

# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance* ▶ GRILLON

*Nom d'usage* ▶

*Prénom* ▶ MICHEL

*Adresse* ▶ 22 rue du val d'aubance, 49120 Chemillé en Anjou

## Titre professionnel visé

*Développeur Web et Web Mobile*

### MODALITE D'ACCES :

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

# DOSSIER PROFESSIONNEL (DP)

## 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

#### Intitulé de l'activité-type n° 1 :

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

- ▶ Maquetter des interfaces utilisateur web ou web mobile ..... p. 6
- ▶ Réaliser des interfaces utilisateur statiques web ou web mobile ..... p. 9
- ▶ Développer la partie dynamique des interfaces utilisateur web ou web mobile ..... p. 12

#### Intitulé de l'activité-type n° 2 :

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

- ▶ Mettre en place une base de données relationnelle ..... p. 15 et 22
- ▶ Développer des composants d'accès aux données SQL..... p. 15 et 22
- ▶ Développer des composants métier côté serveur ..... p. 15 et 22
- ▶ Documenter le déploiement d'une application dynamique web ou web mobile ..... p. 22

Titres, diplômes, CQP, attestations de formation *(facultatif)* p. 24

Déclaration sur l'honneur p. 25

Documents illustrant la pratique professionnelle *(facultatif)* p. 26

Annexes *(Si le RC le prévoit)* p. 27

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

## Activité-type 1

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

Exemple n°1 ► Maquetter des interfaces utilisateur web ou web mobile

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

M. Deschamps souhaite créer un site internet afin de promouvoir son activité : la location de cabanes insolites en Bretagne. Il s'agit d'un site vitrine dont le but est de présenter ses différentes cabanes, les services, les activités touristiques et la région.

Pour répondre à la demande du client, j'ai réalisé l'arborescence du site, les zonings de 2 pages dont la page d'accueil du site et une page de mon choix en version desktop et mobile. Je propose enfin le wireframe de la page d'accueil en version desktop et mobile, en utilisant des pictogrammes libres de droits ainsi que du faux-texte.

J'ai commencé par le wireframe de la page mobile puis continué sur le format desktop.

En 1er lieu, j'ai posé les informations suivantes sur une feuille de papier :

- Pour qui (personas)
  - Les informations données par le client, qui ont abouties à la création des différentes rubriques (1ere ébauche)
- Je suis ensuite allé voir ce qui se faisait (recherche google sur cabane en bretagne).
- Je suis revenu à ma feuille où j'ai noté :
- ce que je trouvais pertinent (disposition, informations, iconographie, arborescence, etc.) et le moins bien.
  - J'ai croisé avec les cibles potentielles (personas).
  - Il en sort : principalement des actifs, en fin de journée et surtout le week-end, moyenne âge situé entre 24 et 44 ans, connexion à la maison principalement, desktop en majorité mais de peu par rapport au mobile.
  - On peut en déduire que les personas ont un minima de connaissance web mais il ne faut pas négliger les nouveaux utilisateurs.
  - De ce fait, le site doit être concis, sans trop d'informations mais compréhensible par un néophyte (sous-entendu, avoir des icônes claires et une disposition qui ne noie pas l'utilisateur sous diverses informations dont certaines non pertinentes).
  - Et les informations importantes (photos, description, réservation, paiement, accès, FAQ et CGV) doivent se trouver facilement.

Je suis ensuite allé chercher les résolutions d'écran minimale et moyenne (mobile 360 x 800 et desktop : 1920 x 1080) afin de préparer le zoning et wireframe.

J'ai fait mes premiers croquis sur une feuille pour l'arborescence, en prenant en compte les informations notées au préalable.

### 2. Précisez les moyens utilisés :

# DOSSIER PROFESSIONNEL (DP)

J'ai mis au propre l'arborescence via Visio de Microsoft (fichier cabanesInsolitesDeschamps.vsdx).

J'ai ensuite pris Illustrator d'Adobe, ou j'ai créé les fichiers de zonage pour commencer, en prenant en compte les résolutions d'écran indiqué ci-dessus : zonageHomeMobile.ai, zonageCabane1Mobile.ai, zonageDesktopHome.ai, zonageCabane1Desktop.ai.

La disposition des rubriques a été faite suivant les informations du client et la synthèse énoncée plus haut, à savoir : pertinence et classifications des informations, accessibilité, ce qui se fait en général sur d'autres sites plus anciens et populaires en n'oubliant pas que le client doit trouver rapidement l'information sans trop se poser de questions

Pour finir, j'ai pris les fichiers de zonage pour faire le wireframe en essayant d'être le plus clair sur les informations indiquées.

## 3. Avec qui avez-vous travaillé ?

Seul, en distanciel.

Ce travail a ensuite été corrigé par Nolwenn Gouzien, formatrice au CEFIL.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶

CEFIL.

Chantier, atelier, service ▶

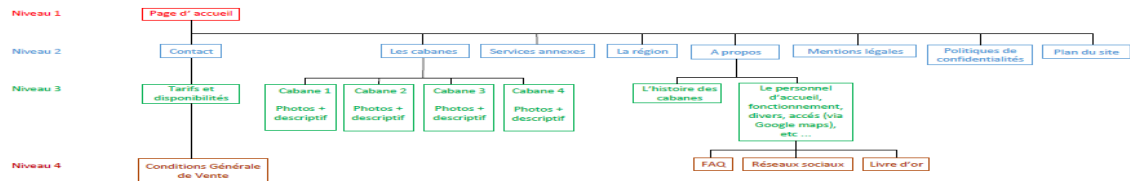
Période d'exercice ▶ Du 12/07/2023 au 16/07/2023

# DOSSIER PROFESSIONNEL (DP)

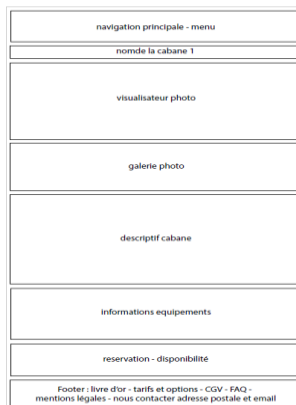
## 5. Informations complémentaires (facultatif)

### Arborescence du site :

#### Arborescence du site : Les Cabanes Insolites de M. Deschamps



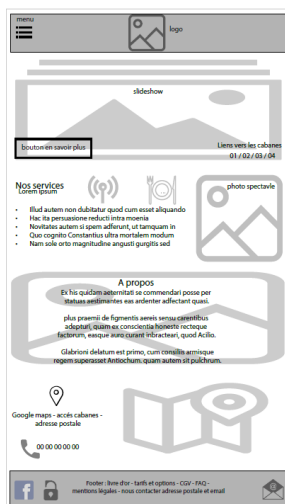
### Zonage mobile



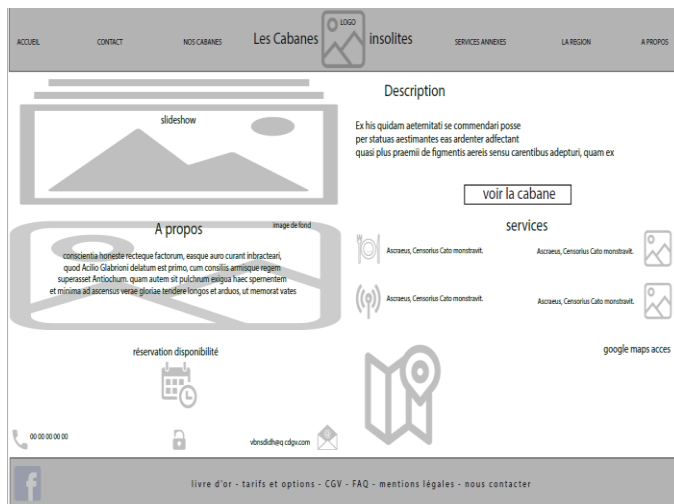
### Zonage desktop :



### Wireframe mobile :



### Wireframe desktop :



# DOSSIER PROFESSIONNEL (DP)

## Activité-type 1

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

**Exemple n°2** ► Réaliser des interfaces utilisateur statiques web ou web mobile : CEFii Golf

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

Pour le projet du site web du CEFii Golf, j'ai conçu la page d'accueil en suivant les directives fournies. J'ai d'abord créé la structure HTML en utilisant des balises sémantiques et structurantes comme <header>, <nav>, <main>, <section>, <article> et <footer>. Cela permet de donner du sens et de la structure à la page.

Ensuite, j'ai utilisé diverses techniques CSS pour mettre en forme les différents éléments. J'ai notamment employé : Flexbox et Grid Layout pour organiser la mise en page et le positionnement des éléments, des tableaux pour présenter les informations de manière tabulaire, un formulaire de contact avec les éléments appropriés (<form>, <input>, <textarea>, etc.) et l'insertion d'une vidéo YouTube et d'une carte Google Maps

J'ai aussi ajouté des animations et des transitions, par exemple sur le titre de la page, pour rendre certains éléments plus dynamiques.

Bien que l'approche "mobile-first" soit généralement recommandée, la consigne de l'exercice était de commencer par développer la version desktop de la page, avant de l'adapter aux versions tablette et mobile. Cela m'a permis de mettre en pratique mes connaissances en conception responsive.

### 2. Précisez les moyens utilisés :

Utilisation du logiciel Visual studio code et de Filezilla pour le transfert ftp vers le serveur du CEFII.

Des recherches Google sur différents forums lorsque bugs rencontrés...

Utilisés : HTML, CSS.

### 3. Avec qui avez-vous travaillé ?

Seul, en distanciel et Olivier Pesce, formateur du CEFII, pour les corrections et les aides.



# DOSSIER PROFESSIONNEL (DP)

## 4. Contexte

Nom de l'entreprise, organisme ou association ► Centre de formation: CEFii







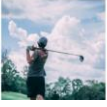
Chantier, atelier, service ► Atelier et cours de soutien.

Période d'exercice ► Du : 07/2023 au : 08/2023










## 5. Informations complémentaires (facultatif)

Ce projet avait pour but de me faire appliquer les notions vues en cours, comme l'utilisation sémantique du HTML, l'application de différents sélecteurs CSS, la mise en forme de polices, de textes, de listes et de tableaux, le positionnement des éléments, l'utilisation de transitions/transformations et d'animations. Je pense avoir réalisé un travail satisfaisant, en respectant les consignes de mise en page et de charte graphique fournies dans l'énoncé.

Ci-dessous, des copies d'écrans.

La page d'accueil copie d'écran version desktop)	Aperçu version mobile :
	
<div data-bbox="164 1671 272 1809">  </div> <div data-bbox="292 1671 422 1809"> <h3>À propos de nous</h3> <p>Codifié en Écosse en 1754 par le Royal &amp; Ancient Golf Club de Saint-André, le golf est un sport de précision se jouant en plein air, qui consiste à envoyer une balle dans un trou à l'aide de clubs. Le but du jeu consiste à effectuer, sur un parcours défini, le moins de coups possible. Précision, endurance, technique, concentration sont des qualités primordiales pour cette activité.</p> </div> <div data-bbox="164 1832 272 1971">  </div> <div data-bbox="292 1832 422 1971"> <h3>En savoir plus</h3> <ul style="list-style-type: none"> <li>↳ Notre histoire</li> <li>↳ Nos clubs</li> <li>↳ Votre parcours</li> <li>↳ Vos souvenirs</li> </ul> </div> <div data-bbox="459 1671 592 1832">  <h3>À la une</h3> <p>04/08</p> <p>Comment augmenter la puissance de votre swing ?</p> <p>Pour gagner de la vitesse il faut disposer d'énergie et l'utiliser au mieux. À la moindre accumulation de l'énergie potentielle de percuter. Pendant la descente nous transformons cette énergie en énergie cinétique avec un minimum de pertes pour accélérer jusqu'au finish.</p> <p>Lire la suite</p> </div>	<div data-bbox="879 1671 994 1832">  <h3>Information</h3> <p>Les parcours sont momentanément fermés</p> </div> <div data-bbox="839 1832 1034 2027"> <h3>À propos de nous</h3> <p>Codifié en Écosse en 1754 par le Royal &amp; Ancient Golf Club de Saint-André, le golf est un sport de précision se jouant en plein air, qui consiste à envoyer une balle dans un trou à l'aide de clubs. Le but du jeu consiste à effectuer, sur un parcours défini, le moins de coups possible. Précision, endurance, technique, concentration sont des qualités primordiales pour cette activité.</p>  </div>

# DOSSIER PROFESSIONNEL (DP)

<div><p>Découvrez notre dernière vidéo !</p><div><p>Rejoignez à la newsletter</p></div></div>																			
<div><div><p>Accueil Partenaires Le domaine Les membres Contact</p><p>18 rue du Golf 40131 GOLF-LANDE sur MER</p><p>Suivez nous ! Instagram Facebook</p><p>© Copyright - Production membre - 40131 Golf-lande - CFFG - Michel Lefebvre - Olivier RIZO - 2023</p></div></div>	<div><p>Rejoignez à la newsletter</p><div><p>18 rue du Golf 40131 GOLF-LANDE sur MER</p><p>Suivez nous ! Instagram Facebook</p><p>© Copyright - Production membre - 40131 Golf-lande - CFFG - Michel Lefebvre - Olivier RIZO - 2023</p></div></div>																		
La page tarifs et contact :																			
<div><p><b>Tarifs</b></p><p>Le CFFG Golf vous propose également des stages, avec ou sans hébergement, pendant les vacances scolaires. Découvrez au perfectionnement, les stages sont accessibles à différents niveaux.</p><table><tr><th></th><th>Tarifs publics</th><th>Tarifs membres</th></tr><tr><td>9 trous</td><td>30 €</td><td>20 €</td></tr><tr><td>18 trous*</td><td>45 €</td><td>20 €</td></tr><tr><td>2 jours de golf à volonté</td><td>110 €</td><td>90 €</td></tr><tr><td>3 jours de golf à volonté</td><td>150 €</td><td>120 €</td></tr><tr><td>7 jours de golf à volonté</td><td>250 €</td><td>200 €</td></tr></table><p>*Chereté matériel et sous-développement offerts avec votre green fee 18 trous !</p></div>		Tarifs publics	Tarifs membres	9 trous	30 €	20 €	18 trous*	45 €	20 €	2 jours de golf à volonté	110 €	90 €	3 jours de golf à volonté	150 €	120 €	7 jours de golf à volonté	250 €	200 €	
	Tarifs publics	Tarifs membres																	
9 trous	30 €	20 €																	
18 trous*	45 €	20 €																	
2 jours de golf à volonté	110 €	90 €																	
3 jours de golf à volonté	150 €	120 €																	
7 jours de golf à volonté	250 €	200 €																	
<div><p><b>Contact</b></p><p>Devenez membre dès aujourd'hui ! Remplissez le formulaire et nous vous recontacterons !</p><div><p>Votre nom : <input type="text"/></p><p>Votre adresse mail : <input type="text"/></p><p>Votre téléphone : <input type="text"/></p><p>Subject : <input type="text"/></p><p>Cherchez un club : <input type="text"/></p><p>Message : <input type="text"/></p><p><input type="button" value="Envoyer"/></p></div><div></div></div>	<div><p><b>Contact</b></p><p>Devenez membre dès aujourd'hui ! Remplissez le formulaire et nous vous recontacterons !</p><div><p>Votre nom : <input type="text"/></p><p>Votre adresse mail : <input type="text"/></p><p>Votre téléphone : <input type="text"/></p><p>Subject : <input type="text"/></p><p>Cherchez un club : <input type="text"/></p><p>Message : <input type="text"/></p><p><input type="button" value="Envoyer"/></p></div><div></div></div>																		
<div><p><b>Foire aux questions</b></p><div><p>Puis-je accéder au CFFG Golf sans être membre ? <input type="text"/></p><p>Je souhaite essayer le golf, comment faire ? <input type="text"/></p><p>Comment effectuer une réservation de groupe ? <input type="text"/></p><p>Mes enfants peuvent-ils jouer au golf ? <input type="text"/></p></div></div>																			
<div><div></div></div>																			

## Activité-type 1

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

**Exemple n° 3** ► Développer la partie dynamique des interfaces utilisateur web ou web mobile

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

J'ai créé un petit panier de produits avec une page HTML/CSS simple, nommée "panier.php". Cette page comporte deux principales parties via deux DIV :

1. La section "Produits" :
  - J'ai affiché une liste de 4 produits (disque dur, TV, clé USB) avec leur nom, prix et un bouton "Ajouter au panier".
  - Lorsqu'un produit est ajouté, la quantité s'affiche à côté du bouton, grâce au code suivant :

```
<?php echo isset($_SESSION['nombreArticleProduitarticle1']) ? 'Quantité : ' .  
$_SESSION['nombreArticleProduitarticle1'] : ''; ?>
```

La section "Panier" :

- J'ai affiché un récapitulatif du panier, avec le nombre total d'articles et la quantité de chaque article.
- J'ai ajouté un bouton "Vider mon panier" qui permet de réinitialiser le contenu du panier, avec le lien :

```
<a class="btn btn-warning" title="viderPanier" href="modifierPanier.php?vider=1" role="button">Vider  
mon panier</a>
```

J'ai utilisé les sessions PHP pour "mémoriser" le contenu du panier. Dans un premier temps, je me suis limité à stocker en session la quantité totale de produit, en utilisant la variable

```
$_SESSION['nombreArticlesPanier'].
```

Les liens "Ajouter au panier" de chaque produit pointent vers un fichier "modifierPanier.php". Dans ce fichier, je démarre une session et je gère l'incrémentation du nombre total d'articles dans le panier et du nombre d'articles spécifiques à chaque produit :

```
// Vérifie si la session existe pour le nombre total d'articles
```

```
if (!isset($_SESSION['nombreArticlesPanier'])) {
```

```
    $_SESSION['nombreArticlesPanier'] = 1;
```

# DOSSIER PROFESSIONNEL (DP)

```
} else {  
    $_SESSION['nombreArticlesPanier']++;  
}  
  
// Incrémente le nombre d'articles pour le produit spécifique  
  
if (isset($_SESSION['nombreArticleProduit' . $idProduit])) {  
    $_SESSION['nombreArticleProduit' . $idProduit]++;  
} else {  
    $_SESSION['nombreArticleProduit' . $idProduit] = 1;  
}  
}
```

Ensuite, j'ai ajouté un lien "Vider mon panier" dans la section "Panier", qui pointe vers "modifierPanier.php" avec le paramètre `?vider=1`. Dans ce fichier, j'ai réinitialisé les variables de session à 0 pour chaque article et pour le nombre total d'articles.

Enfin, j'ai amélioré l'affichage du panier pour qu'il montre les libellés et les quantités de chaque produit, en utilisant des variables de session spécifiques à chaque produit, comme :

```
<?php echo isset($_SESSION['nombreArticleProduitarticle1']) ? $_SESSION['nombreArticleProduitarticle1']  
: '0'; ?>
```

Ce projet d'application e-commerce de base m'a permis de mettre en pratique mes connaissances en HTML, CSS et PHP, notamment l'utilisation des sessions pour mémoriser le contenu du panier.

## 2. Précisez les moyens utilisés :

Visual studio code.

HTML, CSS, PHP et Google (forum, etc...).

# DOSSIER PROFESSIONNEL (DP)

## 3. Avec qui avez-vous travaillé ?

Olivier Pesce, formateur CEFII. En distanciel.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► Centre de formation : CEFII

Chantier, atelier, service ► Atelier et cours de soutien.

Période d'exercice ► Du : 10/2023 au : 10/2023

## 5. Informations complémentaires (facultatif)

Disque Dur

disque dur 1 To | 175€ |

Ajouter au panier

TV

smart tv 24p | 250€ |

Ajouter au panier

Clé usb

clé usb 128Go 500 Go | 100€ |

Ajouter au panier

## Mon PANIER

Nombre total d'articles dans le panier :

0

Total-détail par articles

Quantité pour chaque article dans le panier :

Article 1 :

0

Article 2 :

0

Article 3 :

0

Article 4 :

0

Vider mon panier

Récapitulatif de mon PANIER

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 2

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

**Exemple n° 1** ► Mettre en place une base de données relationnelle,  
Développer des composants d'accès aux données SQL,  
Développer des composants métier côté serveur

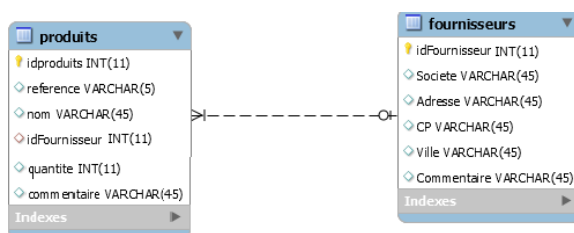
### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

J'ai pour mission de réaliser une application simplifiée de gestion de stock. Cette application doit permettre de gérer les produits et les fournisseurs, avec les fonctionnalités suivantes :

- Un menu principal avec les entrées "Produits" et "Fournisseurs", chacune menant à un sous-menu "Lister" et "Ajouter".
- Un formulaire pour ajouter un nouveau produit, avec les champs Référence, Nom, Quantité, Fournisseur (sous forme de liste déroulante) et Commentaire.
- Un formulaire pour ajouter un nouveau fournisseur, avec les champs Société, Adresse, Code postal, Ville et Commentaire.
- Sur les pages "Produits" et "Fournisseurs", afficher la liste des éléments existants dans un tableau avec des options pour les éditer et les supprimer.
- Utiliser PDO pour communiquer avec la base de données MySQL, en utilisant des requêtes préparées.
- Organiser le code de manière modulaire, avec un fichier index.php central qui gère les différentes pages en fonction d'un paramètre "page" passé en GET.
- Inclure les éléments communs (header, nav, footer) dans l'index.php pour une meilleure organisation et maintenance du code.
- Respecter les bonnes pratiques de codage et bien commenter le code.

Mode opératoire et outils utilisés : pour réaliser cet exercice, je procède de la manière suivante :

1. Je définis le Modèle Logique de Données (MLD) en créant deux tables : "produits" et "fournisseurs", avec les champs spécifiés dans l'énoncé. Via le logiciel Jmerise, le MLD :



# DOSSIER PROFESSIONNEL (DP)

2. Je crée la base de données et les tables correspondantes, exemple avec la table Produits : **Annexe 1**

Le principe sera le même pour la table Fournisseurs.

3. Je crée le fichier de connexion à la base de données (connect.php) avec PDO : **Annexe 2**

Je développe l'application en suivant une architecture modulaire : création du fichier index.php qui servira de page d'entrée et de contrôleur central. Dans celui-ci, je récupère le paramètre défini dans toutes mes URL (menu et formulaire). Extrait du code (PHP) : **Annexe 3**

Création des fichiers séparés pour les vues (header, nav, footer, liste des produits, formulaire d'ajout de produit, etc.) : modifications ou mises à jour plus simple à l'avenir.

Inclusion dans index.php par exemple : `include 'pages/header.html';` ou encore : `include 'pages/nav.html';` et aussi : `include 'pages/footer.html';`

Création des fichiers de classes ou de fonctions pour gérer la logique métier (CRUD sur les produits et fournisseurs). Pour le CRUD (Create, Read, Update et Delete, pour création des éléments dans la base, la lecture de ceux-ci, la mise à jour d'un enregistrement ou la suppression). Pour commencer, on exécute la requête SQL : --  
- Ajout d'un produit par exemple, fichier ajouter\_produit.php : **Annexe 4**

Autre exemple, la suppression d'un produit (supprimer\_produit.php) : **Annexe 5**

Dans ce formulaire il y aura : `<form action='index.php?page=modifProduct' method='POST'>`

`<!-- Champ caché pour l'identifiant du produit -->`

`<input type='hidden' name='id' value='<?php echo $produit['idProduits']; ?>'>`

[...] Champs pour les différentes informations du produit, les actions via les boutons, pour remplir les champs etc.

Le principe sera le même pour la lecture, la modification, tant sur Produits que Fournisseurs. J'utilise PDO pour communiquer avec la BDD afin de créer une classe ou des fonctions pour gérer les connexions à la base de données et d'utiliser des requêtes préparées pour sécuriser les interactions avec BDD. Exemple : **Annexe 6**

Je mets en place le système de navigation avec le paramètre "page" en GET.

Exemple, de la page `editier_produit.php`, pour revenir à la page d'accueil :

`<!-- Bouton pour retourner à l'accueil --> <a href='index.php?page=home' class='btn btn-warning mt-3'>Retour à l'accueil</a>`

J'intègre Bootstrap pour la mise en forme de l'interface utilisateur. En premier lieu, dans `header.html` :

`<link href='https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css' rel='stylesheet'`

`integrity='sha384-***/*' crossorigin='anonymous'>`

## DOSSIER PROFESSIONNEL (DP)

Puis, exemple avec le bouton "annuler l'édition" dans `editer_produit.php` :

```
<!-- Bouton pour annuler l'édition et revenir à la liste des produits -->

<div class='my-3'>

    <a href='index?page=productsList' class='btn btn-danger'>Annuler l'édition et retourner aux produits</a>

</div>
```

Et pour finir, j'ai testé régulièrement l'application et corrigé les bugs (merci Google) et commenté le code de manière claire et concise.

Explications et analyse des choix de conception :

**Utilisation de GET** : J'ai utilisé la méthode **GET** pour récupérer l'action à effectuer sur les pages `fournisseurs.php` et `produits.php`, car elle permet de manipuler l'action directement depuis l'URL, ce qui facilite le débogage et la navigation.

**Utilisation de PDO** : J'ai utilisé l'objet **PHP Data Objects (PDO)** pour interagir avec la base de données, car PDO offre une interface cohérente pour accéder à plusieurs types de bases de données et il supporte les requêtes préparées, ce qui aide à prévenir les injections SQL.

**Requêtes préparées** : Dans les fichiers `ajouter_fournisseurs` et `ajouter_produit.php`, j'ai utilisé une **requête préparée** pour insérer un nouveau fournisseur dans la base de données. C'est une pratique de sécurité recommandée car elle aide à prévenir les injections SQL en séparant les instructions SQL des données.

**Inclusion de fichiers** : J'ai inclus les fichiers `header.html`, `header2.html`, `footer.html`, `footer2.html` et `include.php` dans le code, car cela permet de réutiliser le code et de garder le code organisé.

**Gestion des erreurs** : J'ai utilisé le bloc `try-catch` pour gérer les erreurs potentielles qui peuvent survenir lors de l'exécution de la requête SQL, car cela permet d'éviter que l'application ne se bloque en cas d'erreur.

**Structure du code** : J'ai utilisé une instruction `switch` pour gérer différentes actions, ce qui rend le code plus lisible et facile à maintenir.

Ces choix ont été faits d'abord suite aux recommandations de l'énoncé. Concernant quelques fonctions, si je prends par exemple cet extrait de mon code : **Annexe 7**

Dans ce code, les principales fonctions sont les suivantes :

**include** : Cette fonction est utilisée pour inclure et évaluer le fichier spécifié. Ici, elle est utilisée pour inclure les fichiers `'header2.html'`, `'footer2.html'` et `'connect.php'`.

**isset** : Cette fonction est utilisée pour vérifier si une variable est définie et n'est pas NULL. Ici, elle est utilisée pour vérifier si le paramètre GET `'action'` est défini.

**file\_exists** : Cette fonction est utilisée pour vérifier si un fichier ou un répertoire existe. Ici, elle est utilisée pour vérifier si le fichier `'connect.php'` existe.

**switch** : Cette structure de contrôle est utilisée pour effectuer différentes actions en fonction de différentes conditions. Ici, elle est utilisée pour effectuer différentes actions en fonction de la valeur de la variable `$action`.

**try-catch** : Ces blocs sont utilisés pour capturer et gérer les exceptions. Ici, ils sont utilisés pour gérer les exceptions qui peuvent se produire lors de l'exécution de la requête SQL.

**\$connexion->query** : Cette méthode est utilisée pour exécuter une requête SQL et renvoie un ensemble de résultats en tant qu'objet `PDOStatement`.

**fetchAll** : Cette méthode est utilisée pour retourner un tableau contenant toutes les lignes du jeu de résultats. Le tableau représente chaque ligne comme soit un tableau de valeurs de colonnes, soit un objet avec des propriétés correspondant à chaque nom de colonne. Ici, elle est utilisée pour obtenir toutes les lignes de la requête SQL.



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

Outils : PHP 8+ avec PDO et MySQL pour la gestion de la base de données, Bootstrap pour la mise en forme, Visual Studio Code et le logiciel JMerise pour le MCD, MLD et le MPD

## 3. Avec qui avez-vous travaillé ?

Seul, en distanciel, avec l'aide d'Olivier Pesce (formateur au CEFII).

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Centre de formation : CEFII*

Chantier, atelier, service ► *Atelier et cours de soutien.*

Période d'exercice ► Du : *11/2023* au : *12/2023*

## 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** ► Mettre en place une base de données relationnelle

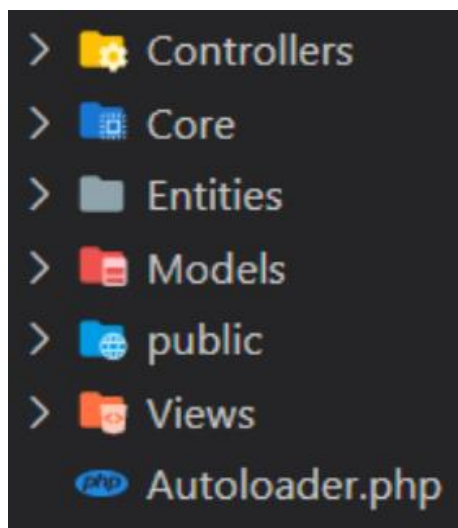
Développer des composants d'accès aux données SQL

Développer des composants métier côté serveur

Documenter le déploiement d'une application dynamique web ou web mobile

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

Je dois créer, pour cet exercice qui fait suite au cours, une application MVC : un site portfolio.  
Voici l'arborescence demandée :



Le rôle de chaque dossier et fichier :

- **Controllers**: Il contiendra tous les contrôleurs en charge de gérer leur entité.
- **Core**: C'est le coeur de l'application (connexion à la base de données, router).
- **Entities**: Il contient les classes représentant les tables de la base de données.
- **Models**: Il contiendra tous les modèles (requêtes SQL) de chaque entité.
- **public**: il contient ce qui est accessible par le navigateur.
- **Views**: Il contiendra toutes les vues (l'affichage des pages) avec un dossier par contrôleur.
- **Autoloader.php**: Il permettra de charger automatiquement nos fichiers de classes.

Je crée la BDD : table creation, colonnes : id\_creation, title, description, created\_at et picture.

Ensuite le routeur : il va récupérer l'URL demandée par l'utilisateur. J'ai donc défini une structure type des URLs. Ces URLs vont contenir comme paramètres, le nom du contrôleur et le nom de la méthode à exécuter.

Elles seront formatées ainsi : `http://localhost/public/index.php?controller=home&action=index`

## DOSSIER PROFESSIONNEL (DP)

Le routeur va donc être une classe que l'on va créer à l'intérieur du dossier "Core". Il est le point d'entrée de l'application, on va donc l'instancier dans un fichier appelé "index.php" dans le dossier "public". (Annexe 1 et 2)

Ensuite, je crée un contrôleur mère pour profiter de l'abstraction et de l'héritage de la POO, je mets ainsi les méthodes communes à tous les contrôleurs. (Annexe 3)

Puis je le contrôleur pour la page d'accueil puis un autre pour les pages des créations de l'utilisateur. (Annexe 4 et 5)

Je passe au modèle : représenté dans la structure MVC en deux parties : un représentant une entité de la base de données et l'autre exécutant les requêtes en direction de la base de données.

Dans certains cas, il retournera pour le contrôleur concerné, les informations récupérées en base de données. Il sera donc représenté par deux classes distinctes.

Pour qu'il puisse réaliser ces actions, il devra en amont être connecté à la base de données. Pour cela, je crée une classe appelée "DbConnect.php" (Annexe 6), stockée dans le dossier "Core", qui va me servir de connexion à la base de données de l'application.

Elle sera une classe mère pour le modèle. Les requêtes seront mises en place dans une classe appelée "CreationModel.php" stockée dans le dossier "Models". (Annexe 7)

Dans cet exercice, j'ai créé une seule entité qui représentera la table "creation" et donc création d'une nouvelle classe appelée "Creation.php" et elle sera stockée dans le dossier "Entities". (Annexe 8).

La vue va récupérer les informations envoyées par le contrôleur pour les afficher au milieu d'une structure HTML, mais PHP n'est pas oublié, par exemple pour boucler sur la liste des créations du portfolio. (Annexe 9)

Une vue représente une page, j'ai créé un fichier par vue et chacun des fichiers sera stocké dans un dossier.

Pour la page d'accueil de l'application, je crée dans le dossier "Views" un dossier appelé "home" reprenant par convention le préfixe du contrôleur "HomeController", et j'y crée à l'intérieur un fichier nommé "index.php" reprenant le nom de la méthode exécutée.

Ensuite pour la page listant les créations, je fais la même chose, c'est-à-dire créer un dossier appelé "creation", dans lequel un fichier "index.php" sera créé. (Annexe 9)

Pour exploiter cette arborescence, mets en place une méthode appelée "render()" dans le contrôleur principal "Controller.php". Elle va permettre de sélectionner la vue que l'on souhaite pour y envoyer des informations. Elle sera exécutée dans chacun des contrôleurs. (Annexe 10)

Ensuite, je crée dans le dossier "Views" un fichier appelé "base.php". Ce fichier sera le template ou gabarit de mes pages. Ensuite, pour afficher mon template, je modifie la méthode "render()". (Annexe 10)

Grace à cela, j'évite les doublons : les deux vues créées ou j'ai répété mon code en ajoutant dans les deux pages, le header, le menu et le footer...

Je vais avoir besoin d'utiliser des formulaires pour ajouter, modifier, voir, supprimer une création.

## DOSSIER PROFESSIONNEL (DP)

Je mets en place un générateur de formulaire (classe "form", permettant l'ajout des champs, générer le corps du formulaire et vérifier la saisie dans les champs) afin d'éviter la duplication de code, de permettre une mise à jour plus simple et rapide. Je peux donc faire autant de formulaire que voulu... avec le même code. (Annexe 11). Le formulaire créé sera donc représenté en tant qu'objet.

Le formulaire sera construit dans un contrôleur, pour être ensuite envoyé à une vue. De ce fait, je crée la vue "add.php" dans le dossier création pour afficher le formulaire. Je teste ensuite les champs, s'ils sont vides (si ce n'est pas le cas, j'ajoute des valeurs dans chaque propriété de la classe "Creation" = hydratation).

Pour le traitement du formulaire d'ajout de création, je mets en place le fichier "CreationController.php" pour compléter la méthode "add".

Dans "CreationController.php", mise en place de l'exécution de la méthode "findAll()" déclarée dans la classe modèle "CreationController" (permet d'aller récupérer les informations des créations dans la BDD). (Annexe 12)

Dans index.php dans le dossier "creation", pour correctement afficher la liste des créations et donc gérer l'affichage des images de chaque création, je crée la feuille de style "style.css" (dans le dossier public à la racine de l'application). Le lien vers ce fichier (style.css) est donc ajouté à base.php. (Annexe 13)

J'en profite pour ajouter une icône et un lien sous forme d'une icône dans le tableau afin de rediriger l'utilisateur vers la page affichant la création choisie. L'utilisation d'icônes "Font Awesome" et l'ajout du CDN dans le fichier "base.php" sont fait. Idem pour modifier (update) et supprimer... (Annexe 16)

Pour respecter la structure MVC, je crée une nouvelle méthode dans le contrôleur "CreationController.php" appelée "showCreation()" (Annexe 14). Pour afficher la création sélectionnée, je crée une nouvelle vue dans le dossier "creation", appelée "showCreation.php" (Annexe 15). Le même principe sera appliqué pour la modification (updateCreation.php), la suppression (deleteCreation.php) ...

Et j'ajoute un message de confirmation pour chacune des actions afin de valider ou pas l'action suite au clic sur l'icône.

Je passe ensuite sur la partie sécurité de l'application, après avoir effectué (au fur et à mesure) les tests de fonctionnement de l'application, par exemple en affichant les erreurs (Annexe 17) (ou en créant un fichier log).

Correction xss (Cross-Site Scripting, vulnérabilité qui permet à un attaquant d'injecter du code malveillant (généralement du JavaScript) dans une page web affichée par un utilisateur.).

Utilisation de :

- la fonction htmlspecialchars() pour échapper les caractères spéciaux lors de l'affichage de données utilisateur dans une page web. (Annexe 18)
- la fonction filter\_input() pour valider les données utilisateur avant de les utiliser dans l'application.
- l'en-tête HTTP Content-Security-Policy pour restreindre les sources autorisées pour les scripts, les feuilles de style, les images, etc.

# DOSSIER PROFESSIONNEL (DP)

Correction CSRF (vulnérabilité qui permet à un attaquant de forcer un utilisateur à effectuer une action (comme la suppression d'une création) sans son consentement) : utilisation d'un jeton CSRF (token) pour vérifier que les requêtes proviennent bien de l'utilisateur et non d'un attaquant. (Annexe 19)

Pour déterminer quels fichiers doivent inclure une protection CSRF, j'ai considéré les points d'interaction où des formulaires sont soumis ou des actions sont déclenchées par des requêtes. Ce qui donne :

Contrôleurs : CreationController.php: Méthode add(), Méthode updateCreation(), Méthode deleteCreation()

Vues : creation/add.php, creation/updateCreation.php, creation/deleteCreation.php

Formulaires : Formulaire d'ajout de création, Formulaire de modification de création

Sans oublier de vérifier que les requêtes POST contiennent bien le jeton CSRF avant de les traiter.

Correction contre le Hijacking (détournement de session), utiliser des connexions HTTPS pour protéger les données échangées entre le client et le serveur, des identifiants de session sécurisés et difficiles à deviner et un en-tête HTTP Strict-Transport-Security pour forcer l'utilisation de connexions HTTPS :

CreationController.php, Controller.php, DbConnect.php, Autoloader.php et Form.php (Annexe 20)

Correction contre l'injection SQL via des requêtes préparées (avec "prepare()" et "execute()" :

CreationController (Suppression de l'échappement des données avec htmlspecialchars() lors de la liaison des paramètres, car PDO gère déjà l'échappement des données lors de l'utilisation de requêtes préparées).

CreationModel (Suppression de l'échappement des données avec htmlspecialchars() lors de la liaison des paramètres, car PDO gère déjà l'échappement des données lors de l'utilisation de requêtes préparées et ajout du type de données PDO::PARAM\_INT pour la liaison du paramètre :id\_creation dans les méthodes find() et delete().)

DbConnect : Dans la méthode executeQuery(), j'ai ajouté une vérification pour s'assurer que des paramètres ont été fournis avant de les lier à la requête préparée. Cela permet d'éviter les erreurs lors de l'exécution d'une requête sans paramètre. (Annexe 21)

Pour les images (faille d'upload) : CreationController (en particulier les fonctions add, update et delete) (Annexe 22)

include() : pas nécessaire, le serveur du CEFII qui héberge l'application a l'option "allow\_url\_include" sur "false" dans son php.ini...

## 2. Précisez les moyens utilisés :

Outils : PHP 8+ avec PDO et MySQL pour la gestion de la base de données, Bootstrap pour la mise en forme, Visual Studio Code.

## DOSSIER PROFESSIONNEL (DP)

### 3. Avec qui avez-vous travaillé ?

Seul, en distanciel, avec l'aide précieuse d'Olivier Pesce (formateur au CEFII).

### 4. Contexte

Nom de l'entreprise, organisme ou association ► Centre de formation : CEFII

Chantier, atelier, service ► Atelier et cours de soutien.

Période d'exercice ► Du : 03/2024 au : 04/2024

### 5. Informations complémentaires (facultatif)

## DOSSIER PROFESSIONNEL (DP)

### Titres, diplômes, CQP, attestations de formation

*(facultatif)*

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	
Cliquez ici.	Cliquez ici pour taper du texte.	

## DOSSIER PROFESSIONNEL (DP)

### Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] Michel Grillon .....,  
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 Saint-Lézin (Chemillé en Anjou)..... le 09 aout 2024  
pour faire valoir ce que de droit.

Signature :

A handwritten signature in black ink, appearing to read 'MICHEL GRILLON', with several horizontal strokes underneath.



## DOSSIER PROFESSIONNEL (DP)

### Documents illustrant la pratique professionnelle

*(facultatif)*

Intitulé
Cliquez ici pour taper du texte.

# DOSSIER PROFESSIONNEL (DP)

## ANNEXES

### Activité type 2, exemple 1 :

#### Annexe 1

```
CREATE TABLE Produits(  
  
    id_produit      Int Auto_increment NOT NULL ,  
  
    reference       Varchar (50) NOT NULL ,  
  
    nom             Varchar (100) NOT NULL ,  
  
    quantite        Int NOT NULL ,  
  
    commentaire_produit Varchar (100) NOT NULL ,  
  
    id_fournisseur  Int Auto_increment NOT NULL  
  
    ,CONSTRAINT Produits_PK PRIMARY KEY (id_produit)  
  
)ENGINE=InnoDB;
```

#### Annexe 2

```
define('HOST', "*****");  
  
define('DBNAME', "****");  
  
define('USER', "****");  
  
define('PASSWORD', "****");  
  
try {  
  
    $connexion = new PDO("mysql:host=" . HOST . ";dbname=" . DBNAME, USER, PASSWORD);  
  
    // Configuration pour afficher les erreurs PDO, ajout de PDOException dans le bloc catch pour capturer les exceptions  
    // spécifiques à PDO  
  
    // Configuration de l'attribut PDO::ATTR_ERRMODE pour afficher les erreurs PDO  
  
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

---

# DOSSIER PROFESSIONNEL (DP)

---

```
//echo "Succès !!! La connexion MySQL est ok !<br />";  
  
} catch (PDOException $e) {  
  
    echo 'Erreur : ' . $e->getMessage();  
  
}
```

## Annexe 3

```
if (isset($_GET['page'])) {  
  
    $page = $_GET['page'];  
  
    switch ($page) {  
  
        case 'productsList':  
  
            include 'pages/produits.php';  
  
            break;  
  
        case 'addProduct':  
  
            include 'pages/ajouter_produit.php';  
  
            break;  
  
        [et ainsi de suite pour toutes page ...]  
  
        [...]  
  
    }  
  
} else {  
  
    // Gérer la page par défaut ici si aucun paramètre n'est spécifié  
  
    //include 'pages/defaultPage.php';  
  
}
```

## Annexe 4

```
// Requête préparée pour l'insertion d'un nouveau produit  
  
$requeteInsertion = "INSERT INTO produits (Reference, Nom, Quantite, idFournisseur, Commentaire) VALUES  
(:Reference, :Nom, :Quantite, :idFournisseur, :Commentaire)";  
  
$statement = $connexion->prepare($requeteInsertion);
```

---

# DOSSIER PROFESSIONNEL (DP)

---

```
// Liaison des paramètres avec les valeurs

$stmtement->bindParam(':Reference', $Reference);

$stmtement->bindParam(':Nom', $Nom);

$stmtement->bindParam(':Quantite', $Quantite);

$stmtement->bindParam(':idFournisseur', $idFournisseur);

$stmtement->bindParam(':Commentaire', $Commentaire);

// Exécution de la requête préparée

$stmtement->execute();

echo "Nouvelle ligne insérée avec succès.";

} catch (PDOException $e) {

    echo "Erreur : ". $e->getMessage();

}

}
```

## Annexe 5

```
// Vérification de la présence de l'identifiant du produit dans l'URL

if (isset($_GET['id'])) {

    $produitId = $_GET['id'];

    try {

        // Requête préparée pour récupérer les informations du produit à éditer

        $requeteProduit = "SELECT * FROM produits WHERE idProduits = :produitId";

        $statement = $connexion->prepare($requeteProduit);

        $statement->bindParam(':produitId', $produitId);

        $statement->execute();

        $produit = $statement->fetch(PDO::FETCH_ASSOC);

        if ($produit) {

<!-- Affichage du formulaire d'édition du produit -->
```

# DOSSIER PROFESSIONNEL (DP)

## Annexe 6

```
// Vérification de la présence de l'identifiant du produit dans l'URL

if (isset($_GET['id'])) {

    $produitId = $_GET['id'];

    try {

        // Requête préparée pour récupérer les informations du produit à éditer

        $requeteProduit = "SELECT * FROM produits WHERE idProduits = :produitId";

        $statement = $connexion->prepare($requeteProduit);

        $statement->bindParam(':produitId', $produitId);

        $statement->execute();

        $produit = $statement->fetch(PDO::FETCH_ASSOC);
```

## Annexe 7

```
// Inclusion du header
include 'header2.html';
// Vérification du paramètre GET 'action'
$action = isset($_GET['action']) ? $_GET['action'] : 'liste';
// Inclusion du fichier contenant la connexion PDO à la base de données
$file = "connect.php";
if (file_exists($file)) {
    include $file;
    switch ($action) {
        case 'liste':
            default:
                // Code pour afficher la liste des fournisseurs
                try {
                    // Requête pour récupérer la liste des fournisseurs
                    $requete = "SELECT idFournisseur, Societe, Adresse, Code_postal, Ville, Commentaire FROM fournisseurs";
                    $resultat = $connexion->query($requete);
                    $rows = $resultat->fetchAll(PDO::FETCH_ASSOC);
                    // Code pour afficher les fournisseurs
                } catch (PDOException $e) {
                    echo "Erreur : ". $e->getMessage();
                }
                break;
    }
}
```

---

# DOSSIER PROFESSIONNEL (DP)

---

```
}  
// Inclusion du footer  
include 'footer2.html';
```

## Activité type 2, exemple 2 :

### Annexe 1 et 2 :

Router.php (extrait) :

```
namespace App\Core;  
  
class Router  
{  
    protected $connection;  
  
    public function __construct($connection)  
    {  
        $this->connection = $connection;  
    }  
  
    public function routes()  
    // Récupération du nom du contrôleur depuis $_GET  
  
        $controllerName = isset($_GET['controller']) ? ucfirst($_GET['controller']) : 'Home';  
  
        $controllerName = '\\App\\Controllers\\' . $controllerName . 'Controller';  
  
    // Récupération de l'action depuis $_GET  
  
        $action = isset($_GET['action']) ? $_GET['action'] : 'index';
```

Index.php (dossier public) :

```
// On importe les namespaces de l'autoloader et du router  
  
use App\Autoloader;
```

---

# DOSSIER PROFESSIONNEL (DP)

---

```
use App\Core\Router;
```

**Annexe 3** : Controller.php :

```
namespace App\Controllers;

// Inclure le fichier contenant la définition de la classe DbConnect

require_once __DIR__ . '/../Core/DbConnect.php';

abstract class Controller
```

**Annexe 4 et 5** : controleur de la page d'accueil (HomeController.php)

```
namespace App\Controllers;

class HomeController extends Controller
{
    // Méthode d'action par défaut

    public function index()
    {
        // Vérifier si une session est déjà démarrée

        if (session_status() === PHP_SESSION_NONE) {

        }

        // Rendu de la vue en incluant le token CSRF dans le formulaire

        $this->render('home/index');

    }
}
```

CreationController.php :

```
namespace App\Controllers;

use App\Core\Form;

use App\Entities\Creation;
```

---

# DOSSIER PROFESSIONNEL (DP)

---

```
use App\Models\CreationModel;

class CreationController extends Controller
{
    // Méthode pour afficher la liste des créations

    public function index()
    {
```

**Annexe 6** : DbConnect.php :

```
namespace App\Core;

use PDO;

use Exception;

class DbConnect
{
    // Propriétés protégées pour la connexion à la base de données et la requête

    protected $connection;

    protected $connexion; // Propriété pour stocker la connexion à la base de données admin

    protected $request;

    // Paramètres de connexion à la base de données

    const SERVER = 'sqlprive-pc2372-001.eu.cloud.db.ovh.net:35167'; // Adresse du serveur de base de données

    const USER = 'cefiudev1385'; // Nom d'utilisateur de la base de données

    const PASSWORD = 'wy56ZSw6'; // Mot de passe de la base de données

    const BASE = 'cefiudev1385'; // Nom de la base de données

    // Constructeur de la classe

    public function __construct()
```

**Annexe 7 :**

```
class CreationModel extends DbConnect
{
```



---

## DOSSIER PROFESSIONNEL (DP)

---

```
// Méthode pour récupérer toutes les créations

public function findAll()
{
    // Construction de la requête SELECT

    $this->request = "SELECT * FROM creation";

    // Exécution de la requête

    $result = $this->connection->query($this->request);

    // Récupération des résultats dans un tableau associatif

    $list = $result->fetchAll();

    // Retourne le tableau de résultats

    return $list;
}
```

### **Annexe 8:** Creation.php :

```
class Creation
{
    // Propriétés privées représentant les champs de la table "creation"

    private $id_creation;

    private $title;

    private $description;

    private $created_at;

    private $picture;

    // Getters et setters avec échappement des données pour le titre et la description

    /**
     * Getter pour récupérer la valeur du titre en échappant les caractères spéciaux
     */

    public function getTitle()
    {
        return htmlspecialchars($this->title);
    }
}
```

# DOSSIER PROFESSIONNEL (DP)

```
/**  
 * Setter pour définir la valeur du titre  
 * @return self  
 */
```

```
public function setTitle($title)  
{  
    $this->title = $title;  
    return $this;  
}
```

**Annexe 9** : index.php (Views/creation)

*//On boucle dans le tableau \$list qui contient la liste des créations*

```
foreach ($list as $value) {  
    echo "<tr>";  
    echo "<td>" . htmlspecialchars($value->id_creation, ENT_QUOTES, 'UTF-8') . "</td>";  
    echo "<td>" . htmlspecialchars($value->title, ENT_QUOTES, 'UTF-8') . "</td>";  
    echo "<td>" . htmlspecialchars($value->description, ENT_QUOTES, 'UTF-8') . "</td>";  
    echo "<td>" . htmlspecialchars($value->created_at, ENT_QUOTES, 'UTF-8') . "</td>";  
    echo "<td><img src='" . htmlspecialchars($value->picture, ENT_QUOTES, 'UTF-8') . "' class='picture'></td>";  
}
```

**Annexe 10** ! fonction render () (Controller.php) :

*// Méthode pour rendre une vue avec les données fournies*

```
public function render(string $path, array $data = [])  
{  
    // Démarrer ou initialiser la session  
    $this->startSession();  
    // Permet d'extraire les données récupérées sous forme de variables  
    extract($data);  
    // On crée le buffer de sortie  
    ob_start();
```

---

## DOSSIER PROFESSIONNEL (DP)

---

```
// Crée le chemin et inclut le fichier de la vue souhaitée

include dirname(__DIR__) . '/Views/' . $path . '.php';

// On vide le buffer dans la variable $content

$content = ob_get_clean();

// On échappe les données pour éviter les attaques XSS

//$content = htmlspecialchars($content, ENT_QUOTES, 'UTF-8');

// On fabrique le "template"

include dirname(__DIR__) . '/Views/Base.php';

}
```

**Annexe 11:** Form.php (Core) extraits :

```
namespace App\Core;

class Form
{
    // Attribut contenant le code du formulaire

    private $formElements;

    // Le getter pour lire le contenu de l'attribut $formElements

    public function getFormElements()
    {
        return $this->formElements;
    }

    // Méthode permettant d'ajouter un ou des attributs

    private function addAttributes(array $attributes): string
    [...]

    // Méthode permettant de démarrer le formulaire

    public function startForm(string $action = '#', string $method = 'POST', array $attributes = []): self
    [...]

    // Méthode permettant d'ajouter un label

    public function addLabel(string $for, string $text, array $attributes = []): self
    [et etc.]]
```

# DOSSIER PROFESSIONNEL (DP)

**Annexe 12** : findAll() CreationController.php :

*// On stocke dans une variable le return de la méthode findAll*

```
$list = $creations->findAll();  
  
$this->render('creation/index', ['list' => $list]);
```

**Annexe 13** : <link rel="stylesheet" href="style.css">

**Annexe 14** :

*// Méthode pour afficher une création*

```
public function showCreation($id)  
{  
  
    // Convertir l'identifiant en entier  
  
    $id = isset($_GET['id']) ? intval($_GET['id']) : 0;  
  
    // On instancie la classe CreationModel  
  
    $creationModel = new CreationModel();  
  
    // On stocke dans une variable le return de la méthode find()  
  
    $creation = $creationModel->find($id);  
  
    $this->render('creation/showCreation', ['creation' => $creation]);  
  
}
```

**Annexe 15** : showCreation.php :

<?php

*// Vérifier si la session n'est pas déjà démarrée*

```
if (session_status() == PHP_SESSION_NONE) {  
  
    session_start();  
  
}
```

*// Vérification de l'existence du jeton CSRF dans la session*

```
if (!isset($_SESSION['csrf_token'])) {  
  
    // Jeton CSRF non trouvé, générer un nouveau jeton
```

# DOSSIER PROFESSIONNEL (DP)

```
$_SESSION['csrf_token'] = bin2hex(openssl_random_pseudo_bytes(32));
}

$title = htmlspecialchars("Mon portfolio - ". $creation->title, ENT_QUOTES, 'UTF-8');

?>

<article class="row justify-content-center text-center">

    <h1 class="col-12"><?php echo htmlspecialchars($creation->title, ENT_QUOTES, 'UTF-8'); ?></h1>

    <p>Date de publication: <?php echo htmlspecialchars(date("d/m/Y", strtotime($creation->created_at)), ENT_QUOTES, 'UTF-8'); ?></p>

    title, ENT_QUOTES, 'UTF-8'); ?>">

    <p><?php echo htmlspecialchars($creation->description, ENT_QUOTES, 'UTF-8'); ?></p>

    <!-- Ajout d'un formulaire avec le jeton CSRF pour la protection CSRF -->

    <form action="#" method="POST">

        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">

    </form>

</article>
```

**Annex 16** : icone et lien :

```
<td><a href='index.php?controller=creation&action=showCreation&id=' . htmlspecialchars($value->id_creation, ENT_QUOTES, 'UTF-8') . "'><i class='fas fa-eye'></i></a></td>;
```

**Annexe 17** : Afficher les erreurs lors des tests (en plus du F12 et de sa console) :

```
// Activer l'affichage des erreurs

ini_set('display_errors', 1);

error_reporting(E_ALL);

[et etc ...]
```

**Annexe 18** : XSS : exemple (Views/creation/index.php) :

```
echo "<td>" . htmlspecialchars($value->id_creation, ENT_QUOTES, 'UTF-8') . "</td>";

echo "<td>" . htmlspecialchars($value->title, ENT_QUOTES, 'UTF-8') . "</td>";
```

# DOSSIER PROFESSIONNEL (DP)

## Annexe 19 : token :

*// Générer un token CSRF*

```
$csrfToken = bin2hex(random_bytes(32));  
  
// Stocker le token CSRF dans la session de l'utilisateur  
  
$_SESSION['csrf_token'] = $csrfToken;  
  
  
// Exemple de vérification d'authentification  
  
if (isset($_SESSION['user_id'])) {  
  
    session_regenerate_id(); // Régénérer l'identifiant de session  
  
}
```

Ou encore, lorsque l'on met en place un formulaire :

*// On construit le formulaire d'ajout*

```
$form->startForm("#", "POST", ['enctype' => "multipart/form-data"]);  
  
$form->addInput("hidden", "csrf_token", ["value" => $csrfToken]);  
  
$form->addLabel("title", "Titre", ["class" => "form-label"]);
```

*[...]*

*// Ajout du token CSRF au formulaire*

```
$form->addInput("hidden", "csrf_token", ["value" => $csrfToken, "hidden" => ""]);
```

## Annexe 20 : Détournement de session (CreationController.php)

*// Exemple de vérification d'authentification*

```
if (isset($_SESSION['user_id'])) {  
  
    session_regenerate_id(); // Régénérer l'identifiant de session  
  
}
```

## Annexe 21 : "prepare()" et "execute()" (DbConnect.php)

*// Méthode pour exécuter une requête SQL sécurisée*

---

# DOSSIER PROFESSIONNEL (DP)

---

```
public function executeQuery($sql, $params = [])
{
    try {
        // Préparation de la requête SQL

        $stmt = $this->connection->prepare($sql);

        // Vérifier si des paramètres ont été fournis et les lier à la requête
        if (!empty($params)) {
            foreach ($params as $key => $value) {
                $stmt->bindValue($key, $value);
            }
        }

        // Exécution de la requête
        $stmt->execute();

        // Renvoi du résultat de la requête
        return $stmt;
    } catch (Exception $e) {
        // Enregistrement ou affichage du message d'erreur en cas d'échec de la requête
        error_log('Erreur lors de l\'exécution de la requête SQL : '. $e->getMessage());
        return false;
    }
}
```

**Annexe 22** : Upload (image), CreationController.php :

```
// Vérification de l'extension

if (!in_array($ext, $allowedExtensions)) {
    $erreur = "Extension de fichier non autorisée.";
}

// Vérification du type MIME

elseif (!in_array($fileType, $allowedMimeType)) {
    $erreur = "Type de fichier non autorisé.";
```

---

## DOSSIER PROFESSIONNEL (DP)

---

```
}  
  
// Vérification de la taille du fichier  
  
elseif ($filesize > $maxFileSize) {  
  
    $erreur = "La taille du fichier est trop grande.";  
  
} else {  
  
    // Génération d'un nom unique pour le fichier  
  
    $uniqueName = uniqid('', true) . '.' . $ext;  
  
    // Chemin de destination  
  
    $destination = "images/" . $uniqueName;  
  
    // Déplacement du fichier uploadé vers le dossier de destination  
  
    move_uploaded_file($_FILES['picture']['tmp_name'], $destination);
```



---

## DOSSIER PROFESSIONNEL (DP)

---

---