Université de Rouen
UFR de Sciences et Technique
Master 1 Informatique

Rapport Projet Langage Web 2 - XML

Service REST .

Réalisé par :

KHICHA Ines
TIRECHE Amira

23 mai 2024

Table des matières

Table des figures

1 Introduction

L'objectif de cette étape est de déployer un service REST pour gérer les documents cv24, conformément à la description du TP n°1, et de valider son bon fonctionnement à l'aide de tests Postman. Ce projet vise à finaliser un service RESTful fonctionnel et conforme aux exigences du cahier des charges, démontrant ainsi notre capacité à concevoir, implémenter et tester un service de manière professionnelle et fiable.

2 Architecture général

On a commencé la réalisation de ce projet en faisant l'architecture ci-dessous,

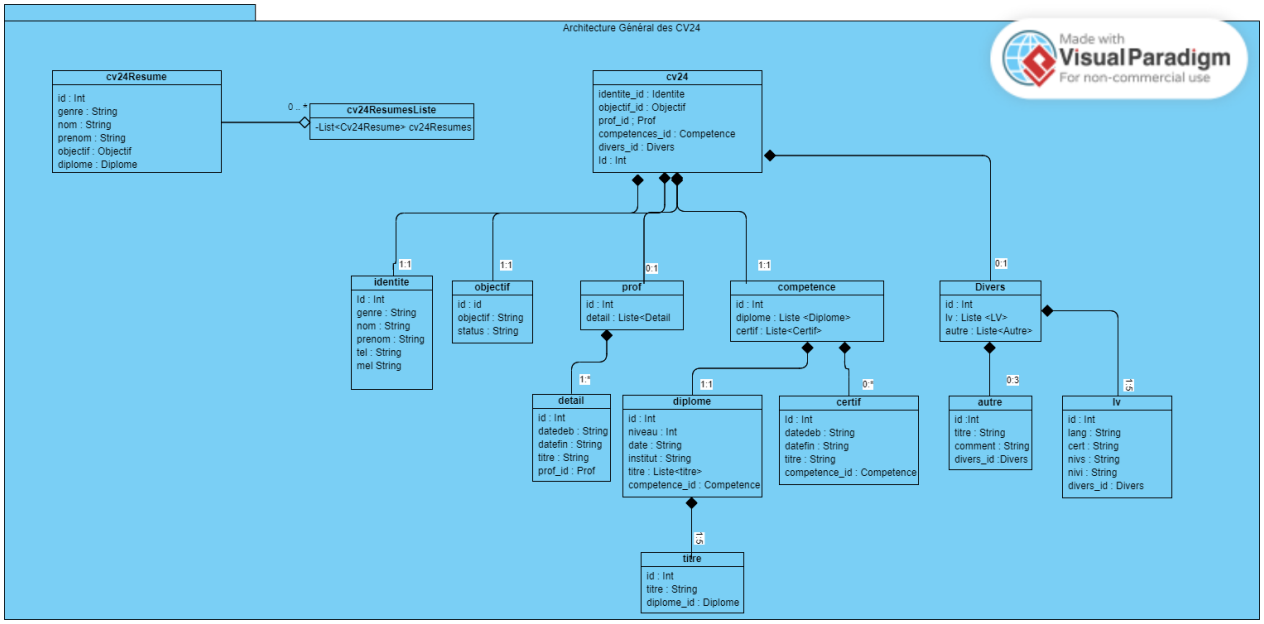

FIGURE 1 – Architecture globale.

3 Choix de la base de données

La sélection de PostgreSQL comme base de données pour ce projet s'est basée sur plusieurs critères. Tout d'abord, la disponibilité sur Clever Cloud a été un facteur clé. Après avoir évalué les options, une approche relationnelle a été préférée à une approche orientée objet. En effet, les bases de données relationnelles offrent une structure de données organisée, facilitant la gestion et la manipulation des informations, notamment dans le contexte des documents cv24. De plus, elles permettent de définir des contraintes d'intégrité et des relations entre les entités de manière explicite, garantissant ainsi la cohérence des données. Le choix entre MySQL et PostgreSQL s'est finalement porté sur PostgreSQL pour ses performances supérieures et sa fiabilité accrue, assurant ainsi la robustesse et l'efficacité du service REST pour la gestion des documents cv24.

3.1 Modification de la base de données

Pour modifier la base de données, vous pouvez utiliser une autre base de données PostgreSQL en ajustant les paramètres dans le fichier `application.properties`. Ce fichier se trouve dans le répertoire suivant :

`src/main/java/fr/univrouen/cv24/resources`

Les paramètres à modifier incluent l'URL de la base de données, le nom d'utilisateur et le mot de passe. Ces ajustements vous permettront de connecter l'application à une instance différente de PostgreSQL, selon vos

besoins spécifiques.

En cas de suppression de la base de donnée, il vous sera possible de la créer à nouveau avec les lignes de codes commentées suivantes :

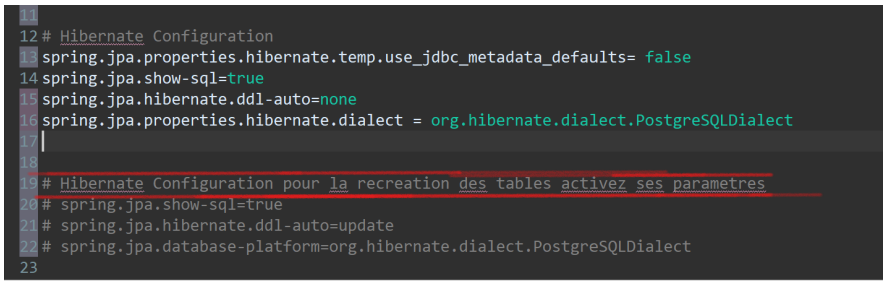

FIGURE 2 – Commandes de création de base de données.

4 Organisation du projet

4.1 Model

Dans notre projet, le package modèle renferme nos classes modèle qui ont été créées à l'aide de JPA (Java Persistence API) et d'Entity. Ces classes sont essentielles pour la représentation et la manipulation des données dans notre application. Grâce à JPA, nous avons pu définir des entités qui correspondent à des objets métier dans notre système. Chaque entité est associée à une table dans la base de données, ce qui nous permet de persister et de récupérer les données de manière transparente.

En utilisant Entity, nous avons annoté nos classes avec des métadonnées qui définissent la manière dont elles sont mappées sur la base de données. Ces annotations comprennent des informations telles que le nom de la table, les colonnes correspondantes, les clés primaires et étrangères, ainsi que les relations entre les entités.

Voici l'ensemble de nos classes modeles :

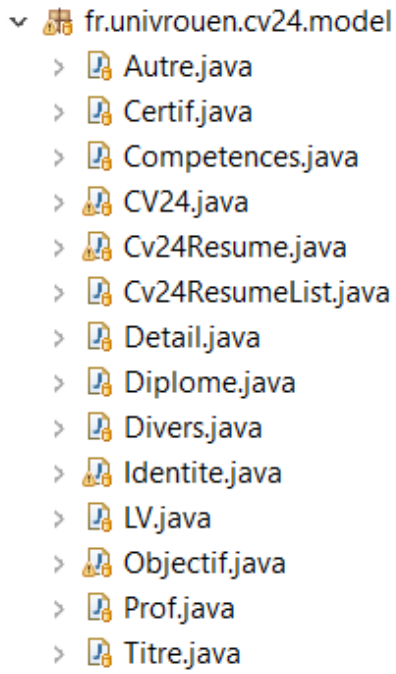

FIGURE 3 – Model.

4.2 controlleurs

4.2.1 IndexController

Ce contrôleur gère les requêtes GET sur la racine de l'application ("/"). Il renvoie la vue "index", qui est la page d'accueil de notre application.

4.2.2 DeleteController

Ce contrôleur gère les requêtes DELETE sur "/cv24/delete". Il prend en charge la suppression d'un CV en fonction de son ID. En cas de succès, il renvoie un message XML indiquant que le CV a été supprimé avec succès.

4.2.3 GetController

Ce contrôleur gère plusieurs types de requêtes GET :

- "/cv24/resume" : récupère une liste résumée de tous les CVs.
- "/cv24/xml" : récupère un CV spécifique au format XML.
- "/cv24/html" : récupère un CV complet au format HTML.
- "/cv24/resume/xml" : récupère une liste résumée de tous les CVs au format XML.
- "/cv24/search" : recherche des CVs en fonction de certains critères (date, objectif).

Il utilise des services pour récupérer les données nécessaires et renvoie les réponses appropriées sous forme de vues ou de contenu XML.

4.2.4 PostController

Ce contrôleur gère les requêtes POST sur "/cv24/insert". Il prend en charge l'insertion d'un nouveau CV à partir d'un flux XML. Après l'insertion réussie, il renvoie un message XML indiquant que le CV a été inséré avec succès.

4.3 Repository

Les repositories jouent un rôle central dans la persistance des entités dans la base de données de notre application. Voici les points importants à retenir :

- **Rôle des Repositories** : Les repositories permettent d'effectuer des opérations CRUD (Create, Read, Update, Delete) sur les entités associées.
- **Utilisation de JpaRepository** : Nous utilisons JpaRepository, une interface fournie par Spring Data JPA, pour simplifier l'accès aux données. Cette interface étend CrudRepository et fournit des méthodes supplémentaires pour les opérations CRUD.
- **Annotations** : Les repositories sont annotés avec `@Repository`, ce qui indique à Spring qu'ils doivent être pris en charge pour l'injection de dépendances et la gestion des transactions.
- **Génériques** : Les repositories prennent deux types génériques : le type de l'entité associée et le type de l'ID de cette entité. Par exemple, `JpaRepository<Prof, Long>` indique que le repository concerne l'entité Prof avec un ID de type Long.

4.4 Service

Les packages de services constituent un élément crucial de l'architecture logicielle de notre application. Ils encapsulent la logique métier, ce qui implique qu'ils sont responsables de la manipulation des données et de l'application des règles métier. Conçus pour être indépendants de la couche de présentation, les services permettent une réutilisation efficace de la logique métier dans différentes parties de l'application. Ils interagissent étroitement avec les repositories pour accéder aux données persistantes, utilisant les méthodes fournies pour effectuer des opérations CRUD sur les entités.

4.5 Exceptions

Le package Exception contient des classes qui représentent des exceptions spécifiques à notre application. Voici les principales classes et leurs rôles :

- **CVNotFoundException** : Cette classe étend `RuntimeException` et est utilisée pour signaler qu'un CV n'a pas été trouvé dans la base de données.
- **DuplicateCVException** : Cette classe étend `RuntimeException` et est utilisée pour signaler qu'une tentative d'insertion d'un CV en double a été détectée.

La classe **GlobalExceptionHandler** est un contrôleur de conseils globaux (`@ControllerAdvice`) qui gère les exceptions globalement pour toute l'application. Voici les principaux points à retenir :

- **Conversion en XML** : La méthode `convertToXml` convertit un objet `ErrorResponse` en XML en utilisant JAXB pour la sérialisation.
- **Gestion des Exceptions** : La classe contient des méthodes annotées avec `@ExceptionHandler` pour gérer différents types d'exceptions, comme `CVNotFoundException`, `JAXBException`, `IOException`, etc.
- **Réponses HTTP** : En fonction du type d'exception, des réponses HTTP appropriées sont renvoyées avec le contenu XML correspondant à l'erreur.

4.6 Validator

Le but de la classe XMLValidator est de valider un document XML par rapport à un schéma XML (XSD). Elle charge le schéma XSD à partir d'un fichier et utilise un validateur (fait durant le TP1) pour vérifier que le contenu du document XML respecte les règles définies dans le schéma.

5 Front end

Dans notre application, les pages HTML sont stockées dans le dossier `src/main/resources/templates`. Chaque page HTML est conçue pour être moderne et entièrement responsive, assurant ainsi une expérience utilisateur optimale sur tous les types de dispositifs, des ordinateurs de bureau aux smartphones.

Un exemple typique de nos pages HTML est la page de détails d'un CV. Cette page utilise Thymeleaf comme moteur de template pour intégrer dynamiquement les données du CV dans le HTML. Les sections de la page sont clairement définies pour afficher différentes parties du CV, telles que l'identité, les objectifs, les expériences professionnelles, les compétences, et divers autres détails. Les données sont insérées dans les balises HTML à l'aide des expressions Thymeleaf.

Les styles CSS associés à ces pages HTML sont stockés dans le répertoire `src/main/resources/css`. Par exemple, le fichier `styleCvDetails.css` est utilisé pour styliser la page de détails du CV. Les styles sont conçus pour être modernes et responsives, ce qui signifie qu'ils s'adaptent automatiquement à la taille et à la résolution de l'écran de l'utilisateur, offrant ainsi une mise en page cohérente et attrayante sur tous les appareils.

6 Accès au projet

6.1 Accès à la version local

Pour utiliser le projet en local, veuillez suivre les étapes suivantes :

- **Récupération du projet** : Utilisez la commande suivante pour cloner le dépôt :

```
git clone https://github.com/ines-web/cv24v1
```

- **Installation des dépendances** : Exécutez la commande suivante pour nettoyer et installer les dépendances du projet :

```
mvn clean install
```

- **Lancement de l'application** : Utilisez la commande suivante pour démarrer l'application :

```
mvn spring-boot:run
```

6.2 Accès à la version déployée

Pour utiliser la version déployée du projet, veuillez suivre le lien suivant :

`https://cv24-khicha.cleverapps.io/`

7 Test

Vous trouverez la série de tests Postman dans le fichier json qui se situe dans src/main/resources/test-Postman.

Voici quelques exemples d'exécution de tests :

Exemple d'exécution en cas de succès de l'end point insertion

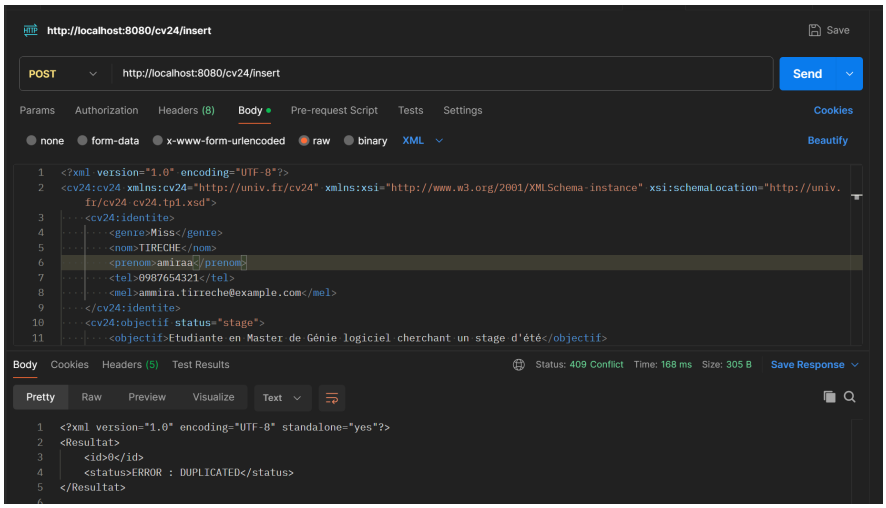

FIGURE 4 – Insertion cas succès.

Exemple d'exécution en cas d'échec de l'end point insertion.

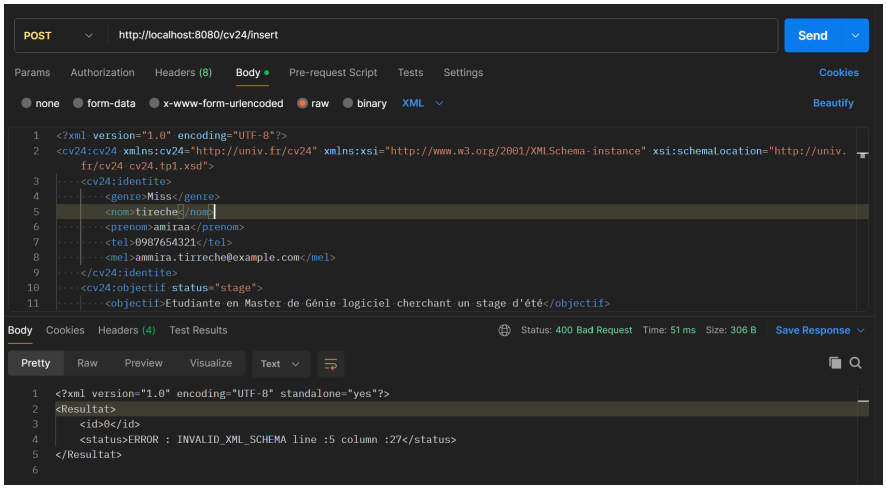

FIGURE 5 – Insertion cas échec.

Recherche d'un CV qui n'existe pas.

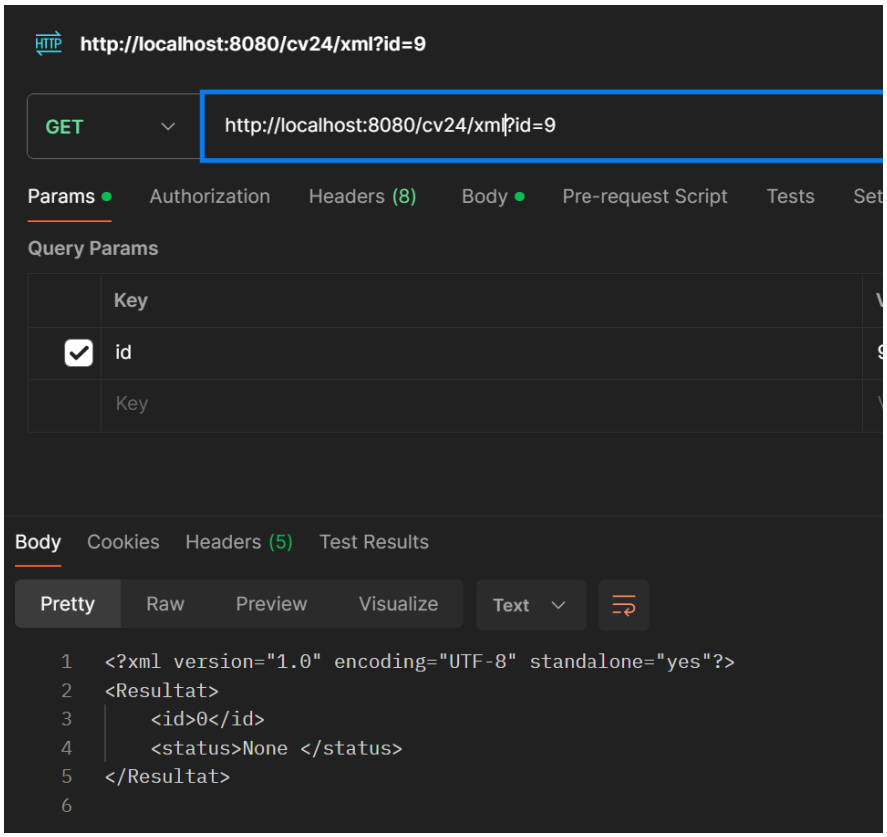

FIGURE 6 – Recherche d'un CV qui n'existe pas.

Suppression d'un CV en cas de succès.

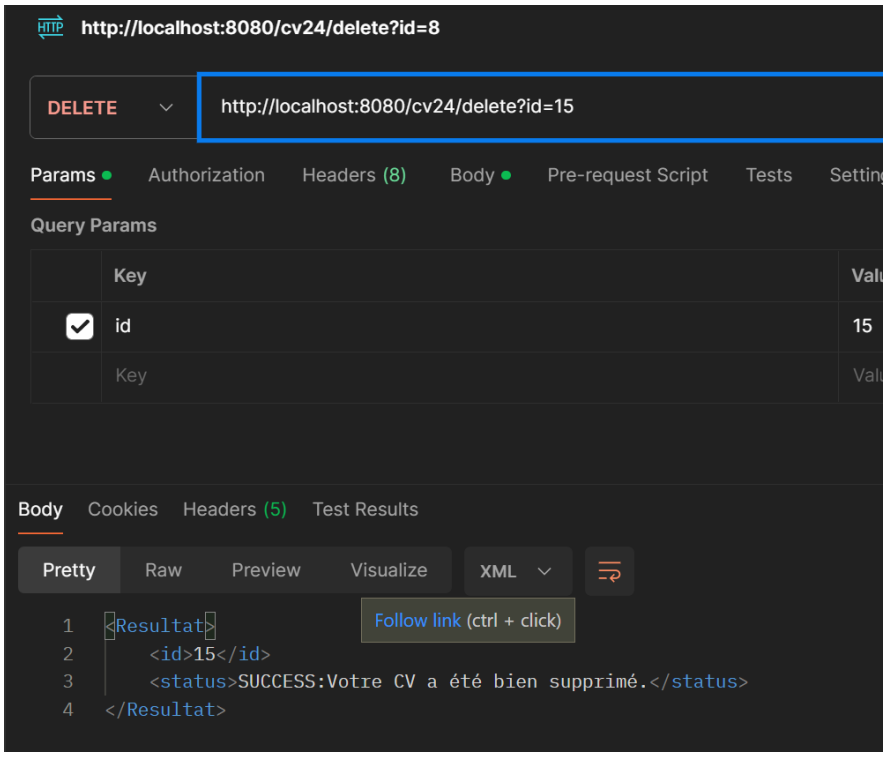

FIGURE 7 – Suppression cas de succès.

Suppression d'un CV en cas d'échec.

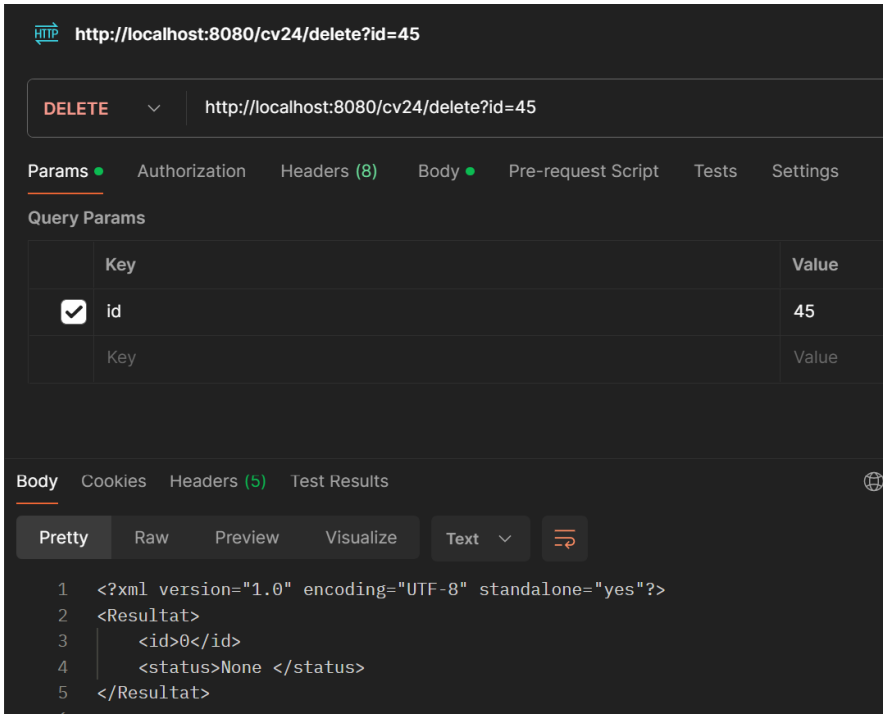

FIGURE 8 – Suppression cas échec.

Affichage de la page d'accueil.

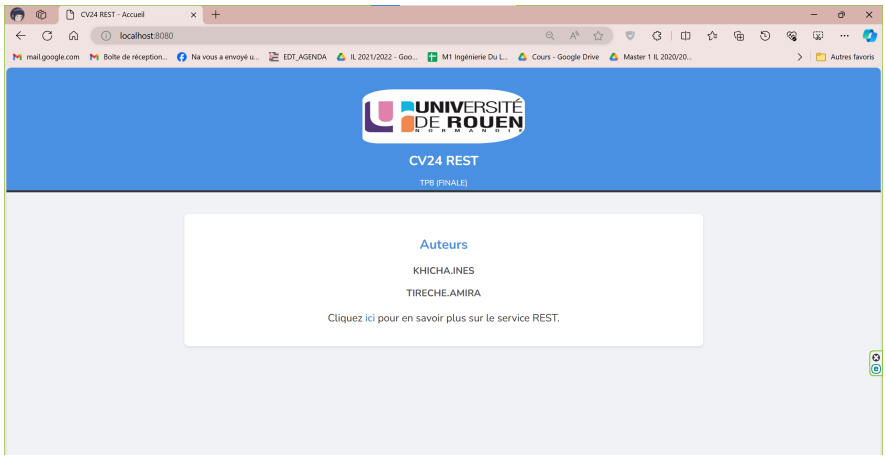

FIGURE 9 – Page d'accueil.

Affichage de la page help.

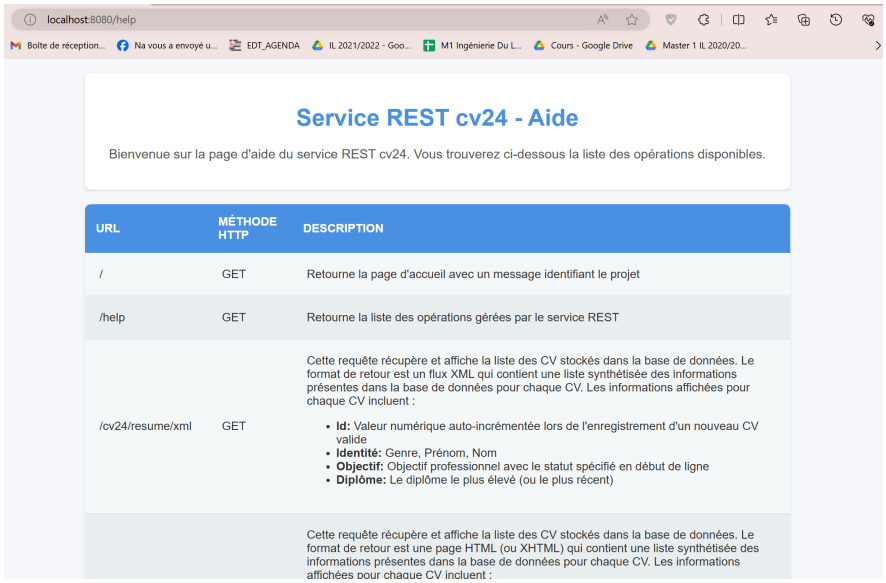

FIGURE 10 – Page help.

Affichage de la page des résumés de CVs.

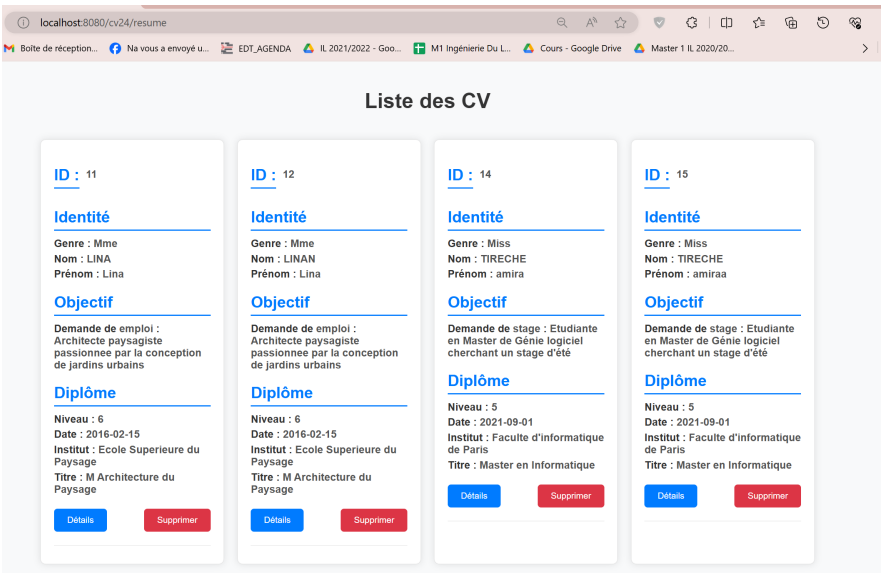

FIGURE 11 – Page des résumés de CVs.

Affichage de la page du détail d'un CV.

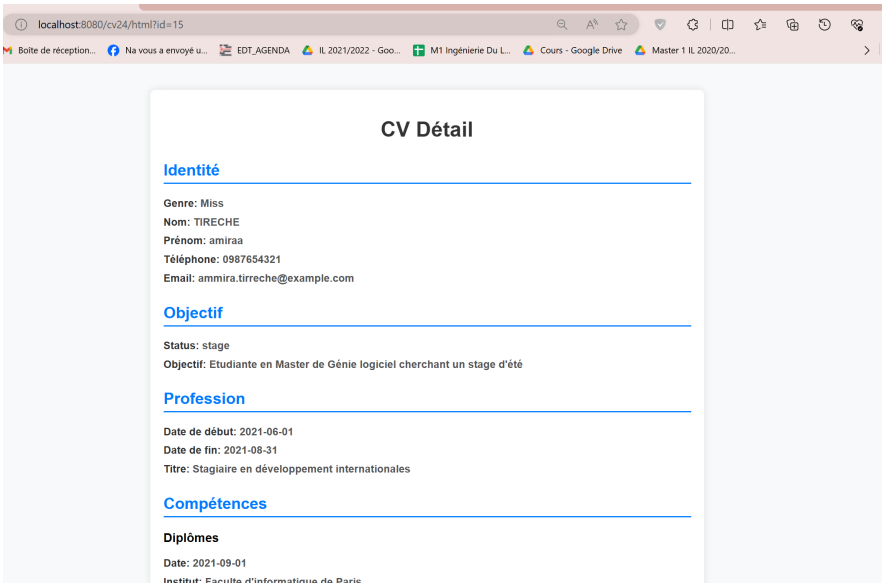

FIGURE 12 – Page du détail d'un CV.

Références

[1] PostgreSQL Global Development Group. *PostgreSQL 13.4 Documentation.* Available : https ://www.postgresql.org/docs/13/index.html

[2] Pivotal Software, Inc. *Spring Boot Reference Documentation.* Available : https ://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[3] Harold, E. R. *XML in a Nutshell.* O'Reilly Media, 2004.

[4] Keith, M., Schincariol, M. *Pro JPA 2 : Mastering the Java Persistence API.* Apress, 2013.

[5] Clever Cloud. *Documentation : PostgreSQL Add-on.* Available : https ://www.clever-cloud.com/doc/addons/postgresql/

[6] Thymeleaf. *Thymeleaf Documentation.* Available : https ://www.thymeleaf.org/documentation.html

[7] Oracle. *Java Platform, Standard Edition Documentation.* Available : https ://-docs.oracle.com/javase/8/docs/