|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Nom de naissance* |  | Pac |
| *Nom d’usage* |  | Entrez votre nom d’usage ici. |
| *Prénom* |  | Pierre |
| *Adresse* |  | 7 rue Rober Bordeux 50580 Pont-Remy |
|  |  |

|  |  |
| --- | --- |
| **Titre professionnel visé** | |
|  | |
| Concepteur Développeur d’Application | |
|  | |
| **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** | | |
| **Intitulé de l’activité-type n° 1 : Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité** | | **p.** | **5** |
|  |  Intitulé de l’exemple n° 1 Développer des composants d’accès aux données p. | p. | 5 |
|  |  Intitulé de l’exemple n° 2 Développer la partie front-end d’une interface utilisateur web p. | p. | 8 |
|  |  Intitulé de l’exemple n° 3 Développer la partie back-end d’une interface utilisateur web p | p. | 12 |
|  |  |  |  |
| **Intitulé de l’activité-type n° 2 : Concevoir et développer la persistance des données en intégrant les recommandations de sécurité** | | **p.** |  |
|  |  Intitulé de l’exemple n° 1 Concevoir une base de données p. | p. | 16 |
|  |  Intitulé de l’exemple n° 2 Mettre en place une base de données p. | p. | 19 |
|  |  Intitulé de l’exemple n° 3 Développer des composants dans le langage d’une base de données p | p. | 23 |
|  |  |  |  |
| **Intitulé de l’activité-type n° 3 : Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité** | | **p.** |  |
|  |  Intitulé de l’exemple n° Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement p. | p. | 26 |
|  |  Intitulé de l’exemple n° 2 Développer des composants métier p. | p. | 28 |
|  |  Intitulé de l’exemple n° 3 Construire une application organisée en couche p | p. | 30 |
|  |  |  |  |
| **Titres, diplômes, CQP, attestations de formation** *(facultatif)* | | **p.** | 33 |
| **Déclaration sur l’honneur** | | **p.** | 34 |
| **Documents illustrant la pratique professionnelle** *(facultatif)* | | **p.** | 35 |
| **Annexes** *(Si le RC le prévoit)* | | **p.** | 36 |

**Exemples de pratique**

**professionnelle**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **1** | **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité** | | | |
| ***Exemple n°1***  | | Développer des composants d’accès aux données | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| ans notre application, élaborée avec Spring Boot, JPA, Hibernate, et Angular, les composants d'accès aux données sont essentiels, faisant le lien entre les utilisateurs et nos bases de données, et assurant une gestion des données fiable. Nous utilisons JPA et Hibernate pour gérer la persistance des données, guidés par nos modèles et repositories. La Java Persistence API (JPA) offre une interface pour gérer la persistance des données dans nos applications Java, favorisant une gestion orientée objet des données relationnelles et augmentant la productivité des développeurs grâce à sa portabilité et sa syntaxe simplifiée. Hibernate, une implémentation notable de JPA, facilite les opérations CRUD et s’intègre harmonieusement avec Spring Boot, optimisant le développement et réduisant la configuration grâce à l'automatisation de Spring Boot.  Configuration de la Source de Données  Ma décision d’opter pour PostgreSQL comme système de gestion de base de données est motivée par sa fiabilité et ses performances dans la gestion des données relationnelles. Ainsi, j'assure la configuration de la connexion à la base de données avec soin : | | | | | |
| Je spécifie ici les paramètres essentiels comme l'URL, le nom d'utilisateur, le mot de passe, et le pilote JDBC pour garantir une connexion fluide et fonctionnelle à la base de données.  Configuration JPA et Hibernate  L’intégration de JPA et Hibernate vise à simplifier l'accès et la gestion des données dans mon application :    Ici, le dialecte PostgreSQL est employé, et la propriété ddl-auto est configurée à update pour synchroniser automatiquement le schéma de la base de données avec les entités au démarrage de l'application.    Le package spécifié indique à JPA où localiser les entités, assurant ainsi une gestion appropriée des objets dans le contexte de persistance.  Afin de faciliter le débogage durant le développement, j’active l’affichage des requêtes SQL et leur formatage :    Cette pratique m’aide à comprendre et optimiser les interactions avec la base de données, améliorant ainsi la performance globale de l’application.  La configuration des composants d’accès aux données a été minutieusement effectuée pour assurer une persistance des données robuste et performante tout au long du cycle de vie de mon application. Bien entendu, cette configuration est susceptible d’être ajustée selon les évolutions et les besoins futurs du projet, en garantissant toujours l’intégrité et la sécurité des données manipulées.  Modélisation des Données et Relations entre Entités  Lors de la création de mon application, j'accorde une attention particulière à la définition des modèles qui serviront à créer les tables dans la base de données et à définir la logique métier.  Rôle des Modèles et Création des Tables  Dans le contexte JPA/Hibernate, le modèle Sandbox dans mon application joue un rôle crucial. Les modèles, annotés avec @Entity, se transforment en tables dans la base de données au démarrage de l'application, suivant la logique déclarative inscrite dans le code.  L’annotation @Entity indique à Hibernate que la classe Sandbox est une entité et doit être mappée à une table de base de données. @Table(name = "SANDBOX") spécifie le nom de cette table. Chaque instance de Sandbox correspond à une entrée dans la table "SANDBOX" de la base de données.    La propriété id est annotée avec @Id, indiquant qu’elle est la clé primaire de la table. @GeneratedValue détermine comment les valeurs de la clé primaire sont générées. Ici, GenerationType.IDENTITY signifie que la base de données génère automatiquement la valeur lors de l'insertion de l'entité.  Relations entre Entités : OneToMany  La gestion des relations entre les entités est cruciale pour refléter la logique métier dans la base de données.    @OneToMany représente une relation un-à-plusieurs entre Sandbox et Entretien. Le paramètre mappedBy indique que la propriété sandbox dans l’entité Entretien est la clé étrangère de cette relation. CascadeType.ALL signifie que toutes les opérations effectuées sur une instance de Sandbox (comme la sauvegarde, la mise à jour, ou la suppression) sont propagées aux entités associées dans entretiens. FetchType.EAGER, d’autre part, signifie que les entités associées sont chargées automatiquement avec l’entité parente.  La compréhension et l'implémentation précises des modèles et des relations sont vitales pour assurer que l'application interagit avec la base de données de manière cohérente et performante, alignant ainsi la persistance des données avec la logique métier souhaitée.  Le Repositoire : Facilitateur d'Accès aux Données  Dans mon projet, le repositorie joue un rôle essentiel en tant que pont entre la logique métier de l'application et les opérations de la base de données, facilitant ainsi la récupération et la persistance des données.  Structure et Fonctionnalité des Repositories    L’annotation @Repository indique à Spring que cette interface est un repositorie, un composant chargé de gérer la persistance des entités Candidat. Il hérite de JpaRepository, qui contient des méthodes pour les opérations de base de données les plus courantes, telles que la sauvegarde, la suppression, et la recherche d’entités, sans qu’il soit nécessaire d’implémenter ces méthodes manuellement. | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Spring Boot, JPA, Hibernate, PostgreSQL | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **1** | **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité** | | | |
| ***Exemple n°2***  | | Développer la partie front-end d’une interface utilisateur web | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Une fois Angular et PrimeNG installés, j'entreprends la création des divers modules, structurant ainsi mon application pour une organisation cohérente des différents composants. Cette démarche permet non seulement de segmenter le code de manière logique, mais aussi de faciliter la maintenance et l'évolutivité du projet, en assurant que chaque partie de l'application soit contenue dans des unités modulaires et fonctionnelles. Je créé donc 5 modules (share, core, admin, candidat et landing-page).  Une fois les modules créés, il est impératif de les déclarer dans les imports de l'app.module (à l’exception du SharedModule qui est, lui, importé directement dans chaque composant concerné) afin de les intégrer à l’ensemble de l’application. De surcroît, il est nécessaire de définir pour chaque module une adresse ou 'path', ainsi qu’une méthode de chargement.    1 - Déclaration des modules dans l'app.module  Pour respecter les bonnes pratiques d'Angular, les différents composants sont chargés en utilisant la technique du Lazy Loading. Angular générant une SPA (Single Page Application), l’intégralité de ses composants sont normalement chargés simultanément. Ceci peut entraîner le chargement de nombreuses bibliothèques ou modules superflus. Le Lazy Loading en Angular permet de contourner ce problème en ne chargeant les composants, modules, ou autres, que lorsqu’ils sont réellement nécessaires, optimisant ainsi la performance de l’application.    2 - Chargement des modules en Lazy Loading  Par la suite, chaque module doit intégrer un routing-module, qui sert à aiguiller vers le bon composant en fonction de la navigation de l'utilisateur. Aucun besoin de recourir de nouveau au Lazy Loading à ce niveau, celui-ci ayant déjà été appliqué au niveau des modules, assurant ainsi une gestion optimale des ressources et une navigation fluide et réactive pour l’utilisateur, qu’il soit un candidat ou un administrateur de l’application.    3 - Déclaration des routes de chaque composant d'un module La page de connexion Chaque profil doit s'identifier pour accéder à l'application web. L'administrateur, se connectera avec son nom d'utilisateur et son mot de passe, tandis que le candidat s'identifiera avec son nom et son prénom. Rappelons que l'application web est destinée, dans un premier temps, à une utilisation interne uniquement. Si l'application devait être déployée sur Internet pour administrer des tests à distance, le moyen d'authentification des candidats serait révisé pour renforcer la sécurité.  À cette fin, un formulaire de type ReactiveForm est employé pour transmettre les données via un service. De plus, nous écoutons la réponse de notre requête qui, en cas de succès, nous transmettra les données de l'utilisateur ainsi qu'un token et un refreshToken que nous enregistrons en session. Les informations de l'utilisateur, et plus spécifiquement son rôle – soit ADMIN, soit CANDIDAT – sont également enregistrées en session. Ces rôles seront utilisés par un guard d'Angular.    4 - Création des deux formulaires de connexion    5 - Envoi du formulaire de connexion et réaction suivant la réponse reçue  Les Route Guards d'Angular sont des interfaces qui, une fois implémentées, nous permettent de contrôler l'accès à une route selon les conditions spécifiées dans la classe implémentant l'interface. Angular propose différents types de guards tels que CanActivate, CanActivateChild, CanLoad, CanDeactivate et Resolve. Dans notre application, nous mettons en œuvre le CanActivate qui vérifie le rôle de l'utilisateur afin d'autoriser ou non l'accès à une page, garantissant ainsi que chaque utilisateur accède uniquement aux fonctionnalités qui lui sont pertinentes et autorisées.    6 - mise en place du Guard basé sur le rôle de l'utilisateur | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Angular, PrimeNg | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **1** | **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité** | | | |
| ***Exemple n°3***  | | Développer la partie back-end d’une interface utilisateur web | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Méthodologie de Création des Tokens  La génération des tokens se décompose en trois phases essentielles, orchestrées par la méthode createTokens. Cette méthode reçoit en paramètre un objet UserDto, pouvant être un AdminDto ou un CandidatDto, et procède à l'invocation de deux méthodes subalternes responsables de la création de l'access token et du refresh token.  Le refresh token produit est ensuite associé à l'instance de AdminDto ou CandidatDto correspondante et est persisté dans notre base de données, permettant une gestion optimisée des sessions utilisateurs.    7 - Création des tokens  Elaboration de l'Access Token  La méthode createAccessToken engendre l'access token, un composant crucial pour l'intégrité et la sécurisation de notre application. Ce token, doté d'une durée de vie d'une heure, encapsule des informations utilisateur telles que le nom et le prénom et est signé via un algorithme HMAC256, assurant ainsi l'authenticité et la confidentialité des informations transmises.    8 - Création de l'access Token  Conception du Refresh Token  En parallèle, la méthode createRefreshToken se charge de formuler un refresh token. Ce token, bien que disposant d'une durée de validité supérieure, s'échelonnant sur une journée, embarque moins d'informations relatives à l'utilisateur pour minimiser d'éventuelles expositions d'informations sensibles.    9 - Création d'un refresh token  Le rôle de ce refresh token est crucial, il permet de réduire les demandes répétitives d'authentification en facilitant la régénération de nouveaux access tokens, sans requérir de nouvelles saisies des identifiants de l'utilisateur. Comme vue précédemment avec l’intercepteur d’Angular, lorsque le token est périmé, l’API va renvoyer une erreur 401 ce qui va déclencher la création d’un nouveau token en utilisant le refresh token. La méthode refreshAccessToken prend donc en argument le refresh token, retrouve l’utilisateur associé à ce refresh token et génère un nouveau token.    10 - Création d'un nouveau token à partir d'un refresh token  Ces mécanismes de création de tokens, en conjugaison avec les méthodes d'authentification et de gestion de tokens, consolident la structure de sécurité de notre application. Ils garantissent une interaction authentifiée et sécurisée, tout en préservant une expérience utilisateur cohérente et sécurisée.  Mécanisme de vérification du token  Une fois le token généré, chaque requête provenant du côté client est interceptée par l'intercepteur d'Angular, lequel, à travers ledit intercepteur, va attacher le token dans l'en-tête de la requête. Cette requête est ensuite interceptée par la méthode doFilterInternal, pour une vérification minutieuse de la présence du token.    11- la méthode doFilterInternal pour vérifier la présence token  Si un token est effectivement présent, deux méthodes supplémentaires sont déployées pour valider ce token, en fonction du type de requête : une méthode spécifique pour les requêtes GET et une autre pour tous les autres types de requêtes.  Pour les requêtes de type GET, la vérification du token se centre principalement sur sa validité afin d'approuver la requête. En revanche, pour toutes les autres requêtes, une étape de vérification supplémentaire est nécessaire. Il est impératif de confirmer non seulement la validité du token, mais également l'existence de l'utilisateur en base de données avant d'approuver la requête. Cette stratégie permet d'assurer une sécurité renforcée et de prévenir d'éventuelles requêtes malveillantes ou non autorisées, garantissant ainsi l'intégrité et la fiabilité de l'application. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Spring Boot, Spring Security, JWT, PasswordEncoder | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Concevoir et développer la persistance des données en intégrant les recommandations de sécurité** | | | |
| ***Exemple n° 1***  | | Concevoir une base de données | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Concevoir une base de données  Avant de concevoir une base de données, il est crucial de conceptualiser et de visualiser le flux de fonctionnalités et d'interactions au sein du système. Pour ce faire, des outils tels que les diagrammes de séquence et d'activité sont indispensables.    Les diagrammes de séquence, illustrés ci-dessus pour un candidat, détaillent les interactions entre différents objets dans un système en mettant l'accent sur l'ordre des interactions. Ils offrent une vue claire et détaillée des fonctionnalités et des responsabilités de chaque composant, permettant ainsi de comprendre précisément le flux de processus et les interactions au sein du système.    D'autre part, les diagrammes d'activité, comme montré ci-dessus pour un candidat, mettent en lumière le flux de travail au sein du système. Ils illustrent les étapes et les activités réalisées par les utilisateurs ou les systèmes, les conditions et les décisions qui structurent le parcours de l'utilisateur.  En illustrant les étapes, les activités, et les décisions prises par un administrateur, ces diagrammes jouent un rôle crucial dans la transmission d'une compréhension unifiée des exigences et comportements souhaités de l'application entre les différentes parties prenantes du projet.  Ces diagrammes préparent le terrain pour la conception de la base de données en offrant une fondation solide pour élaborer une architecture cohérente et réaliser une implémentation sans faille.  Dans le cadre de ce projet, la conceptualisation et la réalisation de diagrammes, notamment les diagrammes de séquence et d'activité, étaient indispensables avant de s'avancer dans la conception de la base de données. Ils permettent de visualiser les interactions et le flux de travail entre les différents utilisateurs du système, qu'il s'agisse des candidats ou des administrateurs, et facilitent une compréhension holistique du comportement attendu et des exigences fonctionnelles de l'application. Les diagrammes d'activité, tels qu'illustrés pour le parcours du candidat, sont particulièrement utiles pour dépeindre de manière structurée les différents scénarios d'utilisation, les conditions et les décisions qui informent l'expérience utilisateur au sein du système.  En regardant à travers le prisme de ces diagrammes, les équipes projet peuvent acquérir une vision commune et unifiée des fonctionnalités requises et des interactions entre les divers composants du système, jetant ainsi les bases d'une conception de base de données rationnelle et cohérente.  Intégrer ces éléments préliminaires avec une structure de base de données bien pensée était donc crucial. Pour ce faire, après une série d'échanges approfondis avec l'équipe projet et une analyse détaillée du cahier des charges, j'ai opté pour l'utilisation de Looping pour construire cette base de données. Cela a été réalisé en dépit de l'absence de certaines fonctionnalités dans Looping, telles que l'intégration d'énumérations ou de classes abstraites, qui auraient pu optimiser davantage la représentation visuelle de la base de données (Annexe N°1).  La nécessité de gérer deux types d'utilisateurs, à savoir les administrateurs et les candidats, a engendré la création d'une classe abstraite, évitant ainsi la redondance et regroupant les champs communs entre ces deux entités distinctes. Les classes administrateur et candidat en héritent, promouvant la modularité et facilitant les modifications futures, tout en maintenant une cohérence structurelle.  En outre, pour satisfaire à des relations complexes, des tables de liaison ont été incorporées pour établir des relations « Many-to-Many » entre divers éléments comme les candidats et les propositions ainsi qu'entre les QCM et les questions, avec des champs supplémentaires pour assurer une flexibilité et une organisation optimales.  Ainsi, cette structure de base de données robuste et rationalisée sert de fondement à l'application, permettant des interactions fluides entre les différentes entités, et garantissant l'intégrité et la durabilité des données manipulées. En alignant rigoureusement la conception de la base de données sur les besoins définis par l'équipe et le cahier des charges, un environnement de données efficace et évolutif a été mis en place, prêt à supporter les évolutions et adaptations futures du projet. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Draw.io, Looping | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Concevoir et développer la persistance des données en intégrant les recommandations de sécurité** | | | |
| ***Exemple n° 2***  | | Mettre en place une base de données | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Dans le développement d'applications avec Spring Boot, la mise en place d'une base de données est une étape cruciale, et pour ce faire, l'utilisation de Java Persistence API (JPA) et Hibernate s'avère souvent indispensable. JPA est une interface de programmation standard qui sert de spécification pour la gestion relationnelle des données dans les applications Java. Elle permet de définir la manière dont les données doivent être persistées, gérées et récupérées, offrant une abstraction élevée par rapport à la base de données sous-jacente.  Hibernate, quant à lui, est l'un des fournisseurs de persistance les plus populaires pour JPA. Il implémente les spécifications de JPA et offre des fonctionnalités supplémentaires, tels que la gestion de cache de second niveau et des mécanismes de requêtage avancés, qui optimisent les performances de l'application.  En utilisant JPA et Hibernate dans un contexte Spring Boot, les développeurs peuvent bénéficier d'une intégration fluide et d'une configuration simplifiée. Cette combinaison permet de manipuler les entités de base de données de manière intuitive et orientée objet, en minimisant le besoin d'écrire des requêtes SQL explicites. Elle encourage ainsi le développement rapide, tout en garantissant la qualité et la robustesse de l'accès aux données dans l'application, rendant la gestion de la persistance des données plus flexible et moins sujette à erreurs.  La mise en place d'une architecture orientée objet efficace est cruciale dans un tel environnement. Une classe abstraite comme Personne a été définie dans notre cas pour représenter les éléments communs à différentes entités, les class Admin et Candidat qui sont en réalité des spécialisations de la classe Personne. Cette approche d'utilisation de classes abstraites et d'héritage, combinée à l'abstraction des données offerte par JPA et Hibernate, permet une représentation cohérente et logique du modèle de données.    Les classes Admin et Candidat, en étendant la classe Personne, héritent de ses attributs et méthodes, mais peuvent également déclarer des propriétés et des comportements qui leur sont propres. La stratégie d'héritage SINGLE\_TABLE implique que toutes les entités de cette hiérarchie sont représentées dans une seule table, et la colonne type\_personne sert à discriminer entre les différentes instances de sous-classes.    12 - class admin qui étend Personne class Candidat qui étend Personne  Pour interagir avec la base de données, des repositories JPA, tels que AdminRepository et CandidatRepository, sont définis. Ces repositories fournissent une interface pour effectuer des opérations CRUD sur les entités correspondantes et définissent des méthodes pour effectuer des requêtes sur la base de données selon divers critères.      L'utilisation de classes abstraites et de l'héritage, en harmonie avec JPA et Hibernate dans Spring Boot, permet de créer un modèle de données structuré et extensible. Cela facilite le développement et la maintenance du code tout en optimisant l'interaction avec la base de données grâce à une interface de repository intuitive, minimisant ainsi les erreurs potentielles et favorisant un développement plus rapide et plus robuste.  Pour établir des relations entre les entités dans le contexte de JPA et Hibernate, divers types de mappages relationnels sont mis à disposition pour refléter fidèlement les relations de base de données dans le modèle objet. Ces mappages facilitent l'interaction avec la base de données en gérant de manière transparente la persistance des données, les mises à jour et les requêtes, tout en préservant les principes de la modélisation objet. Voici deux des mappages relationnels les plus couramment utilisés : OneToMany.  OneToMany  La relation @OneToMany indique qu'une instance d'une entité est associée à plusieurs instances d'une autre entité. L'exemple de votre entité Question illustre une telle relation. La classe Question possède plusieurs instances de l'entité Proposition grâce à la propriété propositions, indiquant ainsi qu'une question peut avoir plusieurs propositions :    Dans ce cas, le mappedBy signifie que la relation est bidirectionnelle et que l'autre entité (Proposition) contient la clé étrangère grâce à la propriété question :    Souvent, dans le développement des modèles de données, les développeurs se confrontent à des situations où une relation ManyToMany devient nécessaire pour décrire adéquatement la relation entre deux entités. Toutefois, en pratique, implémenter directement une relation ManyToMany peut parfois introduire des complications, notamment lorsqu'il s'agit de représenter des informations supplémentaires concernant la relation elle-même. En conséquence, une approche courante consiste à "casser" la relation ManyToMany en introduisant une table intermédiaire, réduisant ainsi la complexité et offrant un mécanisme pour gérer de manière élégante des attributs supplémentaires de la relation.  Prenons l'exemple de votre entité QuestionQcm, qui sert précisément de cette table intermédiaire, assurant une liaison entre les entités Question et Qcm :    La présence de l'objet QuestionQcmId montre que la clé primaire de QuestionQcm est en fait une clé composée des identifiants des deux autres entités liées :    En procédant ainsi, la relation ManyToMany originale entre Question et Qcm est décomposée en deux relations ManyToOne dans QuestionQcm : une vers Question et une autre vers Qcm. Cette technique de décomposition non seulement permet de gérer plus aisément la persistance et la récupération des données avec JPA et Hibernate, mais elle ouvre également la voie pour ajouter d'autres attributs relatifs à la relation (comme ordre dans l'exemple), chose qui aurait été complexe voire impossible avec un mappage ManyToMany direct. Par conséquent, cette approche apporte une flexibilité et une scalabilité accrues, permettant aux développeurs de construire des applications robustes et maintenables. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Spring Boot, JPA, Hibernate, PostgreSQL | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Concevoir et développer la persistance des données en intégrant les recommandations de sécurité** | | | |
| ***Exemple n° 3***  | | Développer des composants dans le langage d’une base de données | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Dans le cadre de la gestion et de l'exploitation optimisée des données, j'ai mis en œuvre le langage SQL à travers plusieurs structures et fonctionnalités distinctes. J'ai débuté avec la création de vues afin de simplifier les interactions futures avec les données. Par exemple, pour la vue PRODUITSCATEGORIES, j'ai exécuté la commande SQL suivante :    Cela m'a permis de créer une représentation virtualisée des informations, facilitant ainsi l'extraction des données sans nécessiter de multiples jointures dans les requêtes futures.  Pour extraire et analyser les données de manière agglomérée, j'ai élaboré diverses requêtes, par exemple :    Ici, je voulais obtenir le chiffre d'affaires mensuel pour 2023, en segmentant les données mois par mois grâce à la fonction DATE\_FORMAT et en agrégeant les ventes via la fonction sum, organisées avec GROUP BY.  Afin d'automatiser certaines réponses aux modifications des données, j'ai mis en place des triggers. Pour créer une entrée dans l'historique à chaque fois qu'une commande est clôturée, j'ai configuré le trigger CreerHistorique :    Ce trigger scrute chaque mise à jour de commande et, lorsque le statut devient 'clos', il insère une nouvelle entrée dans la table HISTORIQUE, assurant ainsi un suivi automatisé des transactions.  Enfin, dans l'objectif d'optimiser l'espace de stockage et de gérer le cycle de vie des données, j'ai défini la procédure stockée SupprimerArchive :    Avec cette procédure, je peux identifier et effacer les données historiques dépassant une certaine ancienneté (ici, 10 ans), ce qui évite une accumulation inutile de données obsolètes.  A travers l'utilisation de vues, requêtes, triggers et procédures stockées, j'ai pu réaliser un système de gestion de base de données qui non seulement préserve et optimise les données, mais qui également automatise de nombreux aspects du traitement et de la gestion des données, attestant ainsi de la flexibilité et de l'efficacité du SQL dans le domaine des bases de données. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| phpMyAdmin | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *AFPA - Amiens* | |
| Chantier, atelier, service | | | Fil-rouge | | |
| Période d’exerciceDu : *22/02/2023*au :*17/07/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **3** | **Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité** | | | |
| ***Exemple n° 1***  | | Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Le déploiement réussi de projets informatiques s'articule souvent autour de l’application judicieuse de méthodologies agiles et de l’utilisation stratégique d’outils de collaboration et de gestion de projet. Au cours du développement de l'application web pour l'entreprise MIPIH, nous avons mis en œuvre une approche qui embrasse à la fois la flexibilité et la collaboration, s'appuyant sur divers instruments et techniques pour optimiser le processus de travail.  L'utilisation de la méthode Agile a été importante dans cette démarche, établissant un cadre opérationnel qui encourage une communication constante et une réactivité aux changements. Les stand-up quotidiens et les sessions de sprint planning, entre autres, ont constitué le aspect important de notre gestion de projet, garantissant de rester aligné sur les objectifs et les échéances définies.  Klaxoon s’est révélé être un outil précieux pour cultiver une dynamique collaborative, en facilitant le partage d’idées et en stimulant une participation active à travers les différentes phases du projet.    13 - exemple d'utilisation de klaxoon pour partager des idées  D'autre part, Rally nous a permis de garder une trace méticuleuse des user stories et d'assurer une planification de sprint rigoureuse, tout en offrant une transparence bienvenue sur le déroulement du projet à tous les niveaux.    Dans cet écosystème de collaboration, GitLab a également joué un rôle central, en offrant un espace où les membres de l'équipe peuvent contribuer et participer à divers aspects essentiellement avec des revue de code dans le cas de mon projet étant le seul développeur sur ce projet. Ce dernier point est essentiel pour assurer la qualité du code et pour permettre une amélioration continue, en tirant parti des connaissances et des compétences de l’ensemble de l’équipe. Grâce à GitLab, nous avons pu instaurer une culture de code review constructive et enrichissante, permettant non seulement de déceler d'éventuelles anomalies ou opportunités d'optimisation mais également de partager les connaissances et les meilleures pratiques au sein de l'équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Klaxxon, Agile, Rally | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **3** | **Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité** | | | |
| ***Exemple n°2***  | | Développer des composants métier | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Les tests unitaires, en testant les attentes fonctionnelles dans des assertions codifiées, forge un bouclier résilient contre les régressions logicielles et les anomalies, tout au long des phases de développement, de maintenance et de refactorisation des applications. Prenons, par exemple, les tests suivants :    Dans ce test, nous validons la création de tokens pour un administrateur, s'assurant que les tokens d'accès et de rafraîchissement sont non nuls après la génération. Cela confirme que notre logique de gestion des tokens est solide et produit des résultats attendus pour des scénarios courants.    En revanche, ce dernier exemple illustre le test d'un cas d'erreur où une exception est déclenchée pour un utilisateur inconnu. En mettant intentionnellement notre système à l'épreuve avec des scénarios d'erreur, nous nous assurons que notre code est robuste et gère les erreurs de manière prévisible, offrant une sécurité supplémentaire pendant les phases de développement ultérieures et lors de déploiements en production.  Les tests unitaires, tels que ceux illustrés, deviennent une carte précise des comportements attendus et des limites du système, fournissant une base stable sur laquelle le code peut évoluer et s'adapter aux besoins changeants. Ils offrent également une trajectoire de correction claire lors de la découverte d'erreurs, assurant que les corrections et les améliorations futures sont construites sur une fondation de logique vérifiée et validée. En couvrant à la fois les chemins d'exécution réussis et les cas d'échec anticipés, nous enveloppons notre code dans un filet de sécurité, minimisant le risque de déploiements défectueux et de maintenance pénible à long terme.  SonarLint occupe une place essentielle dans l'écosystème des outils de développement en offrant une solution proactive pour maintenir et améliorer la qualité du code dès le stade du développement. Intégrée directement dans l'IDE (Environnement de Développement Intégré), SonarLint opère en tant que vigilant silencieux, scrutant le code à la volée et mettant en évidence les anomalies, les bugs potentiels, et les infractions aux conventions de codage avant même qu'ils ne soient commis dans le dépôt du code source. L'utilité de SonarLint ne se limite pas à la simple identification des erreurs de syntaxe ou des bugs, mais s'étend à la promotion des bonnes pratiques de codage et à la prévention des anti-modèles, aidant ainsi les développeurs à adhérer aux standards de qualité et à produire un code plus propre, plus sûr et plus maintenable. L'intégration avec l'IDE signifie que les développeurs reçoivent une rétroaction instantanée, les alertant sur les problèmes potentiels au moment même de la frappe, permettant une résolution immédiate et éduquant sur les meilleures pratiques au fil du temps. Les équipes de développement, par conséquent, peuvent garantir une qualité de code cohérente et élevée tout au long du cycle de vie du projet, réduisant ainsi le coût et l'effort de la détection et de la correction des bugs dans les phases ultérieures. En associant SonarLint avec un pipeline d’intégration continue via SonarQube ou d'autres plateformes de contrôle qualité, les organisations créent un bouclier robuste contre la dégradation de la qualité du code et garantissent que les livrables respectent les critères de qualité établis, depuis le développement jusqu'à la production. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Spring Boot, Sonar Lint, Junit | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **3** | **Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité** | | | |
| ***Exemple n°3***  | | Construire une application organisée en couche | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Modèle-Vue-Contrôleur Spring Boot  Pour respecter les bonnes pratiques d'une API RESTful, j'ai structuré l'application en suivant le modèle MVC (Modèle-Vue-Contrôleur). Le modèle MVC est un patron de conception qui sépare une application en trois composants interconnectés :  **Modèle** : Gère la logique métier, l'accès et la manipulation des données.  **Vue** : Présente les données à l'utilisateur sous une forme appropriée.  **Contrôleur** : Interagit à la fois avec le modèle et la vue, recevant les entrées de l'utilisateur et effectuant les appels correspondants au modèle.  Dans ce contexte, différents packages ont été créés pour isoler les responsabilités, à savoir :  Controllers : Ce package est lié, comme son nom l’indique à la composante Contrôleur du MVC. Il contient les classes qui reçoivent et traitent les requêtes HTTP, délèguent la logique métier et renvoient les réponses HTTP.  DTOs (Data Transfer Objects) : Ce sont, comme dans la partie Angular, des objets qui transportent des données entre les processus. Ils sont généralement utilisés pour regrouper les attributs que vous voulez partager avec l'interface utilisateur ou l'API externe. De plus, ils permettent d’appliquer des logique métier et être donc diffèrent des modèles.  Mappers : Ils sont responsables de la conversion des objets, comme la transformation des entités en DTOs et inversement, afin de garantir l'indépendance entre les modèles de données internes et la représentation externe des données.  Models : Ils correspondent au composant Modèle dans le pattern MVC. Ils représentent les structures de données de l'application et contiennent la logique pour accéder à la base de données.  Repositories : Ils sont utilisés pour gérer la communication entre l'application et la base de données. Ils jouent un rôle crucial dans la manipulation des données du Modèle.  Services : Ils contiennent la logique métier et les règles de l'application. Les services interagissent avec les modèles et les repositories pour traiter les données et renvoyer les résultats au contrôleur.  Chacun de ces packages a une fonction distincte, mais tous interagissent de manière cohérente pour former une application structurée, conformément aux principes du modèle MVC, offrant ainsi une base solide pour le développement d'une API RESTful.  Modèle-Vue-Contrôleur Angular  Dans le cadre d’Angular, un framework structuré et modulaire, l'organisation en couches est essentielle pour édifier des applications robustes et facilement maintenables.  Les composants prennent en charge la Vue, en gérant l’affichage des données et l'interaction avec l'utilisateur. Ils fonctionnent comme la couche de présentation dans l'architecture MVC, offrant une interface utilisateur intuitive et réactive.  Les services, par contraste, se rapprochent plus de la couche Service dans le backend, abritant la logique métier et orchestrant les appels HTTP.  Particulièrement, Angular présente un module spécifique nommé Core, où les services à portée globale, qui sont sollicités à travers toute l’application, sont définis. Ces services, souvent chargés des aspects transversaux tels que l'authentification, le stockage de données globales, et la gestion d’état, sont injectés là où ils sont requis, assurant ainsi une cohérence et une centralisation de la logique fonctionnelle cruciale.  Les modèles en TypeScript supervisent les structures de données, garantissant une gestion des données sûre et prévisible, et assurant la cohérence du type de données à travers l'application.  D'autre part, les directives et les pipes offrent des moyens astucieux de manipuler et de transformer les données directement dans la vue, en ajoutant une couche supplémentaire de logique de présentation.  Finalement, les modules d'Angular permettent de regrouper et de segmenter logiquement les fonctionnalités, facilitant ainsi une séparation nette des préoccupations et une organisation en couches soigneuse.  En somme, Angular ne fournit pas seulement un framework robuste pour le développement frontend, mais sert également de boussole pour élaborer une architecture méticuleusement structurée et bien organisée, tout en facilitant le développement et la maintenance d'applications complexes. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Spring Boot, Angular | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| Mentoré par le référent technique de l’équipe | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Mipih* | |
| Chantier, atelier, service | | | DO-GRH | | |
| Période d’exerciceDu : *17/07/2023*au :*06/10/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

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

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

Pierre Pac

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.

Pont-Rémy

03/10/2023

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** |
|  |
| Annexe 1 : Schéma base de données |