Université de Rouen
UFR Sciences et techniques
Département informatique

Projet XML

Membres :
Zahra CHAHI
Kenza DJELLAOUI

Encadré par :
M. Etienne REITH

Master 1 GIL Promotion : 2023/2024

TABLE DES MATIÈRES

TABLE DES FIGURES

1 Introduction

Dans le cadre de ce projet, il nous est demandé de développer une application de gestion de CV, en prolongement de nos travaux pratiques précédents. Pour ce faire, nous avons poursuivi les travaux entamés dans le TP précédent. Ce rapport débutera par une explication détaillée de l'objectif de l'application, suivie d'une description des étapes et des méthodologies employées pour son développement.

2 Spécification des besoins

Pour commencer, nous avons décidé de nommer l'application "JobFounder" en raison de sa vocation principale qui est la gestion de CV. Ce choix de nom reflète l'objectif de l'application d'aider les utilisateurs à trouver des opportunités professionnelles en mettant en avant leurs compétences et expériences de manière optimale.

FIGURE 1 – Logo JobFounder.

JobFounder offre plusieurs fonctionnalités essentielles pour la gestion des CV :

— **Ajout de CV** : Les utilisateurs peuvent facilement ajouter leurs CV à la base de données de l'application. Chaque CV peut être personnalisé avec des informations détaillées telles que l'expérience professionnelle, la formation, et les compétences.

— **Affichage des CV disponibles** : L'application permet de visualiser tous les CV stockés dans la base de données, offrant ainsi un accès rapide et facile aux documents disponibles.

— **Affichage détaillé d'un CV** : En attribuant un identifiant unique (ID) à chaque CV, JobFounder permet d'afficher les détails complets de n'importe quel CV stocké. Cette fonctionnalité facilite la consultation des informations spécifiques à chaque document.

— **Suppression de CV** : Les utilisateurs peuvent également supprimer des CV de la base de données en utilisant l'identifiant unique associé à chaque document, assurant ainsi une gestion efficace et sécurisée des informations.

JobFounder utilise des flux XML pour le transfert et la manipulation des données des CV. Chaque flux XML est validé par un fichier XSD (XML Schema Definition) afin de garantir l'intégrité et la cohérence des données.

3 Technologies utilisées

Dans cette section, nous allons vous présenter les technologies que nous avons employées lors du développement.

3.1 Technologies de développement

— **Sprint Boot** : Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en JAR, totalement autonome.

FIGURE 2 – Logo Spring boot.

— **PostgreSQL** : est un système de gestion de base de données relationnelle open source (SGBDR) extrêmement puissant et robuste. Conçu pour gérer de grandes charges de travail et des volumes importants de données, PostgreSQL offre de nombreuses fonctionnalités avancées tout en restant conforme aux normes SQL.

FIGURE 3 – Logo PostgreSQL.

3.2 Technologies de déploiement

— **Docker** : est une plateforme open source qui permet de développer, déployer et exécuter des applications dans des conteneurs logiciels.

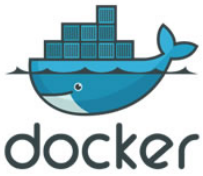

FIGURE 4 – Logo Docker.

— **CleverCloud** : est une plateforme d'hébergement cloud qui offre des services de déploiement, de gestion et d'évolutivité d'applications. Elle permet aux développeurs de déployer leurs applications web, mobiles et IoT avec facilité, en se concentrant sur le développement plutôt que sur la gestion des infrastructures.

FIGURE 5 – Logo CleverCloud.

3.3 Technologies de gestion et de transformation des données

— **XML** : est un langage de balisage conçu pour stocker et transporter des données de manière lisible tant par les humains que par les machines. Il est utilisé pour structurer et décrire des informations dans un format textuel qui peut être facilement analysé et manipulé par les applications logicielles.

— **XSD** : un langage de définition de schéma utilisé pour définir la structure, le contenu et les contraintes d'un document XML. Un fichier XSD définit la structure attendue d'un document XML, spécifiant les éléments autorisés, leurs attributs, leur séquence, leurs types de données, et d'autres règles de validation.

— **XSLT** : est un langage de transformation utilisé pour convertir un document XML en un autre format, généralement un autre document XML, HTML, ou texte. Il permet de définir des règles de transformation pour extraire, filtrer, trier, et formater des données XML selon des critères spécifiés.

3.4 Dépendances utilisées

— **spring-boot-starter-data-jpa** : cette dépendance fournit les fonctionnalités de base pour utiliser JPA (Java Persistence API) avec Spring Boot. Elle configure automatiquement les bean EntityManagerFactory, DataSource, et TransactionManager nécessaires pour l'accès aux données basées sur JPA.

— **jackson-dataformat-xml** : cette dépendance permet à l'application d'utiliser Jackson pour la sérialisation et la désérialisation de données au format XML. Jackson est une bibliothèque populaire de manipulation de données JSON, mais il prend également en charge d'autres formats de données, y compris XML.

— **jackson-databind** : cette dépendance fait partie de la suite Jackson et fournit les fonctionnalités principales de sérialisation et de désérialisation d'objets Java en JSON et d'autres formats de données.

— **spring-boot-starter-web** : Cette dépendance fournit les fonctionnalités de base pour développer des applications web avec Spring Boot, notamment la prise en charge de Spring MVC et du serveur d'application intégré.

— **spring-boot-starter-thymeleaf** : cette dépendance fournit le support de Thymeleaf, un moteur de modèle de serveur permettant de créer des vues HTML de manière élégante et efficace dans les applications Spring Boot.

4 Architecture de JobFounder

La figure illustre l'architecture de l'application, et cette section en détaille le fonctionnement.

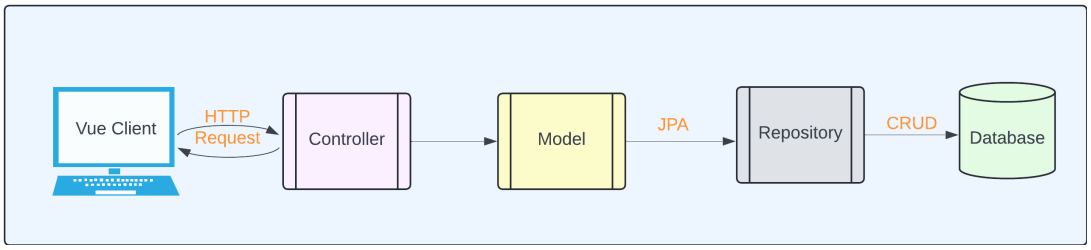

FIGURE 6 – Architetecture de l'application.

— **Client** : Envoie des requêtes directement au contrôleur et reçoit des réponses.

— **Controller** : Reçoit les demandes du client, effectue la logique métier requise, interagit avec le repository pour accéder ou modifier les données, puis renvoie la réponse au client.

— **Repository** : Assure la communication directe avec la base de données et inclut des méthodes pour réaliser des opérations CRUD sur les données. Fonctionne comme une couche d'abstraction entre la base de données et le contrôleur.

— **Model** : Représente les objets de données de l'application.Ces objets sont utilisés pour transférer des données entre les différentes couches de l'application.En utilisant JPA (Java Persistence API), les modèles peuvent être mappés à des tables de base de données.

— **Base de données** : Stocke toutes les données de l'application.Les requêtes CRUD sont exécutées sur cette couche pour gérer les données. Les opérations JPA permettent de manipuler les données en utilisant des entités de modèle.

4.1 Fonctionnement de JobFounder

Flux de requête et de réponse

— Le client envoie une requête au contrôleur.

— Le contrôleur analyse la requête et exécute directement la logique métier nécessaire.

— Pour récupérer ou modifier des données, le contrôleur appelle directement les méthodes de le repository.

— Le repository exécute les opérations de données nécessaires en interagissant avec la base de données.

— Les résultats sont renvoyés au contrôleur, qui construit une réponse et l'envoie au client.

Flux de données

Les modèles sont utilisés pour représenter les données transférées entre le contrôleur et la couche de repository. Les opérations JPA permettent de convertir ces modèles en entités de base de données et de gérer les transactions de données.

5 Conception

Dans cette section, nous allons présenté un diagramme de classe qui va en quelque sorte expliqué comment la base de données est gérée.

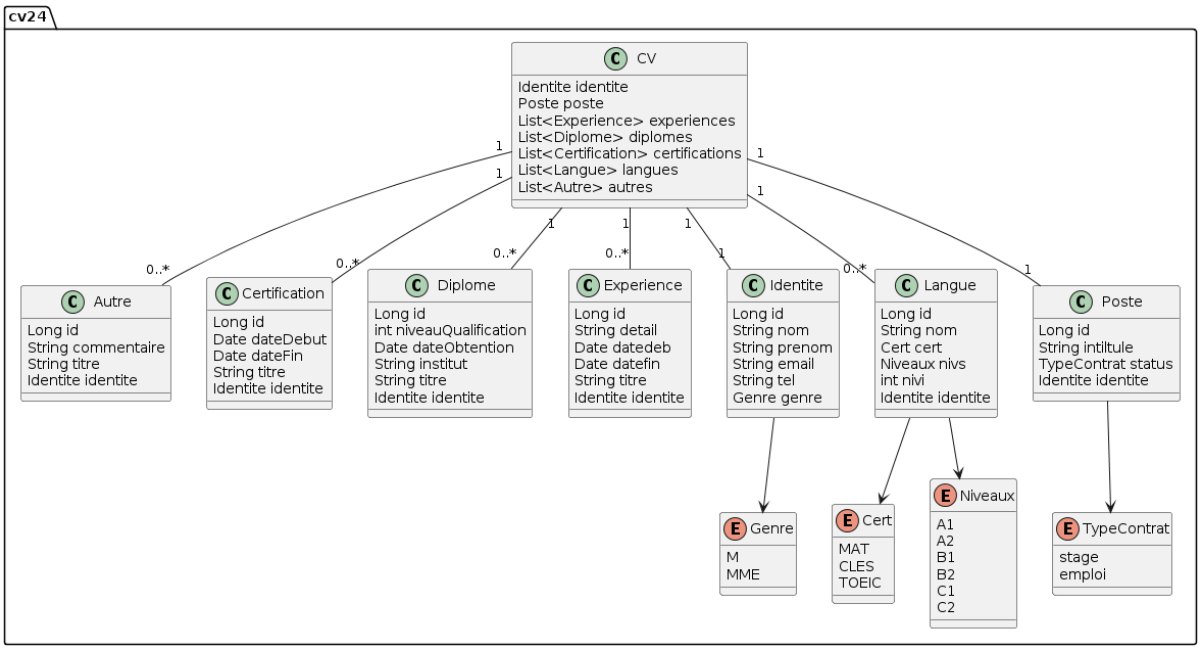

FIGURE 7 – Diagramme de classe.

6 Implémentation

Pour implémenter notre applications nous avons opté pour les technologies définies précédèment pour les raisons suivantes :

— **PostresQL** : Utiliser PostgreSQL pour cette application présente de nombreux avantages, surtout lorsqu'elle est déployée sur une plateforme comme Clever Cloud. Tout d'abord, PostgreSQL est reconnu pour sa fiabilité et sa stabilité, ce qui en fait un choix idéal pour les applications critiques telles que la gestion de CV. Sa capacité à gérer de grandes quantités de données tout en maintenant des performances élevées garantit une expérience utilisateur fluide. De plus, PostgreSQL est conforme aux normes SQL, ce qui simplifie l'intégration avec d'autres systèmes et facilite les migrations de données. Grâce à ses fonctionnalités avancées d'évolutivité et de sécurité, PostgreSQL offre une solution adaptable et sécurisée pour faire face aux besoins croissants de l'application et protéger les données sensibles des utilisateurs. Enfin, l'intégration native de Clever Cloud avec PostgreSQL simplifie le déploiement, la gestion et la mise à l'échelle de la base de données, offrant ainsi une solution complète et fiable pour l'application de gestion de CV.

— **XSD** : Nous avons utilisé XSD pour valider la conception d'un CV, ce qui garantira la création d'un CV de qualité. L'utilisation de XSD présente plusieurs avantages : tout d'abord, elle permet une validation structurée des données, assurant ainsi la cohérence et la conformité des informations incluses. De plus, cette approche permet une détection précoce des erreurs de format ou de contenu, facilitant ainsi leur correction rapide dès la phase de création du CV. En respectant un schéma XSD standard, le CV généré sera également interopérable avec d'autres systèmes ou applications qui utilisent également ce schéma, favorisant ainsi l'échange et la réutilisation des données. Enfin, la séparation de la structure du CV de son contenu offre une meilleure maintenabilité, permettant des mises à jour ou des modifications ultérieures du schéma XSD sans affecter les données existantes, ce qui facilite la maintenance à long terme du système.

— **XSLT** : Nous avons également intégré XSLT dans notre processus, permettant ainsi la transformation des flux XML en flux HTML. Cette approche offre plusieurs avantages : tout d'abord, elle permet de convertir les données structurées du CV au format XML en une présentation HTML conviviale et esthétique. De plus, en utilisant XSLT, nous pouvons appliquer des styles et des mises en forme personnalisés au contenu du CV, offrant ainsi une expérience utilisateur optimisée. Enfin, cette transformation XML vers HTML permet une flexibilité et une adaptabilité accrues dans l'affichage des informations du CV sur différents appareils et plateformes.

Maintenant, nous allons expliquer et présenter les classes que nous avons utilisés ainsi que le role de chacune d'entre elle.

6.1 Package Controller

— **CvController** : Cette classe est un contrôleur Spring MVC qui gère les requêtes HTTP relatives aux CVs. Elle fournit plusieurs méthodes pour afficher les CVs en HTML et en XML, et pour afficher les détails d'un CV spécifique.

— **CvRestController** : Cette classe est un contrôleur REST qui gère les requêtes HTTP pour les opérations CRUD (Create, Read, Update, Delete) relatives aux CVs. Elle inclut des méthodes pour insérer, supprimer, et rechercher des CVs.

— **HelpController** : cette classe est un contrôleur Spring MVC. Elle est capable de traiter les requêtes web et de renvoyer des vues.

— **IndexController** :est un contrôleur Spring MVC destiné à gérer les requêtes à la racine de votre application ("/") et à fournir des informations à afficher sur la page d'accueil.

— **ParserXML** : contient plusieurs méthodes qui traitent des fichiers XML que nous pourrions décrire ici :

  — **Parsage XML vers Objets Java (parseXML)** : Cette méthode lit un fichier XML et convertit les données en un objet CV, avec des sous-objets pour les expériences, diplômes, certifications, langues et autres informations.

  — **Conversion d'une Liste de CV en XML (parseDataToXML)** : Cette méthode génère un fichier XML à partir d'une liste d'objets CV.

  — **Conversion d'un CV en XML (parseDataCVToXML)** : Similaire à la méthode précédente, mais pour un seul objet CV.

  — **Génération de Messages d'Erreur en XML (generateErrorXML)** : Cette méthode crée un message d'erreur en format XML.

  — **Transformation XML en HTML avec XSLT (genereateHTMLWithXSLT)** : Cette méthode utilise un fichier XSLT pour transformer un flux XML en HTML.

6.2 Package entities

— **Autre** : représente une entité JPA (Java Persistence API) utilisée pour mapper les données de la table autre dans une base de données relationnelle.

— **Cert** : est une énumération (enum) en Java qui représente différents types de certifications linguistiques.

— **Certification** : représente une entité JPA pour les certifications dans une application de gestion de CV.

— **CV** : représente un CV dans notre application, en utilisant les annotations de Jackson pour la sérialisation et la désérialisation en XML.

— **Diplome** : représente un diplôme dans une application Java, avec des annotations JPA pour la persistance des données en base de données.

— **Experience** : représente une expérience professionnelle dans une application Java, avec des annotations JPA pour la persistance des données en base de données.

— **Genre** : est une énumération qui représente les genres possibles pour les titres de civilité.

— **Identite** : représente les informations personnelles d'une personne, telles que son nom, prénom, email, numéro de téléphone et genre.

— **Langue** : représente les compétences linguistiques d'une personne, notamment le nom de la langue, le niveau de certification, le niveau de compétence et le niveau de compréhension.

— **Niveaux** est une énumération qui représente les différents niveaux de compétence linguistique, allant de A1 à C2.

— **Poste** une entité associée à un poste occupé par une personne dans le contexte d'un CV.

— **TypeContrat** définit les différents types de contrats associés à un poste dans un cv.

7 Conclusion

En conclusion, notre projet de déploiement d'un service REST efficace pour gérer les CV conformes au standard CV24 semble être une réussite. Avec JobFounder, nous sommes désormais capables d'ajouter, d'afficher, de supprimer et de rechercher des CV de manière efficiente.

URL GITHUB

https ://github.com/Kenza3699/cv24v1