Guide du programmeur
CT-Application

Auteur :
Jules Dejaeghere

Promoteur :
Pr Jean-Noël Colin

Version 1.0.0
Année académique : 2019-2020

Table des matières

1 Introduction

Ce document constitue la documentation technique de l'application développée par Jules Dejaeghere dans le cadre du cours *INFOB318 - Projet individuel*, dispensé à l'Université de Namur par le Professeur Vincent Englebert au cours de l'année académique 2019-2020. Ce projet a été proposé par le Professeur Jean-Noël Colin.

Ce document vise à fournir une description du contexte de développement de l'application ainsi que les clefs de l'architecture du logiciel pour permettre sa maintenance, son évolution ou sa reprise.

2 Description conceptuelle

La description du projet fournie par le Professeur Colin est la suivante.

> L'objectif du projet est de constituer une base de données de noms de domaine enrichie d'informations disponibles publiquement ou calculées à partir de données publiques. Le workflow partira de la base de données *Certificate Transparency Logs* (`https://www.certificate-transparency.org/what-is-ct`), et complétera avec les éléments suivants, à extraire des sources appropriées :
> — comparaison des certificats de CT avec ceux réellement utilisés sur le site web
> — statistiques sur les données des certificats (palmarès des CA, types de certificats...)
> — validation du numéro de TVA du registrant
> — extraction des données personnelles et adresses du site web
> Ces données seront ensuite intégrées à un outil utilisant l'IA pour détecter des noms de domaine malicieux et en bloquer l'accès aussi vite que possible. Le développement sera réalisé en Java et la base de données sera PostgreSQL.

Cette description ainsi que les différents échanges avec le Professeur au cours du projet ont constitué les lignes directrices du développement de l'application.

3 Contexte du développement

L'application a été développée par Jules Dejaeghere, étudiant en bachelier en informatique à l'Université de Namur, entre septembre 2019 et mars 2020. Un des objectifs lors du développement de l'application était d'arriver à produire une application de taille raisonnable mais fonctionnelle plutôt qu'une application plus ambitieuse et non fonctionnelle. Cette application peut donc être améliorée par l'ajout de nouvelles fonctionnalités ou des considérations techniques plus poussées. Plusieurs pistes d'amélioration de l'application seront proposées au cours de ce document. Ces pistes constituent un point de départ intéressant dans le cadre de la poursuite du développement.

4 Sources et documentation de l'application

Actuellement, le code source de l'application est hébergé sur un dépôt GitHub géré par la Faculté d'Informatique de l'Université de Namur. Ce dépôt est actuellement privé et se trouve à l'adresse suivante : `https://github.com/UNamurCSFaculty/1920_INFOB318_CT`.

Ce dépôt contient les fichiers source de l'application ainsi que la documentation technique et de l'utilisateur. Une version compilée du logiciel ainsi que le planning du projet se trouvent également dans le dépôt.

Ce document reprend la documentation technique uniquement. Pour déployer l'application et utiliser les fonctionnalités de cette dernière, le *Guide de l'utilisateur* détaille les différentes étapes à suivre.

5 Environnement de développement

L'application est développée en Java, comme mentionné dans la description du projet. Pour supporter le développement et sur conseil du Professeur Colin, le framework Spring Boot a été utilisé.

L'environnement de développement de JetBrains pour Java, *IntelliJ IDEA 2019.3.3 (Ultimate Edition)*, a permis de faire fonctionner les différents outils utilisés lors du développement de l'application : Maven pour gérer le cycle de vie, JUnit pour développer des tests unitaires ainsi que le framework Spring Boot.

5.1 Compiler l'application

La compilation et la gestion des dépendances est prise en charge par Maven. Grâce à IntelliJ, il est possible de lancer la compilation directement depuis l'environnement de développement, à l'aide du module Maven, et de faire abstraction des commandes exécutées par Maven.

Il est également possible de compiler l'application grâce aux commandes Maven. Une fois dans le répertoire contenant le fichier `pom.xml`, exécuter les commandes suivantes :

```
mvn clean
mvn install
```

Dans les deux cas, l'application compilée avec les dépendances se trouve dans le répertoire `target`.

5.2 Exécuter les tests unitaires

Au cours du développement de l'application, plusieurs classes de test ont été développées. L'outil utilisé pour les tests unitaires est JUnit. Tout comme Maven, il est possible de l'utiliser directement depuis l'environnement de développement pour lancer des tests unitaires. En sélectionnant un fichier de test, il est possible de le lancer directement depuis IntelliJ.

6 Développement

6.1 Fonctionnement général de l'application et librairies utilisées

Au cours de son exécution, l'application télécharge des entrées de logs depuis des serveurs de logs du projet Certificate Transparency, décode les logs, parcourt les sites web repris dans les certificats à la recherche d'un numéro de TVA et sauvegarde les données dans la base de données. Ces différentes étapes seront expliquées dans les prochaines parties.

Pour appuyer les explications suivantes, la figure 1 présente l'agencement des paquets et des fichiers dans le package `be.unamur.ct`. Les paquets sont découpés de manière à refléter le fonctionnement de l'application. Chaque paquet principal est décomposé en sous-paquets pour y répartir les classes en fonction de leur utilité.

`exceptions :` contient les exceptions qui concernent le paquet en question

`model :` contient les définitions d'objets Java utilisées dans le paquet

`service :` contient les fonctions principales de l'application

`thread :` contient la définition des threads que le paquet peut soumettre à des Executors

6.1.1 Télécharger les logs

La première étape lors de l'exécution du programme est de télécharger les logs depuis un serveur du projet Certificate Transparency. Ces logs sont téléchargeables via une API qui fournit des données au format JSON.

Comprendre la manière dont les serveurs de logs fonctionnent n'est pas trivial. Pour y arriver, plusieurs ressources ont été utiles. La première n'est autre que le site de Certificate Transparency (`https://www.certificate-transparency.org`). Il permet d'avoir un aperçu général du projet mais propose également une liste des serveurs de log connus. La seconde est un article du site Medium : *Parsing Certificate Transparency Logs Like a Boss* (`https://medium.com/cali-dog-security/parsing-certificate-transparency-lists-like-a-boss-981716dc506`). L'article explique en détail comment sont organisés les logs et comment y accéder. De plus, l'article détaille la manière dont il est possible d'extraire les informations des logs téléchargés. Bien que l'auteur illustre ses propos par des scripts Python, il est facile de le transposer en Java.

Le contenu de paquet `download` contient les méthodes nécessaires pour connaître la taille d'un log, découper ces logs en tranches pour les télécharger parallèlement et effectivement télécharger les logs. L'implémentation de ces méthodes s'inspire de l'article présenté précédemment.

Pour obtenir ces données, la librairie *OkHttp3* est utilisée pour se connecter au serveur et effectuer les requêtes. Une fois ces données au format JSON téléchargées, elles sont temporairement stockées dans un objet à l'aide de la librairie *Jackson*. Les données téléchargées sont encodées en Base64 et devront être décodées pour être utilisables.

6.1.2 Décoder les logs

Une fois téléchargés, les logs doivent être décodés et triés. En effet, les logs sont téléchargés, encodés en Base64, et seuls les logs concernant des certificats belges nous

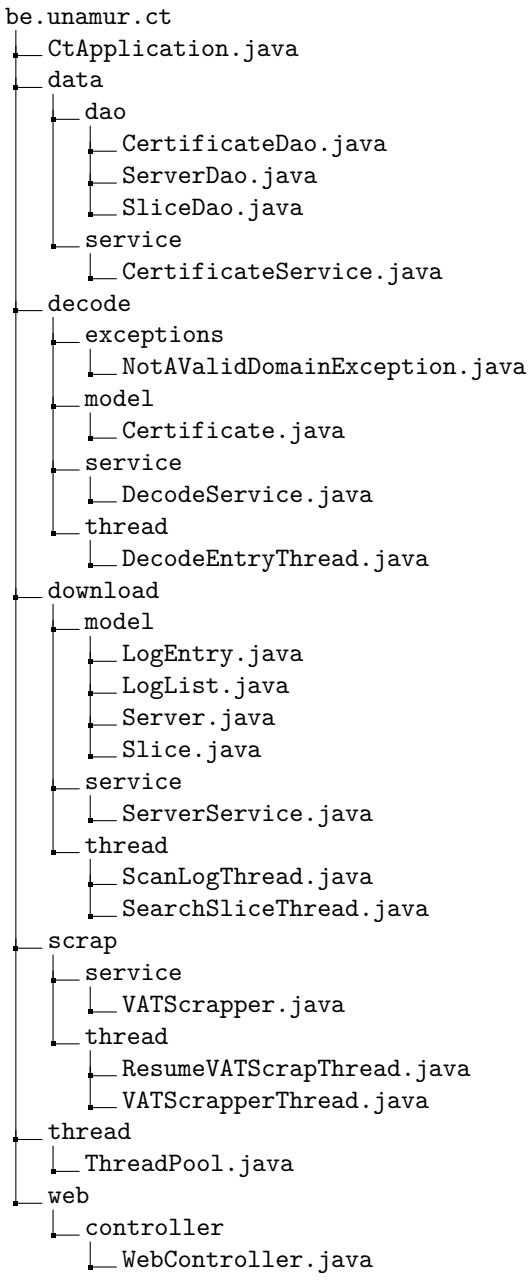

FIGURE 1 – Arborescence des paquets et des fichiers source

intéressent dans notre cas. Le paquet `decode` contient différentes méthodes qui permettent de décoder les données téléchargées et les enregistrer, si elles sont pertinentes, dans un objet Java.

Ce paquet contient une classe, `Certificate`, qui sera utilisée pour stocker les certificats pertinents dans la base de données. Chaque certificat téléchargé sera converti en un objet `Certificate` avant d'être enregistré ou abandonné. Si le programme n'arrive pas à construire un objet de ce type sur base du log reçu, il sera abandonné.

La classe `Certificate` est également utilisée par Spring Boot pour stocker les objets dans la base de données.

Pour décoder les certificats et en extraire les informations nécessaires, la librairie Bouncy Castle a été utilisée.

```
public class Certificate {

    private long id;

    private String subject;

    private String issuer;
    private Date notAfter;
    private Date notBefore;

    private String signatureAlg;
    private int versionNumber;
    private String VAT;
    private boolean vatSearched;
}
```

FIGURE 2 – Variables de la classe `Certificate`

6.1.3 Rechercher le numéro de TVA

Une fois les certificats belges identifiés et sauvegardés, ceux-ci sont traités à la recherche d'un numéro de TVA sur le site lié au certificat. Le paquet `scrap` regroupe toutes les méthodes qui se chargent d'explorer le site web lié à un certificat à la recherche d'un numéro de TVA.

Pour explorer le site, la méthode principale démarre de l'URL qui est renseignée dans le certificat et parcourt la page. Si aucun numéro de TVA n'est trouvé sur la page, les liens présents sur la page pointant vers le même domaine sont parcourus récursivement de la même manière. Pour casser la récursivité, une limite de profondeur est fournie à la fonction. De même, un ensemble d'URL déjà visitées est tenu à jour de manière à ne pas visiter plusieurs fois la même page.

Pour reconnaître un numéro de TVA dans une page web, l'application utilise l'expression régulière de la figure 3. Le résultat est ensuite passé à une fonction qui se charge de normaliser les numéros de TVA avant de les enregistrer dans la base de données.

```
"(?i)((BE)?0([. -])?[0-9]\{3\}([. -])?[0-9]\{3\}([. -])?[0-9]\{3\})"
```

FIGURE 3 – Expression régulière du numéro de TVA

6.2 Paquets additionnels

En plus des paquets déjà présentés, l'application comporte plusieurs paquets qui assurent des fonctions auxiliaires : gérer la persistance des données, gérer le mutli-threading ou encore proposer l'interface web.

Ces paquets sont décrits dans les sections suivantes.

6.2.1 Persistance des données

Le paquet `data` prend en charge toutes les fonctions relatives à la persistance des données ainsi que certaines fonctions d'agrégation des données pour permettre leur affichage sur l'interface web.

Ce paquet contient notamment trois interfaces qui étendent la classe `JpaRepository`. Ces interfaces sont une abstraction permise par Spring Boot pour communiquer avec la base de données. Cela permet notamment de récupérer les données persistantes à l'aide de méthodes Java, sans toujours nécessiter l'écriture de requêtes SQL.

6.2.2 Multi-threading

Pour permettre à l'application d'utiliser au mieux les ressources physiques de la machine, la classe `thread` fournit des méthodes qui permettent à l'application de s'exécuter en plusieurs threads.

La classe contenue dans ce paquet est une implémentation étendue du pattern design singleton. En effet, cette classe garantit l'existence unique de quatre objets de type `ExecutorService`. Un `ExecutorService` permet une abstraction quant à l'implémentation d'une file de threads à exécuter. Quatre files de threads existent dans l'application, chacune avec une fonction spécifique, et peuvent être appelées par d'autres classes pour effectuer des tâches lorsqu'un thread se libère. En utilisant des objets de la classe `ExecutorService`, cela permet d'éviter de créer un nombre trop important de threads.

6.2.3 Interface web

Le paquet `web.controller` implémente les méthodes nécessaires pour fournir l'interface web. Cette interface web a été réalisée à l'aide de Thymeleaf, qui propose une intégration avec Spring Boot. L'interface web fournit principalement des informations relatives à l'état acutel du programme, des statistiques relatives aux données stockées et la possibilité d'ajouter de nouveaux serveurs de logs à consulter.

Les modèles HTML sont stockés dans le répertoire `src/main/ressources/templates`.

7 Poursuivre le développement

Dans l'optique d'une poursuite du développement de l'application, le présent document constitue un point de départ pour comprendre l'agencement des différentes parties

du programme. Cette section propose plusieurs pistes pour entamer la poursuite du développement de l'application.

7.1 Améliorer le code existant

Pendant le développement de l'application, à plusieurs reprises, un choix entre rapidité d'implémentation et efficacité a dû être posé. Peut-être qu'un regard différent sur ces parties de l'application ou un peu plus de temps permettront de trouver une solution plus efficace et plus élégante.

Les points où de tels choix ont dû être posés et qui ne semblent pas satisfaisants ont été annotés d'un commentaire `// TODO`, suivi d'une brève description du problème.

7.1.1 Gestion du multi-threading

La première partie où une amélioration est la bienvenue porte sur l'arrêt des threads en charge de la recherche du numéro de TVA sur les sites web contenus dans les certificats. En effet, après observations, les threads ne semblent pas s'arrêter lorsque la méthode `interrupt()` du thread est appelée.

Pour palier à ce problème, la solution implémentée consiste, lors de chaque entrée dans la fonction récursive principale du thread, à vérifier si l'objet `ExecutorService` dont est supposé dépendre ce thread n'a pas reçu de demande d'arrêt. Si une demande d'arrêt à été émise pour cet objet, le thread sera alors interrompu. Cela pourrait poser problème dans le cas où la méthode est lancée par un thread qui ne dépend pas de cet objet `ExecutorService`. Le code problématique se trouve dans le fichier `be/unamur/ct/scrap/service/VATScrapper.java` est repris dans la figure 4.

```
/* TODO:
 *  Find a better way to stop the thread
 *  The Thread.interrupt() method doesn't seem to work
 */
if (threadPool.getVATScrapperExecutor().isShutdown()) {
    throw new InterruptedException();
}
```

FIGURE 4 – Arrêt manuel du thread

7.1.2 Recherche de l'autorité de certification racine

La seconde partie qui pourrait faire l'objet d'une amélioration concerne la recherche de l'autorité de certification racine dans une entrée du log. Dans l'entrée d'un log, la chaîne de confiance depuis le certificat concerné jusqu'à l'autorité de certification racine doit être présente. Cependant, aucun autre moyen n'a été trouvé pour extraire le certificat de l'autorité de certification racine que de parcourir cette chaîne de confiance, octet après octet, à partir de la fin, en tentant de lire un certificat après chaque décalage d'un octet.

Cette chaine de confiance se trouve dans la partie nommée `extra_data` des données JSON récupérées du serveur. L'itération est donc opérée sur ces données, comme repris dans la figure 5.

```
/*
 * TODO:
 *  find a better  way to determine the RootCA, the approach described
 ↪  above is probably not the best
 */

byte[] extraBin = Base64.decode(extra_data);
int start = extraBin.length - 5;

while (start >= 0) {

    try {
        // Get certificate type (X.509 or PreCert)
        int id = (extraBin[start + 1] & 0xFF) | ((extraBin[start] & 0xFF)
        ↪  << 8);
        int l = (extraBin[start + 4] & 0xFF) | ((extraBin[start + 3] &
        ↪  0xFF) << 8) | ((extraBin[start + 2] & 0x0F) << 16);

        byte[] certBin = Arrays.copyOfRange(extraBin, start + 5, l +
        ↪  start + 5);

        try {
            X509CertificateHolder certX = new
            ↪  X509CertificateHolder(certBin);
            RDN cn = certX.getSubject().getRDNs(BCStyle.CN)[0];
            String cns =
            ↪  IETFUtils.valueToString(cn.getFirst().getValue());
            return cns;
        } catch (IOException e) {
        } catch (IndexOutOfBoundsException e) {
        }
    } catch (Exception e) {
        logger.warn(e.toString());
    }
    start--;
}
```

FIGURE 5 – Recherche de l'autorité de certification racine

7.2 Ajout de nouvelles fonctionnalités

Toujours dans l'optique de la poursuite du développement de l'application, plusieurs fonctionnalités additionnelles peuvent être envisagées. Des fonctionnalités de recherche plus poussées sur les sites web pourraient être implémentées. Par exemple, une recherche de numéro de téléphone, de localité ou d'adresse.