|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Nom de naissance* |  | VAYSSIER |
| *Nom d’usage* |  | VAYSSIER |
| *Prénom* |  | Sylvain |
| *Adresse* |  | 44 rue de chateaudun 63000 Clermont-Ferrand |
|  |  |

|  |  |
| --- | --- |
| **Titre professionnel visé** | |
|  | |
| Concepteur développeur informatique | |
|  | |
| **Modalité d’accès :** | |
|  | |
| * ☐ | Parcours de formation |
| ☐ | Validation des Acquis de l’Expérience (VAE) |
|  | |

**1**

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

**2**

**3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sommaire** | | | |
| **Exemples de pratique professionnelle** | | | |
| **Développer une application N-Tiers** | | **p.** |  |
|  |  OffSec : Gestion de projet par méthode AGILE p. | p. |  |
|  |  OffSec : Développement Back-End architecture MVC/RESTfull  p. | p. |  |
|  |  Offsec : Sécurisation et déploiement  p | p. |  |
|  |  |  |  |
| **Développer les composants d’interfaces** | | **p.** |  |
|  |  OffSec :Maquettage du projet p. | p. |  |
|  |   OffSec :Développement Front-End p. | p. |  |
|  |  Intitulé de l’exemple n° 3  p | p. |  |
|  |  |  |  |
| **Développer la persistance des données** | | **p.** |  |
|  |  Mise en place de la base de données  p. | p. |  |
|  |  OffSec : Développement Front-End  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. |  |

**4**

**Exemples de pratique**

**professionnelle**

**5**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** |  | | |  | | | | |
| **Activité-type** | **1** | | | **Développer une application a N-Tiers** | | | | |
| ***Exemple n°1***  | | | | ***OffSec : Gestion du projet*** | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Offsec est une API RESTfull développée à la demande du groupe Be-Ys, par le biais de M. Charbonnel Rémi, notre commanditaire.  Cette API a pour but objet d’être une console de sécurité offensive permettant d’automatiser certaines tâches de test de pénétration, notamment les scans de vulnérabilités.  Après recueil des besoins auprès de notre commanditaire, nous avons convenu de déployer une application multi-tiers RESTfull aux fonctionnalités suivantes :   * Un système de contrôle d’accès et de droits utilisateurs selon 3 r : *user*,*reporter*,*admin.* * Un système de création de nouveaux jobs, de nouveaux utilisateurs en fonction des droits * Une API REST en Java exposé en http(s) via Springboot 1.5 sécurisée, intégré dans l’environnement de travail d’ Almerys,contenant le cœur de l’application et la base de données principale. * Une API en python exposé aussi en REST grâce à la bibliothèque Hug responsable des jobs offensifs. L’API en java servira de tampon évitant des injections de commande directement dans un shell. * Un Front-End développé en Angular pour une expérience UI/UX maximum vis à vis des données. * Un système de reporting pour automatisé un rapport concernant l’état d’un réseau (optionnel). * Un système de différentiel entre les scans (optionnel).   Suppléant aux besoins de notre client, nous avons décidé de développer cette application dans un contexte de sécurité optimal comme le montre l’analyse de risque ci-dessous,    et avec un prototype livrable dès que possible comme le prévoit la méthode AGILE, permettant le remaniement de l’application si nécessaire.  Afin de rendre compte du travail effectué dans le cadre de notre projet, nous devions régulièrement soumettre une feuille de compte rendu d’activité, et pour parvenir à cela nous devions fixer clairement les objectifs et les tâches afin de pouvoir suivre une roadmapet d’évaluer les délais de notre travail. Il s’avéra que la plupart des tâches que nous avions décrites ne reposaient pas sur l’utilisation d’une technologie particulière.  Au cours de notre formation, nous avions étudié le fonctionnement d’une API RESTfull à l’aide de Springboot, et l’environnement de travail de notre commanditaire reposait elle aussi sur une infrastructure conçue autour de ce Framework. Il nous apparut naturel de choisir ce Framework pour développer le cœur de notre API centrale.  Cependant, la technologie Springboot fournit un environnement de travail redoutablement productif. En effet, la nature de ce support permet au développeur de ne coder que la partie métier d’une application et permet de s’absoudre d’énormément de contraintes. Il génère une quantité phénoménale de fichier de configuration par défaut que le développeur peut surcharger à sa convenance qui font de ce Framework un petit bijou productif et sécurisé. Il est à noter que nos délais pour les tâches préalablement définie furent considérablement transformé : la partie métier du cœur de l’application pris beaucoup plus de temps que prévu alors que toutes la mise en place de la configuration de la base de données et de la communication entre les APIs et le Front-End furent considérablement simplifié grâce à l’architecture de lu schéma ci-dessous.  Le schéma ci-dessous présente aussi la manière dont les composants vont communiquer entre eux à partir de L’API Springboot. Chaque composants (API Java, API python, Font-end Angular) fut développé indépendamment par chacun membre de notre groupe de travail en respectant les conventions habituelles pour permettre aux différents composant de communiquer naturellement une fois leur développement fini.  Pour nous permettre de nous organiser conceptuellement autour d’une vision commune nous avons consulté beaucoup de documentation quand à ce qu’il existé en matière de console de sécurité offensive. Une application reteint notre attention : il s’agit de Nessus : un scanner de vulnérabilités d’une efficacité avéré et dont l’environnement utilisateur doté d’une UI/UX est déjà parfaitement éprouvé. Nous avions donc une idée et un modèle conjoint qui permit à notre groupe de travail de modeler notre projet avec un objectif clairement défini tout en restant chacun parfaitement autonome dans le développement respectif de son composant.  J’aborderai la partie Front-End dans la partie dédiée à ce chapitre tout en traitant des digrammes de cas d’utilisation et de classe ainsi que de la mise en place de la base de données.  **API SPRINGBOOT**  **FRONT-ENT ANGULAR**  FRONT-END ANGULAR   * Framework modulaire, rapide, facilement intégrable parfait pour une application Mono-page. (Nous en verrons plus dans la partie développement Front-End) | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| Pour réaliser cette partie de ce projet j’ai utilisé :   * Un ordinateur sous windows 10 Home Edition * Excel de la suite Office 2016 * Documentation :   - EBIOS 2018 (web)  - OpenClassroom (web)  - Personnal MBA (livre) | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| J’ai réalisé la partie de gestion de ce projet seul <<++collegues>>, étant le responsable du groupe de travail, en travaillant sous la tutelle de mon responsable pédagogique Benjamin Georgeault et mon commanditaire de l’entreprise de Be-Kortalys, Monsieur Rémi Charbonnel | | | | | | | | |
|  | | | | | | | | |
| **4. Contexte** | | | | | | | | |
|  | | | | | |  | | |
| **Nom de l’entreprise, organisme ou association**  | | | | | | ***Be-Kortalys filiale de Be-YS*** | | |
|  | |  | | | | | | |
| **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**  **7** | **1** | | | **Développer une application a N-Tiers** | | | | |
| ***Exemple n°2***  | | | | ***Développement Back-End, architecture MVC/RESTfull*** | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Le Back-End de l’application OffSec est le cœur de l’application. Il gère les échanges entre l’API en python ainsi que les échange avec le Front-end via des objets Json sur des requêtes effectuées sur les « *end points »* des contrôleur Rest. Il est aussi le composant principal car c’est dans ce composant que la logique métier est incorporé :   * Parser les données Json pour les uniformisées et les mettre dans la base de donnée MYSLQ * La logique concernant les jobs et la gestion du *Pool* des jobs * La mise en relation des différents composants notamment par l’utilisation des RestController * La logique entre les différentes classes * La gestion du SGBDR * La sécurisation de l’application * Le contrôle d’accès * Les droits utilisateurs * La gestion des tests unitaire avec la dépendance Junit * La gestion des tests d’intégrations * La documentation de l’application via la dépendance Swagger * L’architecture reposant sur la Java Virtual Machine, le déploiement et le développement de l’application se fait universellement quel que soit le système sur lequel le système opère. * Le déploiement de l’application est très bien documenté et très instinctif * S’ajoute à cela une très bonne résilience au test d’intrusion * Est conçu dans un but de travail collaboratif   Pour cela, Springboot ne s’absout pas de la règle fondamentale d’une bonne réflexion autour d’un diagramme de classe pour permettre un développement rapide. Afin de répondre au besoin de notre commanditaire, qui voulait notamment au moins un scanner de vulnérabilités en tant que job offensif minimal, il a fallu créer les classes de manières commune à tous les jobs de sécurité offensif, à savoir les vulnérabilités (CVE), les services associés (Services), les ports associés (Port) et les hôtes scannées (Host).  La demande de notre client stipulait aussi une gestion avancé des jobs de sécurité offensive (mise en pauvre, statut, file d’attente….) ainsi qu’un système de contrôle d’accès.(cf. Annexe n° I : *cahier des charges* )Pour pallier à ces demandes, l’application se verra implémentée d’une classe User qui héritera d’une classe spécialisée de Springboot permettant la surcharge des composants de sécurité ; et d’une class job permettant l’implémentation de la logique de la gestion des dits « jobs ».  Comme dit précédemment le Framework Springboot permet au développeur de déployer toute ses compétences sur la logique de son application, et c’est dans ce contexte que les classes Job et User (et leur dérivé comme \*Service, \*Controller…) se verront comme étant les composants les plus critiques en matière de développement.    Ce schéma relationnelle de base de données, qui sera repris dans la partie base de données, permet la modélisation des données à travers le Framework Springboot qui se compose d’une architecture bien cadré, comme indiqué ci-dessous.  Springboot fournit un environnement de travail complet notamment grâce au Pom.xml. Ce fichier, catalogue de la bibliothèque de dépendance Maven, permet d’inclure les dépendances nécessaires au développement de son application. Le reste de l’architecture se compose de deux dossiers séparer qui se font écho ; puisque l’un permettra de stocker toutes le code métier du développeur tandis que l’autre permet de coder les tests du code métier associé. Finalement la bonne pratique est de séparé le code en package généralement correspondant à la nomenclature des bonnes pratiques du développeur :   * Un package Domain qui contient la définition des classes avec les annotations Json. * Un package DTO permettant de créer un objet qui est une copie des classes Domain permettant la manipulation des objets sans prendre les instances elle mêmes * Un package Rest Controller permettant de gérer les « *end points* » de l’application. * Un package Model permettant de stocker les CRUD Repository/ JPA Repository permettant le dialogue entre Java et la Base de donnée * Un package (une classe dans mon cas) Security permettant de surcharger les comportements de sécurité par défaut de Springboot * Un package Service correspondant à la logique de travail dans la base de données avec les classes Domaines préalablement écrites.   Afin de permettre un développement aisé et continue, je choisis l’environnement de développement Eclipse IDE qui permet d’adjoindre avec facilité git et les différents outils de développement collaboratif, notre commanditaire nous ayant fourni un dépôt Gitlab local sur lequel travailler.  Je commence par la création de l’architecture de l’applications grâce à Spring Initializr, et l’ajout des dépendances jugé nécessaire pour le projet dans le fichier de configuraiton « pom.xm l »    *Un exemple de dépendance dans le pom.xml*  La tâche suivante fut de construire les classes. Elle se construise de la même manière que les classes en Java mais bénéficie d’annotation (les équivalent des décorateurs en python). Voici un exemple avec la classe Job.    *Un exemple à travers l’implémentation de la classe Job avec Springboot*  Les annotations permettent de décrire le comportement de Springboot quand il est confronté à une donnée. Par exemple la classe Entity permet de spécifier que cette classe sera inscrite dans la base de données alors que « JsonProperty » permettent de spécifier que ces champs sont à inscrire dans la base de données. On pourrait spécifier le champs mais Springboot prend le nom des variables comme nom de champ par défaut dans la base de données.  Une catégories d’annotations particulières est a remarqué : les annotations relationnelles « @OneToMany » et « «@ManyToOne ». Ces relations permettent de créer les liens entre les différentes entités de la même manière que les jointure SQL le font entre les différentes tables de la base de données. Le paramètre « mappedBy » permet de préciser le End-Point qui est primordiale dans les relations de type « @ManyToMany » tandis que les autres paramètres permettent à Springboot de savoir quelle comportement adoptés lords de la suppression ou lors de la création.Enfin l’annotation « @JsonCreator » permet de spécifier quel est le constructeur de l’objet.  Une fois toutes les classes de bases implémentées, il fallut créer les classes DTO et les Repository.  *La classe JobDTO L’interface JobRepository*    *L’arborescence actuelle*  Le fonctionnement des repository avec Springboot ets très instinctif. Pour pouvoior les utiliser iol faut créer une interface qui hérite soit de CRUDRepository soit de JPARepository, JPA permettant une communicaiton plus complètes avec les méthodes de Springboot mais CRUD suffira pour les besoin den notre applicaiton. En effet le foncitonnement de ces repository repose sur l’utilisation de mot clés permettant de définir ce que la méthode va effectuer. Ce système permet de remplacer aisément la plupart des requêtes que le développeur est amené à effectuer. Pour les cas les plus complexe il reste possible d’implémenter les requêtes en renseignant la syntaxe SQL correspondante.  Dans notre cas le mot clé FIND vas parcourir les table et nous renvoyer une colleciton puisque le mot ALL est présent et By permet de préciser la variable surlaquelle portera la recherche.  Je poursuis avec les classes permettant d’utiliser l’interface que nous venons de créer :  *La classe JobService intyerrogeant le Repository*  Les classes principales étant construite il nous reste a construire le point d’entré de l’applciaiton les Rest Controlleurs.  Les Rest Controlleurs sont majoritairement construit autour d’annotations particulière. Le « @GetMapping » permet de créer un « endpoint » avec la méthode get à laquelle l’API répondra et la méthode PostMapping() qui, elle, permet de créer un « endpoint » avec la méthode POST et qui dans notre cas réclammera d’avoir dans le body de la requête un objet Json erconnaissable par l’API.    *Un exemple de Rest Controlleur avec des méthodes GET*    *Un exemple de Post qui stocke les informations dans la Base de données*  Dans l’exemple précédent, lors du développement il m’est apparu necessaire de vérifier le foncitonnement du PostMapping, c’est pourquoi, j’ai implémenter un moyen de nourrir la base de données en postant des objet Json en cascade. La fonction prend en effet un paramètre qui est un JobDTO qui sera décortiquer par Springboot pour le parser directement dans la base de données. Il est alors primordiale de lier les entités lors de l’instanciations des objets :    Enfin il est necessaire d’ordonner la sauvegarde de l’instanciation de ces objets avec :    Cette fonction retour la référence de l’objet ainsi créer à l’URL qui lui correspond.    *Utilisation de Postman pour communiquer avec l’API*  Le premier prototype de base étant fini et opérationnel, la phase suivante du développement était d’implémenter la communication entre l’API en Python qui est responsable de l’éxécution des jobs de sécurités offensives, et le Back-End en Java.    *Le Rest Controlleur de l’API en Python*  Communiquer avec le rest Controlleur de l’API Hug (python) me parut compliqué jusqu’à ce qu’une méthode miraculeuse de Springboot vienne à mon secours :    *Encore une méthode de Springboot redoutablement efficace*  Springboot permet de récupérer directement depuis une URL un objet Json grâce à une template REST qui va automatiquement caster la sortie en JSON en un objet, en l’occurrence HostsDTO qui est une classe de référence à un tableau d’HostDTO. Il est à noter que lors du développement, certains jobs de scan étaient suffisament lourd pour causer un timeout sur la connection. Cette méthode me permti d’implémenter ‘linstruciton de commander un nouveau scan qui me permettait de récupérer les données et de les parser dans la base de données.    *Ah ! J’ai des vulnerabilités sur mon PC ! Ca fonctionne comme prévu.* | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| **Pour développer cette partie, j’ai mis en place l’infrastructure suivante :**   * **Un ordinateur portable sur Windows7 avec un VM de Kali-Linux 4 avec l’API en python** * **Un ordinateur fixe avec Ubuntu 18.04 avec un server Apache vulnérable** * **Un ordinateur fixe avec Windows 10 faisant tourner l’API en JAVA et le Front-End** * **Un dépôt Gitlab gracieusement prêté par Almerys**   **Mes besoins logiciels ont été :**   * **Eclipse IDE 4.10 avec Spring tools Suite** * **Python avec les bibliothèques Hug/requests/urllib.request** * **Postman / curl** * **Wamp Server** * **Firefox/Chrome** * **Oracle JDK** * **Sprinboot 1.5.19 avec les dépendances suivantes : BCrypt/staterJPA/FasterXML/MysqlConnector/spring-boot-starter/httpclient/swagger/jaxb/starter-security/mockmvc** * **Gitbash**   **Mes besoins documentaire :**   * [**https://www.baeldung.com/**](https://www.baeldung.com/) * [**https://stackoverflow.com/**](https://stackoverflow.com/) * **https://repo.spring.io/release/** | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| **J’ai développé le Back-end en java majoritairement seul tandis que l’api en python a été construite majoritairement par Yegor Nickolov, membre de notre groupe de travail. Nous avons la collaboration régulière de notre directeur pedagogique Benjamin Georgeault, notre formateur Springboot Marc Fichant, notre commanditaire Remi Charbonnel ainsi que de mes collègues de formations qui m’ont étaient d’une grande aide quand à la compréhension globale de Springboot** | | | | | | | | |
|  | | | | | | | | |
| **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** | **1** | | | **Développer une application a N-Tiers** | | | | |
| ***Exemple n°3***  | | | | ***Offsec : sécurisation et déploiement*** | | | | |
|  | | | |  | | | | |
| **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** | **1** | | | **Cliquez ici pour entrer l’intitulé de l’activité** | | | | |
| ***Exemple n°1***  | | | | *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. | | | | | | | | |

**7**

|  |  |  |
| --- | --- | --- |
| **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. |

**8**

|  |
| --- |
| **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 taper du texte.*

*Cliquez ici pour choisir une date*

Fait à le

pour faire valoir ce que de droit.

Signature :

**9**

|  |
| --- |
| **Documents illustrant la pratique professionnelle** |
|  |
| *(facultatif)* |
| **Intitulé** |
| Cliquez ici pour taper du texte. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**10**

|  |
| --- |
| **Annexes** |
|  |
| *CAHIER DES CHARGES*  *Exemples d’objet JSON*  *Sources : base de données nvd/ vulners/Nessus* |
|  |

**11**