|  |  |  |
| --- | --- | --- |
|  |  |  |
| *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. |  |
|  |  OffSec : Développement Back-End architecture MVC/RESTfull  . | p. |  |
|  |  Offsec : Sécurisation et déploiement | p. |  |
|  |  |  |  |
| **Développer les composants d’interfaces** | | **p.** |  |
|  |  OffSec :Maquettage du projet . | p. |  |
|  |   OffSec :Développement Front-End . | p. |  |
|  |  | p. |  |
|  |  |  |  |
| **Développer la persistance des données** | | **p.** |  |
|  |  Mise en place de la base de données  . | 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 qui 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 à l’architecture suivante :   * 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.   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,    *Analyse de risque EBIOS 2018*  et avec un prototype livrable dès que possible comme le prévoit la méthode AGILE, permettant le remaniement de l’application si nécessaire.  Nous avons donc organisé notre groupe de travail de manière à ce que chaque membre développe une des 3 composants principaux de notre application. Afin de correspondre à une vision commune du projet, nous avons adopté un modèle pour avoir une « roadmap » sur laquelle se basé.  Nous avons ensuite convenu de l’utilisation des technologies à savoir :   * Back-End Python en tant que Back-End offensif * Back-End Springboot en tant que cœur de l’application * Front-End Angular en tant que client utilisateur   **API SPRINGBOOT** | | | | | | | | |
|  | | | | | | | | |
| **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 en collaboration avec mon groupe de travail, en travaillant sous la tutelle de notre responsable pédagogique Benjamin Georgeault et de notre 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*** | | | | |
| Le Back-End de l’application OffSec est le cœur de l’application. Il gère les échanges entre l’API en python et 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 :   * Parsage des données dans la base de données * La logique concernant les jobs * La mise en relation des différents composants par HTTP * 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   Les besoins de notre client stipulaient au minimum un scanner de vulnérabilités en tant que job offensif, il a donc fallu implémenter les classes remontées par l’API en Python, et la table de gestion des jobs ainsi que la classe User qui nous permettra d’utiliser certaines classes cruciales de Springboot en ce qui concerne l’authentification    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. La bonne pratique est de séparé le code en package généralement correspondant à la nomenclature du modèle:   * Un package Domain qui contient la définition des classes. * Un package DTO 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.   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 configuration « 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 champ 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. Nous verrons plus en details les sépcifités de la relation à la base de données lors de la partie ocnsacrée à cette effet.  Dans notre cas le mot clé FIND vas parcourir les table et nous renvoyer une collection 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 :  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 l’ instruction 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.* | | | | | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
|  | | | | | | | | |
| **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 serveur 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. J’ai eu la collaboration régulière de mon directeur pedagogique Benjamin Georgeault, mon formateur Springboot Marc Fichant, mon commanditaire Remi Charbonnel ainsi que de mes collègues de formations qui m’ont étaient d’une grande aide quant 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 :** | | | | | | | | |
| La sécurisation de l’application se traduit par l’utilisation des TDD, des test d’intégration continue, et des test d’injections de données (Mock), cherchant dans le but de chercher a obtenir de l’application un comportement innatendue.  La sécurisation de l’application a été effectués notamment par les « *Test Driven Development* » qui sont des contrôles quant aux fonctions implémentées dans l’application. Ils permettent en général de contrôler de la viabilité des paramètres passés au fonctions comme le montre l’exemple suivant :    *Un exemple de TDD sur le domain de la class Host*  Comme le montre l’exemple précèdent, nous lançon une série de test qui sont censé retourner des Exceptions parce que nous les faisons passer en utilisant des valeurs qui pourraient entrainer un comportement anormal de l’application.  Certains des autres tests permettent de vérifier que la création des objets se passent sans aucun problème tout en testant certains valeurs critiques.  Les TDD se doivent d’être implémenté en même temps que le développement du code source de l’application et permettent au développeur de contrôler son code au fur et à mesure de sa progression. | | | | | | | | |
|  | | | | | | | | |
| **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** | | | **Développer les composants d’interfaces** | | | | |
| ***Exemple n°1***  | | | | ***OffSec : Maquettage du projet*** | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Le maquettage d’un projet est la partie la plus importante en ce qui concerne l’utilisateur. Une application bien conçue repose sur une interface utilisateur (UI) pertinent et ergonomique pour permettre une expérience d’utilisation satisfaisante (UX). Et dans ce domaine l’interface utilisateur de Nessus, le modèle de l’application OffSec, allie la simplicité et la pertinence :    *L’interface utilisateur de Nessus*  En premier lieu il me fallu décrire le comportement de l’application en fonction des actions utilisateurs en fonction des demandes du commanditaires à travers le cahier des charges situés en annexe. Finalement le prototype n’a que peu d’interaction utilisateur différents comme le montre le diagramme de cas d’utilisaiton suivant qui a été adaté pour un plus aisante lecture :    *Digramme de cas d’utilisation simplifié de l’applicaiton OffSec*  Le besoin du client est simple : ordonner une tâche, verifier le résultat et obtenir le maximum d’informations sur les cibles comme se le veut une applicaiton de sécurité offensive, c’est pour cela que le style simple et épuré de Nessus conviendra parfaitement à nos besoins. Et comme Nessus, OffSec se veut être une application monopage parfaitement accessible depuis un mobile ou une Rasberry.    *Une application monopage, simple et adaptée à tout support*  Maintenant que le design était décidé, il fallut discuter avec le client à propos de la charte graphique de l’application. L’entreprise cliente utilise en effet une charte graphique pour les médias numérique reposant sur la bibliothque Bootstrap, un framework CSS particulièrement reconnu dans le milieu du développement permettant une manipultaiton plus aisé du css en partageant notamment la page en une grille et qui intègre de nombreux composant , notamment couplé avec Js qui permet de développer un Front-End simplement, rapidement et avec esthétisme.  Pour permettre la mise en place d’une application mobile monopage, l’interface utilisateur devra reposer sur du JavaScript, ou tout du moins un ECMAScript.  Mon choix s’est porté sur Angular, un Framework Front-End intégrant un moteur de script parfaitement adaptés au exigences émergentes du développement Web. Ca Framework n’a été construit par rien d’autre que Google qui utilise toute la puissance du TypeScript, un langage ECMAScript développé par Microsoft. Angular n’est donc pas une Framework vraiment faillible et permet au développeur de se concentrer sur l’implémentation des foncitonnalitées et sur l’améliorationn de l’expérience utilisateurs. Il est parfait pour travailler avec des services en lignes. | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| **Pour le maquettage de l’application j’ai utilisé :**   * **Nessus pour l’inspiration** * **Dia / Modelio/ Visio pour les diagrammes** * **Balsamiq Mockup pour les mockups**   **Pour les recherches documentaires :**   * **Personnal MBA**   **Pour vérifier l’UI/UX, j’ai fait appel à Mme Radan Apolline, Infographiste/Metteuse en page** | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| **Pour le maquetage j’ai pu travailler avec mon groupe de travail dédié, M Yegor Nickolov, M Mayet Lionel ainsi que les personnes responsables de l’encadrement pédagogique de la formation, de notre commanditaire M Remi Charbonnel, et enfin de Mme Radan Apolline pour les retour exépriences.** | | | | | | | | |
|  | | | | | | | | |
| **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** | | | **Développer les composants d’interfaces** | | | | |
| ***Exemple n°2***  | | | | ***OffSec : Développement Front-End*** | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Le développement du Front-End se passera avec Angular, un Framework développé par Google dans lequel est incorporé TypeScript un langage ECMA-Script développé par Microsoft. Cet environnement de travail a été développé dans le but de développer des applications mono-pages adaptés à tout support. Dans ce but Angular, permet la conception de composants, divisant le Front-End par ces fonctionnalités incorporées sous la forme d’n Template HTML, d’une implémentation logique en TypeScript, un fichier de test et enfin un fichier CSS permettant de personnaliser le style du composant.    *L’implémentation d’une classe Composant sous Angular*  Comme le montre la capture d’écran précédente l’implémentation d’une classe « Component » se fait par l’instanciation dans le constructeur d’une classe Service qui effectuera les requetes http sur l’API en Java :    Angular fournit donc un moyen simple d’accéder aux données qui sont stockés dans la base de données grâce à l’API Springboot. Il nous reste plus qu’à fournir un Template HTML pour permettre l’affichage des données formatées.    Le code ci-dessus ne fonctionnerait pas en HTML puisqu’il inclut une des spécifités d’Angular : les directives. Les directives d’Angular sont des moyens d’accès aux données dans les Template HTML grâce aux variables déclarées dans le TypeScript à l’aide des deux directives les plus utilise : « \*ngIf » et « \*ngFor » qui sont respectivement la directive conditionnel et la directive itérative de base de ce Framework.  Le code précèdent affichera chaque champ de chaque élément « job » faisant partis du tableau de variable « jobs » dans lequel est stockées les résultats de la méthode GET précédente sur le Back-End en Springboot.  Comme pour l’API en Springboot il est nécessaire de déclarer les classes correspondantes pour un accès plus simple, et surtout formatée. Ci-dessous un exemple avec la définition de classe Job :    Cela me permet de formater les données simplement et rapidement pouvant mener au résultat suivant en quelques lignes de codes.  https://media.discordapp.net/attachments/510083843945005058/553131378737414160/unknown.png?width=1018&height=450 | | | | | | | | |
|  | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| **Pour cet exemple j’ai utilisé :**   * **Un ordinateur fixe sous Window 10 portant le Fron-End et le Back-End** * **Un ordinateur Portable sous Kali 4 portant l’API Python** * **Un ordinateur fixe sous Ubuntu 18.04 avec un server Apache vulnerable** * **Framework Angular** * **BootStrap / ng-BootStrap** * **NPM/Node.js** * **Angular CLI** * **Navigateur Web Firefox/Google Chrome**   **Mes besoins doculentaires ont été :**   * **Wikipedia.fr** * **Angular.io** * **Openclassroom.fr** | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| **Pour cet exemple j’ai travaillé avec M Mayet Lionel dont le rôle était de développer le composant Front-End, avec M Yegor Nickolov qui dévleopper en paralelle une autre version potentiel du Front-End, ainsi qu’avec les membres de l’équipe pédagogique qui nous ont formé a Angular avec qu’avec notre commanditaire M Charbonnel Remi.** | | | | | | | | |
|  | | | | | | | | |
| **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** | | | **Développer la persistance des données** | | | | |
| ***Exemple n°1***  | | | | ***OffSec : Base de données et persistance*** | | | | |
|  | | | |  | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | | | | |
| Avec un travail sur des données issue d’une base de données tels que celle de NVD, il est nécessaire d’uniformiser afin de persister les données pour une utilisation ultérieure. Pour cela Springboot utilise Hibernate, un Framework permettant la persistance des objets dans la base de données. Afin de satisfaire cette contrainte, il m’a fallu créer tout d’abord la base de données qui sur laquelle allait se construire l’application, et préciser à Springboot où trouver le data source et les informations de connexions à la base de données. La base de données reposera sur MySql qui fait partie du pack WampServer.    La base de données décrira le schéma relationnelle suivant :    Springboot utilise donc le Framework Hibernate avec JPA et CRUD pour pouvoir effectuer les requêtes. En effet, Springboot permet au développeur de s’absoudre de la communcation SQL. CRUD et JPA sont des classes java qui hérite de l’interface sus-mentionné et qui permettent au développeur une dynamique de codage axé essentiellement sur le métier :    *Une interface héritée de CRUD repository*  Dans l’exemple ci-dessus l’implémentation de la classe Job qui hérite de l’interface « CRUD Repository » permet de définir a Springboot et Hibernate le comportelent a adpopter visàvis de la logique de cette classe. En effet le CRUD Repository tout comme le JPA Repository permet de créer un similis e requête SQL en construisant les fonctions à base de mot clefs :  Dans l’exemple précèdent les trois fonctions implémentées repondent respsectivement au requêtes :    Dans l’activité Type numéro 1, j’avais présenté comment Springboot utilisait les annotations pour définit les relations entre les différentes entités de la base données. C’est de cette manière que Hibernate va induire les relations entres les différentes dtables de la base de données et faire les jointures entre les tables correspondant en fonction des annotations Hibernate « @OneToMAny », « @ManyToOne », « @ManyToMany » corespondant aux relations visibles dans n’importe schéma UML ou MCD.  Ce système permet de faire des requêtes simples, été il existe de nombreuses autres manières de formuler des requêtes plus compliqué avec des « @Query » ou encore l’utilisation de Predicate.  Après avoir implémenter toutes les classes correspondantes, le travail concernant la base de données est complémtement automatisé et le développeur n’a même plus a tenir compte des actions sur sa base de données. Ci-dessous une capture de la base de données vu par PhpMyAdmin après un scan reussi lancé depuis le Front-End.    *Hibernate gère la base de données seulement avec les classes et les annotations* | | | | | | | | |
| **Ack Wamp** | | | | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | | | | |
| **Pour la réalisaiton de cet exemple j’ai utilisé :**   * **Un ordinateur fixe sous Windows 10 portant le Back-End et le Front-End** * **Un ordinateur fixe sous Ubuntu18.04 avec un serveur Apache vulnérable** * **Un ordinateur portable sous Windows 7 portant l’API en Python** * **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**   **Ressouces Documentaire :**   * **Baeldung.com** * **StackOverflow.com** * **OpenClassRomm.fr** * **Wikipedia.fr** * **NullBytes/Wonder-how-to.com** | | | | | | | | |
|  | | | | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | | | | |
| **Pour cet exemple j’ai travaillé essentiellement seul en étroite collaborations avec mon groupe de travail avec lequel nous réfléchissions à la forme de la base de données.** | | | | | | | | |
|  | | | | | | | | |
| **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**