|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Nom de naissance* | ► | Mcheik |
| *Nom d’usage* | ► | Entrez votre nom d’usage ici. |
| *Prénom* | ► | Mahdi |
| *Adresse* | ► | 67 avenue de Paris, 17210 Chevanceaux |
|  |  |

|  |  |
| --- | --- |
| **Titre professionnel visé** | |
|  | |
| **Concepteur développeur d’applications** | |
|  | |
| **Modalité d’accès :** | |
|  | |
| ☑ | 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 une application sécurisée** | | **p.** |  |
|  | ► Intitulé de l’exemple n° 1  p. | p. |  |
|  | ► Intitulé de l’exemple n° 2  p. | p. |  |
|  | ► Intitulé de l’exemple n° 3  p | p. |  |
|  |  |  |  |
| **Concevoir et développer une application sécurisée organisée en couches** | | **p.** |  |
|  | ► Intitulé de l’exemple n° 1  Analyser les besoins et maquetter une application p. | p. |  |
|  | ► Intitulé de l’exemple n° 2  p. | p. |  |
|  | ► Intitulé de l’exemple n° 3  p | p. |  |
|  |  |  |  |
| **Préparer le déploiement d’une application sécurisée** | | **p.** |  |
|  | ► Intitulé de l’exemple n° 1  p. | p. |  |
|  | ► Intitulé de l’exemple n° 2  p. | p. |  |
|  | ► Intitulé de l’exemple n° 3  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 une application sécurisée** | | | | |
| ***Exemple n°1*** ► | | | | Installer et configurer son environnement de travail en fonction du projet | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Dans le cadre de mon travail chez Beecoming, j’ai installé les outils et logiciels utilisés pour le développement des applications web et natives principalement en dotnet et en angular  Pour les applications natives en dotnet WPF   * Visual Studio 2022 * Vscode * Windows 10 SDK   Pour la partie web   * Node Js * Angular CLI * NPM Package manager pour node * Capacitor et Ionic   Base de données   * Postgres * SQL express * PgAdmin et SSMS   Transverses   * Postman * Git/Gitkraken * Git bash * Dbeaver     Figure 1: Versions de node, NPM et VS code dans une console Bash  C:\Users\mahdi\Pictures\Screenshots\Screenshot 2025-09-23 221815.png  Figure 2 Versions 2022 et 2026 insiders de Visual studio, la version insiders est instalée pour me familiariser avec dotnet 10.0 previews | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Ordinateur Portable Dell avec CPU 10ème génération et 16 GB de mémoire RAM avec double écrans 1080 P | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai installé ces outils sous la surveillance de mon référent Etienne Bouysset, principalement pour avoir les mêmes versions utilisées par mes collègues. | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Cliquez ici pour taper du texte.** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Beecoming | | | | | |
| **Période d’exercice** | | ► | **Du** | | Aout 2024 | | **au** | Septembre 2024 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **1** | | | **Développer une application sécurisée** | | | | |
| ***Exemple n°2*** ► | | | | Développer des interfaces utilisateur | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Dans le cadre de ma formation **Concepteur Développeur d’Applications (CDA)**, j’ai réalisé un projet intitulé **Skill-hive.fr** pour un client particulier. Il s’agit d’une **plateforme d’aide scolaire** intégrant un **système de paiement** et de **facturation** automatisé.  Le développement de l’application repose sur deux technologies principales :   * **Angular** pour la partie **frontend** * **.NET** pour la partie **backend**  🏠 Page d’accueil de l’application   Figure 3 Dancs cette image, on voit le <topbar-widget /> et le <hero-widget />  La page d’accueil fait partie d’un **rendu dynamique**, dans lequel le **header** et le **footer** sont communs à plusieurs pages similaires. La structure principale du composant parent est la suivante :  **<div class="bg-surface-0 dark:bg-surface-900">**  **<div id="home" class="landing-wrapper overflow-hidden">**  **<topbar-widget**  **class="py-6 px-6 mx-0 md:mx-12 lg:mx-20 lg:px-20 flex items-center justify-between relative lg:static"**  **/>**  **<router-outlet></router-outlet>**  **<footer-widget />**  **</div>**  **</div>**  La balise **<router-outlet>** est remplacée dynamiquement par le contenu correspondant à la page active — ici, la page d’accueil. J’ai décomposé la structure HTML en plusieurs **composants Angular** afin de faciliter la **maintenance** et l’**évolution** future de l’application.  Ainsi, le code associé à la page d’accueil est simplement composé de trois composants réutilisables :  **<hero-widget />**  **<features-widget />**  **<highlights-widget />** Exemple : la bannière principale (hero-widget) Le composant de bannière met en avant l’objectif principal de la plateforme : permettre aux utilisateurs de réserver facilement des cours particuliers dans différentes matières.  **<div**  **id="hero"**  **class="flex flex-col pt-6 px-6 lg:px-20 overflow-hidden"**  **style="background: linear-gradient(0deg, rgba(255, 255, 255, 0.2), rgba(255, 255, 255, 0.2)),**  **radial-gradient(77.36% 256.97% at 77.36% 57.52%, rgb(238, 239, 175) 0%, rgb(195, 227, 250) 100%);**  **clip-path: ellipse(150% 87% at 93% 13%);"**  **>**  **<div class="mx-6 md:mx-20 mt-0 md:mt-6">**  **<h1 class="text-6xl font-bold text-gray-900 leading-tight">**  **<span class="font-light block">Vos cours à votre portée</span>**  **Réservez dès maintenant**  **</h1>**  **<p class="font-normal text-2xl leading-normal md:mt-4 text-gray-700">**  **Mathématiques, Physique et Chimie, tous niveaux...**  **</p>**  **...**  **</div>**  **<div class="flex justify-center md:justify-end">**  **<img src="assets/skillHiveSecondaryBlack.svg" alt="Hero Image" class="w-9/12 md:w-auto" />**  **</div>**  **</div>** | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Ordinateur sous windows 11  IDE : VS code et visual studio | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai développé l’application seul | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Beecoming** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Skill-Hive | | | | | |
| **Période d’exercice** | | ► | **Du** | | Octobre 2024 | | **au** | Janvier 2025 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **1** | | | **Développer une application sécurisée** | | | | |
| ***Exemple n°3*** ► | | | | *Cliquez ici pour entrer l’intitulé de l’exemple* | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Cliquez ici pour taper du texte.** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Cliquez ici pour taper du texte. | | | | | |
| **Période d’exercice** | | ► | **Du** | | Cliquez ici | | **au** | Cliquez ici |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **2** | | | **Concevoir et développer une application sécurisée organisée en couches** | | | | |
| ***Exemple n°1*** ► | | | | Analyser les besoins et maquetter une application | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Dans le cadre du projet **Inspire for the Wild**, mes collègues et moi avons préparé la **maquette Figma** de l’application, en versions **mobile** et **desktop**.**C:\Users\mahdi\Pictures\Screenshots\figma.png** | | | | | | | | |
| Cahier des charges  Dans le cadre de mon travail, j’ai également participé à **l’analyse et à l’adaptation des nouveaux besoins clients**, afin de garantir une intégration fluide des évolutions fonctionnelles de l’application. Cette expérience m’a permis de renforcer ma compréhension de l’importance de **l’ergonomie** et de **l’expérience utilisateur (UX)** dans la conception d’interfaces efficaces et intuitives.  Exemple : Opteeam projet de gestion des missions/utilisateurs  **Opteeam** est un projet générique de **gestion des missions et des opérateurs**. Il a pour objectif de **faciliter l’affectation des missions aux équipes**, ainsi que la **gestion et le suivi de ces équipes**. Dans ce contexte, le **cahier des charges** a été présenté sous forme de **questions-réponses**, afin de mieux clarifier les besoins et les attentes fonctionnelles du projet.  C:\Users\mahdi\Downloads\image.png  Figure 4 Une partie du cahier des charges | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Developpement sur figma | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| Marie Delaire et Mathieu Chauveau | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Wild code school** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Formation Poec Java / angular | | | | | |
| **Période d’exercice** | | ► | **Du** | | Avril 2024 | | **au** | Juin 2024 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **2** | | | **Concevoir et développer une application sécurisée organisée en couches** | | | | |
| ***Exemple n°2*** ► | | | | Définir l’architecture logicielle d’une application | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Dans le cadre de ma formation CDA, j’ai réalisé un projet intitulé **Skill-hive.fr** pour un client particulier. Il s’agit d’une plateforme d’aide scolaire, intégrant un système de paiement et de facturation.  J’ai développé l’application en **Angular** et **.NET**, avec **PostgreSQL** comme base de données.   * **Front-end** : réalisé en **Angular 19**, sous forme de SPA, intégrant **FullCalendar**, **PrimeNG** et **Tailwind CSS**. * **Back-end** : développé en **.NET 8.0 (version LTS)**, utilisant notamment **Hangfire**, **Identity**, **QuestPDF**, ainsi que **Stripe** pour la gestion des paiements.   Librairies installées pour l’API :  <PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="8.0.10" />  <PackageReference Include="QuestPDF" Version="2025.7.0" />  <PackageReference Include="SixLabors.ImageSharp" Version="3.1.11" />  <PackageReference Include="SkiaSharp" Version="3.119.0" />  <PackageReference Include="Stripe.net" Version="47.3.0" />  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.9.0" />  Et pour le client (Angular) :      Figure 5 : Librairies instalées: FullCalendar, SignalR et primeNg | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Cette application a été développé sous windows avec les outils :  Visual studio 2022 pour le backend  Vscode pour le frontend  PgAdmin et postgres pour la base de données  Postman pour le tests | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai développé l’application seul | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Simplon** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Concepteur développeur d’applications | | | | | |
| **Période d’exercice** | | ► | **Du** | | Aout 2024 | | **au** | Novembre 2025 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **2** | | | **Concevoir et développer une application sécurisée organisée en couches** | | | | |
| ***Exemple n°3*** ► | | | | Concevoir et mettre en place une base de données relationnelle | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Dans le même contexte que la partie précédente, j’ai conçu la base de données afin de répondre aux besoins du client, un professeur qui dispense des cours particuliers.   * **Professeur** : il dispose d’un profil et peut créer plusieurs créneaux horaires (**1,N**). Cependant, il ne peut créer qu’un seul créneau par période donnée, et les créneaux ne doivent pas se chevaucher. * **Élève** : chaque élève est un utilisateur ayant également un profil. * **Créneau** : un créneau est proposé par le professeur (**1,N**) et peut être réservé une seule fois par un seul élève (**0,1**). * **Réservation** : une réservation est toujours liée à un seul créneau (**1,1**) et à un seul élève (**1,1**). * **Commande** : une réservation appartient à une seule commande (**N,1**), mais une commande peut contenir plusieurs réservations (**1,N**).   Ainsi, la structure relationnelle permet de gérer :   * la création des créneaux par le professeur, * la réservation unique d’un créneau par un élève, * et la centralisation des réservations dans une commande unique regroupant éventuellement plusieurs créneaux.   C:\git\template-angular-19-dashboard\dossier-projet-images\profilDB.png  Figure 1 : Dans cet exemple, il s’agit des tables liées directement à l’**utilisateur** et à l’**authentification**. Description des Entités Principales USER (Utilisateur) : Entité centrale du système représentant les utilisateurs (étudiants, formateurs, administrateurs). Chaque utilisateur peut avoir plusieurs rôles et possède un profil complet avec informations personnelles et professionnelles.  SLOT (Créneau) : Représente les créneaux horaires disponibles créés par les formateurs. Chaque créneau a un prix, peut avoir une réduction et est typé selon le service proposé.  BOOKING (Réservation): Entité de liaison entre un utilisateur et un créneau. Une réservation contient les détails de la demande d'aide et peut inclure des communications via chat.  ORDER (Commande) : Regroupe une ou plusieurs réservations pour le processus de paiement. Gère le cycle de vie commercial avec statuts, méthodes de paiement et TVA.  Curus : Représente les parcours de formation structurés par niveaux et catégories, permettant une organisation pédagogique cohérente. Vu d’ensemble   Figure 6Illustration générale sans tenir compte des tables d’authentification générées par Entity Framework. | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| **1. Moyens logiciels et techniques**   * **SGBD** : PostgreSQL et PgAdmin * **Langage de requête** : SQL et dotnet en utilisant la librairie Entity Framework   Le developpement a ete realise sous windows en utilisant pgadmin et visual studio | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai travailé seul | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Simplon** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | CDA | | | | | |
| **Période d’exercice** | | ► | **Du** | | Aout 2024 | | **au** | Novembre 2025 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **3** | | | **Préparer le déploiement d’une application sécurisée** | | | | |
| ***Exemple n°1*** ► | | | | Préparer et exécuter les plans de tests d’une application | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| *Les tests unitaires :*  Les test unitaires suivent les meilleures pratiques du framework xUnit pour .NET avec une isolation complète utilisant des mocks et stubs pour isoler les unités testées de leurs dépendances, une couverture qui cible des cas nominaux, cas d'erreur, et cas limites, une convention de nommage claire décrivant le scénario testé et le résultat attendu. EntityFrameworkCore.  Techno : XUnit, Moq et InMemory(entity Framework)  **Note :** InMemory (Entity Framework) simule une base de données. Cependant, les relations entre les différentes entités ne sont pas prises en charge et les contraintes relationnelles n’influencent pas les tests unitaires. Ces relations sont exploitées plus en détail dans les tests d’intégration présentés dans la section suivante.  [Fact]  public async Task Login\_UserNotFound\_ReturnsErrorResponse()  {  Environment.SetEnvironmentVariable("JWT\_KEY", "verylongj...key");  var userLoginDTO = new UserLoginDTO  {  Email = "nonexistent@example.com",  Password = "TestPassword123!"  };  var mockResponse = new Mock<HttpResponse>();  // configurer le UserManager pour retourner "null"  \_mockUserManager.Setup(x => x.FindByEmailAsync(userLoginDTO.Email))  .ReturnsAsync((UserApp?)null);  var result = await \_authService.Login(userLoginDTO, mockResponse.Object);  // verifier les resultats  Assert.NotNull(result);  Assert.Equal(404, result.Status);  Assert.Equal("L'utilisateur n'existe pas ", result.Message);  Assert.Null(result.Data);  } | | | | | | | | |
| *Les tests d’integrations :*  Les tests d'intégration de notre API servent a mettre réellement notre application à l'épreuve dans des conditions quasi-réelles. Contrairement aux tests unitaires qui isolent chaque composant comme dans un laboratoire stérilisé, nos tests d'intégration imitent la complexité du monde réel en faisant interagir tous les éléments ensemble : contrôleurs, services, base de données, authentification, autorisation, et même la sérialisation JSON. Ce qui rend cette approche particulièrement puissante, c'est l'utilisation intelligente de conteneurs Docker avec Testcontainers pour PostgreSQL, nous permettant de créer un environnement de test complètement isolé et reproductible. Chaque fois qu'un test s'exécute, une nouvelle base PostgreSQL fraîche est créée dans un conteneur, peuplée avec des données de test soigneusement préparées, puis détruite une fois les tests terminés. Cette approche nous donne une confiance énorme : si nos tests d'intégration passent, nous savons que notre API fonctionnera en production, car nous testons avec une vraie base de données PostgreSQL, de vrais appels HTTP, et une vraie pile d'authentification JWT.  public async Task InitializeAsync()  {  await \_postgresContainer.StartAsync();  Console.OutputEncoding = System.Text.Encoding.UTF8;  Environment.SetEnvironmentVariable("API\_BACK\_URL", "https://localhost:7113");  Environment.SetEnvironmentVariable("API\_FRONT\_URL", "https://localhost:4200");  Environment.SetEnvironmentVariable("SMTP\_BREVO\_PORT", "587");  Environment.SetEnvironmentVariable("SMTP\_BREVO\_SERVER", "smtp-relay.brevo.com");  ...  }  **Dans la phase de configuration**, la première étape consiste à lancer le **container de test** (containerTest), qui servira de base d’exécution isolée pour les scénarios de test. Une fois le conteneur démarré, je procède à la configuration des **variables d’environnement** nécessaires (par exemple les chaînes de connexion, les clés ou paramètres spécifiques aux services).  Ensuite, j’**override** la configuration par défaut du projet, en particulier le fichier **Program.cs** ainsi que le **DbContext** d’Entity Framework. Cette redéfinition permet de remplacer la base de données réelle par une **base de données de test** dédiée. Ainsi, toutes les opérations effectuées par l’application au cours des tests sont redirigées vers un environnement contrôlé, garantissant l’indépendance des tests vis-à-vis de l’infrastructure de production et facilitant leur reproductibilité.  services.AddDbContext<ApiDefaultContext>(options =>  {  options.UseNpgsql(\_postgresContainer.GetConnectionString());  options.EnableSensitiveDataLogging();  });    var serviceProvider = services.BuildServiceProvider();  // creer la base de donnees et appliquer les migrations de base  using var scope = serviceProvider.CreateScope();  var context = scope.ServiceProvider.GetRequiredService<ApiDefaultContext>();  var userManager = scope.ServiceProvider.GetRequiredService<UserManager<UserApp>>();  var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<Role>>();  Dans ce code, je remplace les options de services.AddDbContext par de nouvelles options pointant vers la base de données de test. J’y configure également les services dans le même contexte. Les tests qui en découlent sont similaires aux tests unitaires, c’est pourquoi je ne les détaille pas dans cette section.  *Intégration continue :*  Pour l’automatisation de l’intégration continue (CI), j’ai mis en place un pipeline à l’aide de **GitHub Actions**. Cette solution me permet d’exécuter automatiquement les étapes de compilation, de restauration des dépendances ainsi que les différents types de tests (unitaires et d’intégration) à chaque modification du code. L’objectif est de garantir une qualité constante du projet et de détecter rapidement d’éventuelles régressions avant le déploiement.  Dans le pipeline CI, les tests sont organisés en deux étapes distinctes. La première étape exécute les **tests unitaires** afin de valider le comportement isolé des différentes composantes de l’application. Une fois ces tests passés avec succès, la seconde étape lance les **tests d’intégration**. Ceux-ci sont exécutés sur un environnement Ubuntu configuré via GitHub Actions : le code est d’abord restauré et compilé, puis les tests d’intégration sont lancés à l’aide de la commande dotnet test sur le projet dédié. Cette organisation garantit qu’aucun test d’intégration n’est exécuté tant que les tests unitaires n’ont pas été validés, assurant ainsi une approche progressive et fiable de la validation du code.  test-unitaire:  name: Tests Unitaires  runs-on: ubuntu-latest  steps:  ...  test-integration:  name: Tests d'Intégration  runs-on: ubuntu-latest  needs: test-unitaire  steps:  - uses: actions/checkout@v3  - name: Setup .NET  uses: actions/setup-dotnet@v3  with:  dotnet-version: "8.0.x"  - name: Restore dependencies  run: dotnet restore  - name: Build  run: dotnet build --no-restore  - name: Lancer les tests d'intégration  run: dotnet test --no-build --verbosity detailed ./TerminalTestIntegration/TerminalTestIntegration.csproj | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Visual studio 2022 pour le backend  Docker desktop  Github Actions | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai travaillé seul pour ce projet | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Cliquez ici pour taper du texte.** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Skill hive | | | | | |
| **Période d’exercice** | | ► | **Du** | | Juillet 2025 | | **au** | Aout 2025 |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **3** | | | **Préparer le déploiement d’une application sécurisée** | | | | |
| ***Exemple n°2*** ► | | | | Préparer et documenter le déploiement d’une application | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| L’**architecture de documentation** mise en place repose sur une génération automatique à partir du code source. Cette approche permet d’éliminer les risques de désynchronisation entre l’implémentation et la documentation, un problème récurrent dans les projets logiciels. En effet, chaque modification du code est immédiatement prise en compte et reflétée dans la documentation, ce qui garantit une fiabilité constante et une mise à jour continue des informations techniques.  Du côté du **back-end**, j’ai configuré **Swagger** afin de documenter automatiquement l’ensemble des API exposées. Chaque endpoint est enrichi de commentaires et d’annotations spécifiques, telles que [Consumes("application/json")] ou [Produces("application/json")]. Ces métadonnées facilitent la compréhension des contrats d’échange entre le client et le serveur et rendent la documentation plus explicite et exploitable, aussi bien pour les développeurs que pour les testeurs ou intégrateurs.  C:\git\template-angular-19-dashboard\dossier-projet-images\docs.png  Pour la partie **front-end**, j’ai intégré **Compodoc** afin de générer la documentation du code Angular. Cet outil analyse automatiquement les composants, services, modules et directives, puis produit une documentation complète sous forme de fichiers HTML statiques. Ceux-ci peuvent être facilement consultés depuis un navigateur, par exemple via l’URL [doc.skill-hive.fr](http://doc.skill-hive.fr), ce qui facilite le partage et la consultation par toute l’équipe de développement.    En complément, une grande partie des services front-end est générée automatiquement grâce à la librairie **openapi-typescript-codegen**. Celle-ci exploite la documentation fournie par le back-end (via OpenAPI/Swagger) pour produire des services Angular fortement typés. Cette approche présente un double avantage : elle assure une stricte cohérence entre le front et le back, tout en permettant de réutiliser la documentation du back-end dans le front-end. | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Cliquez ici pour taper du texte.** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Cliquez ici pour taper du texte. | | | | | |
| **Période d’exercice** | | ► | **Du** | | Cliquez ici | | **au** | Cliquez ici |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | | | |
| **Activité-type** | **3** | | | **Préparer le déploiement d’une application sécurisée** | | | | |
| ***Exemple n°3*** ► | | | | Contribuer à la mise en production dans une démarche DevOps | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Déploiement en Environnement de Production Dans la continuité du contexte précédemment évoqué, l’application a été déployée sur un serveur **VPS Ubuntu 24.04 LTS** hébergé chez **Hostinger**. L’objectif est d’assurer un environnement de production stable, automatisé et facilement maintenable grâce à une infrastructure basée sur **Docker**, **Nginx Proxy Manager** et **GitHub Actions**. Configuration de l’Environnement  * **Serveur :** VPS Ubuntu 24.04 LTS (Hostinger) * **Orchestration :** Docker Compose pour la gestion et le déploiement des conteneurs * **Reverse Proxy :** Nginx Proxy Manager pour la gestion des domaines et la configuration automatique des certificats SSL * **CI/CD :** GitHub Actions pour l’automatisation des tests, de la construction des images Docker et du déploiement  Conteneurisation avec Docker La conteneurisation repose sur une **approche multi-stage**, permettant d’optimiser la taille des images Docker tout en facilitant la génération de différents environnements (développement, préproduction et production). Cette architecture garantit des déploiements rapides, cohérents et reproductibles sur le VPS.  **Architecture de déploiement sur VPS**    La conteneurisation repose sur une **approche multi-stage**, permettant d’optimiser la taille des images Docker tout en facilitant la génération de différents environnements (développement, préproduction et production). Cette architecture garantit des déploiements rapides, cohérents et reproductibles sur le VPS. Tests et Validation avant Déploiement Avant chaque mise en production, un processus d’intégration continue (CI) exécute automatiquement une série de vérifications pour garantir la fiabilité du code. Ce pipeline se déroule en trois étapes principales :   1. **Tests unitaires** : Vérification du bon fonctionnement des composants individuels (services, contrôleurs, logique métier) dans un environnement isolé. 2. **Tests d’intégration** : Validation du bon comportement global de l’application : interactions entre les services, base de données, API, authentification et communication inter-modules. Ces tests s’exécutent dans de véritables conteneurs Docker afin de reproduire fidèlement les conditions de production. 3. **Déploiement automatique** : Si toutes les étapes précédentes sont validées avec succès, le déploiement sur le VPS est déclenché automatiquement via SSH.     Dans cet exemple : le workflow GitHub Actions automatise entièrement le processus de **livraison continue (CD)** de l’application. Il permet de garantir que chaque mise à jour validée sur le dépôt est testée, empaquetée et déployée sur le serveur de production de manière fiable et reproductible.  Le pipeline se déclenche à chaque nouvelle version ou push sur la branche principale. Il commence par définir la version de l’image Docker à utiliser, puis récupère le code source du projet. Ensuite, il se connecte à **Docker Hub** via des identifiants stockés en toute sécurité dans les **secrets GitHub**, construit l’image Docker correspondante, et la pousse sur le registre distant.  Une fois l’image publiée, le workflow établit une connexion SSH vers le **VPS Hostinger** pour effectuer le déploiement. Le serveur télécharge la dernière version de l’image, puis relance le conteneur à l’aide de **Docker Compose**. Cette étape garantit que la mise à jour est appliquée immédiatement, sans intervention manuelle, tout en maintenant la continuité du service. | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Par rapport à la dernière section, les moyens supplémentaires utilisés sont :   * Un **VPS** hébergé chez **Hostinger** (2 CPU et 8 Go de RAM) * Un **compte GitHub** et un **compte Docker Hub** | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association** ► | | | | | | **Cliquez ici pour taper du texte.** | | |
|  | |  | | | | | | |
| **Chantier, atelier, service** | | ► | Cliquez ici pour taper du texte. | | | | | |
| **Période d’exercice** | | ► | **Du** | | Cliquez ici | | **au** | Cliquez ici |
|  | | | | | | | | |
|  | | | | | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | | | | | |
| Cliquez ici pour taper du texte. | | | | | | | | |

|  |  |  |
| --- | --- | --- |
| **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. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |
| Cliquez ici. | Cliquez ici pour taper du texte. | Cliquez ici pour sélectionner une date. |

|  |
| --- |
| **Déclaration sur l’honneur** |
|  |
|  |

*Cliquez ici pour taper du texte.*

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.

*Cliquez ici pour choisir une date*

*Cliquez ici pour taper du texte.*

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)* |
|  |