Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance	▸	DEHEZ
Nom d'usage	▸	DEHEZ
Prénom	▸	Olivier
Adresse	▸	127 Boulevard Auguste Blanqui, 75013 PARIS

Titre professionnel visé

DEVELOPPEUR 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

Sommaire

Exemples de pratique professionnelle

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ▸ *Site CV d'Olivier DEHEZ*

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

Pour mettre en exergue le savoir acquis durant la partie front-end de la formation, j'ai décidé de réaliser mon site CV. J'ai opté pour un design One Page afin de faciliter la navigation, je trouve que cela apporte une certaine fluidité et offre des possibilités en termes de design très intéressantes.

Ce site CV a été réalisé en 5 jours approximativement, je n'ai pas rencontré de problème particulièrement, et 35% du temps de développement a été consacré à la réflexion et à la réalisation du Responsive Design.

Langages utilisés : HTML/CSS/Javascript.

Les user stories :

Une fois les bases du projet posées, il convient de rédiger les user stories du site que nous voulons créer. Un récit utilisateur (ou user story) est une phrase simple qui permet de décrire une fonctionnalité du site web du point de vue de l'utilisateur final.

Par exemple : « **En tant qu**'utilisateur, **je veux pouvoir** consulter la page *Réalisations* **afin de** connaître tous mes projets en développement. » En fin de projet, ces récits permettent de s'assurer que toutes les fonctionnalités requises ont bien été développées.

User stories du site Mon CV :

- **Accueil**

« **En tant qu**'utilisateur, **je veux pouvoir** consulter une page d'accueil présentant les informations essentielles (nom, métier, accroche) **afin d**'avoir une vision rapide du profil. »

- **À propos de moi**

« **En tant qu**'utilisateur, **je veux pouvoir** consulter la page *À propos de moi* **afin de** découvrir le parcours, la personnalité et les principales compétences de l'auteur du site. »

- **Compétences**

« **En tant qu**'utilisateur, **je veux pouvoir** visualiser les compétences techniques et transversales, regroupées par domaine (front-end, back-end, outils), **afin d'**évaluer le niveau de maîtrise de chaque technologie. »

- **CV**

« **En tant qu**'utilisateur, **je veux pouvoir** accéder à la section *CV*, **afin de** consulter les expériences professionnelles, les formations suivies et télécharger le CV au format PDF. »

- **Réalisations**

« **En tant qu**'utilisateur, **je veux pouvoir** consulter la page *Réalisations*, **afin de** découvrir les projets développés, les technologies utilisées et les liens vers les démos ou les dépôts GitHub. »

- **Navigation**

« **En tant qu**'utilisateur, **je veux pouvoir** naviguer facilement entre les différentes sections du site via un menu latéral clair et réactif, **afin de** trouver rapidement les informations souhaitées. »

- **Accessibilité / Responsive**

« **En tant qu**'utilisateur, **je veux** que le site s'adapte automatiquement à mon appareil (ordinateur, tablette, mobile), **afin d'**avoir une expérience fluide quelle que soit la taille de l'écran. »

Les wireframes pour se projeter :

Maintenant, il convient de réfléchir sur l'ergonomie, à savoir la capacité à répondre efficacement aux attentes des utilisateurs et à leur fournir un confort de navigation. Il faut également tenir compte du parcours visuel en « **zigzag** » commençant à l'angle supérieur gauche, jusqu'à l'angle inférieur droit.

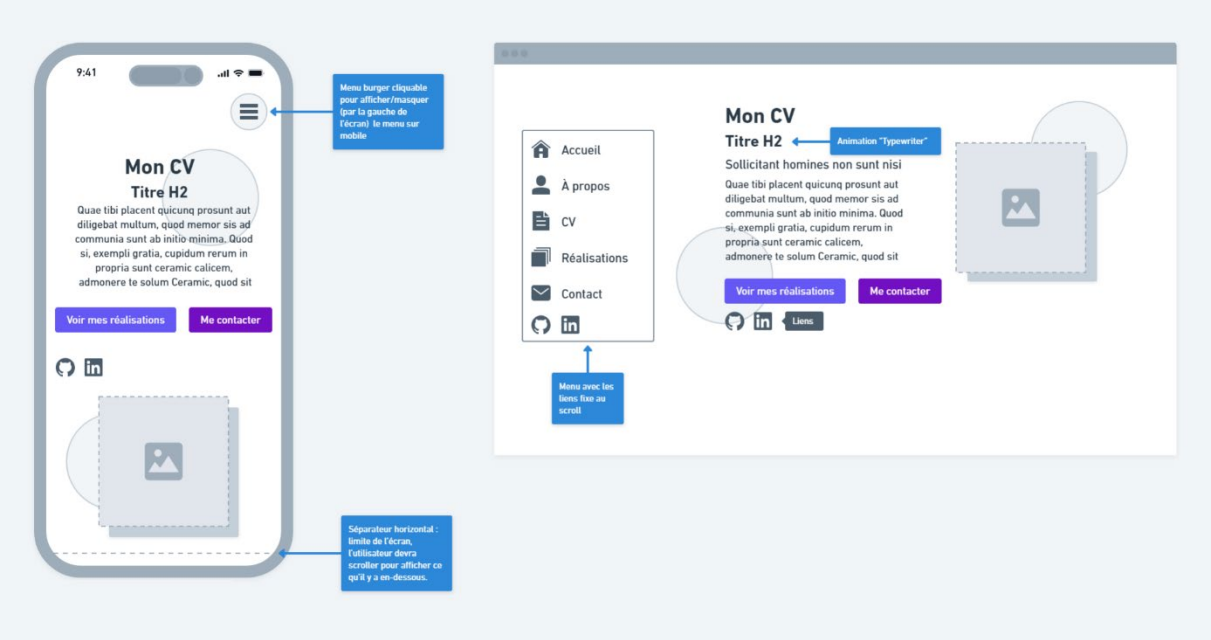
La conception et l'organisation du site sur une seule page est réalisée à l'aide, dans un premier temps, de la technique de « **zoning** » consistant à représenter le site à l'aide de boites qui symbolisent les grandes fonctionnalités et les zones principales de contenu. On résonnera ici qu'en terme de zones (boites). Chaque boite (carrés, rectangles …) représente une entité majeure du site qui doit trouver sa place sur l'écran en fonction deux critères principaux :

- Son importance hiérarchique.
- Son encombrement spatial.

Dans un deuxième temps on détaille le contenu des « boites » définies dans l'étape de zoning. Il s'agit de réfléchir non pas sur le graphisme mais sur les aspects fonctionnels et ergonomiques des éléments qui rempliront ces boites. Le but est de proposer un aperçu graphique du futur site grâce à une série de wireframes. Les wireframes, à ne pas confondre avec les maquettes, sont des schémas utilisés lors de la conception d'une interface utilisateur pour définir les zones et les composants que cette interface doit contenir. À contrario, la maquette est une version aboutie du site que le développeur front-end devra reproduire à l'identique.

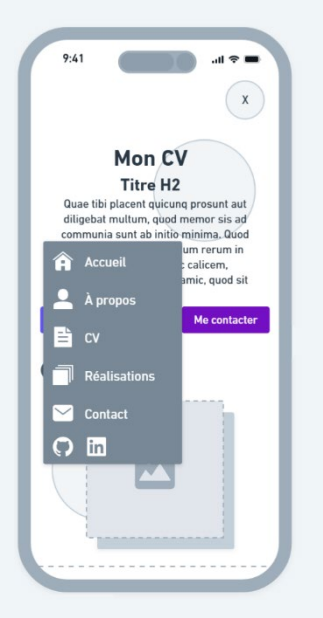
Pour ce projet, j'ai réalisé deux wireframes pour chaque page : un wireframe pour la version mobile et un wireframe pour la version desktop.

Wireframes de la page d'accueil du site (vue globale) :



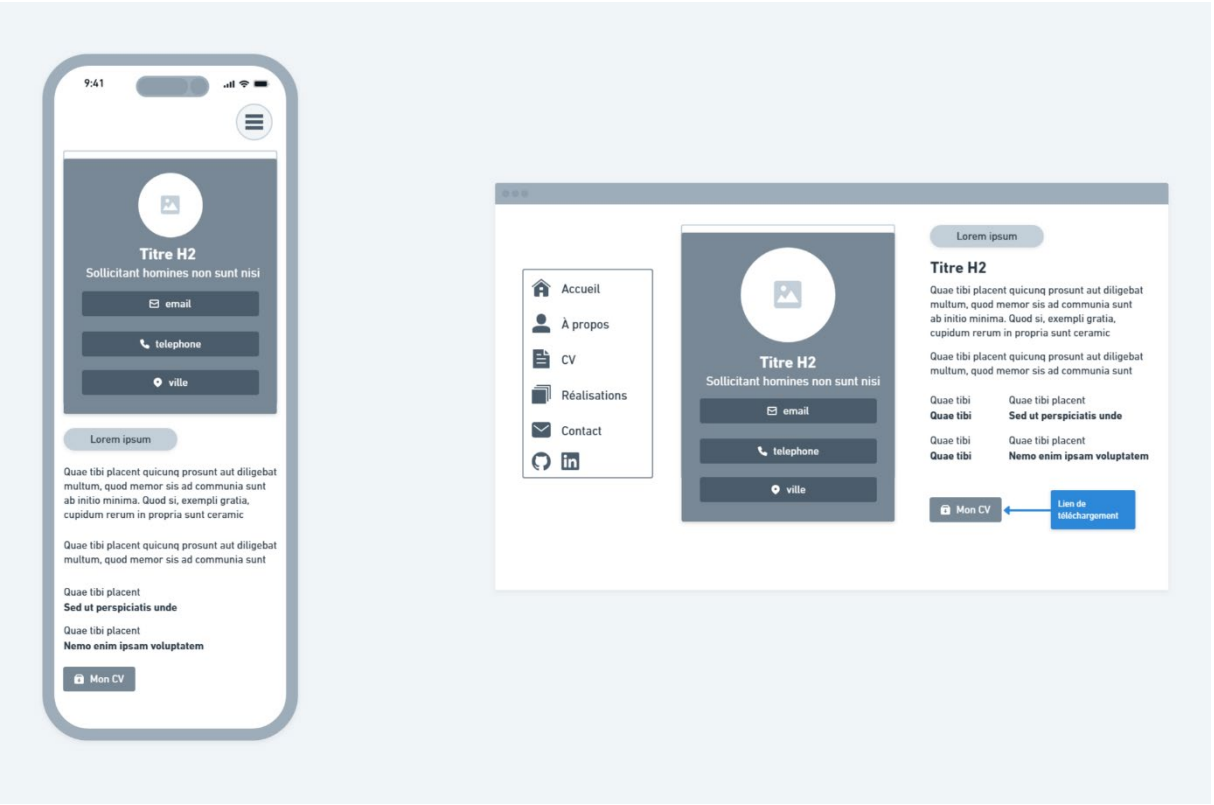
Sur ces wireframes, on peut voir la déclinaison mobile et la déclinaison desktop de la page d'accueil du futur site web CV.

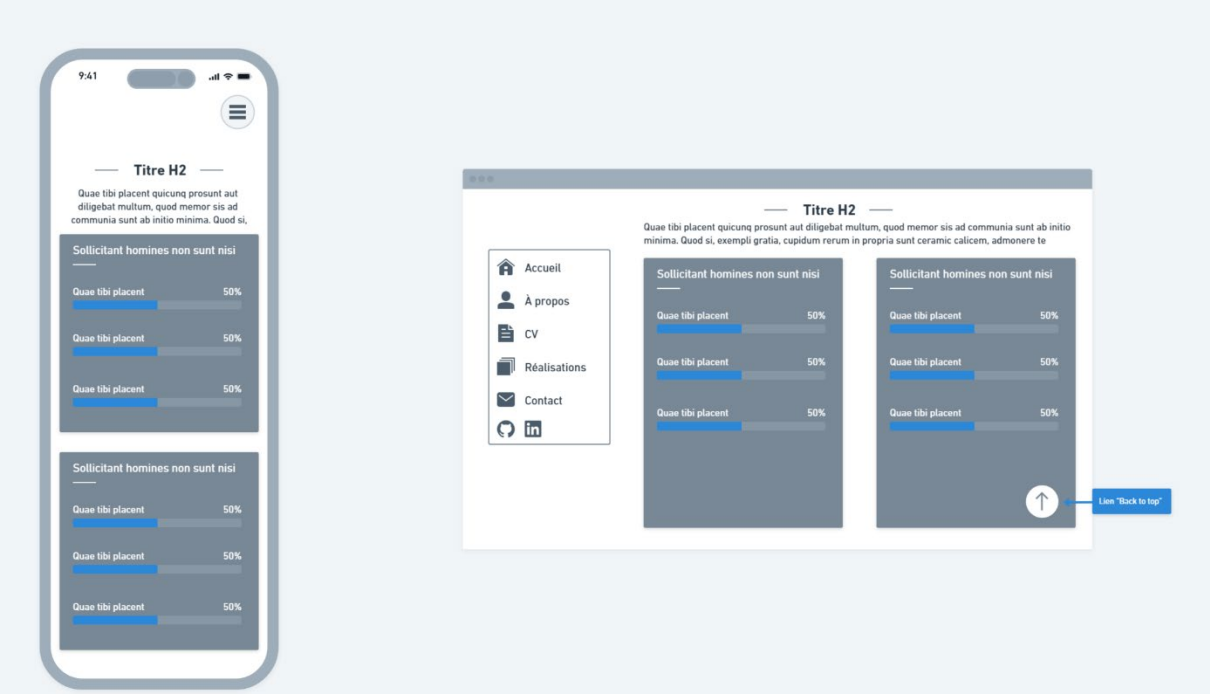
Wireframes de la page d'accueil du site (détails version mobile) :



Quand on crée des wireframes, il faut être le plus visuel possible pour que notre client / product owner puisse se projeter. Cela passe donc par la mise en place de formes simples et parlantes. Si nous avons besoin de préciser une fonctionnalité, il convient de placer des annotations comme je l'ai fait ici avec le menu burger.

Wireframes de la section « A propos de moi » (vue globale) :





Réaliser des interfaces utilisateur statiques web ou web mobile :

J'ai réalisé ce site en HTML en respectant la sémantique et les bonnes pratiques d'organisation du contenu d'une page web et effectué la mise en page à l'aide du framework CSS Bootstrap.

Pour rendre mon code le plus lisible possible, notamment pour le référencement naturel / SEO (Search Engine Optimization) et pour les lecteurs d'écran des personnes en situation de handicap, je l'ai découpé en blocs logiques qui permettent de bien comprendre la structuration de mon code.

Il convient aussi de développer plusieurs types d'affichage et d'interactions en fonction du device de l'utilisateur final : c'est ce qu'on appelle ***la conception responsive***.

Le code HTML est découpé en bloc « logique » pour respecter la sémantique du langage HTML. Prenons le bloc `<body>` de la page d'accueil :

```
<body class="index-page">
  <header id="header" class="header dark-background d-flex flex-column justify-
content-center"> …
  </header>

  <main class="main"> …
  </main>
  <footer id="footer" class="footer position-relative"> …
  </footer>
```

```
</body>
```

On voit ici que le bloc <body> est découpé en 3 sous-blocs logiques : un **<header>** qui va permettre notamment de placer une balise <nav>, un <main> où sera développé le contenu de la page et un <footer> pour les informations de pied-de-page.

Faire du HTML sémantique est très important pour de nombreuses raisons comme le référencement ou l'accessibilité par exemple. Et mon code sera aussi facilement compréhensible par les futurs développeurs qui arriveront sur le projet.

Pour que le site soit **responsive**, c'est à dire qu'il s'adapte au format du device utilisé (mobile ou desktop, en l'occurrence), il faut mettre en place des ***media queries*** : un outil de responsive design qui permet d'adapter la feuille de style CSS en fonction de différents paramètres ou caractéristiques de l'appareil. On peut, par exemple, appliquer différentes mises en forme selon la taille de la zone d'affichage de l'appareil utilisé.

Extrait du fichier CSS :

```
@media (min-width: 1200px) {
    .header~main,
    .header~#footer {
        margin-left: 330px;
    }
}
@media (max-width: 1199px) {
    .header {
        left: -100%;
    }
}
@media (max-width: 768px) {
  .header {
    position: static;
    width: 100%;
  }
  .navmenu ul {
    flex-direction: row;
    justify-content: center;
  }
  .navmenu a {
    padding: 10px;
    font-size: 14px;
  }
}
```

Extrait du fichier HTML

```
  <header id="header" class="header dark-background d-flex flex-column justify-
content-center">
    <div class="header-container d-flex flex-column align-items-start">
      <nav id="navmenu" class="navmenu">
        <ul>
          <li><a href="#hero" class="active"><i class="bi bi-house
navicon"></i>Accueil</a></li>
          <li><a href="#about"><i class="bi bi-person navicon"></i> À propos de
moi</a></li>
          <li><a href="#resume"><i class="bi bi-file-earmark-text navicon"></i>
CV</a></li>
          <li><a href="#portfolio"><i class="bi bi-images navicon"></i>
Réalisations</a></li>
          <li><a href="#contact"><i class="bi bi-envelope navicon"></i>
Contact</a></li>
        </ul>
      </nav>
    </div>
  </header>
```

Ici, les styles à l'intérieur de la règle de Media Query ne seront appliqués que lorsque la largeur de l'écran est de 768 pixels ou moins.

2. Précisez les moyens utilisés :

Pour réaliser ces wireframes, j'ai utilisé le site whimsical.com qui permet de réaliser facilement des wireframes grâce à la mise à disposition de bibliothèques d'éléments et d'icônes.

Pour la rédaction du code :

- IDE : VISUAL STUDIO CODE,
- Supports de cours.

3. Avec qui avez-vous travaillé ?

Pour ce projet j'ai décidé de travailler en autonomie. C'était pour moi l'occasion de tester mes connaissances sur un projet très personnel.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ **STUDI**

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du** 25/08/2025 **au** 29/08/2025

5. Informations complémentaires *(facultatif)*

Le repository de ce projet personnel est accessible ici : **https://github.com/olivier435/cv-one-page-od**

Le mini-site web est accessible ici : **https://olivier435.github.io/cv-one-page-od/**

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°2 ▸ *Création d'un formulaire avec JavaScript*

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

Dans le cadre de la compétence « **Développer une interface utilisateur web dynamique** », j'ai réalisé la conception et le développement d'un formulaire interactif permettant à l'utilisateur de passer une commande et d'obtenir en temps réel le calcul du montant total TTC en fonction de ses choix et de ses quantités. L'objectif était d'offrir une expérience fluide, réactive et ergonomique, tout en intégrant les recommandations en matière de sécurité et d'accessibilité.

Cette activité m'a permis de mobiliser mes compétences en HTML, CSS, JavaScript et Bootstrap, mais aussi d'appliquer une logique de programmation rigoureuse et structurée.

Objectif : Effectuer des vérifications et des calculs dans un formulaire.

Langages utilisés : HTML/CSS/JavaScript.

Demandes :
Les champs grisés se remplissent automatiquement en fonction des choix et des quantités saisies par l'internaute. Le total TTC sera affiché avec 2 chiffres après la virgule. Chaque liste déroulante propose différentes options.
Fonctionnalité des boutons :

- Vérifier avant envoi déclenche les opérations suivantes : contrôle des champs obligatoires (si le champ n'est pas rempli, celui-ci prend une bordure rouge et un message s'affiche dans une boîte de dialogue) ; contrôle la présence de "@" dans le champ e-mail ; le nom passe en majuscule.
- Imprimer déclenche l'impression de la page au format paysage.
- Réinitialiser efface les données saisies
- Envoyer ma commande envoie le tout par mail (mailto) si le formulaire a été vérifié au préalable.

1) Création du formulaire en HTML et CSS

L'interface a été bâtie à partir des composants de **Bootstrap 5**, afin de bénéficier d'un rendu responsive et cohérent sur tous types d'écrans. Les éléments de formulaire sont disposés à l'aide de la grille Bootstrap (`row`, `col-md-*`) et enrichis de classes comme `form-control`, `input-group` ou `invalid-feedback`, garantissant une présentation claire des erreurs et un retour visuel immédiat pour l'utilisateur. Les champs non modifiables, tels que les sous-totaux ou le total TTC, utilisent la classe `form-control-plaintext` : ils restent lisibles, mais inaccessibles à la saisie, ce qui prévient les modifications involontaires.

Chaque champ est associé à une **étiquette sémantique** (`<label for="…">`) pour assurer la compatibilité avec les lecteurs d'écran et les outils d'assistance. De plus, le champ du total TTC est muni d'un attribut `aria-`

live="polite", permettant d'annoncer automatiquement la mise à jour du montant aux utilisateurs ayant un lecteur d'écran, sans perturber la navigation. Ces éléments garantissent la **conformité aux bonnes pratiques d'accessibilité**.

2) JavaScript

Sur le plan fonctionnel, le JavaScript repose sur une approche structurée et modulaire. Les tarifs sont centralisés dans un objet constant TARIFS, jouant le rôle de table de correspondance (mapping). Cette architecture évite la multiplication des conditions (if / else if) et permet une évolution aisée du code (ajout d'un nouveau type de prestation sans modifier la logique principale).

a) Séparation logique : table des tarifs et formatage

```
const TARIFS = Object.freeze({
    demijour: 8,
    jour: 15,
    repas: 7,
});

const TVA = 0.20;
const DEBUG = false;

const fmt = (n) => Number(n || 0).toLocaleString('fr-FR', {
    minimumFractionDigits: 2,
    maximumFractionDigits: 2
});
```

- La constante TARIFS centralise les prix : **pas de if/elseif** dispersés.
- **Object.freeze** : empêche toute modification accidentelle.
- **toLocaleString('fr-FR')** : virgule décimale + 2 chiffres.

b) Calculs dynamiques : une fonction, un rôle

```
function getLignes() {
    return Array.from(document.querySelectorAll('.ligne')).map((row) => {
        const select = row.querySelector('.choix');
        const qteEl = row.querySelector('.nbpl');
        const stEl = row.querySelector('.stt');
        return {
            select,
            qteEl,
            stEl
        };
    });
}
```

```
function calculeSousTotals() {
    let sousTotal = 0;

    getLignes().forEach(({
        select,
        qteEl,
        stEl
    }) => {
        const type = select.value;
        const prix = TARIFS[type] ?? 0;
        const qte = Number.parseInt(qteEl.value, 10);

        if (Number.isNaN(qte) || qte < 0) {
            qteEl.classList.add('is-invalid');
            stEl.value = fmt(0);
            return;
        }
        qteEl.classList.remove('is-invalid');

        const st = qte * prix;
        stEl.value = fmt(st);
        sousTotal += st;
    });

    document.getElementById('st').value = fmt(sousTotal);
    const ttc = sousTotal * (1 + TVA);
    document.getElementById('total').value = fmt(ttc);

    debugLog('Sous-total HT:', sousTotal, 'Total TTC:', ttc);
}
```

- Validation des quantités (retour visuel et calcul sûr).
- Sélection par ligne : forte cohésion select/qté/sous-total.
- Contrôle négatifs/NaN : champs marqués invalides + calcul sûr.
- L'appel multiple de calculeSousTotals() ne casse rien.

c) Validation téléphone (regex FR robuste)

```
const telRegex = /^(?:0[1-9](?:[ .-]?\d{2}){4}|(?:\+33|0033)[ .-]?[1-9](?:[ .-
]?\d{2}){4})$/;

function valideTel(input) {
    const value = (input.value || '').trim();
    const ok = telRegex.test(value);
```

```
    ok ? clearInvalid(input) : setInvalid(input, 'Numéro invalide (ex: 01 23 45 67 89).');
    return ok;
}
```

- Autorise 01 23 45 67 89, 01.23.45.67.89, +33 1 23 45 67 89, etc.
- **Évite** les numéros trop courts/longs ou mal formés.

d) Processus « Vérifier », « Envoyer »

```
let wasVerified = false;

function verifierAvantEnvoi() {
    const form = document.getElementById('monFormulaire');
    const nom = document.getElementById('nom');
    const prenom = document.getElementById('prenom');
    const tel = document.getElementById('tel');
    const mail = document.getElementById('mail');
    const cg = document.getElementById('cg');

    if (!form.checkValidity()) {
        form.classList.add('was-validated');
    }

    const validations = [
        valideNom(nom),
        validePrenom(prenom),
        valideTel(tel),
        valideEmail(mail),
        valideCG(cg),
    ];

    calculeSousTotals();
    const qtesOK = Array.from(document.querySelectorAll('.nbpl'))
        .every((el) => !el.classList.contains('is-invalid'));

    const ok = validations.every(Boolean) && qtesOK;

    wasVerified = ok;
    if (!ok) {
        alert('Merci de corriger les champs indiqués en rouge.');
    } else {
        alert('☑ Formulaire vérifié, vous pouvez envoyer votre commande.');
    }
```

```
    debugLog('Vérification globale OK ?', ok);
    return ok;
}

function envoyer() {
    if (!wasVerified) {
        alert('Veuillez d\'abord cliquer sur "Vérifier avant envoi".');
        return;
    }
    document.getElementById('monFormulaire').submit();
}
```

- Impossible d'envoyer sans vérification réussie.
- **État unique** wasVerified : garde-fou.
- Séparation de la vérification et de la soumission.

e) Raccordement des événements

```
window.addEventListener('DOMContentLoaded', () => {
    document.getElementById('imprimer').addEventListener('click', impression);
    document.getElementById('verifier').addEventListener('click',
verifierAvantEnvoi);
    document.getElementById('envoyer').addEventListener('click', envoyer);
    document.getElementById('monFormulaire').addEventListener('reset', () => {
        setTimeout(resetForm, 0);
    });

    document.querySelectorAll('.choix').forEach((el) =>
el.addEventListener('change', calculeSousTotals));
    document.querySelectorAll('.nbpl').forEach((el) =>
el.addEventListener('input', calculeSousTotals));

    calculeSousTotals();
});
```

- Écouteurs centrés sur les interactions utiles (change/input).

f) Extensibilité : ajout d'une prestation ou d'une ligne

```
const TARIFS = Object.freeze({
    demijour: 8,
    jour: 15,
    repas: 7,
    soiree: 9
});
```

Il conviendra d'ajouter une ligne dans le HTML (div.ligne ...) et la prise en compte sera automatique par getLignes() et calculeSousTotals().

En matière de **sécurité front-end**, plusieurs principes ont été appliqués. Les champs calculés (sous-totaux et total TTC) ne sont pas modifiables par l'utilisateur et ne servent qu'à l'affichage : ils ne doivent jamais être considérés comme des données fiables. En contexte professionnel, ces montants seraient **recalculés côté serveur** à partir des choix et quantités, afin d'éviter toute manipulation des valeurs via les outils de développement. De même, les vérifications côté client ont pour rôle principal d'améliorer l'ergonomie et de réduire les erreurs, mais ne sauraient remplacer les contrôles de sécurité et de validation réalisés côté serveur (vérification du format des données, encodage, filtrage des entrées, etc.). Cette distinction claire entre la logique front-end (UX, confort d'utilisation) et la logique back-end (sécurité et intégrité des données) est une bonne pratique professionnelle.

2. Précisez les moyens utilisés :

- IDE : VISUAL STUDIO CODE,
- Supports de cours.

3. Avec qui avez-vous travaillé ?

Pour ce projet j'ai décidé de travailler en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ **STUDI**

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du** 21/10/2025 **au** 22/10/2025

5. Informations complémentaires *(facultatif)*

Le repo de ce projet est accessible à l'adresse suivante : https://github.com/olivier435/verifications-form.

Le mini-site web est accessible ici : https://olivier435.github.io/verifications-form/

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ▸ Mettre en place une base de données relationnelle

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

Dans le cadre d'un projet mené durant la formation dédiée à la **programmation orientée objet en PHP**, nous avons conçu et développé une **plateforme web de mise en relation entre chercheurs d'emploi et recruteurs du secteur informatique**.

Cette application permet :

- aux utilisateurs de consulter les offres disponibles et d'y postuler en ligne,
- aux entreprises de créer et publier leurs propres offres d'emploi depuis un espace dédié.

Dictionnaire des données (sous JMerise)

Lors de la conception de la base de données du projet **Jobiz**, j'ai utilisé **JMerise** pour générer automatiquement le **dictionnaire des données** à partir de mon **MCD**. Cet outil me permet de décrire **de manière structurée et normalisée** l'ensemble des champs qui composeront les tables de la base.

Le dictionnaire des données joue un rôle essentiel dans la phase de conception :

- il **formalise la correspondance** entre les concepts métier du MCD et les futures colonnes physiques des tables,
- il **garantit la cohérence des types de données** (chaînes de caractères, entiers, dates, etc.),
- il **facilite la communication entre les développeurs et les concepteurs**,
- et il sert de **référence unique** avant la génération du modèle logique ou du script SQL.

Pour coller au plus près à la forme finale de ma base de données, j'ai choisi de **traduire les noms de champs** définis en français dans le MCD vers l'**anglais** dans le dictionnaire des données.

Cette convention facilite l'intégration avec le code PHP et respecte les standards de nommage utilisés dans les frameworks modernes (camelCase / snake_case).

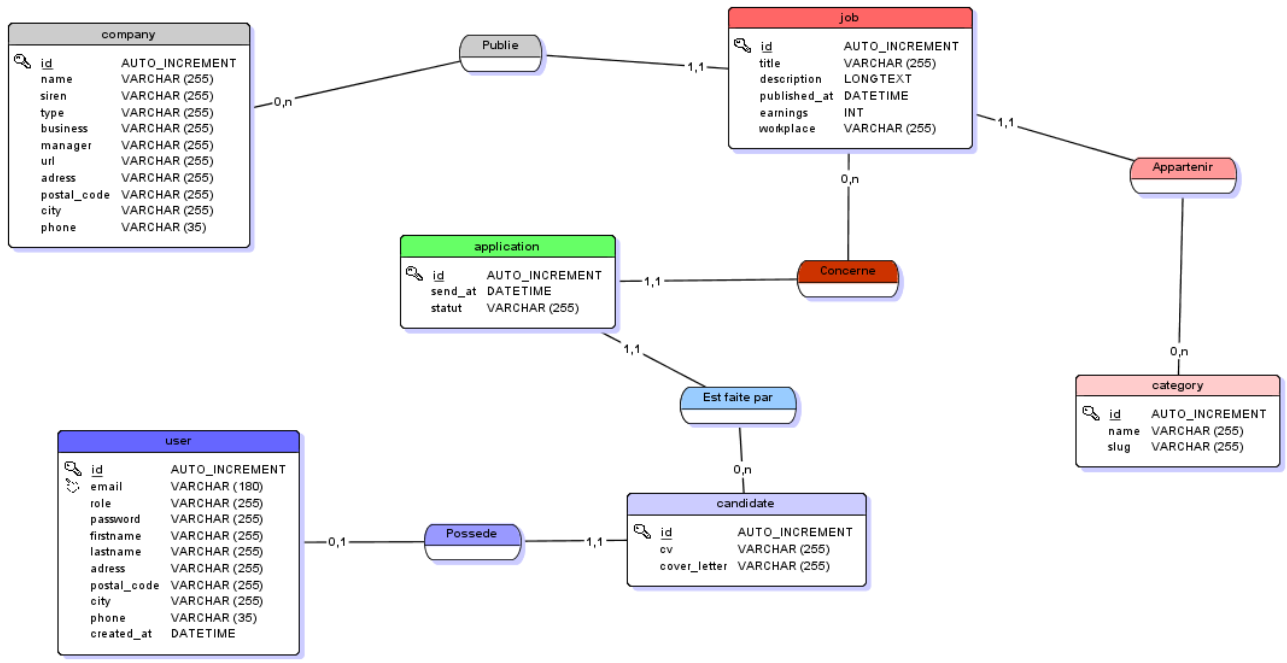
Exemple de dictionnaire des données :

Entité : User

Field	Type	Details	Description
id	Integer	Primary Key, Auto Increment, Not Null	User's unique identifier.
lastname	Varchar(255)	Not Null	User's last name.
firstname	Varchar(255)	Not Null	User's first name.
email	Varchar(180)	Unique, Not Null	User's email address used for login.
password	Varchar(255)	Not Null	Encrypted password of the user.
phone	Varchar(35)	Nullable	User's phone number (optional).
adress	Varchar(255)	Nullable	Street and number of the user's address.
city	Varchar(255)	Nullable	User's city of residence.
postal_code	Varchar(255)	Nullable	User's postal code.
role	Varchar(255)	Default = 'user'	Defines user permissions (candidate / company / admin).
created_at	Datetime	Not Null	Date and time when the user account was created.

Le MCD, Modèle Conceptuel de Données

Pour visualiser au mieux ce projet et comprendre les relations entre les différentes tables de la base de données à créer, j'ai créé un MCD : un Modèle Conceptuel des Données.



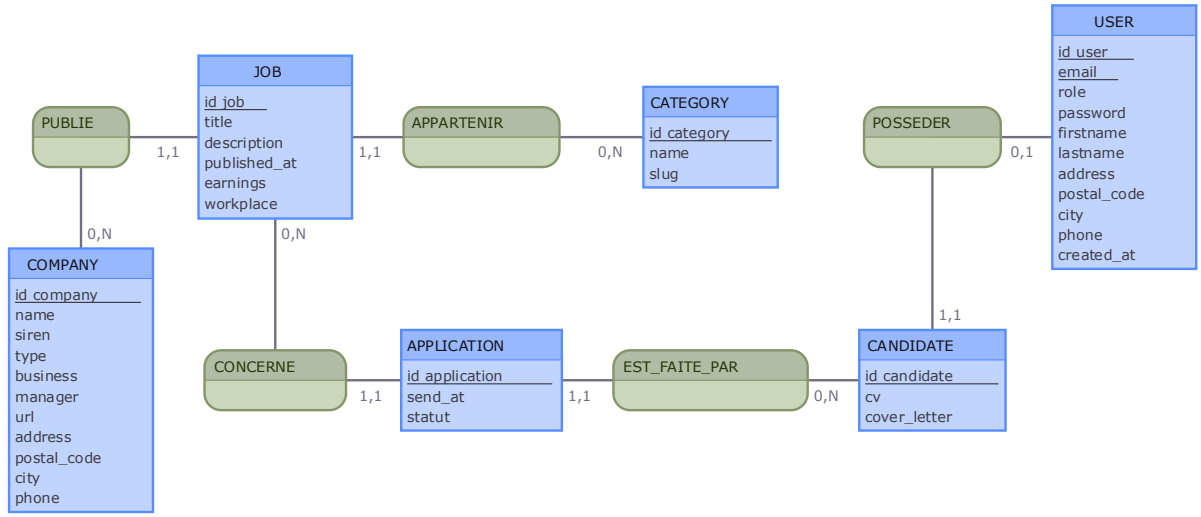
Choix et évolution des outils de modélisation

Au départ, j'ai conçu le **Modèle Conceptuel de Données (MCD)** du projet *Jobiz* à l'aide du logiciel **JMerise**. Cet outil graphique m'a permis de **visualiser facilement les entités**, leurs **attributs**, ainsi que les **associations** et **cardinalités** grâce à une interface intuitive.
JMerise facilite la compréhension des liens logiques entre les tables et reste particulièrement adapté à une **phase d'apprentissage ou de validation visuelle**.

Cependant, lors de la rédaction du dossier, j'ai souhaité **adopter une approche plus textuelle et formelle**, afin de pouvoir :

- **décrire la structure du modèle sous forme de script**,
- **générer automatiquement le diagramme** en différents formats (SVG, PNG, SQL),
- **éviter les décalages graphiques** liés aux déplacements manuels des entités,
- et **documenter la conception de manière reproductible**.

Pour cela, j'ai utilisé **Mocodo**, un outil en ligne basé sur une **syntaxe Merise déclarative**. Mocodo m'a permis de réécrire mon MCD sous forme de script textuel, garantissant une **traçabilité claire** entre la logique métier (le texte) et la représentation graphique (le diagramme généré automatiquement).



Le **modèle conceptuel des données** (MCD) est une représentation logique des informations manipulées par le système d'information.

Il permet de **décrire de manière claire et structurée** les données utilisées par l'application ainsi que les **relations** existantes entre elles.

Les informations sont modélisées à l'aide de règles et de conventions graphiques propres à la méthode **Merise** :

- **Les entités** : représentées par des rectangles, elles correspondent aux objets manipulés par le système. *Exemple : le rectangle USER représente une entité contenant les informations relatives aux utilisateurs de la plateforme.*
- **Les propriétés (ou attributs)** : elles décrivent les caractéristiques propres à chaque entité. *Exemple : l'entité USER possède les attributs id, firstname, lastname, email, etc.*
- **Les relations** : elles expriment les liens logiques entre les entités.
- *Exemple : la relation PUBLIE relie les entités COMPANY et JOB pour indiquer qu'une entreprise publie une ou plusieurs offres.*
- **Les cardinalités** : exprimées par les symboles (0, 1, N), elles précisent combien d'occurrences d'une entité peuvent être associées à une autre.
  Elles sont toujours exprimées de part et d'autre d'une relation.

Explications des principales relations

1. **CANDIDATE ↔ USER (possède)**
   La table candidate est associée à la table user par une cardinalité **(1,1)** côté candidate, car chaque candidat est obligatoirement lié à un utilisateur unique.
   Inversement, la cardinalité **(0,1)** côté user indique qu'un utilisateur peut ne pas être candidat (par exemple, un recruteur ou un administrateur), ou ne l'être qu'une seule fois.
2. **COMPANY ↔ JOB (publie)**
   La relation publie indique qu'une **entreprise peut publier zéro ou plusieurs offres d'emploi** — cardinalité **(0,N)** côté company.
   Chaque **offre** appartient obligatoirement à **une seule entreprise**, d'où la cardinalité **(1,1)** côté job.

3. **APPLICATION ↔ CANDIDATE (est faite par)**
   Une candidature est toujours **faite par un seul candidat** — cardinalité **(1,1)** côté application.
   Un candidat peut, quant à lui, **postuler à plusieurs offres**, d'où la cardinalité **(0,N)** côté candidate.
4. **APPLICATION ↔ JOB (concerne)**
   Une candidature **concerne une seule offre** — cardinalité **(1,1)** côté application.
   Une offre peut, à l'inverse, **recevoir plusieurs candidatures**, d'où la cardinalité **(0,N)** côté job.
5. **JOB ↔ CATEGORY (appartient à)**
   Chaque offre appartient à **une seule catégorie** — cardinalité **(1,1)** côté job.
   Une catégorie peut regrouper plusieurs offres, d'où la cardinalité **(0,N)** côté category.

Le MLD, Modèle Logique de Données

Une fois le **MCD** défini, je peux mettre en place un **Modèle Logique de Données**, ou **MLD**, qui est la représentation des données d'un système d'information. À partir du MCD, le MLD introduit donc les éléments, les relations et les détails contextuels qui vont être essentiels à la structuration des données.

Voici le MLD correspondant à mon MCD :

TABLE **USER** (id user, email, role, password, firstname, lastname, address, postal_code, city, phone, created_at)

TABLE **CANDIDATE** (id candidate, cv, cover_letter, *#id user*)

TABLE **COMPANY** (id company, name, siren, type, business, manager, url, address, postal_code, city, phone)

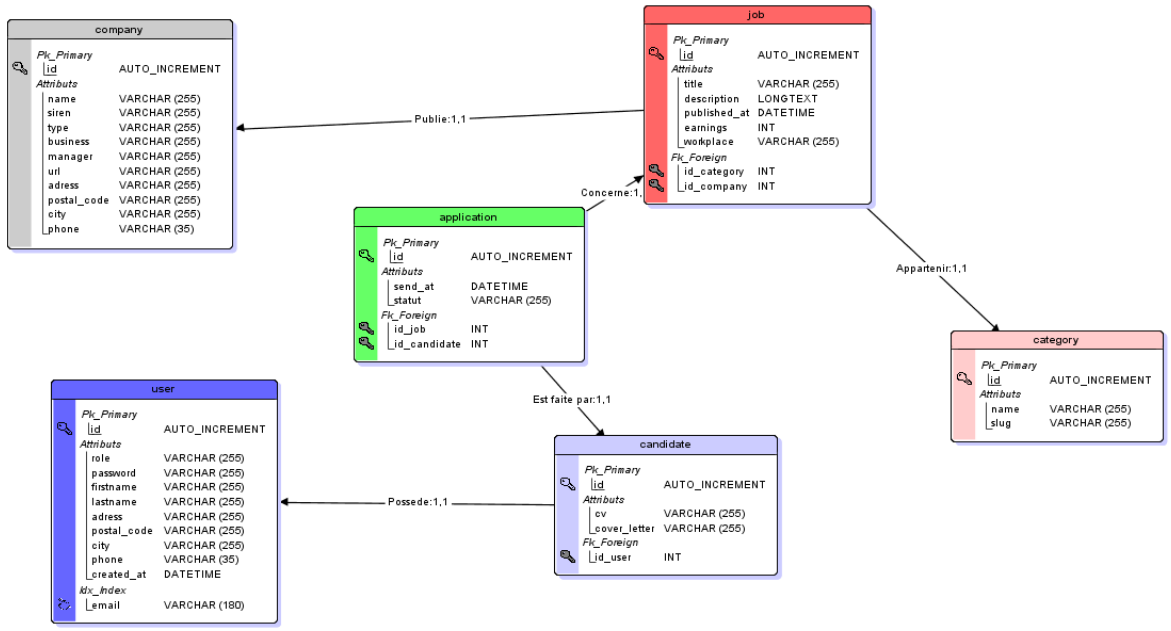
TABLE **JOB** (id job, title, description, published_at, earnings, workplace, *#id company*, *#id category*)

TABLE **CATEGORY** (id category, name, slug)

TABLE **APPLICATION** (id application, send_at, statut, *#id job*, *#id candidate*)

Côté vocabulaire et conventions :

- On note que les entités du MCD deviennent des tables dans le MLD.
- Les clés principales, appelées clés primaires, sont soulignées ainsi que les contraintes.
- Les clés étrangères, c'est à dire une clé qui fait référence à la clé primaire d'une autre table, sont représentées en étant préfixées du symbole # dièse.



Le Modèle Physique de Données (MPD)

Après la rédaction du **Modèle Logique de Données (MLD)**, j'ai pu passer à l'étape suivante : la conception du **Modèle Physique de Données (MPD)**.
Cette étape consiste à traduire le modèle logique en une **structure exploitable par le Système de Gestion de Base de Données (SGBD)** — ici, **MySQL**.

Le **MPD** représente la structure finale de la base de données telle qu'elle sera effectivement créée : il permet de **visualiser les tables, les liens physiques entre elles** et les **contraintes d'intégrité** (clés primaires, clés étrangères, unicité, etc.).

Le vocabulaire évolue par rapport au MCD :

- **Les entités** deviennent des **tables** dans la base de données.
  Exemple : l'entité USER devient la table user.
- **Les propriétés** deviennent des **champs** ou **colonnes** des tables.
  Exemple : firstname, lastname, email, etc.
- **Les propriétés situées dans une relation** peuvent soit :
  - être **intégrées à l'une des tables existantes**,
  - soit **donner naissance à une nouvelle table** lorsque la relation est de type N:N.
    Exemple : la relation POSTULER entre CANDIDATE et JOB est matérialisée par la table application.
- **Les identifiants** deviennent des **clés primaires (PRIMARY KEY)**.
  Chaque table doit au minimum posséder une clé primaire unique, souvent auto-incrémentée.
- Les **relations** et **cardinalités** définies dans le MCD sont traduites, au niveau du **Modèle Physique de Données (MPD)**, en **clés étrangères** (*FOREIGN KEY*). Une clé étrangère est un champ d'une table qui **référence la clé primaire d'une autre table**, afin de **matérialiser le lien logique** existant entre deux entités.

```
#------------------------------------------------------------
#        Script MySQL.
#------------------------------------------------------------


#------------------------------------------------------------
# Table: user
#------------------------------------------------------------

CREATE TABLE user(
        id          Int  Auto_increment  NOT NULL ,
        role        Varchar (255) NOT NULL ,
        password    Varchar (255) NOT NULL ,
        firstname   Varchar (255) NOT NULL ,
        lastname    Varchar (255) NOT NULL ,
        adress      Varchar (255) NOT NULL ,
        postal_code Varchar (255) NOT NULL ,
        city        Varchar (255) NOT NULL ,
        phone       Varchar (35) NOT NULL ,
        created_at  Datetime NOT NULL ,
        email       Varchar (180) NOT NULL
        ,CONSTRAINT user_Idx INDEX (email)
        ,CONSTRAINT user_PK PRIMARY KEY (id)
)ENGINE=InnoDB;


#------------------------------------------------------------
# Table: candidate
#------------------------------------------------------------

CREATE TABLE candidate(
        id           Int  Auto_increment  NOT NULL ,
        cv           Varchar (255) NOT NULL ,
        cover_letter Varchar (255) NOT NULL ,
        id_user      Int NOT NULL
        ,CONSTRAINT candidate_PK PRIMARY KEY (id)

        ,CONSTRAINT candidate_user_FK FOREIGN KEY (id_user) REFERENCES user(id)
        ,CONSTRAINT candidate_user_AK UNIQUE (id_user)
)ENGINE=InnoDB;


#------------------------------------------------------------
# (Suites des tables : company, category, job, application)
```

Ce script SQL, généré automatiquement à partir du modèle physique dans **JMerise**, garantit la conformité de la base de données avec la modélisation conceptuelle initiale.
Il constitue une **preuve de cohérence** entre les étapes d'analyse, de conception et d'implémentation du projet *Jobiz*.

2. Précisez les moyens utilisés :

Pour réaliser les différents schémas et tableaux de cette compétence professionnelle, j'ai utilisé :

- MOCODO Online et **JMerise** pour réaliser le Modèle Conceptuel de Données,
- Le site SQL.sh pour bien comprendre le code SQL délivré par mon MPD.

3. Avec qui avez-vous travaillé ?

Pour ce projet j'ai décidé de travailler en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸	**STUDI**
Chantier, atelier, service ▸	Cliquez ici pour taper du texte.
Période d'exercice ▸	**Du** 06/01/2026 **au** 07/01/2026

5. Informations complémentaires *(facultatif)*

En annexe vous trouverez une capture d'écran du projet (annexe 3)

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.	

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] .....,

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 ..... le .....

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

(Si le RC le prévoit)

DOSSIER PROFESSIONNEL (DP)