|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Nom de naissance* | ► | *Dreiding* |
| *Nom d’usage* | ► |  |
| *Prénom* | ► | *Georges* |
| *Adresse* | ► | *149 rue des Rabats, 92160 Antony* |
|  |  |

|  |  |
| --- | --- |
| **Titre professionnel visé** | |
|  | |
| *Développeur web et web mobile* | |
|  | |
| **Modalité d’accès :** | |
|  | |
| X | 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**](http://travail-emploi.gouv.fr/titres-professionnels) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sommaire** | | | |
| **Exemples de pratique professionnelle** | | |
| **Développer la partie front-end d’une application web ou web mobile sécurisée** | | **p.** | **5** |
|  | ► CP 3 : Réaliser des interfaces utilisateur statiques web ou web mobile. p. | p. |  |
|  | ► CP 4 : *Développer la partie dynamique des interfaces utilisateur web ou web mobile.* p. | p. |  |
|  |  |  |  |
| **Développer la partie back-end d’une application web ou web mobile sécurisée** | | **p.** |  |
|  | ► CP 6 : *Développer des composants d’accès aux données SQL et NoSQL.* p. | p. |  |
|  | ► CP 5 : *Mettre en place une base de données relationnelle.* p. | p. |  |
|  |  |  |  |
| **Titres, diplômes, CQP, attestations de formation** *(facultatif)* | | **p.** |  |
| **Déclaration sur l’honneur** | | **p.** |  |
| **Documents illustrant la pratique professionnelle** *(facultatif)* | | **p.** |  |
| **Annexes** *(Si le RC le prévoit)* | | **p.** |  |

**Exemples de pratique**

**professionnelle**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Activité-type** | **1** | **Développer la partie front-end d’une application web ou web mobile sécurisée** | | | |
|  | ***Exemple n°1*** ► | | *CP 3 : Réaliser des interfaces utilisateur statiques web ou web mobile.* | | | |
|  |  | | | | | |
|  |  | | | | | |
|  | **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  |  | | | | | |
|  | En fin d’enseignement du module 3 de la formation “Développeur web et web mobile” du GRETA 92, nous avons eu une évaluation de nos connaissances. Cette évaluation a consisté à rédiger le code HTML et CSS des pages suivantes (c’est moi qui ai ajouté sur l’image les balises en rouge et jaune) :  Accueil :    Expériences :    J’ai rédigé les pages “accueil.html”, “experiences.html”, “accueil.css”, “experiences.css”.  Arborescence du projet :  ./**accueil.html**  ./**experiences.html**  ./assets/images/**\*.jpg**  ./assets/styles/**accueil.css**  ./assets/styles/**experiences.css**  Les fichiers HTML donnent la structure des pages. Les fichiers CSS donnent l’apparence des pages.  Les balises <header>, <main>, <section>, <aside>, <footer> sont utilisées pour structurer le code des pages; elles n’agissent pas directement sur l’affichage des pages. Elles sont également utilésé dans le Search Engine Optimization (SEO) : optimisation pour les moteurs de recherche afin qu’ils “comprennent” mieux à quoi correspond une page, quel est son sujet, et la placent en bonne position dans les résultats d’une recherche. Les outils d’édition de code comme VSCode peuvent aussi les utiliser pour se repérer dans la structure.  Chaque fichier CSS commence par quelques directives CSS pour réduire les différences de rendu entre les navigateurs en supprimant les marges, les paddings et autres styles par défaut :   |  |  | | --- | --- | | /\* Reset \*/  h1 , p , figure , ul , li {  /\* Pas de marge externe pour chaque élément. \*/  margin: 0;  /\* Pas de marge interne pour chaque élément. \*/  padding: 0;  }  body {  background-color:darkblue; | /\* Affichage flexible, en colonne, centré verticalement. \*/  display: flex; flex-direction: column;  align-items: center;  font-family: sans-serif;  }  ul , li {  /\* Pas de style par défaut pour les listes. \*/  list-style: none;  }  /\* Fin reset \*/ |   Les fichiers HTML commencent tous ainsi :  <!DOCTYPE html>  <html lang="fr-FR">  <head>  <meta charset="UTF-8">  <!-- À propos de la fenêtre d’affichage : la largeur de la page doit être celle de l’écran de l’appareil avec un niveau de zoom de 100%, c’est-à-dire que la page remplit l’écran. -->  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <!-- Pour cette page, utiliser les styles CSS définis dans le fichier ‘accueil.css’. -->  <!-- Dans la page ‘expériences.html’, c’est ‘expériences.css’. Dans la page ‘formulaireDeContact.html’, c’est ‘formulaireDeContact.css’. -->  <link rel="stylesheet" href="./assets/styles/accueil.css">  <title>Les expériences de pensée d'Albert Einstein - Accueil</title>  </head>  <body>  <!-- Barre de navigation. -->  <nav>  <!-- Un clique sur le logo renvoie à la page d’accueil. -->  <a href="./accueil.html">  <img src="./assets/images/logo1.png" alt="Vers la page d'accueil." width="50">  </a>  <div>  <!-- Un clique sur ‘Accueil’ de la barre de navigation dirige à la page d’accueil. -->  <a href="./accueil.html">Accueil</a>  <!-- Un clique sur ‘Expérience’ de la barre de navigation dirige vers la page Expériences. -->  <a href="./experiences.html">Expérience</a>  <!-- Un clique sur ‘Contact’ de la barre de navigation dirige vers la page Formulaire de contact. -->  <a href="./formulaireDeContact.html">Contact</a>  </div>  </nav>  Styles appliqués à la barre de navigation :  nav {  background-color: white;  /\* Affichage flexible des éléments enfants de la balise <nav>, centré verticalement. \*/  display: flex;  align-items: center;  /\* Hateur de 60 pixels. \*/  height: 60px;  /\* Largeur de 70% de la largeur de la fenêtre. \*/  width: 70vw;  }  /\* Styles pour les <div> de premier niveau se trouvant dans la balise <nav>. \*/  nav > div {  /\* Prendre toute la largeur de la balise <nav>. \*/  width: 100%;  /\* Affichage flexible des enfants des balises <div> (de premier niveau se trouvant dans la balise <nav>). \*/  display: flex;  /\* Répartir les enfants des balises <div> régulièrement sur une ligne … \*/  justify-content: space-around;  /\* et les centrer verticalement. \*/  align-items: center;  }  nav > div > a {  /\* Les balises <a> filles de premier niveau des balises <div>, elles-mêmes enfants de premier niveau de la balise <nav> ne doivent pas être soulignées, comme c’est le cas habituel. \*/  text-decoration: none;  }  Les fichiers HTML se terminent tous par ce footer :  <footer>  <a href="https://x.com/?lang=fr">  <img src="./assets/images/logo-twitter-blanc.png" alt="" width="30">  </a>  <a href="https://www.facebook.com/">  <img src="./assets/images/FaceBook.png" alt="" width="30">  </a>  <a href="https://www.instagram.com/">  <span>in</span>  </a>  </footer>  Page “accueil.html” : elle est essentiellement composée des balises <header> et <main>.  <header>  <section id="header\_section\_1">  <h1>Einstein</h1>  <p>  <span>Albert Einstein</span>, né le …  </p>  </section>  <section id="header\_section\_2">  <img src="./assets/images/einstein.jpg" alt="Albert Einstein" width="180">  </section>  </header>  <main>  <section>  <div id="div\_photos\_1">  <img src="./assets/images/Einstein\_enfant.jpg" alt="Albert Einstein enfant">  <img src="./assets/images/Einstein\_jeune\_homme.jpg" alt="Albert Einstein jeune homme">  </div>  <div id="div\_photos\_2">  <img src="./assets/images/Einstein\_adulte.jpg" alt="Albert Einstein adulte">  <img src="./assets/images/Einstein\_groupe.jpg" alt="Albert Einstein et collègues">  </div>  </section>  <aside>  <h3>Biographie</h3>  <img src="./assets/images/Albert\_Einstein\_1947.jpg" alt="Albert Einstein 1947" width="100">  <p> Son père, Hermann Einstein ... </p>  <p> Les Einstein sont ... </p>  <p> L’intérêt d’Albert ... </p>  <p> Max Talmey, ... </p>  </aside>  </main>   |  |  | | --- | --- | | main {      background-color: white;      color: black;      width: 70vw;      height: 800px;      display: flex;      justify-content: center;      padding-top: 60px;  }  main > section {      display: flex;      flex-direction: column;      align-items: center;      padding-left: 60px;  } | main > section > div > img {      margin: 5px;  }  #div\_photos\_1 {      width: 100%;      display: flex;      justify-content: center;  }  #div\_photos\_2 {      width: 100%;      display: flex;      flex-direction: column;      align-items: space-between;  } | | | | | | |
|  |  | | | | | |
|  | Page “Expériences” :  Lorsque le pointeur de la souris passe sur une image ou intitulé d'une expérience, doit apparaître une fenêtre pop-up décrivant en détail l'expérience correspondante.      <header>           <img src="./assets/images/depuis-quand-mesure-t\_on-le-temps-bleu.jpg" alt="">      </header>        <main>          <section class="titreMain">              <h1>Les étranges expériences de pensée<br>d'Albert Einstein</h1>          </section>          <section class="galerie">              <article>                  <h3>Einstein</h3>                  <img src="./assets/images/einstein.jpeg" alt="Einstein">                  <div class="div\_explication">                      <p class="titre\_article">Einstein</p>                      <div class="div\_explication\_texte">                          En 1927, ...                      </div>                  </div>              </article>              <article> … </article>  <article> … </article>  <article> … </article>  <article> … </article>  <article> … </article>          </section>      </main>   |  |  | | --- | --- | | .galerie {      display: flex;      justify-content: center;      flex-wrap: wrap;      width: 1040px;      margin-bottom: 150px;  }  .galerie > article {      color:rgb(43, 153, 226);      margin: 5px;      position: relative;  }  .div\_explication {      background-color:#4a5197;      color:white;      border: ridge 10px #4a5197; | opacity: 0; /\* Le div contenant les explications doit être transparent. \*/      position: absolute;      top: 5px;      left: -25px;        width: 330px;      height: 180px;      font-size: 1em;      overflow-y: scroll;      overflow-wrap:break-word;  }  .div\_explication:hover {      opacity: 1; /\* Si le curseur de la souris passe sur le div contenant les explications, il devient visible, est mis au premier plan, et est un peu agrandi. \*/      transition: opacity .3s ease-in-out;      transform: scale(1.3);      z-index: 1;  } | |  |  | | | | | | |
|  |  | | | | |
|  |  | | | | | |
|  | **2. Précisez les moyens utilisés :** | | | | | |
|  | VSCode.  Google Chrome.  Debugger de Google Chrome. | | | | | |
|  |  | | | | | |
|  |  | | | | | |
|  |  | | | | | |
|  |  | | | | | |
|  | **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | Comme il s’agissait d’un TP d’évaluation à rendre, j’ai travaillé seul. | | | | | |
|  |  | | | | | |
|  |  | | | | | |
|  |  | | | | |
|  |  | | | | | |
|  | **4. Contexte** | | | | | |
|  |  | | | |  | |
|  | Nom de l’entreprise, organisme ou association► | | | | *Greta 92* | |
|  | Chantier, atelier, service► | | | *Formation DWWM – Module 3* | | |
|  | Période d’exercice► Du : *28 novembre 2024*au :*13 décembre 2024* | | | | | |
|  |  | | | | |
|  |  | | | | |
|  | **5. Informations complémentaires** *(facultatif)* | | | | |
|  |  | | | | | |
|  | Dépôt GitHub :  https://github.com/Georges-github/TP-Module-3-HTML-CSS | | | | | |
|  |  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **1** | **Développer la partie front-end d’une application web ou web mobile sécurisée** | | | |
| ***Exemple n°2*** ► | | *CP 4 : Développer la partie dynamique des interfaces utilisateur web ou web mobile.*  *ReactJS.* | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Dans le cadre de l’enseignement du module 4 de la formation “Développeur web et web mobile” du GRETA 92, nous avons réalisé un exercice pratique avec la bibliothèque ReactJS consistant à se connecter à l’api REST “<https://restcountries.com/v2>”, y récupérer des informations sur l’ensemble des pays, et afficher un nombre variable de drapeaux de chaque continent.    Lorsque la page est chargée, l’API REST ‘restcountries.com’ est sollicitée. Elle retourne les informations suivantes pour chaque pays : nom, population, continent, Capitale, drapeau. Les drapeaux sont affichés l’un à la suite de l’autre. Lorsque le curseur de la souris est placé sur un drapeau, une info bulle indique le nom du pays et sa Capitale. Si la check-box associée à un nom de continent est cochée, seuls les drapeaux des pays de ce continent sont affichés. Le curseur horizontal à gauche des check-box permet de spécifier le nombre de drapeaux à afficher.  Pour obtenir ce fonctionnement, deux composants principaux ont été développés, <Pays /> et <Card />. Les hook ‘useEffect’ et ‘useState’ sont utilisés.  Arborescence et fichiers du projet :  **node\_modules/**les modules composant ReactJS  **public/**  public/**img/**  **src/**  src/**assets/**  **src/components/**  src/components/**Logo.jsx**  src/components/**Navigation.jsx**  src/components/**Pays.jsx**  src/components/**Card.jsx**  **src/pages/**  src/pages/**Home.jsx**  src/pages/**About.jsx**  src/pages/**NotFound.jsx**  src/style/components/**\_logo.scss** , **\_navigation.scss** , **\_pays.scss** , **\_card.scss**  src/style/pages/**\_settings.scss** , **\_NotFound.scss**  **App.jsx**  **main.jsx**  **index.html**  Voyons de plus près les composants <Pays /> et <Card />. Les explications sont dans les commentaires verts.  **Pays.jsx :**  // Importation des Hook ‘useEffect’ et ‘useState’.  import { useEffect , useState } from "react";  // Importation du composant <Card />. Il est utilisé pour afficher les images des drapeaux renvoyé  import Card from "./Card";  // axios sera le nom utilisé pour utiliser la bibliothèque cliente HTTP, axios. Elle permet d’effectuer des requêtes HTTP depuis le navigateur.  import axios from "axios";  // Déclaration et définition du composant <Pays />. On peut lui spécifier des propriétés qui seront accessibles au code du composant via l’argument ‘props’.  const Pays = ( props ) => {      // useState() permet d’indiquer à la librairie ReactJS qu’à chaque mise à jour de ‘data’ via ‘setData()’, le composant ‘Pays’ sera redessiné. Même logique pour ‘sortData’, ‘selectRadio’ et ‘rangeValue’, chacun via sa fonction de mise à jour : setSortData(), …      const [ data , setData ] = useState( [] );      const [ sortData , setSortData ] = useState( [] );      const [ selectRadio , setSelectRadio ] = useState( "Tous" );      const [ rangeValue , setRangeValue ] = useState( 35 );      const radio = [ "Africa" , "America" , "Asia" , "Europe" , "Oceania" , "Tous" ];      // Charger (une seule fois, au montage du composant) via une requête HTTP axios les données de tous les pays (all) proposées par l'API REST “<https://restcountries.com/v2>” : nom, population, region, capitale.      useEffect( () => {          axios.get( "https://restcountries.com/v2/all?fields=name,population,region,capital,flag" )              .then( ( res ) => setData( res.data ) ); // Une fois les données reçues de l’API REST, mettre à jour ‘data’ avec ces données.      } , [] ); // C’est le tableau vide spécifié en second argument de ‘useEffect()’ qui spécifie que la fonction fléchée sera exécutée une seule fois, au montage du composant <Pays />.      useEffect( () => {  // Trier les pays du plus peuplé au moins peuplé.          if ( data.length ) {              const sortedArray = [...data].sort( ( a , b ) => b.population - a.population );  // La syntaxe ‘...data’ indique que c’est une copie de data qui est triée. Elle est désignée ‘sortedData’.              sortedArray.length = rangeValue;  // La liste résultat du trie est tronquée à ‘rangeValue’ pays.              setSortData(sortedArray) ;  // Mise à jour de ‘data’ à la valeur de ‘sortedArray’.          }      }, [ data , rangeValue ] );  // Chaque fois que ‘data’ et ‘rangeValue’ sont mises à jour via leur fonction ‘set…()’, la fonction fléchée, premier argument de ‘useEffect()’, est exécutée.       // Le corps html du composant <Pays />.      return (          <div className="pays">                <div className="sort\_container">                  <input type="range" min="1" max="250" value={rangeValue} onChange={ ( e ) => setRangeValue( e.target.value ) }></input>                  <ul>  // Parcours du tableau des noms des boutons radios (continents) via la méthode ‘map()’ de Array : pour chaque nom, la fonction fléchée renvoie un élément <li> contenant un bouton radio identifié par ce nom (id). Lorsque ce bouton radio est coché, la fonction fléchée associée à ‘onChange’ met à jour ‘selectRadio’ à la valeur de ce nom, cette mise à jour provoque le re-dessin du composant; le code JS situé dans la balise <div className=’pays\_liste’> (voir plus loin) est alors exécuté.                      { radio.map( (r) => {                              return(                                  <li key={r}>                                      <input type="radio"                                      id={ r }                                      value={ r }                                      checked={ r === selectRadio }                                      onChange={ ( e ) =>  setSelectRadio( e.target.value ) }/>                                      <label htmlFor={ r }>{ r }</label>                                  </li>                              )                         }                      ) }                  </ul>              </div>              <div className="pays\_liste">                  // Le code suivant est exécuté à chaque re-dessin du composant. ‘sortData’ y est filtrée, c’est-à-dire que seront retenus de la liste des pays seulement ceux qui dont l’attribut ‘région’ égale à valeur de selectRadio. La liste résultat est parcourue par la méthode ‘map()’ de Array pour générer un élément <Card /> auquel les attributs ‘country’ et ‘[country.name](http://country.name)’ sont transmis.                  {                      sortData.filter( ( c ) => c.region.includes( selectRadio === "Tous" ? "" : selectRadio ) )                      .map( ( country ) => <Card country={ country } key={ country.name } /> )                  }              </div>          </div>      );  };  // Donner la possibilité au code externe de connaître la constante ‘Pays’.  export default Pays;  **Card.jsx :**  // Ce composant est simple. Il reçoit en paramètre un élément de la liste des pays.  const Card = ( props ) => {      // Récupération de l’élément de la liste des pays passé en argument.      const {country} = props;      return (          // Corps html du composant <Card />.          <ul>              <li className="card">                  <img src={country.flag} alt="flag" title={country.name + " / " + country.capital}/>              </li>          </ul>      );  };  export default Card; | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| VSCode.  ReactJS. | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Pour cet exemple, j’ai travaillé avec notre formateur Christian Delorme qui nous a enseigné Figma, HTML, CSS, Bootstrap, Javascript, ReactJS.  Il s’agissait d’un exercice pratique. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association► | | | | *Greta 92* | |
| Chantier, atelier, service► | | | *Formation DWWM – Module 4* | | |
| Période d’exercice► Du : *16 décembre 2024*au :*10 janvier 2025* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
| Dépôt GitHub :  https://github.com/Georges-github/TP-Module-4-ReactJS-Drapeaux | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Développer la partie back-end d’une application web ou web mobile sécurisée** | | | |
| ***Exemple n° 1*** ► | | *CP 6 : Développer des composants d’accès aux données SQL et NoSQL.*  *Architecture logicielle : Modèle - Vue - Contrôleur.* | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Exercice d'évaluation des connaissances enseignées durant le module 6 de la formation “Développeur web et web mobile” du Greta 92.  Il nous a été demandé de concevoir une base de données d'étudiants, de rédiger le code PHP de connexion à cette base, puis le code PHP des quatre opérations dites CRUD, c'est-à-dire "Creation" d'un étudiant, "Read" d'un étudiant, "Update" d'un étudiant, "Delete" d'un étudiant. Chacune de ces opérations donnant lieu à un affichage du résultat dans le navigateur. Par exemple, l’url “<http://localhost/Evaluation_Module_6/index.php?page=etudiants>” renvoie le résultat de l’opération de lecture dans la base de données de l’ensemble des étudiants :    L’architecture logicielle utilisée est : Modèle – Vue – Contrôleur.  La base de données a été construite via "phpMyAdmin". Voici quelques lignes de l'export SQL :  CREATE TABLE `etudiants` (  `id` int(11) NOT NULL,  `prenom` varchar(100) NOT NULL,  `nom` varchar(100) NOT NULL,  `email` varchar(100) NOT NULL,  `cv` mediumtext NOT NULL DEFAULT 'CV de l\'étdudiant',  `dt\_naissance` date DEFAULT NULL,  `isAdmin` tinyint(1) NOT NULL DEFAULT 0,  `dt\_mise\_a\_jour` datetime NOT NULL DEFAULT current\_timestamp()  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;  INSERT INTO `etudiants` (`id`, `prenom`, `nom`, `email`, `cv`, `dt\_naissance`, `isAdmin`, `dt\_mise\_a\_jour`) VALUES  (1, 'Édith', 'Piaf', 'edith.piaf1925@email.com', 'CV de l\'étudiant.', '1915-12-19', 0, '2025-03-02 20:45:31'),  (2, 'Josephine', 'Baker', 'josephine.baker1925@email.com', 'CV de l\'étudiant', '1906-06-03', 0, '2025-02-20 10:10:23'),  … ;  Arborescence du projet :  **Index.php**  controller/ **AbstractController.php**  controller/ **ErreurController.php**  controller/ **FrontController.php**  model/**BDD.php**  vue/**page\_etudiants.php**  vue/**page\_etudiant.php**  vue/**etudiant\_form.php**  vue/**page\_supprimer.ph**p  vue/**header.php**  vue/**footer.php**  vue/**page\_erreur.php**  BDD.php : code d'accès au modèle de données, c'est-à-dire code d'accès à la base de données. Les requêtes SQL sont construites dans les méthodes des classes contrôleur, essentiellement dans les méthodes de la classe 'FrontController'.  Les explications sont dans les commentaires verts.  <?php  class BDD {  // L'instance actuelle la présente classe.      private static $instance = null;  // L'instance du PHP Data Object; permet l'accès à la BD.      private $connexion = null;      private function \_\_construct() {  // La BD mysql s'appelle "module6\_GDG".          $dsn = "mysql:host=localhost;dbname=module6\_GDG;charset=utf8mb4";          $login = "root";          $password = "";          $this->connexion = new PDO( $dsn , $login , $password );      }      public static function getInstance() { … }  // Execution d'une requête SQL transmise en paramètre avec ses paramètres.      public function query( string $sql , array $params = [] ) {  // Afin d'empêcher les injections SQL (écriture de requêtes SQL dans les formulaires de saisie).          $stmt = $this->connexion->prepare( $sql );          $stmt->execute( $params );  // Si la requête SQL est un INSERT …          if ( str\_starts\_with( $sql , "INSERT INTO" ) ) {  // … retourner l'index de la dernière ligne insérée.              return $this->connexion->lastInsertId();  // Si la requête SQL est un UPDATE …          } else if ( str\_starts\_with( $sql , "UPDATE" ) ) {  // … retourner le nombre de ligne modifiées.              return $stmt->rowCount();          }          // Si la requête SQL est un SELECT, retourner le résultat sous la forme d'un tableau de tableaux associatifs. Chaque tableau associatif représentant une ligne de la table interrogée.          return $stmt->fetchAll( PDO::FETCH\_ASSOC );      }    }  AbstractController.php : code de construction de la page à afficher dans le navigateur.  <?php  abstract class AbstractController {  // La méthode render() va construire par inclusions de fichiers PHP la page à afficher.      protected function render( string $template , array $data = [] ) { // $data est un tableau associative.          extract( $data ); // Faire de chaque clef du tableau associatif $data une variable contenant la valeur de la clef.          require\_once( "vue/header.php" ); // Inclure le contenu du fichier vue/header.phph.          require\_once( "vue/$template.php" ); // Inclure le contenu du fichier vue/'valeur du paramètre $template'.php.          require\_once( "vue/footer.php" ); // Inclure le contenu du fichier vue/footer.php.      }  }  index.php : code orientant vers la page associée à une url. C'est le point d'entrée de la partie Frontend de l'application; c'est aussi le "routeur", c’est-à-dire qu’il va orienter vers la méthode adéquate de la classe du contrôleur adéquat.  <?php  session\_start();  // define( "URL\_SITE" , "http://192.168.33.10/Evaluation\_Module\_6/" ); /// Pour VM Vagrant.  define( "URL\_SITE" , "http://localhost/Evaluation\_Module\_6/" );  $page = ( isset($\_GET[ "page" ] ) && $\_GET["page"] !== '' ) ? $\_GET[ "page" ] : "/";  // "url" => [ "nom d'une méthode" , "nom du contrôleur dans lequel cette méthode est définie" ]  $routes = [      "/"                 => [ "etudiants" , "FrontController" ] ,      "etudiants"         => [ "etudiants" , "FrontController" ] ,      "etudiant"          => [ "etudiant"  , "FrontController" ] ,      "supprimer"         => [ "supprimer" , "FrontController" ] ,      "nouveau"           => [ "nouveau"   , "FrontController" ] ,      "modifier"          => [ "modifier"  , "FrontController" ]  ];  // Inclure le code d'accès au modèle.  require\_once( "model/BDD.php" );  // Inclure les contrôleurs dans lesquels sont définies les méthodes associées aux url.  require\_once( "controller/AbstractController.php" );  require\_once( "controller/FrontController.php" );  require\_once( "controller/ErreurController.php" );  // Si l'url fait partie des url prises en compte, appeler la méthode correspondant à cette url.  if ( array\_key\_exists( $page , $routes ) ) {      $methode = $routes[ $page ][ 0 ];      $classe = $routes[ $page ][ 1 ];  // Instancier la classe contenant la méthode sollicitée.      $c = new $classe();  // Appeler la méthode sollicitée.      call\_user\_func( [ $c , $methode ] );  } else { // Si l'url ne fait pas partie des url prises en compte, appeler la méthode 'erreur()' de 'ErreurController'.      $c = new ErreurController();      $c->erreur( 404 , "La page demandée n'existe pas." );  }  La classe 'FrontController' contient de nombreuses méthodes publiques. Chacune effectue le traitement associé à une url. **Ce sont des méthodes de la partie Contrôleur du MVC, c’est-à-dire qu'elles mettent en forme les données extraites de la partie Modèle du MVC, pour les transmettre à la partie Vue du MVC.** Il est important de noter que 'FrontController' étend 'AbstractController'.  J'explique d'abord le fonctionnement de la méthode 'etudiants()' qui est la plus simple. Elle transmet à la Vue la liste des étudiants :  public function etudiants() {      $data = [          "titre" => "Etudiants" ,          "etudiants" => BDD::getInstance()->query( "SELECT \* FROM etudiants" )      ];      $this->render( "page\_etudiants" , $data );  }  L'url '**http://127.0.0.1/Evaluation\_Module\_6/**' ayant été écrite dans la barre d'adresse du navigateur, le code du fichier 'index.php' est exécuté par le serveur Apache. L'exécution de ce code aboutit à la sélection de la méthode 'etudiants()' de la classe 'FrontController', puis à l'instanciation de cette classe en vue d’un appel de la méthode à partir de cette instance. Le code de la méthode consiste à construire les données '$data' transmises à la méthode 'render()' héritée de la classe 'AbstractController' : la clef 'etudiants' contient le résultat de la requête SQL 'SELECT \* FROM etudiants' effectuée via la méthode 'query()' de la classe BDD.  Est également transmis à la méthode, le nom de la page à afficher, 'page\_étudiants'(.php) .  Pour l'url '**http://127.0.0.1/Evaluation\_Module\_6/page=etudiant&id=1**', ce sera la méthode 'etudiant()' de l'instance de 'FrontController()' qui sera appelée. Cette méthode récupère dans la variable super globale '$\_GET' la valeur de 'id' de l'étudiant et c'est la requête SQL suivante qui est exécutée : $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE id = :id ", [ "id" => $id ] ); . Les données transmises à la page 'étudiant'(.php) via l'appel à la méthode 'render()', sont :          $data = [              "titre" => "Etudiant",              "etudiant" => ( count( $erreurs ) === 0 ? $etudiant : [] ) ,              "erreurs" => $erreurs          ];          $this->render( "page\_etudiant" , $data ) ;  Le code : … WHERE id = :id ", [ "id" => $id ] … est équivalent à : … WHERE id = $id " . C’est-à-dire que si $id vaut 1 alors le code sera : … WHERE id = `1` " .  Pour l'url '**http://127.0.0.1/Evaluation\_Module\_6/page=supprimer&id=1**', ce sera la méthode 'supprimer()' de l'instance de 'FrontController()' qui sera appelée. Cette méthode récupère dans la variable super globale '$\_GET' la valeur de 'id' de l'étudiant et c'est la requête SQL suivante qui est exécutée : BDD::getInstance()->query( "DELETE FROM etudiants WHERE id = :id" , [ "id" => $id ] );.  Pour l'url '**http://127.0.0.1/Evaluation\_Module\_6/page=modifier&id=1**', ce sera la méthode 'modifier()' de l'instance de 'FrontController()' qui sera appelée. Cette méthode récupère dans la variable super globale '$\_GET' la valeur de 'id' de l'étudiant et, après diverses vérifications de la validité des valeurs des champ du formulaire, c'est la requête SQL suivante qui est exécutée : BDD::getInstance()->query( "UPDATE etudiants SET prenom = :prenom , nom = :nom , email = :email , cv = :cv , dt\_naissance = :dt\_naissance , isAdmin = :isAdmin , dt\_mise\_a\_jour = :dt\_mise\_a\_jour WHERE id= :id" , [                  "id" => $id ,                  "prenom" => $prenom ,                  "nom" => $nom ,                  "email" => $email ,                  "cv" => $cv ,                  "dt\_naissance" => $dt\_naissance ,                  "dt\_mise\_a\_jour" => ( new DateTime() )->format('Y-m-d H:i:s') ,                  "isAdmin" => ( $role === "admin" ? 1 : 0 )              ] );    Pour l'url '**http://127.0.0.1/Evaluation\_Module\_6/page=nouveau**', ce sera la méthode 'nouveau()' de l'instance de 'FrontController()' qui sera appelée. Là aussi, des vérifications des valeurs des champs du formulaire d'ajout d'étudiant sont réalisées, puis la requête SQL suivante est exécutée :              $id = BDD::getInstance()->query( "INSERT INTO etudiants( prenom , nom , email , cv , dt\_naissance , isAdmin ) VALUES ( :prenom , :nom , :email , :cv , :dt\_naissance , :role )" , [                  "prenom" => $prenom ,                  "nom" => $nom ,                  "email" => $email ,                  "cv" => $cv ,                  "dt\_naissance" => $dt\_naissance ,                  "role" => ( $role === "admin" ? 1 : 0 )              ] );  Contenu complet de “FrontController.php” :  <?php  require\_once( "model/BDD.php" );  class FrontController extends AbstractController {  public function etudiants() {  $data = [  "titre" => "Etudiants" ,  "etudiants" => BDD::getInstance()->query( "SELECT \* FROM etudiants" )  ];  $this->render( "page\_etudiants" , $data );  }  public function etudiant() {  $erreurs = [];  $etudiant = [];  if ( array\_key\_exists( "id" , $\_GET ) ) {  $id = $\_GET[ "id" ] ?? "";  $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE id = :id ", [ "id" => $id ] );  if ( empty( $etudiant ) ) {  $erreurs[] = "Aucun étudiant ne correspond à l'id indiqué.";  }  }    $data = [  "titre" => "Etudiant",  "etudiant" => ( count( $erreurs ) === 0 ? $etudiant : [] ) ,  "erreurs" => $erreurs  ];  $this->render( "page\_etudiant" , $data ) ;  }  public function supprimer() {  $erreurs = [];  $etudiant = [];  if ( array\_key\_exists( "id" , $\_GET ) ) {  $id = $\_GET[ "id" ] ?? "";  $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE id = :id ", [ "id" => $id ] );  if ( empty( $etudiant ) ) {  $data = [  "titre" => "Supprimer" ,  "contenu" => [  "num" => 404 ,  "message" => "Aucun étudiant ne correspond à l'id indiqué."  ]  ];  $this->render( "page\_erreur" , $data );  die();  }  }  $data = [  "titre" => "Supprimer",  "etudiant" => $etudiant ,  "erreurs" => $erreurs  ];  if ( count( $erreurs ) === 0 ) {  BDD::getInstance()->query( "DELETE FROM etudiants WHERE id = :id" , [ "id" => $id ] );  }  $this->render( "page\_supprimer" , $data );  }  public function nouveau() {  $id = "";  $prenom = "";  $nom = "";  $email = "";  $cv = "";  $dt\_naissance = "";  $role = "";  $erreurs = [];  if ( ! empty( $\_POST ) ) {  $prenom = isset( $\_POST[ "prenom" ] ) ? $\_POST[ "prenom" ] : $prenom;  $nom = isset( $\_POST[ "nom" ] ) ? $\_POST[ "nom" ] : $nom;  $email = isset( $\_POST[ "email" ] ) ? $\_POST[ "email" ] : $email;  $cv = isset( $\_POST[ "cv" ] ) ? $\_POST[ "cv" ] : $cv;  $dt\_naissance = isset( $\_POST[ "dt\_naissance" ] ) ? $\_POST[ "dt\_naissance" ] : $dt\_naissance;  $role = isset( $\_POST[ "role" ] ) ? $\_POST[ "role" ] : $role;  if ( empty( $prenom ) ) {  $erreurs[] = "Vous devez spécifier un prénom.";  }  if ( empty( $nom ) ) {  $erreurs[] = "Vous devez spécifier un nom.";  }  if ( ! filter\_var( $email , FILTER\_VALIDATE\_EMAIL ) ) {  $erreurs[] = "L'adresse courriel n'est pas conforme.";  }    if ( preg\_match( '/^\d{4}-\d{2}-\d{2}$/' , $dt\_naissance ) ) {  list( $year , $month , $day ) = explode( '-' , $dt\_naissance );  if ( ! checkdate( $month , $day , $year ) ) {  $erreurs[] = "Date de naissance invalide.";  }  } else {  $erreurs[] = "Format de date de naissance invalide.";  }  $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE email = :email" , [ "email" => $email ] );  if ( ! empty( $etudiant ) ) {  $erreurs[] = "Cette adresse courriel est déjà utilisée.";  }  $roles = [ "etudiant" , "admin" ];  if ( ! in\_array( $role , $roles ) ) {  $erreurs[] = "Sélectionnez un rôle";  }  }  if ( count( $erreurs ) === 0 && ! empty( $\_POST ) ) {  $id = BDD::getInstance()->query( "INSERT INTO etudiants( prenom , nom , email , cv , dt\_naissance , isAdmin ) VALUES ( :prenom , :nom , :email , :cv , :dt\_naissance , :role )" , [  "prenom" => $prenom ,  "nom" => $nom ,  "email" => $email ,  "cv" => $cv ,  "dt\_naissance" => $dt\_naissance ,  "role" => ( $role === "admin" ? 1 : 0 )  ] );  header( "Location: " . URL\_SITE . "index.php?page=etudiants" );  die();  }  $data = [  "titre" => "Ajout" ,  "data\_form" => [  "id" => $id ,  "prenom" => $prenom ,  "nom" => $nom ,  "email" => $email,  "cv" => $cv,  "dt\_naissance" => $dt\_naissance ,  "role" => $role  ] ,  "erreurs" => $erreurs  ];  $this->render( "etudiant\_form" , $data );  }  public function modifier() {  $erreurs = [];  $etudiant = [];    $id = null;  if ( array\_key\_exists( "id" , $\_GET ) ) {  $id = $\_GET[ "id" ];  }  $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE id = :id" , [ "id" => $id ] );    if ( empty( $etudiant ) ) {  $data = [  "titre" => "Modifier" ,  "contenu" => [  "num" => 404 ,  "message" => "L'étudiant que vous souhaitez modifier n'est pas dans la base de données."  ]  ];    $this->render( "page\_erreur" , $data );  die();  }  $prenom = $etudiant[ 0 ][ "prenom" ];  $nom = $etudiant[ 0 ][ "nom" ];  $email = $etudiant[ 0 ][ "email" ];  $cv = $etudiant[ 0 ][ "cv" ];  $dt\_naissance = $etudiant[ 0 ][ "dt\_naissance" ];  $role = ( $etudiant[ 0 ][ "isAdmin" ] === 1 ? "admin" : "etudiant" );  if ( ! empty( $\_POST ) ) {  $prenom = isset( $\_POST[ "prenom" ] ) ? $\_POST[ "prenom" ] : $prenom;  $nom = isset( $\_POST[ "nom" ] ) ? $\_POST[ "nom" ] : $nom;  $email = isset( $\_POST[ "email" ] ) ? $\_POST[ "email" ] : $email;  $cv = isset( $\_POST[ "cv" ] ) ? $\_POST[ "cv" ] : $cv;  $dt\_naissance = isset( $\_POST[ "dt\_naissance" ] ) ? $\_POST[ "dt\_naissance" ] : $dt\_naissance;  $role = isset( $\_POST[ "role" ] ) ? $\_POST[ "role" ] : $role;  if ( empty( $prenom ) ) {  $erreurs[] = "Vous devez spécifier un prénom.";  }  if ( empty( $nom ) ) {  $erreurs[] = "Vous devez spécifier un nom.";  }  if ( ! filter\_var( $email , FILTER\_VALIDATE\_EMAIL ) ) {  $erreurs[] = "L'adresse courriel n'est pas conforme.";  }    if ( preg\_match( '/^\d{4}-\d{2}-\d{2}$/' , $dt\_naissance ) ) {  list( $year , $month , $day ) = explode( '-' , $dt\_naissance );  if ( ! checkdate( $month , $day , $year ) ) {  $erreurs[] = "Date de naissance invalide.";  }  } else {  $erreurs[] = "Format de date de naissance invalide.";  }  $etudiant = BDD::getInstance()->query( "SELECT \* FROM etudiants WHERE email = :email AND id != :id" , [ "email" => $email , "id" => $id ] );  if ( ! empty( $etudiant ) ) {  $erreurs[] = "Cette adresse courriel est déjà utilisée.";  }  $roles = [ "etudiant" , "admin" ];  if ( ! in\_array( $role , $roles ) ) {  $erreurs[] = "Sélectionnez un rôle";  }  }  if ( count( $erreurs ) === 0 && ! empty( $\_POST ) ) {  $ret = BDD::getInstance()->query( "UPDATE etudiants SET prenom = :prenom , nom = :nom , email = :email , cv = :cv , dt\_naissance = :dt\_naissance , isAdmin = :isAdmin , dt\_mise\_a\_jour = :dt\_mise\_a\_jour WHERE id= :id" , [  "id" => $id ,  "prenom" => $prenom ,  "nom" => $nom ,  "email" => $email ,  "cv" => $cv ,  "dt\_naissance" => $dt\_naissance ,  "dt\_mise\_a\_jour" => ( new DateTime() )->format('Y-m-d H:i:s') ,  "isAdmin" => ( $role === "admin" ? 1 : 0 )  ] );  if ( $ret === 1 || $ret === 0 ) {  header( "Location: " . URL\_SITE . "index.php?page=etudiants" );  die();  } else {  $erreurs[] = "Problème de mise à jour de la BD.";  }  }  $data = [  "titre" => "Modifier" ,  "data\_form" => [  "id" => $id ,  "prenom" => $prenom ,  "nom" => $nom ,  "email" => $email ,  "cv" => $cv ,  "dt\_naissance" => $dt\_naissance ,  "role" => $role  ] ,  "erreurs" => $erreurs  ];  $this->render( "etudiant\_form" , $data );  }  } | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| VSCode.  MySQL, PhpMyAdmin, PDO. | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Comme il s’agissait d’un TP d’évaluation à rendre, j’ai travaillé seul, en m'aidant du cours de notre formateur Malik Harriz. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association► | | | | *Greta 92* | |
| Chantier, atelier, service► | | | *Formation DWWM – Module 6* | | |
| Période d’exercice► Du : *27 janvier 2025*au :*10 février 2025* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
| Dépôt GitHub :  https://github.com/Georges-github/Evaluation\_Module\_6 | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Développer la partie back-end d’une application web ou web mobile sécurisée** | | | |
| ***Exemple n° 2*** ► | | *CP 5 : Mettre en place une base de données relationnelle.*  *Modèle Conceptuel des Données - Modèle Logique des Données - Modèle Physique des Données.* | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Une partie de l’évaluation du module 5 a consisté à établir le MCD, le MLD et le MPD de la problématique suivante :  On veut informatiser la gestion des séjours des patients d’un hôpital. Chaque patient possède un numéro d’identification, un nom, un prénom, une adresse, et le nom de sa caisse d’assurance maladie.  Le patient effectue un ou plusieurs séjours à l’hôpital. On attribue à chaque séjour un numéro.  Un séjour possède une date d’arrivée et une date de départ, et s’effectue dans une chambre  (que l’on supposera être la même tout au long du séjour). Une chambre a un numéro, un étage,  et le nom du service correspondant.  Enfin, lors d’un séjour peuvent être pratiquées une ou plusieurs interventions. Chaque opération  a un code, une date, un nom, et est dirigée par un médecin. Chaque médecin a un code, un  nom, un prénom et une spécialité.  MCD : Modèle Conceptuel des Données.    Le MCD est une **représentation abstraite et indépendante du système informatique** des données d’un domaine. Il sert à décrire de manière structurée les **entités**, leurs **attributs**, et les **relations** entre ces entités.  Si on adopte la convention MERISE, comme c’est le cas sur le schéma, les cardinalités portées par les extrémités des relations sont à lire ainsi : par exemple, la relation entre patient et séjour dans le MCD affiché plus bas, se lit : un patient effectue un ou plusieurs séjours à l’hôpital; un séjour concerne un et un seul patient.  Un séjour est effectué dans une et une seule chambre; dans une chambre peuvent être effectués plusieurs séjours (par des patients différents).  Durant un séjour, plusieurs opérations peuvent être effectuées; une opération est effectuée durant un seul séjour.  Une opération est dirigée par un seul médecin; un médecin dirige zéro ou n opérations.  (Si on adopte la convention UML, alors “1 , 1” doit être écrit “1 .. 1”, et doit se trouver à proximité de patient. La cardinalité “1 , n” doit être écrite “1 .. n” et doit se trouver à proximité de séjour.)  Les entités donneront les tables du MPD, de la base de données.  Les attributs des entités donneront les noms et types des colonnes de ces tables.  Dans le cas d’une relation “n , n” entre deux entités, cette relation est représentée en base de données par une table associative dont chaque ligne contient une paire de clefs étrangères; leurs valeurs seront celles des clefs primaires des entités qu’elle relie.  MLD : Modèle Logique des Données.    C’est la **traduction du MCD** dans un format plus **proche de la structure d’une base de données relationnelle**, mais **encore indépendant du système de gestion de base de données**.  Le point important est qu’à partir de maintenant le système de données étudié est représenté selon le modèle relationnel.  Il s’agit ici de représenter les entités sous forme de tables. C’est dans ce modèle que les clefs primaires et étrangères font leur apparition; elles représentent dans la base de données relationnelle, les relations entre les entités du MCD. Les clefs primaires servent avant tout à identifier chaque instance d’une entité. Ce sont en fait les clefs étrangères qui, renvoyant aux clefs primaires, représentent les relations entre entités.  MPD : Modèle Physique des Données.  Le MPD est une représentation concrète de la structure d’une base de données adaptée au Système de Gestion de Base de Données choisi. Il est constitué des commandes SQL définissant les tables, leurs atrributs et les relations entre tables.  Le type des attributs est indiqué avec précision.  DROP TABLE IF EXISTS patient;  CREATE TABLE IF NOT EXISTS patient (  numero\_patient INTEGER PRIMARY KEY AUTOINCREMENT ,  prenom VARCHAR(255) NOT NULL ,  nom VARCHAR(255) NOT NULL ,  assurance VARCHAR(255) NOT NULL  );  DROP TABLE IF EXISTS medecin;  CREATE TABLE IF NOT EXISTS medecin (  code\_medecin INTEGER PRIMARY KEY AUTOINCREMENT ,  prenom VARCHAR(255) NOT NULL ,  nom VARCHAR(255) NOT NULL ,  specialite VARCHAR(255) NOT NULL  );  DROP TABLE IF EXISTS chambre;  CREATE TABLE IF NOT EXISTS chambre (  numero\_chambre INTEGER PRIMARY KEY AUTOINCREMENT ,  etage INTEGER NOT NULL,  service VARCHAR(255) NOT NULL  );  DROP TABLE IF EXISTS sejour;  CREATE TABLE IF NOT EXISTS sejour (  numero\_sejour INTEGER PRIMARY KEY AUTOINCREMENT ,  date\_arrivee DATE DEFAULT( CURRENT\_DATE ) ,  date\_depart DATE DEFAULT( CURRENT\_DATE ) ,  numero\_patient INTEGER NOT NULL ,  numero\_chambre INTEGER NOT NULL ,  FOREIGN KEY( numero\_patient ) REFERENCES patient( numero\_patient ) ,  FOREIGN KEY( numero\_chambre ) REFERENCES chambre( numero\_chambre )  );  DROP TABLE IF EXISTS operation;  CREATE TABLE IF NOT EXISTS operation (  nom VARCHAR(255) NOT NULL ,  date DATE NOT NULL ,  code\_medecin INTEGER NOT NULL ,  numero\_sejour INTEGER NOT NULL ,  FOREIGN KEY( code\_medecin ) REFERENCES medecin( code\_medecin ) ,  FOREIGN KEY( numero\_sejour ) REFERENCES sejour( numero\_sejour )  ); | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| VSCode.  SQLite. | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| J’ai travaillé seul car il s’agissait d’une évaluation des connaissances enseignées durant le module 5. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association► | | | | *Greta 92* | |
| Chantier, atelier, service► | | | *Formation DWWM - Module 5* | | |
| Période d’exercice► Du : *13 janvier 2025*au :*22 janvier 2025* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |
| --- | --- | --- |
| **Titres, diplômes, CQP, attestations de formation** | | |
|  | | |
| *(facultatif)* | | |
| **Intitulé** | **Autorité ou organisme** | **Date** |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **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)* |