

Dossier Projet

Léo Ségalini--Briant

leo.segalini@outlook.com



Dossier Projet

Site Padel - Agence Digital Iroko

Formation suivie à l'IFR (Saint-Pierre - Réunion) financée par le pole-emploi.
Année 2022-2023

Table Des matières

Introduction	5
Description du Projet	6
Compétences couvertes	6
Présentation du projet	6
Cahier des charges	8
La maquette	10
Le Front end	12
Le début du code	12
Un peu de dynamisme	15
Il n'y a pas que le code dans la vie	18
Respectons ces règles pour éviter les amendes	20
Le Back End	22
A l'attaque du MCD	22
Mettons en place la base de données	23
Match fait, match supprimé !	29
Publication	31
Réalisations	32
Les points clés	35
Glossaire	388
Annexes	40

Introduction

Un parcours des plus singuliers : après avoir obtenu un diplôme en biologie, j'ai exercé en tant qu'organisateur de mariages, chef de cuisine, commerciale en assurance, et ai travaillé dans le recouvrement. Me voici maintenant dans une formation en développement web. Pourquoi l'informatique maintenant ? Simplement, car ma passion, c'est d'apprendre. J'ai toujours touché à l'informatique, pas obligatoirement dans le développement web, mais plutôt pour aider les gens à résoudre leurs soucis de PC.

Suite à une envie de renouveau, je suis parti vivre à la Réunion. Arrivé sur l'île, j'ai cherché à me reconvertir et je me suis tourné vers le développement web. Et c'est pendant mes recherches que j'ai commencé à me former en autodidacte grâce à OpenClassroom. J'ai pu découvrir les bases d'HTML, CSS et JS et c'est à ce moment où j'ai compris que je prenais du plaisir dans ce domaine. De là, a commencé mon aventure avec l'IFR de Saint-Pierre.

J'ai réalisé mon stage dans l'entreprise digitale Iroko, basé à Saint-Denis. C'est une société qui réalise différents projets web, allant du site web simple via un CMS, en passant par des applications mobiles ou des logiciels pour aider des entreprises.

J'ai pu durant mon stage réaliser 6 projets différents, et découvrir que dans une boîte de digital le no-code reste important. Il est logique que nous ne pouvons pas nous permettre de faire constamment des sites à la main, en sachant que nous pouvons avoir des outils à disposition pour nous faire gagner du temps.

J'ai pu participer aux réunions clients, aux propositions de devis, aux validations de projet. Et j'ai pu mettre à disposition mes compétences acquises dans mes anciens métiers.

Le stage a été intense, cela a demandé une grande capacité d'adaptation, une prise en main rapide des différents outils utilisés. Le projet que je vais vous présenter est un site pour trouver des coéquipiers pour jouer au padel sur l'île. J'ai eu l'entière liberté sur l'utilisation du langage pour le backend, j'ai donc choisi JS avec NodeJs, Express pour le backend, MongoDB pour la base de données, et React pour le frontend.

Description du Projet

Compétences couvertes

Activité type n°1 : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

Activité type n°2: Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

Présentation du projet

Le projet est un site web permettant de réunir les joueurs de padel. Sur l'île, beaucoup de personnes ont du mal à trouver des partenaires pour effectuer un match, cela permet de se réunir en fonction de son niveau.

Le site doit être responsive et doit permettre une connexion utilisateur. Le choix des couleurs a pour but de rappeler les couleurs de la Réunion.

Chaque utilisateur aura accès à différents composants :

Les utilisateurs sans comptes :

- Pourrons s'abonner à la newsletter
- Pourrons contacter les administrateurs via un formulaire

Les utilisateurs avec un compte :

- Pourrons voir les différents matchs postés
- Pourrons contacter l'utilisateur qui a posté le match

- Pourrons poster des matchs et supprimer leurs matchs
- Modifier leurs données personnelles

Chaque jour à minuit, les matchs de la journée seront automatiquement supprimés grâce à une fonction qui tourne dans le backend.

Le site sera hébergé via infomaniak.

Actuellement, le site web n'est pas terminé totalement, il sera par la suite amélioré sur le plan front end pour créer plus de mouvements et d'animations. Également des pages seront créées, regroupant chaque club de padel de la réunion en fonction du secteur avec les informations du club (mail, adresse, téléphone, nom des différents terrains...). Ces pages seront accessibles via les flip-card de la page principale.

Méthode de travail

Avec mon responsable de stage, nous avons décidé d'utiliser la méthode SCRUM pour la gestion du travail. Cela nous a permis d'améliorer la réalisation des projets et de résoudre les points problématiques efficacement.

Chaque matin à 9h, nous effectuons un daily SCRUM pour indiquer ce que nous avons fait la veille, les problèmes rencontrés et ce que j'allais faire aujourd'hui. Cela permettait une meilleure analyse des tâches à effectuer et de mieux organiser sa journée de travail.

Les journées avec les objectifs annoncés laissés malgré tout une certaine liberté en cas d'imprévu. Le délai annoncé était surestimé, de manière à pouvoir réagir rapidement sur des retours client et les intégrer dans nos tâches à effectuer.

J'utilisais trello pour planifier mes journées avec : à faire, en cours et terminé. En fin de journée, j'envoyais un mail débriefant mon avancement de la journée à mon responsable, et régulièrement durant la journée, je le tenais informé lors de la clôture d'une tâche. C'est grâce à cette méthode de travail que j'ai pu fournir un total de 6 projets.

De mon côté, j'effectuais une veille constante, chaque soir après mes journées, je regardais les différents outils pouvant m'aider à résoudre les problèmes rencontrés, les évolutions et mises à jour des plugging utilisés...

Cahier des charges

Objectif

Iroko est une jeune entreprise de développement digital. Ce projet a pour but de réunir des amoureux ou des curieux du padel. Ce sport se jouant à deux contre deux, il est régulièrement compliqué de trouver des partenaires de jeu pour effectuer un match.

L'objectif est de faciliter l'organisation des rencontres sportives de padel en développant un outil numérique qui répond précisément aux attentes et aux habitudes de la cible. Ce projet vise à simplifier la planification et la gestion des événements de padel, offrant ainsi une expérience optimale aux utilisateurs. Par le biais de cet outil, l'entreprise cherche également à accroître la notoriété du padel, en mettant en avant les avantages et la convivialité de ce sport.

En parallèle, l'entreprise aspire à renforcer sa propre visibilité. En mettant en avant la valeur ajoutée du site web, l'entreprise cherche à attirer l'attention du public cible, des amateurs de padel, et des organisateurs d'événements sportifs. L'objectif final est de positionner l'entreprise comme un acteur clé dans la promotion du padel à travers une solution numérique performante, tout en consolidant sa propre renommée dans le domaine du développement d'outils spécialisés.

Spécificités fonctionnelles

- Création utilisateur
- Espace de connexion utilisateur / admin
- Espace de gestion BDD pour l'admin
- Ajout de nouveau club de padel pour l'admin
- Suppression d'un compte utilisateur
- Formulaire de contact
- Inscription à la newsletter
- Création/ modification et suppression de match
- Suppression des matchs de la journée à minuit
- Réservation de match
- Mise en place de champs de recherche
- Mise en place du système d'authentification

Spécifités Techniques

Front end :

- Utilisation du framework REACT
- Language HTML, CSS et JS
- Media query mis en place pour le responsive
- Utilisation d'axios pour la liaison avec le backend

Back end :

- Utilisation de NodeJs et Express
- Base de données gérées avec MongoDB et mongoose
- Mise en place de JWT
- Utilisation de node-cron

La maquette

La première étape consiste à étudier et à optimiser le parcours utilisateur. On met en place dans un premier temps un wireframe puis on passe à la maquette.

Wireframes

Commençons par définir ce qu'est un wireframe : un wireframe est un schéma de la structure et des fonctionnalités de l'application mobile ou du site. Il permet de gagner du temps lors de la création d'un site internet.

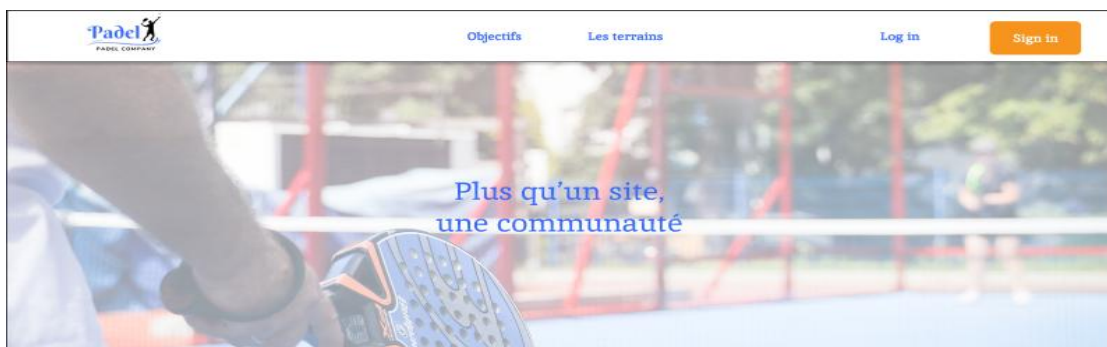


Pour créer le wireframe je suis passé par *Lucidchart* (Annexe 1).

Maquettage

La phase de maquettage constitue une étape déterminante dans la concrétisation de la vision pour le futur de notre site. Guidé par la consigne d'incorporer des couleurs évoquant l'esprit de la réunion, j'ai eu l'opportunité de laisser libre cours à ma créativité en ce qui concerne l'utilisation du logiciel et le design du site.

Doté d'une liberté totale dans le choix du logiciel et du design, j'ai opté pour *Figma*, un outil polyvalent qui m'a permis de donner vie à la structure et à l'apparence de notre plateforme. La flexibilité offerte par Figma a été un atout pour traduire de manière visuelle les idées et les fonctionnalités prévues pour le site.



Afin de garantir une expérience utilisateur fluide, j'ai intégré des liens interactifs entre les différentes pages de la maquette. Cela permet de simuler le parcours des utilisateurs à travers le site, offrant ainsi une vision concrète de la navigation. De plus, j'ai collaboré étroitement avec mon responsable de stage lors de discussions approfondies sur le design. Ces échanges ont été primordiaux pour valider les choix esthétiques et fonctionnels, assurant ainsi la cohérence avec la vision globale du projet.

Ces discussions ont permis d'anticiper et de discuter des défis potentiels, notamment dans la mise en œuvre du front-end. Cela a contribué à une meilleure compréhension des enjeux techniques et à une préparation adéquate pour les étapes suivantes du développement.

Il est important de préciser que la maquette est très proche du site final, mais il existe toujours quelques évolutions qui se font en cours de développement (Annexe 2).

Le Front end

Le début du code

Cette étape a consisté en l'intégration statique du site, sans la mise en place de la base de données.

Pour développer mon site, j'ai utilisé l'IDE Visual Studio. Aujourd'hui Visual Studio est un incontournable pour coder, il offre un grand nombre de plugins permettant de faciliter la création du code. Ces plugins ont pour but de nous aider, il ne faut pas avoir tendance à en devenir dépendant et à laisser l'apprentissage du code de côté.

Structure HTML

La structure HTML de la page a été élaborée avec le framework **REACT**.

React est divisé en plusieurs dossiers (Annexe 3-4) :

- Le dossier « node_modules » avec toutes les extensions installées
- Le dossier « public » permettant de faire le lien entre HTML et les fichiers javascript permettant de créer les pages
- Le dossier « src » permettant de mettre en place les dossiers et les pages pour le site.

Nous allons nous concentrer sur le dossier « src ». J'ai choisi de monter le dossier en cinq sous-dossiers. Ce qui nous intéresse le plus, pour le moment, ce sont les sous-dossiers « components » et « pages ».

Dans le dossier « components » il y a plusieurs sous-dossiers : le dossier « private », « public », « admin » et « img ». De même pour le dossier « pages ».

Et dans chaque dossier, il y a encore des sous-dossiers. Le système que j'ai utilisé pour monter les dossiers REACT pourrait être comparé à un système d'échafaudage. Cela peut paraître complexe à première vue, mais ça permet de bien séparer chaque section et de faciliter par la suite les modifications ou rajouts.

Il est important de noter que chaque page REACT est dans un fichier javascript. Le site complet a été passé sur le site **W3C** de façon à valider la structure HTML et avoir un site avec un meilleur référencement.

Intégration du CSS

Pour mettre en place le CSS, j'ai assigné à chaque page son dossier CSS personnalisé. Aucun framework n'a été utilisé. J'ai décidé de ne pas en utiliser pour éviter l'import d'une librairie extérieure qui pourrait ralentir la performance du site.

Pour faciliter la mise en place du CSS et surtout pour simplifier le responsive, j'ai utilisé des flexbox. Flexbox (boîte flexible) est un modèle de disposition pour la conception de l'interface utilisateur. Cela permet d'afficher les éléments en ligne ou en colonne.

Plusieurs autres éléments ont dû être utilisés, comme le z-index me permettant de définir quel élément est le plus en avant.

Par exemple, ma barre de navigation est en « position : fixed », ce qui permet lorsque l'on scroll vers le bas de la page d'avoir toujours le menu de navigation en haut. Mais il faut préciser sur certains éléments l'ordre des différentes couches pour éviter que des éléments passent devant la barre de navigation. Ci-dessous, on peut observer un z-index de ma barre de navigation à 1000 (chiffre exagéré, mais permettant d'être certain qu'il soit le premier en cas de modification par la suite) et à côté mon overlay_header en « position: absolute » mais avec un z-index de 1 pour éviter qu'il ne passe devant ma « nav_section_all ».

```
.nav_section_all {  
  width: 100%;  
  height: 80px;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  position: fixed;  
  top: 0;  
  background-color: white;  
  z-index: 1000;  
  box-shadow: 0px 2px 5px 0px rgba(0, 0, 0, 0.1);  
}
```

```
.overlay_header {  
  position: absolute;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background-color: rgba(255, 255, 255, 0.6);  
  z-index: 1;  
  transition: background-color 0.3s ease-in-out;  
}
```

Gestion du responsive design avec les requêtes média.

Commençons par définir ce que sont les requêtes média :

Les requêtes média (media queries) permettent de modifier l'apparence d'un site ou d'une application en fonction du type d'appareil (impression ou écran par exemple) et de ses caractéristiques (la résolution d'écran ou la largeur de la zone d'affichage (viewport) par exemple).

Il y a différentes façons d'utiliser les médias queries. Personnellement, j'ai opté pour 3 modes de responsive un mode PC supérieur à 1200px, un compris en 767px et

1200px et un inférieur à 797px (Annexe 5).

J'ai surtout utilisé ça, suite à l'expérience du stage. Dans d'autres sites produits, les clients avaient tendance à regarder en premier temps sur des écrans de 2000px ce qui pouvait entraîner des problèmes de responsive. En appliquant une largeur de 1200px en mode PC, 767px en mode tablette et 100% de la largeur en mode téléphone permet de rendre plus esthétique le site et de faciliter la gestion du responsive.

Les effets au survol

Dans la conception du site web, j'ai apporté une dimension visuelle enrichie en intégrant des éléments de style, conférant ainsi une profondeur esthétique. L'utilisation judicieuse d'effets tels que « box-shadow », « transition », et l'interaction au survol (hover) ont été des choix pour améliorer l'expérience visuelle globale.

Les boutons ont été particulièrement mis en valeur. De plus, l'utilisation de transitions fluides améliore la sensation d'interactivité, rendant les changements d'état plus agréables visuellement.



Les effets au survol (hover) ont été appliqués de manière stratégique pour générer des overlays dynamiques. Cette technique permet de mettre en évidence certaines zones interactives du site, offrant ainsi une rétroaction visuelle immédiate aux utilisateurs lorsqu'ils interagissent avec des éléments spécifiques.

```
.header:hover .overlay_header {  
  background-color: rgba(255, 255, 255, 0.8);  
}
```

L'intégration de l'ombre portée (box-shadow) crée une illusion de profondeur, conférant aux éléments une présence visuelle qui attire l'attention de manière subtile. J'ai pu mettre ça en place sur la barre de navigation rajoutant un effet de dominance dans le site.

```
box-shadow: 0px 2px 5px 0px rgba(0, 0, 0, 0.1);  
}
```

Ces choix de conception ne visent pas seulement à embellir l'esthétique du site, mais à renforcer l'ergonomie et à guider les utilisateurs de manière intuitive.

Un peu de dynamisme

Fonctionnement du routeur

L'avantage avec REACT, c'est le rafraîchissement automatique de section à chaque action. Le fait de cloisonner des blocs de pages facilite derrière l'actualisation des éléments. Cela permet d'éviter le rafraîchissement de toute la page à chaque fois, seules les parties concernées sont actualisées.

Le premier fichier important est le fichier « index.js », c'est ici où l'on fait le lien entre notre fichier HTML et nos fichiers JavaScript. Ici, on appelle tout simplement le dossier « App.js » en le liant au dossier « index.html ».

Ensuite, on peut mettre en place le Router pour appeler nos différentes pages ou accès dans « App.js ».

Chaque section (admin, public ou private) a son fichier router et layout indépendant. Cela permet d'éviter les appels répétitifs de composants entre chaque page et facilite le chargement.

Prenons l'exemple de ma partie Utilisateur, j'ai mis en place un fichier « UserRouter.js » et « UserLayout.js ». Au lieu d'appeler toutes mes routes dans « App.js », j'ai appelé mes routes dédiées aux utilisateurs (accessible seulement par les utilisateurs, avec identifiant et un mot de passe) dans le dossier « UserRouter.js ».

Puis dans l'« UserLayout.js », j'ai appelé mes composants fixes, comme la barre de navigation, le footer ou les navigations pour les mentions légales. Enfin, j'ai incorporé au centre un élément appelé « Outlet ». Il s'agit d'un composant spécial utilisé pour déterminer l'emplacement où les éléments de la route doivent être affichés dans une mise en page. Cela me permet de faciliter par la suite la gestion de mes pages. À la construction du site, cela peut paraître fastidieux, mais chaque effort fournit en amont permet de simplifier la tâche aux futurs développeurs qui pourrait travailler sur ce projet, tout comme le fait de commenter son code.

C'est ainsi qu'on peut rendre un site dynamique, grâce à REACT au lieu qu'à chaque clique la page complète se recharge, cela recharge seulement la partie « Outlet ».

Champs de recherche

J'ai dû également mettre en place une barre de recherche pour les utilisateurs et l'admin.

Fonctionnement :

- **State React** : J'utilise le hook `useState` de React pour gérer l'état local du composant. Deux états sont utilisés pour la recherche : `searchTerm` pour stocker le terme de recherche entré par l'utilisateur, et `filteredUsers` pour stocker la liste filtrée d'utilisateurs en fonction de ce terme.

```
const [searchTerm, setSearchTerm] = useState(""); // Terme de recherche pour filtrer les utilisateurs
const [filteredUsers, setFilteredUsers] = useState([]); // Liste des utilisateurs filtrés en fonction de la recherche
```

- **Gestion de la recherche** : La fonction `handleSearch` est appelée chaque fois que l'utilisateur modifie le champ de recherche. Cette fonction met à jour l'état `searchTerm` avec la nouvelle valeur et filtre la liste des utilisateurs en fonction de ce terme.

```
// Fonction pour gérer la recherche d'utilisateurs
const handleSearch = (searchTerm) => {
  setSearchTerm(searchTerm); // Mise à jour du terme de recherche
  // Filtrage des utilisateurs en fonction du terme de recherche
  const filteredUsers = users.filter((user) =>
    user.email.toLowerCase().includes(searchTerm.toLowerCase())
  );
  setFilteredUsers(filteredUsers); // Mise à jour de la liste des utilisateurs filtrés
};
```

- **Affichage des résultats** : Les résultats de la recherche sont affichés dans la partie du code où vous mappez les `filteredUsers` pour créer les cartes d'utilisateurs affichées à l'écran.

```
{filteredUsers.map((user) => (
  <div key={user._id} className="user-card">
```

- **Réinitialisation de la recherche** : J'ai une fonction `handleShowAllUsers` qui réinitialise le terme de recherche et affiche tous les utilisateurs non filtrés.

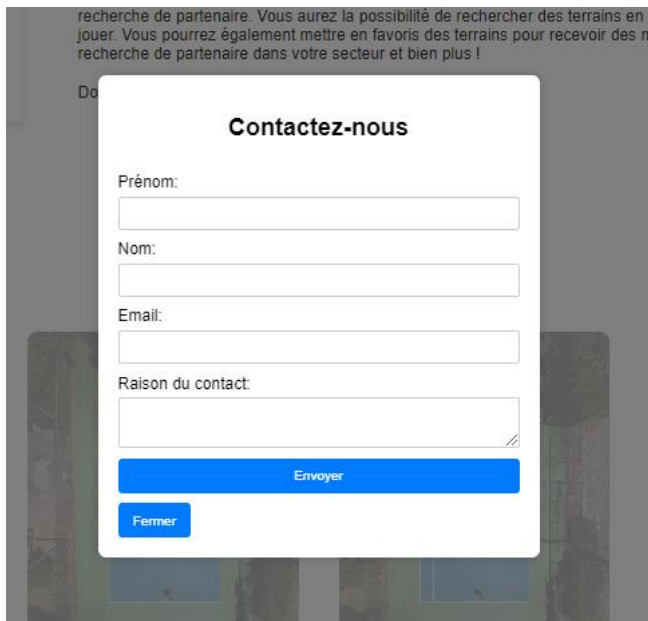
```
// Fonction pour afficher tous les utilisateurs
const handleShowAllUsers = () => {
  setFilteredUsers(users); // Affichage de tous les utilisateurs
  setSearchTerm(""); // Réinitialisation du terme de recherche
};
```

En résumé, l'implémentation permet une recherche en temps réel côté client, sans nécessité de requête réseau supplémentaire. Cela améliore l'efficacité et la réactivité de l'interface utilisateur en évitant des chargements fréquents depuis le serveur lors

de la recherche d'utilisateurs.

Il faut que ça Pop-up !

J'ai rendu l'expérience utilisateur de notre site web plus dynamique en intégrant judicieusement des pop-ups, améliorant particulièrement la section de contact et l'interface admin. Pour simplifier le processus de communication, j'ai mis en place des pop-ups interactifs qui ajoutent une touche conviviale à notre plateforme.



Dans la section contact, j'ai créé un pop-up astucieux qui s'affiche lorsque les utilisateurs cliquent, présentant ainsi un formulaire intuitif. Cela permet aux visiteurs de remplir les champs de manière pratique et rapide, améliorant ainsi l'accessibilité et l'efficacité de la communication.

Du côté de l'administration, la gestion des demandes de suppression de données est désormais simplifiée grâce à l'ajout d'un pop-up de validation. Lorsqu'une requête de suppression est soumise, ce pop-up offre une confirmation visuelle, apportant une couche supplémentaire de sécurité et de contrôle sur les actions administratives.

J'ai implémenté ces fonctionnalités interactives en utilisant des scripts JavaScript, renforçant ainsi la dynamique du site. Ces pop-ups ne se contentent pas d'améliorer l'esthétique, mais ils optimisent également l'ergonomie et l'efficacité des processus, offrant ainsi une expérience utilisateur plus fluide.

```
/* Popup de contact */
{isPopupOpen && (
  <ContactUsPopup
    formData={formData}
    onClose={() => setIsPopupOpen(false)}
    setFormData={setFormData}
    onChange={handleInputChange}
    onSubmit={handleContactUs}
  />
)}
</div>

const [isPopupOpen, setIsPopupOpen] = useState(false);
const [formData, setFormData] = useState({
  firstName: "",
  lastName: "",
  email: "",
  reason: "",
});
```

Il n'y a pas que le code dans la vie

Accessibilité

Dans cette optique, j'ai pris l'initiative de soumettre le site au W3C, un organisme de standardisation du web. Ce processus m'a permis d'identifier et de corriger des lacunes telles que des oublis d'attributs, des erreurs de balisage, et d'autres aspects impactant l'accessibilité.

Le recours au W3C s'inscrit dans une démarche proactive visant à conformer le site aux normes internationales d'accessibilité. Ces corrections apportées permettent d'assurer que le site est utilisable par tous, indépendamment des capacités physiques ou des dispositifs d'assistance utilisés par les visiteurs.

En considérant l'importance croissante accordée à l'accessibilité, cette démarche contribue à renforcer la qualité globale du site tout en démontrant l'engagement envers une expérience utilisateur inclusive. La conformité aux normes du W3C sert également à anticiper les évolutions futures du web, garantissant une pérennité et une adaptabilité du site aux avancées technologiques et aux exigences réglementaires.

Référencement

Le référencement est un point très important notamment en termes de visibilité en ligne, de trafic organique, et d'impact sur la notoriété et la réussite d'une entreprise. J'ai donc dû appliquer les règles de référencement (SEO - Search Engine Optimization).

Exemples de Bonnes Pratiques SEO Appliquées :

- **Optimisation des Mots-Clés** : Une recherche approfondie des mots-clés pertinents a été effectuée pour chaque page du site, intégrant naturellement ces termes dans le contenu, les titres, et les balises méta pour améliorer la pertinence aux yeux des moteurs de recherche.
- **Contenu de Qualité** : La création de contenu de qualité, informatif et engageant, est cruciale pour le SEO. Les articles, descriptions de produits et autres contenus ont été soigneusement rédigés, répondant aux besoins des utilisateurs tout en respectant les exigences des moteurs de recherche.
- **Optimisation des Images** : Les images ont été compressées sans compromettre

la qualité, et les balises alt ont été utilisées de manière descriptive pour optimiser le référencement des images.

Sauvegarde

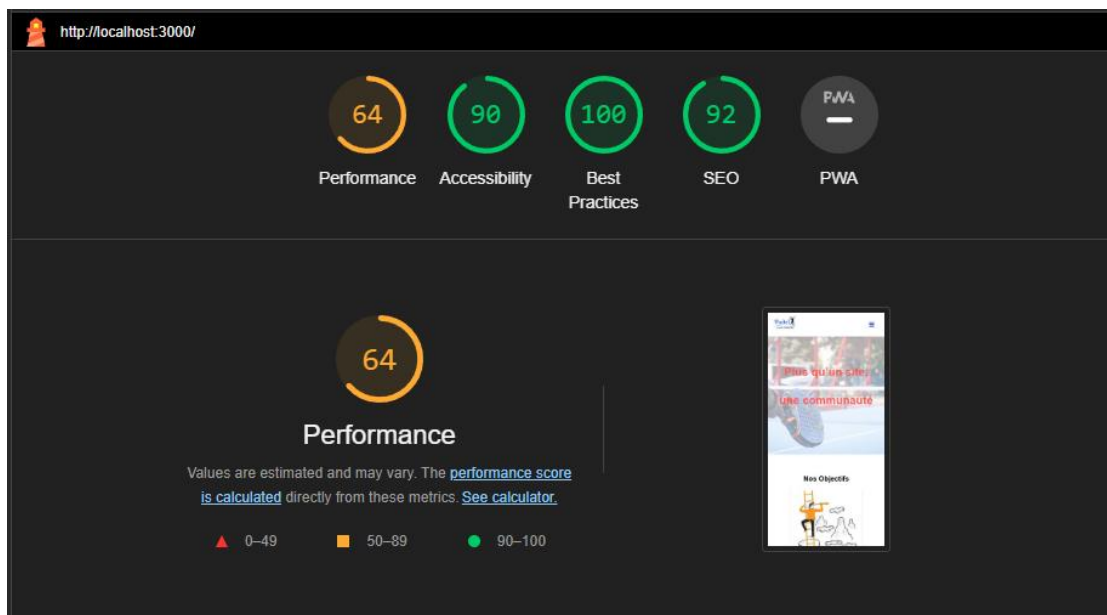
Chaque jour, je veillais à la sécurité de mes projets en effectuant une sauvegarde sur GitHub en mode privé, afin de prévenir tout accès non autorisé de la part de la concurrence. En parallèle, en fin de journée, j'adoptais la pratique de sauvegarder mon site sur GitLab à l'aide de Git Bash. Cette approche triple me garantissait trois copies de mes fichiers : une en local sur mon PC, et deux en ligne, renforçant ainsi la protection de mes données.

De plus, j'ai mis en place une sauvegarde de ma BDD à minuit.

Cette méthode rigoureuse de sauvegarde s'est avérée être une bouée de sauvetage lorsqu'un problème informatique est survenu au cours de ma formation, entraînant la perte totale de mes fichiers. Cet incident m'a enseigné l'importance cruciale de la sauvegarde régulière de chaque projet.

Optimisation

Après une analyse approfondie de la performance de mon site avec *Lighthouse*, j'ai identifié un problème majeur avec un score initial de 64.



L'origine de cette baisse de performance était attribuée aux plugins utilisés, en particulier *FontAwesome*. Bien que cet outil facilite l'intégration d'icônes en ligne, j'ai constaté que son utilisation était excessive pour mes besoins limités en icônes. Pour résoudre ce problème, j'ai décidé de télécharger directement les icônes au format

SVG et de les intégrer manuellement.

Ce simple ajustement a considérablement amélioré mon score de performance, le faisant passer à 85.

Poursuivant mon engagement pour des performances optimales, j'ai exploré d'autres aspects impactant la vitesse de chargement des pages. En effectuant une veille technologique approfondie, j'ai identifié un problème lié au chargement excessif des éléments. Pour résoudre ce problème, j'ai intégré la fonction Lazyload() à certains endroits spécifiques de mon code, permettant le chargement des éléments uniquement au moment où ils sont nécessaires.

Respectons ces règles pour éviter les amendes

La RGPD, un élément important des sites web et surtout à ne pas à négliger ! Mais qu'est-ce que la RGPD ?

La conformité à la Règlementation Générale sur la Protection des Données (RGPD) est une préoccupation centrale dans le développement d'un site web. Cette réglementation européenne vise à protéger la vie privée des utilisateurs en imposant des normes strictes sur la collecte, le traitement et la conservation des données personnelles. Dans le cadre du développement, cela signifie intégrer des fonctionnalités de confidentialité telles que des bannières de consentement, des paramètres de confidentialité, et des mécanismes de suivi des cookies. Les développeurs doivent également veiller à une gestion sécurisée des données, à la mise en place de mesures de sécurité robustes, et à la documentation adéquate des pratiques liées à la protection des données. La RGPD a ainsi remodelé le paysage du développement web en mettant l'accent sur la transparence, le consentement éclairé et la sécurité des données tout au long du cycle de vie d'un site web.

Application sur le site

Collecte et Gestion des Données Personnelles sur Notre Site

Au sein de la plateforme, dédiée à la mise en relation des passionnés de padel, la confidentialité et à la sécurité des informations personnelles de nos utilisateurs sont un domaine important. Les données recueillies, telles que le nom, le prénom et le numéro de téléphone, sont essentielles pour faciliter la mise en lien entre sportifs. C'est grâce à cela que les utilisateurs peuvent réserver les créneaux de match.

Transparence et Consentement des Utilisateurs

Il est impératif de souligner que chaque donnée collectée est utilisée exclusivement dans le contexte spécifique de la mise en relation des sportifs. Afin d'assurer la transparence et de recueillir le consentement des utilisateurs, j'ai mis en place une solution proactive pour la gestion des cookies. Un pop-up dédié sera intégré au site lors de son déploiement, grâce à [CookiesYes](#), sollicitant ainsi l'approbation des utilisateurs pour l'acceptation des cookies.

Mentions Légales et Page Dédiée à la Confidentialité

Dans le souci de renforcer la confiance des utilisateurs, j'ai pris des mesures supplémentaires en élaborant des mentions légales, complètes. Ces dernières détaillent l'ensemble de nos engagements, les droits des utilisateurs et les procédures de gestion des données personnelles.

L'avantage est que pour générer des mentions légales il existe des sites nous facilitant la création (ex: [Legalsart](#)).

De plus, une page dédiée spécifiquement à la gestion des données personnelles a été intégrée au site. Cette page informe les utilisateurs sur la manière dont leurs données sont collectées, stockées, et utilisées. Elle offre également un accès facile aux paramètres de confidentialité, permettant aux utilisateurs de contrôler leurs préférences en matière de données personnelles.

Engagement en Matière de Confidentialité

L'engagement envers la protection des données personnelles ne se limite pas à la conformité légale, mais vise à établir une relation de confiance durable avec nos utilisateurs.

Le Back End

A l'attaque du MCD

La Modélisation Conceptuelle des Données (MCD) est une étape avant la création d'un site web. Elle consiste à concevoir de manière abstraite la structure des données et les relations entre celles-ci. L'importance du MCD réside dans sa capacité à fournir une vision claire et structurée de la manière dont les données seront organisées et interconnectées sur le site.

En définissant les entités, les attributs et les relations, le MCD offre une base solide pour la création de la base de données du site. Cette démarche préliminaire permet de visualiser et de planifier efficacement l'architecture des données, facilitant ainsi le développement du site en assurant une cohérence dans la gestion des informations et en réduisant les risques d'erreurs lors de l'implémentation. En somme, le MCD constitue un outil essentiel pour garantir la cohérence, la qualité, et l'efficacité dans la gestion des données dès les premières étapes du processus de développement.

J'ai utilisé le site [Lucidchart](#) pour créer mon MCD (Annexe 6).

Dans un MCD il y a 2 choses importantes à comprendre.

En premier lieu, les bords des rectangles dans un MCD peuvent avoir deux formes distinctes : angulaires ou arrondis. Les rectangles à bords angulaires dénotent les entités fortes, qui sont des entités indépendantes et capables de subsister par elles-mêmes. À l'inverse, les rectangles à bords arrondis représentent des entités faibles, qui dépendent d'une autre entité pour exister.

Dans un second temps la cardinalité, qui indique le nombre d'occurrences d'une entité qui peuvent être associées à une occurrence de l'autre entité. Les cardinalités sont généralement indiquées par des chiffres (0, N). Ces notations permettent de définir les relations, qu'elles soient de type un-à-un, un-à-plusieurs, ou plusieurs-à-plusieurs entre les entités.

Mettons en place la base de données

Pour ma part, sachant que j'allais utiliser *NodeJs* et *Express* pour mon backend, j'ai choisis *MongoDB* pour ma base de données.

MongoDB est une base de données NoSQL réputée pour sa flexibilité et son stockage de données basé sur des documents au format JSON-like. L'utilisation de MongoDB commence souvent par la création d'un cluster, géré par MongoDB Atlas dans le cloud, qui permet le stockage redondant et la disponibilité des données. MongoDB Compass, un outil graphique officiel, facilite la création de bases de données et de collections, ainsi que la manipulation visuelle des documents.

J'ai donc créé une collection "padel_section". À l'intérieur de cette collection, j'ai organisé mes données en cinq dossiers distincts : « *contacts* », « *matchpadels* », « *newsletters* », « *terrainpadels* », et « *users* ». Chacun de ces dossiers contient des fichiers JSON spécifiques, représentant différents types d'entités ou d'informations.

Par exemple, « *contacts* » stock les personnes ayant essayé de joindre l'admin, « *matchpadels* » contient les différents matchs poster par les utilisateurs, « *newsletters* » stock les mails des personnes souhaitant s'inscrire à la newsletter, « *terrainpadels* » contient les différents clubs de padel sur l'île, et « *users* » enregistre les informations des utilisateurs.

Ces dossiers ont été intégrés via le code mis en place dans le backend. Pour la mise en place de mon dossier backend j'ai segmenté en trois parties (Annexe 7). La première est le dossier « *models* ». Le dossier « *models* » contient mes différents fichiers qui définissent les modèles de données de l'application. Un modèle représente une entité dans la base de données et définit sa structure. Dans ces fichiers, on y trouve :

- **Schéma de Données** : La structure des données associée à l'entité, décrivant les champs, les types de données, les contraintes, etc.
- **Logique Métier liée aux Données** : Certaines opérations métiers qui sont spécifiques aux données gérées par le modèle.

Ensuite le dossier « *controller* ». Il contient des fichiers qui définissent les contrôleurs de l'application. Un contrôleur est responsable de la gestion des requêtes HTTP, de la logique métier et de la coordination des actions à effectuer en réponse à ces requêtes. Chaque fichier controller peut être dédié à un ensemble spécifique d'actions ou de fonctionnalités liées. Dans ces fichiers, on y trouve :

- **Interactions avec la Base de Données** : Les requêtes et les opérations nécessaires pour interagir avec la base de données, telles que la création, la lecture, la mise à jour et la suppression (CRUD).
- **Fonctions de Contrôleur** : Des fonctions qui gèrent les différentes routes définies dans le dossier "route". Ces fonctions interagissent avec les modèles pour récupérer ou manipuler des données.
- **Validation des Données** : La vérification et la validation des données entrantes des requêtes pour s'assurer de leur intégrité et de leur conformité.
- **Logique Métier** : La logique spécifique à l'application qui est exécutée lors de l'appel des différentes routes.

```
// Fonction pour obtenir la liste de tous les utilisateurs
module.exports.getUsers = async (req, res) => {
  const users = await UserModel.find();
  res.status(200).json(users);
};

// Fonction pour créer un nouvel utilisateur
module.exports.createUser = async (req, res) => {
  // Vérifiez si tous les champs nécessaires sont présents
  if (!req.body.nom || !req.body.motDePasse || !req.body.email) {
    return res
      .status(400)
      .json({ message: "Merci de remplir tous les champs" });
  }

  // Créez un nouvel utilisateur
  const user = await UserModel.create({
    role: req.body.role,
    nom: req.body.nom,
    prenom: req.body.prenom,
    email: req.body.email,
    motDePasse: req.body.motDePasse,
    photo: req.body.photo,
    dateDeNaissance: req.body.dateDeNaissance,
    niveauPadel: req.body.niveauPadel,
    secteurJeu: req.body.secteurJeu,
    telephone: req.body.telephone
  });

  res.status(200).json(user);
};
```

Et pour finir le dossier « route ». Il contient des fichiers qui définissent les routes de l'application, c'est-à-dire les points d'entrée accessibles via les requêtes HTTP. Dans ces fichiers, nous avons :

- **Définition des Routes** : L'assignation des fonctions de contrôleur aux différentes routes de l'API. Cela spécifie quel contrôleur et quelle fonction doit être appelée pour chaque type de requête (GET, POST, PUT, DELETE, etc.).
- **Gestion des Paramètres de Requête** : La récupération et la validation des paramètres de requête, des paramètres de chemin, des données de formulaire, etc.

Rajoutons un peu de sécurité

Différents points ont été mis en place pour plus de sécurité.

Déjà la mise en place d'un fichier « *db.js* » et « *.env* », pour simplifier la connexion à la base de données. Au lieu d'avoir à chaque fonction l'appel de la base de données avec son URL dans lequel on peut lire le nom utilisateur et le mot de passe de la base de données. Nous séparons les informations. Dans « *db.js* » on retrouve le squelette (Annexe 8). Ce code assure la configuration de la connexion à la base de données MongoDB en encapsulant les détails de la connexion dans une fonction réutilisable. En utilisant Mongoose, cela facilite l'interaction avec la base de données, permettant ainsi à l'application Node.js d'accéder et de manipuler les données stockées dans MongoDB de manière asynchrone.

Ensuite le « *.env* » qui regroupe toutes les informations sensibles telles que le nom d'utilisateur, le mot de passe...etc.

Cela permet lors des sauvegardes en ligne, grâce au fichier « *.gitignore* » de préciser quel fichier nous ne souhaitons pas importer et de conserver certaines données secrètes.

Pour la partie « *users* » j'ai mis en place, pour plus de sécurité, un encodage du mot de passe et une obligation de remplir certains points.

```
motDePasse: {
  type: String,
  required: true,
  validate: {
    validator: function (value) {
      return /^(?=.*[A-Z])(?=.*\d)(?=.*[\W_])/.test(value);
      // Validation du mot de passe : au moins une lettre majuscule, un chiffre, un caractère spécial
    },
    message:
      "Le mot de passe doit contenir au moins une lettre majuscule, un chiffre, un caractère spécial",
  },
}
```

La personne est obligée de mettre une lettre majuscule, un chiffre et un caractère spécial si elle veut créer son mot de passe. Cela rajoute dans un premier temps un aspect sécurité.

Pour plus de sécurité, j'ai rajouté un middleware pour hacher le mot de passe avant de le sauvegarder dans la base de données. De cette manière dans la base de données, le mot de passe n'est pas visible. (Annexes 9).

Un autre point de sécurité (oui ça en fait de la sécurité !) mis en place est le token d'authentification ou JWT. Il a plusieurs avantages en termes de sécurité. Je vais en lister, certain mais je détaillerai plus tard les démarches effectuées et commenterai mon code.

- **Protection des Informations Sensibles** : Les tokens JWT permettent de stocker des informations spécifiques à l'utilisateur de manière sécurisée, évitant ainsi la nécessité de stocker ces données côté serveurs.
- **Sécurité des Identifiants** : Les informations d'identification sont vérifiées par rapport à la base de données, assurant que seuls les utilisateurs authentiques peuvent générer des tokens.
- **Gestion Sécurisée de la Déconnexion** : La liste noire offre un mécanisme sécurisé pour invalider les tokens après une déconnexion, empêchant ainsi toute utilisation ultérieure.
- **Prévention de l'Usurpation d'Identité** : La vérification du token avec la clé secrète garantit l'authenticité du token, réduisant ainsi les risques d'usurpation d'identité.

Testons notre base de données

Pour tester la fonctionnalité de ma base de données MongoDB, j'ai employé Postman, un outil de test d'API qui offre une interface conviviale pour effectuer des requêtes HTTP et évaluer les réponses. Cette approche m'a permis de soumettre des fichiers JSON statiques directement dans ma base de données, en testant ainsi l'efficacité des méthodes CRUD (Create, Read, Update, Delete) (Annexe 10).

En utilisant Postman, j'ai pu facilement simuler des requêtes GET pour récupérer des données, des requêtes POST pour ajouter de nouvelles informations, des requêtes PUT pour mettre à jour des données existantes, et des requêtes DELETE pour supprimer des enregistrements. En envoyant des fichiers JSON en dur via ces requêtes, j'ai pu évaluer la réactivité de ma base de données et m'assurer que chaque opération fonctionnait correctement.

Cette approche de test avec Postman a joué un rôle essentiel dans la validation de la robustesse et de la fonctionnalité de ma base de données MongoDB. Les retours visuels immédiats fournis par Postman ont grandement simplifié le processus de développement et de débogage, garantissant ainsi le bon fonctionnement de mes méthodes d'interaction avec la base de données.

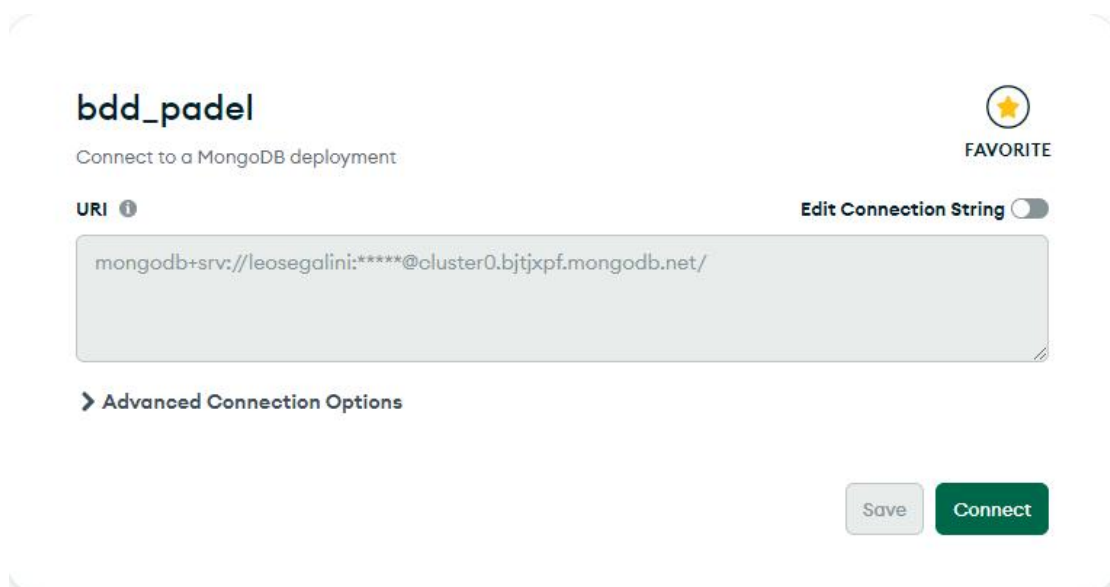
Une autre approche de test est possible avec les tests unitaire ou les test d'intégrations. Ils jouent un rôle essentiel dans le développement logiciel en permettant aux développeurs de valider individuellement chaque composant de leur code. Ces tests visent à évaluer le bon fonctionnement des fonctions ou méthodes spécifiques d'une application de manière isolée. Lorsqu'un environnement de test est

correctement mis en place, il peut être activé à chaque modification du code, offrant ainsi une validation continue du bon état de l'application.

J'ai opté dans mon cas pour un test d'intégration avec la route « GET /users », cela me permet de voir si je peux récupérer tous les utilisateurs (Annexes 11). J'utilise dedans Chai et Chai-Http pour faciliter la rédaction des assertions qui valide le comportement de la route. Dans l'exemple le test vérifie simplement que la réponse a un statut HTTP 200.

Voici le JWT et l'authentification

Une fonctionnalité que j'ai mise en place et la possibilité à un admin d'avoir accès au backend via un URI d'authentification de mongoDB avec un identifiant et un mot de passe. Bien évidemment l'admin ne va pas constamment faire les modifications de la base de données en passant par mongoDB compass, cela serait trop compliqué et pas ergonomique.



C'est pour cela que j'ai mis en place également un espace admin mettant à disposition toutes les possibilités envisageables dans le backend.

Pour que l'admin puisse s'identifier, deux choses rentrent en jeu : son role qui est définis sur admin et son token. Qui se génère lors de sa connexion. J'ai donc mis en place dans le controller des fonctions me permettant de générer ce JWT (Annexe 12).

Commençons par définir ce qu'est un JWT. JWT signifie "JSON Web Token". C'est un format compact et autonome pour représenter des informations entre deux parties de manière sécurisée. Ces informations peuvent être vérifiées et validées, car le token est signé numériquement. Un JWT est généralement utilisé pour l'authentification et l'échange sécurisé de données entre différentes parties d'une application.

Dans mon code, j'ai plusieurs fonctions importantes me permettant de générer le JWT.

La fonction **authenticateUser** : Cette fonction gère le processus d'authentification des utilisateurs. Lorsqu'un utilisateur soumet son email et son mot de passe, le serveur vérifie ces informations par rapport à la base de données à l'aide du modèle UserModel. Si les informations d'identification sont valides, un token JWT est généré en utilisant la bibliothèque jsonwebtoken. Ce token contient des informations spécifiques à l'utilisateur telles que son identifiant, son rôle, son nom, son prénom, sa photo, son email et son numéro de téléphone. Le token est ensuite renvoyé en tant que réponse au client, lui permettant de s'authentifier dans les requêtes ultérieures.

La fonction **logout** : Cette fonction gère la déconnexion des utilisateurs. Lorsqu'un utilisateur souhaite se déconnecter, le client envoie le token JWT associé dans l'en-tête d'autorisation de la requête. Le serveur extrait ce token, l'ajoute à une liste noire (blacklistedTokens) et confirme la déconnexion en renvoyant un message de réussite. La liste noire est utilisée ultérieurement pour vérifier la validité des tokens dans le middleware authenticateToken.

Le **middleware authenticateToken** : Ce middleware est utilisé dans d'autres parties de l'application pour valider la présence et la validité des tokens JWT dans les requêtes. Tout d'abord, il vérifie si un token est présent dans l'en-tête d'autorisation. Ensuite, il consulte la liste noire pour s'assurer que le token n'a pas été invalidé suite à une déconnexion. Enfin, il utilise la clé secrète pour vérifier l'authenticité du token. Si tout est en ordre, le middleware autorise la poursuite du traitement de la requête en ajoutant les informations de l'utilisateur au paramètre req.user.

En résumé, ce code met en œuvre une gestion sécurisée de l'authentification des utilisateurs, garantissant la confidentialité des informations sensibles, la protection contre l'usurpation d'identité, et une déconnexion gérée de manière sûre. Les JWT facilitent le transfert d'informations d'authentification de manière sécurisée entre le client et le serveur.

L'authentification en front end

De plus du côté front end, pour être certain que la personne qui se connecte est un rôle admin et soit bien redirigé vers le bon compte, j'ai mis une fonction JavaScript (Annexe 13). Elle gère la soumission d'un formulaire de connexion dans une application web. Lorsqu'un utilisateur tente de se connecter, la fonction vérifie d'abord le nombre d'échecs de connexion. Si ce nombre atteint trois, un message indiquant que le compte est bloqué est affiché, et la fonction se termine.

Ensuite, la fonction fait appel à un service (accountService) pour effectuer la connexion en fournissant les informations du formulaire. Si la requête de connexion réussit, les informations de l'utilisateur sont stockées localement et son token est sauvegardé. Selon le rôle de l'utilisateur, la navigation est redirigée soit vers "/admin" s'il s'agit d'un administrateur, soit vers "/homeco" pour un utilisateur standard.

J'ai également rajouté dans la partie Dashboard administrateur une sécurité complémentaire pour éviter qu'un utilisateur quand il se connecte puisse, juste en modifiant l'URL, avoir accès à la partie admin.

```
// Vérification si l'utilisateur a le rôle d'administrateur
if (!user || user.role !== 'admin') {
  // Redirection de l'utilisateur vers une page non autorisée ou autre gestion en cas d'accès non autorisé
  return <div className="acces_bloque">Accès non autorisé</div>;
}
```

Match fait, match supprimé !

Un point important du backend qui m'a demandé une veille importante est la fonction mise en place pour supprimer les matchs passés de la journée. J'ai du utiliser *node-cron* (Annexe 14).

La bibliothèque node-cron est employée pour planifier une tâche cron qui se déclenche tous les jours à minuit. Cette tâche est essentielle pour effectuer la maintenance régulière de la base de données.

La fonction de la tâche « cron » est définie de manière asynchrone et comporte plusieurs étapes. Premièrement, elle utilise la bibliothèque « moment » pour obtenir la date et l'heure actuelle. Ensuite, elle procède à la suppression des documents de la collection Match dont la date est antérieure au début de la journée actuelle.

En cas d'erreur pendant l'opération de suppression des documents, un message d'erreur est affiché dans la console, assurant ainsi une gestion robuste des erreurs. Enfin, quelle que soit l'issue de l'opération, le script se déconnecte proprement de la base de données MongoDB à l'aide de mongoose.disconnect().

L'utilité de node-cron réside dans son automatisation d'une tâche de nettoyage quotidienne de la base de données. En particulier, il cible la collection Match et supprime les documents qui ne sont plus nécessaires, améliorant ainsi l'efficacité et les performances de la base de données.

Node-cron est une bibliothèque populaire qui offre une solution simple et efficace pour planifier des tâches récurrentes en Node.js. Son intégration dans ce script facilite la gestion régulière des données dans la base de données MongoDB, contribuant ainsi à maintenir la cohérence et l'intégrité des données au fil du temps. Ce script représente une bonne pratique dans le contexte de la maintenance automatisée des bases de données dans les applications web.

Publication

Le projet a été entièrement mis en place en environnement local, sans recourir à l'utilisation de machines virtuelles. Actuellement, nous avons atteint une étape clé du développement en partageant une première démo vidéo avec les clients, dans l'attente de leurs retours. Cette démarche préliminaire vise à recueillir des commentaires essentiels qui orienteront les ajustements futurs du site.

Afin de garantir la robustesse et la sécurité du site, des tests approfondis ont été menés tant par moi-même que par mon responsable de stage. Ces tests ont couvert divers aspects, notamment la sécurité du site web et de la base de données, assurant ainsi une infrastructure solide et résiliente face aux éventuelles vulnérabilités.

Nous anticipons le passage à une phase d'hébergement sur la plateforme Infomaniak dans un avenir proche. Cette transition vers un environnement de production en ligne confirmera la performance du site dans des conditions réelles et constituera une étape cruciale avant son déploiement final.

Par la suite, notre vision pour le projet s'étend vers le développement d'une application mobile avec Flutter. Cette approche multi-plateforme nous permettra de fournir une expérience utilisateur homogène et performante, tout en maximisant l'efficacité du développement.

En somme, le projet évolue de manière prometteuse, avec une attention particulière portée à la sécurité et à la performance. Les démarches actuelles, du développement local à l'attente des retours client, préparent le terrain pour une transition réussie vers la phase d'hébergement sur Infomaniak et l'expansion vers le développement mobile avec Flutter, renforçant ainsi notre engagement envers la réussite du projet à chaque étape.

Réalisations

Realisation importante

J'ai rencontré quelques soucis au niveau du mot de passe par la suite, ce qui m'a forcé à modifier mon code pour le middleware. Lorsqu'un utilisateur souhaitait modifier son mot de passe, le nouveau mot de passe n'était pas haché. J'ai donc dû effectuer une veille technologique sur le sujet pour m'aider.

J'ai trouvé ma réponse sur <https://blog.logrocket.com/implementing-secure-password-reset-node-js/>.

Pourquoi ce site ? Quels ont été mes démarches ?

J'ai commencé par faire une recherche avec « password hache mongoDB reset ». Je suis tombé sur quatre sites, un de stack overflow, deux de mongoDB et le blog de logrocket.

J'ai visionné les quatre sites et celui qui se rapprocher le plus du code que j'avais écrit était logrocket. Donc j'ai suivi leur logique et l'ai appliqué à mon code. Joie et bonheur, j'ai ressenti, quand le code à fonctionné !

----> Text Original

The user model will define how user data is saved in the database. It is important to ensure that passwords are stored securely, as storing them in plaintext is a security risk. To avoid this, we can use a secure one-way hashing algorithm such as *bcrypt*, which includes a salt to increase the strength of the hashing further.

In the code below, we use *bcrypt* to hash the passwords to protect them and make it almost impossible to reverse the hashing process even if the database is compromised. Even as administrators, we should not know the plaintext passwords of our users, and using a secure hashing algorithm helps to ensure this as well:

The password is hashed using the pre-save MongoDB Hook before saving it, as shown in the code below. A salt of 10 is used, as specified in the .env file, to increase the strength of the hashing and reduce the likelihood of passwords being guessed by malicious actors:

```
// user.model.js
const mongoose = require("mongoose");
const bcrypt = require("bcrypt");
const Schema = mongoose.Schema;
const bcryptSalt = process.env.BCRYPT_SALT;
const userSchema = new Schema(
  {
    name: {
      type: String,
      trim: true,
      required: true,
      unique: true,
    },
    email: {
      type: String,
      trim: true,
      unique: true,
      required: true,
    },
    password: { type: String },
  },
  {
    timestamps: true,
  }
);
userSchema.pre("save", async function (next) {
  if (!this.isModified("password")) {
    return next();
  }
  const hash = await bcrypt.hash(this.password, Number(bcryptSalt));
  this.password = hash;
  next();
});
module.exports = mongoose.model("user", userSchema);
```

-----> Traduction :

Le modèle utilisateur définit la manière dont les données utilisateur sont enregistrées dans la base de données. Il est essentiel de garantir la sécurité du stockage des mots de passe, car les conserver en texte brut présente un risque pour la sécurité. Pour éviter cela, nous pouvons utiliser un algorithme de hachage à sens unique sécurisé tel que bcrypt, qui inclut un sel pour renforcer davantage la robustesse du hachage.

Dans le code ci-dessous, nous utilisons bcrypt pour hacher les mots de passe afin de les protéger et de rendre le processus de hachage pratiquement irréversible, même en cas de compromission de la base de données. En tant qu'administrateurs, il est crucial de ne pas connaître les mots de passe en texte brut de nos utilisateurs, et l'utilisation d'un algorithme de hachage sécurisé contribue à garantir cela également.

Le mot de passe est haché en utilisant le hook MongoDB pre-save avant d'être enregistré, comme indiqué dans le code ci-dessous. Un sel de 10 est utilisé, tel que spécifié dans le fichier .env, pour renforcer le hachage et réduire la probabilité que des acteurs malveillants devinent les mots de passe.

----> Explication :

Le souci que je rencontrais était au niveau de mon « *if (this.isModified("motDePasse"))* » je n'avais pas appelé « *isModified* ». Parfois, ce sont les choix les plus simples où l'on passe le plus de temps.



Connexion en tant qu'Administrateur :

- **Email** : leo@iroko.io (adresse e-mail associée à un compte administrateur).
- **Mot de passe** : Leo.16seg (mot de passe correct pour l'administrateur).
- **Attendu** : Après la connexion, la page redirige vers la page d'administration (/admin). La capture d'écran de la page d'administration réussie est incluse.

```
▶ {data: {...}, status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...} ConnectionPage.js:42
admin ConnectionPage.js:43
▶ {token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI...TgwzQ0.eyJ1c2VySWQiOiI...TgwzQ0', role: 'admin', nom: 'Segalini Admin', prenom: 'Léo Admin', ...} ConnectionPage.js:44
```

En fournissant ces jeux d'essai, je démontre le fonctionnement de la page de connexion avec des cas de réussite, des identifiants incorrects et le blocage du compte après 3 tentatives. Les captures d'écran accompagnant ces scénarios illustrent visuellement chaque étape du processus de connexion.

Les points clés

La veille technologique

La veille technologique est essentielle à ma routine quotidienne en tant que développeur web. Chaque jour, je consacre du temps à me tenir informé des dernières évolutions technologiques, des tendances du marché et des nouvelles méthodologies émergentes. La flexibilité de pouvoir choisir le moment idéal pour cette veille quotidienne m'a permis de l'intégrer de manière naturelle dans mon emploi du temps.

Lorsque je me retrouvais confronté à des défis ou des blocages sur des sujets spécifiques, j'utilise ces sessions de veille comme une pause constructive. En prenant du recul et en explorant de nouvelles idées, je trouvais souvent des perspectives nouvelles qui m'aidaient à revenir sur le problème initial avec une approche plus fraîche et créative. Cette démarche contribue non seulement à élargir mes connaissances, mais elle joue également un rôle crucial dans la résolution de problèmes.

La maîtrise de l'anglais a été un atout déterminant dans cette pratique quotidienne, me permettant d'accéder à une richesse de ressources et de documentations disponibles dans cette langue. Cela a grandement enrichi ma compréhension des dernières technologies et des bonnes pratiques dans le domaine du développement web.

En résumé, ma veille technologique quotidienne ne se limite pas à une simple mise à jour des connaissances, mais elle agit comme un catalyseur pour résoudre les obstacles et stimuler ma créativité.

La sécurité

Au cours de mon stage, j'ai acquis une compréhension approfondie de la sécurité dans le développement de sites web. Pour renforcer la protection de notre site, j'ai mis en œuvre des mesures de sécurité robustes, notamment l'intégration d'un middleware performant et l'utilisation de JSON Web Tokens (JWT). Ces initiatives sont conçues pour garantir une authentification sécurisée et une gestion différenciée des accès, en tenant compte des niveaux d'autorisation des utilisateurs.

Le middleware occupe une place cruciale dans notre architecture en interceptant les requêtes HTTP et en appliquant des contrôles de sécurité. Ceci renforce la résilience du site en face de potentielles vulnérabilités, tout en assurant un filtrage efficace des requêtes malveillantes. Cela s'avère essentiel pour contrer des menaces telles que les attaques XSS (Cross-Site Scripting) et les attaques CSRF (Cross-Site Request Forgery), qui pourraient compromettre la sécurité du site.

L'intégration de JSON Web Tokens (JWT) offre une couche supplémentaire de sécurité en générant des tokens cryptographiquement sécurisés. Cela renforce l'authentification et garantit la protection des données sensibles échangées entre le frontend et le backend, limitant ainsi les risques d'exploitation.

En vue d'une sécurité renforcée, je planifie l'implémentation de la double authentification, une étape cruciale qui exigera une validation à deux étapes lors des connexions. Cette mesure vise à fournir une protection accrue contre les accès non autorisés, notamment les attaques DDoS (Distributed Denial of Service) qui peuvent surcharger le site et compromettre sa disponibilité.

Parallèlement, j'ai entrepris de permettre aux utilisateurs de se connecter via des comptes Google et Facebook, tout en garantissant le respect des normes de sécurité rigoureuses, propres à ces services. Cette démarche vise à offrir une expérience utilisateur fluide tout en maintenant des standards de sécurité élevés.

En résumé, la sécurité demeure une préoccupation centrale dans le développement du site, avec une approche proactive visant à contrer les failles potentielles, conformément aux directives OWASP, et à renforcer la résilience du site face aux attaques XSS, CSRF, DDoS, tout en assurant une protection optimale des données sensibles.

Conclusion

La conclusion de mon projet du site web marque une étape significative de mon parcours lors de mon stage intensif. La présence et le soutien de mon maître de stage ont été des éléments clé de cette expérience, apportant guidance et expertise tout au long du processus.

Ce projet, qui représente le point culminant de mon stage, a été une source d'apprentissage considérable. En particulier, il a renforcé mes compétences en JavaScript et en CSS, domaines cruciaux du développement web. La manipulation du code, la création d'interfaces utilisateur dynamiques, et l'optimisation du design ont été des aspects pratiques où j'ai grandement progressé.

Une observation essentielle découlant de ce projet est la compréhension croissante du rôle prépondérant du no-code dans le développement. La nécessité de gagner du

temps et de maximiser l'efficacité devient de plus en plus évidente, et le no-code émerge comme une solution pertinente pour certaines tâches.

Ce projet a également été l'occasion d'améliorer ma méthodologie de veille technologique et de perfectionner mes compétences en recherche. La nécessité de rester à jour avec les dernières technologies et tendances est désormais ancrée dans ma pratique professionnelle.

En somme, ce dernier projet a été une expérience enrichissante qui a façonné ma progression professionnelle. Les compétences techniques, la maîtrise de la veille technologique, et la compréhension des avantages du no code sont autant de points forts qui me positionnent favorablement pour les défis futurs dans le domaine du développement web. Je quitte ce stage avec une expertise accrue et une confiance renouvelée dans ma capacité à aborder des projets de développement web complexes.

Glossaire

SCRUM: SCRUM est un cadre de travail Agile utilisé dans le développement de logiciels. Il favorise la collaboration, la flexibilité et la réactivité aux changements tout au long du processus de développement.

CRUD: CRUD représente les opérations de base pour la manipulation des données dans un système informatique : Create (Créer), Read (Lire), Update (Mettre à jour), et Delete (Supprimer). Ces opérations sont couramment utilisées dans le développement des applications.

NOSQL: NoSQL (Not Only SQL) est une approche de gestion de base de données qui diffère des bases de données relationnelles classiques. Elle permet une flexibilité accrue dans la gestion des données non structurées ou semi-structurées.

FRAMEWORK: Un framework est un ensemble structuré de concepts, de pratiques et de codes prêts à l'emploi qui facilitent le développement d'applications. Il fournit une structure de base pour simplifier la conception et la mise en œuvre de logiciels.

W3C: Le World Wide Web Consortium (W3C) est une organisation internationale qui élabore des normes pour le World Wide Web. Il joue un rôle clé dans le développement de technologies web standardisées pour assurer l'interopérabilité.

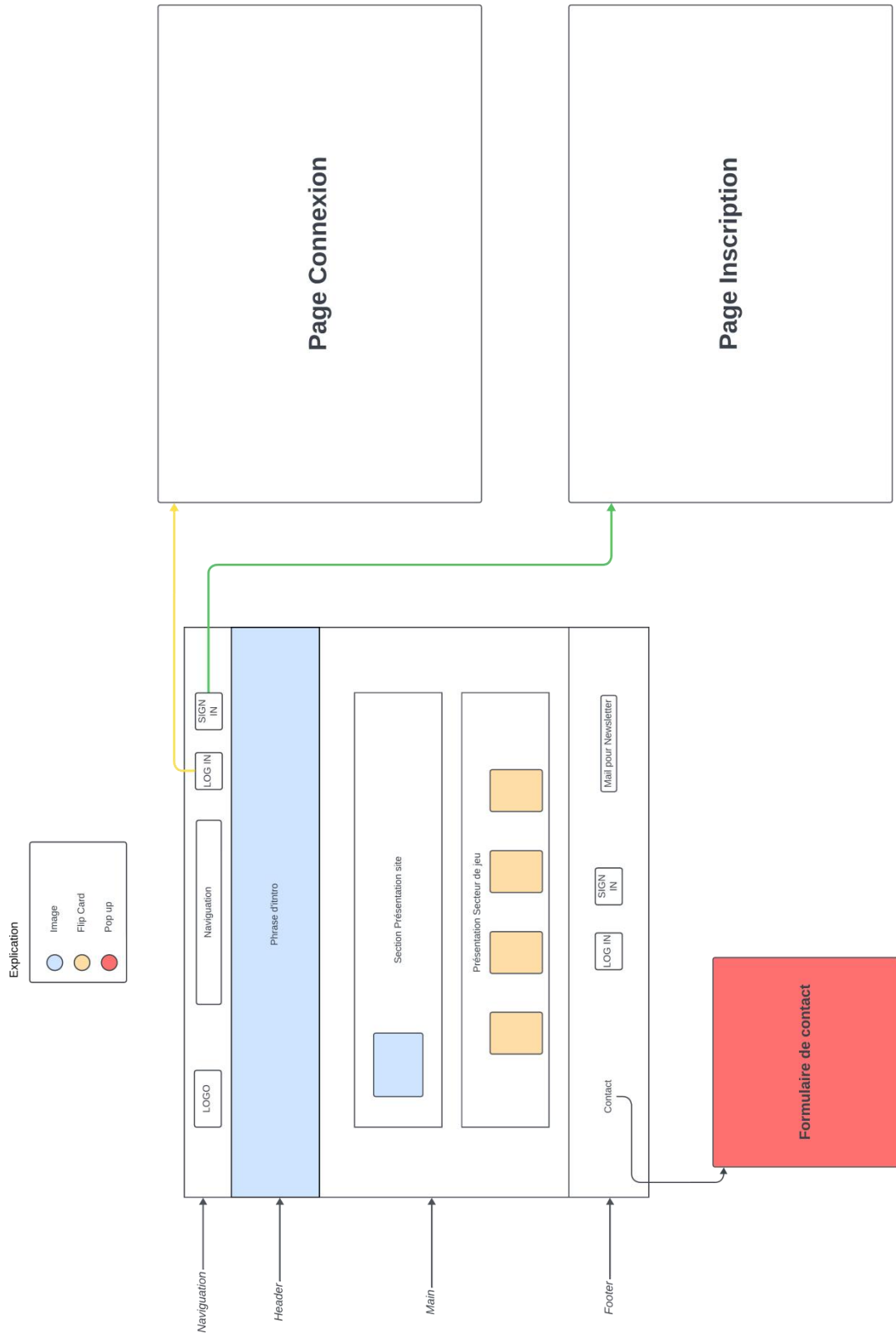
Responsive: Le design web responsive vise à créer des sites web qui s'adaptent dynamiquement à différentes tailles d'écrans, offrant ainsi une expérience utilisateur optimale sur des appareils variés tels que les ordinateurs de bureau, les tablettes et les smartphones.

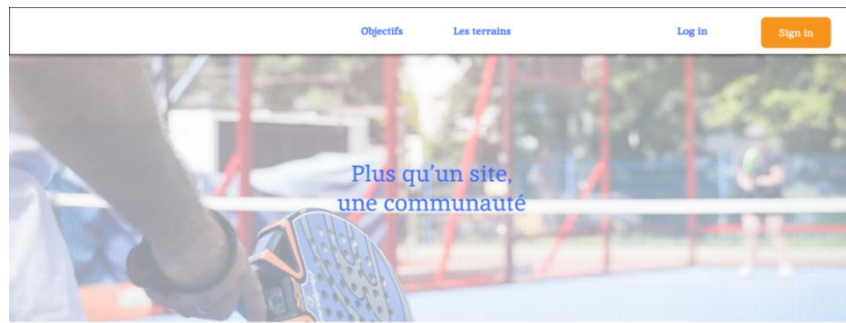
JWT: JSON Web Token (JWT) est un standard ouvert (RFC 7519) qui définit une manière compacte et autonome de représenter de l'information entre deux parties. Il est souvent utilisé pour sécuriser les échanges d'informations entre les parties dans le contexte de l'authentification et de l'autorisation.

API REST: Une API REST (Representational State Transfer) est un style d'architecture logicielle pour la conception de services web. Elle se base sur des principes simples tels que l'utilisation d'URLs comme identifiants de ressources et l'utilisation des méthodes HTTP standard (GET, POST, PUT, DELETE) pour manipuler ces ressources.

Annexes

Annexe 1, Wireframe



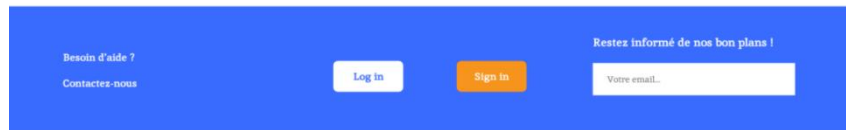


Nos Objectifs



Arrêtez de chercher n'importe où des coéquipiers nous sommes là !
 A travers ce site nous souhaitons créer une communauté autour d'un sport commun le padel. Régulièrement on a du mal à trouver à jours pour faire des parties, ici vous pourrez rencontrer d'autres passionnés et vous réunir autour d'un terrain pour échanger quelques balles !
 Chaque personnes aura un espace personnel dans lequel elle pourra indiquer sa disponibilité, ou trouver un groupe en recherche de partenaire. Vous aurez la possibilité de rechercher les terrains en fonction du secteur ou vous souhaitez jouer. Vous pourrez également mettre en favoris des terrains pour recevoir des mails vous indiquant un groupe en recherche de partenaire dans votre secteur et bien plus !
 Donc n'hésitez plus et **lancez vous** !

Nos Terrains

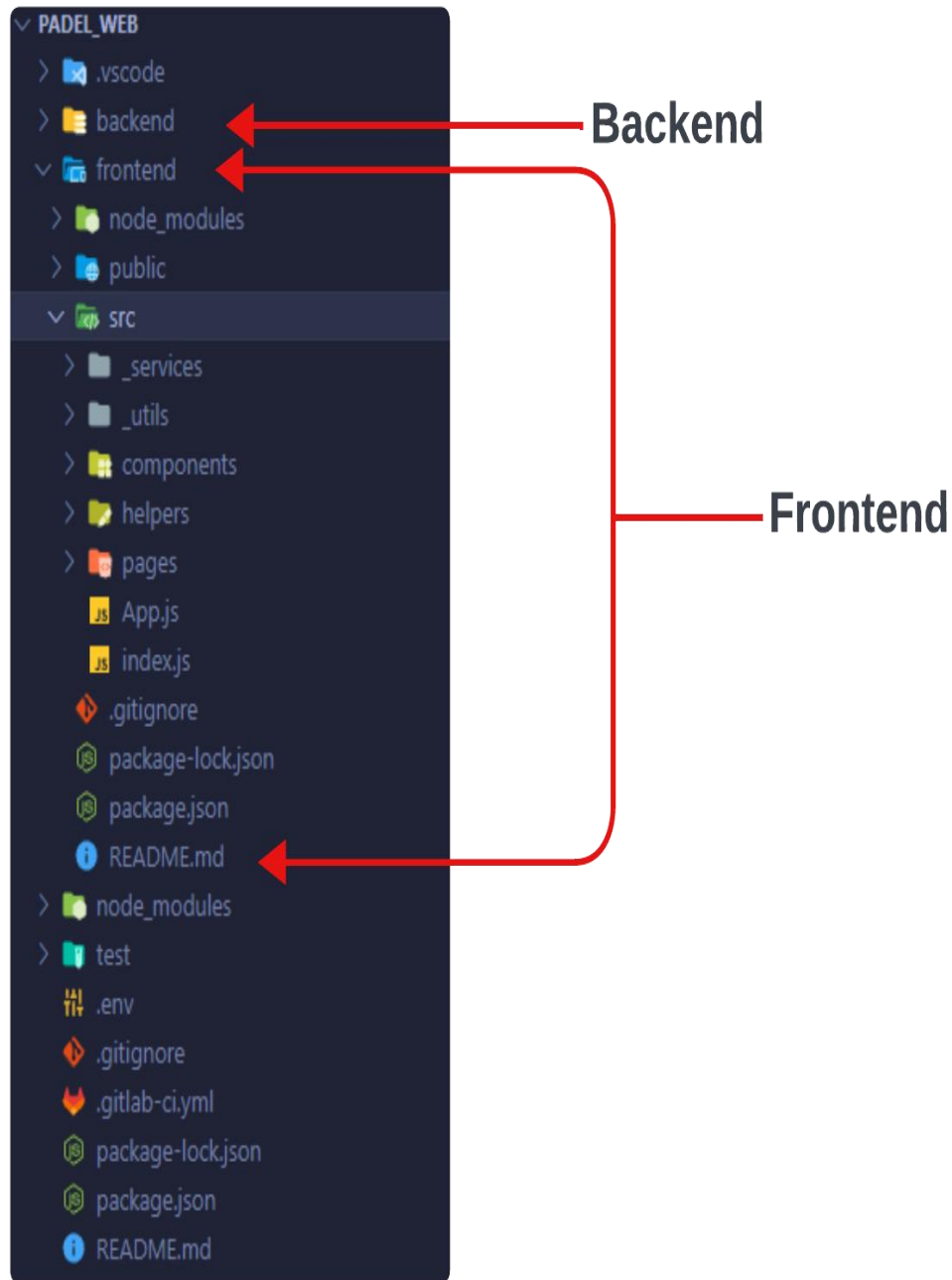


Mentions Légales Données personnelles Gestion cookies

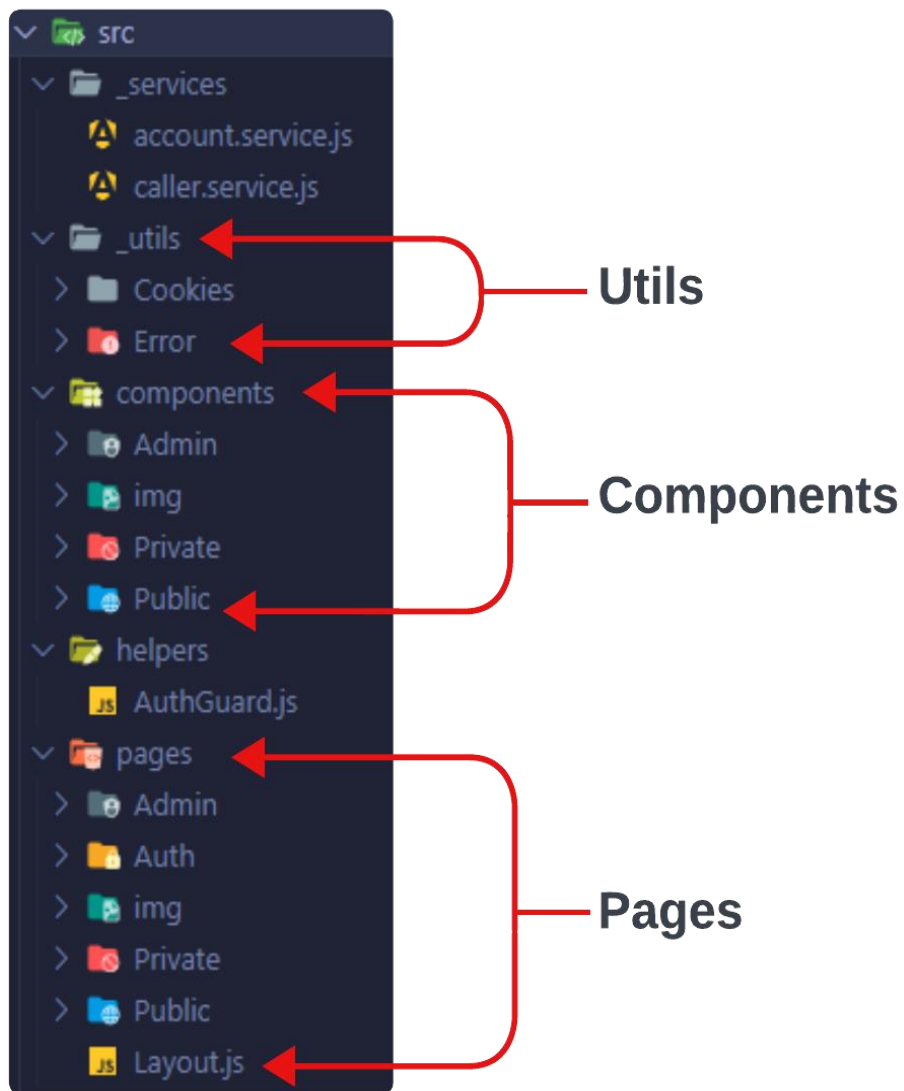
Connexion

[S'inscrire](#)
[Mot de passe oublié](#)

Annexe 3 , Dossier Front end



Annexe 4 , Sous-Dossier Front End

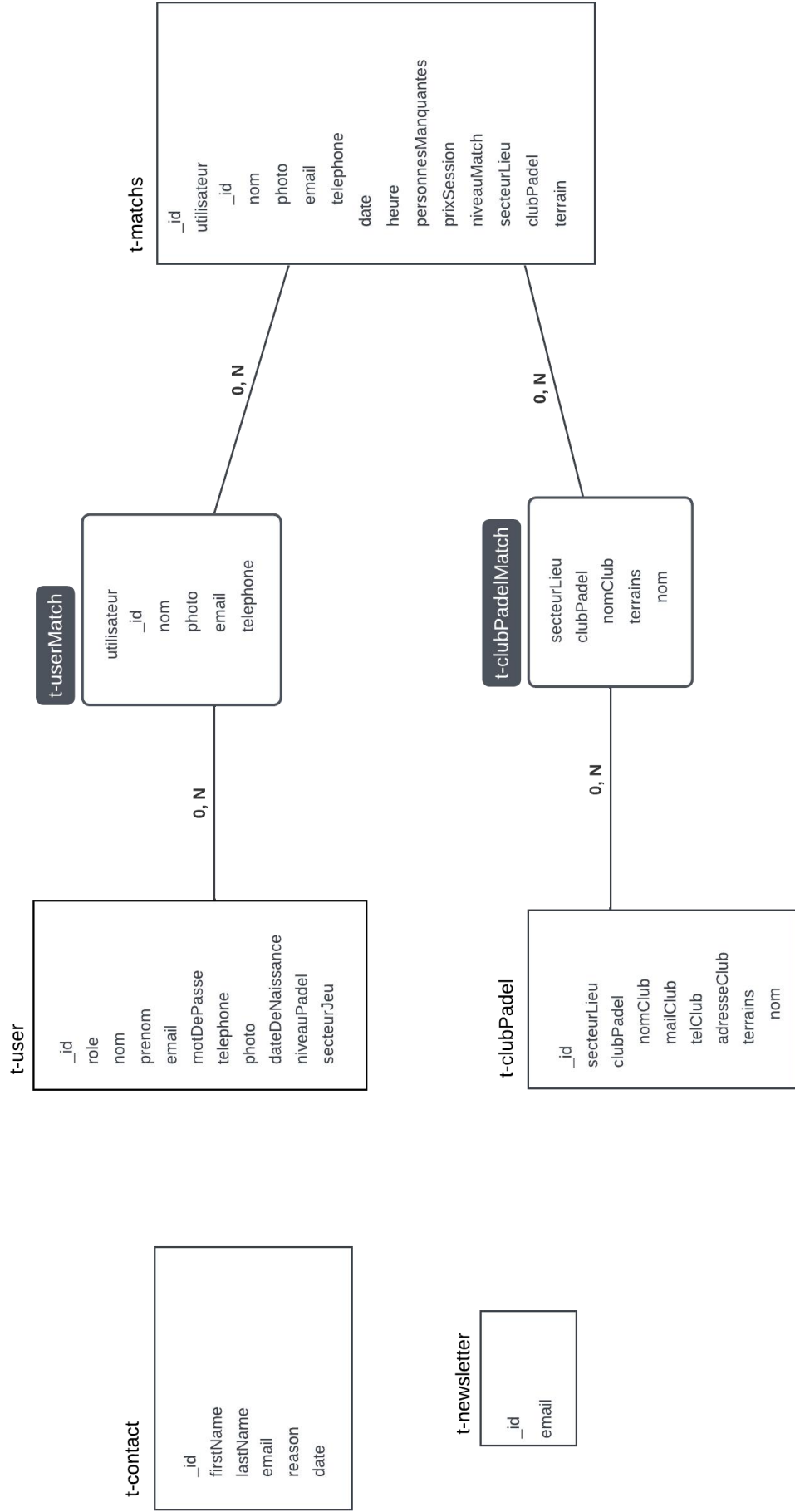


Annexe 5 , Media Queries

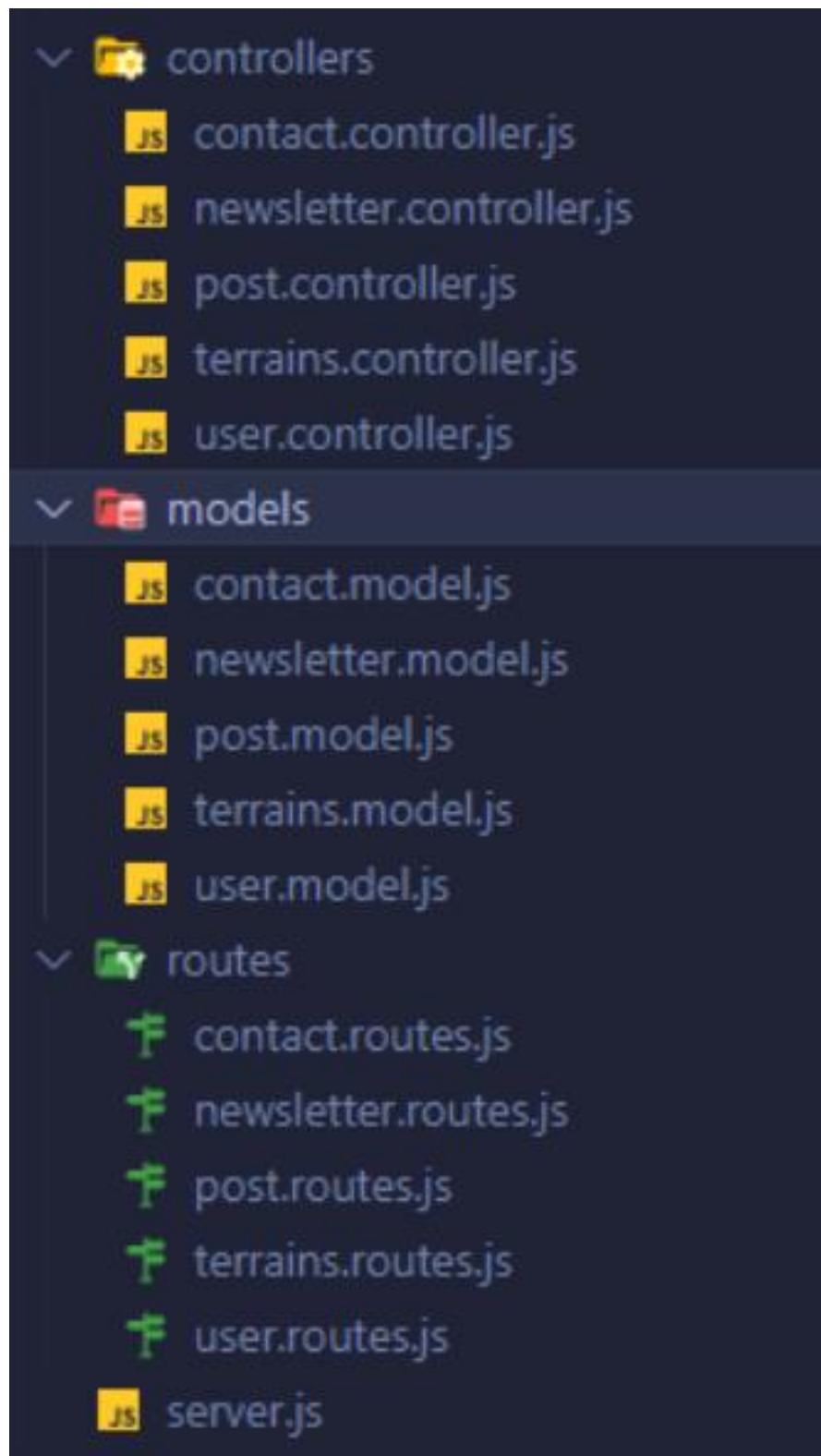
```
@media only screen and (min-width: 768px) and (max-width: 1200px) {  
  
  .objectifs-section,  
  .terrains-section {  
    width: 767px;  
    height: auto;  
  }  
  
  .objectifs-content {  
    flex-direction: column;  
  }  
  
  .objectifs-text {  
    width: 100%;  
  }  
}
```

```
@media only screen and (max-width: 767px) {  
  
  .objectifs-section,  
  .terrains-section {  
    width: 100%;  
    height: auto;  
  }  
  
  .objectifs-content {  
    flex-direction: column;  
  }  
  
  .objectifs-text p {  
    padding: 0 10px;  
    text-align: center;  
  }  
  
  .header-content h1 {  
    font-size: 40px;  
  }  
  
  .objectifs-image {  
    width: 100%;  
    height: auto;  
  }  
  
  .objectifs-image img {  
    height: 100%;  
    width: 250px;  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
    transition: transform 0.3s ease-in-out;  
  }  
}
```

Annexe 6 , MCD



Annexe 7 , Dossier backend



Annexe 8 , db.js

```
const mongoose = require("mongoose");

// Fonction asynchrone pour se connecter à la base de données MongoDB
const connectDB = async () => {
  // Construction de l'URL de la base de données en utilisant les variables d'environnement
  const url = "mongodb+srv://" + process.env.HOST + "/" + process.env.DB_NAME;

  try {
    // Tentative de connexion à la base de données MongoDB
    await mongoose.connect(url, {});

    // Affichage d'un message de réussite si la connexion est établie
    console.log('MongoDB connecté');
  } catch (err) {
    // Affichage de l'erreur en cas d'échec de la connexion
    console.log(err);
  }
};

// Export de la fonction connectDB pour être utilisée dans d'autres fichiers
module.exports = connectDB;
```


Annexe 9 , Media Queries

```
// Middleware pre('save') pour hacher le mot de passe avant de sauvegarder dans la base de données
userSchema.pre("save", async function (next) {
  try {
    console.log("Middleware pre('save') triggered");
    if (this.isModified("motDePasse")) { // You, la semaine dernière • first commit
      // Vérifier si le mot de passe a été modifié
      // Générer un sel et hacher le mot de passe avec 10 tours.
      const salt = await bcrypt.genSalt(10);
      const hashedPassword = await bcrypt.hash(this.motDePasse, salt);
      this.motDePasse = hashedPassword;

      console.log(this.motDePasse);
    }
    next();
  } catch (error) {
    console.error(error);
    next(error);
  }
});
```

Annexe 10 , Postman

GET http://localhost:5000/

POST http://localhost:5000/

POST http://localhost:5000/

+

...

http://localhost:5000/users

GET http://localhost:5000/users

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Body

Cookies

Headers (19)

Test Results

Pretty

Raw

Preview

Visualize

JSON

↗

```
1 {
2   "id": "654e2cfb9b646c246dbb10d2",
3   "role": "admin",
4   "nom": "Segalini Admin",
5   "prenom": "Léo Admin",
6   "email": "leo@iroko.io",
7   "photo": "https://camo.githubusercontent.com/48d099298b4cb2d7937bcd96e8497cf1845b54a819a432c70cf944b60b40c77/
8     68747470733a2f2f7261776769742e6366f6d2f676f72616e67616a69632f72656163742d696366f6e732e737667",
9   "motDePasse": "$2b$10$HpxHdtPnzBqgd1vQ32mUBuACaaYRoo8R7Y708rcoaJl1mq8veQ0zme",
10  "dateDeNaissance": "1995-12-11T00:00:00.000Z",
11  "niveauPadel": 2,
12  "secteurJeu": "Est",
13  "___v": 0,
14  "telephone": "+33670963371"
15 },
16
17 {
18   "id": "6560371b525ab4dd4423d3c0",
19   "role": "user",
20   "nom": "Segalini",
21   "prenom": "Leo",
22   "email": "leo@gmail.com",
23   "telephone": "0670963371",
24   "photo": "http://localhost:3000/static/media/logo_padel-preview.6a02254f71f1014a38fd.webp",
25   "motDePasse": "$2b$10$8XZMW7keYY73JAGnbkE23u5Z4XNfvdmm3mGZGqTemvGAY",
26   "dateDeNaissance": "1995-11-16T00:00:00.000Z",
27   "niveauPadel": 5,
28   "secteurJeu": "Sud",
29   "___v": 0
30 }
```

Annexe 11 , test unitaire

```
process.env.NODE_ENV = "test";

const chai = require("chai");
const chaiHttp = require("chai-http");
const { app } = require("../backend/server");

chai.use(chaiHttp);

describe("User Controller Tests", () => {
  // Testez la route GET /users
  describe("GET /users", (done) => {
    it("should get all users", () => {
      chai.request(app)
        .get("/users")
        .end((err, res) => {
          if(res) res.should.have.status(200);
          done();
        })
    })
  });
});
```

Annexe 12, JWT

```
// Fonction pour authentifier l'utilisateur et générer un token JWT
module.exports.authenticateUser = async (req, res) => {
  const { email, motDePasse } = req.body;

  try {
    // Vérifiez l'e-mail de l'utilisateur et le mot de passe avec la base de données
    const user = await UserModel.findOne({ email });

    if (!user || !(await user.isValidPassword(motDePasse))) {
      return res
        .status(401)
        .json({ message: "Email ou mot de passe incorrect" });
    }

    // Créez un jeton d'authentification avec le rôle inclus dans le payload
    const token = jwt.sign(
      {
        userId: user._id,
        role: user.role,
        nom: user.nom,
        prenom: user.prenom,
        photo: user.photo,
        email: user.email,
        telephone: user.telephone,
      },
      process.env.JWT_SECRET
    );

    res
      .status(200)
      .json({
        token,
        userId: user._id,
        role: user.role,
        nom: user.nom,
        prenom: user.prenom,
        photo: user.photo,
        email: user.email,
        telephone: user.telephone,
      });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Erreur lors de l'authentification" });
  }
};
```

Annexe 13 , fonction connexion

```
// Fonction pour gérer la soumission du formulaire de connexion
const onSubmit = (e) => {
  e.preventDefault();
  // Si le nombre d'échecs de connexion atteint 3, bloquer le compte
  if (loginAttempts >= 3) {
    setError("Compte bloqué. Veuillez contacter l'assistance.");
    return;
  }

  accountService
    .login(credentials)
    .then((res) => {
      console.log(res);
      console.log(res.data.role);
      console.log(res.data);

      localStorage.setItem("user", JSON.stringify(res.data));
      accountService.saveToken(res.data.token);
      if (res.data && res.data.role === "admin") {
        navigate("/admin");
      } else {
        navigate("/homeco");
      }
    })
    .catch((error) => {
      setLoginAttempts(loginAttempts + 1); // Incrémentez le compteur d'échecs de connexion
      setError("Identifiants invalides");
      console.error(error);
    });
};
```


Annexe 14, Node-Cron

```
require('dotenv').config();

const mongoose = require('mongoose');
const moment = require('moment');
const cron = require('node-cron');
const Match = require('../models/post.model');

const url = "mongodb+srv://" + process.env.HOST + "/" + process.env.DB_NAME;

mongoose.connect(url, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Exécutez le nettoyage tous les jours à minuit
cron.schedule('0 0 * * *', async () => {
  try {
    const currentDate = moment().toISOString();

    // Supprime les matchs dont la date est passée
    await Match.deleteMany({ date: { $lt: moment().startOf('day') } });

    console.log('Suppression des matchs passés terminée.');
```

```
  } catch (error) {
    console.error('Erreur lors de la suppression des matchs:', error);
  } finally {
    mongoose.disconnect();
  }
});
```

Bonus

Je voudrais remercier mon maitre de stage Ali qui a été très disponible durant tout le stage. Merci, ton soutien et ton écoute constante ont fait de cette expérience une véritable aventure enrichissante.

Tes conseils éclairés sur le fais qu'il fallait aller boire un café et reprendre le code plus tard, permet effectivement de se détendre.

Et puis, quel bonheur de découvrir le design des sites web de Côte d'Ivoire, leurs styles anciens ont leur charme.

Un grand merci aussi à Pole Emploi pour avoir rendu tout cela possible avec leur soutien financier. Ça a été une sacrée chance de pouvoir suivre cette formation.

Et bien sûr, merci à l'IFR et à nos formateurs géniaux ! Leurs conseils, leurs blagues et leur énergie ont fait de nos journées de formation des moments super sympas.