassword2';
  const storedHash = hash;  // On stocke le hash

  // Comparer le mot de passe entré avec le hash stocké
  bcrypt.compare(userInput, storedHash, (err, isMatch) => {
    if (err) throw err;
    if (isMatch) {
      console.log('Le mot de passe est correct.');
    } else {
      console.log('Le mot de passe est incorrect.');
    }
  });
});
```

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

Les attaques XSS (Cross-Site Scripting) se produisent lorsque des attaquants injectent du code malveillant, généralement en JavaScript, dans une page web vue par d'autres utilisateurs. Cela leur permet de voler des informations sensibles, comme des cookies de session. Pour se protéger contre les attaques XSS en PHP, il est essentiel d'échapper toutes les données avant de les afficher.

1. Échapper ou valider toutes les entrées utilisateur côté serveur et client.
2. Utiliser des politiques de Content Security Policy (CSP) exemple(interdire l'exécution de scripts inline.)
3. Utiliser des bibliothèque de sécurtité comme htmlspecialchars() qui transformer des caractères spéciaux comme <, >, et &

7.4. Protection contre les injections SQL

Les injections SQL sont des attaques visant à manipuler une base de données en injectant du code SQL malveillant via les entrées de l’application. Pour protéger une application contre ces attaques.

1.Utiliser des requêtes préparées (paramétrées) : Cela empêche l’injection de code SQL en séparant les instructions SQL des données fournies par l’utilisateur.

Exemple en PHP avec PDO :

Exemple simple en PHP :

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
$stmt->execute([$_GET['username']]);
```

2.Échapper les entrées utilisateur : En dernier recours, si les requêtes paramétrées ne sont pas possibles, il faut toujours échapper les données avant de les insérer dans une requête SQL.

Exemple simple en PHP :

```
$username = mysqli_real_escape_string($conn, $_GET['username']);
$query = "SELECT * FROM users WHERE username = '$username'";
```

3.Valider et filtrer les entrées : S’assurer que les données entrées respectent les types attendus (entiers, chaînes, etc.).

Exemple en PHP :

```
if (filter_var($_GET['email'], FILTER_VALIDATE_EMAIL)) {
    $email = $_GET['email'];
}
```

4.Limiter les privilèges de la base de données : L’utilisateur de la base de données utilisé par l’application doit avoir seulement les permissions nécessaires.