

Master 1 GIL - XML Services Web  
Document d'Architecture Logiciel  
CV24

Etienne BINGINOT et Axelle MAZUY

30 avril 2024

Version	2
Date	30 avril 2024
Rédigé par	BINGINOT Etienne MAZUY Axelle

## Mises à jour du document

Version	Date	Modification réalisée
1	30 avril 2024	Création du document

# Table des matières

<b>1</b>	<b>Objet du document</b>	<b>4</b>
<b>2</b>	<b>Documents de références</b>	<b>5</b>
<b>3</b>	<b>Terminologie</b>	<b>5</b>
<b>4</b>	<b>Configuration requise</b>	<b>6</b>
4.1	Utilisation de l'application déployée . . . . .	6
4.2	Utilisation de l'application en déploiement local . . . . .	6
4.3	Configuration de la base de données . . . . .	6
4.4	Utilisation de l'application . . . . .	6
4.5	Configurations en environnement de développement . . . . .	6
<b>5</b>	<b>Architecture générale</b>	<b>7</b>
<b>6</b>	<b>Architecture statique - Les entités</b>	<b>8</b>
6.1	Description des composants du MCD . . . . .	8
<b>7</b>	<b>Architecture statique - Le service</b>	<b>12</b>
<b>8</b>	<b>Architecture statique - Le contrôleur</b>	<b>13</b>
<b>9</b>	<b>Choix d'architecture</b>	<b>14</b>
9.1	Choix de la base de données . . . . .	14
9.2	Validation d'un fichier XML . . . . .	14
9.3	Organisation des entités - Désérialisation des fichiers XML . . . . .	14
9.4	Dossier configuration . . . . .	14
9.5	Transformations XML - HTML . . . . .	14
9.6	Gestion des exceptions . . . . .	14
9.7	Mise en place de tests . . . . .	14
9.8	Mise en place de logs . . . . .	15
<b>10</b>	<b>Limitations techniques</b>	<b>15</b>
10.1	XSD en version 1.1 . . . . .	15
10.2	Utilisation d'un générateur de templates . . . . .	15

# 1 Objet du document

Ce document a pour but de décrire l'architecture logicielle du projet CV24. Nous allons décrire l'architecture de la bibliothèque de code.

Nous devons réaliser une API REST comprenant plusieurs routes pour la gestion de CV :

- les CV doivent être sauvegardés en Base de Données
- ils doivent pouvoir être supprimés
- il est possible de consulter l'ensemble des informations générales des CV en XML et HTML
- il est également possible de visualiser un CV en format XML et HTML
- une aide est proposée en accédant à un Swagger

L'enjeu majeur est donc de proposer une API REST convenable et utilisable.

Le déploiement est également un point important de ce projet.

## 2 Documents de références

Nos références utilisées sont les sujets de TP et du projet, ainsi que les Documentation d'Architecture Logicielle réalisés pour les différents projets cette année.

## 3 Terminologie

**API REST :** ou API RESTful, désigne une interface de programmation d'application respectant les contraintes d'architecture REST (Representational State Transfer) et interagit avec les services web Restful.

## 4 Configuration requise

### 4.1 Utilisation de l'application déployée

Pour accéder au projet déployé, il est possible d'utiliser l'URL suivante : <https://app-bc9d6f71-8df4-42dd-acdd-9421c5b77daf.cleverapps.io/>

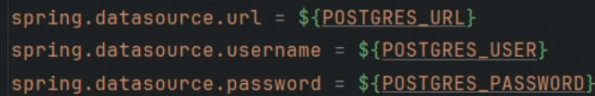
Le serveur y est déployé sur cette URL, avec un serveur Postgres également déployé sur CleverCloud.

Aucune configuration n'est donc requise pour accéder au projet, si ce n'est un navigateur avec une connexion internet.

### 4.2 Utilisation de l'application en déploiement local

Pour utiliser l'application en local, il est possible d'utiliser deux moyens :

- Utiliser la commande docker compose up --build qui lancera les conteneurs docker du projet et l'application sera accessible en localhost :8080
- Utiliser la commande mvn spring-boot :run dans le dossier app à la racine du projet, pour télécharger les dépendances et lancer l'application. Cependant il est nécessaire d'avoir un serveur Postgres en écoute sur le port 5432 pour que le serveur fonctionne. Il est également nécessaire de modifier le fichier application.properties dans le dossier app pour y rajouter l'url du serveur, ainsi que l'utilisateur et le mot de passe.



```
spring.datasource.url = ${POSTGRES_URL}
spring.datasource.username = ${POSTGRES_USER}
spring.datasource.password = ${POSTGRES_PASSWORD}
```

FIGURE 1 – Lignes à modifier dans le cas d'un déploiement sans Docker

### 4.3 Configuration de la base de données

La base de données est automatiquement configurée au démarrage de l'application via l'utilisation de la bibliothèque Jakarta. Il n'y a donc pas de script de création de la base de données.

### 4.4 Utilisation de l'application

Une fois l'application déployée en local ou sur Clevercloud, il est possible d'accéder à la route d'aide avec l'url /help. Ainsi, une interface Swagger sera disponible pour facilement tester l'application.

En cas de déploiement local via Docker, il est possible de choisir entre deux Dockerfile dans le fichier .env à la racine du projet. Le premier, Dockerfile, réalise un lancement classique de springboot. Le seconde, Dockerfile\_auto\_refresh utilise un script sh pour permettre à l'application de redémarrer automatiquement en cas de modification des fichiers.

### 4.5 Configurations en environnement de développement

D'un point de vue développement, nous avons utilisé :

- Ubuntu en version 22.04 (LTS)
- Java en version JDK 17
- Spring boot en version 3.2
- Github pour le partage de code
- Maven pour la gestion de dépendances
- PostgreSQL en version 16.2
- Docker

## 5 Architecture générale

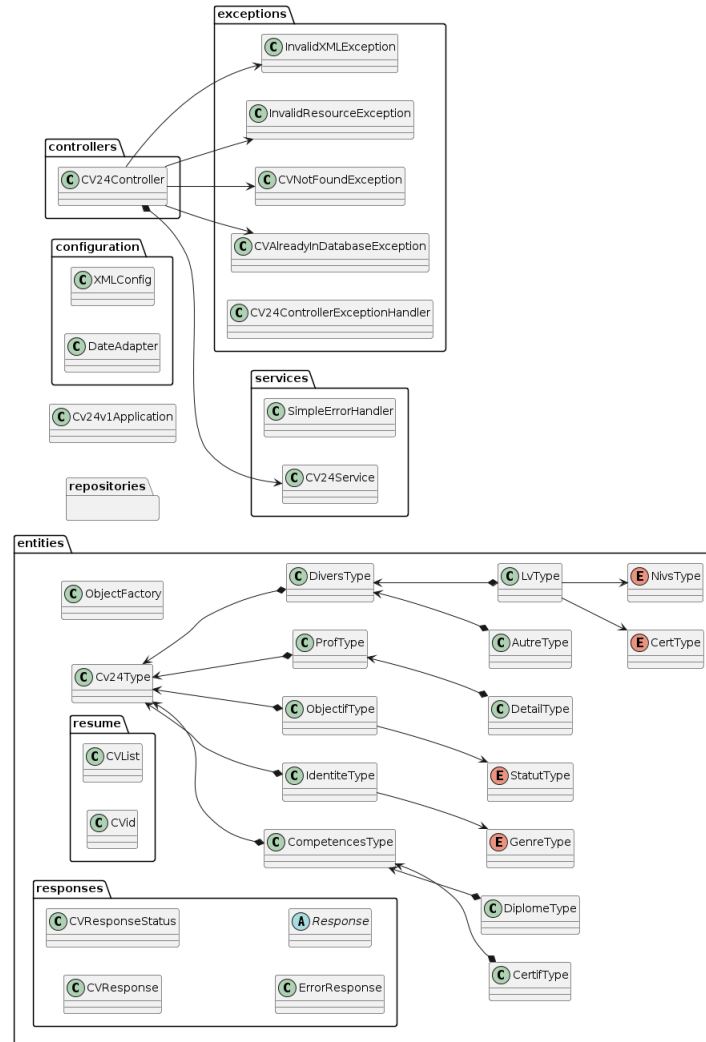


FIGURE 2 – Diagramme de classe de l'architecture générale des paquets et classes

On peut voir sur le diagramme d'architecture générale que l'application respecte un modèle MVC (bien que la vue ne figure pas sur le diagramme de classe, les fichiers se trouvant dans un paquetage resources).

Chaque dossier a son utilisé, on retrouve donc non seulement les contrôleurs (controllers) et les entités (entities), mais aussi un paquetage configuration permet d'adapter une date par rapport à une string, ou la configuration de la transformation d'un fichier XML en les entités que nous définissons. Il s'agit donc de la configuration d'éléments utiles, d'après l'API XML.

Nous retrouvons également un dossier exceptions, qui regroupe les types d'exceptions que nous avons défini nous mêmes pour les retours des routes.

De même, on retrouve un dossier services contenant le code métier, et repositories permettant d'interroger la Base de données.

## 6 Architecture statique - Les entités

Les entités sont définies sur la base du xsd qui a pu être écrit.

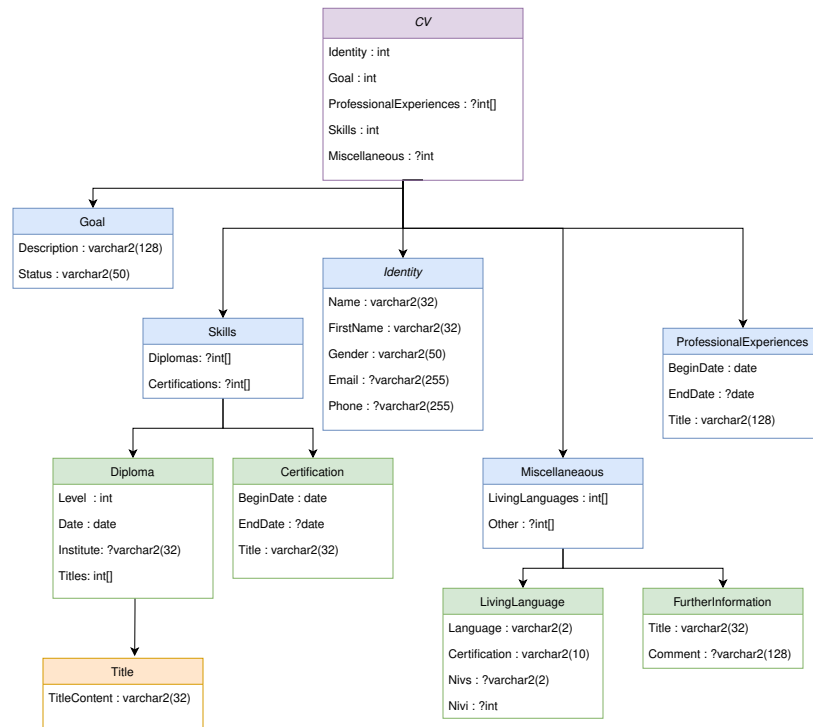


FIGURE 3 – Caption

### 6.1 Description des composants du MCD

Description de l'entité AutreType :

Champ	Description
id	Un entier identifiant l'objet
titre	Une chaîne de caractères correspondant au titre de la section autre
comment	Une chaîne de caractères correspondant au commentaire associé à la section autre

TABLE 1 – Tableau de description des composants représentant la section autre d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section Autre.

Description de l'entité CertifType :

Champ	Description
id	Un entier identifiant l'objet
titre	Une chaîne de caractères correspondant au titre de la certification
datedeb	Une date indiquant le début de la certification
datefin	Une date indiquant la fin de la certification

TABLE 2 – Tableau de description des composants représentant une certification d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans une certification.



### Description de l'entité CertType :

Champ	Description
MAT	Désigne un niveau langue maternelle
CLES	Désigne une certification de type CLES
TOEIC	Désigne une certification de type TOEIC

TABLE 3 – Tableau de description des valeurs possibles pour une certification linguistique

Cette énumération sert notamment à distinguer les différents types de certification linguistiques.

### Description de l'entité CompetencesType :

Champ	Description
id	Un entier identifiant l'objet
diplome	Une liste d'objet diplômes possédés par l'utilisateur
certif	Une liste d'objet certifications possédées par l'utilisateur

TABLE 4 – Tableau de description des composants contenues dans la section Compétences d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section compétences d'un CV.

### Description de l'entité Cv24Type :

Champ	Description
id	Un entier identifiant l'objet
identite	Un objet contenant les informations de la section identité d'un CV
objectif	Un objet contenant les informations de la section objectif d'un CV
prof	Un objet contenant les informations de la section expériences professionnelles d'un CV
competences	Un objet contenant les informations de la section compétences d'un CV
divers	Un objet contenant les informations de la section divers d'un CV

TABLE 5 – Tableau de description des composants représentant un CV

Ces informations servent donc à contenir les données représentant un CV en Base de données.

### Description de l'entité DetailType :

Champ	Description
id	Un entier identifiant l'objet
titre	Une chaîne de caractères correspondant au titre du détail
datedeb	Une date indiquant le début du détail
datefin	Une date indiquant la fin du détail

TABLE 6 – Tableau de description des composants représentant le détail d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans un détail.

### Description de l'entité DiplomeType :

Champ	Description
id	Un entier identifiant l'objet
titre	Une chaîne de caractères correspondant au titre de la certification
date	Une date indiquant la date d'obtention du diplôme
institut	Une chaîne de caractères indiquant le lieu d'obtention du diplôme
niveau	Un entier indiquant le niveau du diplôme

TABLE 7 – Tableau de description des composants représentant un diplôme d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans un diplôme.

#### Description de l'entité **DiversType** :

Champ	Description
id	Un entier identifiant l'objet
lv	Une liste de langues parlées par l'utilisateur
autre	Une liste de mentions autre

TABLE 8 – Tableau de description des composants représentant la section divers d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section divers d'un CV.

#### Description de l'entité **GenreType** :

Champ	Description
M.	Désigne un genre masculin en français
Mme.	Désigne un genre féminin en français
Mrs	Désigne "madame" en anglais
Miss	Désigne "mademoiselle" en anglais
Mr	Désigne "monsieur" en anglais

TABLE 9 – Tableau de description des valeurs possibles pour un genre

Cette énumération sert notamment à distinguer les différents types de genre.

#### Description de l'entité **IdentiteType** :

Champ	Description
id	Un entier identifiant l'objet
genre	Une valeur désignant le genre de l'utilisateur
nom	Une chaîne de caractères indiquant le nom de l'utilisateur
prenom	Une chaîne de caractères indiquant le prénom de l'utilisateur
tel	Une chaîne de caractères indiquant le numéro de téléphone de l'utilisateur
mel	Une chaîne de caractères indiquant le mail de l'utilisateur

TABLE 10 – Tableau de description des composants représentant la section identité d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section identité d'un CV.

**Description de l'entité LvType :**

Champ	Description
id	Un entier identifiant l'objet
lang	Une chaîne de caractères indiquant la langue parlée par l'utilisateur
cert	Un objet indiquant la certification sur cette langue
nivs	Un objet indiquant le niveau de la certification possiblement eu
nivi	Un entier indiquant le score de la certification possiblement eu

TABLE 11 – Tableau de description des composants représentant une langue sur un CV

Ces informations servent donc à contenir les données pouvant être présentes dans une langue sur un CV.

**Description de l'entité NivsType :**

Champ	Description
A1	Désigne un niveau A1
A2	Désigne un niveau A2
B1	Désigne un niveau B1
B2	Désigne un niveau B2
C1	Désigne un niveau C1
C2	Désigne un niveau C2

TABLE 12 – Tableau de description des valeurs possibles pour une certification CLES

Cette énumération sert notamment à distinguer les différents types de niveaux pour une certification CLES.

**Description de l'entité ObjectFactory :** Cette classe est une factory méthode pour chaque type d'objet. Elle permet d'instancier les objets des différentes entités à partir d'un fichier XML.

**Description de l'entité ObjectifType :**

Champ	Description
id	Un entier identifiant l'objet
value	Une chaîne de caractères indiquant l'intitulé de l'objectif poursuivi avec le CV
statut	Un objet indiquant le statut du poste recherché

TABLE 13 – Tableau de description des composants représentant l'objectif d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section objectif d'un CV.

**Description de l'entité ProfType :**

Champ	Description
id	Un entier identifiant l'objet
detail	Une liste de détails des expériences professionnelles d'un utilisateur

TABLE 14 – Tableau de description des composants représentant la section expériences professionnelles d'un CV

Ces informations servent donc à contenir les données pouvant être présentes dans la section expériences professionnelles d'un CV.

#### Description de l'entité StatutType :

Champ	Description
STAGE	Désigne un objectif de poste en tant que stagiaire
EMPLOI	Désigne un objectif de poste en tant qu'employé

TABLE 15 – Tableau de description des valeurs possibles pour un statut de poste

Cette énumération sert notamment à distinguer les différents types de niveaux pour un statut de poste.

## 7 Architecture statique - Le service

Le service CV24Service est donc le coeur du code métier de l'application. C'est ici que sont manipulées les entités et le passage de XML/HTML aux entités (ou inversement).

#### Description du service CV24Service :

Méthode	Type de retour	Description
getXMLCvDocumentFromInputStream (InputStream cv)	Document	Parse le flux d'entrée de l'utilisateur et renvoie un document XML s'il est valide
isValidCV(Document cv)	boolean	Indique si le document XML est valide et s'il est conforme à la xsd
isAlreadyInDatabase (Document cv)	boolean	Vérifie si le CV XML fourni possède déjà une entrée avec la même identité en BDD
transformCVToXML (Cv24Type cv)	String	A partir d'une entité CV, renvoie le code XML
saveCV(Cv24Type cv)	Cv24Type	Enregistre le CV en BDD
createHTML(Cv24Type cv)	String	Applique la xslt et transforme un cv en HTML
createCVsResume()	String	Renvoie la concaténation des résumés de l'ensemble des CV de la BDD en XML
createHTMLResume()	String	Renvoie la concaténation des résumés de l'ensemble des CV de la BDD en HTML

TABLE 16 – Tableau de description des valeurs possibles pour un statut de poste

## 8 Architecture statique - Le contrôleur

Le contrôleur est ainsi le coeur de l'application, puisque c'est ici que sont définies les routes de l'application et de gestion des CV.

### Description du service CV24Service :

Méthode HTTP	Nom de méthode	Type de retour	Description
GET	index()	ModelAndView	Affiche la page d'accueil de l'application
GET	resumeXML()	String	Affiche la liste des CV sous format XML
GET	findXMLCVById(id)	ResponseEntity<String>	Récupère et renvoie un CV au format XML avec l'identifiant donné en paramètre
GET	findHTMLCVById(id)	ResponseEntity<String>	Récupère et renvoie un CV au format HTML avec l'identifiant donné en paramètre
POST	insertCV(String cv)	ResponseEntity<CVResponse>	Insère dans la BDD le cv donné dans le corps de la requête HTTP si celui-ci n'existe pas déjà et est valide
DELETE	deleteCV(id)	ResponseEntity<CVResponse>	Supprime le CV de la BDD si l'identifiant donné en paramètre est valide

## 9 Choix d'architecture

### 9.1 Choix de la base de données

Nous avons choisi d'utiliser comme logiciel de base de données Postgres. En effet, c'est celui qui convient le mieux à notre application, du fait de sa rapidité, de sa facilité d'intégration avec spring boot, des possibilités fournies par les API Postgres (XPath directement inclus en Postgres). De plus, son utilisation est gratuite avec Clevercloud.

### 9.2 Validation d'un fichier XML

Nous avons utilisé un fichier XSD pour vérifier la structure que le fichier est conforme à ce qui est attendu (un fichier valide), auquel cas le XML est ajouté en Base de données, le cas échéant une erreur est générée.

La XSD permet de s'assurer des contraintes demandées par rapport aux CVs.

### 9.3 Organisation des entités - Désérialisation des fichiers XML

Nous avons fait le choix d'utiliser l'API Jakarta XML pour avoir une désérialisation automatique des fichiers XML en différentes entités que nous avons pu sauvegarder en BDD via l'utilisation de l'API Jakarta Persistence. Pour la désérialisation en différentes entités, l'API se base sur notre xsd à partir des types définis. C'est pourquoi, les noms des types dans la xsd sont les mêmes que les noms des entités.

Nous utilisons donc l'objet Marshall avec la méthode marshall pour sérialiser depuis une entité vers un fichier XML et unmarshall pour désérialiser.

### 9.4 Dossier configuration

Nous avons fait le choix d'utiliser un dossier configuration contenant certaines configurations de l'application et de ne pas tout configurer via le application.properties.

Cela permet ainsi de garder ce qui est destiné à changer dans l'application.properties et de garder les configurations permanentes à l'extérieur dans le dossier configuration.

### 9.5 Transformations XML - HTML

Nous utilisons des fichiers xslt pour la transformation d'un document XML en HTML ainsi, cette transformation nous permet d'utiliser XQuery et de nous assurer de la validité des informations que nous transmettons.

De plus, nous utilisons également xslt pour la transformation d'un CV en une version résumée afin d'avoir un code clair et concis.

### 9.6 Gestion des exceptions

Pour gérer les exceptions nous avons choisi d'utiliser un Handler d'exceptions qui est lié à notre contrôleur. Ainsi les différentes exceptions que nous avons défini et qui sont déclenchées par le contrôleur seront capturées par ce handler, ce qui permet une gestion centralisée des erreurs.

Nous avons ainsi différents fichiers d'exceptions nous permettant d'envoyer des erreurs personnalisées

### 9.7 Mise en place de tests

Nous avons également mis en place des tests unitaires sur notre application pour nous assurer de sa qualité.

De plus, nous avons réalisé de nombreux tests fonctionnels via Postman et Swagger, pour nous assurer du bon fonctionnement et de la qualité du code.

## **9.8 Mise en place de logs**

Un système de log a été mis en place pour réaliser des messages de logs en cas d'insertions ou de suppressions CV (log info), et également dans le cas d'erreur de ressources introuvables (log critique, il s'agit d'une erreur interne au serveur).

# **10 Limitations techniques**

## **10.1 XSD en version 1.1**

Il aurait été possible de passer notre XSD en version 1.1 pour nous permettre d'utiliser des assert et ainsi coller au plus proche de ce qui a pu être demandé lors du TP1.

Cependant, du fait de limitations techniques liées à notre environnement de développement, notamment notre IDE, il n'a pas été possible de configurer le préprocesseur XSD en 1.1, ce qui nous a empêché de réaliser cette transformation.

## **10.2 Utilisation d'un générateur de templates**

Nous avons choisi de ne pas configurer de générateurs de templates dans un souci de performance de l'applications.

En effet, il n'aurait pu être utilisé qu'à un seul endroit, lors du retour d'erreur d'un fichier HTML pour pouvoir facilement y placer l'identifiant du CV et le statut. N'étant utilisé qu'ici, nous n'avons pas jugé qu'ajouter la dépendance vers le générateur était utile.