



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL(DP)

*Nom de naissance*

► MARCHAND

*Nom d'usage*

►

*Prénom*

► Amandine

*Adresse*

► 15 rue des pensées  
62150 Fresnicourt-le-Dolmen

## Titre professionnel visé

Développeur(se) Web et Web Mobile

### MODALITE D'ACCES :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)

## Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.  
**Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.  
Il est consulté par le jury au moment de la session d'examen.

### Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

*[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]*

### Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

*Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.*



<http://travail-emploi.gouv.fr/titres-professionnels>

# DOSSIER PROFESSIONNEL(DP)

## Sommaire

### Exemples de pratique professionnelle

<b>Intitulé de l'activité-type n° 1 :</b> <b>Développer la partie front-end d'une application web ou web mobile sécurisée</b>	p.	5
► Intitulé de l'exemple n° 1 : Installer et configurer l'environnement de travail .....	p.	5
► Intitulé de l'exemple n° 2 : Créer les maquettes du projet.....	p.	7
► Intitulé de l'exemple n° 3 : Réaliser des interfaces statiques .....	p.	10
► Intitulé de l'exemple n° 4 : Réaliser des interfaces dynamiques .....	p.	13
<b>Intitulé de l'activité-type n° 2 :</b> <b>Développer la partie back-end d'une application web ou web mobile sécurisée</b>	p.	16
► Intitulé de l'exemple n° 1 : Réaliser une base de données relationnelle .....	p.	16
► Intitulé de l'exemple n° 2 : La page "Actualités" gérée par l'Administrateur.....	p.	18
► Intitulé de l'exemple n° 3 : la Gestion de la Sécurité sur le site.....	p.	21
► Intitulé de l'exemple n° 4 : le Déploiement du Site Web.....	p.	23
<b>Titres, diplômes, CQP, attestations de formation</b>	p.	25
<b>Déclaration sur l'honneur</b>	p.	26
<b>Documents illustrant la pratique professionnelle</b>	p.	27
<b>Annexes</b>	p.	28

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 1

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

*Exemple n°1> Installer et configurer l'environnement de travail*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Avant de lancer le projet, je me suis posée les questions suivantes : quels langages je vais utiliser ? Et de quels outils je vais avoir besoin ? C'était important d'organiser tout cela en amont, pour essayer d'anticiper un maximum de besoins.

Avant d'entamer la formation DWWM, 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 bâtir une expérience, j'ai décidé de coder avec deux méthodes différentes.

La réalisation du site allait être faite en deux temps et a été codée dans deux projets différents, histoire de séparer la partie front de la partie back.

- 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.

### 2. Précisez les moyens utilisés :

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.



# DOSSIER PROFESSIONNEL(DP)

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.



## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Michaël Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 1

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

*Exemple n° 2 ► Créer les maquettes du projet*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

J'ai 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/>.

CHARTE GRAPHIQUE



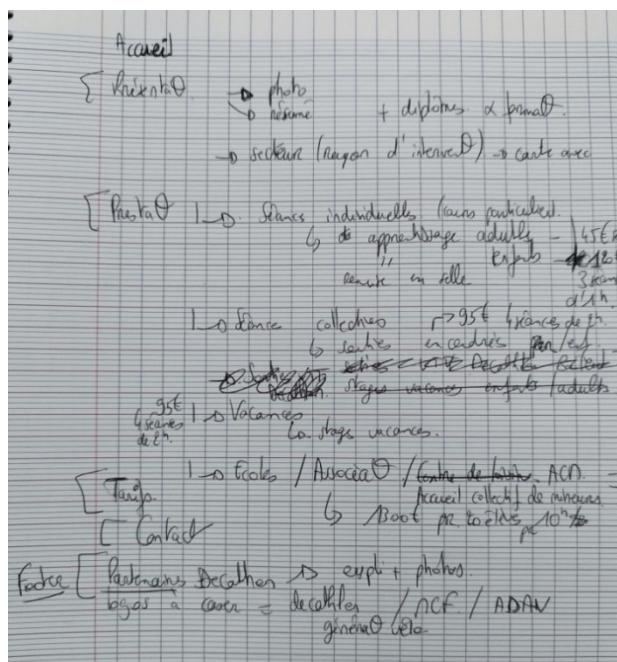
Les polices de caractères ont été choisies sur Google Fonts.

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.

Police :  
Entête nom en : Sedan  
Entête fonction sous titre en : Teachers  
Le reste des textes en : Verdana

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 :

# DOSSIER PROFESSIONNEL(DP)

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

## 2. Précisez les moyens utilisés :

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. Voici un extrait en version mobile et version web (maquettes complètes en annexe 1) :

# DOSSIER PROFESSIONNEL(DP)

Voici les changements qui seront faits au niveau de l'entête pour l'intégration dans un second temps de déploiement, des réservations et de l'espace client :



## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association▶ *Michaël Moniteur Cycliste*

Chantier, atelier, service ▶ Non concerné

Période d'exercice ▶ Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 1

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

*Exemple n° 3▶ Réaliser une interface statique*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

J'avais en tête de ne pas tout coder directement en HTML, mais de réaliser une conception assez dynamique, sous forme de sections HTML vides alimentées par des fonctionnalités en Javascript.  
Cette conception que j'avais mise en place assez spontanément, était "orientée objet".

Néanmoins, le squelette des certaines sections plus sensibles comme le formulaire de contact, a été réalisé en HTML directement.

#### 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>⇒</td> <td>Commun à tous</td>	⇒	Commun à tous
<link rel="stylesheet" href=" = SITE_URL_STYLES . 'tarifs.css'?&gt;"&gt;</td <td>⇒</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>⇒</td> <td>Commun</td>	⇒	Commun
<script src=" = SITE_URL . 'fonctions.js'?&gt;" type="module" defer&gt;&lt;/script&gt;</td <td>⇒</td> <td>Commun</td>	⇒	Commun
<script src=" = SITE_URL . 'tarifs.js'?&gt;" type="module" defer&gt;&lt;/script&gt;</td <td>⇒</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> :*

# DOSSIER PROFESSIONNEL(DP)

```
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="logoMMC" 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"> flex
        <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>
```

# DOSSIER PROFESSIONNEL(DP)

J'ai fait le choix de coder par exemple, le formulaire de contact 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". Extrait :

```
<section class="formulaire">
  <form action="" method="post" name="formulaire" id="formulaire">
    <div class="form_objet">
      <label for="objet">Objet*</label>
      <select name="objet" id="objet">
        <option value="selectionner" id="objetAMasquer" selected>Sélectionner un objet...</option>
        <option value="Réserver une prestation">Réserver une prestation</option>
        <option value="Demande de devis">Demander un devis</option>
        <option value="Demande de renseignements">Demander des renseignements</option>
        <option value="Autre demande">Autre demande</option>
      </select><br>
      <span id="error-message-objet" class="error-message"></span>
    </div>

    <div class="form_message">
      <p><label for="message" id="titreMessage">Message*</label></p>
      <span id="error-message-message" class="error-message"></span>
      <div class="zoneMessage">
        <textarea id="message" placeholder="Votre message ..." name="message" rows="10" cols="100"
          maxlength=1000></textarea>
        <p id="nombreCaracteresRestants">Caractères restants : 1000/1000</p>
      </div>
    </div>
  </form>
</section>
```

## 2. Précisez les moyens utilisés :

J'ai utilisé les langages Javascript, HTML et CSS pour la réalisation des parties statiques.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Michaël Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 1

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

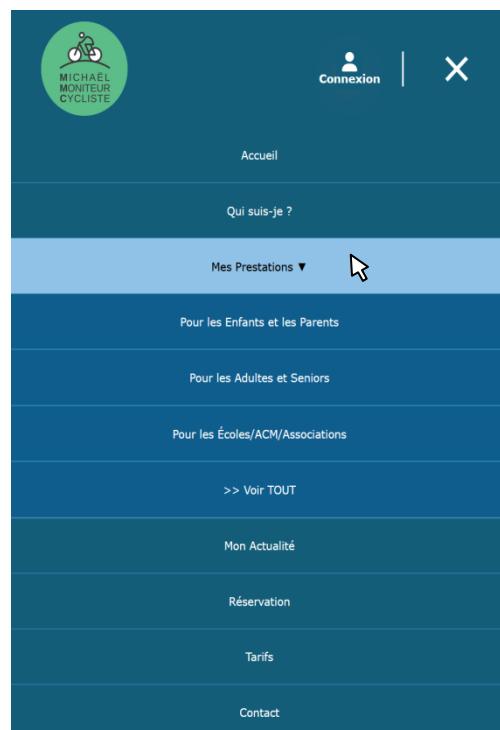
*Exemple n° 4 ► Réaliser des interfaces dynamiques*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Le site est totalement Responsive et j'ai utilisé beaucoup de fonctions d'écoute en Javascript pour déclencher certaines actions au click par exemple ou au lancement du DOM.

### 2. Précisez les moyens utilisés :

J'ai utilisé les Media Queries CSS pour adapter la mise en page aux différentes tailles d'écran :  
En dessous de 900px, l'image de fond de l'entête 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 :



L'URL de l'image de fond se supprime grâce à une fonction de surveillance de taille d'écran en Javascript :

```
const imageFondMediaQuery = window.matchMedia( '(max-width : 900px)' );

function changementImageFond() {
if (imageFondMediaQuery.matches === false) {
    header.style.backgroundImage = `url(${URLImages}header_fond.webp)`;
} else if (imageFondMediaQuery.matches === true) {
    header.style.backgroundImage = 'none';
}
changementImageFond();
imageFondMediaQuery.addEventListener('change', changementImageFond);
```

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 :

# DOSSIER PROFESSIONNEL(DP)

```

@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 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;
}

```

Je me suis servie également des dispositions FLEX et GRID dans tout le projet :

Pour ces 3 éléments, 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, le Flex passe 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;
    }
}

```

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



```

.role1,
.role2,
.role3,
.role4{
    display: grid;
}
.role1Photo,
.role3Photo{
    grid-column: 1/2;
    grid-row: 1/2;
    margin-right:20px;
}
.role1Texte,
.role3Texte{
    grid-column: 2/3;
    grid-row: 1/2;
    text-align: left;
}
.role2,
.role4{
    grid-template-columns: 70% 20%;
    margin-left:50px;
    margin-right:-70px;
}
.role2Photo,
.role4Photo{
    grid-column: 2/3;
    grid-row: 1/2;
    margin-left:20px;
}
.role2Texte,
.role4Texte{
    grid-column: 1/2;
    grid-row: 1/2;
    text-align: right;
}

```

J'ai exploité les listeners en JS pour déclencher certaines actions. 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 :

# DOSSIER PROFESSIONNEL(DP)

```
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';
    });
});

/* Style des rayons de la roue */
.rayon {
  position: absolute;
  width: 3px;
  height: 50px;
  background: #ffffff;
  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); }

/* Mise en forme du loader */
#loader {
  position: fixed;
  left: 0;
  top: 0;
  width: 100%;
  height: 100vh;
  background: linear-gradient(0deg, transparent 45px, black 45px, black 55px, transparent 55px);
  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 linear-gradient(45deg, transparent 5px, black 5px, black 15px, transparent 15px);
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

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



## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association▶

*Michaël Moniteur Cycliste*

Chantier, atelier, service



Non concerné

Période d'exercice



Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 2

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

*Exemple n° 1 ► Réaliser une base de données relationnelle*

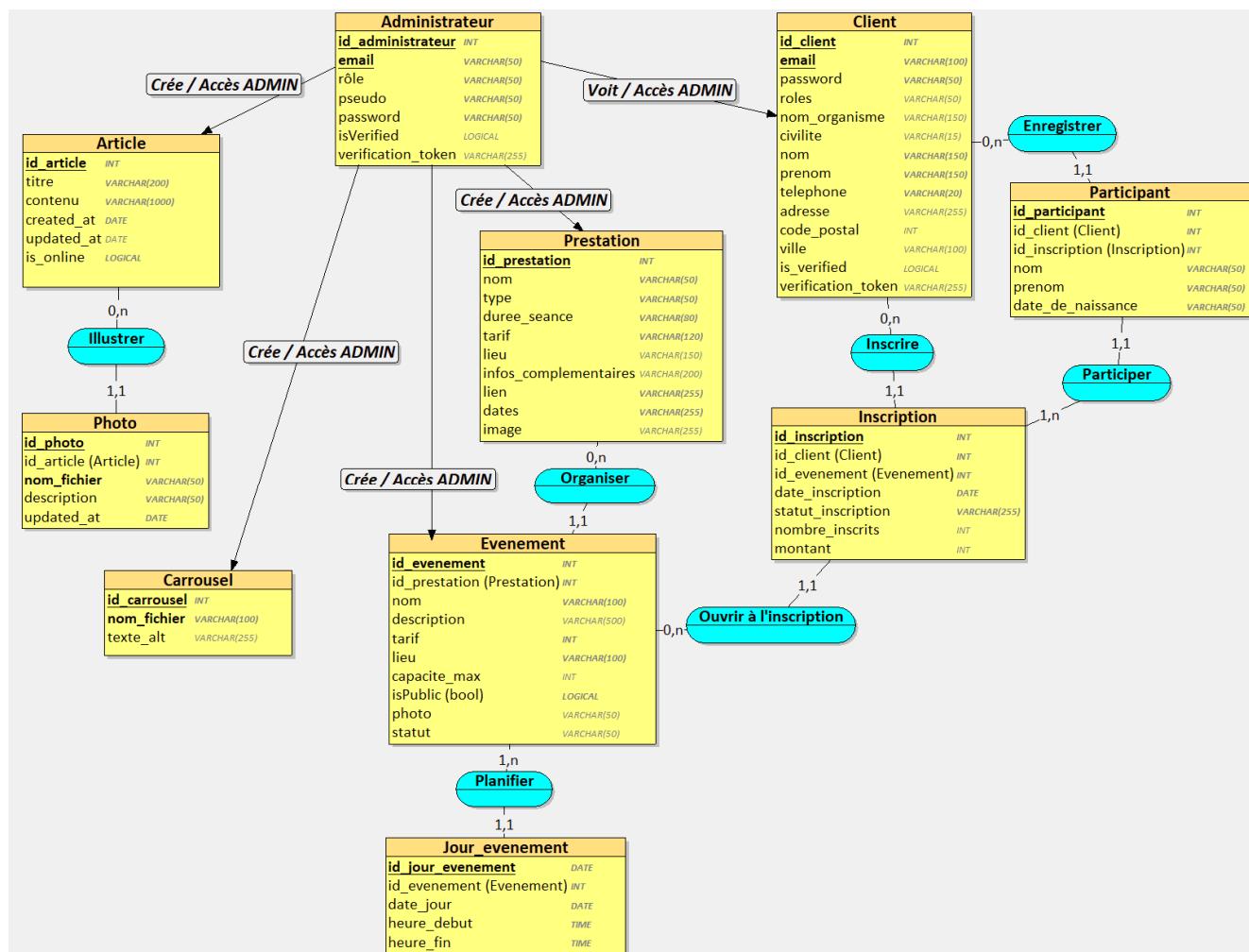
### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour gérer toute la partie back-end, j'ai créé un modèle de base de données relationnelle.

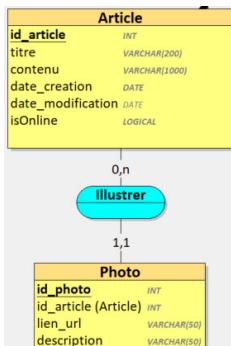
Le principe de conception est le suivant : Michaël est seul gérant de son entreprise, il pilotera tout, avec un accès spécial administrateur. Et le client pourra créer sur le site, son compte en ligne et inscrire des participants à des événements.

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.



# DOSSIER PROFESSIONNEL(DP)



Regardons de plus près ces deux entités Article et Photo et leur relation. Chaque article peut ne contenir aucune photo ou il peut en contenir une ou plusieurs (0,n).

Chaque photo est, quant à elle, rattachée à 1 article unique (1,1).

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.

## 2. Précisez les moyens utilisés :

J'ai utilisé le logiciel Looping, pour réaliser le modèle conceptuel de base de données.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Michaël Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 2

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

**Exemple n° 2** La page "Actualités" gérée par l'Administrateur

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

La page "Actualités" sur le site, fonctionne sur le même principe qu'un blog. L'Administrateur doit pouvoir depuis son Espace Administrateur gérer son actualité : créer une nouvelle publication, consulter, modifier ou supprimer une publication existante.

L'Actualité fait référence à deux entités : Article (toute la partie textuelle de la publication) et Photo (reliée à Article, servant à l'Administrateur d'intégrer une seule photo ou une collection de photos à son article).

La route globale sur Symfony dédiée à l'actualité est /admin/article :

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

The form is titled "Créer une nouvelle publication". It has two input fields: "Titre" and "Contenu". A note below says: "Mise en ligne ? Si oui, cocher la case. Si non, ne pas cocher, les données seront enregistrées, mais pas publiées." There is a red "Ajouter une photo" button and a green "Sauvegarder" button.

Nouvelle publication : 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. (codes en Annexe 2)

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". (codes en Annexe 2bis)

The modal has a text input field labeled "Description de la photo" with the placeholder "à renseigner obligatoirement". Below it is a file input field with the placeholder "Télécharger une photo (3Mo max par photo)" and a "Parcourir..." button. A note at the bottom says "Aucun fichier sélectionné.". There is a red "Annuler" button and a red "Ajouter une photo" button.

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

# DOSSIER PROFESSIONNEL(DP)

```
##[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'
```

## Consultation d'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.

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>
```

(codes en Annexe 3)

## Modification d'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. (visuel et codes en Annexe 4)

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'
```

# DOSSIER PROFESSIONNEL(DP)

## Suppression d'une photo :

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. (code en Annexe 5)

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>
```

## **2. Précisez les moyens utilisés :**

J'ai utilisé les fonctionnalités de Symfony et des fonctions Javascript.

## **3. Avec qui avez-vous travaillé ?**

J'ai travaillé seule.

## **4. Contexte**

Nom de l'entreprise, organisme ou association ► *Michaël Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *15/04/2024* au : *26/10/2024*

## **5. Informations complémentaires (facultatif)**

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 2

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

*Exemple n° 3> La Gestion de la Sécurité sur le site*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour assurer la sécurité du site, j'ai utilisé différents moyens :

- Le test reCaptcha pour la validation d'envoi du formulaire de contact et pour la création d'un nouveau compte client,
- La sécurisation des formulaires via les contraintes intégrées au HTML (type, maxlength), des regex pour toutes les saisies et des fonctions JS qui écoutent les saisies et bloquent l'envoi du formulaire en cas de non-conformité,
- L'utilisation d'Asserts sur Symfony, pour valider les saisies dans les formulaires,
- Des mots de passe sécurisés selon les préconisations de la CNIL,
- "Htmlspecialchars" pour que les saisies de l'utilisateur ne soient pas interprétées comme du HTML (attaques XSS), utilisation de Twig pour la partie Symfony (échappement intégré),
- La génération et l'utilisation de Tokens :
  - en stockage en base de données lors de la création d'un nouveau client ou nouvel admin, le temps que ce nouvel utilisateur valide son compte,
  - en stockage en base de données lorsqu'un utilisateur réinitialise son mot de passe, un lien contenant le token lui est envoyé par mail lui permettant de modifier son mot de passe,
  - comme moyen d'authentification, en stockage en sessionStorage lorsqu'un utilisateur se connecte et qui lui permet d'accéder à certaines pages. Le sessionStorage limite l'exposition des données sensibles.
- Validation du nouveau compte client ou admin via un lien envoyé par mail, sinon il ne peut pas se connecter,
- Hashage du mot de passe en base de données,
- Sécurisation des routes API, gestion des accès en fonction des rôles et des connexions :

```
#[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  
{
```

- Utilisation de PHPMailer pour l'envoi de mails (sécurité intégrée)

# DOSSIER PROFESSIONNEL(DP)

## 2. Précisez les moyens utilisés :

J'ai exploité ce que les langages et fonctionnalités peuvent permettre de faire, en m'inspirant de conseils et préconisations actuels pour contrer le piratage.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Michael Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *15/04/2024* au : *26/10/2024*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL(DP)

## Activité-type 2

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

### Exemple n° 4 ► Le Déploiement du Site

#### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

- 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.

- 1ère 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.

- 2ème 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)

#### 2. Précisez les moyens utilisés :

J'ai utilisé les documentations et aides en ligne pour le déploiement.

#### 3. Avec qui avez-vous travaillé ?

J'ai déployé le site seule.

# DOSSIER PROFESSIONNEL(DP)

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Michael Moniteur Cycliste*

Chantier, atelier, service ► Non concerné

Période d'exercice ► Du : *01/07/2024* au : *26/10/2024*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL(DP)

## Titres, diplômes, CQP, attestations de formation

(*facultatif*)

Intitulé	Autorité ou organisme	Date
BTS Assistant de Direction	Lycée Anatole France, Lillers (62)	2006
Baccalauréat Littéraire	Lycée Lavoisier, Auchel (62)	2003

# DOSSIER PROFESSIONNEL(DP)

## Déclaration sur l'honneur

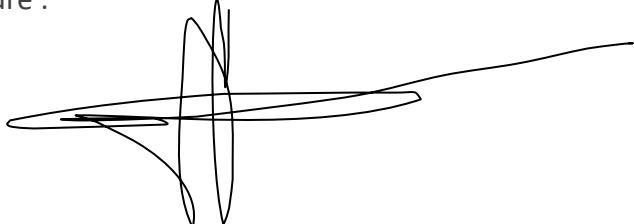
Je soussigné(e) [prénom et nom] Amandine MARCHAND .....

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis  
l'auteur(e) des réalisations jointes.

Fait à Fresnicourt-le-Dolmen le 17/10/2024

pour faire valoir ce que de droit.

Signature :



# DOSSIER PROFESSIONNEL(DP)

## Documents illustrant la pratique professionnelle

Intitulé
Annexe 1 : Maquette du Site
Annexe 2 : Création d'un nouvel Article (publication)
Annexe 2bis : Insertion de Photos dans un Article
Annexe 3 : Visualisation des Publications
Annexe 4 : Modification d'une publication et/ou d'une photo de la publication
Annexe 5 : Suppression d'un article avec les photos qui y sont rattachées
Annexe 6 : Documentation du déploiement du site "Michaël Moniteur Cycliste"

# DOSSIER PROFESSIONNEL(DP)

## ANNEXES

Annexe 1 : Maquettes du site :

### Maquette version mobile 360px de largeur



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

# DOSSIER PROFESSIONNEL(DP)

## Maquette version ordinateur 1440 px de largeur



Annexe 2 : Création d'un nouvel Article (publication) :

*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,
        ]);
    }
}
```

# DOSSIER PROFESSIONNEL(DP)

## 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',
                        'mimeType' => [
                            'image/jpeg',
                            'image/png',
                            'image/webp'
                        ],
                        'mimeTypeMessage' => 'Veuillez télécharger une image valide (JPEG, PNG, WEBP et maxi 3Mo).',
                    ])
                ],
            ]),
        ],
    },
}
```

## 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 %}
```

# DOSSIER PROFESSIONNEL(DP)

## Annexe 2bis : Insertion de Photos dans un Article

```
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);
    }
});
```

*Exemple :*

```
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);
});
```



# DOSSIER PROFESSIONNEL(DP)

## Annexe 3 : Visualisation des Publications

Toutes les publications :

Espace Admin

Il y a 3 articles en ligne.

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

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

Statut : en ligne

Mon article test

Voir/modifier Supprimer

```
#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,
    ]);
}
```

Une publication précise :

Espace Admin

Publication :

Créé le : 14/10/2024 13:57

Venez me retrouver ce samedi 19/10 à Armentières ! Identifiant : 5  
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,
    ]);
}
```

## Annexe 4 : Modification d'une publication et/ou d'une photo de la publication

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,
]);
}
```

# DOSSIER PROFESSIONNEL(DP)

```
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);
    }
}
```

## Annexe 5 : Suppression d'un article avec les photos qui y sont rattachées

```
#[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');
}
```

# DOSSIER PROFESSIONNEL(DP)

## Annexe 6 : Documentation du Déploiement

### 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

# DOSSIER PROFESSIONNEL(DP)

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.