



**michael-moniteur-cycliste.fr**



Projet réalisé par Amandine MARCHAND,  
dans le cadre de la présentation au Titre Professionnel  
**Développeur(se) Web et Web Mobile**  
à l'AFCI de Liévin (novembre 2024)





## Remerciements

Je tiens tout d'abord à remercier Christophe Le Goff, notre Formateur en Développement Web/Web Mobile et Julien Bianco, le Directeur de l'AFCI Hauts de France, de m'avoir accordé leur confiance pour suivre ce cursus de formation.

Je remercie également Michaël Soetens, mon tuteur, pour sa patience et sa confiance en moi. Etant en reconversion professionnelle, comme lui l'a été, et passionnés de cyclisme tous les deux, cette collaboration a pris une signification particulière pour moi.

Je souhaite aussi remercier Christophe Le Goff, pour sa pédagogie. Grâce à son expertise, nous avons pu bénéficier de solides connaissances et de précieux conseils. J'ai apprécié sa façon de nous former, nous laissant une autonomie nécessaire pour progresser et acquérir une certaine ouverture d'esprit.

Aussi, je veux remercier Mathieu Demont, Assistant de Formation à l'AFCI, pour son accueil souriant chaque matin et sa disponibilité.

Enfin, je remercie mon conjoint, qui m'apporte son soutien chaque jour, depuis le début de ma reconversion.

## Sommaire

Introduction .....	5
Compétences couvertes par ce projet .....	6
Contexte du projet .....	7
La Préparation du Projet.....	9
La Charte Graphique .....	9
Les Maquettes du Projet.....	10
Choix des Technologies.....	12
Les Réalisations .....	12
La Conception du Front-End .....	12
Un Site Dynamique.....	14
Le Formulaire de Contact .....	20
La Base de Données Relationnelle .....	22
L'Espace Administrateur .....	24
La Page Actualités.....	24
•    Créer une nouvelle publication .....	24
•    Voir une publication.....	28
•    Modifier une publication .....	28
•    Supprimer une publication.....	30
Enregistrement d'un nouveau Client ou d'un nouvel Admin.....	31
L'Espace Client.....	33
•    Le Login .....	33
•    La réinitialisation du mot de passe .....	35
•    Présentation de l'Espace Client.....	37
Les Réservations .....	40
•    Côté Admin .....	40
•    Côté Client .....	41
L'Accessibilité .....	45
Le SEO .....	45
Le Déploiement.....	46
•    L'Hébergement.....	46
•    1 <sup>ère</sup> phase : le déploiement du site vitrine .....	46
•    2 <sup>ème</sup> phase : le déploiement de la page Actualités et l'Espace Administrateur .....	46
Conclusion .....	47
Annexes .....	48

## Introduction

Je suis passionnée par l'informatique depuis toute jeune. Mais ce n'est pas dans cette voie que j'ai effectué ma carrière.

J'ai exercé différents métiers dans la grande distribution pendant dix-huit ans : Approvisionneuse, Cheffe de Projets et Responsable Flux Marchandises.

Pour faciliter le travail de mes équipes et améliorer nos résultats, j'ai développé de nombreux outils de travail sur Excel dont certains ont été déployés au niveau national dans l'entreprise. Je réalisais déjà du développement informatique sans le savoir.

Eprouvant une profonde lassitude dans mon ancien domaine d'activité et une volonté de donner un second souffle à ma carrière, j'ai décidé de changer d'orientation. A la suite d'un bilan de compétences et d'une année de réflexion, c'est vers le développement informatique que je me suis dirigée.

Mon objectif étant, à termes, de créer, avec mon conjoint, notre propre entreprise de conception de sites web/web mobile et de montages audiovisuels.

J'ai appris les bases du codage en autodidacte. Désireuse d'avoir une formation complète et officielle, j'ai décidé de suivre celle-ci au centre de Liévin de l'AFCI. Après plusieurs mois d'apprentissage en salle et d'élaboration de petits projets personnels, nous avons à réaliser un projet réel de notre choix.

Et c'est pour un auto-entrepreneur, qui vient de démarrer son activité, que je réalise celui-ci : Michaël, Moniteur Cycliste.

## Compétences couvertes par ce projet

### 1. Développer la partie front-end d'une application web ou web mobile sécurisée

Installer et configurer son environnement de travail en fonction du projet

Maquetter des interfaces utilisateur

Réaliser des interfaces utilisateur statiques

Développer la partie dynamique des interfaces

J'ai développé la partie front-end en Javascript, dans une démarche orientée objet. Le style est rédigé en CSS, avec l'adaptabilité du site à toutes les tailles d'écran. J'ai travaillé avec Docker pour la simulation de l'environnement PHP et Mailhog pour l'envoi de mails (pour le formulaire de contact qui a été déployé en même temps que le projet front), GitHub pour la sauvegarde de mon projet et Figma pour la réalisation de la maquette.



### 2. Développer la partie back-end d'une application web ou web mobile sécurisée

Mettre en place une base de données relationnelle

Développer des composants d'accès aux données SQL

Développer des composants métiers côté serveur

Documenter le déploiement d'une application dynamique

J'ai choisi d'utiliser Symfony et PHP pour le développement back-end, langage et framework robustes permettant de sécuriser plus facilement l'accès aux données et orientés objets. J'ai utilisé Docker, pour la simulation d'envoi de mails avec l'environnement Mailpit et l'environnement MySQL pour la gestion de la base de données. J'ai utilisé DBeaver pour consulter la base de données. J'ai utilisé APIplatform pour communiquer avec le front et testé les routes et accès avec Postman.



## Contexte du projet

### Présentation de "Michaël Moniteur Cycliste"

Michaël Soetens est un ancien ambulancier. Passionné de cyclisme, il s'est reconvertis en Moniteur Cycliste fin 2022/début 2023. Il a créé son auto-entreprise à l'été 2023, à Armentières (59), sous le nom de "Michaël Moniteur Cycliste".

Un Moniteur Cycliste apprend aux individus (enfants et adultes) à rouler et circuler à vélo, se remettre en selle après un long arrêt, ainsi que les règles de sécurité routières à vélo.

Ses clients sont des particuliers de tous âges, des écoles et des associations. Il dispense des séances d'apprentissage individuelles ou collectives, en fonction des besoins des clients.

Pendant les vacances scolaires, il organise des stages vacances, dédiés aux enfants, pour apprendre en groupe, à rouler sans les petites roues ou pour se perfectionner. Il organise également des sorties collectives spéciales comme "parents-enfants" ou encore "remise en selle seniors".

### Besoins de l'entreprise

Lorsque j'ai rencontré Michaël la première fois en mai 2024, il m'a fait part de sa situation. J'ai relevé trois points essentiels :

- **Le manque de clients**

A ce moment là, il utilisait tous ces moyens pour développer sa clientèle :

- Démarchage des entreprises et des associations locales liées au sport,
- Obtention de différentes certifications officielles et partenariats (MCF, SRAV, etc), qui le crédibilisent dans ce domaine d'activités,
- Partenariat avec Decathlon Bailleul, pour lequel il accompagne les groupes participant aux randonnées VTT organisées par le magasin.
- Distribution de sa carte de visite et de son flyer dès qu'il en a l'occasion.
- Activité régulière sur les réseaux sociaux : Facebook, Instagram et plus récemment Tiktok.

- **La gestion laborieuse des réservations pour les activités organisées**

Quand il organise une activité collective, il publie l'information sur les réseaux sociaux. Les personnes intéressées doivent l'appeler ou lui envoyer un message pour réserver. Cela génère beaucoup d'échanges, via différents biais de communication (mail, SMS, appel, whatsapp, etc...)

- **Le paiement de ses prestations pas toujours assuré**

En effet, les paiements se font de la main à main, soit par chèque ou espèces. Parfois le virement est possible selon la clientèle. Dans tous les cas, pas de paiement en carte bancaire possible et aucune assurance que les chèques soient approvisionnés.

Je me suis tout de suite projetée quand il m'a expliqué sa situation et ses problématiques.

J'imaginais déjà un côté "vitrine", pour qu'il puisse présenter son activité, son offre et donner de la visibilité sur internet à son entreprise, et aussi un côté "événements/réservation", lui permettant de gérer ses prestations directement en ligne, avec réservation et paiement via le site.

### **Contraintes et objectifs**

Etant en plein apprentissage des langages "Front" lors de ma rencontre avec Michaël, je me sentais capable de pouvoir lui développer dans un premier temps, un site vitrine pour le démarrage de l'été. En effet, cette période estivale est cruciale pour faire décoller son activité, la météo plus ensoleillée favorisant la pratique des activités extérieures. C'était une opportunité à ne pas négliger pour lui comme pour moi.

J'avais donc comme premier objectif de développer et déployer un site vitrine pour juillet 2024.

Et toute la partie "Back", avec la gestion des événements, réservation en lignes, la publication de son actualité (à la manière d'un blog), serait développée dans un second temps, d'août à octobre 2024, pendant la période de stage officielle consacrée dans notre planning de formation.

### **Environnement humain et technique**

Je travaille seule sur ce projet. Etant autonome et autodidacte par nature, cela me convient, même si cela reste un très bon challenge d'apprentissage.

J'utilise un ordinateur portable, qui me permet de travailler d'où je le souhaite.

La conception et la réalisation du projet étant quand même assez denses, je me suis créé une check-list avec toutes les tâches à réaliser pour obtenir le projet abouti. A chaque fin de semaine, je fais un point sur l'avancée de chaque tâche et planifie les semaines qui suivent en fonction des priorités.

Pendant cette phase de conception, je fais le point une fois par semaine avec Michaël sur l'avancée de mes travaux en m'appuyant sur cette check-list.

## La Préparation du Projet

### La Charte Graphique

L'entreprise de Michaël n'avait pas de véritable identité visuelle, son logo ne reflétait pas son activité. C'était difficile pour moi de me projeter au niveau design en partant de presque rien.

Je me suis donc documentée :

- J'ai recherché des informations sur des sites spécialisés dans l'élaboration d'une identité visuelle et la signification des couleurs (exemple : <https://www.code-couleur.com/>).
- J'ai consulté des sites d'autres Moniteurs Cyclistes, pour m'inspirer de leur conception.
- Je me suis rapprochée d'un Designer pour avoir des conseils et m'aider à concevoir un nouveau logo.

Après la validation du logo par Michaël, j'ai ensuite élaboré la charte graphique, à l'aide d'outils comme les générateurs de palettes de couleurs : <https://coolors.co/> et <https://color.adobe.com/fr/>.

Les couleurs ont été choisies pour leur symbole : le vert, pour la stabilité, le bleu, pour la sécurité et l'orange, pour la communication positive.

Je me suis assurée que le choix des couleurs était compatible avec l'accessibilité utilisateur, grâce à l'outil "Color Contrast Analyzer (CCA)", <https://developer.paciellogroup.com/color-contrast-checker/>.

Les polices de caractères ont été choisies sur Google Fonts.

Le nom et la fonction, présentés en guise de titre sur le site, ont des polices "stylisées".

Le reste du site est rédigé avec la police Verdana, réputée pour sa bonne lisibilité et son accessibilité, ainsi que sa compatibilité avec tous les navigateurs et environnements.

### CHARTE GRAPHIQUE



Polices :

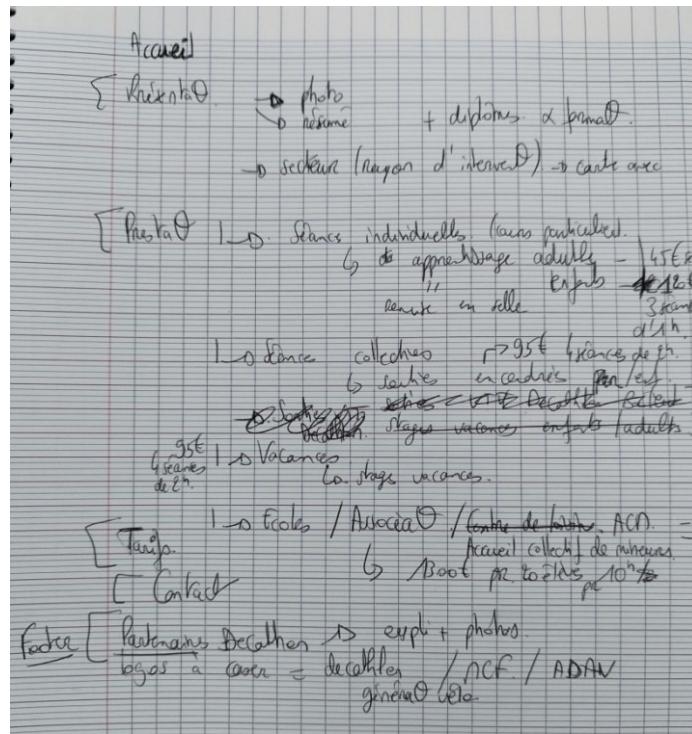
Entête nom en : Sedan

Entête fonction sous titre en : Teachers

Le reste des textes en : Verdana

## Les Maquettes du Projet

Lors de mon premier rendez-vous avec Michaël, j'ai élaboré un plan sommaire à la main du site, tel que nous l'envisagions en fonction de ses besoins.



Après réflexion, le plan élaboré ne semblait pas judicieux d'un point de vue expérience utilisateur. Il fallait aussi penser à la suite avec l'espace client. J'ai donc réalisé un nouveau plan du site :

Niveau 1	Niveau 2	Niveau 3
Accueil		
Qui suis-je ?		
Présentation de toutes les Prestations	Prestations Enfants	Apprentissage Individuel Stage Vacances Sorties groupes parents-enfants
	Prestations Adultes	Apprentissage Individuel Remise en selle Coaching Sorties groupes adultes-seniors
		Prestations Ecoles/Associations/ACM
Réservation	Détails d'un événement	Inscription
Actualités		
Tarifs		
Contact		
Nouveau Client	Enregistrement	
Espace Client	Login/Logout	Mon Compte Mes Réservations

Après validation de la charte graphique et du plan du site par Michaël, j'ai réalisé la maquette du site vitrine sur Figma, en version mobile et en version ordinateur. Exemple ci-dessous :  
*(Maquette complète en Annexe 1)*

The image shows a comparison of Michael Soetens' website design between desktop and mobile versions. A blue arrow points from the desktop view on the left to the mobile view on the right.

**Desktop Version (Left):**

- Header:** Features a circular logo with a bicycle icon and the text "MICHAËL SOETENS" and "Moniteur Cycliste dans le Nord-Pas-de-Calais". Social media icons for Facebook, Instagram, and YouTube are also present.
- Navigation Bar:** Includes links for Accueil, Qui suis-je ?, Pour les Enfants et les Parents, Pour les Adultes, Pour les Écoles/ ACM/Associations, Mon Actualité, Réservations, Tarifs, and Contact.
- Content Area:**
  - A welcome message from Michael: "Bienvenue, je suis Michaël Soetens, Moniteur Cycliste dans le Nord-Pas-de-Calais, spécialisé dans l'apprentissage tout public."
  - Two images: one of Michael wearing a helmet and another of him with children on bicycles.
  - A section titled "Vous êtes..." with three options: "Un adulte", "Un parent", and "Une école ? Une association ? Un Accueil Collectif de Mineurs ?". Each option has a corresponding image and a large orange arrow pointing to the right.
  - A section titled "Pourquoi faire appel à moi ?" with a large orange arrow pointing to the right.
  - Four main service sections with images and descriptions:
    - POUR APPRENDRE À ROULER:** "Apprendre à s'élançer, tenir l'équilibre, freiner et savoir se diriger à vélo. Prendre confiance sur son vélo et adopter les bons réflexes, autour de jeux ludiques que je mets en place." An image shows children on bicycles.
    - POUR SE REMETTRE EN SELLE:** "Vous vous remettez au vélo après un long arrêt ? Je vous accompagne avec un programme personnalisé et adapté en fonction de votre situation." An image shows two people riding bicycles on a path.
    - POUR UN COACHING:** "Vous souhaitez être formé ou coaché sur une thématique particulière ? Je suis à votre écoute pour vous accompagner avec un programme individuel." An image shows a person riding a bicycle.
    - POUR UN PROJET PÉDAGOGIQUE:** "J'interviens dans les écoles, associations et accueils collectifs de mineurs, pour apprendre aux enfants et adolescents à rouler à vélo. Je les forme également sur la sécurité à vélo." An image shows a group of children on bicycles.

**Mobile Version (Right):**

- Header:** Shows Michael's profile picture and social media links.
- Content Area:**
  - A welcome message: "Bienvenue, je suis MICHAËL SOETENS Moniteur Cycliste, dans le Nord-Pas-de-Calais spécialisé dans l'apprentissage tout public."
  - Two images: one of Michael and one of him with children on bicycles.
  - A section titled "Vous êtes..." with three options: "Un adulte", "Un parent", and "Une école ? Une association ? Un Accueil Collectif de Mineurs ?". Each option has a corresponding image and a small orange arrow pointing down.
  - A section titled "Pourquoi faire appel à moi ?" with a small orange arrow pointing down.
  - Four main service sections with images and descriptions:
    - POUR APPRENDRE À ROULER:** "Apprendre à s'élançer, tenir l'équilibre, freiner et savoir se diriger à vélo. Prendre confiance sur son vélo et adopter les bons réflexes, autour de jeux ludiques que je mets en place." An image shows children on bicycles.
    - POUR SE REMETTRE EN SELLE:** "Vous vous remettez au vélo après un long arrêt ? Je vous accompagne avec un programme personnalisé et adapté en fonction de votre situation." An image shows two people riding bicycles on a path.
    - POUR UN COACHING:** "Vous souhaitez être formé ou coaché sur une thématique particulière ? Je suis à votre écoute pour vous accompagner avec un programme individuel." An image shows a person riding a bicycle.
    - POUR UN PROJET PÉDAGOGIQUE:** "J'interviens dans les écoles, associations et accueils collectifs de mineurs, pour apprendre aux enfants et adolescents à rouler à vélo. Je les forme également sur la sécurité à vélo." An image shows a group of children on bicycles.

## Choix des Technologies

---

Avant d'entamer la formation, je n'avais réalisé que quelques projets en HTML et CSS. Tout le reste était donc à découvrir !

Je me suis renseignée sur les différents langages, ai observé quelques codes, testé en minis-projets, en cours avec le formateur, ou en aparté, avant de me décider.

Curieuse de tester les diverses façons de développer pour m'en faire mon propre avis et me bâtrir une expérience, j'ai décidé de coder avec deux méthodes différentes.

- Le front serait codé en Javascript et CSS natifs. Javascript m'avait attiré par son côté "gestion dynamique" et le fait de pouvoir utiliser un même élément (une fonction par exemple) dans différentes pages.
- Le back serait codé à l'aide d'un framework : Symfony, qui me plaisait pour sa gestion en "modèle-vue-contrôleur" et ses multiples fonctionnalités à imbriquer.

## Les Réalisations

### La Conception du Front-End

---

#### Une Programmation Orientée Objets

J'avais en tête de ne pas coder directement en HTML, mais de réaliser une conception assez dynamique, sous forme de sections HTML vides alimentées par des fonctionnalités en Javascript. Le fait de pouvoir construire des blocs de données, utilisables à plusieurs endroits si besoin, me fascinait.

Cette conception que j'avais mise en place assez spontanément, était "orientée objet".

#### Structure du front-end

L'idée était de regrouper les éléments similaires ou repris à plusieurs reprises (objets, fonctions, feuilles de style, etc.) dans des fichiers communs à toutes les pages, puis de créer des fichiers exclusifs à chaque page pour leurs éléments spécifiques.

Cela concernait les 3 blocs d'éléments identiques à tous : le <header>, la <nav> et le <footer>. J'ai créé un fichier "structure.js" dans lequel j'ai codé le contenu à injecter dans ces 3 parties.

Une feuille de style CSS style.css unique et commune, gère la mise en page de ces 3 sections.

J'ai procédé de la même façon avec les fonctions et les constantes qui allaient servir sur tout le site. J'ai réuni celles-ci dans un fichier commun à tout le projet : fonctions.js.

*Exemple de rattachement de scripts et de feuilles de style pour la page Tarifs :*

<link rel="stylesheet" href=" = SITE_URL_STYLES . 'style.css'?&gt;"&gt;</td <td>⇒ Commun à tous</td>	⇒ Commun à tous
<link rel="stylesheet" href=" = SITE_URL_STYLES . 'tarifs.css'?&gt;"&gt;</td <td>⇒ Spécifique à la page Tarifs</td>	⇒ Spécifique à la page Tarifs
<script src=" = SITE_URL . 'structure.js'?&gt;" type="module" defer&gt;&lt;/script&gt;</td <td>⇒ Commun</td>	⇒ Commun
<script src=" = SITE_URL . 'fonctions.js'?&gt;" type="module" defer&gt;&lt;/script&gt;</td <td>⇒ Commun</td>	⇒ Commun
<script src=" = SITE_URL . 'tarifs.js'?&gt;" type="module" defer&gt;&lt;/script&gt;</td <td>⇒ Spécifique à la page Tarifs</td>	⇒ Spécifique à la page Tarifs

*Exemple de la création de l'objet Facebook (logo + lien), qui sera utilisé dans une <div> du <header> et dans une <div> du <footer> :*

```
let reseauxSociaux = {
  class: 'divEnteteFbInsta',
  id: 'reseauxSociaux',
}

let facebook = {
  class: 'logoFacebook',
  href: 'https://www.facebook.com/profile.php?id=100092366783727',
  target: '_blank',
  alt: "Logo de Facebook",
  ariaLabel: "Page Facebook de Michaël Moniteur Cycliste",
  style: `background-image: url(${URLImages}logo_facebook.png)`,
}

creationElement(reseauxSociaux, 'div', header);
const divEnteteFbInsta = document.querySelector('.divEnteteFbInsta');
creationElement(facebook, 'a', divEnteteFbInsta);

creationElement(reseauxSociauxFooter, 'div', footer);
const divReseauxSociauxFooter = document.querySelector('.divReseauxSociauxFooter');
creationElement(logosReseauxSociaux, 'div', divReseauxSociauxFooter);
const divLogosReseauxSociaux = document.querySelector ('.divLogosReseauxSociaux');
creationElement(facebook, 'a', divLogosReseauxSociaux);
```

*Exemple de la fonction qui servira à créer tous les éléments HTML (utilisée dans l'exemple ci-dessus) avec rattachement de chaque attribut et de contenu de texte s'il y a :*

```
// fonction pour créer des éléments HTML, avec attributs et propriétés //
export function creationElement(element, type, parent) {
  const contenantElement = document.createElement(type);
  contenantElement.textContent = element.textContent;
  Object.keys(element).forEach(key => {
    contenantElement.setAttribute(key, element[key]);
    contenantElement.removeAttribute('textContent');
  });
  parent.append(contenantElement);
};
```

Ainsi, tous les éléments de front-end ont été créés, tout en respectant le principe des balises HTML. Exemple de code généré sur la page d'accueil :

```
▼<header id="header" class="classHeader" style="background-image: url("https://www.michael-moniteur-cycliste.fr/assets/images/header_fond.webp");"> flex [débordement]
  <div id="logoMC" class="divLogo" alt="logo de Michaël Moniteur Cycliste" title="Michaël Soetens, Moniteur Cycliste" style="background-image: url("https://www.michael-moniteur-cycliste.fr/assets/images/logo_michael_soetens.png");"></div>
  ▼<div id="nomFonction" class="divEnteteNom">
    <h1 id="enteteNom">Michaël SOETENS</h1>
    <h2 id="enteteFonction">Moniteur Cycliste dans le Nord-Pas-de-Calais</h2>
  </div>
  ▼<div id="reseauxSociaux" class="divEnteteFbInsta" style="flex: 1; border-right: 1px solid #ccc; padding-right: 10px; margin-right: 10px; border-bottom: 1px solid #ccc; padding-bottom: 10px; margin-bottom: 10px; position: relative; z-index: 1; background-color: white; border-radius: 5px; height: 100px; width: 100px; display: flex; align-items: center; justify-content: center; gap: 10px; font-size: 0.8em; font-weight: bold; color: black; text-decoration: none; transition: all 0.3s ease; ">
    <a class="logoFacebook" href="https://www.facebook.com/profile.php?id=100092366783727" target="_blank" alt="Logo de Facebook" ariaLabel="Page Facebook de Michaël Moniteur Cycliste" style="background-image: url(https://www.michael-moniteur-cycliste.fr/assets/images/logo_facebook.png)"></a>
    <a class="logoInstagram" href="https://www.instagram.com/michael_moniteur_cycliste?igsh=ZDJ3dmc2ZjZoaTl4" target="_blank" alt="Logo d'Instagram" ariaLabel="Page Instagram de Michaël Moniteur Cycliste" style="background-image: url(https://www.michael-moniteur-cycliste.fr/assets/images/logo_instagram.png)"></a>
    <a class="logoTiktok" href="https://www.tiktok.com/@michael.moniteur?is_from_webapp=1&sender_device=pc" target="_blank" alt="Logo de TikTok" ariaLabel="Page TikTok de Michaël Moniteur Cycliste" style="background-image: url(https://www.michael-moniteur-cycliste.fr/assets/images/logo_tiktok.png)"></a>
  </div>
  ▶<div id="enteteHamburger" class="divEnteteHamburger"> ... </div>
</header>
▼<nav id="navBar" class="navBar open">
  ▼<ul class="menu"> flex [débordement]
    ▶<li class="menu0"> ... </li> event
    ▶<li class="menu1"> ... </li> event
    ▶<li class="menu2 open"> ... </li> event
    ▶<li class="menu3"> ... </li> event
    ▶<li class="menu4"> ... </li> event
  </ul>
</nav>
```

# Un Site Dynamique

## Manipulation du DOM

La manipulation du DOM est omniprésente dans mon projet, suite à mon choix de conception. Cela vaut aussi bien pour les parties statiques que pour certaines parties dynamiques.

C'est le cas par exemple, sur la page d'accueil, de photos qui défilent à intervalle régulier :



Ces photos sont stockées dans le back, en base de données. On les récupère en front via un fetch, pour les stocker dans un tableau. La fonction changePhoto s'exécute toutes les 5 secondes. Elle permet de passer à l'index suivant (+1), du tableau dans lequel sont stockées les photos :

```
let tabPhoto = [];
let currentIndex = 1;

fetch('https://127.0.0.1:8000/api/carrousel') //MODE DEV
  .then(response => response.json())
  .then(data => {
    data.forEach(item => {
      tabPhoto.push({
        nomFichier: item.nomFichier,
        texteAlt: item.texteAlt
      });
    });
    if (tabPhoto.length > 0) {
      displayImage(0);
      divLoading.style.display = 'none';
    }
    setInterval(changePhoto, 5000);
  })
  .catch(error => {
    console.error('Erreur lors de la récupération des images :', error);
    divLoading.style.display = 'none';
  });
}

function changePhoto() {
  if (tabPhoto.length === 0) return;
  divPhotosSeances.classList.toggle('apparition');
  // Affiche l'image à l'index actuel
  displayImage(currentIndex);
  // Boucle à travers les images
  currentIndex = (currentIndex + 1) % tabPhoto.length;
}
```

Certains attributs sont affectés à la div où va apparaître la photo :

```
function displayImage(index) {
  const photo = tabPhoto[index];
  divPhotosSeances.setAttribute('style', `background-image: url(${photo.nomFichier});`);
  divPhotosSeances.setAttribute('name', photo.nomFichier);
  divPhotosSeances.setAttribute('alt', photo.texteAlt);
}
```

En s'exécutant toutes les 5 secondes, "changePhoto" permet de changer le lien URL de l'image d'arrière plan de la <div>. Au démarrage, on commence par la photo 0, puis on enchaîne avec setInterval, qui démarre avec l'index 1 pour ne pas afficher deux fois la photo à l'index 0. Arrivés à la dernière photo, on repasse à la première.

*Exemple d'exécution, avec la photo carrousel4.webp (index 3 du tableau) :*

```
► <div class="photosSeances apparition" style="background-image: url(https://127.0.0.1:8000/uploads/photos/carrousel4-6706a2951866b.webp)" name="carrousel4-6706a2951866b.webp" alt="Pêle-mêle de photos montrant des enfants en train de s'élanter sur leurs vélos.">...</div> flex
```

*... qui s'enchaîne après 5 secondes, avec la photo carrousel5.webp (index 4 du tableau) :*

```
► <div class="photosSeances" style="background-image: url(https://127.0.0.1:8000/uploads/photos/carrousel5-6706a2cb32338.webp)" name="carrousel5-6706a2cb32338.webp" alt="Photo de 2 enfants en train de franchir des obstacles de plots.">...</div> flex
```

J'ai ajouté quelques effets de style (opacité, zoom) pour l'arrivée et le départ des photos :

```
@keyframes apparition{
  from{
    transform: scale(95%);
    opacity: 0.7;
  }
  to{
    transform: scale(100%);
    opacity: 1;
  }
}
.photosSeances.apparition{
  animation: apparition 2s ease-out;
}

@keyframes disparition{
  from{
    transform: scale(100%);
    opacity: 1;
  }
  to{
    transform: scale(95%);
    opacity: 0.7;
  }
}
.photosSeances{
  animation: disparition 2s ease-out;
  animation-delay: 3s;
```

## Une écoute active

Beaucoup de "listeners" ont été posés à travers tout le site, de sorte à lancer certaines actions selon un événement précis ou au chargement du DOM.

Par exemple, le visiteur du site doit accepter la Politique de Confidentialité. Une bannière, dédié à cela, apparaît, et une écoute au click sur "Accepter" est posée. En l'acceptant, un "true" se stocke dans la sessionstorage. Une écoute au chargement de chaque page sera faite. La bannière disparaît dès qu'il accepte.

Dans le cas de la page de login, les champs de saisie (input) sont désactivés tant que le client n'a pas accepté les conditions :

```
window.addEventListener('DOMContentLoaded', function() {
  if (sessionStorage.getItem('privacyAccepted') !== 'true') {
    document.getElementById('privacy-banner').classList.remove('hidden');
    document.querySelectorAll('input').forEach(input => input.disabled = true);
  } else {
    document.getElementById('privacy-banner').classList.add('hidden');
  }

  document.getElementById('banniere-accept').addEventListener('click', function() {
    sessionStorage.setItem('privacyAccepted', 'true');
    document.getElementById('privacy-banner').classList.add('hidden');
    document.querySelectorAll('input').forEach(input => input.disabled = false);
  });
});
```

Pour les pages en front, faisant appel à la base de données, j'ai mis en place des "loaders" pendant le temps de chargement des données, combinant des systèmes d'écoute et mises en forme CSS. Voici un exemple, qui représente une roue de vélo :

```
document.addEventListener('DOMContentLoaded', () => {
  const loader = document.getElementById("loader");
  const prestasIndiv = document.querySelector('#individuel-container');
  loader.style.display = 'flex';

  fetch('https://127.0.0.1:8000/api/evenements') // MODE DEV
    .then(response => response.json())
    .then(data => {
      const evenements = data['hydra:member'];
      const filteredSortedEvents = filterAndSortEvents(evenements);
      return displayEvents(filteredSortedEvents);
    })
    .then(() => {
      loader.style.display = 'none';
      prestasIndiv.style.display = 'flex';
    })
    .catch(error => {
      console.error('Erreur:', error);
      loader.style.display = 'none';
    });
});
```

```
/* Mise en forme du loader */
#loader {
  position: fixed;
  left: 0;
  top: 0;
  width: 100%;
  height: 100vh;
  background: #rgba(0, 0, 0, 0.7);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 9999;
}

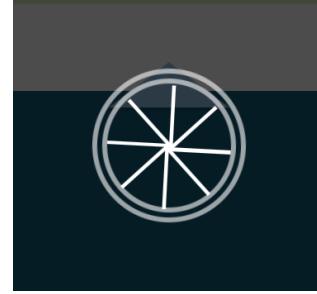
/* Style de la roue */
.roue {
  position: relative;
  width: 100px;
  height: 100px;
  border: 12px double #rgba(255, 255, 255, 0.6);
  border-radius: 50%;
  animation: spin 1s linear infinite;
```

```

/* Style des rayons de la roue */
.rayon {
    position: absolute;
    width: 3px;
    height: 50px;
    background: #fffff;
    top: 50%;
    left: 50%;
    transform-origin: top;
    transform: translate(-50%, 0);
}

/* Position des rayons */
.rayon:nth-child(1) { transform: rotate(0deg) translate(-50%, 0); }
.rayon:nth-child(2) { transform: rotate(45deg) translate(-50%, 0); }
.rayon:nth-child(3) { transform: rotate(90deg) translate(-50%, 0); }
.rayon:nth-child(4) { transform: rotate(135deg) translate(-50%, 0); }
.rayon:nth-child(5) { transform: rotate(180deg) translate(-50%, 0); }
.rayon:nth-child(6) { transform: rotate(225deg) translate(-50%, 0); }
.rayon:nth-child(7) { transform: rotate(270deg) translate(-50%, 0); }
.rayon:nth-child(8) { transform: rotate(315deg) translate(-50%, 0); }

```



```

/* Animation de rotation */
@keyframes spin {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}

```

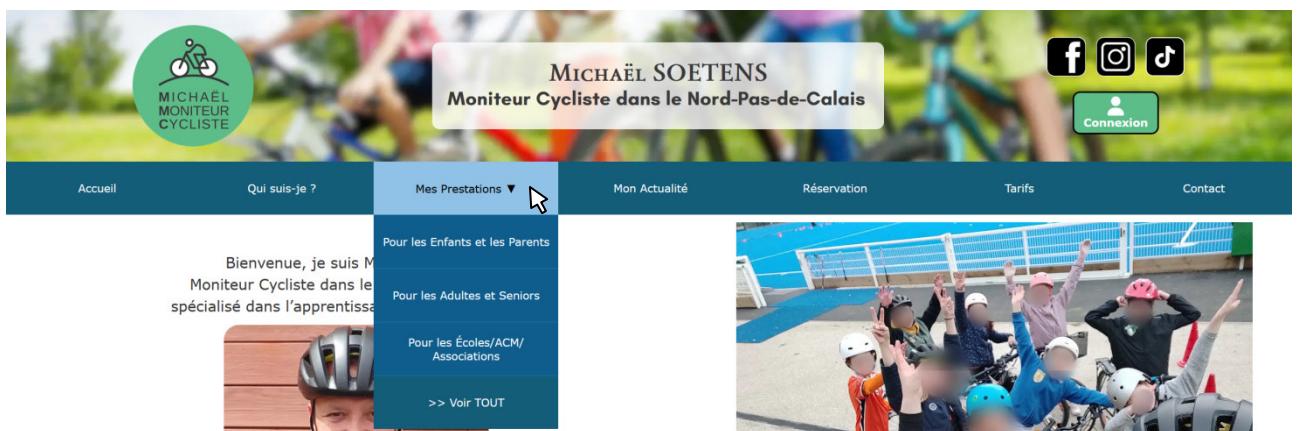
## Un site "Responsive" : adaptabilité aux différentes tailles d'écran

Avec près de 70% de trafic web s'effectuant via un mobile, avoir un site Responsive est incontournable. Sur le site de Michaël, actuellement, nous sommes à 56% de mobile.

J'ai utilisé les Media Queries CSS pour chaque page, en complément d'autres fonctionnalités. Je n'ai pas fixé de breakpoints correspondant aux standards (360px mobile/768px tablette/1920px ordinateur), car cela ne couvrait pas correctement les tailles intermédiaires. J'ai ajusté à chaque fois que cela était nécessaire, en couvrant l'ensemble des réajustements de 360px à 1920px. Cette tâche est longue et fastidieuse, mais elle a au le mérite d'être précise et adaptée à tous les écrans.

### Exemple d'adaptation mobile avec l'entête et la barre de navigation :

Au-delà de 900px, l'entête présente une photo de fond, le nom et le métier de Michaël. La barre de navigation fait apparaître les titres des menus. Le sous-menu se déplie au survol de la souris :



```

.sousMenu{
    display: none;
    position: absolute;
    width:100%;

.menu li:hover .sousMenu{
    display: block;
}

```

Mise en forme CSS pour les sous-menus qui se déroulent, en jouant sur le display: block (apparition) / display: none (disparition), au survol du menu.

En dessous de 900px, l'image de fond disparaît au profit d'une couleur unie, la barre de navigation devient un menu "hamburger" dont le logo se transforme au clic, et le menu se déroule. Les sous-menus se déroulent au clic :

Le menu hamburger était caché sur les grands écrans avec un Media Screen CSS :

```
@media screen and (min-width:900px){
    .divEnteteHamburger{
        display: none;
    }
}
```

Sous les 900px, il apparaît, toujours grâce à un Media Screen, avec toutes ses particularités de mise en forme :

```
@media screen and (max-width: 900px){
    .divEnteteHamburger{
        display: block;
    }
    #iconeHamburger span span{
        display: block;
        width:40px;
        height:5px;
        background-color:#white;
        margin:5px;
        position: relative;
    }
    #iconeHamburger.button.open span:nth-child(2){
        width:0px;
        height:0px;
    }
    #iconeHamburger.button.open span:first-child{
        transform: translateY(5px) rotate(45deg);
        transition: 1s ease;
    }
    #iconeHamburger.button.open span:nth-child(3){
        transform: translateY(-5px) rotate(-45deg);
        transition: 1s ease;
    }
    #iconeHamburger.button.open span span{
        background-color: #white;
    }
}

#navBar.navBar .menu{
    box-shadow: none;
    flex-direction: column;
    justify-content: flex-start;
    transition: 2s ease;
    width: 0%;
    position: absolute;
    right:-10%;
    visibility:hidden;
}
#navBar.navBar .menu a{
    white-space: nowrap;
}
#navBar.navBar.open .menu{
    width: 100%;
    height:60px;
    right:0%;
    transition: 2s ease;
    position: absolute;
    visibility:visible;
}
#iconeHamburger.button span:first-child,
#iconeHamburger.button span:nth-child(3){
    transition: 1s ease;
}
#iconeHamburger.button span:nth-child(2){
    transition:0.5s ease-in-out;
}
```

```

.navBar ul li {
    border-bottom : 1px solid #rgb(67, 142, 200);
}
.navBar ul li ul li {
    background-color: #0E5D8D;
}
.menu li:hover .sousMenu{
    display: none;
}
.menu2 > a::after{
    content: ' ▾';
    font-size: 15px;
}
.menu2.open > a::after{
    content: ' ▼';
    font-size: 15px;
    transition: 0.5s ease;
}

.navBar.navBar.open li.open ul.sousMenu{
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    width:100%;
    height:100%;
    position:static;
    transition: 1s ease;
}
.navBar.navBar.open li ul.sousMenu{
    height:0px;
    transition: 1s ease;
}

```

Cela comprend :

- La mise en forme (couleurs, tailles) de tous les éléments,
- La visibilité et la position des éléments : display (bloc/flex/none), position (static/relative),
- L'ouverture des sous-menus au clic avec changement de la flèche en bout de menu (vers la droite quand le sous-menu est fermé à vers le bas quand il est ouvert),
- La transformation du bouton du menu qui passe de la forme "hamburger" à celle de "croix". Cela est accompagné d'une fonction JS qui écoute le clic sur le bouton et attribue une classe toggle : open ou pas. CSS applique des mises en forme en fonction de la class :

```

blocIcôneHamburger.addEventListener('click', () => {
    navBar.classList.toggle('open');
    blocIcôneHamburger.classList.toggle('open');
});

const ecouteMenu = document.querySelectorAll('.menu > li');
const ecouteSousMenu = document.querySelectorAll('.menu > li > ul > li');

for (let i = 0; i < ecouteMenu.length; i++) {
    ecouteMenu[i].addEventListener('click', (e) => {
        ecouteMenu[i].classList.toggle('open');
    });
}

for (let i = 0; i < ecouteSousMenu.length; i++) {
    ecouteSousMenu[i].addEventListener('click', (e) => {
        e.stopPropagation();
    });
}

```

## Grid ou Flex ? Les deux !

### Mise en page avec Grid

La mise en page en Grid permet d'emboiter les éléments dans des "cases" virtuelles. Je m'en suis servie notamment pour la mise en forme des éléments ci-après : 4 blocs de données créés de la même façon avec une fonction JS (fonction ci-après et premier objet en exemple ci-dessous) :

**POUR APPRENDRE A ROULER**  
Apprendre à s'élançer, tenir l'équilibre, freiner et savoir se diriger à vélo. Prendre confiance sur son vélo et adopter les bons réflexes, autour de jeux ludiques que je mets en place.

[En savoir plus pour : un enfant /un adulte](#)

**POUR SE REMETTRE EN SELLE**  
Vous vous remettez au vélo après un long arrêt ? Je vous accompagne avec un programme personnalisé et adapté en fonction de votre situation.

[En savoir plus](#)

**POUR UN COACHING**  
Vous souhaitez être formé ou coaché sur une thématique particulière ? Je suis à votre écoute pour vous accompagner avec un programme individuel.

[En savoir plus](#)

**POUR UN PROJET PEDAGOGIQUE**  
J'interviens dans les écoles, associations et accueils collectifs de mineurs, pour apprendre aux enfants et adolescents à rouler à vélo. Je les forme également sur la sécurité à vélo.

[En savoir plus](#)

```

let role1 = {
    class: 'role1',
    titre: 'POUR APPRENDRE A ROULER',
    explications: 'Apprendre à s'élançer, tenir l'équilibre, freiner et savoir se diriger à vélo. Prendre confiance sur son vélo et adopter les bons réflexes, autour de jeux ludiques que je mets en place.',
    introlien: 'En savoir plus pour : ',
    lien1: 'apprentissage_enfants.php',
    texteLien1: 'un enfant /',
    lien2: 'apprentissage_adultes.php',
    texteLien2: 'un adulte',
};

```

```

export function creationRole(role, parent) {
    const div = document.createElement('div');
    div.setAttribute('class', role.class);
    const photo = document.createElement('div');
    photo.setAttribute('class', `${role.class}Photo`);
    const texte = document.createElement('div');
    texte.setAttribute('class', `${role.class}Texte`);
    const titre = document.createElement('h4');
    titre.setAttribute('class', `${role.class}Titre`);
    titre.textContent = role.titre;
    const explications = document.createElement('p');
    explications.setAttribute('class', `${role.class}Explications`);
    explications.textContent = role.explications;
    const enSavoirPlus = document.createElement('p');
    enSavoirPlus.setAttribute('class', `${role.class}EnSavoirPlus`);
    enSavoirPlus.textContent = role.introLien;
    const a1 = document.createElement('a');
    a1.setAttribute('href', `${URL}${role.lien1}`);
    a1.textContent = role.texteLien1;
    const a2 = document.createElement('a');
    a2.setAttribute('href', `${URL}${role.lien2}`);
    a2.textContent = role.texteLien2;
    parent.append(div);
    div.append(photo, texte);
    texte.append(titre, explications, enSavoirPlus);
    enSavoirPlus.append(a1, a2);
}

```

Avec la gestion en Grid j'ai positionné les photos alternativement à gauche et à droite :

```

.role1,
.role2,
.role3,
.role4{
    display: grid;
}

.role1,
.role2,
.role3{
    grid-template-columns: 20% 70%;
    margin-left: 50px;
    margin-right:50px;
}

.role1,
.role2,
.role3,
.role4{
    grid-template-columns: 70% 20%;
    margin-left:50px;
    margin-right:-70px;
}

.role1Photo,
.role3Photo{
    grid-column: 1/2;
    grid-row: 1/2;
    margin-right:20px;
}

.role2Photo,
.role4Photo{
    grid-column: 2/3;
    grid-row: 1/2;
    margin-left:20px;
}

.role1Texte,
.role3Texte{
    grid-column: 2/3;
    grid-row: 1/2;
    text-align: left;
}

.role2Texte,
.role4Texte{
    grid-column: 1/2;
    grid-row: 1/2;
    text-align: right;
}

```

## Mise en page avec Flex

La mise en page en Flex permet de présenter les éléments à la suite, en ligne, ou en colonne, avec retour à la ligne ou non. J'ai utilisé le Flex, pour beaucoup d'éléments de mon projet, car il est particulièrement intéressant quand on gère l'adaptation aux différentes tailles d'écran.

Par exemple, pour ces 3 éléments, j'ai utilisé le Flex en ligne (par défaut) pour les grands écrans, étirés sur toute la largeur.

```

.categories{
    display: flex;
    width: 100%;
    justify-content: space-around;
}

```



Sous les 750px, j'ai paramétré le Flex en colonne pour que les 3 éléments soient les uns en dessous des autres :

```

@media screen and (max-width: 750px){
    .categories{
        flex-direction: column;
        align-items: center;
    }
}

```

## Le Formulaire de Contact

L'enjeu était pour moi de créer un formulaire de contact ultra-sécurisé. C'était également le premier pas dans ce projet vers le back-end.

... ou via ce formulaire de contact 

Civilité\*  M.  Mme  Autre

Nom\*  Prénom\*

Organisme   
Organisme/Société/École

Email\*  Téléphone\*   
exemple@email.fr ex: 0610101010

Adresse   
Nº de rue, rue

Complément   
Complément d'adresse

Code postal  Ville   
Code Postal Ville

Objet\*

Message\*

Caractères restants : 1000/1000

J'accepte que mes coordonnées soient utilisées pour recevoir une réponse à ma demande. Si je deviens client, mes données seront conservées pour la gestion de notre relation commerciale, conformément à notre [Politique de confidentialité](#).

Je ne suis pas un robot   
reCAPTCHA  
Confidentialité - Conditions

\* = à compléter obligatoirement

### La conception

En soi, créer un formulaire de contact est très simple, mais le sécuriser demande beaucoup d'attentions. J'ai fait le choix de le coder en HTML, pour profiter de sa simplicité de mise en place et des verrous de base, comme le type "d'input", ou la longueur maximum de saisie ("maxlength"). Les formulaires HTML sont aussi bien conçus d'un point de vue utilisateur pour leur clarté et leur accessibilité avec les balises "label".

```
<label for="mail">Email*</label>
<input id="mail" name="mail" type="email" maxlength=40 placeholder="exemple@email.
fr" oninput="validateMail(event)">
```

### Saisie verrouillée

Pour verrouiller les saisies, j'ai mis des règles de validation de **Regex** en JS. Je voulais aussi que des messages d'alerte personnalisés s'affichent en fonction des erreurs.

J'ai posé une écoute JS sur le bouton de validation du formulaire, pour qu'elle vérifie que tous les **champs obligatoires** soient bien remplis au moment de la soumission et bloque l'envoi si ce n'est pas le cas, avec affichage de messages d'erreurs.

```

function validateTelephone(event) {
    const input = event.target;
    const value = input.value;
    const regexTelephone = /^[0][1-9][0-9]{8}$/;
    const regexFauxTel = /^[1-9][1][0-9]{0,9}$/;
    const regexCaracteres = /^[^\D\s{0,10}]/;
    const regexMixte = /^(?=.*\d)(?=.*[a-zA-Z]).*[^\w\s]).+$/;

    const messageAAfficher = document.getElementById("error-message-telephone");

    if (regexCaracteres.test(value)) {
        input.value = value.slice(0, -1);
        messageAAfficher.textContent = "Seuls les chiffres sont acceptés";
        messageAAfficher.style.color = "red";
    } else if (regexFauxTel.test(value)) {
        input.value = value.slice(0, -1);
        messageAAfficher.textContent = "Numéro invalide (commencez par 0...)";
        messageAAfficher.style.color = "orange";
    } else if (regexMixte.test(value)) {
        input.value = value.slice(0, -1);
        messageAAfficher.textContent = "Seuls les chiffres sont acceptés.";
        messageAAfficher.style.color = "orange";
    } else if (!regexTelephone.test(value)) {
        messageAAfficher.textContent = "Numéro incomplet";
        messageAAfficher.style.color = "orange";
    } else {
        messageAAfficher.textContent = "";
    }
}

document.getElementById("formulaire").addEventListener("submit", function(event) {
    // Empêche l'envoi du formulaire par défaut
    event.preventDefault();

    let formIsValid = true;

    // Appel de tous les champs du formulaire
    const nom = document.getElementById("nom");
    const prenom = document.getElementById("prenom");
    // Fonction pour afficher le message d'erreur
    function afficherErreur(champ, message) {
        document.getElementById(`error-message-${champ.id}`).textContent = message;
        formIsValid = false;
    }

    // Réinitialiser les messages d'erreur
    document.querySelectorAll(".error-message").forEach(function(element) {
        element.textContent = "";
    });

    // Valider chaque champ
    if (nom.value.trim() === "") {
        afficherErreur(nom, "Veuillez saisir votre nom.");
    }
})

```

Afin de limiter les risques de piratage ou d'envoi de mails par des robots, j'ai intégré un contrôleur **reCaptcha V2**, qui me paraissait la version la plus appropriée pour un formulaire (case à cocher ou choix d'images). J'ai paramétré les clés de contrôle de reCaptcha et rattaché le site web au compte Google. La clé secrète est stockée dans le fichier .env, dont l'accès a été bloqué pour un utilisateur extérieur via .htaccess.

```

www > private > htaccess
1 # www/private/.env
2 Order allow,deny
3 Deny from all

```

J'ai créé une règle de traitement en PHP natif, car j'ai développé cette partie directement dans mon projet front :

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Vérification de la réponse reCAPTCHA
    $recaptchaResponse = $_POST['g-recaptcha-response'];

    $url = 'https://www.google.com/recaptcha/api/siteverify';
    $data = [
        'secret' => $secretKey,
        'response' => $recaptchaResponse,
        'remoteip' => $_SERVER['REMOTE_ADDR']
    ];

    $options = [
        'http' => [
            'method' => 'POST',
            'header' => 'Content-type: application/x-www-form-urlencoded',
            'content' => http_build_query($data)
        ]
    ];

    $context = stream_context_create($options);
    $verify = file_get_contents($url, false, $context);
    $captchaSuccess = json_decode($verify);
}

```

Si le test reCaptcha est réussi et le formulaire complété selon toutes les règles établies, toutes les informations et le message saisis par l'utilisateur sont récupérés. J'ai ajouté la mention "**htmlspecialchars**" pour que chaque contenu soit retravaillé, pour éviter les attaques XSS (Cross-Site Scripting) :

```

$message .= '<b>Email : </b>' .
    htmlspecialchars($_POST['mail']) . '<br>
<b>Objet du message : </b>' .
    htmlspecialchars($_POST['objet']) . '<br>
<b>Message : </b>' .
    htmlspecialchars($_POST['message']) . '</p>';

```

J'ai utilisé "**PHPmailer**" pour l'envoi des données saisies dans le formulaire, par mail :

```

$mail = new PHPMailer(true);
$mail->isSMTP();
$mail->Host = 'mailhog';
$mail->Port = 1025;
$mail->CharSet = 'UTF-8';
$mail->SMTPAuth = false;

```

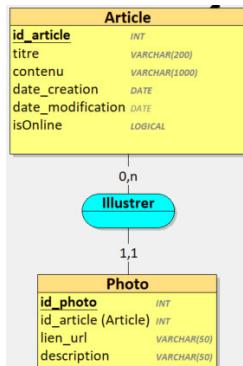
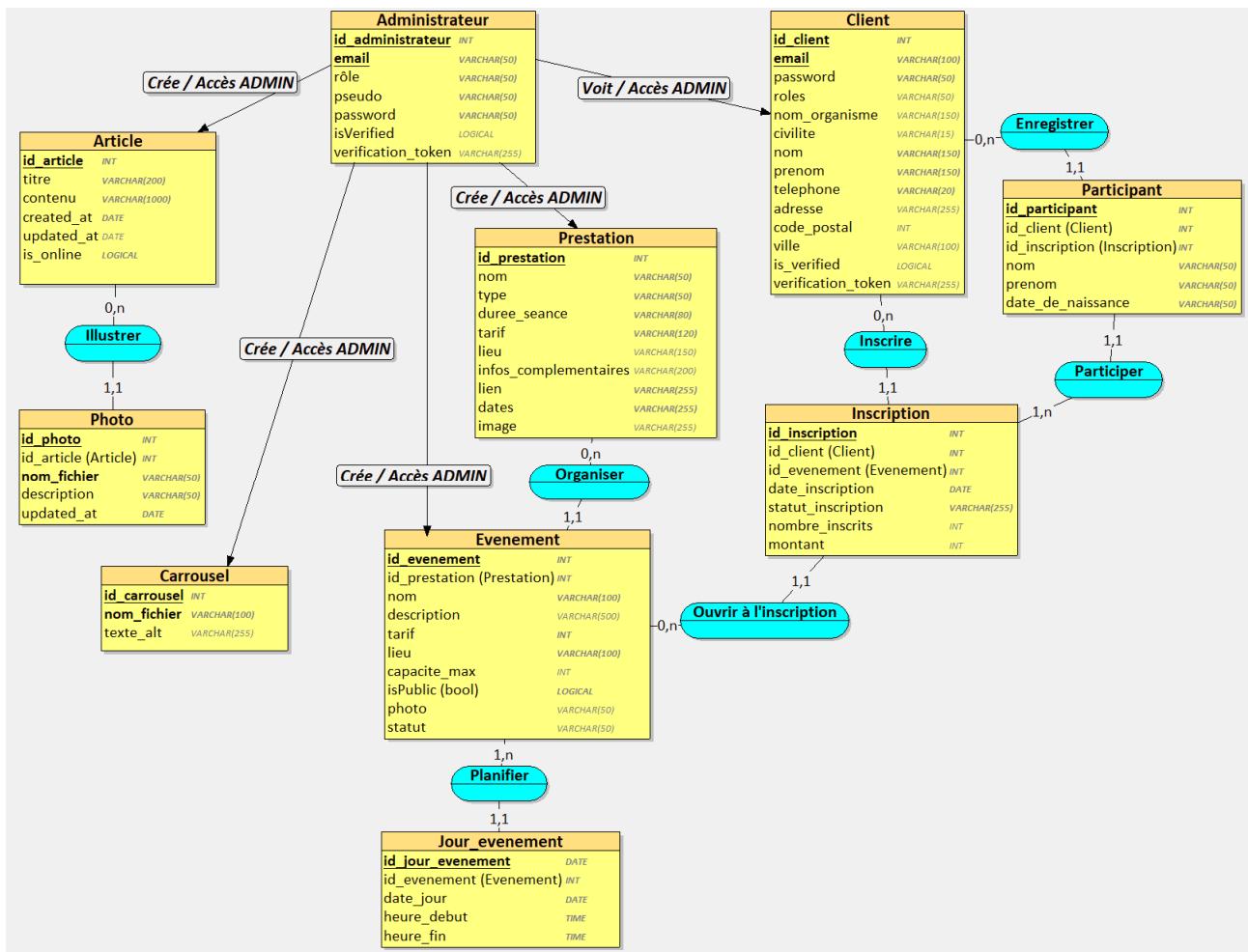
## La Base de Données Relationnelle

---

Pour gérer toute la partie back-end, j'ai créé un modèle de base de données relationnelle. Le principe de conception est le suivant : Michaël est seul gérant de son entreprise, il pilotera tout, avec un accès spécial administrateur. Et le client pourra créer sur le site, son compte en ligne et inscrire des participants à des événements.

Il pourra :

- Gérer et mettre à jour ses prestations (tarifs par ex.) pour que ce soit effectif sur son site.
- Ajouter ou supprimer des photos, parmi les photos qui défilent sur sa page d'accueil.
- Publier des articles sur son actualité (style blog).
- Planifier, ouvrir à la réservation des événements de groupe qu'il organise (cours collectifs).
- Récupérer la liste des participants aux événements.
- Avoir accès aux données clients qui se sont enregistrés sur son site.



Voici le code SQL pour les tables Article et Photo :

```

CREATE TABLE Article (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titre VARCHAR(200) NOT NULL,
    contenu TEXT NOT NULL,
    created_at DATETIME NOT NULL,
    updated_at DATETIME DEFAULT NULL,
    is_online BOOLEAN NOT NULL
);

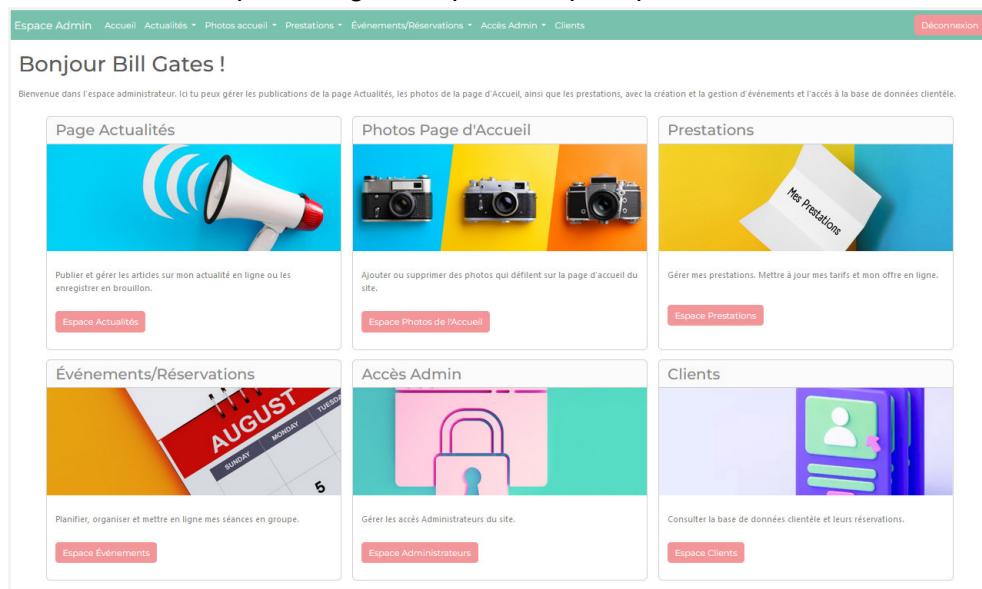
CREATE TABLE Photo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    lien_url VARCHAR(255) NOT NULL,
    article_id INT,
    FOREIGN KEY (article_id) REFERENCES Article(id)
);

```

J'ai généré le code d'initialisation SQL complet, en vue de la mise en ligne du site. En local, j'ai utilisé la fonctionnalité make:migration de Doctrine sous Symfony.

# L'Espace Administrateur

Mon idée était de créer un espace de gestion plaisant, pratique et facile à utiliser, et sécurisé :



The screenshot shows the main dashboard of the administrator space. At the top, there's a navigation bar with links for 'Espace Admin', 'Accueil', 'Actualités', 'Photos accueil', 'Prestations', 'Événements/Réservations', 'Accès Admin', 'Clients', and a 'Déconnexion' button. Below the navigation is a welcome message 'Bonjour Bill Gates !'. A descriptive text explains the purpose of the space: 'Bienvenue dans l'espace administrateur. Ici tu peux gérer les publications de la page Actualités, les photos de la page d'accueil, ainsi que les prestations, avec la création et la gestion d'événements et l'accès à la base de données clientèle.' The dashboard is organized into several cards:

- Page Actualités**: Shows a megaphone icon. Description: 'Publier et gérer les articles sur mon actualité en ligne ou les enregistrer en brouillon.' Button: 'Espace Actualités'.
- Photos Page d'Accueil**: Shows three camera icons. Description: 'Ajouter ou supprimer des photos qui défilent sur la page d'accueil du site.' Button: 'Espace Photos de l'Accueil'.
- Prestations**: Shows a document labeled 'Mes Prestations'. Description: 'Gérer mes prestations. Mettre à jour mes tarifs et mon offre en ligne.' Button: 'Espace Prestations'.
- Événements/Réservations**: Shows a calendar for August. Description: 'Planifier, organiser et mettre en ligne mes séances en groupe.' Button: 'Espace Événements'.
- Accès Admin**: Shows a padlock icon. Description: 'Gérer les accès Administrateurs du site.' Button: 'Espace Administrateurs'.
- Clients**: Shows a user profile icon. Description: 'Consulter la base de données clientèle et leurs réservations.' Button: 'Espace Clients'.

## Choix de conception

J'ai géré le back et le visuel de l'Espace Administrateur sur **Symfony**. Celui-ci n'étant pas dédié au public, j'ai utilisé **Bootstrap**, afin d'être efficiente au niveau de la conception visuelle. J'ai aussi utilisé **Twig**, pour pouvoir mêler HTML et PHP et bénéficier de sa sécurité intégrée.

## Les Accès

J'ai créé une entité "Admin" pour les identifiants de connexion administrateurs.

```
#Route('/admin/register', name: 'admin_register')
#[IsGranted('ROLE_ADMIN')]
public function register(Request $request, UserPassw
```

L'accès à toutes les pages de l'Espace Administrateur, sauf celle de la connexion, n'est possible que pour les profils ayant le 'ROLE\_ADMIN' :

## La Page Actualités

Depuis l'Espace Admin, Michaël peut publier des articles sur son actualité en ligne, à la manière d'un blog. La route globale dédiée à l'actualité est /admin/article :

```
#Route('/admin/article')
class ArticleController extends AbstractController
{
```

- **Créer une nouvelle publication**



The form is titled 'Créer une nouvelle publication'. It has two input fields: 'Titre' and 'Contenu'. Below these is a note: 'Mise en ligne ? Si oui, cocher la case. Si non, ne pas cocher, les données seront enregistrées, mais pas publiées.' followed by a checkbox. At the bottom are two buttons: 'Ajouter une photo' (in red) and 'Sauvegarder' (in green).

Un formulaire (ArticleType complété d'un PhotoType) est créé à partir des entités Article et Photo pour saisir les données de la publication et y intégrer des photos.

Le contrôleur ArticleController gère la génération du formulaire sur la route /new, contrôle la saisie selon les critères (Asserts) renseignés dans les entités et les "forms", persiste et flushes les données dans les deux tables Article et Photo.

### Squelette du formulaire Article :

```
class ArticleType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('titre', TextType::class, [
                'label' => "Titre",
                'required' => true,
                'constraints' => [
                    new Assert\NotBlank(['message' => 'Le titre est obligatoire.']),
                ],
            ])
            ->add('contenu', TextareaType::class, [
                'label' => "Contenu",
                'attr' => [
                    'cols' => "40",
                    'rows' => "5",
                ],
                'constraints' => [
                    new Assert\NotBlank(['message' => 'Le titre est obligatoire.']),
                ],
                'required' => true,
            ])
            ->add('isOnline', CheckboxType::class, [
                'label' => "Mise en ligne ? Si oui, cocher la case. Si non, ne pas cocher, les données seront enregistrées, mais pas publiées.",
                'required' => false,
            ]);
        if ($options['include_photos']) {
            $builder->add('photos', CollectionType::class, [
                'entry_type' => PhotoType::class,
                'allow_add' => true,
                'allow_delete' => true,
                'by_reference' => false,
                'required' => false,
                'attr' => [
                    'class' => 'photo-collection',
                ],
            ]);
        }
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Article::class,
            'include_photos' => true,
        ]);
    }
}
```

### Squelette du formulaire Photo :

```
class PhotoType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('description', TextType::class, [
                'label' => 'Description de la photo',
                'attr' => [
                    'class' => "photoDescription",
                    'placeholder' => "à renseigner obligatoirement",
                ],
                'constraints' => [
                    new Assert\NotBlank(['message' => 'La description de la photo est obligatoire.']),
                ],
                'required' => true,
            ])
            ->add('imageFile', FileType::class, [
                'label' => 'Télécharger une photo (3Mo max par photo)',
                'required' => false,
                'attr' => [
                    'class' => "photo",
                ],
                'constraints' => [
                    new Assert\File([
                        'maxSize' => '3M',
                        'mimeTypes' => [
                            'image/jpeg',
                            'image/png',
                            'image/webp'
                        ],
                        'mimeTypesMessage' => 'Veuillez télécharger une image valide (JPEG, PNG, WEBP et maxi 3Mo).',
                    ])
                ],
            ]);
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Photo::class,
        ]);
    }
}
```

## Contrôleur :

```

class ArticleController extends AbstractController
{
    #[Route('/new', name: 'article_new', methods: ['GET', 'POST'])]
    #[IsGranted('ROLE_ADMIN')]
    public function new(Request $request): Response
    {
        $article = new Article();
        $form = $this->createForm(ArticleType::class, $article);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            foreach ($article->getPhotos() as $photo) {
                $photo->setArticle($article);
                $this->entityManager->persist($photo);
            }
            $this->entityManager->persist($article);
            $this->entityManager->flush();

            return $this->redirectToRoute('article_view', ['id' => $article->getId()]);
        }

        return $this->render('article/new.html.twig', [
            'form' => $form,
        ]);
    }
}

```

## Visuel Twig :

```

{% extends 'base.html.twig' %}
{% block title %}Nouvelle publication | Actualités{% endblock %}
{% block styles %}<link href="/styles/article_new.css" rel="stylesheet">{% endblock %}
{% block script %}<script src="/js/article_new.js" defer></script>{% endblock %}

{% block body %}

<h1>Créer une nouvelle publication</h1>
{{ form_start(form) }}
{{ form_row(form.titre) }}
{{ form_row(form.contenu) }}
{{ form_row(form.isOnLine) }}

<div id="photos-section">
    <div id="photos-collection" data-prototype="{{ form_widget(form.photos.vars.prototype)|e('html_attr') }}>
        {% for photo in form.photos %}
            <div class="photo-form-group">
                {{ form_widget(photo) }}
                <img class="preview" style="max-width: 150px; margin-top: 10px;" src="" alt="Aperçu de la photo"/>
            </div>
        {% endfor %}
    </div>
    <button type="button" id="add-photo" class="btn btn-secondary mt-2">Ajouter une photo</button>
</div>

<div id="validation-publication">
    <button type="submit" id="bouton_sauv_publi" class="btn btn-primary">Sauvegarder</button>
</div>
{{ form_end(form) }}
{% endblock %}

```

Pour la gestion des photos, il est possible d'y joindre une ou plusieurs photos, alors j'ai utilisé la méthode de "collections" de Symfony. Un prototype de formulaire d'ajout de photos est intégré dans le HTML/Twig (voir ci-dessus). Grâce à une écoute au click en JS, il se génère une copie du prototype quand l'Admin clique sur "Ajouter une photo", avec prévisualisation de celle-ci, grâce à l'interface FileReader intégrée à Symfony, qui lit les fichiers sélectionnés dans des input type "file".

The screenshot shows a file upload interface. At the top, there is a text input field labeled "Description de la photo" with the placeholder "à renseigner obligatoirement". Below it is a file selection input field with the placeholder "Télécharger une photo (3Mo max par photo)" and a "Parcourir..." button. A message indicates "Aucun fichier sélectionné.". At the bottom left is a red "Annuler" button, and at the bottom right is a pink "Ajouter une photo" button.

```

document.addEventListener('DOMContentLoaded', function() {
    const addPhotoButton = document.getElementById('add-photo');
    const photosCollection = document.getElementById('photos-collection');
    const prototype = photosCollection.getAttribute('data-prototype');

    function addPhotoField() {
        const newIndex = photosCollection.children.length;
        const newFormHtml = prototype.replace(/__name__/g, newIndex);

        const newFormDiv = document.createElement('div');
        newFormDiv.setAttribute('class', 'photo-form-group');
        newFormDiv.innerHTML = newFormHtml;

        const previewImg = document.createElement('img');
        previewImg.setAttribute('class', 'preview');
        previewImg.setAttribute('style', 'max-width: 150px; margin-top: 10px;');
        newFormDiv.appendChild(previewImg);

        const removePhotoButton = document.createElement('button');
        removePhotoButton.setAttribute('type', 'button');
        removePhotoButton.setAttribute('class', 'btn btn-danger remove-photo');
        removePhotoButton.textContent = 'Annuler';
        newFormDiv.appendChild(removePhotoButton);

        photosCollection.appendChild(newFormDiv);
    }

    const fileInput = newFormDiv.querySelector('input[type="file"]');

    fileInput.addEventListener('change', function(event) {
        if (fileInput.files && fileInput.files[0]) {
            const reader = new FileReader();

            reader.onload = function(e) {
                previewImg.src = e.target.result;
            };
            reader.readAsDataURL(fileInput.files[0]);
        } else {
            previewImg.src = '';
        }
    });

    removePhotoButton.addEventListener('click', function() {
        newFormDiv.remove();
    });
}

addPhotoButton.addEventListener('click', addPhotoField);
});

```

**Exemple :**



Quand la validation du formulaire s'effectue, les données sont enregistrées en base de données. Les photos ont un nom\_fichier qui leur est attribué, et elles sont stockées dans le dossier uploads/photos grâce au bundle VichUploader.

id	article_id	nom_fichier	description	updated_at
12	5	header-fond-670d23654f93a816987089.webp	Quatre enfants assis sur leurs vélos, arrêtés dans l'herbe	2024-10-14 13:57:57
13	5	categ-adultes-670d236550bde022036072.webp	Un homme et une femme qui roulent à vélo en ville	2024-10-14 13:57:57

```

#[Vich\UploadableField(mapping: 'photo_image', fileNameProperty: 'nomFichier')]
private ?File $imageFile = null;

public function setImageFile(?File $imageFile = null): void
{
    $this->imageFile = $imageFile;

    if ($imageFile instanceof UploadedFile) {
        // It is required that at least one field changes if you are using doctrine
        // otherwise the event listeners won't be called and the file is lost
        $this->updatedAt = new \DateTimeImmutable();
    }
}

```

```

vich_uploader:
  db_driver: orm
  mappings:
    photo_image:
      uri_prefix: /uploads/photos
      upload_destination: '%kernel.project_dir%/public/uploads/photos'

```

- **Voir une publication**

La route /views permet de voir toutes les publications enregistrées. Elle utilise un QueryBuilder de Doctrine, permettant de saisir une requête SQL pour la compilation des données. Les publications vont apparaître de la plus récente à la plus ancienne (en fonction soit de la date de modification ou de création grâce à COALESCE, qui prendra la première non vide). Un compteur du nombre d'articles "en ligne" est également mis en place :

Espace Admin

Il y a 3 articles en ligne.

- Venez me retrouver ce samedi 19/10 à Armentières !**  
Statut : en ligne  
Venez me retrouver ce samedi 19/10 à Armentières !  
Je serai présent au Salon des Vélos d'Armentières, pour vous présenter mon activité. Retrouvez-moi toute la journée sur mon stand qui sera situé en allée B.  
Dernière modification : 14/10/2024  
**Voir/modifier** **Supprimer**
- Parlons vélo !**  
Statut : en ligne  
Parlons vélo !  
Rendez vous pour une réunion bla-bla autour du vélo le lundi 11/11 au café du coin.  
Dernière modification : 11/10/2024  
**Voir/modifier** **Supprimer**
- Mon article test**  
Statut : en ligne

```
#Route('/views', name: 'article_views', methods: ['GET'])
#[IsGranted('ROLE_ADMIN')]
public function views(ArticleRepository $articleRepository): Response
{
    $articles = $articleRepository->createQueryBuilder('a')
        ->addSelect('COALESCE(a.updatedAt, a.createdAt) AS HIDDEN effectiveDate')
        ->orderBy('effectiveDate', 'DESC')
        ->getQuery()
        ->getResult();

    $articlesOnlineCount = $articleRepository->count(['isOnline' => true]);

    return $this->render('article/views.html.twig', [
        'articles' => $articles,
        'articlesOnlineCount' => $articlesOnlineCount,
    ]);
}
```

La route /id permet de voir une publication précise, récupérée via son id. On y accède par exemple via les boutons "voir/modifier" de la page views, contenant un lien vers la page de l'article concerné :

```
<a class="btn btn-outline-dark mt-auto" href="{{ path('article_view', {id: article.id}) }}>Voir/modifier</a>
```

Espace Admin

Publication :

Venez me retrouver ce samedi 19/10 à Armentières ! Identifiant : 5  
Créé le : 14/10/2024 13:57  
Venez me retrouver ce samedi 19/10 à Armentières ! Je serai présent au Salon des Vélos d'Armentières, pour vous présenter mon activité. Retrouvez-moi toute la journée sur mon stand qui sera situé en allée B.

**Retour aux articles** **Modifier** **Supprimer**

```
#Route('/{id}', name: 'article_view', requirements: ['id' => '\d+'], methods: ['GET'])
#[IsGranted('ROLE_ADMIN')]
public function view(?Article $article): Response
{
    return $this->render('article/view.html.twig', [
        'article' => $article,
    ]);
}
```

- **Modifier une publication**

La route /{id}/edit permet d'accéder à la modification de la publication selon son id. Le contrôleur reprend tous les éléments préalablement enregistrés pour cette publication, photos comprises. Pour la modification des éléments "texte" et l'ajout de photos, le fonctionnement est plus ou moins similaire à celui de la page new. Les modifications sont persistées en base de données.

Pour la suppression des photos existantes, j'ai éprouvé quelques difficultés à gérer cela, car je voulais une suppression instantanée. J'ai créé un contrôleur secondaire sur l'entité Photo, qui gère la suppression, sur la route /photo/{id}/delete, nommée photo\_delete.

## Modifier une publication

Titre  
Venez me retrouver ce samedi 19/10 à Armentières !

Contenu  
Je serai présent au Salon des Vélos d'Armentières, pour vous présenter mon activité. Retrouvez-moi toute la journée sur mon stand qui sera situé en allée B.

Mise en ligne ? Si oui, cocher la case. Si non, ne pas cocher, les données seront enregistrées, mais pas publiées.

Description de la photo  
Un homme et une femme qui roulent à vélo en ville

Télécharger une photo (3Mo max par photo) [Parcourir...]. Aucun fichier sélectionné.



[Supprimer](#)

Description de la photo  
Homme roulant à vélo devant un mur en brique rouge

Télécharger une photo (3Mo max par photo) [Parcourir...]. Aucun fichier sélectionné.



[Supprimer](#)

[Ajouter une photo](#)

[Enregistrer](#)

```

#[Route('/{id}/edit', name: 'article_edit', requirements: ['id' => '\d+'], methods: ['GET', 'POST'])]
#[IsGranted('ROLE_ADMIN')]
public function edit(Request $request, Article $article): Response
{
    $form = $this->createForm(ArticleType::class, $article, [
        'include_photos' => true, // Inclut les photos du formulaire
    ]);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $photos = $article->getPhotos();

        foreach ($photos as $photo) {
            if ($photo->getId() === null) {
                $this->entityManager->persist($photo);
            }
        }
        $this->entityManager->flush();

        return $this->redirectToRoute('article_view', ['id' => $article->getId()]);
    }

    return $this->render('article/edit.html.twig', [
        'form' => $form->createView(),
        'article' => $article,
    ]);
}

```

```

class PhotoController extends AbstractController
{
    #[Route('/photo/{id}/delete', name: 'photo_delete', requirements: ['id' => '\d+'], methods: ['DELETE'])]
    #[IsGranted('ROLE_ADMIN')]
    public function delete(int $id, EntityManagerInterface $entityManager): Response
    {
        $photo = $entityManager->getRepository(Photo::class)->find($id);

        if (!$photo) {
            return new Response('Photo not found', 404);
        }

        $nomFichier = $photo->getNomFichier();

        if ($nomFichier) {
            $filePath = $this->getParameter('photos_directory') . '/' . $nomFichier;

            if (file_exists($filePath)) {
                try {
                    unlink($filePath);
                } catch (\Exception $e) {
                    return new Response("Failed to delete file: " . $e->getMessage(), 500);
                }
            }
        }

        $entityManager->remove($photo);
        $entityManager->flush();

        return new Response('Photo deleted successfully', 200);
    }
}

```

La suppression d'une photo est effectuée quand l'Admin clique sur "Supprimer". J'ai mis une écoute sur le bouton en JS, qui réalise ensuite un fetch avec la route de suppression.

```

{% for photo in form.photos %}
    <div class="photo-form-group" data-photo-id="{{ photo.vars.value.id }}>
        {{ form_widget(photo) }}
        {% if photo.vars.value is not null and photo.vars.value.nomFichier is not null %}
            
        {% else %}
            <img class="preview" style="max-width: 150px; margin-top: 10px;" src="" alt="Aperçu de la photo"/>
        {% endif %}
        <button type="button" class="btn btn-danger delete-photo">Supprimer</button>
    </div>
{% endfor %}

```

Le chemin de stockage des photos est renseigné dans les "parameters" du fichier "services.yaml". J'ai choisi de gérer la suppression de façon simple en recherchant le nom du fichier et en le "unlink" du dossier, pour qu'il disparaisse de dossier de stockage, en plus d'être supprimé de la base de données.

```

parameters:
    photos_directory: '%kernel.project_dir%/public/uploads/photos'

```

```

document.addEventListener('DOMContentLoaded', function() {
    function handleDeleteButtons() {
        document.querySelectorAll('.photo-form-group .delete-photo').forEach(deleteButton => {
            deleteButton.addEventListener('click', function() {
                const photoFormGroup = this.parentElement;
                const photoId = photoFormGroup.getAttribute('data-photo-id');
                if (photoId) {
                    fetch('/admin/article/photo/${photoId}/delete', {
                        method: 'DELETE',
                        headers: {
                            'X-Requested-With': 'XMLHttpRequest'
                        }
                    })
                    .then(response => {
                        console.log('Response:', response);
                        if (response.ok) {
                            photoFormGroup.remove();
                            updateNumeroIndex();
                        } else {
                            console.error('Erreur lors de la suppression de la photo.');
                        }
                    })
                    .catch(error => console.error('Erreur lors de la suppression de la photo :', error));
                }
            });
        });
    }

    addPhotoButton.addEventListener('click', addPhotoField);
    handleDeleteButtons();
});

```

### • Supprimer une publication

Pour supprimer une publication complète + les photos qui y sont rattachées, une route /{id}/delete a été créée. Elle récupère tous les éléments correspondants à la publication selon son id ainsi que toutes les photos y étant rattachées. Les photos sont d'abord supprimées du dossier de stockage, puis supprimées de la table Photo, tout comme les éléments de la table Article.

```

#[Route('/{id}/delete', name: 'article_delete', requirements: ['id' => '\d+'], methods: ['POST'])]
#[IsGranted('ROLE_ADMIN')]
public function delete(Article $article): RedirectResponse
{
    $photos = $article->getPhotos();

    foreach ($photos as $photo) {
        $nomFichier = $photo->getNomFichier();

        if ($nomFichier === null) {
            $this->addFlash('error', 'Le nom du fichier est manquant pour une photo.');
            continue;
        }

        $filePath = $this->getParameter('photos_directory') . '/' . $nomFichier;

        if (file_exists($filePath)) {
            try {
                unlink($filePath);
            } catch (FileException $e) {
                $this->addFlash('error', 'Erreur lors de la suppression du fichier : ' . $e->getMessage());
            }
        } else {
            $this->addFlash('error', "Le fichier n'existe pas : " . $filePath);
        }

        $this->entityManager->remove($photo);
    }

    try {
        $this->entityManager->remove($article);
        $this->entityManager->flush();
    } catch (\Exception $e) {
        $this->addFlash('error', 'Erreur lors de la suppression de l\'article : ' . $e->getMessage());
    }

    $this->addFlash('success', 'Article supprimé avec succès.');

    return $this->redirectToRoute('article_views');
}

```

La suppression se lance via un bouton dont l'action est rattachée à cette route :

```

<form method="post" action="{{ path('article_delete', {id: article.id}) }}" style="display: inline;" onsubmit="return confirmDelete();">
    <button class="btn btn-outline-danger mt-auto" type="submit">Supprimer</button>
</form>

```

## Enregistrement d'un nouveau Client ou d'un nouvel Admin

Un Administrateur peut créer d'autres profils administrateurs à partir de l'Espace Admin. Un nouveau Client peut s'enregistrer dans la base de données, pour pouvoir ensuite accéder aux inscriptions aux événements.

Même si les données ne sont pas stockées dans les mêmes tables, j'ai repris des principes similaires pour la gestion des créations de profils, notamment par rapport à la sécurité.

### Le Mot de Passe sécurisé

J'ai consulté les préconisations de la CNIL en matière de caractéristiques de mots de passe :

**Exemple 1 :** les mots de passe doivent être composés d'au minimum 12 caractères comprenant des majuscules, des minuscules, des chiffres et des caractères spéciaux à choisir dans une liste d'au moins 37 caractères spéciaux possibles.

Source : [www.cnil.fr](http://www.cnil.fr)

J'ai ensuite utilisé un regex pour sa validation : mini 12 caractères maxi 100, avec au moins 1 chiffre, 1 minuscule, 1 majuscule, 1 caractère spécial sauf : < > & \ (qui pourraient être interprétés dans le code) :

Validation via Assert sous Symfony pour la création d'un Admin :

```
#Assert\Regex('/^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*()_+=[\]\{};\";.\?])(!.*[<>\&]).{12,100}$/')
```

```
#[ORM\Column]
```

```
private ?string $password = null;
```

Validation via une fonction JS pour la création d'un Client :

```
// Fonction de validation pour le mot de passe
function validatePassword(value) {
    const regexPassword = /^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*()_+=[\]\{};\";.\?])(!.*[<>\&]).{12,100}$/;
    if (!regexPassword.test(value)) {
        afficherErreur(document.getElementById("password"), "Mot de passe non valide", "orange");
    } else {
        document.getElementById("error-message-password").textContent = "";
    }
}
```

### Validation du compte par mail

Une fois le nouveau membre enregistré, un "token" lui est affecté dans la base de données. Et son compte est considéré comme non vérifié. Cela lui bloque notamment l'accès au login.

\$verificationToken = bin2hex(random\_bytes(32)); ➔ crée dans le contrôleur qui gère le POST

The screenshot shows a user registration form and a corresponding verification email. The registration form includes fields for id, email, role, password, nom\_organ, civilité, nom, prenom, telephone, adresse, code, ville, is\_verified, and verification\_token. The verification token is generated as \$2y\$13\$gg79. The verification email is sent to test@test.fr and contains a link to validate the account. The email body includes a logo for 'MICHAEL MONITEUR CYCLISTE' and a message asking the user to click the link to validate their account. The link in the email is https://127.0.0.1:8000/api/validate-account?token=3f5072d6f3ccc81a2d78c78a9aff2c2060158118abed54b76948df807430bd8c.

Ce token est utilisé dans la création d'un lien. Ce lien lui est envoyé par mail, pour qu'il valide définitivement son compte en cliquant dessus.

Une route spécifique a été créée pour la vérification :

```
#Route('/admin/verify/email', name: 'admin_verify_email')
public function verifyUserEmail(Request $request, AdminRepository $adminRepository, EntityManagerInterface $entityManager)
: Response
{
    $token = $request->query->get('token');

    if (null === $token) {
        return $this->redirectToRoute('admin_register');
    }

    $user = $adminRepository->findOneBy(['verificationToken' => $token]);

    if (null === $user) {
        return $this->redirectToRoute('admin_register');
    }

    $user->setIsVerified(true);
    $user->setVerificationToken(null); // suppression du token une fois vérifié
    $entityManager->flush();

    $this->addFlash('success', 'Votre adresse email a été vérifiée.');

    return $this->redirectToRoute('admin_register_validation');
}
```

Une fois que le compte est vérifié, il passe en "true" pour "is\_verified" et le token se supprime. Cela lui ouvre l'accès à la connexion.

id	email	role	password	nom_organ	civilite	nom	prenom	telepho	adresse	code	vill	is_verified	verification_token
81	test@test.fr		\$2y\$13\$gg7a9	Madame	Flanflan	Pimpreline	0610101010	10 avenue Jean . 59000	Lille		1	[NULL]	

Ce principe permet de sécuriser le compte, de vérifier l'authenticité du mail et de l'identité de la personne qui crée le compte.

### Hashage du mot de passe

Les mots de passe sont tous hashés avant d'intégrer la base de données. Cela se fait depuis les contrôleurs qui gèrent les POST :

```
$hashedPassword = $this->passwordHasher->hashPassword($client, $data['password']);
$client->setPassword($hashedPassword);
```

### Contrôles supplémentaires pour la création du Client

La création du compte client se faisant en front, j'ai rajouté des contrôles de saisie similaires à ceux utilisés dans le formulaire de contact :

- des regex pour tous les champs de saisie. Exemple :

```
export function validateNom(event) {
    const input = event.target;
    const value = input.value;
    const regexNom = /^[a-zA-ZÀ-Ӄ]*$/;
    if (!regexNom.test(value)) {
        input.value = value.slice(0, -1);
        document.getElementById("error-message-nom").textContent = "Saisir votre nom (lettres uniquement)";
    } else {
        document.getElementById("error-message-nom").textContent = "";
    }
}
```

- un reCaptcha V2 est mis en place à la fin de formulaire d'enregistrement :

□ J'accepte que mes coordonnées soient utilisées et conservées pour la gestion de notre relation commerciale, conformément à notre [Politique de confidentialité](#).

Je ne suis pas un robot   
Confidentialité + Conditions

**Enregistrer**

\* = à compléter obligatoirement

## Contrôle supplémentaire pour la création de l'Admin

L'accès à /admin n'est autorisé que pour les administrateurs connectés, sauf pour le login. J'ai aussi complété cet spécificité d'accès en le précisant sur la route de l'enregistrement :

```
config/packages/security.yaml
1 security:
77 |     access_control:
78 |         # Connexion Administrateur
79 |         - { path: ^/admin/login, roles: PUBLIC_ACCESS }
80 |         # Espace Administrateur
81 |         - { path: ^/admin, roles: ROLE_ADMIN }
```

#[IsGranted('ROLE\_ADMIN')]  
#[Route('/admin/register', name: 'admin\_register')]

## L'Espace Client

### • Le Login

Le client doit se connecter pour accéder à son espace.

Une icône d'œil permet de visualiser en clair le mot de Passe, grâce à une fonction d'écoute JS qui passe de type "password" à "text" l'input de saisie :

Mot de passe :  Mot de passe :

```
document.getElementById('togglePassword').addEventListener('click', function () {
    const passwordField = document.getElementById('password');
    const icon = this;
    if (passwordField.type === 'password') {
        passwordField.type = 'text';
        icon.classList.remove('fa-eye');
        icon.classList.add('fa-eye-slash');
    } else {
        passwordField.type = 'password';
        icon.classList.remove('fa-eye-slash');
        icon.classList.add('fa-eye');
    }
});
```

La demande de connexion du client s'effectue via un fetch vers apiPlatform. Un contrôle de la conformité des données saisies est effectué. En réponse, un token généré par Lexik JWT est envoyé comportant des informations sur l'utilisateur ainsi qu'une durée de validité d'une heure. Ce token sera stocké en sessionstorage.

Decoded JWT	
Key	Value
authToken	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.e
customerId	78
privacyAccepted	true
tokenExpiry	1728997694692

A partir de ce token, l'email de l'utilisateur sera extrait pour récupérer ses données clients. Le token sera utilisé dans les entêtes http pour accéder aux pages qui nécessitent une connexion. Le rôle sera utilisé pour bloquer/autoriser l'accès à certaines pages. Et la connexion client expirera à la fin de validité du token.

```

document.getElementById('login').addEventListener('submit', async function (event) {
    event.preventDefault();

    const loader = document.getElementById("loader");
    loader.style.display = "flex";

    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const errorMessage = document.getElementById('error-message');

    errorMessage.textContent = "";

    try {
        const response = await fetch('https://127.0.0.1:8000/api/login_check', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                email: email,
                password: password
            })
        });
        const data = await response.json();

        if (!response.ok) {
            throw new Error(data.message || 'Erreur lors de la connexion.');
        }

        if (data.token) {
            // Stockage du token dans sessionStorage
            const expiryDate = new Date().getTime() + 3600 * 1000;
            sessionStorage.setItem('authToken', data.token);
            sessionStorage.setItem('tokenExpiry', expiryDate);

            // Utilisation de jwt-decode pour extraire l'email
            const decodedToken = jwt_decode(data.token);
            const userEmail = decodedToken.username;

            // Récupérer les données du client par email
            const clientData = await getClientDataByEmail(userEmail, data.token);
            if (clientData) {
                sessionStorage.setItem('customerId', clientData.id);
            } else {
                console.error("Impossible de récupérer les données du client.");
            }

            // Redirection
            const redirectUrl = sessionStorage.getItem('redirectAfterLogin');
            window.location.href = redirectUrl || 'http://localhost:8086/client/myaccount.php';
        }
    } catch (error) {
        // Affiche le message d'erreur
        errorMessage.textContent = error.message;
    } finally {
        // Masquer le loader
        loader.style.display = "none";
    }
});

async function getClientDataByEmail(email, token) {
    try {
        const response = await fetch(`https://127.0.0.1:8000/api/clients/email/${encodeURIComponent(email)}`, {
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${token}`,
                'Content-Type': 'application/json'
            }
        });
        if (!response.ok) {
            throw new Error(`Erreur ${response.status}: ${response.statusText}`);
        }
        const data = await response.json();
        return data; // Retourne les données du client
    } catch (error) {
        console.error('Erreur lors de la récupération des données du client :', error);
        return null;
    }
}

```

## • La réinitialisation du mot de passe

Une fonctionnalité permet de réinitialiser le mot de passe depuis la page de login, le client aura à saisir son adresse email :

### Réinitialisation du mot de passe

Si votre adresse email est reconnue, vous allez recevoir un mail contenant un lien temporaire vous permettant de modifier votre mot de passe. Pensez à vérifier dans vos spams.

Si l'email saisi est reconnu dans la base de données, un mail lui est envoyé, avec un lien, contenant un token.

Ce token est généré depuis le contrôleur qui gère la demande de réinitialisation.

Adresse email :

Envoyer le lien de réinitialisation

Il est stocké dans la base de données client :

ID	Email	Mot de passe	Nom	Civilité	Prénom	Téléphone	Adresse	Code postal	Ville	IS	Token de vérification
78	amandine@gmail.com	["RC \$2y\$13\$SLPP!Bugs & Madame	Marchand	Madame	Amandine	0625252525	1 rue des I	62150	Fresnicourt	1	1cb8cfca3babbfaf113abbf50223919125f2743742057d5941af5388f3657ed9

```
#Route(path: '/forgot_password', name: 'client_forgot_password')
public function forgotPassword(Request $request, EntityManagerInterface $em, MailerInterface $mailer)
: Response
{
    if ($request->isMethod('POST')) {
        $email = $request->request->get('email');
        $client = $em->getRepository(Client::class)->findOneBy(['email' => $email]);

        if ($client) {
            // Génère un token unique pour la réinitialisation du mot de passe
            $token = bin2hex(random_bytes(32));
            $client->setVerificationToken($token);
            $em->flush();

            // Envoie de l'email avec le lien de réinitialisation
            $url = $this->generateUrl('client_reset_password', ['token' => $token], UrlGeneratorInterface::ABSOLUTE_URL);
            $email = (new MimeEmail())
                ->from('contact@michael-moniteur-cycliste.fr')
                ->to($client->getEmail())
                ->subject('Réinitialisation de votre mot de passe')
                ->html($this->renderView('clientsecurity/reset_password_email.html.twig', [
                    'url' => $url,
                ]));

            $mailer->send($email);
            $this->addFlash('success', 'Un email avec un lien de réinitialisation a été envoyé.');
        } else {
            $this->addFlash('error', "Aucun compte n'est associé à cet e-mail.");
        }
        return $this->render('clientsecurity/envoi_email_reset.html.twig');
    }
    return $this->render('clientsecurity/forgot_password.html.twig');
}
```

Email envoyé :

From <contact@michael-moniteur-cycliste.fr>  
To <amandine@gmail.com>  
Subject Réinitialisation de votre mot de passe  
Date Tue, 15 Oct 2024, 1:45 pm (2.4 kB)  
Tags Add tags...

HTML HTML Source Text Headers Raw HTML Check 0% Link Check

  
MICHAËL MONITEUR CYCLISTE

Bonjour,

Vous avez fait une demande de réinitialisation de mot de passe sur le site de Michaël Moniteur Cycliste.

Cliquez sur le bouton ci-dessous pour réinitialiser votre mot de passe :

[Réinitialiser mon mot de passe](#)

Si vous n'avez pas demandé cette réinitialisation, ignorez ce message.

```
<div class="container">
<h1>Bonjour,</h1>
<p>Vous avez fait une demande de réinitialisation de mot de passe sur le site de Michaël Moniteur Cycliste.</p>
<p>Cliquez sur le bouton ci-dessous pour réinitialiser votre mot de passe :</p>
<a href="{{ url }}" class="button">Réinitialiser mon mot de passe</a>
<p class="footer">Si vous n'avez pas demandé cette réinitialisation, ignorez ce message.</p>
</div>
```



Le lien unique, comportant le token rattaché au client, mène vers une page de réinitialisation:

<https://127.0.0.1:8000/reset-password/1cb8cfca3babffaa113abbf50223919125f2743742057d5941af5388f3657>

Le nouveau mot de passe doit répondre aux mêmes critères que lors de la création. Et les deux saisies doivent correspondre.

Une fois que le formulaire soumis est valide, le mot de passe est mis à jour dans la base de données et le token supprimé :

AZ adres	AZ code	AZ ville	123 is_	AZ verification_token
1 rue des t	62150	Fresnicourt	1	[NULL]

```
#Route(path: '/reset-password/{token}', name: 'client_reset_password')
public function resetPassword(string $token, Request $request, EntityManagerInterface $em,
UserPasswordEncoderInterface $passwordHasher): Response
{
    $client = $em->getRepository(Client::class)->findOneBy(['verificationToken' => $token]);

    if (!$client) {
        throw $this->createNotFoundException('Accès invalide.');
    }
    $form = $this->createForm(ResetPasswordType::class);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $data = $form->getData();
        $newPassword = $data['new_password'];
        $confirmPassword = $data['confirm_password'];

        if ($newPassword === $confirmPassword) {
            $hashedPassword = $passwordHasher->hashPassword($client, $newPassword);
            $client->setPassword($hashedPassword);
            $client->setVerificationToken(null);
            $em->flush();

            $this->addFlash('success', 'Votre mot de passe a été réinitialisé avec succès.');
            return $this->render('clientsecurity/reset_password_success.html.twig');
        } else {
            $this->addFlash('error', 'Les mots de passe ne correspondent pas.');
        }
    }
    return $this->render('clientsecurity/reset_password.html.twig', [
        'form' => $form->createView(),
        'token' => $token,
    ]);
}
```

La page bascule ensuite sur une autre page où on annonce au client le succès de la mise à jour, puis automatiquement 5 secondes plus tard, il est redirigé sur la page de login :

```
<p class="success">Votre mot de passe a été réinitialisé avec succès.</p>
<p class="redirect-message">Vous allez être redirigé vers la page de connexion.</p>
<a href="http://localhost:8086/client/login.php" class="button">Retour à la connexion</a> ←—
MODE DEV —→
<script>
    // Redirection automatique après 5 secondes
    setTimeout(function() {
        window.location.href = "http://localhost:8086/client/login.php";
    }, 5000);
</script>
```

- Présentation de l'Espace Client



Le client retrouve sur son espace deux menus : "Mon Compte" et "Mes Réservations". Il a également la possibilité de se déconnecter soit avec le bouton dans l'entête de la page, soit directement depuis le menu de son compte. Pour déconnecter le client, la fonction logout supprime de la sessionStorage le token d'authentification et ses données annexes stockées (idClient et expiration du token) :

```
export function logout() {
    sessionStorage.removeItem('tokenExpiry');
    sessionStorage.removeItem('authToken');
    sessionStorage.removeItem('customerId');
    window.location.href = '/client/login.php';
}

document.addEventListener('DOMContentLoaded', () => {
    const logoutButton = document.getElementById('button-logout');
    if (logoutButton) {
        logoutButton.addEventListener('click', logout);
    }
});
```

En arrivant sur cette page, une vérification de la présence d'un token non expiré est faite à l'aide de fonctions en JS. Auquel cas, le client est rebasculé sur la page de login.

L'espace client est géré sur une seule page HTML. Une fonction JS et deux règles CSS gèrent l'affichage, en faisant apparaître le contenu du menu sélectionné au click et en fermant les autres :

```
93 |     
94 |     <div class="content" id="contenu">
95 |
96 |         
97 |         <div id="account-accueil" class="section">...
101 |             </div>
102 |
103 |         
104 |         <div id="account-section" class="section">...
106 |             </div>
107 |
108 |         
109 |         <div id="reservation-section" class="section">...
113 |             </div>
114 |         </div>
115 |     </div>
```

```
.section {
    display: none;
}
.section.active {
    display: block;
}

document.getElementById('account-accueil').classList.add('active');
document.getElementById('account-link').addEventListener('click', () => {
    document.getElementById('account-accueil').classList.remove('active');
    document.getElementById('account-section').classList.add('active');
    document.getElementById('reservation-section').classList.remove('active');
});

document.getElementById('reservation-link').addEventListener('click', () => {
    document.getElementById('account-accueil').classList.remove('active');
    document.getElementById('account-section').classList.remove('active');
    document.getElementById('reservation-section').classList.add('active');
});
```

L'espace client se charge grâce à deux appels API différents : un pour les coordonnées du client et un autre pour le récap des réservations effectuées par le client. La page s'affiche une fois que toutes les données sont chargées.

Pour sécuriser les appels API, j'ai précisé des règles de sécurité pour chaque entité mises à disposition sous apiPlatform. Par exemple, ici pour les données du client, seul le client lui-même (et l'admin) peut récupérer, modifier ou supprimer ses données. Seul l'admin peut récupérer toutes les données de tous les clients. Et la création d'un nouveau client est ouverte à tous ceux non connus dans la base.

## Premier menu : "Mon Compte"

```
async function loadClientData(customerId, token) {
  try {
    const response = await fetch(`https://127.0.0.1:8000/api/clients/${customerId}`, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json'
      }
    });
    const data = await response.json();
    console.log("Données du client : ", data);

    // Pré-remplir le formulaire avec les données récupérées
    document.getElementById('civilite').value = data.civilite;
    document.getElementById('nom').value = data.nom;
    document.getElementById('prenom').value = data.prenom;
    document.getElementById('organisme').value = data.nomOrganisme;
    document.getElementById('adresse').value = data.adresse;
    document.getElementById('codePostal').value = data.codePostal;
    document.getElementById('ville').value = data.ville;
    document.getElementById('telephone').value = data.telephone;
  } catch (error) {
    console.error('Erreur lors de la récupération des données client :', error);
  }
}
```

Le client peut modifier ses coordonnées (mêmes règles que pour les autres formulaires du site).

Il peut supprimer définitivement son compte avec toutes les données lui étant rattachées, par exemple ses inscriptions, grâce à l'effet "cascade" de la suppression (relation entre l'entité Client et l'entité Inscription) :

```
/** 
 * @var Collection<int, Inscription>
 */
#[ORM\OneToMany(targetEntity: Inscription::class, mappedBy: 'id_client', cascade: ['remove'])]
private Collection $inscriptions;
```

Il peut aussi modifier son mot de passe.

Dans tous les cas, avant un appel à l'api, l'authentification du client est toujours vérifiée :

```
const customerId = sessionStorage.getItem('customerId');
if (customerId) { ... etc...
```

```
#[ApiResource(
  operations: [
    new Get(security: "is_granted('ROLE_USER') and object.email = user.email"),
    new GetCollection(security: "is_granted('ROLE_ADMIN')"),
    new Post(security: "is_granted('PUBLIC_ACCESS')"),
    new Patch(security: "is_granted('ROLE_USER') and object = user"),
    new Delete(security: "is_granted('ROLE_USER') and object = user"),
  ]
)]
#[ORM\Entity(repositoryClass: ClientRepository::class)]
#[ORM\UniqueConstraint(name: 'UNIQ_IDENTIFIER_EMAIL', fields: ['email'])]
class Client implements UserInterface, PasswordAuthenticatedUserInterface
```

## Mon compte

## Deuxième menu : "Mes Réservations"

Dans cette partie, le client retrouve toutes les réservations qu'il a effectuées, avec un rappel des informations.

### Mes réservations

Inscription n° : 54  
Date d'inscription : 12/10/2024  
Statut : Validée  
Montant : 190 €  
Nombre d'Inscrits : 2

**Apprendre à rouler sans les petites roues !**

Lieu : Parking sécurisé de la rue des ponts, à Nantes

**Jours de l'événement :**  
Le 18/11/2024, de 09:00 à 12:00  
Le 19/11/2024, de 09:00 à 12:00

**Participants :**  
Dupont Rémi (né(e) le 15/05/2019)  
Dupont Joséphine (né(e) le 19/04/2015)

[Voir l'événement](#)

Pour ma part, ce visuel a été complexe à construire, car il fait appel à 5 tables dans la base. Au départ, j'avais commencé à faire des "fetchs dans les fetchs" et cela est devenu ingérable et trop complexe. J'ai donc créé une route spécifique sur Symfony compilant les données des 5 tables à l'aide des foreach. Et avec un seul fetch, je fais appel à cette route en front. Voici des extraits :

```
#Route('/api/recap-inscriptions-client/{userId}', name: 'recap-inscriptions-client', methods: ['GET'])
public function recapInscriptionsEspaceClient(int $userId, ClientRepository $clientRepository, IncriptionRepository $inscriptionRepository): Response {
    $client = $clientRepository->find($userId);

    if (!$client) {
        return new Response('Client non trouvé.', Response::HTTP_NOT_FOUND);
    }
    $inscriptions = $inscriptionRepository->findBy(['id_client' => $client]);

    if (empty($inscriptions)) {
        return new Response('Aucune inscription trouvée pour ce client.', Response::HTTP_NOT_FOUND);
    }

    $result = [];

    foreach ($inscriptions as $inscription) {
        // Récupérer les informations de l'inscription
        $inscriptionData = [
            'id_inscription' => $inscription->getId(),
            'date_inscription' => $inscription->getDateInscription()->format('Y-m-d'),
            'statut_inscription' => $inscription->getStatutInscription(),
            'nombre_inscrits' => $inscription->getNombreInscrits(),
            'montant' => $inscription->getMontant(),
            'participants' => []
        ];
        ...
        // Récupérer les participants associés à l'inscription
        foreach ($inscription->getParticipants() as $participant) {
            $inscriptionData['participants'][] = [
                'nom' => $participant->getNom(),
                'prenom' => $participant->getPrenom(),
                'date_de_naissance' => $participant->getDateDeNaissance()->format('Y-m-d'),
            ];
        }
        ...
    }
}

async function loadReservations(customerId, token) {
    const reservationSection = document.getElementById('reservation-section');

    try {
        const response = await fetch(`https://127.0.0.1:8000/api/recap-inscriptions-client/${customerId}`, { //MODE DEV
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${token}`,
                'Content-Type': 'application/json',
                'Accept': 'application/json',
            }
        });

        // Vérifier si la réponse est du JSON
        const contentType = response.headers.get("Content-Type");

        if (contentType && contentType.includes("application/json")) {
            // Si la réponse est JSON, on la traite comme telle
            const data = await response.json();
            console.log('Données récupérées :', data);

            if (data.length === 0) {
                // Si aucune inscription n'a été trouvée
                reservationSection.textContent = 'Aucune réservation effectuée pour le moment.';
            } else {
                // Afficher les inscriptions
                data.forEach(inscription => {
                    const inscriptionContainer = document.createElement('div');
                    inscriptionContainer.classList.add('inscription');

                    // Créer et ajouter les informations de l'inscription
                    const inscriptionNumero = document.createElement('p');
                    inscriptionNumero.textContent = `Inscription n° : ${inscription.id_inscription}`;
                    inscriptionContainer.appendChild(inscriptionNumero);

                    const dateInscription = document.createElement('p');
                    // Formatage de la date en format régional
                    const date = new Date(inscription.date_inscription).toLocaleDateString();
                    dateInscription.textContent = `Date d'inscription : ${date}`;
                    inscriptionContainer.appendChild(dateInscription);
                });
            }
        }
    }
}
```

# Les Réervations

## • Côté Admin

### Créer un nouvel événement

Statut  
Ouvert

Nom de l'événement  
Séance collective spéciale remise en selle seniors

Description de l'événement  
Vous voulez vous remettre au vélo ? Participez à une séance en groupe avec d'autres seniors, dans une ambiance conviviale, pour vous

Lieu/Adresse  
Stade des Coquelicots, 59000 Lille

Nombre de participants maximum  
25

Type de prestation  
Séance Collective Adultes/Seniors

Tarif  
100

Type de tarif  
Par participant

Public = cocher / Privé = décocher

Photo/Illustration (optionnelle)  
Parcourir... prestations\_adulte1.webp



Annuller la photo

Jour(s) de l'événement  
Ajouter un jour

Selectionner un jour et des heures de prestations  
Date  
14/12/2024

Heure de début  
09:00

Heure de fin  
11:00

Annuler

Enregistrer l'événement

### Récapitulatif des participants

Nombre total d'inscrits : 2

Télécharger la liste des inscrits

Nom : Dupont  
Prénom : Rémi  
Âge : 5 ans  
Statut inscription : Validée  
N° d'inscription : 54

Détails de la réservation

Nom : Dupont  
Prénom : Joséphine  
Âge : 9 ans  
Statut inscription : Validée  
N° d'inscription : 54

Détails de la réservation

L'Administrateur peut créer des événements depuis son espace.

Il peut choisir de les ouvrir à la réservation en ligne, ou leur affecter un autre statut (en attente, par exemple)

Il renseigne les informations et les jours et horaires. Cela alimentera deux tables : Evenement et JourEvenement.

### Extrait du formulaire :

```
class EvenementType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('statut', ChoiceType::class, [
                'label' => "Statut",
                'choices' => [
                    'En attente' => 'attente',
                    'Ouvert' => 'ouvert',
                    'Clôture' => 'clôturé',
                    'Annulé' => 'annulé',
                ],
            ]);
    }
}
```

### Extrait de la mise en page Twig :

```
{% block body %}
<div class="container mt-5">
    <h1 class="mb-4">Créer un nouvel événement</h1>

    {{ form_start(evenementForm, { 'attr': { 'class': 'form-horizontal', 'enctype': 'multipart/form-data' } }) }}

    <div class="form-group">
        {{ form_label(evenementForm.statut, null, { 'attr': { 'class': 'form-label' } }) }}
        {{ form_widget(evenementForm.statut, { 'attr': { 'class': 'form-control' } }) }}
        {{ form_errors(evenementForm.statut) }}
    </div>
```

L'Admin peut également bien sûr, consulter, modifier et supprimer ses événements.

### Liste des événements

Nom	Type	Lieu	Public/ Privé	Date Début	Date Fin	Actions
Séance collective spéciale remise en selle seniors	Séance Collective Adultes/Seniors	Stade des Coquelicots, 59000 Lille	Public	ouvert	14/12/2024	<a href="#">Voir/ Modifier</a>
Apprendre à rouler sans les petites roues !	Stage Vacances	Parking sécurisé de la rue des ponts, à Nantes	Public	ouvert	18/11/2024	<a href="#">Voir/ Modifier</a>

Ce qui est intéressant pour lui, en plus de tout cela, c'est que quand il consulte un événement, il y voit aussi les inscrits, c'est-à-dire, quels participants ont été inscrits par les clients à chacun de ses événements, ce qui fait appel à 3 autres tables : inscription, participant et client.

Quand il clique sur "Télécharger la liste des inscrits", cela génère un fichier PDF, récapitulant les éléments les plus importants et les coordonnées des clients, grâce à Dompdf :

Inscriptions					
Nombre total de participants : 4					
N° Inscr.	Nom du client	Coordonnées	Participants	Nb Inscr.	Statut
54	Marchand Amandine	amandine@gmail.com 0625252525	Rémi Dupont (5 ans) Joséphine Dupont (9 ans)	2	Validée
55	Flanflan Pimpreline	test@test.fr 0610101010	Jean Flanflan (23 ans) Jeanette Piqueline (15 ans)	2	Validée

```

class PdfController extends AbstractController
public function generatePdf(int $id, EvenementRepository
    // ...
    // Préparer les options de Dompdf
    $options = new Options();
    $options->set('defaultFont', 'Courier');

    // Créer une instance de Dompdf
    $dompdf = new Dompdf($options);

    // (Optionnel) Configurer la taille et l'orientation du papier
    $dompdf->setPaper('A4', 'landscape');

    // Générer le HTML pour le PDF
    $html = $this->renderView('pdf/evenement.html.twig', [
        'evenement' => $evenement,
        'inscriptions' => $inscriptions,
        'totalParticipants' => $totalParticipants,
    ]);

    // Charger le HTML dans Dompdf
    $dompdf->loadHtml($html);

    // Rendre le PDF
    $dompdf->render();

    // Retourner le PDF en tant que réponse
    return new Response($dompdf->output(), 200, [
        'Content-Type' => 'application/pdf',
        'Content-Disposition' => 'attachment; filename="evenement_' .
            $id . '.pdf"',
    ]);
}

```

## • Côté Client

L'ensemble des séances ouvertes pas encore passées, sont listées sur la page d'accueil des réservations :

**Séances collectives pour les particuliers**

Inscrivez-vous et participez à l'une des prochaines sessions en groupe.

**Apprendre à rouler sans les petites roues !**

C'est l'objectif de ce stage de 2 jours. Réussir à rouler sans les petites roues. Dès 4 ans.

Le 18/11/2024 au 19/11/2024

[En savoir plus](#)

**Séance collective spéciale remise en selle seniors**

Vous voulez vous remettre au vélo ? Participez à une séance en groupe avec d'autres seniors, dans une ambiance conviviale, pour vous remettre en selle.

Le 14/12/2024

[En savoir plus](#)

**Apprendre à rouler sans les petites roues !**

C'est l'objectif de ce stage de 2 jours. Réussir à rouler sans les petites roues. Dès 4 ans.

Lieu : Parking sécurisé de la rue des ponts, à Nantes

95€ (prix participant)

Cela comprend 2 séances :

Le 18/11/2024 de 10:00 à 13:00  
Le 19/11/2024 de 10:00 à 13:00

Places restantes : 11 participants

[S'inscrire](#)

En cliquant sur "en savoir plus", cela bascule sur le détail de l'événement récupéré en fetch grâce à son id :

```
moreInfoButton.href = `events_details.php?id=${evenement.id}`;
```

localhost:8086/events/events\_details.php?id=18

Pour calculer le nombre de places encore disponibles, il y a d'abord une fonction qui permet de compter le nombre d'inscrits actuels :

```
const inscritCount = await fetchInscriptionCount(eventId);
async function fetchInscriptionCount(eventId) {
    const response = await fetch(`https://127.0.0.1:8000/api/evenement/${eventId}/
    inscrits`);

    // Vérifie si la réponse est OK
    if (!response.ok) {
        console.error('Erreur lors de la récupération des inscriptions:', response.
        statusText);
        return 0;
    }
    const data = await response.json();
    return data.nombreInscrits; // Retourne le nombre total d'inscrits
}
```

Cette donnée est ensuite reprise dans la fonction qui récupère toutes les infos de l'événement. Si jamais il n'y a plus de places, alors un message apparaît pour le signaler et bouton "s'inscrire" passe à "me contacter" avec un changement de lien href pour rediriger vers la bonne page :

```
function displayEventDetails(event, jours, inscritCount) {
    // ... //

    // Places restantes
    const placesRestantes = document.createElement('p');
    const calculPlacesRestantes = event.capacite_max - inscritCount; // Calcule le
    nombre de places restantes

    // Bouton "Inscriptions"
    const inscriptionButton = document.createElement('a');
    inscriptionButton.className = 'inscription-button';

    if (calculPlacesRestantes <= 0) {
        placesRestantes.textContent = "Complet. Veuillez me contacter pour étudier
        une autre possibilité.";
        inscriptionButton.href = 'http://localhost:8086/contact.php'; // MODE DEV
        inscriptionButton.textContent = "Me contacter";

    } else {
        placesRestantes.textContent = `Places restantes : ${calculPlacesRestantes}
        participant${calculPlacesRestantes > 1 ? 's' : ''}`;
        inscriptionButton.href = `inscription.php?id=${event.id}`;
        inscriptionButton.textContent = "S'inscrire";
    }
    container.appendChild(placesRestantes);
    container.appendChild(inscriptionButton);
```

L'inscription ne peut se faire que par un client connu et connecté (vérification du token valide). Elle se fait ensuite pour l'instant via un formulaire où le client renseigne simplement les nom, prénom et date de naissance du ou des participants. Il peut en rajouter autant qu'il souhaite dans la limite du nombre de places restantes disponibles.

Le prix total est mis à jour automatiquement en fonction du nombre de participants et si le prix est unique ou par participant :

```
// Fonction pour mettre à jour le prix total
const updatePrice = () => {
    if (typeTarif === "participant"){
        totalPrice = participantCount * prixUnitaire;
    } else {
        totalPrice = prixUnitaire;
    }

    totalPriceElem.textContent = `Prix total : ${totalPrice} €`;
};
```

**Apprendre à rouler sans les petites roues !**

Lieu : Parking sécurisé de la rue des ponts, à Nantes  
2 séances :  
le 18/11/2024 de 10:00 à 13:00  
le 19/11/2024 de 10:00 à 13:00  
Places restantes : 11 participants

**Formulaire d'inscription**

Ajoutez ici les participants :

**Participant 1**

Nom :   
Prénom :   
Date de naissance :

**Participant 2**

Nom :   
Prénom :   
Date de naissance :

Prix total : 190 €

```
// Vérifie si l'événement est complet
if (placesRestantesValue <= 0) {
    // Cache le bouton d'ajout de participants et la zone de formulaire
    addParticipantBtn.style.display = 'none';
    participantsContainer.style.display = 'none';

    // Cache le total € et le bouton de validation de l'inscription
    const totalContainer = document.getElementById('total-container');
    totalContainer.style.display = "none";

    // Crée et affiche un bouton "Me contacter"
    const contactButton = document.createElement('a');
    contactButton.className = 'contact-button';
    contactButton.href = 'http://localhost:8086/contact.php';
    contactButton.textContent = 'Me contacter';
    participantsContainer.parentElement.appendChild(contactButton); // Ajoute le bouton après le container parent

    return; // Arrête l'initialisation du formulaire si l'événement est complet
}

// Fonction pour ajouter un participant
addParticipantBtn.addEventListener('click', () => {
    // Vérifie si le nombre de participants actuel est inférieur aux places restantes
    if (participantCount > placesRestantesValue) {
        alert("Nombre maximum de participants atteint ! Veuillez me contacter pour étudier une autre possibilité.");
        return;
    }
    participantCount++;
    const participantDiv = document.createElement('div');
    participantDiv.classList.add('participant');
    participantDiv.setAttribute('data-participant-id', participantCount);

    // Création des éléments
    const title = document.createElement('h4');
    title.textContent = `Participant ${participantCount}`;
    // ... etc ... //
}
```

Les données sont ensuite envoyées via un fetch vers les tables de données. Extrait :

```
form.addEventListener('submit', async (event) => {
    const inscriptionUrl = 'https://127.0.0.1:8000/api/inscriptions/${inscriptionId}'; // MODE DEV

    // Envoyer les participants
    for (const participant of participants) {
        const participantData = {
            id_inscription: inscriptionUrl,
            id_client: clientUrl,
            nom: participant.nom,
            prenom: participant.prenom,
            date_de_naissance: participant.date_de_naissance
        };

        const participantResponse = await fetch('https://127.0.0.1:8000/api/participants', { // MODE DEV
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify(participantData)
        });

        if (!participantResponse.ok) {
            loader.style.display = 'none';
            throw new Error("Erreur lors de l'ajout du participant.");
        }
    }
});
```

Le client bascule sur une page avec un formulaire de paiement en ligne, configuré avec Stripe.

<https://127.0.0.1:8000/payment?inscriptionId=87&montant=100&token=eyJ0eXAiOiJKV1QiLCJhbGciOiJS...>

Paiement par carte

Montant à régler : 100 €

Numéro de carte  
1234 1234 1234 1234

Date d'expiration  
MM / AA

CVC  
CVC

Payer

[Annuler la transaction] Cela supprimera définitivement vos données d'inscription.

L'id de la réservation, le montant et le token d'authentification sont contenus dans le lien URL.

Une fois I

Une fois le paiement validé, le client reçoit un mail récapitulatif généré via un contrôleur Symfony :

From <contact@michael-moniteur-cycliste.fr>  
To <test@test.fr>  
Subject Confirmation d'inscription à : Apprendre à rouler sans les petites roues !  
Date Tue, 15 Oct 2024, 5:13 pm (3 kB)  
Tags Add tags...

HTML HTML Source Text Headers Raw HTML Check Link Check



**Confirmation d'inscription**

Merci pour votre inscription à l'événement Apprendre à rouler sans les petites roues !

Lieu : Parking sécurisé de la rue des ponts, à Nantes

Détails de l'événement

18/11/2024 de 09:00 à 12:00  
19/11/2024 de 09:00 à 12:00

Participants enregistrés

Tata Toto - Date de naissance : 06/08/2009  
Tutu Tata - Date de naissance : 12/04/2010

Montant total : 190 €

Pour plus d'informations, retrouvez la page de l'événement à : [cet endroit](#)  
Ou accédez à votre espace client : [ici](#)

```
#Route('/api/send-confirmation-email/{inscriptionId}', name: 'send_confirmation_email', methods: ['POST'])
public function sendConfirmationEmail(Int $inscriptionId, MailerInterface $mailer, IncriptionRepository $inscriptionRepository, Environment $twig): Response {
    // Récupérer l'inscription par ID
    $inscription = $inscriptionRepository->find($inscriptionId);

    if (!$inscription) {
        return new Response('Inscription non trouvée.', Response::HTTP_NOT_FOUND);
    }

    // Récupérer les informations de l'événement
    $evenement = $inscription->getIdEvenement();

    // Récupérer les participants associés à l'inscription
    $participants = [];
    foreach ($inscription->getParticipants() as $participant) {
        $participants[] = [
            'nom' => $participant->getNom(),
            'prenom' => $participant->getPrenom(),
            'date_de_naissance' => $participant->getDateDeNaissance(),
        ];
    }

    // Récupérer les jours et heures d'événement
    $joursEvenements = [];
    foreach ($evenement->getJourEvenements() as $jourEvenement) {
        $joursEvenements[] = [
            'jour' => $jourEvenement->getDateJour(),
            'heure_debut' => $jourEvenement->getHeureDebut(),
            'heure_fin' => $jourEvenement->getHeureFin(),
        ];
    }

    // Rendre le template avec les données
    $htmlContent = $twig->render('inscription/mail_inscription.html.twig', [
        'evenement' => [
            'id' => $evenement->getId(),
            'nom' => $evenement->getNom(),
            'lieu' => $evenement->getLieu(),
        ],
        'joursEvenements' => $joursEvenements,
        'participants' => $participants,
        'montant' => $inscription->getMontant(),
    ]);

    // Créer l'email de confirmation
    $email = (new Email())
        ->from('contact@michael-moniteur-cycliste.fr')
        ->to($inscription->getIdClient()->getEmail())
        ->subject("Confirmation d'inscription à : " . $evenement->getNom())
        ->html($htmlContent);

    // Envoyer l'email
    $mailer->send($email);

    return new Response('Email de confirmation envoyé avec succès.', Response::HTTP_OK);
}
```

## L'Accessibilité

---

Dans tout le projet, j'ai veillé à bien respecter les **structures et balises types HTML**.

J'ai contrôlé la structure à l'aide d'outils comme l'extension du navigateur **HeadingsMap** qui liste le nombre de balises en <h..> sur chaque page.



Et j'ai également lancé des scans de chaque page avec **LightHouse** sur le navigateur, pour avoir une évaluation et les points à retravailler en termes d'accessibilité, performance, mais aussi de SEO :



Toutes les images et photos de grande taille ont été converties en **format .webp**, plus léger pour la navigation web.

## Le SEO

---

En plus du **respect des structures et balises HTML**, j'ai recherché les autres bonnes pratiques pour le référencement.

J'ai utilisé les **balises <meta>** pour renseigner la description et les caractéristiques de chaque page, en essayant d'intégrer un maximum de mots clés que les utilisateurs pourraient taper dans leur barre de recherche. Les **balises type "og"** servent pour le partage de la page sur Facebook.

```
<meta property="og:type" content="website">
<meta property="og:url" content="https://michael-moniteur-cycliste.fr/">
<meta property="og:title" content="Michaël, Moniteur Cycliste, dans le Nord-Pas-de-Calais 59/62, autour de Lille, Armentières, Béthune, Lens, Hazebrouck ">
<meta property="og:description" content="Apprendre à rouler à vélo ou se remettre en selle, avec Michaël, Moniteur Cycliste qualifié, référencé Moniteur Cycliste Français. Apprentissage du vélo tout public, pour les enfants, adultes et pour les écoles/associations.">
<meta property="og:image" content="https://michael-moniteur-cycliste.fr/assets/images/logo_michael_soetens.png">
<meta name="language" content="fr">
<meta name="description" content="Apprendre à rouler à vélo ou se remettre en selle, avec Michaël, Moniteur Cycliste qualifié, référencé Moniteur Cycliste Français, dans le Nord-Pas-de-Calais 59/62, autour de Lille, Armentières, Béthune, Lens, Hazebrouck. Apprentissage du vélo tout public, pour les enfants, adultes et pour les écoles/associations.">
```

J'ai paramétré et j'utilise **Google Search Console** et **Google Analytics**, deux outils de monitoring de performance et de fréquentation du site.

Je m'en suis servie pour améliorer le référencement du site. Je trouve que ce sont des outils très intéressants, qui permettent de paramétrer toute sorte de mesures pointues, en plus de toutes les statistiques déjà proposées de base.

#### *Paramétrage du Tag de Google Manager sur chaque page du site :*

```
<!-- Google Tag Manager (noscript) -->
<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-5CHZH528"
height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>
<!-- End Google Tag Manager (noscript) -->
```

J'ai mis en place un fichier **sitemap.xml** qui permet à Google d'avoir un visu plus précis de l'architecture du site avec une priorité à accorder à chaque page.

#### *Extrait du fichier sitemap.xml :*

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset
  xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

  <url>
    <loc>https://www.michael-moniteur-cycliste.fr/</loc>
    <lastmod>2024-07-30</lastmod>
    <changefreq>monthly</changefreq>
    <priority>1.0</priority>
  </url>
  <url>
    <loc>https://www.michael-moniteur-cycliste.fr/qui_suis_je.php</loc>
    <lastmod>2024-07-30</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.5</priority>
  </url>
</urlset>
```

J'ai ajouté le lien du site de Michaël sur mon portfolio en ligne, et demandé à Michaël d'en faire de même sur sa page de contact du site "Moniteur Cycliste Français" dont il fait partie. En effet, les **liens externes** aident à augmenter la crédibilité et le référencement du site dans les moteurs de recherche.

## Le Déploiement

---

### • L'Hébergement

J'ai accompagné Michaël, dans le choix de l'hébergement. J'ai consulté les avis récents, comparé les offres pour lui proposer une liste d'hébergeurs fiables et appropriés, au bon rapport qualité-prix comme O2Switch, LWS et Hostinger. C'est vers ce dernier qu'il s'est tourné.

### • 1<sup>ère</sup> phase : le déploiement du site vitrine

Déployer le site vitrine a été rapide et facile à effectuer, car j'avais déjà eu l'expérience de mettre en ligne 2 sites. J'ai utilisé FileZilla pour la bascule des fichiers vers le serveur (connexion FTP).

J'ai effectué une veille et des tests, les semaines qui suivaient, pour améliorer le référencement du site sur Google. J'ai modifié ou complété le contenu des balises `<meta>` en fonction.

### • 2<sup>ème</sup> phase : le déploiement de la page Actualités et l'Espace Administrateur

Pour plus de simplicité et une indépendance d'accès, j'ai déployé l'Espace Administrateur dans un sous-domaine.

Pour ce premier déploiement de back-end, sur un site existant, je me suis retrouvée confronter à plusieurs difficultés et nouveautés à appréhender : la création de la base de données sur le serveur d'Hostinger, l'accès SSH, le gestionnaire de cache Hostinger versus le cache à gérer directement avec la connexion SSH, la mise à jour des dépendances, le .htaccess... Avec cette expérience, je suis prête pour la 3<sup>ème</sup> phase !

- **3<sup>ème</sup> phase : le déploiement de la page Réservation et du Compte Client**

Il sera effectué d'ici fin novembre.

(Documentation sur le Déploiement en Annexe 2)

## Conclusion

Toute cette période est passée si vite ! Et l'apprentissage intense !

C'était vraiment l'occasion de tester un maximum de choses, je l'ai fait et je ne regrette pas.

Cela m'a permis de voir que j'arrivai un peu à toucher à tout, ce qui est primordial pour la suite de ma carrière. Je n'ai pas de préférence de langages, mais plutôt des préférences de méthodes.

Avec le recul, je pense que j'adopterai pour la conception de mes prochains projets la démarche "mobile first" qu'utilisent les Designers Web, c'est-à-dire penser d'abord au projet version mobile pour ensuite l'étendre en mode "écrans d'ordinateur". Je trouve qu'il est plus facile d'agrandir que de réduire. Et également, alléger la mise en place des Media-Queries avec peut-être des conceptions plus aérées, facilitant la mise en page.

Côté code, même si j'apprécie particulièrement coder en natif pour "avoir la main sur tout", il faut bien avouer que l'utilisation de frameworks puissants et fiables comme Symfony, est un vrai gain de temps et j'ai apprécié certaines de ses fonctions intégrées.

La tendance étant également à gagner toujours plus de vitesse et de performance, pour rivaliser avec celles des applications, j'opterai pour la conception en "Single Page Applications", comme le permet par exemple React, que j'ai eu l'occasion de tester dans d'autres projets.

Il me faut penser davantage : "efficacité et simplicité".

## Annexes

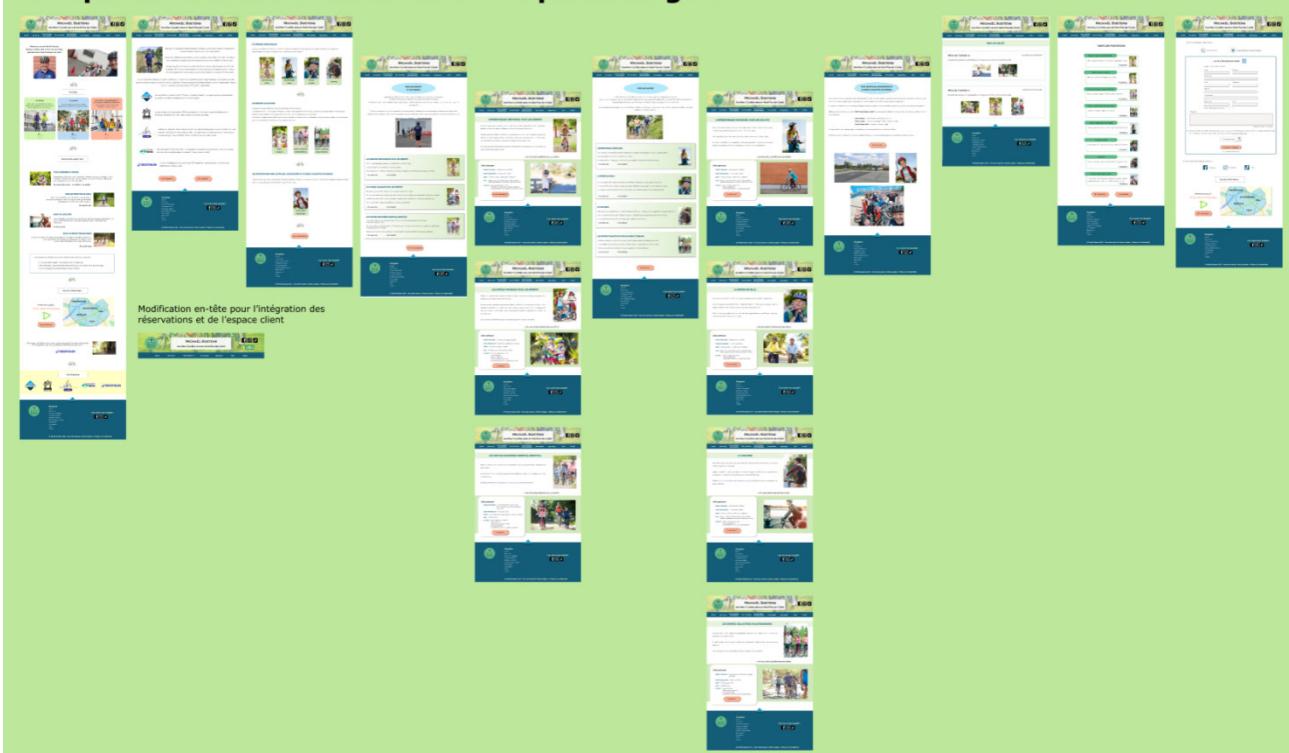
### Annexe 1 : Maquettes du site

#### Maquette version mobile 360px de largeur



Modification en-tête pour  
l'intégration de l'espace client

#### Maquette version ordinateur 1440 px de largeur



## Documentation de Déploiement du site "Michaël Moniteur Cycliste"

Cette documentation explique le processus de déploiement du site web "Michaël Moniteur Cycliste".

Le site est développé dans deux projets : un front sous HTML/JS/CSS et un back sous Symfony.

La base de données est gérée avec MySQL.

### Prérequis

Avant de commencer, il faut avoir ces éléments installés et configurés :

- Docker et Docker Compose
- Git
- Un hébergeur/serveur à distance, avec accès SSH
- Composer (pour la gestion des dépendances PHP)
- Filezilla

### Configuration de l'Environnement

#### 1. Clone des Répertoires Git

```
git clone https://github.com/marchandamandine/Front\_Michael\_Moniteur\_Cycliste  
git clone https://github.com/marchandamandine/Back\_Michael\_Moniteur\_Cycliste
```

#### 2. Configuration des Variables d'Environnement

Mettre à jour les fichiers ` `.env` pour configurer les variables d'environnement spécifiques à la production.

DATABASE\_URL  
MYSQL\_DATABASE  
MYSQL\_USER  
MYSQL\_PASSWORD  
MYSQL\_ROOT\_PASSWORD  
MAILER\_DSN  
APP\_ENV

#### 3. Modifier tous les "localhost" et "127.0.0.1" avec les vrais URL du site.

#### 4. Renseigner les vrais clés publiques et privées pour Stripe, à la place des clés tests.

#### 5. Contrôler les données du .htaccess pour bloquer les accès aux fichiers sensibles.

### Déploiement

1. Installation/Mise à jour des Dépendances dans Symfony avec : composer update
2. Vider le cache Symfony : php bin/console cache:clear
3. Configuration PHP : sélectionner chez l'hébergeur la version PHP utilisée.
4. Basculer les fichiers avec une connexion FTP, via Filezilla.

Se connecter avec le nom d'hôte, le nom utilisateur, mot de passe et port fournis par l'hébergeur. Basculer les fichiers dans le dossier source : public\_html (ou www selon l'hébergeur)

5. Créer la base de données chez l'hébergeur depuis leur interface, puis réaliser une migration.  
Vérifier dans l'interface prévue par l'hébergeur (phpMyAdmin par exemple) que les tables ont bien été créées.
6. Se connecter en SSH, et vider le cache Symfony.

### **Après le déploiement**

Faire quelques tests fonctionnels sur le site. Si des anomalies sont constatées, il est possible soit de modifier les fichiers et les rebasculer dans Filezilla ou directement les modifier dans le Gestionnaire de Fichiers disponible sur la page de l'hébergeur. Penser à vider le cache, avant toute recherche.