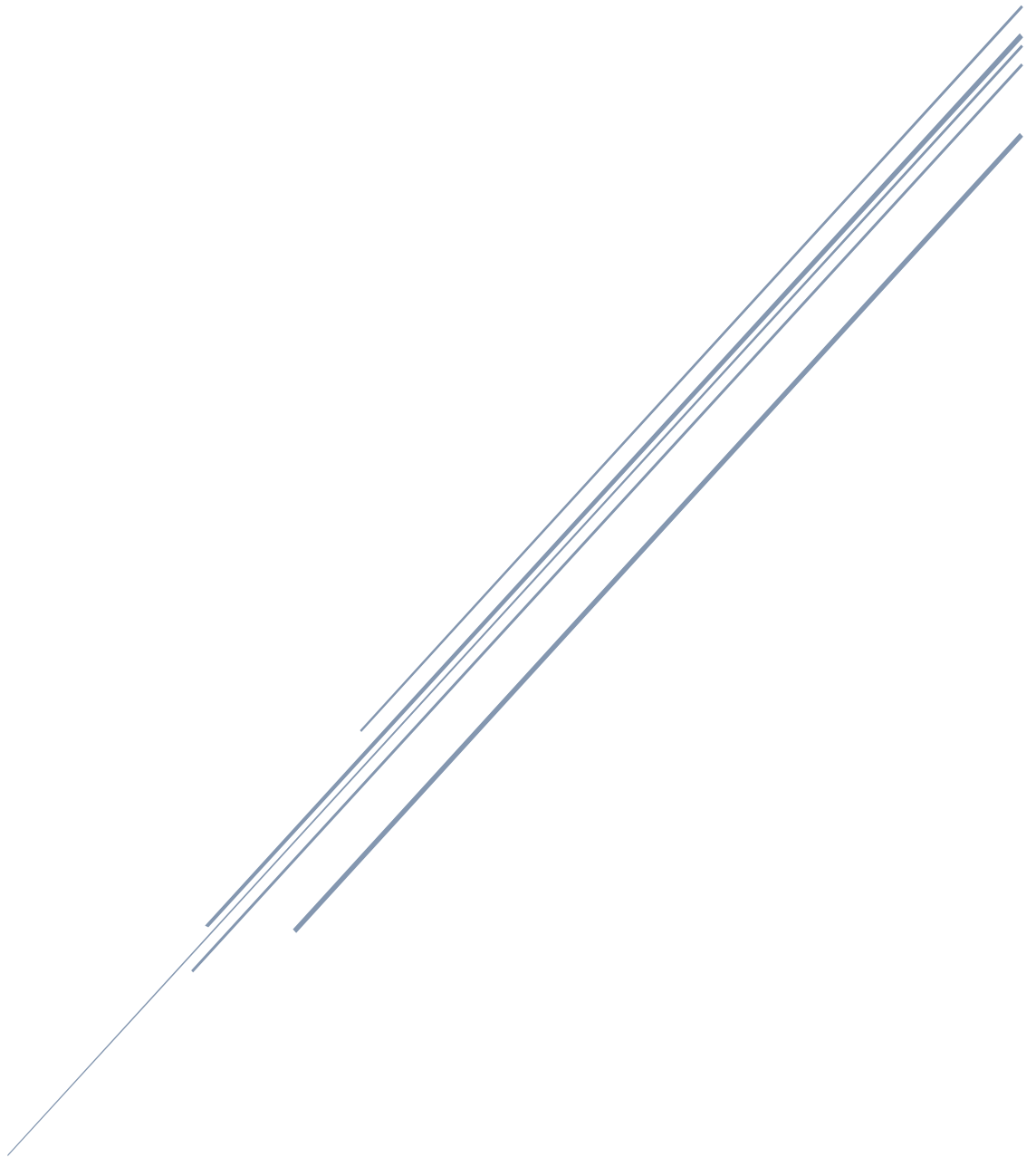


PINGFORM

BOURLET Cyril

Enseignant : Robert Tomczak



Résumé

Mon projet consiste à créer un système d'inscription en ligne pour un tournoi de tennis de table, incluant une page web pour l'organisateur, un formulaire dynamique pour les joueurs, et une base de données pour gérer automatiquement les inscrits.

Il vise à développer une plateforme d'inscription en ligne pour un tournoi de tennis de table, permettant aux organisateurs de générer un formulaire personnalisé et aux joueurs de s'inscrire facilement. Les informations des participants sont automatiquement stockées dans une base de données pour simplifier la gestion de la liste des inscrits.

Principaux résultats attendus :

- Une page principale avec la liste des tournois qui ont un formulaire.
- Une interface pour l'organisateur permettant de configurer le formulaire d'inscription.
- Un formulaire dynamique pour les joueurs, adapté aux informations du tournoi.
- Un système d'enregistrement automatisé qui stocke et organise les données des participants dans une base de données pour faciliter la gestion du tournoi.

Table des matières

Résumé	1
I. Première partie : partie commune	3
1) Présentation :	3
2) Analyse des besoins exigences fonctionnelles :	3
3) Cahier des charges : Sécurité -Cybersécurité	6
4) Phases, planification et répartition des tâches :	7
II. Partie personnelle : une partie par étudiant(e)	9
1) Introduction :	9
2) Conception :	10
3) Critères d'acceptation :	15
III. Rapport de mon travail	16
1. Introduction	16
2. Technologies et langages utilisés.....	17
2.1. Langages de programmation.....	18
2.2. Technologies de développement côté serveur	18
2.3. Base de données	19
2.4. Outils de développement.....	19
3. Structure et architecture du projet	20
3.1. Structure du projet	20

3.2. Description des dossiers	21
3.3. Architecture logique	23
4. Fonctionnalités réalisées	24
4.1. Fonctionnalités pour les joueurs	24
4.2. Fonctionnalités pour les organisateurs.....	25
4.3. Fonctionnalités générales	26
5. Développement du back-end	27
5.1. Mise en place du serveur Node.js avec Express	27
5.2. Création de l'API REST.....	27
5.3. Gestion des inscriptions.....	28
5.4. Connexion base de données	29
6. Développement du front-end	30
6.1. Pages HTML.....	30
6.2. CSS : design thème sombre.....	31
6.3. JavaScript : comportement dynamique	32
7. Gestion des rôles et sécurité.....	33
7.1. Rôles utilisateurs	33
7.2. Sécurité minimale.....	34
8. Difficultés rencontrées et solutions.....	37
8.1. Difficultés techniques	37
8.2. Solutions mises en place.....	38
9. Résultat obtenu	38
9.1. Fonctionnalités terminées	38
9.2. Démonstration	39
10. Améliorations futures.....	43
10.1. Sécurité	43
10.2. Nouvelles fonctionnalités.....	43
10.3. Responsive design	43
11. Conclusion	43
11.1. Bilan personnel.....	43
11.2. Ouverture	44

I. Première partie : partie commune

1) Présentation :

Lors de la création d'un tournoi, l'organisateur doit faire un formulaire où il introduit lui-même des informations tel que les séries (catégories) du tournoi, où le tournoi se déroule, et bien d'autres informations afin que le tournoi se déroule bien et que tout le monde soit informé correctement. Une fois le formulaire disponible, les joueurs voulant participer au tournoi, il remplit le formulaire avec leurs informations en tant que joueur et l'envoi. Lorsque le formulaire est envoyé, l'organisateur le reçoit et doit enregistrer l'inscription du joueur dans un tableau Excel par exemple pour que le juge-arbitre (celui qui gère le tournoi en lui-même et qui crée le déroulé des matchs) puisse créer des poules et ensuite le tableau. Cela demande beaucoup de temps et d'organisation, car dans un tournoi, il peut y avoir plus de 500 inscriptions sachant que les joueurs parfois s'inscrivent et se désinscrivent au dernier moment. Le fait de réaliser ce projet permettra donc de gagner du temps et d'être mieux organisé lors du déroulement d'un tournoi. Ce projet pourrait être utilisé plus tard par les associations de tennis de table où même d'autres sport, mais dans ce cas peut-être qu'il faudrait y apporter quelques modifications. Pourquoi pas par la suite, créer un menu déroulant pour choisir le sport que l'on veut comme le padel, badminton, tennis ou autre. Au tennis de table, pour faire une inscription, nous devons soit envoyer un mail ou un message ce qui peut créer des erreurs de saisie ou autre, ou soit l'organisateur créer un formulaire Google par exemple où il reprend les informations donc peut être plus clairement, mais il y a toujours les risques d'erreurs de saisie lors de l'enregistrement dans le fichier Excel. C'est donc pour cela que j'ai eu l'idée de ce projet.

Mon projet consiste à faciliter la création des fiches d'inscription à un tournoi de tennis de table donc pour l'organisateur du tournoi. Donc l'idée est de demander quelques informations à l'organisateur grâce à des questions simples qui en fonction des réponses données, va créer le formulaire d'inscription qui sera envoyé aux joueurs pour qui puissent s'inscrire aux tournois rapidement et facilement.

Lorsque que l'organisateur va rentrer le nom du club, les informations complémentaires seront recherchées à l'aide de la base de données qui sera faite. Quand les informations seront soumises par l'organisateur cela créera un formulaire automatique avec les différentes informations données auparavant.



Une fois le formulaire créé et envoyé aux joueurs, le joueur qui voudra s'inscrire devra entrer son numéro de licence (recherche dans une base de données) les informations seront automatiquement complétées. Le joueur devra ensuite cocher les séries (ce sont des catégories où il y a une limite de points pour que certains joueurs n'y ont pas accès.) dont il veut faire partie durant le tournoi. Quand le formulaire sera soumis, les informations seront mises dans une base de données pour pouvoir avoir la liste des inscrits avec la possibilité de recherche par série, par nom etc.

L'organisateur du tournoi pourra donc avoir accès à la base de données pour rechercher certains joueurs par critère.

Bien sûr si le joueur ayant rentré son numéro de licence et ayant 1865 points par exemple, il ne pourra pas s'inscrire dans une série (catégorie) limitée à 1800 points, ou encore dans une série féminine si le joueur est un garçon.

2) Analyse des besoins exigences fonctionnelles :

Description détaillée des besoins du logiciel

	Projet	
	COMPTE RENDU	

Le projet vise à créer un système de gestion des inscriptions pour un tournoi de tennis de table. Ce système doit permettre aux organisateurs de créer des tournois et aux joueurs de s'inscrire facilement à ces événements. Le logiciel doit être accessible via une interface web intuitive et permettre une gestion efficace des données des joueurs et des inscriptions.

Rédaction du cahier des charges fonctionnelles : liste détaillée des fonctionnalités du projet

- **Fonctionnalités pour les organisateurs :**
 - Création d'un tournoi avec des informations telles que le nom, la date, le lieu et les séries.
 - Modification et suppression des tournois existants.
 - Consultation de la liste des joueurs inscrits à un tournoi.
- **Fonctionnalités pour les joueurs :**
 - Inscription à un tournoi en remplissant un formulaire.
 - Consultation des tournois disponibles.
 - Modification ou annulation de leur inscription à un tournoi.
- **Fonctionnalités administratives :**
 - Gestion des comptes utilisateurs (organisateur et joueurs).
 - Gestion des séries et des règles spécifiques à chaque tournoi.
 - Exportation des données d'inscription pour des rapports statistiques.



Spécifications fonctionnelles

Description des interactions utilisateur

- **Organisateur :**
 - Se connecte à son compte pour accéder à l'interface de gestion des tournois.
 - Remplit un formulaire pour créer un nouveau tournoi.
 - Consulte la liste des joueurs inscrits et peut les contacter.
- **Joueur :**
 - Accède à la page des tournois disponibles.
 - Remplit un formulaire d'inscription à un tournoi.
 - Reçoit une confirmation de son inscription par e-mail.

Scénarios d'utilisation ou cas d'utilisation

- **Créer un tournoi :**
 - L'organisateur se connecte.
 - Il accède à la section "Créer un tournoi".
 - Il remplit les informations nécessaires (nom, date, lieu, séries).

	Projet	
	COMPTE RENDU	

- Il soumet le formulaire, et le tournoi est ajouté à la base de données.
- **S'inscrire à un tournoi :**
 - Le joueur accède à la liste des tournois.
 - Il sélectionne un tournoi et remplit le formulaire d'inscription.
 - Il soumet le formulaire, et les informations sont enregistrées dans la base de données.
- **Consulter les inscriptions :**
 - L'organisateur accède à la liste des tournois.
 - Il sélectionne un tournoi et consulte la liste des joueurs inscrits.

Spécifications non fonctionnelles

Exigences liées à la performance, la sécurité, l'ergonomie, etc.

- Performance : Le système doit pouvoir gérer un grand nombre d'inscriptions simultanées sans dégradation des performances.
- Sécurité : Les données des utilisateurs doivent être protégées par des mesures de sécurité appropriées (chiffrement des mots de passe, authentification sécurisée).
- Ergonomie : L'interface doit être intuitive et facile à utiliser, avec une navigation claire et accessible.

Normes et réglementations à respecter

- Respect des normes RGPD (Règlement Général sur la Protection des Données) pour la gestion des données personnelles des utilisateurs.
- Conformité avec les réglementations locales sur l'organisation d'événements sportifs.

Contraintes techniques

Plateformes et technologies à utiliser ou à éviter

- **À utiliser :**
 - Technologies web modernes (HTML5, CSS3, JavaScript, frameworks comme React ou Vue.js).
 - Backend en PHP, Node.js ou Python avec un framework adapté (Laravel, Express, Django).
 - Base de données relationnelle (MySQL ou PostgreSQL).
- **À éviter :**
 - Technologies obsolètes qui ne sont plus maintenues.
 - Systèmes qui n'offrent pas de compatibilité avec les normes de sécurité modernes.

Compatibilité avec d'autres systèmes ou logiciels

- Intégration possible avec des systèmes de gestion de bases de données pour l'exportation et l'importation de données.
- Compatibilité avec des services de messagerie pour l'envoi de confirmations d'inscription (par exemple, SMTP pour l'envoi d'e-mails).

3) Cahier des charges : Sécurité-Cybersécurité

Objectifs de sécurité

La sécurité de l'application web de gestion des tournois constitue une exigence majeure, afin de garantir la confidentialité, l'intégrité et la disponibilité des données. L'application manipule des informations personnelles relatives aux joueurs (nom, prénom, club, nombre de points, sexe, catégorie d'âge) et aux organisateurs. Il est donc primordial de protéger ces données contre les accès non autorisés et les mauvaises manipulations.

Les principaux objectifs de sécurité du projet sont :

- Assurer une authentification fiable des utilisateurs, notamment lors de la connexion.
- Gérer les rôles et permissions de manière stricte (distinction entre joueurs et organisateurs).
- Valider les données envoyées par les utilisateurs pour empêcher toute injection de contenu malveillant.
- Garantir la disponibilité des fonctionnalités, tout en empêchant les actions interdites selon le rôle de l'utilisateur.

Gestion des rôles et droits d'accès

Deux types de profils utilisateurs ont été définis :

Joueur

- Peut s'inscrire à un tournoi et sélectionner jusqu'à trois séries.
- Peut consulter la liste des tournois et des joueurs inscrits.

Organisateur

- Dispose de droits supplémentaires : création de tournois, association de séries, gestion des inscriptions.
- Peut accéder à des fonctionnalités de gestion non disponibles pour les joueurs.

Accès différencié :

- Le bouton "Créer un tournoi" est désactivé et masqué par défaut. Il s'affiche uniquement si l'utilisateur est connecté en tant qu'organisateur.
- L'application vérifie le rôle de l'utilisateur sur chaque page où des restrictions sont nécessaires.

Mesures de sécurité mises en œuvre

Authentification et contrôle d'accès

L'utilisateur doit s'authentifier via un formulaire de connexion avec adresse email et mot de passe.

Après une authentification réussie, les données du joueur (nom, prénom, licence, club, points, rôle) sont stockées localement dans le localStorage du navigateur.

Le rôle (est_organisateur) est exploité pour afficher ou non certaines fonctionnalités de l'interface.

Validation des données

Côté client :

- Les formulaires HTML utilisent les attributs required, type, min, max, etc., pour empêcher l'envoi de champs invalides.
- Les règles métiers sont vérifiées via JavaScript, par exemple :
 - Limitation à trois séries lors d'une inscription.
 - Vérification des points du joueur, de son sexe et de sa catégorie d'âge.

Côté serveur (back-end) :

- Toutes les données envoyées à l'API REST sont vérifiées côté serveur avec Express.js et Sequelize.

- Des contrôles supplémentaires sont effectués pour éviter les dépassements de quotas, les inscriptions multiples, et l'accès à des ressources non autorisées.

Gestion des erreurs et réponses HTTP

L'API renvoie des codes HTTP normalisés pour informer l'application des résultats d'une action :

- 200 OK pour une requête réussie.
- 400 Bad Request en cas de mauvaise saisie.
- 404 Not Found pour une ressource inexistante.
- 500 Internal Server Error pour toute erreur interne.

Les erreurs sont affichées clairement à l'utilisateur grâce à des messages adaptés, évitant ainsi toute confusion ou mauvaise interprétation.

Prévention des failles

- Sécurisation des appels API en filtrant et contrôlant les données échangées.
- Évitement du stockage de données sensibles en clair dans le navigateur (pas de mot de passe stocké dans le localStorage).
- Aucun traitement direct d'entrée utilisateur sans contrôle.
Exemple : les identifiants de série ou de tournoi sont toujours validés avant d'être utilisés en base de données.

Mesures de sécurité envisagées pour une mise en production

Bien que le projet se limite à un usage pédagogique, voici des pistes pour renforcer la sécurité dans une version destinée à un environnement réel :

- Hashage des mots de passe avec un algorithme sécurisé (comme bcrypt).
- Utilisation de tokens JWT (JSON Web Tokens) pour la gestion des sessions utilisateur côté serveur.
- Mise en place d'un système de middleware d'authentification pour sécuriser l'ensemble des routes API.
- Protection contre les attaques XSS et CSRF, notamment en filtrant les contenus injectés.
- Sécurisation HTTPS des échanges client-serveur pour protéger les données sensibles en transit.

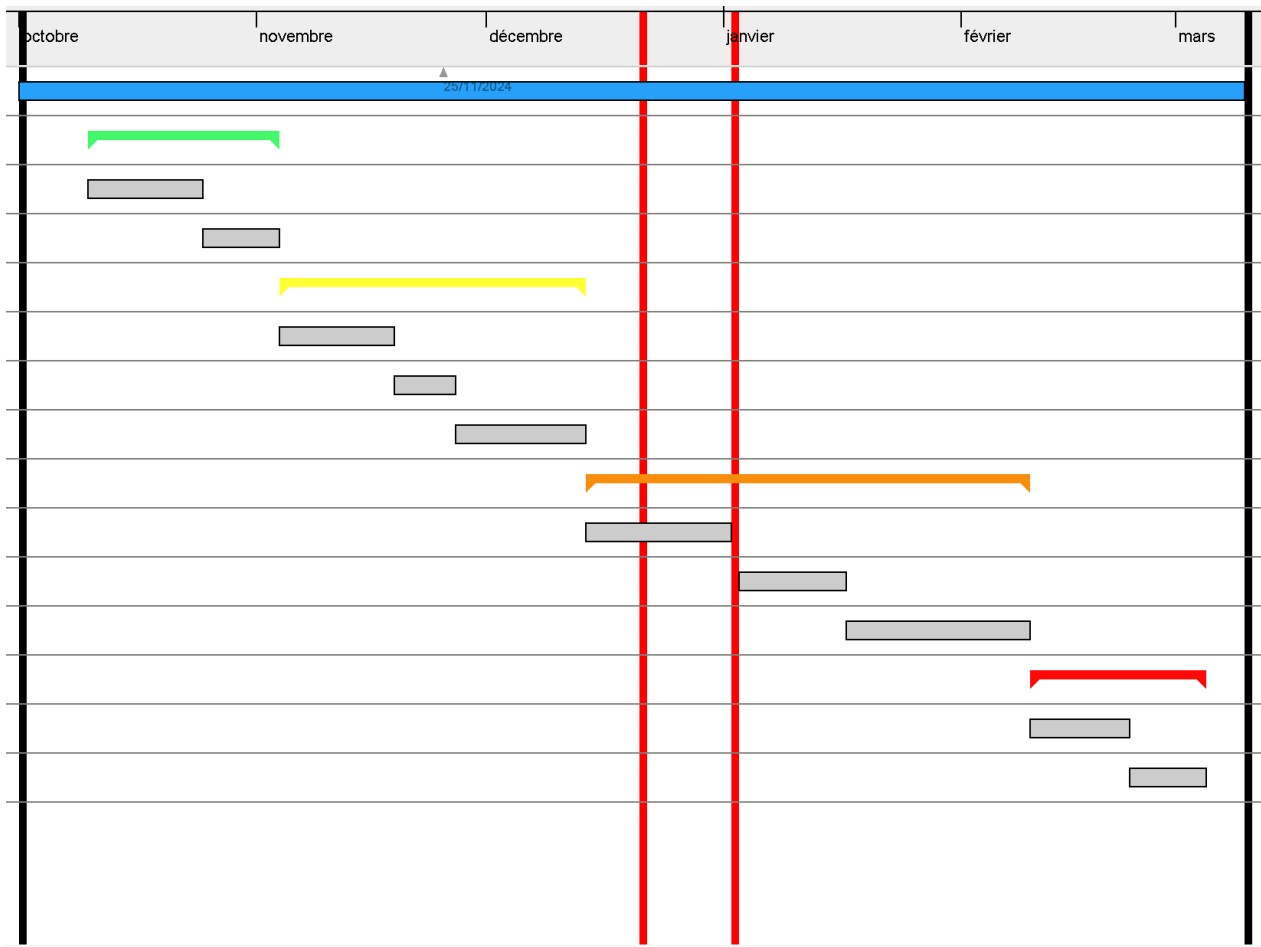
Synthèse

L'aspect sécurité de l'application, même s'il reste basique dans cette version de développement, respecte un minimum de bonnes pratiques :

- Gestion des rôles utilisateurs.
- Validation systématique des données côté client et serveur.
- Protection des points sensibles de l'application contre les erreurs de manipulation ou d'abus.

4) Phases, planification et répartition des tâches :

Diagramme de Gantt			Diagramme des Ressources		
GANTT project					
Nom			Date de début		Date de fin
Phase 0 : Documentation			01/10/2024		09/03/2025
▼ Phase 1 : Planification			10/10/2024		03/11/2024
Analyser les besoins			10/10/2024		24/10/2024
Définir les fonctionnalités			25/10/2024		03/11/2024
▼ Phase 2 : Dev. front-end			04/11/2024		13/12/2024
Créer la page d'accueil			04/11/2024		18/11/2024
Développer le formulaire			19/11/2024		26/11/2024
Interface Joueur/Organisateur			27/11/2024		13/12/2024
▼ Phase 3 : Dev. back-end			14/12/2024		09/02/2025
Conception de base de données			14/12/2024		01/01/2025
Intégration Formulaire/BDD			03/01/2025		16/01/2025
Vérifications automatiques			17/01/2025		09/02/2025
▼ Phase 4 : Tests			10/02/2025		04/03/2025
Inscription des joueurs			10/02/2025		22/02/2025
Corriger les bugs			23/02/2025		04/03/2025



II. Partie personnelle : une partie par étudiant(e)

1) Introduction :

Le projet PingForm a pour objectif de simplifier et d'automatiser le processus d'inscription à un tournoi de tennis de table. Les tâches principales consistent à :

- 1- Développer une interface web pour l'organisateur du tournoi, lui permettant de configurer et générer un formulaire d'inscription personnalisé en fonction des informations spécifiques au tournoi (nom, date, lieu, etc.).
- 2- Créer un formulaire dynamique pour les joueurs qui souhaitent participer, afin qu'ils puissent saisir leurs informations (nom, contact, etc.) et soumettre leur inscription en ligne.
- 3- Concevoir une base de données pour stocker et organiser les informations des participants, permettant une gestion facilitée et centralisée de la liste des inscrits.

Ce projet s'inscrit dans une démarche de digitalisation et d'optimisation de la gestion des tournois de tennis de table, permettant aux organisateurs de gagner du temps et de réduire les erreurs manuelles liées aux inscriptions papier ou non centralisées. Il répond à une demande accrue pour des solutions en ligne, qui offrent aux participants un accès rapide et simplifié au tournoi tout en allégeant la charge de travail des organisateurs.

2) Conception :

Diagramme de cas d'utilisation :

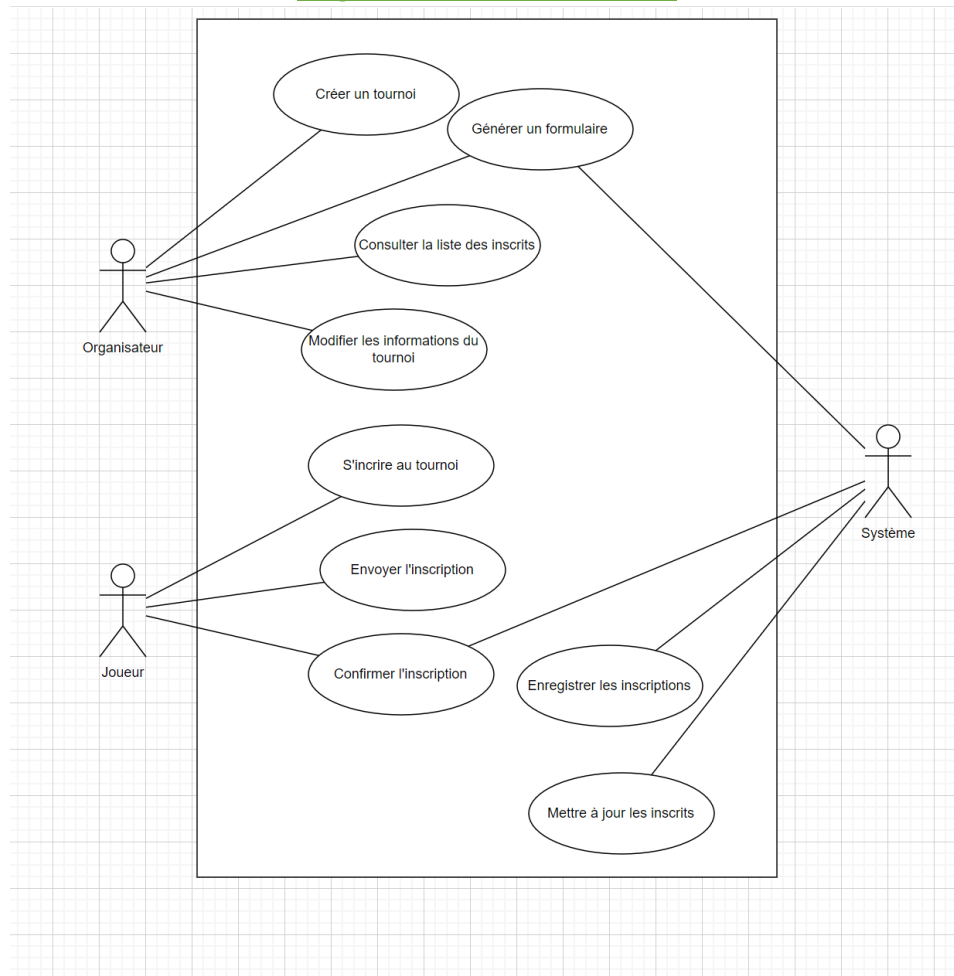
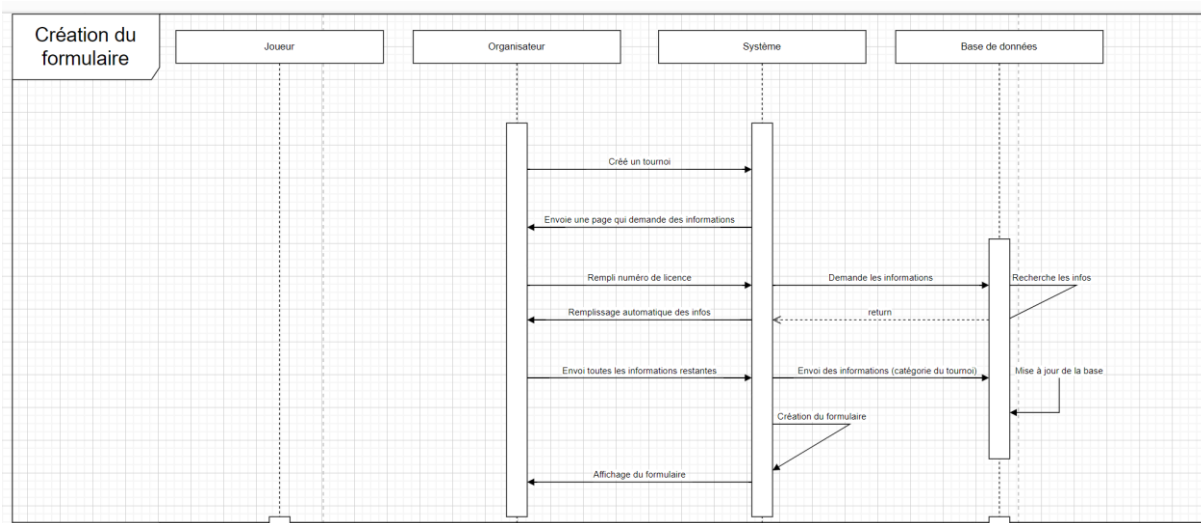


Diagramme de séquence :



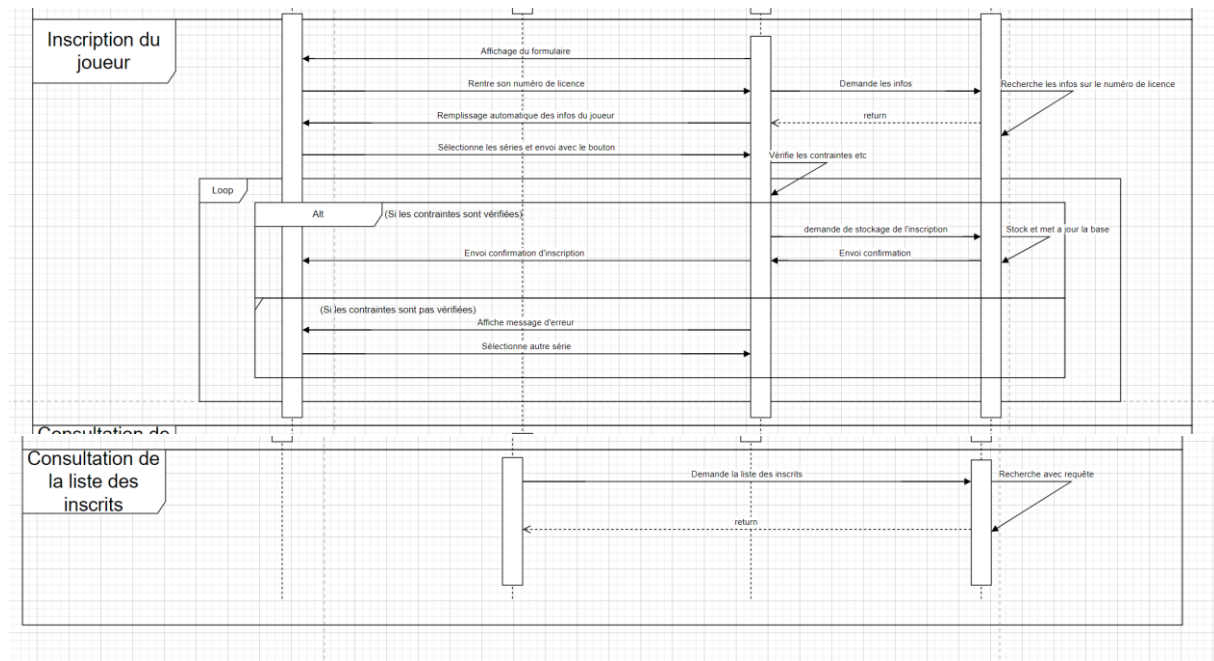
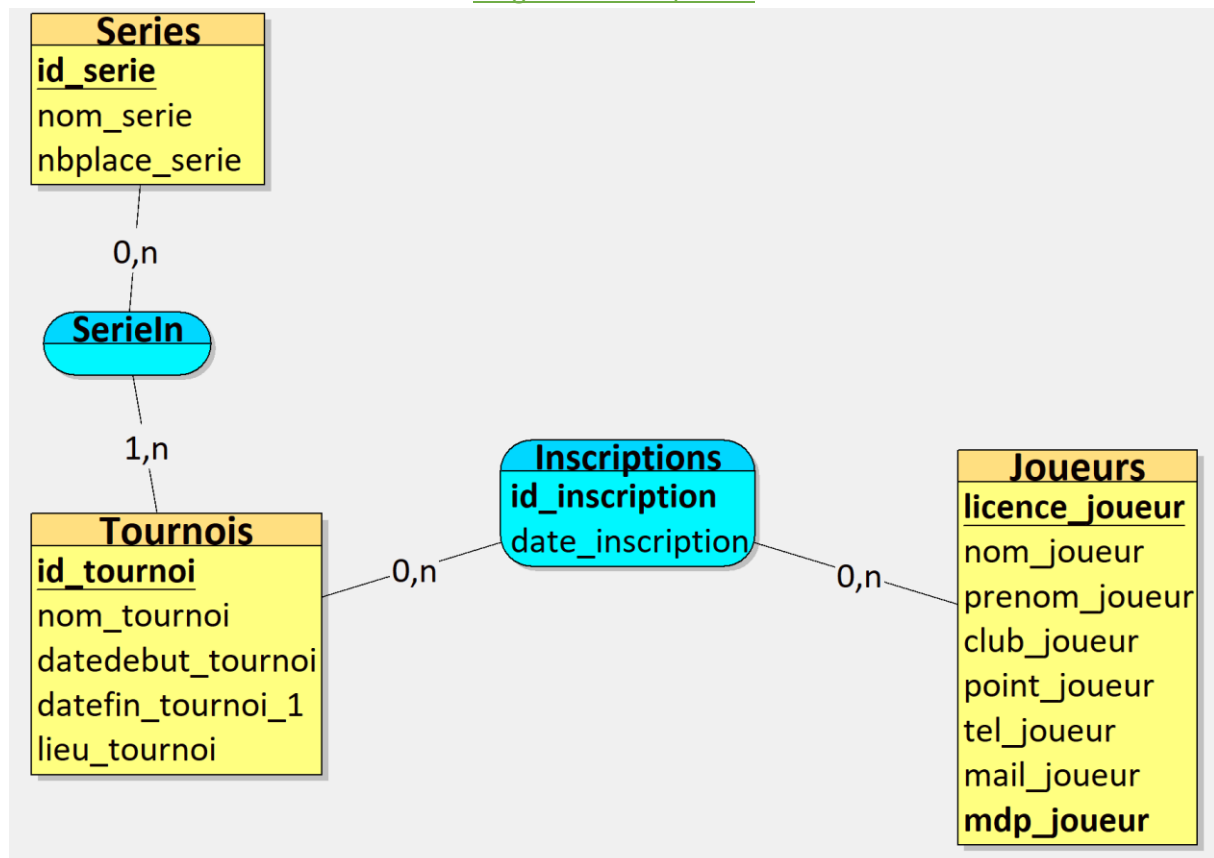
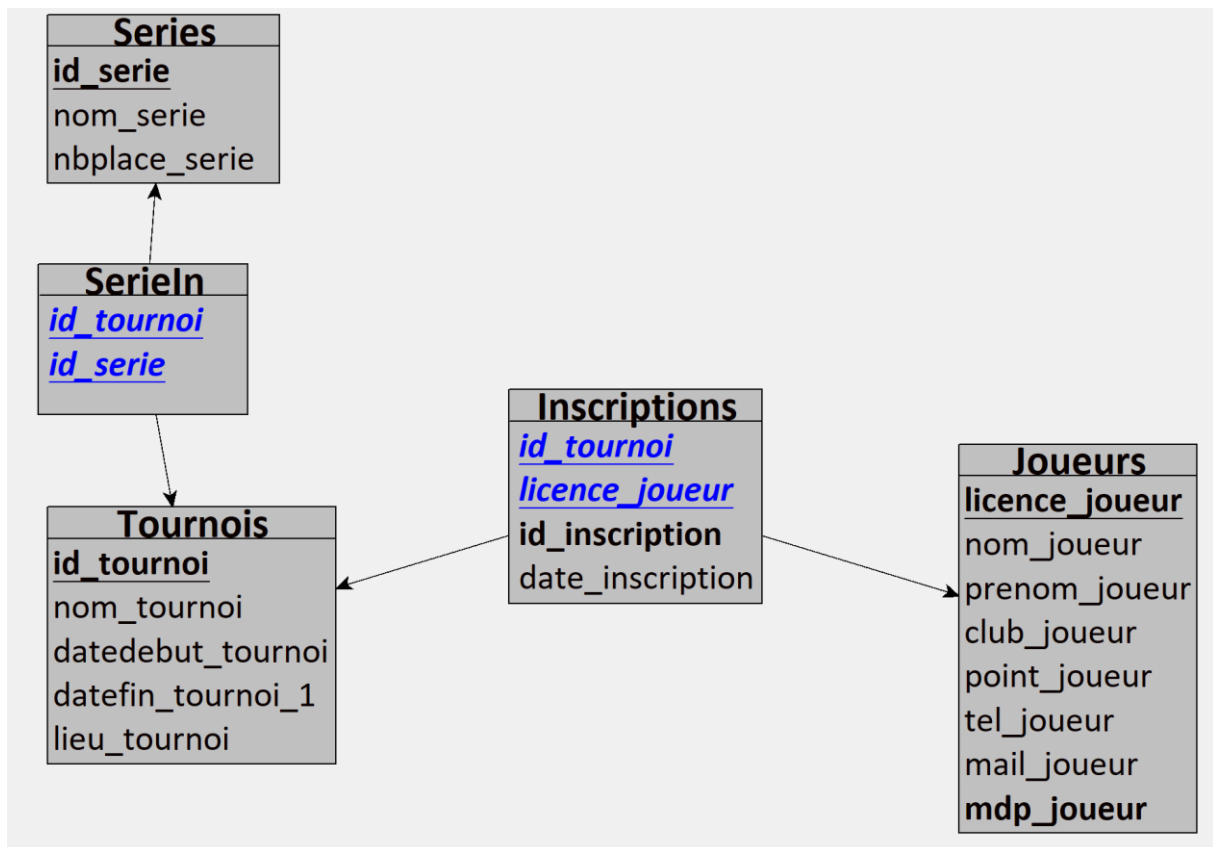


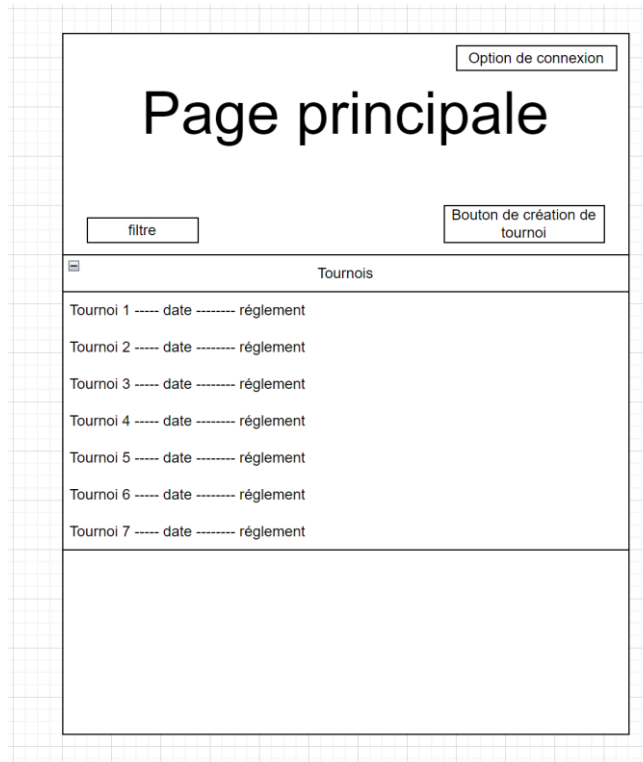
Diagramme UML/MLD :





Le modèle relationnel de données correspond au MLD

Page principale :

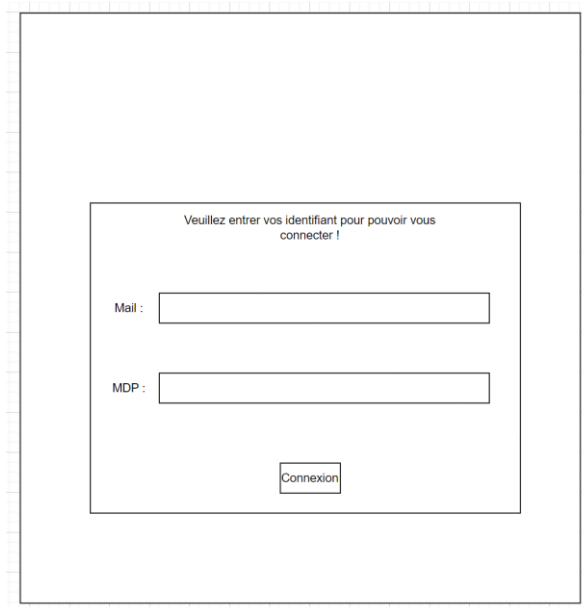


The screenshot shows a web interface titled "Page principale". At the top right is a button labeled "Option de connexion". Below the title, there are two buttons: "filtre" on the left and "Bouton de création de tournoi" on the right. A table titled "Tournois" is displayed, containing seven rows of data. Each row represents a tournament and includes columns for the tournament name, date, and règlement (rules). The rows are labeled "Tournoi 1" through "Tournoi 7". Below the table is a large empty rectangular box.

Ceci est la page principale où la liste de tous les tournois seront affichés. Chaque ligne de la liste correspondra à un tournoi différent, et sera cliquable pour que l'utilisateur puisse avoir accès à son formulaire d'inscription personnalisé.

Un utilisateur pourra donc cliquer sur la ligne du tournoi dont il veut s'inscrire pour être envoyé sur la page du formulaire associé à ce tournoi.

De plus, il y a le bouton d'option de connexion pour permettre à l'utilisateur de s'identifier en tant que joueur ou organisateur. Le bouton de création d'un tournoi est seulement cliquable lorsque l'utilisateur se connecte en tant qu'organisateur.



Veillez entrer vos identifiant pour pouvoir vous connecter !

Mail :

MDP :

[Page de création :](#)

Création du formulaire d'inscription

En fonction du nombre donné

Nom série 1

Nom série 2

Nom série 3

En fonction du nombre donné

Nom série 1

Nom série 2

Nom série 3

Cette page apparaît quand il y a un organisateur qui veut créer un tournoi, donc les informations qui peuvent être remplis automatiquement sont remplis et les autres informations nécessaires pour créer le tournoi sont remplis manuellement par l'organisateur avec les différentes séries correspondant au jour du tournoi, le règlement du tournoi, etc...

Emplacement prédéfinis des informations remplis :

Création du formulaire d'inscription

En fonction du nombre donné

Nom série 1

Nom série 2


Nom série 3

En fonction du nombre donné

Nom série 1

Nom série 2

Nom série 3



Tournoi avec nom du club

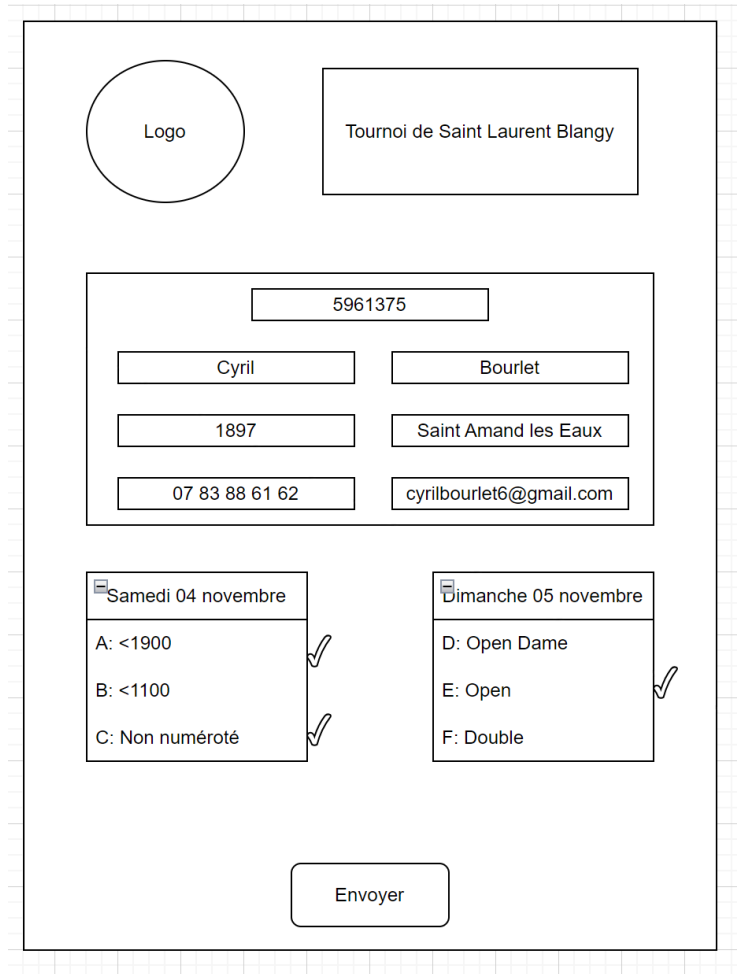
Série jour 1
 Série 1
 Série 2
 Série 3

Série jour 2
 Série 1
 Série 2
 Série 3

les infos sur le tournoi : Adresse règlement etc...

Le schéma de gauche représente une page initiale où l'organisateur du tournoi va rentrer les informations nécessaires et celui de droite représente le formulaire créer automatiquement et les informations de la page de gauche se mettront directement aux emplacements définis à droite.

Formulaire :



The form is titled 'Tournoi de Saint Laurent Blangy'. It contains a 'Logo' placeholder. A central box contains the following fields: a number '5961375', a name 'Cyril Bourlet', a year '1897', a phone number '07 83 88 61 62', and an email 'cyrilbourlet6@gmail.com'. Below this, there are two columns for selecting tournament series. The left column is for 'Samedi 04 novembre' and includes options 'A: <1900', 'B: <1100', and 'C: Non numéroté', each with a checkmark. The right column is for 'Dimanche 05 novembre' and includes options 'D: Open Dame', 'E: Open', and 'F: Double', each with a checkmark. At the bottom center is an 'Envoyer' button.

La page du formulaire apparaît lorsque le joueur clique sur le tournoi dont il veut faire partie, l'encadrer des informations sur le joueur se remplit automatiquement. Le joueur coche juste les séries qu'il veut faire et quand il coche, en back-end il y a les différentes vérifications pour savoir si le joueur peut s'inscrire dans la série qu'il coche. S'il ne peut pas s'inscrire il y a un message qui s'affiche.

3) Critères d'acceptation :

Conditions à remplir pour que le projet soit considéré comme terminé

1) Fonctionnalités complètes :

- L'ensemble des fonctionnalités définies dans le cahier des charges fonctionnelles doit être implémenté, y compris la création de tournois, l'inscription des joueurs, et la gestion des comptes utilisateurs.

2) Tests fonctionnels réussis :

- Tous les tests fonctionnels doivent être réalisés et les résultats doivent être conformes aux attentes. Chaque fonctionnalité doit être validée à travers des scénarios d'utilisation définis.

3) Sécurité et performance :

- Les mesures de sécurité doivent être mises en place, y compris le chiffrement des mots de passe et la protection des données personnelles. Les tests de performance doivent démontrer que le système peut gérer un nombre élevé d'utilisateurs simultanément sans ralentissement.

4) Interface utilisateur :

- L'interface doit être intuitive et conviviale. Les retours des utilisateurs doivent être pris en compte et les améliorations nécessaires doivent être apportées avant la livraison finale.

5) **Documentation :**

- Toute la documentation technique et utilisateur doit être complétée, incluant le guide d'utilisation pour les organisateurs et les joueurs, ainsi que les spécifications techniques du système.

6) **Conformité réglementaire :**

- Le projet doit être conforme aux réglementations en vigueur, notamment le RGPD pour la gestion des données personnelles des utilisateurs.

Méthodes de validation et de vérification

1) **Tests unitaires :**

- Des tests unitaires doivent être réalisés pour chaque module de code afin de s'assurer que chaque fonctionnalité fonctionne individuellement comme prévu.

2) **Tests d'intégration :**

- Des tests d'intégration doivent être effectués pour vérifier que les différents modules interagissent correctement entre eux et que les données circulent comme attendu dans le système.

3) **Tests fonctionnels :**

- Des scénarios d'utilisation basés sur le cahier des charges doivent être exécutés pour valider que toutes les fonctionnalités répondent aux exigences spécifiées.

4) **Tests d'acceptation utilisateurs (UAT) :**

- Une phase de tests d'acceptation utilisateurs doit être prévue où des utilisateurs finaux (organismes et joueurs) testeront le système pour s'assurer qu'il répond à leurs besoins et attentes.

III. Rapport de mon travail

1. Introduction

Dans le cadre du projet de fin de licence informatique, j'ai été amené à concevoir et développer une application web complète, couvrant le front-end (interface utilisateur) et le back-end (gestion des données et logique métier).

Le projet s'inscrit dans un besoin croissant de digitalisation des événements sportifs, notamment dans le domaine du tennis de table. Aujourd'hui encore, de nombreux tournois s'organisent manuellement via des fichiers papier ou tableurs. L'idée est d'apporter une solution moderne, fiable et simple d'utilisation, qui peut être déployée à l'échelle d'un club ou d'une fédération.

Le projet consiste à développer une application web de gestion de tournois de tennis de table. L'objectif principal est de fournir une plateforme permettant aux joueurs de s'inscrire facilement à des tournois, tout en facilitant aux organisateurs la gestion des inscriptions, la création des tournois et le suivi des participants.

Ce site propose une interface conviviale, à la fois pour les joueurs et les organisateurs, en simplifiant les interactions administratives habituelles. Le projet répond à un besoin réel d'automatiser la gestion des tournois, d'en centraliser les données et d'en améliorer l'accessibilité.

Le projet vise plusieurs objectifs :

- Proposer une interface d'inscription en ligne pour les joueurs.
- Permettre aux organisateurs de créer des tournois et d'en définir les modalités (dates, lieux, séries...).
- Gérer les inscriptions aux séries, avec des règles de validation précises (sexe autorisé, limite de points, catégorie d'âge...).
- Assurer une séparation des rôles : les fonctionnalités disponibles diffèrent selon que l'utilisateur est joueur ou organisateur.

- Offrir une expérience utilisateur fluide grâce à une interface simple et moderne, avec un design en thème sombre.
- Stocker les données de manière fiable et sécurisée dans une base de données relationnelle.
- Rendre l'application responsive et accessible sur différents types de supports (PC, tablette).

Le développement du projet s'est déroulé en plusieurs étapes successives :

- Analyse des besoins : identification des fonctionnalités principales et des cas d'usage pour les joueurs et organisateurs.
- Modélisation des données : création des modèles de base de données (Joueur, Tournoi, Série, Inscription).
- Conception de l'architecture : définition de la structure du projet en suivant un modèle MVC simplifié (Modèles, Routes/Contrôleurs, Vues statiques).
- Développement back-end : création d'une API REST à l'aide de Node.js et Express.js, avec une base de données MariaDB gérée via Sequelize.
- Développement front-end : réalisation des pages HTML, intégration CSS (thème sombre) et programmation en JavaScript vanilla pour la logique client.
- Tests fonctionnels : vérification des fonctionnalités principales via Postman, Thunder Client et tests utilisateurs.
- Mise en place des contrôles d'accès et de la logique de rôles (joueur/organisateur).
- Amélioration du design et du confort d'utilisation (UI/UX).

Accueil - Tournois

Jean Dupont
Déconnexion

Ajouter un tournoi

Trier par :

Numéro d'ID

Appliquer le tri

ID	Nom	Date Début	Date Fin	Lieu	Liste des inscrits
1	Championnat National	2024-06-15	2024-06-16	Paris	Voir les inscrits
2	Open Régional	2024-07-10	2024-07-11	Lyon	Voir les inscrits
3	Tournoi Jeunes	2024-08-05	2024-08-06	Marseille	Voir les inscrits
4	Coupe des Vétérans	2024-09-12	2024-09-13	Bordeaux	Voir les inscrits
5	Open Mixte	2024-10-20	2024-10-21	Toulouse	Voir les inscrits
6	challenge triith	2025-03-22	2025-03-23	trith saint leger	Voir les inscrits
7	Tournoi National B Saint Laurent Blangy	2025-04-05	2025-04-06	Saint Laurent Blangy	Voir les inscrits
8	Tournoi 3	2025-06-07	2025-06-08	Saint amand les eaux	Voir les inscrits

2. Technologies et langages utilisés

Le développement de l'application de gestion de tournois de tennis de table repose sur un ensemble de technologies modernes. Le projet suit une architecture full-stack JavaScript, permettant un développement fluide et cohérent sur l'ensemble de l'application, aussi bien côté client (front-end) que côté serveur (back-end).

Cette partie présente en détail les technologies, les langages de programmation ainsi que les outils utilisés tout au long du développement.

2.1. Langages de programmation

HTML

Le HTML5 (HyperText Markup Language) est le langage utilisé pour structurer les pages web de l'application. Il a permis de concevoir :

- La structure sémantique des différentes interfaces (Accueil, Connexion, Formulaire d'inscription, etc.).
- Les formulaires d'inscription et de connexion.
- L'intégration des tableaux pour l'affichage des tournois et des inscrits.

Le HTML5 a également facilité la compatibilité avec les navigateurs modernes et l'accessibilité, grâce à des balises spécifiques (<header>, <main>, <section>, etc.).

CSS

Le CSS (Cascading Style Sheets) a été utilisé pour gérer la présentation graphique du site :

- Mise en forme des différentes pages et composants (boutons, formulaires, tableaux...).
- Gestion de la responsivité afin d'assurer un affichage correct sur différents types d'appareils (ordinateurs, tablettes).
- Conception d'un thème sombre pour offrir une meilleure lisibilité et un design moderne, en adéquation avec les tendances actuelles.
- Personnalisation des éléments interactifs, notamment avec des transitions et des effets au survol (hover).

JavaScript

Le JavaScript natif, ou Vanilla JS, a été choisi pour apporter du dynamisme aux pages web :

- Gestion des événements utilisateurs (clics sur les boutons, formulaires d'inscription...).
- Réalisation d'appels asynchrones (AJAX / Fetch API) vers l'API RESTful du serveur, afin de récupérer et afficher dynamiquement les données (liste des tournois, inscrits...).
- Manipulation du DOM (Document Object Model) pour afficher les données sans recharger la page.
- Validation simple des formulaires côté client avant envoi des données au serveur.

Le choix de Vanilla JS permet de garder un code léger, sans dépendre de frameworks lourds comme React ou Vue.js, ce qui est suffisant dans le cadre d'un projet de cette ampleur.

2.2. Technologies de développement côté serveur

Node.js

Le back-end repose sur Node.js, un environnement d'exécution JavaScript orienté serveur.

Node.js est :

- Basé sur un modèle asynchrone et événementiel, ce qui le rend performant et adapté aux applications nécessitant un traitement rapide des requêtes.
- Parfaitement adapté à la création d'API REST.
- Compatible avec le même langage que celui utilisé en front-end (JavaScript), facilitant la maintenance et le développement.

Express.js

Express.js est le framework utilisé pour construire l'API REST de l'application :

- Il simplifie la création de routes HTTP et la gestion des middlewares (authentification, traitement JSON...).
- Il offre une grande flexibilité et une rapidité de développement grâce à sa légèreté.
- Les routes mises en place permettent de gérer les fonctionnalités principales :
 - Connexion des utilisateurs
 - Création de tournois et séries

- Inscriptions des joueurs
- Consultation des inscrits par tournoi

Le serveur Express permet également de servir les fichiers statiques de l'application (HTML, CSS, JS du front-end).

2.3. Base de données

MariaDB

La base de données relationnelle choisie est MariaDB, un fork de MySQL.

Elle a été sélectionnée pour :

- Sa robustesse, fiabilité et performance dans la gestion de grandes quantités de données.
- Son support complet des transactions, garantissant l'intégrité des données.
- Sa compatibilité avec de nombreux ORM, dont Sequelize.

MariaDB est utilisée pour stocker :

- Les utilisateurs (joueurs/organisateur) et leurs informations personnelles.
- Les tournois, avec leurs dates, lieux et détails.
- Les séries, et leurs contraintes (limite de points, sexe autorisé, etc.).
- Les inscriptions, avec le lien entre joueurs, séries et tournois.

Sequelize

Sequelize est l'ORM (Object-Relational Mapping) utilisé pour interagir avec MariaDB :

- Il permet de manipuler les données via des objets JavaScript sans écrire de requêtes SQL complexes.
- Il facilite la création et la gestion des modèles de données (Joueur, Tournoi, Serie, Inscription).
- Il prend en charge la synchronisation automatique des tables, les relations (associations entre modèles) et les requêtes complexes.
- Sequelize permet également une meilleure portabilité si un changement de SGBD devait intervenir dans le futur.

2.4. Outils de développement

Visual Studio Code (VS Code)

Le développement a été réalisé sur Visual Studio Code, un IDE puissant et léger :

- Prise en charge complète de JavaScript, HTML et CSS.
- Extensions utiles comme ESLint (analyse du code), Prettier (formatage automatique) et Sequelize Snippets (aide à la modélisation des données).
- Terminal intégré facilitant l'exécution de commandes Node.js et Git.

Postman

Cet outil a permis de :

- Tester les différentes routes de l'API REST (GET, POST, PUT, DELETE).
- Simuler les requêtes HTTP pour s'assurer du bon fonctionnement des contrôles d'accès, de la validation des données et du traitement des erreurs.
- Vérifier les codes de réponse et le format des données JSON échangées entre le client et le serveur.
- Thunder Client, intégré à VS Code, a été particulièrement pratique pour effectuer des tests rapides sans sortir de l'éditeur.

Git

Git sert à la gestion du versioning. Cependant je ne sais pas l'utiliser efficacement, donc je ne vois pas l'évolution de mon projet étape par étape alors que cela aurait été un grand avantage. De plus Git permet de sauvegarder le projet en cas de problème, on a toujours la possibilité de récupérer la version la plus récente du projet, ce qui est aussi un avantage majeur.

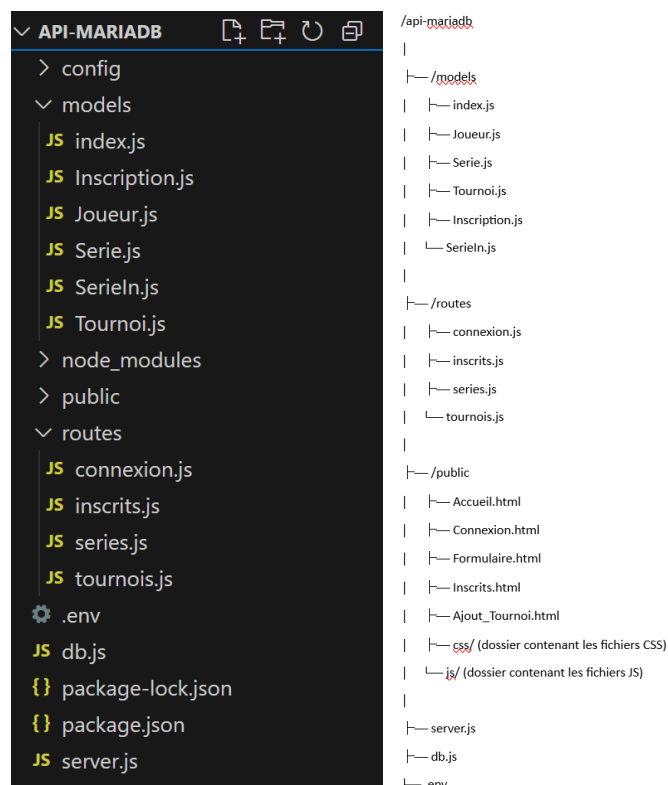
3. Structure et architecture du projet

Le projet repose sur une architecture claire, modulaire et évolutive, facilitant son développement et sa maintenance. Cette structure est inspirée du modèle MVC simplifié (Modèle-Vue-Contrôleur), ce qui garantit une séparation nette entre les différentes couches de l'application : la gestion des données, la logique métier, et l'interface utilisateur.

Dans cette partie, nous allons détailler l'organisation des fichiers et des dossiers, puis expliquer le fonctionnement de l'architecture logique adoptée.

3.1. Structure du projet

L'arborescence de mon projet est la suivante :



3.2. Description des dossiers

- /models : définition des modèles Sequelize (Joueur, Serie, Tournoi, Inscription...).

```
models > JS Inscription.js > <unknown> > exports > id_tournoi
1  const { DataTypes, Model } = require('sequelize');
2
3  module.exports = (sequelize) => {
4    class Inscription extends Model {} // création du modèle Inscription
5
6    Inscription.init({
7      id_inscription: {
8        type: DataTypes.INTEGER,      // identifiant unique de l'inscription
9        primaryKey: true,             // clé primaire
10       autoIncrement: true           // incrémentation automatique
11     },
12     id_tournoi: {
13       type: DataTypes.INTEGER,      // identifiant du tournoi associé
14       allowNull: false,             // obligatoire
15       references: {                 // clé étrangère vers la table Tournois
16         model: 'Tournois',
17         key: 'id_tournoi'
18       },
19     },
20     licence_joueur: {
21       type: DataTypes.INTEGER,      // identifiant du joueur (licence)
22       allowNull: false,             // obligatoire
23       references: {                 // clé étrangère vers la table Joueurs
24         model: 'Joueurs',
25         key: 'licence_joueur'
```

- /routes : définition des routes API pour gérer les entités.

```
// Route GET : Liste des inscrits d'un tournoi
router.get('/:id_tournoi', async (req, res) => {
  try {
    const { id_tournoi } = req.params;

    // Recherche toutes les inscriptions d'un tournoi donné
    const inscriptions = await Inscription.findAll({
      where: { id_tournoi },
      include: [
        {
          model: Joueur,
          attributes: ['licence_joueur', 'nom_joueur', 'prenom_joueur', 'club_joueur', 'point_joueur']
        },
        {
          model: Serie,
          attributes: ['nom_serie']
        }
      ]
    });

    // Si aucune inscription trouvée, retourne un message
    if (!inscriptions || inscriptions.length === 0) {
      return res.status(404).json({ message: "Aucun joueur inscrit pour ce tournoi." });
    }
  }
});
```

```
// Route POST : Inscrire un joueur à une ou plusieurs séries
router.post('/', async (req, res) => {
  try {
    const { id_tournoi, licence_joueur, series } = req.body;

    // Vérifie que les informations sont complètes
    if (!id_tournoi || !licence_joueur || !series || series.length === 0) {
      return res.status(400).json({ message: "Tous les champs sont requis." });
    }

    // Recherche le joueur concerné
    const joueur = await Joueur.findOne({ where: { licence_joueur } });
    if (!joueur) {
      return res.status(404).json({ message: "Joueur non trouvé." });
    }

    // Recherche les inscriptions existantes de ce joueur dans ce tournoi
    const inscriptionsExistantes = await Inscription.findAll({
      where: { id_tournoi, licence_joueur }
    });

    // Vérifie si le joueur dépasse le nombre max d'inscriptions (3 séries par tournoi)
```

- /public : fichiers front-end (HTML, CSS, JS).
- server.js : point d'entrée du serveur Node.js.
- db.js : configuration de la base de données Sequelize.

/models (la couche Modèle du pattern MCV)

Ce dossier contient les modèles Sequelize, qui représentent la structure des données en base.

Chaque fichier de ce dossier correspond à une table dans la base de données MariaDB, et définit les attributs, types de données, et relations entre entités.

- index.js : centralise l'import et la configuration de tous les modèles Sequelize.
- Joueur.js : définit les attributs du joueur (licence, nom, prénom, club, points, etc.).
- Serie.js : représente les séries proposées dans un tournoi (nom, catégorie, limite de points...).
- Tournoi.js : contient les données relatives à chaque tournoi (nom, date, lieu).
- Inscription.js : enregistre les inscriptions des joueurs dans les séries des tournois.
- SerieIn.js : assure la liaison entre les séries et les tournois.

/routes (la couche Contrôleur du pattern MCV)

Ce dossier regroupe l'ensemble des routes API REST de l'application. Chaque fichier gère les endpoints relatifs à une entité précise.

Les routes sont définies à l'aide du framework Express.js et permettent de manipuler les données via des méthodes HTTP (GET, POST, etc.).

- connexion.js : gère la connexion des utilisateurs (vérification du mail et mot de passe).
- inscrits.js : gère la consultation et l'inscription des joueurs dans les séries.
- series.js : récupère les séries proposées pour chaque tournoi.
- tournois.js : permet la création et l'affichage des tournois.

/public (la couche Vue du pattern MCV)

Ce dossier contient l'ensemble des fichiers front-end, accessibles directement par les utilisateurs.

Il est organisé en sous-dossiers pour séparer le HTML, le CSS et le JavaScript.

- Accueil.html : page d'accueil affichant la liste des tournois.
- Connexion.html : page permettant aux utilisateurs de se connecter.
- Formulaire.html : formulaire d'inscription à un tournoi et ses séries.
- Inscrits.html : liste des inscrits d'un tournoi donné.
- Ajout_Tournoi.html : interface de création de nouveaux tournois (réservée aux organisateurs).
- /css/ : contient les fichiers de styles CSS (design, thème sombre).
- /js/ : contient les scripts JavaScript assurant la dynamique des pages, la gestion des événements, les appels API et le contrôle des données utilisateur.

server.js

Ce fichier est le point d'entrée principal de l'application côté serveur. Il configure et lance le serveur HTTP avec Express.js.

Fonctions principales :

- Initialisation d'Express.
- Mise en place des middlewares : cors, express.json(), express.static() pour servir les fichiers statiques.
- Importation et utilisation des routes API.
- Gestion des erreurs 404 et des réponses génériques.
- Synchronisation avec la base de données via Sequelize.
- Lancement du serveur sur le port défini dans l'environnement.

db.js

Ce fichier contient la configuration de la connexion à la base de données MariaDB avec Sequelize. Il :

- Récupère les variables d'environnement (nom de la BDD, utilisateur, mot de passe).
- Initialise l'instance Sequelize.
- Tente de se connecter à la base de données.
- Exporte l'instance Sequelize pour la réutiliser dans l'application.

3.3. Architecture logique

L'architecture suit un modèle MVC simplifié, garantissant séparation des responsabilités, modularité et facilité de maintenance.

Modèle (Model)

La gestion des données et des relations avec la base MariaDB est assurée par Sequelize, dans le dossier /models. Chaque modèle représente une entité (Joueur, Tournoi, etc.) avec ses relations, ses règles de validation et ses comportements.

Les requêtes SQL sont générées automatiquement par l'ORM.

Vue (View)

Le front-end, contenu dans /public, fournit l'interface graphique pour les utilisateurs :

- Les pages HTML forment la structure visuelle.
- Les fichiers CSS assurent le design responsive, avec une charte graphique sombre.

- Les fichiers JS côté client rendent les pages dynamiques, en réalisant des appels vers l'API et en gérant les interactions utilisateurs (filtres, tri, formulaire d'inscription...).

Contrôleur (Controller)

Les routes dans /routes sont responsables de :

- La logique applicative.
- La validation des données avant enregistrement.
- L'appel aux modèles Sequelize pour récupérer ou manipuler les données.
- Le traitement des erreurs et le retour d'un résultat clair au front-end (en JSON).

Flux de données dans l'application (schéma simplifié)

- L'utilisateur agit sur l'interface (par exemple : clique sur "S'inscrire").
- Une requête est envoyée via JavaScript (Fetch API) à l'API REST sur le serveur Express.js.
- La route concernée vérifie les données, interagit avec le modèle Sequelize, puis traite la logique métier.
- Le modèle accède à MariaDB et retourne le résultat au contrôleur.
- La réponse JSON est envoyée au front-end pour mettre à jour l'affichage.

Synthèse

L'architecture MVC simplifiée permet de :

- Séparer proprement le traitement des données, l'affichage et la logique métier.
- Faciliter la maintenance et l'ajout de nouvelles fonctionnalités.
- Garantir une application modulaire, lisible et évolutive.

4. Fonctionnalités réalisées

Le projet de gestion de tournois de tennis de table s'articule autour de deux profils d'utilisateurs : les joueurs et les organisateurs. Chaque rôle dispose de fonctionnalités dédiées qui répondent à ses besoins spécifiques.

Ce chapitre détaille les fonctionnalités majeures qui ont été développées pour rendre l'application intuitive, fonctionnelle et performante.

4.1. Fonctionnalités pour les joueurs

Les joueurs peuvent consulter les tournois, s'y inscrire et gérer leurs participations à travers une interface simple et ergonomique.

Inscription à un tournoi

Les joueurs ont accès à une liste de tournois disponibles sur la page d'accueil.

En cliquant sur un tournoi, ils accèdent à un formulaire d'inscription dynamique, prérempli avec leurs informations personnelles stockées dans le localStorage (nom, prénom, licence, points...).



Ce formulaire est accessible uniquement aux joueurs connectés. Si un utilisateur n'est pas connecté, il est redirigé vers la page de connexion.

Sélection des séries

Les joueurs peuvent choisir jusqu'à trois séries différentes pour un même tournoi.

Des contrôles automatiques sont réalisés avant de valider l'inscription :

- Vérification du nombre maximum d'inscriptions à 3 séries.
- Limite de points : le nombre de points du joueur ne doit pas dépasser la limite définie pour la série.

	Projet	
	COMPTE RENDU	

- Sexe autorisé : certaines séries peuvent être réservées à un sexe spécifique (homme/femme).
- Catégorie d'âge : si une catégorie d'âge est précisée dans la série, le joueur doit appartenir à cette catégorie.
- Vérification de la disponibilité des places dans la série (en fonction de la capacité maximale de participants).

Filtrage des inscrits

Les joueurs peuvent consulter la liste des inscrits d'un tournoi donné.

Un système de filtres avancés est mis en place pour faciliter la recherche :

- Filtrer par série : affiche uniquement les joueurs inscrits à une série donnée.
- Filtrer par nom : recherche d'un joueur par son nom.
- Filtrer par nombre de points : recherche des joueurs dont le nombre de points est supérieur ou égal à une valeur définie.

La liste est dynamique et s'actualise à chaque modification des filtres.

Connexion / Authentification

Le joueur peut se connecter avec son adresse e-mail et son mot de passe.

Après authentification réussie, les informations de l'utilisateur sont stockées dans le localStorage :

- Nom, prénom, licence, club, points, téléphone, e-mail.
- Rôle de l'utilisateur : joueur ou organisateur.

L'interface est adaptée en fonction du rôle :

- Les joueurs n'ont pas accès aux fonctionnalités réservées aux organisateurs.
- Le bouton "Ajouter un tournoi" est désactivé pour eux.

4.2. Fonctionnalités pour les organisateurs

Les organisateurs disposent de privilèges supplémentaires pour gérer les tournois et contrôler les inscriptions.

Création d'un tournoi

Accès à la page "Créer un nouveau tournoi" réservé uniquement aux utilisateurs ayant le rôle d'organisateur.

Le formulaire de création de tournoi comprend :

- Le nom du tournoi.
- La date de début et la date de fin.
- Le lieu du tournoi.

Sélection des séries associées au tournoi :

- L'organisateur choisit des séries dans une liste prédéfinie de séries disponibles en base de données.
- Possibilité de sélectionner plusieurs séries pour le même tournoi.
- Les séries peuvent être ajoutées et retirées dynamiquement avant validation.

Ajout et gestion des séries

- Les séries proposées à l'inscription sont issues d'une liste centralisée et validée par l'administrateur.
- Chaque série contient des règles spécifiques (limite de points, sexe autorisé, catégorie d'âge, nombre de places).
- L'organisateur peut s'assurer qu'aucune série n'est surchargée en surveillant les inscriptions en temps réel.

Visualisation des inscrits

Possibilité de voir la liste complète des inscrits à un tournoi.

L'affichage est regroupé par joueur :

- Affiche les informations personnelles de chaque joueur.

- Affiche la liste des séries auxquelles le joueur est inscrit.

Cette vue permet à l'organisateur de contrôler le respect des règles d'inscription et la bonne répartition des joueurs.

Accès restreint aux fonctionnalités avancées

Seuls les organisateurs peuvent :

- Accéder à la création de tournoi.
- Visualiser et gérer les séries d'un tournoi.
- Le bouton de création de tournoi est masqué ou désactivé pour les simples joueurs.

4.3. Fonctionnalités générales

Certaines fonctionnalités sont communes à tous les utilisateurs afin de garantir une expérience fluide et intuitive.

Affichage dynamique des données

Les listes de tournois sont affichées de façon dynamique :

- Les données sont récupérées depuis l'API REST via des requêtes HTTP (Fetch API).
- Les tournois sont triables selon différents critères :
 - Numéro d'ID.
 - Nom.
 - Lieu.
 - Date de début.

La table HTML est mise à jour immédiatement selon le tri choisi.

Navigation fluide entre les pages

L'utilisateur peut naviguer facilement entre :

- Accueil : affiche les tournois disponibles.
- Inscrits : affiche la liste des joueurs inscrits à un tournoi spécifique.
- Formulaire d'inscription : pour s'inscrire à un tournoi.
- Connexion : pour accéder à son compte et gérer ses inscriptions.
- Création de tournoi (uniquement pour les organisateurs).

Système de filtres avancés

Des filtres sur les inscrits permettent de faciliter la recherche et la consultation :

- Les filtres sont combinables (par série, par nom, par points).
- Ils permettent de restreindre l'affichage à une sous-population de joueurs ciblés.

Ces filtres sont particulièrement utiles pour :

- Les organisateurs qui souhaitent contrôler les inscriptions.
- Les joueurs qui cherchent à savoir qui sont leurs concurrents dans une série.

Synthèse des fonctionnalités

Le projet a permis de développer une application web complète, respectant les besoins des joueurs et des organisateurs, grâce à une interface intuitive et des contrôles robustes.

Chaque fonctionnalité a été pensée pour :

- Simplifier les inscriptions et la gestion des tournois.
- Garantir le respect des règles propres aux tournois de tennis de table.
- Offrir une expérience utilisateur fluide, grâce à l'utilisation combinée de HTML5, CSS3, JavaScript, Node.js, Express, MariaDB et Sequelize.

5. Développement du back-end

Cette partie présente la mise en place et le développement de la partie serveur de l'application, réalisée en Node.js avec le framework Express.js. L'API REST permet la gestion complète des tournois, séries, joueurs et inscriptions. La logique métier et les règles d'inscription sont intégrées directement dans le back-end, garantissant la cohérence et la sécurité des données.

5.1. Mise en place du serveur Node.js avec Express

Configuration du serveur (server.js)

Le point d'entrée de l'application est le fichier server.js. C'est ici que sont initialisés le serveur Node.js et les routes d'API.

- Le serveur utilise Express.js comme framework pour la création d'un serveur HTTP léger et performant.
- Le fichier importe également la configuration de la base de données et synchronise les modèles avec celle-ci.

```
const express = require('express'); // Framework pour créer le serveur HTTP
const cors = require('cors'); // Middleware pour autoriser les requêtes cross-origin
const path = require('path'); // Module pour gérer les chemins de fichiers
const { sequelize } = require('./models'); // Import de l'instance Sequelize connectée à la base

const app = express(); // Création de l'application Express
```

Mise en place des middlewares

Les middlewares Express permettent de gérer les requêtes entrantes et sortantes :

- `express.json()` : ce middleware est utilisé pour parser le corps des requêtes HTTP au format JSON. Cela permet de récupérer et manipuler facilement les données envoyées par le client.
- `cors()` : ce middleware active la politique de Cross-Origin Resource Sharing, indispensable pour accepter les requêtes provenant d'un autre domaine (exemple : une interface front-end sur un autre port lors du développement).

```
// Active CORS pour permettre les requêtes depuis d'autres origines
app.use(cors());

// Permet de lire les données JSON envoyées dans le corps des requêtes
app.use(express.json());
```

Autres fonctionnalités :

- Le serveur expose également des fichiers statiques via la méthode `express.static()` (accès aux fichiers HTML, CSS, JS dans le dossier public).
- Gestion d'erreurs 404 et routes non trouvées :

```
// Gestion des routes non trouvées : retourne une erreur 404 en JSON
app.use((req, res) => {
  res.status(404).json({ message: "Page non trouvée" });
});
```

5.2. Création de l'API REST

Une API RESTful a été créée afin de fournir un ensemble de points d'accès (endpoints) permettant d'interagir avec les ressources. Chaque ressource est gérée à travers des routes Express regroupées par fonctionnalité.

Endpoints principaux développés

- GET `/api/tournois` :
 - Récupération de la liste de tous les tournois présents en base de données.

- POST /api/tournois :
 - Création d'un nouveau tournoi. Cette route est sécurisée : seuls les utilisateurs ayant le rôle d'organisateur peuvent l'utiliser.
 - Elle permet aussi d'associer les séries sélectionnées à ce tournoi.
- GET /api/series/:id_tournoi :
 - Retourne les séries spécifiques à un tournoi donné, en fonction de son id_tournoi.
- POST /api/inscrits :
 - Inscription d'un joueur à un ou plusieurs séries dans un tournoi donné.
 - Cette route vérifie automatiquement toutes les règles de participation (points, sexe, âge, places disponibles).
- POST /api/login :
 - Authentification d'un joueur ou organisateur à partir de son email et de son mot de passe.
 - Si les informations sont valides, les informations du joueur sont retournées (y compris le rôle est_organisateur).

5.3. Gestion des inscriptions

Limitation à trois séries maximums par joueur

Lors de l'inscription à un tournoi, le back-end vérifie qu'un joueur n'est pas déjà inscrit à plus de trois séries pour ce même tournoi :

- La vérification est réalisée à partir des inscriptions existantes en base de données.
- Si un joueur tente de s'inscrire à une série supplémentaire alors qu'il a déjà atteint la limite, un message d'erreur clair est renvoyé.

```
// Vérifie si le joueur dépasse le nombre max d'inscriptions (3 séries par tournoi)
const nbSeriesExistantes = inscriptionsExistantes.length;
if (nbSeriesExistantes + series.length > 3) {
  return res.status(400).json({ message: "Vous ne pouvez pas vous inscrire à plus de 3 séries par tournoi." });
}
```

Vérification des règles métier sur les séries

Avant chaque inscription, le back-end contrôle le respect de plusieurs règles métier :

- Points : le nombre de points du joueur ne doit pas dépasser la limite de points définie pour la série.
- Sexe autorisé : certaines séries sont réservées aux hommes ou aux femmes.
- Catégorie d'âge : certaines séries exigent une catégorie d'âge spécifique.
- Nombre maximum de participants : si une série est complète, l'inscription est refusée.

```
// Vérifie le sexe autorisé de la série
if (serie.sexe_autorise === 'Femme' && joueur.sexe !== 'Femme') {
  return res.status(400).json({ message: `La série "${serie.nom_serie}" est réservée aux femmes.` });
}

if (serie.sexe_autorise === 'Homme' && joueur.sexe !== 'Homme') {
  return res.status(400).json({ message: `La série "${serie.nom_serie}" est réservée aux hommes.` });
}

// Vérifie la catégorie d'âge si définie
if (serie.categorie_age && joueur.categorie_age !== serie.categorie_age) {
  return res.status(400).json({ message: `Cette série est réservée à la catégorie ${serie.categorie_age}.` });
}

// Vérifie la limite de points si définie
if (serie.limite_points !== null && joueur.point_joueur > serie.limite_points) {
  return res.status(400).json({
    message: `Votre nombre de points (${joueur.point_joueur}) dépasse la limite autorisée de ${serie.limite_points}`
  });
}
```

Gestion des erreurs et réponses claires

Le serveur retourne des codes d'état HTTP appropriés :

- 400 Bad Request : erreurs côté client, par exemple si les règles d'inscription ne sont pas respectées.

```
// Vérifie que les informations sont complètes
if (!id_tournoi || !licence_joueur || !series || series.length === 0) {
  return res.status(400).json({ message: "Tous les champs sont requis." });
}
```

- 404 Not Found : ressources non trouvées, par exemple un tournoi ou une série inexistante.

```
// Si aucune inscription trouvée, retourne un message
if (!inscriptions || inscriptions.length === 0) {
  return res.status(404).json({ message: "Aucun joueur inscrit pour ce tournoi." });
}
```

- 500 Internal Server Error : erreur inattendue sur le serveur.

```
} catch (err) {
  // En cas d'erreur serveur
  res.status(500).json({ message: "Erreur serveur", error: err.message });
}
});
```

5.4. Connexion base de données

Le projet utilise MariaDB comme système de gestion de base de données relationnelle.

Synchronisation avec Sequelize

L'ORM Sequelize est utilisé pour :

- Définir les modèles de données (Joueur, Tournoi, Serie, Inscription...).
- Gérer les relations entre les tables via des clés primaires et étrangères.
- Effectuer la synchronisation des modèles Sequelize avec la base MariaDB.

```
// Synchronisation de la base de données avec les modèles Sequelize
sequelize.sync()
  .then(() => console.log("Base de données synchronisée avec Sequelize"))
  .catch(err => console.error("Erreur de synchronisation :", err));
```

Gestion des relations entre les tables

Les relations suivantes sont mises en place dans Sequelize :

- Un Joueur peut avoir plusieurs Inscriptions.
- Un Tournoi est lié à plusieurs Séries via SerieIn.
- Une Serie est associée à plusieurs Inscriptions.

```
// Définition des relations entre les modèles

// Un joueur peut participer à plusieurs tournois via les inscriptions
Joueur.belongsToMany(Tournoi, { through: Inscription, foreignKey: 'licence_joueur' });
Tournoi.belongsToMany(Joueur, { through: Inscription, foreignKey: 'id_tournoi' });

// Une série peut exister dans plusieurs tournois via SerieIn
Serie.belongsToMany(Tournoi, { through: SerieIn, foreignKey: 'id_serie' });
Tournoi.belongsToMany(Serie, { through: SerieIn, foreignKey: 'id_tournoi' });

// Relations directes sur SerieIn (liaison entre Série et Tournoi)
SerieIn.belongsTo(Serie, { foreignKey: 'id_serie' });
SerieIn.belongsTo(Tournoi, { foreignKey: 'id_tournoi' });
Serie.hasMany(SerieIn, { foreignKey: 'id_serie' });
Tournoi.hasMany(SerieIn, { foreignKey: 'id_tournoi' });
```

Contraintes d'intégrité

- Les clés étrangères permettent de garantir l'intégrité des données.
- Une inscription ne peut exister que si le Joueur, le Tournoi et la Série référencés existent en base.

Synthèse de la partie back-end

Le back-end assure :

- La gestion complète des règles métiers pour l'inscription aux tournois.
- La sécurité des opérations en garantissant que seules les actions valides sont acceptées.
- Une communication fluide avec le front-end grâce à une API REST claire et cohérente.

Ce back-end robuste permet une gestion centralisée des tournois et des participants, avec une structure scalable et facilement maintenable.

6. Développement du front-end

Le développement du front-end a été réalisé en HTML5, CSS3 (avec un thème sombre moderne) et JavaScript Vanilla (sans frameworks externes). Cette partie décrit en détail la conception des différentes pages, le design visuel ainsi que les interactions utilisateurs rendues possibles par JavaScript.

6.1. Pages HTML

Accueil.html

C'est la page d'accueil de l'application, accessible à tous les utilisateurs.

Elle affiche la liste de tous les tournois existants sous forme de tableau.

Un bouton "Ajouter un tournoi" est présent et activé uniquement pour les utilisateurs organisateurs (grâce à la vérification de leur rôle depuis localStorage).

Chaque ligne de tournoi dans le tableau est cliquable et mène au formulaire d'inscription.

Un lien "Voir les inscrits" est également disponible sur chaque tournoi.

Connexion.html

Page d'authentification permettant aux joueurs et organisateurs de se connecter.

Le formulaire de connexion demande un email et un mot de passe.

Une fois connecté, l'utilisateur est redirigé vers l'Accueil. Les données du compte (nom, prénom, rôle, etc.) sont stockées dans localStorage pour gérer l'accès aux fonctionnalités.

Formulaire.html

Page affichant le formulaire d'inscription à un tournoi, prérempli avec les données du joueur connecté (nom, prénom, licence, club, etc.).

Les séries disponibles pour le tournoi sélectionné sont affichées sous forme de cases à cocher.

L'utilisateur peut sélectionner jusqu'à trois séries, en respectant les règles d'accès (nombre de points, sexe, catégorie d'âge, etc.).

La validation du formulaire effectue un appel POST vers l'API des inscriptions.

Inscrits.html

Affiche la liste des joueurs inscrits pour un tournoi donné.

Comprend des filtres dynamiques :

- Par nom (champ de recherche texte).
- Par série (liste déroulante).
- Par nombre de points (champ numérique).

Cette page permet aux utilisateurs de consulter les inscrits, triés ou filtrés en fonction de leurs critères.

Ajout_Tournoi.html

Page permettant à un organisateur de créer un nouveau tournoi.

Formulaire de création :

- Nom du tournoi.
- Date de début et date de fin.
- Lieu.

Une fois le tournoi créé, l'organisateur peut associer des séries prédéfinies via un sélecteur.

Cette page utilise des boutons dynamiques pour ajouter et supprimer les séries associées à un tournoi.

6.2. CSS : design thème sombre

Le design du site repose sur un thème sombre, offrant une expérience utilisateur agréable et moderne. Le choix des couleurs, des contrastes et des effets visuels assure une bonne lisibilité et une navigation fluide.

Principaux choix de design

- Fond principal : #121212 (gris très foncé, presque noir).
- Texte principal : #f1f1f1 (blanc légèrement cassé, doux pour les yeux).
- Composants clairs sur fond sombre, assurant une bonne contraste et lisibilité.

Mise en page et ergonomie

- Utilisation de flexbox pour gérer la disposition des éléments sur la page.
- Design responsive, adapté aux différentes tailles d'écran (PC, tablette).
- Les tableaux sont centrés et fluides, avec un hover pour indiquer l'interactivité.
- Les filtres et sélecteurs sont accessibles et bien espacés.

Boutons

Boutons aux couleurs harmonieuses :

- Bleu clair (#2196f3) pour les actions principales (connexion, tri, retour).
- Vert (#4CAF50) pour la validation (inscriptions, création tournoi).

- Rouge (#e53935) pour les actions de suppression (ex : suppression d'une série).

Effets de hover sur tous les boutons :

- Changement de couleur d'arrière-plan.
- Effet de translation léger vers le haut (`translateY(-2px)`) pour renforcer le côté interactif.

Transitions fluides pour les interactions utilisateur :

- Animation sur le hover.
- Changements de couleurs et d'ombre portées.

6.3. JavaScript : comportement dynamique

L'application front-end utilise JavaScript Vanilla pour assurer la logique côté client et l'interaction avec l'API REST côté serveur.

Requêtes HTTP avec fetch()

Récupération des tournois à afficher dans Accueil.html via un GET sur `/api/tournois`.

Récupération des séries spécifiques à un tournoi lors de l'affichage du formulaire d'inscription ou des inscrits.

POST des données :

- Lors de l'inscription à un tournoi (`/api/inscrits`).
- Lors de la création d'un tournoi (`/api/tournois`).
- Lors de la connexion (`/api/login`).

Manipulation du DOM

Création dynamique de tableaux HTML à partir des données JSON obtenues via les API.

Affichage dynamique des listes déroulantes, cases à cocher, et tableaux d'inscrits.

Mise à jour des contenus sans recharger la page, pour une navigation fluide et réactive.

Gestion de l'état utilisateur avec localStorage

Lorsqu'un utilisateur se connecte, ses informations (nom, prénom, rôle, licence, etc.) sont stockées en localStorage. Le rôle (`est_organisateur`) permet de débloquent certaines actions côté front (comme l'accès au bouton "Créer un tournoi").

En cas de déconnexion, le localStorage est vidé, et les boutons sensibles sont désactivés.

Gestion des événements

Click sur les boutons :

- Déclenche la redirection vers la bonne page (exemple : vers `Formulaire.html?id=1`).
- Déclenche la suppression d'une série dans le formulaire d'ajout de tournoi.

Submit sur les formulaires :

- Valide les champs avant d'envoyer les données à l'API.
- Empêche l'action par défaut du navigateur pour envoyer manuellement la requête avec `fetch()`.

Exemples de vérifications et conditions dynamiques

Affichage du bouton de création de tournoi :

- Activé uniquement si `localStorage.getItem("est_organisateur") === "true"`.

Contrôle de sélection des séries lors de l'inscription :

- Limitation à 3 séries maximum.
- Vérification des règles d'accès en amont.

Conclusion du développement front-end

L'interface utilisateur a été conçue pour offrir simplicité, efficacité et clarté. Le design sombre améliore le confort de lecture et donne une identité moderne au site. Grâce à l'utilisation de JavaScript Vanilla, le projet reste léger et performant, sans dépendre de frameworks tiers, tout en proposant une expérience fluide et interactive.

7. Gestion des rôles et sécurité

7.1. Rôles utilisateurs

Le projet distingue deux types d'utilisateurs, chacun ayant des droits spécifiques au sein de l'application.

Joueur

Le joueur est l'utilisateur de base de la plateforme. Il peut :

- Consulter la liste des tournois ouverts.
- S'inscrire à un tournoi en sélectionnant jusqu'à trois séries au maximum, selon les règles d'éligibilité (limite de points, sexe, catégorie d'âge).
- Visualiser la liste des inscrits d'un tournoi.
- Se connecter à son espace personnel (les informations du joueur sont stockées temporairement dans localStorage après connexion).

Le joueur n'a pas accès aux fonctionnalités d'administration ou de gestion de tournoi.

Organisateur

L'organisateur dispose de droits étendus pour administrer les tournois. Il peut :

- Accéder à l'interface de création de tournoi.
- Associer des séries prédéfinies à un tournoi lors de sa création.
- Consulter les joueurs inscrits sur ses tournois.
- Le bouton de création de tournoi n'est affiché que si l'utilisateur connecté est reconnu comme organisateur, grâce à la variable `est_organisateur` stockée en localStorage après connexion.

Ce rôle permet de contrôler l'accès aux fonctions critiques pour la gestion des événements.

Accès différencié aux fonctionnalités

Les fonctionnalités sensibles (comme la création de tournoi) sont restreintes aux organisateurs :

- Le bouton "Ajouter un tournoi" est désactivé par défaut et activé uniquement pour les organisateurs via un test dans le code front-end :

```
function gererBoutonCreation() {
  const estOrganisateur = localStorage.getItem("est_organisateur"); // On récupère l'info du stockage local
  const boutonCreation = document.getElementById("btn-creation"); // Le bouton "Ajouter un tournoi"

  // Si l'utilisateur est organisateur (valeur "true" sous forme de string), le bouton est activé
  if (estOrganisateur === "true") {
    boutonCreation.disabled = false;
  } else {
    boutonCreation.disabled = true;
  }
}
```

- La navigation conditionnelle et l'affichage de certains boutons sont gérés côté client à partir du localStorage.
- Côté serveur, les API REST contrôlent également l'accès à certaines actions par l'intermédiaire des vérifications de données (par exemple, seul un utilisateur avec le rôle d'organisateur peut envoyer certaines requêtes POST pour créer un tournoi).

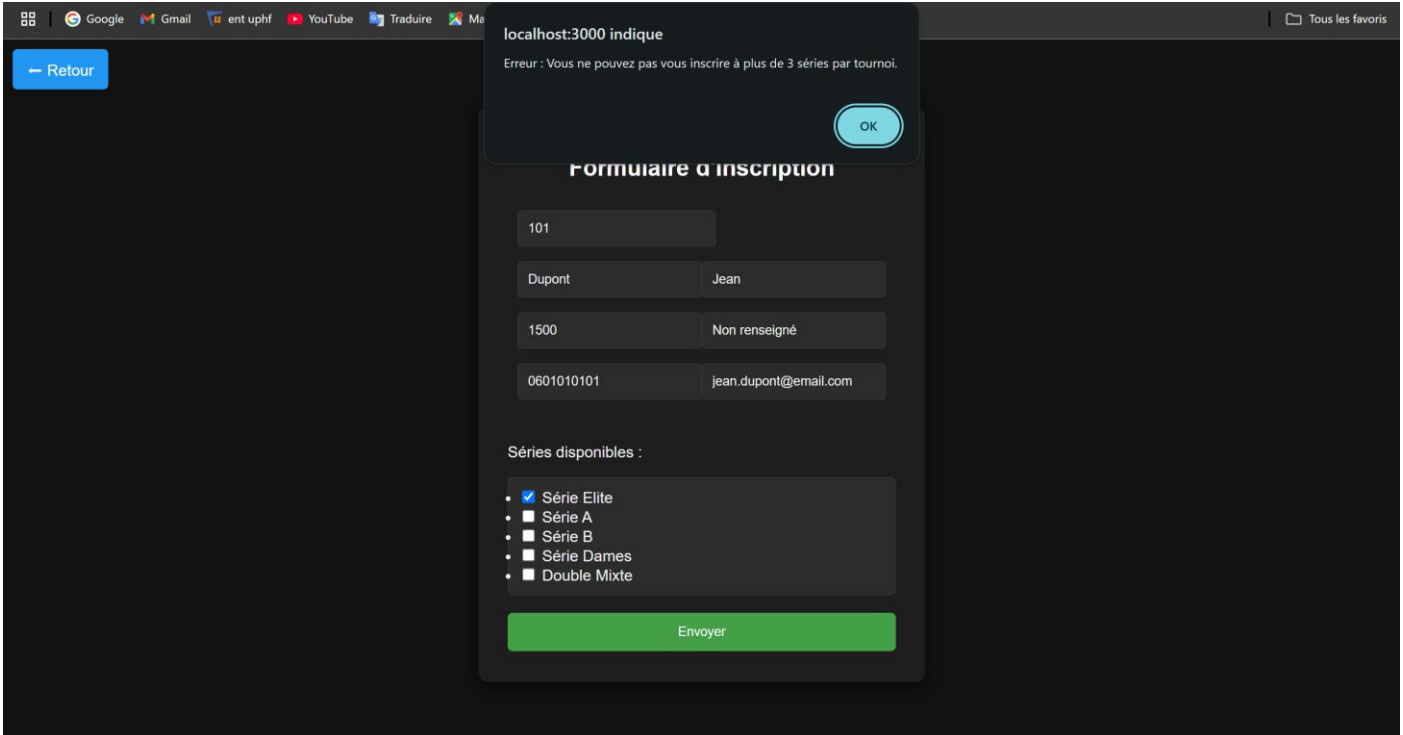
7.2. Sécurité minimale

Validation des formulaires (front-end et back-end)

Côté front-end, les formulaires utilisent les attributs HTML5 (required, type="email", type="number", etc.) pour vérifier la validité des données saisies.

Des vérifications en JavaScript sont également effectuées pour :

- Limiter le nombre de séries sélectionnées (3 maximum).



The screenshot shows a web browser window with a dark theme. At the top, there's a navigation bar with a "Retour" button. Below it, a modal dialog box is displayed with the title "localhost:3000 indique" and the message "Erreur : Vous ne pouvez pas vous inscrire à plus de 3 séries par tournoi." with an "OK" button. The main content is a "Formulaire d'inscription" (Registration Form) with the following fields:

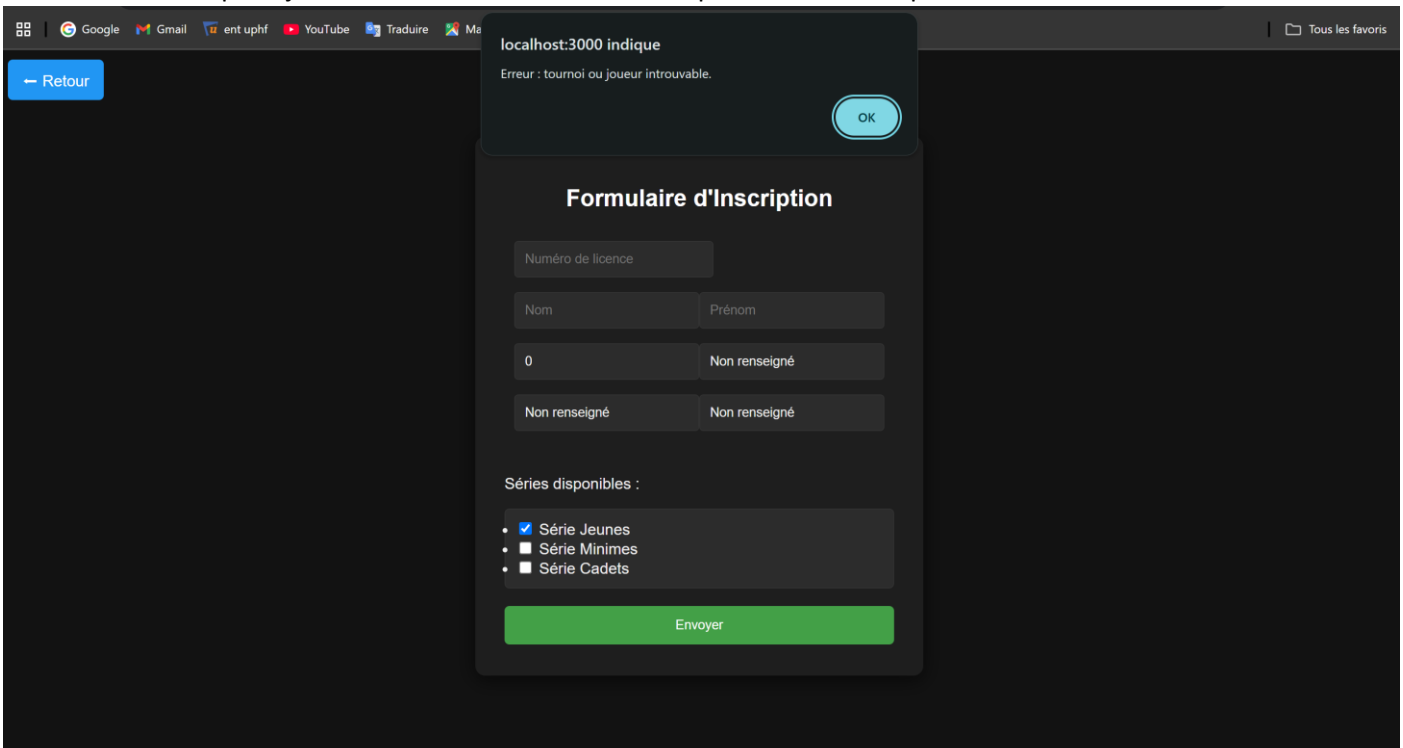
- 101 (input field)
- Dupont (input field)
- Jean (input field)
- 1500 (input field)
- Non renseigné (input field)
- 0601010101 (input field)
- jean.dupont@email.com (input field)

Below the form, there's a section "Séries disponibles :" with a list of series:

- ☒ Série Elite
- ☐ Série A
- ☐ Série B
- ☐ Série Dames
- ☐ Double Mixte

At the bottom of the form is a green "Envoyer" button.

- S'assurer qu'un joueur est bien connecté avant de procéder à l'inscription.



The screenshot shows a web browser window with a dark theme. At the top, there's a navigation bar with a "Retour" button. Below it, a modal dialog box is displayed with the title "localhost:3000 indique" and the message "Erreur : tournoi ou joueur introuvable." with an "OK" button. The main content is a "Formulaire d'Inscription" (Registration Form) with the following fields:

- Numéro de licence (input field)
- Nom (input field)
- Prénom (input field)
- 0 (input field)
- Non renseigné (input field)
- Non renseigné (input field)
- Non renseigné (input field)

Below the form, there's a section "Séries disponibles :" with a list of series:

- ☒ Série Jeunes
- ☐ Série Minimes
- ☐ Série Cadets

At the bottom of the form is a green "Envoyer" button.

- Afficher des messages d'erreur clairs en cas de problème (exemple : "Veuillez sélectionner au moins une série").

Côté back-end, l'API vérifie de nouveau toutes les données reçues avant de les enregistrer en base de données :

- Validation de la cohérence des données (exemple : vérifier que le joueur n'est pas déjà inscrit à cette série, qu'il respecte les limites de points, etc.).
- Rejets des requêtes mal formées avec des codes HTTP adaptés (400 Bad Request, 404 Not Found, etc.).

Vérification des rôles côté client

Le rôle (est_organisateur) est stocké dans le localStorage après une connexion réussie.

Ce rôle est consulté sur chaque page où un contrôle d'accès est nécessaire (comme l'affichage du bouton d'ajout de tournoi).

Exemple de contrôle dans le JavaScript :

```
function gererBoutonCreation() {
  const estOrganisateur = localStorage.getItem("est_organisateur"); // On récupère l'info du stockage local
  const boutonCreation = document.getElementById("btn-creation"); // Le bouton "Ajouter un tournoi"

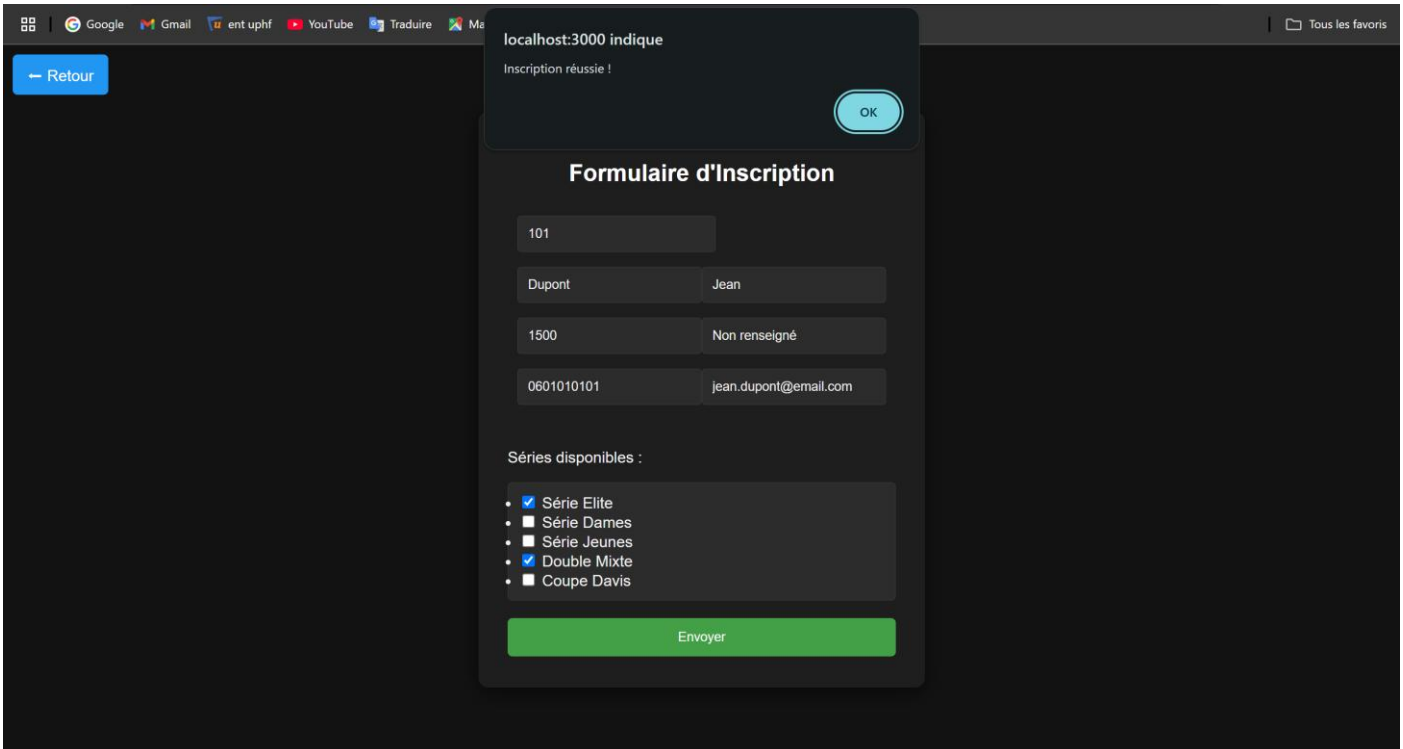
  // Si l'utilisateur est organisateur (valeur "true" sous forme de string), le bouton est activé
  if (estOrganisateur === "true") {
    boutonCreation.disabled = false;
  } else {
    boutonCreation.disabled = true;
  }
}
```

Cette vérification est uniquement indicative côté client : dans une application de production, il est recommandé d'effectuer un contrôle strict côté serveur (middleware d'authentification et de gestion des rôles).

Gestion des erreurs HTTP dans l'API

Les routes Express.js du back-end renvoient des codes HTTP appropriés pour indiquer le statut des requêtes :

- 200 OK pour une réponse réussie.



localhost:3000 indique
Inscription réussie !

OK

Formulaire d'Inscription

101

Dupont Jean

1500 Non renseigné

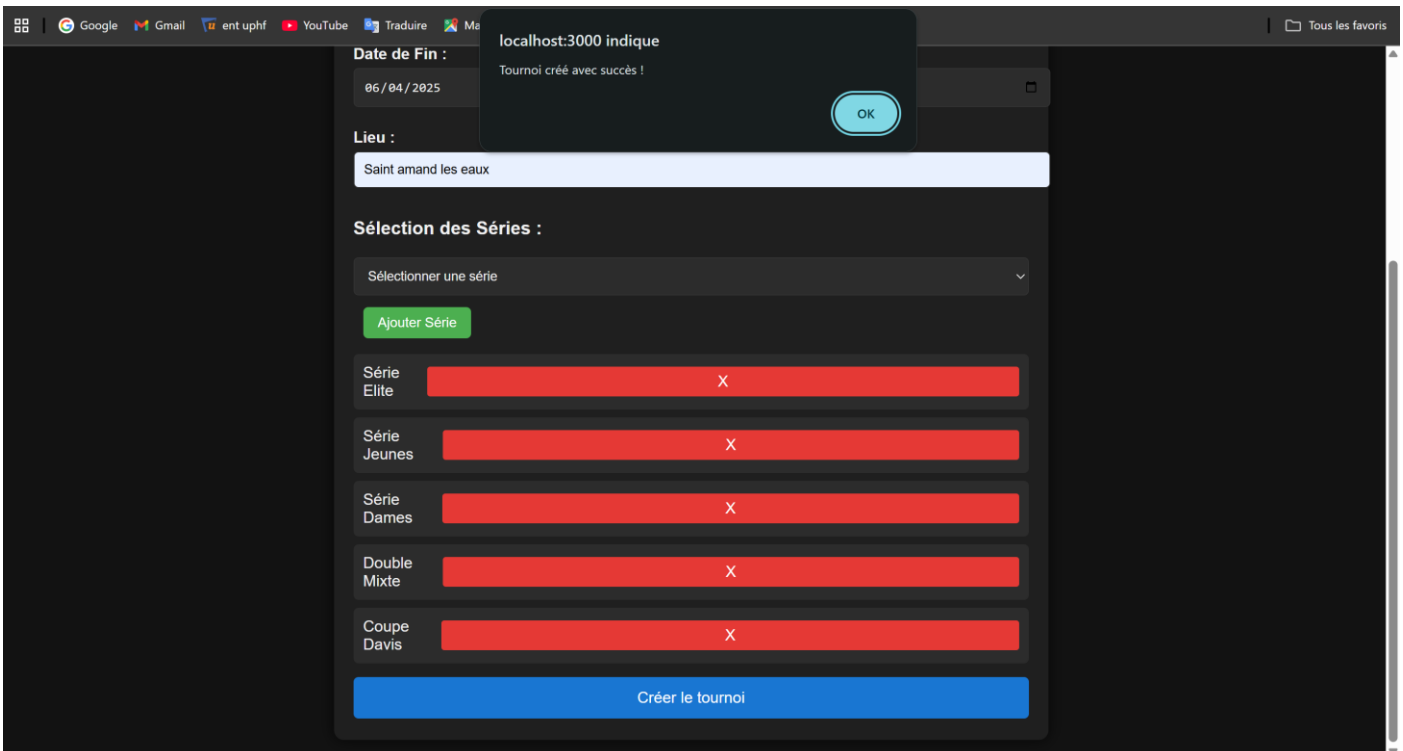
0601010101 jean.dupont@email.com

Séries disponibles :

- ☒ Série Elite
- ☐ Série Dames
- ☐ Série Jeunes
- ☒ Double Mixte
- ☐ Coupe Davis

Envoyer

- 201 Created pour une ressource nouvellement créée (exemple : tournoi ajouté).



localhost:3000 indique
Tournoi créé avec succès !

OK

Date de Fin :
06 / 04 / 2025

Lieu :
Saint amand les eaux

Sélection des Séries :

Sélectionner une série

Ajouter Série

Série Elite	X
Série Jeunes	X
Série Dames	X
Double Mixte	X
Coupe Davis	X

Créer le tournoi

- 400 Bad Request si les données sont incomplètes ou incorrectes.
- 404 Not Found si un joueur, une série ou un tournoi demandé n'existe pas.
- 500 Internal Server Error en cas d'erreur inattendue côté serveur.

Les messages d'erreur renvoyés en JSON sont clairs et informatifs, facilitant la compréhension pour l'utilisateur et le débogage.

Exemple de réponse JSON :

```
// Vérifie si le joueur dépasse le nombre max d'inscriptions (3 séries par tournoi)
const nbSeriesExistantes = inscriptionsExistantes.length;
if (nbSeriesExistantes + series.length > 3) {
  return res.status(400).json({ message: "Vous ne pouvez pas vous inscrire à plus de 3 séries par tournoi." });
}
```

8. Difficultés rencontrées et solutions

Le développement de cette application web de gestion de tournois a présenté plusieurs difficultés techniques et ergonomiques qu'il a fallu identifier et résoudre au fur et à mesure de l'avancement du projet. Ces défis ont été liés à l'apprentissage et à la mise en œuvre de technologies spécifiques, ainsi qu'à la nécessité de proposer une interface utilisateur à la fois fonctionnelle et agréable à utiliser.

8.1. Difficultés techniques

Intégration de Sequelize avec MariaDB

L'intégration de l'ORM Sequelize avec le SGBD MariaDB a posé des problèmes au début du projet. Il a fallu comprendre la configuration précise de la connexion, notamment :

- La gestion des dialectes de bases de données dans Sequelize (MariaDB nécessite une configuration similaire à MySQL, mais certains détails diffèrent).
- La synchronisation des modèles avec la base de données (création automatique des tables, contraintes et relations).

Une des principales difficultés a été de gérer les clés étrangères et les associations entre les tables (Joueur, Tournoi, Serie, Inscription). Sequelize offre plusieurs méthodes (belongsTo, hasMany, belongsToMany), mais leur usage nécessite de bien comprendre les relations directionnelles pour garantir un mapping correct.

Création de l'API REST avec Express.js

La mise en place de l'API REST a représenté un point clé et complexe du projet :

- Il a fallu structurer correctement les routes et les contrôleurs pour garantir une architecture logique et maintenable.
- La gestion des méthodes HTTP (GET, POST) et des paramètres dynamiques dans les URL a demandé une attention particulière, notamment pour sécuriser et valider les requêtes.
- La gestion des erreurs HTTP s'est révélée cruciale : il était nécessaire de renvoyer des codes d'état explicites (400, 404, 500), mais également d'envoyer des messages d'erreur clairs à destination du client (par exemple, lors de l'inscription à une série complète).

Gestion des relations et des jointures

Il a été complexe de réaliser des requêtes imbriquées pour afficher les inscrits d'un tournoi avec les détails de leurs séries.

- Il fallait imbriquer les modèles Sequelize avec include pour récupérer les joueurs, leurs inscriptions et les séries associées.
- L'agrégation des données dans un format lisible par le front-end a nécessité la manipulation de structures complexes, avec groupements par joueur.

Manipulation complexe du DOM et filtres imbriqués

Sur le front-end, la mise en œuvre des filtres dynamiques sur la page des inscrits (filtrage par série, par nom et par points) a nécessité :

- Une manipulation du DOM plus poussée.

- Une synchronisation des états : la modification d'un filtre ne devait pas perturber les autres.
- Une optimisation des performances, notamment pour éviter de parcourir à chaque interaction une liste importante d'inscrits.

8.2. Solutions mises en place

Découpage des scripts JavaScript par page

Afin d'alléger le code et de faciliter sa maintenance, le code JavaScript a été découpé par page :

- script_accueil.js : gestion des tournois, tri et affichage conditionnel des boutons.
- script_inscrits.js : gestion des inscrits et filtres avancés.
- script_formulaire.js : gestion de l'inscription d'un joueur à un tournoi.
- script_connexion.js : gestion de l'authentification.
- script_ajout_tournoi.js : création d'un tournoi avec sélection des séries.

Cette organisation permet une meilleure lisibilité, une modularité accrue et facilite le débogage.

Séparation des fichiers CSS par page, puis regroupement par thème

Dans un premier temps, les fichiers CSS étaient séparés par page pour tester différents designs. Une fois le thème sombre validé, les styles ont été harmonisés pour :

- Respecter une cohérence graphique (couleurs, polices, boutons).
- Assurer un design responsive, notamment sur la page d'accueil et la page de filtrage des inscrits.

Débogage avec console.log() et tests API via Postman

Le debugging a été un élément indispensable pour la correction des erreurs :

- Console.log() a permis de vérifier les états des variables, la réception des données API, ainsi que les actions utilisateurs sur le DOM.
- Postman (ou Thunder Client dans VS Code) a été utilisé pour tester toutes les routes API, simuler des requêtes avec différents scénarios, et s'assurer de la validité des réponses serveur (statuts HTTP, erreurs logiques, validation des champs).

Refonte et clarification de l'API

Face à la complexité de la gestion des inscriptions, l'API a été refondue à plusieurs reprises :

- Séparation claire des responsabilités dans les routes Express.js (connexion, tournois, séries, inscriptions).
- Vérification des règles métier côté back-end, pour s'assurer qu'un joueur ne puisse pas s'inscrire à plus de trois séries ou à des séries qui ne correspondent pas à son profil.

9. Résultat obtenu

9.1. Fonctionnalités terminées

L'ensemble des fonctionnalités essentielles a été mis en place et testé. Voici la liste des fonctionnalités opérationnelles :

Côté Joueur

- ✓ Connexion et authentification via un formulaire sécurisé.
- ✓ Inscription à un tournoi avec sélection des séries disponibles.
- ✓ Validation des critères d'inscription (limite de points, catégorie d'âge, sexe autorisé).
- ✓ Affichage des tournois disponibles avec tri et filtres dynamiques.
- ✓ Consultation des joueurs inscrits à un tournoi avec filtrage par série, nom et nombre de points.

Côté Organisateur

- ✓ Création de tournois via un formulaire détaillé (nom, lieu, dates).
- ✓ Ajout et gestion des séries associées à un tournoi.
- ✓ Affichage dynamique des inscrits et regroupement des séries par joueur.
- ✓ Accès restreint aux fonctionnalités spécifiques (uniquement accessibles aux organisateurs).

Fonctionnalités générales

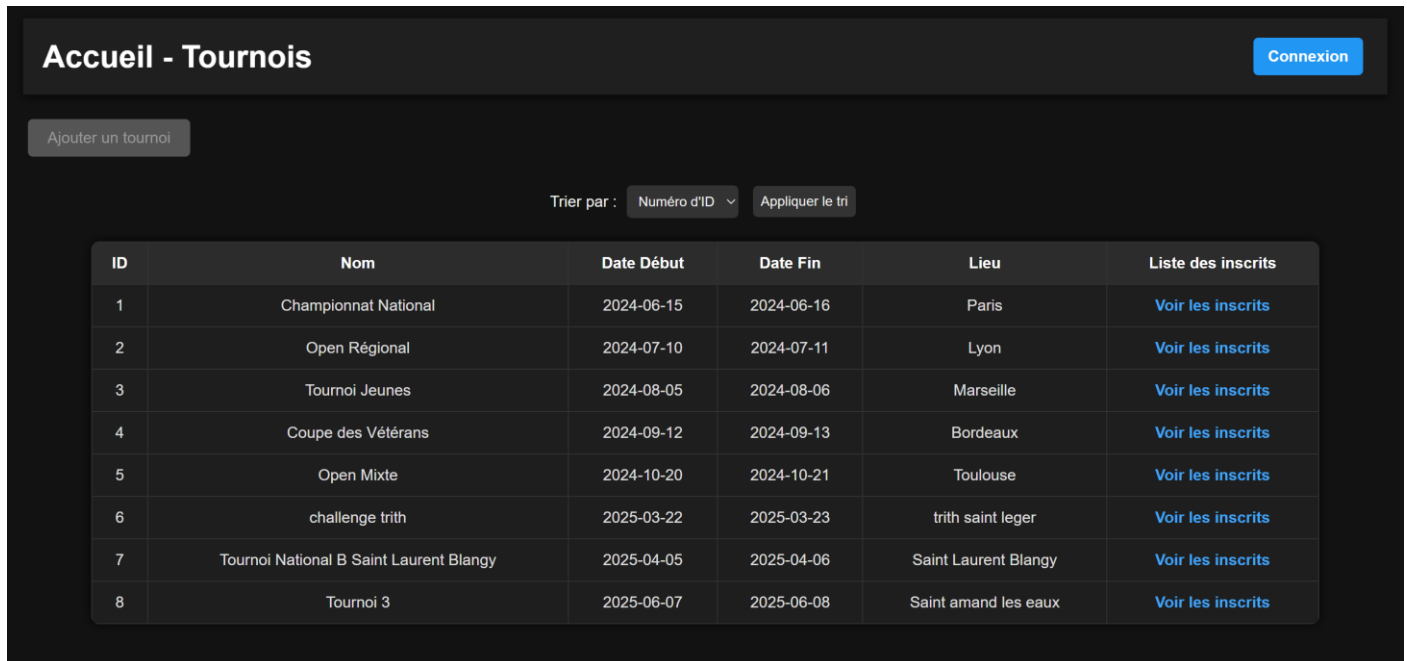
- ✓ Navigation fluide entre les différentes pages du site.
- ✓ Thème sombre pour un design moderne et agréable.
- ✓ Gestion des erreurs et messages clairs en cas d'actions invalides.
- ✓ Stockage des informations utilisateur via localStorage pour une meilleure expérience utilisateur.

9.2. Démonstration

Captures d'écran

Des captures d'écran seront incluses pour illustrer les différentes fonctionnalités :

- Page d'accueil affichant la liste des tournois.



ID	Nom	Date Début	Date Fin	Lieu	Liste des inscrits
1	Championnat National	2024-06-15	2024-06-16	Paris	Voir les inscrits
2	Open Régional	2024-07-10	2024-07-11	Lyon	Voir les inscrits
3	Tournoi Jeunes	2024-08-05	2024-08-06	Marseille	Voir les inscrits
4	Coupe des Vétérans	2024-09-12	2024-09-13	Bordeaux	Voir les inscrits
5	Open Mixte	2024-10-20	2024-10-21	Toulouse	Voir les inscrits
6	challenge trith	2025-03-22	2025-03-23	trith saint leger	Voir les inscrits
7	Tournoi National B Saint Laurent Blangy	2025-04-05	2025-04-06	Saint Laurent Blangy	Voir les inscrits
8	Tournoi 3	2025-06-07	2025-06-08	Saint amand les eaux	Voir les inscrits

- Page de connexion pour les joueurs et organisateurs.

Connexion

E-mail :

Mot de passe :

[Se connecter](#)

Ajouter un tournoi

Jean Dupont est enregistré dans la BDD en tant qu'organisateur donc le bouton est allumé.

- Formulaire d'inscription avec les séries disponibles.

[← Retour](#)

Formulaire d'Inscription

Séries disponibles :

- ☒ Série Elite
- ☒ Série A
- ☒ Série B
- ☒ Série Dames
- ☒ Double Mixte

[Envoyer](#)

- Tableau des inscrits avec filtres dynamiques.

← Retour

Liste des Inscrits

Filter

Filtrer par série :

Série Elite

Rechercher par nom :

Dupont

Rechercher par points :

1400

Appliquer les filtres

LICENCE	NOM	PRÉNOM	CLUB	NOMBRE DE POINTS	SÉRIES INSCRITES
101	Dupont	Jean	TT Club Paris	1500	Série Elite, Série A, Double Mixte

- Formulaire de création de tournoi avec ajout de séries.

← Retour

Créer un Nouveau Tournoi

Nom du Tournoi :

Tournoi Noel

Date de Début :

05 / 04 / 2025

Date de Fin :

06 / 04 / 2025

Lieu :

Saint amand les eaux

Sélection des Séries :

Sélectionner une série

Ajouter Série

Série Elite X

Double Mixte X

Coupe Davis X

Série Jeunes X

Créer le tournoi

Parcours d'un Joueur

- Connexion au site
 - L'utilisateur accède à la page de connexion et entre ses identifiants (e-mail et mot de passe).

- Si les identifiants sont valides, il est redirigé vers la page d'accueil et son statut (joueur ou organisateur) est détecté.
- S'il échoue à se connecter, un message d'erreur apparaît.
- b) Consultation des tournois disponibles
 - Sur la page d'accueil, il visualise la liste des tournois actifs.
 - Il peut trier les tournois par nom, date, lieu ou ID.
 - Il peut rechercher un tournoi spécifique grâce aux filtres.
- c) Inscription à un tournoi
 - L'utilisateur clique sur un tournoi pour afficher le formulaire d'inscription.
 - Il voit automatiquement ses informations personnelles préremplies (nom, prénom, licence, points, club).
 - Il choisit les séries disponibles, avec un maximum de 3 et sous conditions (points, sexe, âge).
 - Après soumission, un message de confirmation s'affiche.
 - Si une série est complète ou si l'utilisateur ne respecte pas les critères, un message d'erreur apparaît.
- d) Consultation des joueurs inscrits
 - Depuis la page des inscrits, l'utilisateur peut voir la liste des joueurs inscrits à un tournoi.
 - Il peut filtrer les inscrits par nom, série ou nombre de points.
 - Les séries d'un joueur sont affichées regroupées sous son nom.
- e) Déconnexion
 - L'utilisateur peut se déconnecter via le bouton prévu.
 - Toutes ses informations stockées en localStorage sont supprimées pour la sécurité.

Parcours d'un Organisateur

- a) Connexion avec un compte organisateur
 - Comme un joueur, l'organisateur se connecte via la page de connexion.
 - Son rôle est détecté automatiquement.
 - Il a accès à des fonctionnalités supplémentaires, notamment la création de tournoi.
- b) Création d'un tournoi
 - Depuis la page d'accueil, il accède à la page de création de tournoi.
 - Il remplit un formulaire avec les informations suivantes :
 - Nom du tournoi
 - Date de début et de fin
 - Lieu
 - Séries associées au tournoi
 - Il sélectionne les séries dans une liste dynamique et peut en ajouter ou en retirer avant validation.
 - Après soumission, le tournoi est enregistré dans la base de données et visible sur la page d'accueil.
- c) Gestion des séries et inscriptions
 - Une fois un tournoi créé, il peut consulter les joueurs inscrits.
 - La liste des inscrits est groupée par joueur, chaque joueur affichant toutes ses séries.
 - Il peut analyser les inscriptions et voir les critères respectés ou non.
- d) Navigation et supervision
 - L'organisateur peut naviguer entre les pages pour superviser les différents tournois.
 - Son rôle lui permet de visualiser toutes les informations mais il ne peut pas modifier directement les inscriptions des joueurs.
- e) Déconnexion
 - Comme les joueurs, il peut se déconnecter, ce qui supprime ses données du localStorage pour éviter tout accès non autorisé.

10. Améliorations futures

Bien que le projet soit fonctionnel et opérationnel, plusieurs améliorations peuvent être apportées pour renforcer la sécurité, enrichir les fonctionnalités et améliorer l'expérience utilisateur.

10.1. Sécurité

- Hasher les mots de passe (bcrypt) (Mots de passe en clair dans la BDD).
- Authentification sécurisée (JWT, sessions) (Le joueur reste connecté malgré le serveur qui est arrêté et relancé).
- Vérification des rôles côté serveur et sécurisation des routes API (Sécuriser le site un minimum).

10.2. Nouvelles fonctionnalités

- Séries doubles avec choix du partenaire et vérification des points/sexe (Faire les séries double).
- Correction du système des séries et choix des séries par jour (Faire le système des séries, problème de départ que j'ai dû modifier pour avancer. Le choix des séries par jour aussi).
- Suppression des tournois expirés avec une tâche automatique (Faire en sorte que les tournois qui sont dépassés en date soient supprimés de la base de données).
- Amélioration du menu déroulant des séries pour ne pas proposer celles déjà sélectionnées (Quand on choisit les séries dans un nouveau tournoi, lorsqu'on choisit une série, elle est toujours proposée dans le menu déroulant mais on ne peut pas l'ajouter. Il faudrait qu'elle ne soit pas proposée simplement).
- Affichage du tournoi sélectionné sur le formulaire d'inscription (Sur le formulaire il faudrait mettre le tournoi en question pour savoir sur quel tournoi on s'inscrit).
- Envoi d'un mail de confirmation au joueur et à l'organisateur (Envoyer un mail de confirmation au joueur et à l'organisateur pour prendre en compte l'inscription).
- Extension à plusieurs sports (Extension de plusieurs sports).

10.3. Responsive design

- Suppression ou explication du bouton "Ajouter un tournoi" pour les joueurs non organisateurs (Ajouter une phrase pour prévenir les joueurs qui n'ont pas accès au bouton ajouter tournoi, ou alors juste le supprimer).
- Amélioration du design et meilleure organisation des boutons et filtres (Design).
- Optimisation mobile complète avec Flexbox et Grid (Optimisation mobile complète).

11. Conclusion

11.1. Bilan personnel

Ce projet m'a permis de développer un ensemble de compétences techniques et organisationnelles essentielles à la réalisation d'une application web complète. En travaillant sur ce système de gestion de tournois, j'ai été confronté à différentes problématiques qui m'ont amené à approfondir mes connaissances en développement full-stack.

Compétences techniques acquises

- I. Back-end avec Node.js et Express
 - Mise en place d'un serveur Express pour gérer les requêtes HTTP.
 - Création d'une API REST permettant de gérer les tournois, les joueurs et les inscriptions.
 - Gestion des routes et middlewares (CORS, gestion des erreurs, parsing JSON).
- II. Base de données avec MariaDB et Sequelize
 - Modélisation relationnelle des entités (Joueur, Tournoi, Série, Inscription).
 - Gestion des relations entre les tables avec Sequelize (One-to-Many, Many-to-Many).

- Requêtes SQL optimisées via Sequelize pour récupérer et manipuler les données.
- III. Développement front-end dynamique
- Utilisation de HTML5, CSS3 et JavaScript pour concevoir une interface interactive.
 - Manipulation du DOM pour afficher dynamiquement les tournois, joueurs et inscriptions.
 - Utilisation de fetch() pour effectuer des appels API et récupérer les données en temps réel.
- IV. Tests et débogage
- Utilisation de Postman et Thunder Client pour tester les routes API.
 - Ajout de logs et messages d'erreur clairs pour faciliter la correction des bugs.
 - Mise en place de vérifications et validations des données en front-end et back-end.

Gestion de projet et autonomie

Outre les compétences techniques, ce projet m'a permis d'améliorer mes compétences en gestion de projet. J'ai suivi une approche structurée en :

- Rédigeant un cahier des charges détaillé pour définir les objectifs et fonctionnalités.
- Planifiant le développement en différentes étapes : mise en place du serveur, création des modèles, développement des routes API, intégration front-end.
- Gérant les imprévus techniques, notamment les erreurs liées aux relations entre les tables Sequelize ou la gestion des rôles utilisateurs.
- Améliorant progressivement le projet grâce aux tests utilisateurs et aux retours obtenus.

Ce projet m'a donné une vision plus claire de la gestion d'un projet web, de sa conception initiale à sa mise en production.

11.2. Ouverture

I. Réutilisation des acquis pour d'autres projets



Les compétences acquises au cours de ce projet me seront très utiles pour d'autres développements web, notamment :

- Création d'applications web dynamiques avec un front-end interactif et un back-end robuste.
- Utilisation d'APIs et gestion de bases de données relationnelles avec Sequelize et MariaDB.
- Développement sécurisé avec une meilleure gestion des authentifications et des droits d'accès.

II. Améliorations possibles du projet

Le projet peut être amélioré et enrichi par l'ajout de nouvelles fonctionnalités :

- Sécurité et authentification avancée
 - Hashage des mots de passe avec bcrypt pour éviter leur stockage en clair.
 - Mise en place de JWT (JSON Web Token) pour une authentification sécurisée.
 - Gestion des sessions pour éviter que les utilisateurs restent connectés même après un redémarrage du serveur.
- Nouvelles fonctionnalités pour les joueurs et organisateurs
 - Gestion des doubles dans les séries : ajout d'un choix de partenaire et vérification des points/sexe pour les doubles mixtes.
 - Envoi automatique d'e-mails pour confirmer les inscriptions des joueurs et avertir les organisateurs.
 - Système d'archivage automatique pour supprimer ou masquer les tournois dépassés afin d'améliorer la clarté des données.
- Amélioration du design et de l'expérience utilisateur
 - Amélioration du thème sombre avec des couleurs plus harmonieuses et un design plus fluide.
 - Affichage dynamique des tournois et des séries disponibles sur la page d'inscription.
 - Optimisation mobile complète en utilisant Flexbox et Grid CSS pour un affichage plus réactif.

	Projet	
	COMPTE RENDU	

d) Gestion avancée des compétitions

- Génération automatique des grilles de tournoi avec les matchs à jouer.
- Tableau de bord permettant aux organisateurs de visualiser les statistiques des joueurs et du tournoi.
- Gestion de plusieurs sports pour élargir le projet à d'autres disciplines (badminton, football, basket, etc.).

III. Prochaines étapes et perspectives

Si le projet devait être poursuivi, les améliorations suivantes seraient à prioriser :

- Sécurisation du site avec hashage des mots de passe et vérifications serveur sur toutes les actions sensibles.
- Affinage de l'ergonomie pour rendre le site plus intuitif et facile d'utilisation.
- Ajout d'un mode administrateur avancé avec gestion des tournois et des joueurs directement depuis l'interface web.
- Publication du projet sur un hébergement en ligne pour permettre son utilisation réelle.

Conclusion générale

Ce projet m'a permis de mettre en application mes connaissances en développement web full-stack en combinant Node.js, Express, Sequelize et MariaDB pour le back-end, et HTML, CSS et JavaScript pour le front-end.

J'ai pu expérimenter l'intégration complète d'une API REST, la gestion de bases de données relationnelles et la création d'une interface utilisateur dynamique et ergonomique.

Au-delà des compétences techniques, ce projet m'a appris à travailler de manière autonome, à structurer un projet et à résoudre des problèmes concrets liés au développement web.

Ce travail constitue une base solide qui pourrait être étendue avec des fonctionnalités avancées et une meilleure sécurité. Ce projet représente une expérience précieuse qui pourra être réutilisée et améliorée dans de futurs développements.

Lien GitHub : <https://github.com/CyrilBourlet/ProjetL3>