SAPIENZA
UNIVERSITÀ DI ROMA

Reingegnerizzazione del software ‘OpenWhistleblowing’: analisi, sviluppo e rischi imprenditoriali

Facoltà di Ingegneria dell’Informazione, Informatica e Statistica
Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Corso di laurea in Ingegneria Informatica

Paolo Rollo
Matricola 1713885

Relatore
Riccardo Lazzeretti

Relatori esterni
Emiliano Fedeli
Daniele Savoretti

A.A. 2018-2019

Indice dei contenuti

Indice delle figure

1 Introduzione

L'elaborato prevede come scopo primario la riprogettazione del software OpenWhistleblowing 1.0.1, rilasciato come prodotto Open Source dall'Autorità Nazionale Anticorruzione, affinché esso possa essere adottato per il riuso dalle Pubbliche Amministrazioni Centrali e Locali, risolvendo alcune problematiche della piattaforma attuale e integrando le richieste e considerazioni espresse dai primi utilizzatori.

Nel secondo capitolo vengono illustrati i decreti legislativi italiani 190/2012 e 179/2017 e le direttive europee 2013/36, 2015/849, 2018/843, e le loro implicazioni sugli strumenti informatici.

Nel terzo capitolo dell'elaborato è mostrato il processo di whistleblowing in tutte le sue parti, con particolare attenzione posta sul processo di gestione, sugli attori e sul risultato di quest'ultimo.

Il quarto capitolo è dedicato alla piattaforma OpenWhistleblowing 1.0.1 di partenza e ne vengono illustrate l'architettura e le tecnologie utilizzate, per poi infine soffermarsi sul framework di base dell'applicativo: GlobaLeaks.

Nel quinto capitolo viene illustrata la nuova piattaforma OpenWhistleblowing 2.0.0 con la sua architettura, le tecnologie utilizzate e le scelte implementative adottate per risolvere i problemi ricorrenti. Per ognuna delle caratteristiche sopra elencate, s'illustrano al lettore le motivazioni che ne hanno suggerito il suo uso, le complessità che ne derivano e il miglioramento che hanno apportato. Successivamente, vengono specializzate le logiche astratte esposte: vengono illustrate ed applicate concretamente le tecnologie illustrate all'inizio del capitolo, mettendo in luce i pro e i contro di un sistema complesso come quello dell'elaborato, enunciando l'approccio al problema e il raggiungimento di una soluzione in termini di performance (ovvero tempo ed implementazione) e di progettazione.

Il sesto capitolo illustra le metodologie di Security Assessment utilizzate, con un focus particolare al framework OpenVas e a come quest'ultimo verrà utilizzato per testare la sicurezza dell'applicativo; inoltre, vengono descritte le pratiche di sicurezza adottate dalla piattaforma.

Il settimo ed ultimo capitolo, invece, illustra le mie considerazioni finali riguardanti l'esperienza di lavoro presso la Laser Romae s.r.l., cosa ho imparato dalla mia attività e gli sviluppi futuri della piattaforma OpenWhistleblowing 2.0.0.

2 Il quadro normativo whistleblowing

2.1 Il D.lgs. 190/2012

Le ripetute richieste rivolte all'Italia da parte di organismi internazionali con funzioni presidiarie dei fenomeni corruttivi e sull'onda della "marea nera" della crisi economico-finanziaria, innescata anche da comportamenti fraudolenti, hanno convinto il legislatore italiano ad emanare il D.lgs. 190/2012 recante «Disposizioni per la prevenzione e la repressione della corruzione e dell'illegalità nella Pubblica Amministrazione».

Per la prima volta appare, nell'ordinamento italiano, quell' "**istituto**" noto già ai paesi anglosassoni con il nome di **whistleblowing**, riconosciuto come elemento indispensabile di una corretta **corporate governance**, poiché strumento utilizzabile da soggetti che, sebbene abbiano ruoli e funzioni all'interno dei soggetti economici, privati e pubblici, devono attivarsi al fine di scongiurare il compimento di atti di corruzione.

La legge ha come obbiettivo tradurre in norma l'essenza dei **Whistleblowing Schemes**, ovvero la **disciplina della segnalazione** di condotte illecite, preoccupandosi di proteggere il segnalante da possibili ritorsioni, che risulteranno illegittime se determinate da motivi collegati, direttamente o indirettamente, alla sua segnalazione.

Inoltre, è mantenuta riservata l'identità del segnalante e non può essere rivelata senza il suo consenso, fatta eccezione per quanto accade in caso di segreto di Stato, esigenze di salvaguardia della sicurezza, a difesa nazionale, etc.

2.2 Il D.lgs. 179/2017

L'**Agenzia Nazionale Anti Corruzione** (**ANAC**), il 28 Aprile 2015, ha emesso le "Linee guida in materia di tutela del dipendente pubblico che segnala illeciti", con la chiara indicazione che le segnalazioni, al fine di tutelare il segnalante, debbano essere trattate con sistemi informatizzati e crittografici.

Da qui nasce il D.lgs. 179/2017, approvato il 15 Novembre 2017, a tutela del dipendente pubblico e privato. Questo decreto prevede la disposizione di almeno un canale alternativo di segnalazione idoneo a garantire, con modalità informatiche, la riservatezza del segnalante.

Una grande differenza con il precedente è l'**estensione della disciplina al settore privato**, infatti si rivolge anche a chi lavora in imprese che forniscono beni e servizi alla Pubblica Amministrazione.

L'ANAC, a cui l'interessato o i sindacati comunicano gli eventuali atti discriminatori (di cui nel D.lgs. precedente), applicano all'ente (se responsabile) una sanzione pecuniaria amministrativa da 5.000 a 30.000 euro; ne è prevista un'altra, da 10.000 a 50.000 euro, per il responsabile che non effettua le attività di verifica e analisi delle segnalazioni ricevute.

2.3 Direttiva 2013/36/UE

A seguito della crisi economico-finanziaria iniziata nel 2009, l'Unione Europea ha provveduto a costituire un nuovo corpus normativo organico in materia di regolazione, supervisione e risoluzione degli enti creditizi e delle imprese di investimento.

Con riferimento alla direttiva 2013/36/UE, sono state introdotte il 26 giugno 2013 ulteriori regolamentazioni riguardanti l'accesso all'attività e la vigilanza prudenziale delle/sulle banche, dividendo le competenze fra le Autorità di vigilanza interessate, Banca d'Italia e Consob.

Vengono quindi disciplinate a livello di normativa europea le modalità di segnalazione delle violazioni, interne agli intermediari e verso l'Autorità di vigilanza, tenendo conto dei profili di riservatezza e della protezione dei soggetti coinvolti; viene dedicata particolare attenzione anche alla disciplina delle sanzioni verso quelle società o enti nei cui confronti sono accertate le violazioni e i presupposti che determinano una responsabilità dei soggetti coinvolti (ovvero coloro che svolgono funzioni di amministrazione, direzione o controllo, dipendenti e coloro che hanno un rapporto diverso dal rapporto di lavoro subordinato).

L'entità della sanzione applicabile alla società in caso di violazione è compresa tra un minimo di 30.000 euro fino ad un massimo del 10% del fatturato; quella applicabile alle persone fisiche è compresa tra un minimo di 5.000 euro fino ad un massimo 5 milioni di euro.

2.4 Direttive 2015/849/UE e 2018/843/UE

Queste direttive rafforzano le norme UE destinate a prevenire il riciclaggio di denaro e il finanziamento del terrorismo e fanno parte di un piano d'azione dell'Unione Europea in risposta agli attacchi terroristici del 2015 e 2016 a Parigi e Bruxelles, nonché alle fughe di notizie provenienti dai "Panama Papers".

Queste normative introducono misure più severe, ampliando l'obbligo da parte delle entità finanziare di svolgere un'attività di "due diligence" nei confronti della propria clientela, e si rivelano essere di grande utilità per i Paesi in via di sviluppo e per la lotta contro i flussi illeciti di denaro in uscita.

Oltre ad imporre controlli più rigorosi su valute virtuali come Bitcoin e carte prepagate e il potenziamento dei controlli sulle operazioni che coinvolgono paesi ad alto rischio, viene introdotta una maggior protezione per gli informatori che segnalano un caso sospetto di riciclaggio o di finanziamento del terrorismo: essi sono tutelati legalmente da qualsiasi minaccia o atto ostile o di ritorsioni, specialmente in ambito lavorativo, e possono far affidamento su uno strumento informatico per l'invio della segnalazione dell'illecito.

3 Il processo di whistleblowing

A che cosa ci si riferisce quando si parla di **whistleblowing**? Il termine, come già anticipato, è anglosassone e si traduce in italiano con "soffiando il fischietto". Si pensa che l'origine sia dovuta al fatto che le forze dell'ordine erano solite soffiare nel fischietto per richiamare l'attenzione dei colleghi di fronte a un crimine o a una situazione di pericolo.

Parafrasando il termine, oggi si vuole indicare l'atto di "**denunciare**" sul luogo di lavoro.

Questo termine, introdotto negli Stati Uniti già nel 1863, è stato e viene utilizzato a livello globale per identificare tutti quei sistemi che vengono impiegati per segnalare illeciti che si verificano sul posto di lavoro.

L'applicativo di partenza OpenWhistleblowing 1.0.1 è stato realizzato per **l'Autorità Nazionale Anti Corruzione** (di seguito **ANAC**) facendo il porting da un prototipo realizzato su un'altra piattaforma tecnologica dalla società Laser Romae, la quale è risultata vincitrice di un bando di gara per la manutenzione ed evoluzione della piattaforma.

La piattaforma, oltre a consentire la segnalazione degli illeciti utilizzando modalità digitali, ottempera alle direttive **ANAC93** e alle norme di riferimento, ovvero **deve**:

- separare i dati identificativi del segnalante dal contenuto della segnalazione, prevedendo l'adozione di codici sostitutivi dei dati identificativi, in modo che la segnalazione possa essere processata in modalità anonima e rendere possibile la successiva ricostruzione dell'identità del segnalante solo nei casi consentiti;
- gestire le segnalazioni in modo trasparente attraverso un iter procedurale definito e comunicato all'esterno con termini certi per l'avvio e la conclusione dell'istruttoria;
- mantenere, per quanto possibile, riservato il contenuto delle segnalazioni durante l'intera fase di gestione della segnalazione;
- adottare protocolli sicuri per il trasporto dei dati in rete, nonché l'utilizzo di strumenti di crittografia per i contenuti delle segnalazioni e dell'eventuale documentazione allegata;
- adottare adeguate modalità di conservazione dei dati e della documentazione (fisico, logico, ibrido);
- adottare politiche di tutela della riservatezza attraverso strumenti informatici (disaccoppiamento dei dati del segnalante rispetto alle informazioni relative alla segnalazione e crittografia dei dati e dei documenti allegati);
- adottare politiche di accesso ai dati (funzionari abilitati all'accesso, amministratori del sistema informatico).

3.1 Gli attori

Il sistema di partenza prevede 3 attori principali:

- Il **segnalante** ("Whistleblower") ovvero colui che segnala l'illecito all'Ente o all'Amministrazione;
- L'**istruttore**, colui che gestisce l'istruttoria sapendo l'argomento ma non conoscendo, se non su specifica richiesta giustamente motivata, l'identità del segnalante;
- Il **custode** (generalmente il Responsabile Anticorruzione), colui che possiede le chiavi di accesso per permettere all'istruttore che ne faccia richiesta di visualizzare le informazioni di riconoscimento del segnalante.

Figura 1 – Attori del processo di whistleblowing

3.2 Il processo

Il processo di gestione della segnalazione, invece, si divide in 5 fasi di trattazione, cosi distinte:

- ricezione della segnalazione;
- esame della segnalazione (screening dei falsi positivi);
- istruttoria (e iterazione con il segnalante);
- definizione (chiusura della pratica) con esito di archiviazione o di inoltro;
- trasmissione agli uffici competenti in caso di inoltro.

3.3 Il risultato

Al termine del processo di whistleblowing, se l'istruttoria ha avuto esito positivo (e quindi non si tratta di un falso positivo), la segnalazione e le informazioni relative al segnalante (se sono state inserite) vengono trasmesse agli uffici competenti, che inizieranno un processo giudiziario sulla base della suddetta segnalazione.

Come si ricorda, l'azione penale in Italia è obbligatoria a differenza di altre Nazioni. Questo comporta una quasi certezza di trasmissione degli atti all'Autorità Giudiziaria, a meno di una palese segnalazione non attinente alla realtà.

4 La piattaforma OpenWhistleblowing 1.0.1

4.1 Introduzione

L'attuale piattaforma di whistleblowing OpenWhistleblowing 1.0.1 è stata pubblicata per il riuso in data 15/01/2019, sull'account GitHub dell'Autorità Nazionale Anticorruzione (https://github.com/anticorruzione/openwhistleblowing) ed è basata sul software GlobaLeaks 2.60.144, rilasciato in data 14 Marzo 2016.

GlobaLeaks è un prodotto open-source che ha lo scopo di permettere la segnalazione di illeciti in modo sicuro ed anonimo. Il software è stato sviluppato dall'Hermes Center for Transparency and Digital Human Rights.

4.2 Architettura

Di seguito un'immagine riassuntiva dell'architettura della piattaforma, che sarà vista più nel dettaglio nei paragrafi seguenti.

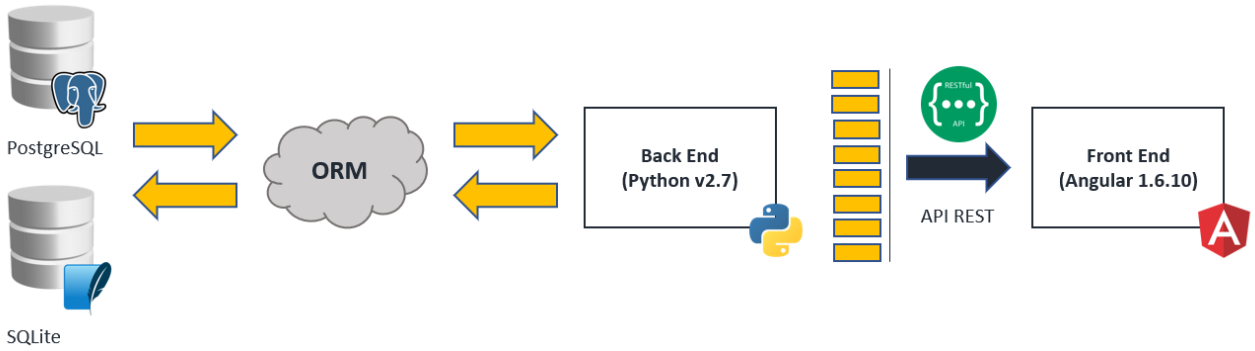

Figura 2 – Architettura della piattaforma OpenWhistleblowing 1.0.1

4.2.1 Linguaggi e framework

L'applicazione OpenWhistleblowing 1.0.1 si basa sul pattern MVC, un paradigma di programmazione che separa il software in 3 diverse componenti:

1. **Model** – rappresenta la business logic dell'applicativo: contiene i metodi di accesso ai dati e la definizione di quest'ultimi;
2. **View** – rappresenta l'user interface della piattaforma: si occupa di esporre all'utente il risultato della business logic, gestendo l'interazione fra quest'ultimo e l'infrastruttura sottostante;
3. **Controller** – rappresenta il livello di controllo: riceve le chiamate dell'utente attraverso la View ed esegue delle operazioni sul Model o comunque operazioni che portano un cambiamento nella View.

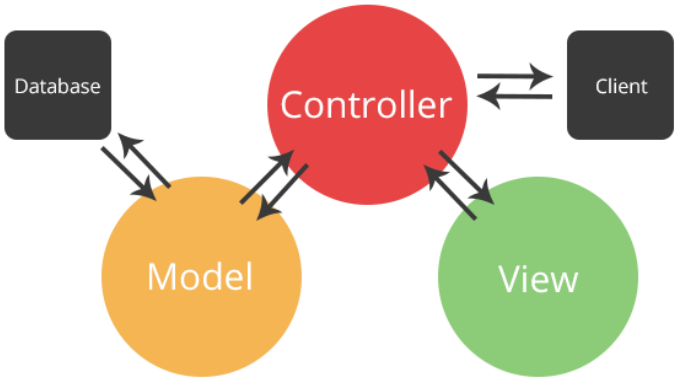

Figura 3 – Schema Model-View-Controller

Il pattern MVC implica la separazione fra la logica applicativa (business logic – a carico del Model e del Controller) e l'interfaccia utente (user interface – a carico della View).

I linguaggi utilizzati sono Javascript su framework Angular JS + Bootstrap per il frontend, mentre sono stati utilizzati Python, nella versione 2.7 con Twisted + Cyclone per il backend. Lo sviluppo è stato effettuato su sistema operativo Centos 7, mentre il rilascio su infrastruttura ANAC è stato effettuato su RedHat Linux 7.

Le componenti software possono essere così di seguito generalizzate:

4.2.2 Frontend

Qui risiede l'interfaccia client di GlobaLeaks: è una client-side web application scritta utilizzando il framework AngularJS, che comunica con il backend attraverso richieste HTTP. Essa è disponibile per tutti gli attori che interagiscono con la piattaforma.

AngularJS è un framework open source per single-page web applications, principalmente sviluppato da Google: è nato con lo scopo di semplificare lo sviluppo e il testing di questo tipo di applicazioni, fornendo un framework basato sul pattern cosiddetto MVW (Model-View-Whatever) – incoraggiando l'organizzazione del codice e la separazione dei compiti nei vari componenti.

Bootstrap, invece, è una raccolta di strumenti per la creazione di siti e applicazioni web compatibile con tutte le ultime versioni di tutti i principali browser: contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia che per le varie componenti dell'interfaccia, come ad esempio tabelle e pulsanti, implementando anche alcune estensioni opzionali di JavaScript. Il framework supporta il responsive web design (ovvero il layout delle pagine si regola dinamicamente), configurandosi come una libreria multidispositivo (desktop, tablet o telefono cellulare) e multipiattaforma (tutti i browser, fatta eccezione per quelli obsoleti come Internet Explorer 8 e 9).

4.2.3 Backend

Questa componente espone una interfaccia HTTP API, che permette la comunicazione fra i segnalanti e gli istruttori, e una interfaccia REST, che consente ad altre applicazioni di

interagirvi.

Qui risiede tutta la logica che permette il trattamento di nuove segnalazioni da parte dei segnalanti, la notifica di quest'ultime e l'interazione fra segnalanti ed istruttori. Questa API, inoltre, permette agli amministratori di sistema la configurazione del nodo GlobaLeaks.

Il backend inoltre, dovrebbe essere abbastanza flessibile da permettere la sua esecuzione e il suo funzionamento anche se alcune delle sue componenti sono rimosse o non configurate (come il servizio di notifica via email ad esempio).

4.2.4 Tor hidden service

Al fine di preservare l'identità del segnalante, il sistema si integra con la rete Tor, esponendo il servizio su tale rete. Tale configurazione permette l'anonimizzazione dell'indirizzo IP da cui si collega il segnalante.

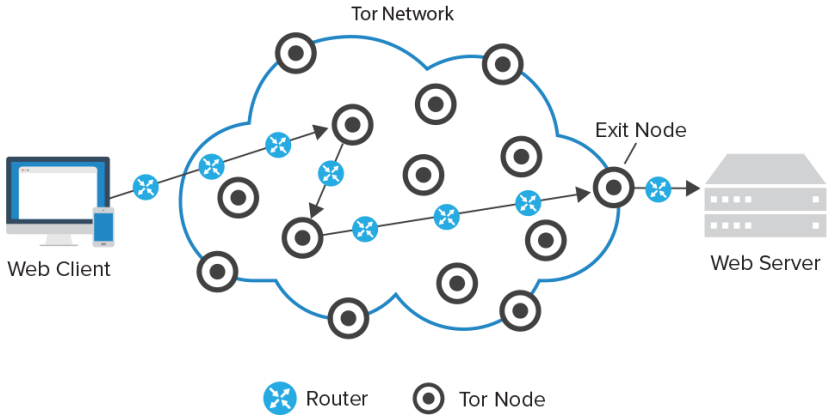

Figura 4 – Funzionamento della rete Tor

Tor, acronimo di The Onion Router, è una rete che garantisce l'anonimato grazie all'utilizzo di alcuni nodi particolari, gli Onion Routers: il traffico dei dati viene instradato in un circuito virtuale crittografato instaurato tra un numero N di nodi. Il protocollo di comunicazione di Tor è stato sviluppato per garantire l'integrità dei dati, la segretezza della comunicazione e il controllo della congestione della rete.

Lo scopo principale della rete Tor è quello di disaccoppiare l'utente dalla destinazione richiesta: i pacchetti, che contengono i dati della comunicazione, non vengono instradati lungo un percorso prevedibile, ma bensì lungo uno casuale e dinamico attraverso Onion Routers che ignorano la sorgente e la destinazione finale. Per evitare qualsiasi MITM attack (Man In The Middle attack), ogni router comunica con quello successivo tramite lo scambio di coppie di chiavi crittografate.

I punti di maggiore criticità di una rete Tor sono il nodo di entrata, detto **Guard Node** – nodo di guardia, e il nodo di uscita, detto **Exit Node**: questi due nodi sono necessariamente fidati.

Esistono infine altri tipi di nodi fidati che svolgono la funzione di **Directory Server**: detengono le liste dei router e le informazioni sullo stato della rete, fornendoli ai client Tor

che le richiedono.

Attraverso il **Tor2Web** node (che verrà esposto in seguito), la piattaforma espone il proprio Tor Hidden Service (**THS**). Ciò garantisce un anonimato **bidirezionale** con il segnalante. Il server non conosce l'indirizzo IP dell'utente, ma a sua volta quest'ultimo non conosce l'indirizzo IP del server.

Un THS è un indirizzo **.onion,** che la rete Tor identifica e permette di creare un proxy che reindirizza l'utente al servizio anonimo che si "nasconde" dietro a quell'indirizzo.

4.3 Il software GlobaLeaks

GlobaLeaks è basato su due elementi fondamentali: **node** e **context,** necessari per capire il successivo workflow e gli altri elementi da cui discendono e sono collegati.

4.3.1 Node

Il node rappresenta una istanza di GlobaLeaks, quindi contiene in sé tutte quelle informazioni generali necessarie sia per la configurazione che per il funzionamento della piattaforma. Le informazioni del nodo vengono esposte dal backend attraverso le API di tipo REST.

L'amministratore di sistema può modificare nell'interfaccia web di amministrazione il nome del nodo (utilizzato nel template delle notifiche e in ogni pagina) e la descrizione di quest'ultimo (utilizzata nella homepage). Altri testi rappresentati nella User Interface (UI) sono "hard-coded" nelle pagine HTML, quindi per modificarli è necessario intervenire direttamente sul codice del client.

Dalla stessa interfaccia web, l'amministratore può modificare le impostazioni delle notifiche predefinite (nuovo commento, nuovo messaggio, nuova segnalazione, etc.): di default il servizio di notifica di GlobaLeaks prevede le mail, quindi è necessario configurare un server SMTP (Simple Mail Transfer Protocol) ed attivare una casella di posta per permettere l'invio delle notifiche. Il sistema non prevede alla versione 2.60.144, un protocollo di sicurezza che non sia TLS o SSL. Le email inviate vengono cifrate se l'istruttore che le deve ricevere ha impostato la propria chiave pubblica PGP (Pretty Good Privacy) nelle proprie preferenze.

Infine, l'amministratore può effettuare le funzioni CRUD (Create, Read, Update, Delete – Creare, Leggere, Modificare, Cancellare) sugli utenti; può gestire le impostazioni dei singoli istruttori, ovvero se sono abilitati, se possono ricevere notifiche, etc.; può gestire i **contexts**, di cui nel paragrafo in seguito.

4.3.2 Context

Un node può gestire uno o più context, che rappresentano la descrizione logica dell'argomento gestito dagli istruttori. Ogni contesto ha una differente configurazione iniziale, statistiche associate e una lista degli istruttori. Generalmente l'admin sceglie un context (ad esempio "la corruzione a Roma" o "l'inquinamento ambientale a Taranto"), ne dà una descrizione che ne identifica l'iniziativa, configura il formato delle segnalazioni (ad

esempio “descrizione-luogo-data”), definisce se esse possano avere o meno dei file allegati ed infine configura la lista degli istruttori. L’admin inoltre imposta altre proprietà per il context, come il metodo di notifica per gli istruttori, il tempo di scadenza per le segnalazioni e il tipo di dati/file attesi.

4.3.3 Workflow

Il workflow del software GlobaLeaks si può riassumere nei seguenti passaggi:

- un whistleblower sceglie il contesto della segnalazione ed inizia a compilarla utilizzando una serie di form in cui è possibile compilare dei campi e allegare dei files;
- una volta premuto il tasto per la sottomissione della segnalazione, il software genera un codice unico e lo restituisce al whistleblower;
- utilizzando il codice di cui al punto precedente, il segnalante potrà accedere e visualizzare la segnalazione;
- una volta entrato nella sua segnalazione, il segnalante potrà caricare ulteriori file, leggere/scrivere commenti e avere una comunicazione asincrona con l’istruttore che ha in carico la sua segnalazione.

4.3.4 Segnalazione

La segnalazione è un oggetto che si può trovare in due stati fondamentali:

- lo stato di inserimento, in cui il segnalante inserisce i dati. Per evitare sovraccarichi, ovvero l’inserimento di segnalazioni non valide, il sistema è fornito di una feature anti-spam, che permette di configurare una protezione CAPTCHA che viene attivata quando un certa soglia submission/tempo viene raggiunta;
- lo stato di inserita, in cui il segnalante possiede una receipt (ovvero il codice fornito a fine inserimento) che gli permette di visualizzare la propria segnalazione.

4.3.5 Receipt

É la stringa segreta che consente al segnalante di accedere alla propria segnalazione: è necessaria se quest’ultimo vuole modificare o aggiornare il materiale contenuto, o se vuole interagire con gli istruttori. Una receipt è generata automaticamente dal node al termine di una sottomissione ed è fornita al segnalante attraverso una schermata, visualizzata una e una sola volta.

Una receipt è generata casualmente tramite una fonte sicura e il formato è specificato da un’espressione regolare inversa. Viene fornita sotto forma di codice di carta di credito per favorirne la memorizzazione.

4.3.6 Fields

Rappresentano i dati contestuali associati ad una segnalazione. Essi sono forniti tramite un API che abilita il client a renderizzarli con un nome e una descrizione. I client possono opzionalmente fornire un campo aggiuntivo per una identificazione lato client del

segnalante.

4.3.7 Comments

Ogni sottomissione permette, come detto in precedenza, una comunicazione asincrona fra il segnalante e i relativi istruttori. Questi ultimi possono sfruttare questa funzionalità per ottenere maggiori informazioni sulla segnalazione o per chiedere al segnalante di allegare nuovo materiale. Ogni commento contiene il nome dell'autore (l'username dell'istruttore o il generico "Whistleblower").

4.4 Analisi e considerazioni

L'azienda Laser Romae s.r.l. mi ha dato l'opportunità di analizzare l'applicativo di partenza "sul campo" attraverso lo sviluppo, la manutenzione e la customizzazione della piattaforma stessa per l'Autorità e per altri amministrazioni pubbliche, sia centrali che locali, che usufruiscono dell'applicativo (Banca d'Italia ed AMA): durante queste attività sono emersi i punti di forza e le criticità della piattaforma, anche grazie alle segnalazioni di bug dei clienti.

I punti di forza riscontrati nella piattaforma OpenWhistleblowing 1.0.1 sono i seguenti:

- Facilità e velocità di deployment – dovuti alla struttura monolitica dell'applicazione;
- Customizzazione del questionario;
- Internazionalizzazione.

Sono molte di più, invece, le criticità riscontrate:

- Indisponibilità alla cifratura del database – l'applicativo, as-is, supporta solamente db di tipo SQLite e Postgres non cifrati;
- Accessibilità ad informazioni sensibili all'interno del DBMS – la struttura dati del software GlobaLeaks presenta numerose tabelle, volte a scindere il più possibile le informazioni sensibili; queste tabelle tuttavia sono collegate fra di loro tramite degli UUID, perciò un utente malevolo, con accesso (ottenuto forzatamente o meno) al database, può ricostruire le informazioni sensibili presenti al suo interno seguendo questi identificativi;
- Mancanza di una struttura modulare – il Frontend della piattaforma, che quando viene buildato è inserito in un singolo file denominato **scripts.js**, è una componente monolitica;
- Circolazione di informazioni sensibili – nei JSON inviati al Client spesso compaiono informazioni sensibili e talvolta inutili alla costruzione dinamica della pagina a cui si è appena fatto accesso: ad esempio è possibile vedere gli username degli istruttori, la cui password di default è **globaleaks**, con i quali è possibile ridurre del 50% l'effort di un eventuale attacco brute-force (o del 100% nel caso in cui l'istruttore, per dimenticanza o per mancanza di tempo, non avesse cambiato la password di default);
- Linguaggio utilizzato in dismissione – la versione Python 2.7 a cui fa riferimento il

backend della piattaforma è in dismissione (end of support) il 01/01/2020;

- ORM deprecato – il framework utilizzato per l'ORM, **Storm**, è deprecato da prima ancora che la piattaforma stessa venisse consegnata ad ANAC sotto forma di prototipo;
- Latenza invio/ricezione richieste HTTP – per via delle scelte implementative sbagliate elencate precedentemente (Frontend monolitico, ORM deprecato) viene riscontrata un'eccessiva latenza nella comunicazione fra Server e Client, con tempi dell'ordine di secondi;
- Assenza supporto sistemi di autenticazione – la piattaforma non prevede alcun tipo di integrazione per sistemi di autenticazione utenti in contesti più evoluti (ad esempio LDAP/AD, OAuth2, SAML) spesso utilizzati dalla pubblica amministrazione.

Queste sono le motivazioni che hanno spinto l'Autorità Nazionale Anticorruzione e l'azienda Laser Romae s.r.l. a progettare e sviluppare un nuovo software per la segnalazione degli illeciti nella pubblica amministrazione.

5 La piattaforma OpenWhistleblowing 2.0.0

5.1 Introduzione

La nuova piattaforma OpenWhistleblowing 2.0.0 (di seguito OWB2) si basa su alcuni concetti fondamentali, alcuni dei quali esplorati solo in parte nella piattaforma GlobaLeaks:

- **Sicurezza**
- **Modularità**
- **Semplicità**
- **Usabilità**
- **Integrabilità**
- **Low-Requirements**

OWB2 nasce con lo scopo di svincolarsi dal precedente e limitante framework, mantenendone la vision con cui quest'ultimo è stato sviluppato – ovvero fornire uno strumento informatico al whistleblower per segnalare un illecito nel rispetto delle normative vigenti – e alcuni degli elementi principali, come il workflow della creazione di una segnalazione o la receipt.

5.2 Architettura

L'applicazione OpenWhistleblowing 2.0.0 si basa sul pattern MVC ed è divisa in due componenti principali (Backend e Frontend) che verrano analizzate nei paragrafi seguenti.

Di seguito un'immagine riassuntiva dell'architettura della piattaforma.

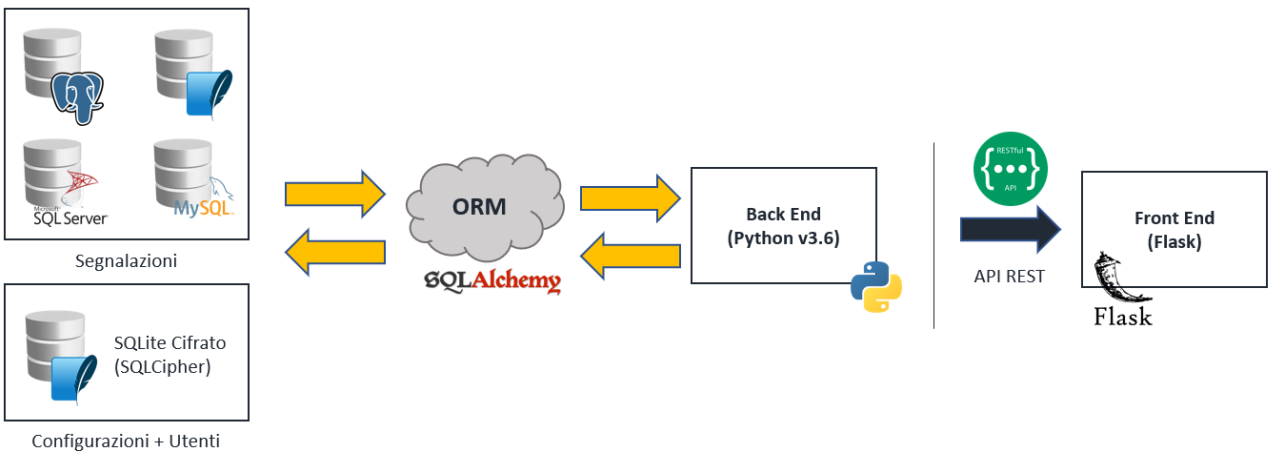

Figura 5 - Architettura della piattaforma OpenWhistleblowing 2.0.0

5.2.1 Linguaggi e framework

La prima scelta implementativa che è stata realizzata consiste nell'abbandono di Python 2.7 (ultima versione rilasciata mid-2010 – il supporto terminerà nel 2020 [https://pythonclock.org/]) a favore di Python 3.6.

Le features di default (lista non esaustiva) che sono disponibili solo nelle versioni 3.x (e che non sono state riportate sulle versioni 2.x) sono le seguenti:

- Le stringhe sono considerate come Unicode di default
- Separazione distinta fra Unicode e bytes
- **Exception chaining** – nelle versioni 2.x durante l'error handling di un'eccezione A poteva verificarsi un'altra eccezione B, causando una propagazione di quest'ultima verso l'esterno ed una perdita delle informazioni utili legate all'eccezione A. Nelle versioni 3.x il supporto per questo tipo di situazioni è implementato di default, aumentando la possibilità di debugging del codice (**PEP 3134**);
- **Keyword-only arguments** – proprio come i positional arguments, i keyword arguments possono essere specificati nella chiamata di una funzione o utilizzando il loro ordine o specificando il loro nome (**PEP 3102**).

Questi sono solo alcuni dei cambiamenti fra le due versioni e sicuramente l'evoluzione del linguaggio non si limita a cambiamenti sintattici o semantici: sono numerosi anche i cambiamenti alla standard library di Python 3.x (disponibili per la consultazione qui https://docs.python.org/3/whatsnew/) e al Python Package Index (PyPI), conosciuto anche come **The Cheese Shop**, ovvero la repository ufficiale per i third-party softwares.

Per sviluppare la piattaforma OWB2 si è scelto di utilizzare come framework di base **Flask**. Flask fu inizialmente sviluppato da Armin Ronacher come un pesce d'aprile nel 2010, ma nonostante ciò crebbe in popolarità molto velocemente, tanto da porsi come alternativa a **Django** e alla sua struttura monolitica.

Figura 6 – Flask microframework

Flask è considerato un **microframework** perché è stato progettato per essere semplice ed estendibile: l'idea dietro è quella di avere uno strumento solido per sviluppare web applications di qualsiasi difficoltà. Le motivazioni che mi hanno portato a scegliere questo framework, oltre ai motivi precedentemente citati che hanno portato alla rimozione di GlobaLeaks, sono le seguenti:

- possiede un server di sviluppo, un debugger e un modulo per il testing integrati;
- la documentazione è comprensiva e ben strutturata;
- RESTful request dispatching – la gestione delle API di base è coerente;
- HTTP request handling – le funzionalità per gestire le richieste HTTP sono intuitive e semplici da utilizzare;

- WSGI 1.0 compliant – sfrutta il protocollo di trasmissione WSGI (Web Server Gateway Interface) che stabilisce e descrive le interazioni tra server e applicazioni web in Python, facilitando il deploy dell'applicazione in produzione;
- ORM-agnostic – non integra di default nessun ORM, lasciando carta bianca al programmatore;
- High flexibilty – possiede un core semplice ma estendibile con le numerose librerie disponibili per il framework;
- Frontend support – Jinja2 (un linguaggio di templating per Python) e gestione sicura dei cookie integrati.

Sebbene Flask sia relativamente giovane, è uno dei migliori framework disponibili per la realizzazione di web applications grazie all'elevato numero di features e alla forte propensione all'estensione.

5.2.2 Frontend

Qui risiede l'interfaccia client della nuova piattaforma: è una client-side web application scritta utilizzando il framework Flask, che comunica con il backend attraverso richieste HTTP e costruisce dinamicamente l'interfaccia utente per ogni attore del sistema utilizzando i templates Jinja2 sulla base del risultato delle chiamate precedenti.

Per quanto riguarda il design del Frontend, l'applicativo fa riferimento alle "Linee guida di design servizi digitali della PA", un sistema condiviso di riferimenti progettuali e visivi relativi al design dei siti e dei servizi dalla Pubblica Amministrazione (sviluppato da AGID, Agenzia per l'Italia Digitale – sono reperibili al link https://www.agid.gov.it/it/design-servizi/linee-guida-design-servizi-digitali-pab): queste linee guida hanno lo scopo di migliorare e rendere coerente la navigazione e l'esperienza del cittadino online in quanto utente di un sito web di una pubblica amministrazione.

5.2.3 Backend

Questa componente espone una interfaccia HTTP API, che permette la comunicazione fra i segnalanti e gli istruttori, e una interfaccia REST, che permette ad altre applicazioni di interagirvi.

Tramite queste interfacce l'amministratore è in grado di configurare le impostazioni di sistema dell'applicativo: può effettuare modifiche sull'interfaccia grafica o può abilitare/disabilitare i vari moduli della piattaforma.

Il backend, infine, gestisce in modo asincrono alcune task da eseguire in background, come ad esempio il servizio di notifica dashboard/mail per gli utenti o il servizio che effettua l'integrity check, grazie a due sistemi che si possono definire complementari: Celery e Redis. Il primo è un sistema distribuito in grado di processare code di task e lo scheduling di quest'ultime; il secondo, invece, è un DBMS NoSQL che funge da message broker.

L'amministratore della piattaforma può monitorare in tempo reale lo stato di tutte le task

asincrone dell'applicazione, segnalando eventuali problemi riscontrati durante l'esecuzione di quest'ultime, le può controllare in via remota (quindi potrà effettuare azioni come lo shutdown o il restart di una di esse) e può visionare statistiche relative ai tempi di esecuzione e risultati.

5.2.4 Nuove features

Le nuove features presenti nella nuova versione OpenWhistleblowing 2.0.0 sono quelle descritte nei successivi paragrafi.

5.2.4.1 Supervisore

E' previsto un nuovo ruolo per l'applicativo: il **Supervisore**. Questo tipo di utenza riceve tutte le nuove segnalazioni, è in grado di smistarle verso gli istruttori e, di conseguenza, visualizzarle tutte e riassegnarle ad un istruttore differente in caso di necessità (ferie, malattia, etc.).

5.2.4.2 Rimozione del single point of failure

Le segnalazioni degli illeciti non vengono più salvate sul database (che conterrà solo le informazioni relative agli utenti e alle notifiche) ma vengono criptate e salvate in un file .json su filesystem: questa modifica è volta ad eliminare il forte **single point of failure** (**SPOF**) della versione precedente; inoltre, le informazioni relative all'identità del segnalante sono salvate su un file .json separato da quello con il contenuto della segnalazione e criptato con la chiave pubblica del Custode.

5.2.4.3 Logging locale e centralizzato

Il logging è previsto in locale, ma anche mediante l'utilizzo di sistemi avanzati per il log centralizzato come Graylog3, Elasticsearch o Fluentd:

1. **Graylog3** – è uno strumento open-source basato su Java che può essere utilizzato per raccogliere, indicizzare ed analizzare qualsiasi log server da una postazione centralizzata, fornendo un linguaggio di query avanzate, abilità di avviso e una pipeline per la trasformazione dei dati. Graylog3 è costituito da 3 componenti: **MongoDB**, un DBMS non relazionale orientato ai documenti; **Graylog server**, che riceve ed elabora i messaggi dai vari input e fornisce un'interfaccia web per l'analisi e il monitoraggio; **Elasticsearch**, di cui in seguito;
2. **Elasticsearch** – è un server di ricerca basato su **Lucene**, una API gratuita ed open-source realizzata in Java. Tutte le funzionalità di Elasticsearch sono esposte nativamente tramite un'interfaccia RESTful e le informazioni sono gestite come dei documenti JSON;
3. **Fluentd** – è un progetto software open-source multipiattaforma per la raccolta di dati, scritto principalmente in Ruby, che permette l'analisi di log di eventi, log di applicazioni e clickstreams.

5.2.4.4 Database

Il database di base è un SQLite criptato tramite la libreria **SQLCipher** e gestito tramite Python con pysqlcipher per aumentare la sicurezza dell'applicativo.

SQLCipher è un'estensione per SQLite (utilizzata da molte organizzazione ed aziende, fra cui **Samsung**, **NASA**, **UBS**) che fornisce encrypting AES 256-bit (in modalità **CBC**, **Cipher Block Chaining**) per il database: ogni pagina di quest'ultimo è crittata e decrittata individualmente (la dimensione della pagina di default è 4096 bytes, modificabile anche a runtime per migliorare le prestazioni di determinate query); include un **Initialization Vector** randomico, generato crittograficamente tramite un secure random number generator (**OpenSSL** RAND_bytes), salvato nel fondo della pagina e rigenerato ad ogni scrittura, così che non venga mai utilizzato lo stesso IV per scritture consecutive sugli stessi dati, ed un **Message Authentication Code** (**HMAC-SHA512**) per il testo cifrato e l'IV. Il MAC viene controllato ogni volta che la pagina viene letta e se il testo cifrato o l'IV sono stato manomessi o corrotti, il controllo HMAC ritornerà un errore.

Quando il database viene inizializzato con una passphrase, SQLCipher deriva la chiave utilizzando **PBKDF2-HMAC-SHA512**: ogni database è inizializzato utilizzando un salt unico e randomico nei primi 16 bytes del file, i cui scopi sono la derivazione della chiave e la garanzia che anche se due database sono inizializzati con la stessa password, non avranno mai la stessa chiave crittografica. La configurazione di default prevede 256.000 iterazioni per la derivazione della chiave (anch'essa modificabile a runtime tramite PRAGMA kdf_iter), che è differente dalla chiave utilizzata per calcolare gli HMACs di ogni pagina. Inoltre, la memoria allocata da SQLCipher è "lockata" quando viene utilizzata (tramite **mlock/VirtualLock**) e cancellata prima che venga liberata.

5.2.4.5 Workflow

La gestione del workflow è interna alla piattaforma, ma può anche essere delegata verso plugin esterni come ad esempio **Activiti**, un workflow engine open-source scritto in Java in grado di eseguire processi aziendali descritti da BPMN 2.0 (Business Process Model and Notation – una rappresentazione grafica per specificare processi di business in un modello) utilizzato dall'ANAC.

Il workflow degli stati della segnalazione permette i seguenti passaggi di stato:

- Non ancora presa in carico -> Presa in carico
- Presa in carico -> In istruttoria
- In istruttoria -> Presa in carico
- In istruttoria -> Archiviata
- In istruttoria -> Inoltrata alle autorità competenti
- Inoltrata alle autorità competenti -> Archiviata

Di seguito un'immagine riassuntiva con un esempio di workflow con cambio istruttore durante lo stato di istruttoria.

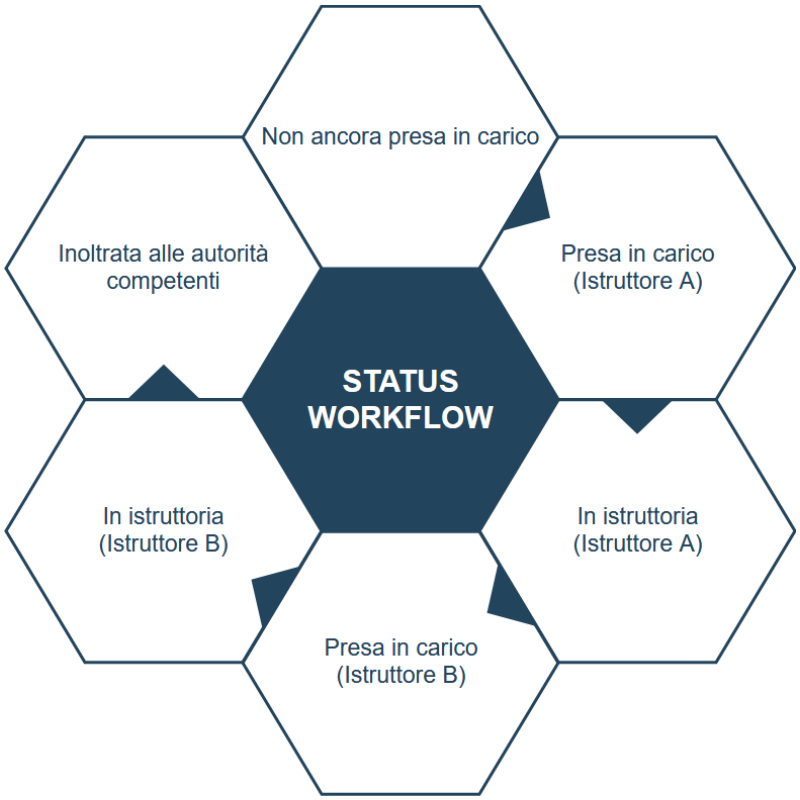

Figura 7 – Esempio del workflow degli stati di una segnalazione

5.2.4.6 Sistemi per l'autenticazione utente

Oltre alla classica autenticazione utente tramite username e password, sono stati implementati dei sistemi per facilitare il login agli utenti già censiti in un'azienda:

- **LDAP** – acronimo di Lightweight Directory Access Protocol, è un insieme di protocolli open usati per accedere alle informazioni conservate centralmente attraverso una rete. Lo standard LDAP viene talvolta indicato come una directory contente informazioni sulla categoria e sulla gerarchia – tra le suddette informazioni si possono trovare nomi, indirizzi e-mail, numeri di telefono, etc. Il beneficio principale dell'utilizzo di LDAP, da parte di un'azienda, è quello di mantenere tutte le informazioni consolidate in un deposito centrale, ovvero come una directory centrale accessibile da qualsiasi posizione della rete. LDAP, inoltre, supporta SSL (**Secure Sockets Layer**) e TLS (**Transport Layer Security**) e possiede una API ben definita utilizzabile per l'autenticazione/autorizzazione degli utenti;
- **SAML** – acronimo di Security Assertion Markup Language, è uno standard informatico per lo scambio di dati (formato XML) di autenticazione e autorizzazione. SAML si basa sul fatto che l'utente sia registrato presso almeno un identity provider, che deve provvedere ad autenticare l'utente;
- **OAuth2** – contrazione di Open Authorization 2, è un protocollo universale nato dal progetto SAML di cui condivide l'idea di base, ovvero la possibilità di

autenticare/autorizzare utenti senza passare credenziali ma tramite un identity provider esterno (ad esempio Google).

5.2.4.7 ORM

Per lo sviluppo della nuova piattaforma è stato rimosso il vecchio ORM Storm a favore di **SQLAlchemy**: questo toolkit fornisce una suite completa di modelli di persistenza, progettati per un accesso al database efficiente ed ad alte prestazioni, adatto per un dominio semplice come quello dell'applicativo in questione.

I database SQL, comportandosi come delle raccolte di oggetti, più aumentano le dimensioni più le prestazioni e l'astrazione diventano importanti: lo scopo di SQLAlchemy è quello di soddisfare entrambi questi principi.

SQLAlchemy, infatti, considera il database non solo come un insieme di tabelle, ma come un motore algebrico relazionale: le righe possono essere selezionate non solo dalle tabelle, ma anche tramite istruzioni di join e select; ognuna di queste unità può anche essere composta in una operazione più complessa.

L'approccio di SQLAlchemy è completamente diverso da quello della maggior parte degli strumenti SQL/ORM, radicati in un approccio detto "**complimentarity**": invece di nascondere i dettagli SQL e gli oggetti relazionali dietro ad un "muro di automazione", tutti i processi sono completamente esposti all'interno di una serie di strumenti trasparenti e, al tempo stesso, compatti. La libreria svolge il compito di automatizzare le attività ridondanti, mentre lo sviluppatore mantiene il controllo su come è organizzato il database.

5.3 Sviluppo

Per lo sviluppo della piattaforma si è fatto ricorso ad una CI/CD pipeline (continuous integration / continuous delivery) per fornire un elevato grado di automazione al flusso di lavoro e al tempo stesso un aumento sia della velocità che della qualità dei diversi processi di sviluppo coinvolti.

Uno degli strumenti più importanti durante la creazione di una CI/CD pipeline è il **source code repository** (letteralmente "deposito del codice sorgente"): la piattaforma utilizzata per gestire il repository **git** della nuova piattaforma OpenWhistleblowing è **Gitlab**.

Git è un **DVCS** (**Distributed Version Control System**), cioè un sistema software per il controllo di versione distribuito – scritto da Linus Torvalds, il padre di Linux. Il **versioning** (controllo di versione) è un meccanismo tramite il quale è possibile tenere traccia delle modifiche apportate ad un progetto con il passare del tempo: si tende ad associare il concetto di versioning allo sviluppo di applicazioni o all'implementazione di diverse release, tuttavia è possibile associarlo a qualsiasi file sia suscettibile di modifiche.

L'utilizzo di un VCS consente di semplificare il lavoro in team, fornendo ai componenti di quest'ultimo gli strumenti per monitorare i cambiamenti effettuati, per annullare una

variazione o per ripristinare la release di un file o di un progetto ad una versione precedente.

Esistono diversi tipi di VCS – LVCS (Local VCS), CVCS (Centralized VCS) e DVCS (Distributed VCS) – ma mi soffermerò solo su l'ultimo di questi che è il tipo più vantaggioso e quello utilizzato nella realizzazione della piattaforma OpenWhistleblowing 2.0.0.

I repository sono delle entità fondamentali nel funzionamento di qualsiasi VCS, ma diventano ancora più rilevanti nei DVCS: in questo tipo di sistema, è previsto che il controllo degli ultimi snapshot generati ("istantanee" dei file allo stato corrente) sia affidato ai client che eseguiranno delle copie complete dei repository; in questo modo, nel caso di un'interruzione del servizio da parte dei server, il repository presente in qualsiasi client potrà essere utilizzato per il ripristino delle informazioni. Tutto ciò è possibile mediante la funzionalità denominata **checkout**, che permette la creazione di backup completi – una copia di un repository locale o remoto tramite un comando di clonazione.

A differenza dei sistemi centralizzati per il controllo versione, i DVCS superano l'impostazione gerarchica eliminando quel limite a carico del livello di produttività che, in tali casi, si può rilevare determinante nella gestione del flusso di lavoro: grazie alla possibilità di operare su più repository in remoto, si ha la possibilità di lavorare in contemporanea sullo stesso progetto con gruppi di collaboratori differenti (e magari seguendo metodologie diverse).

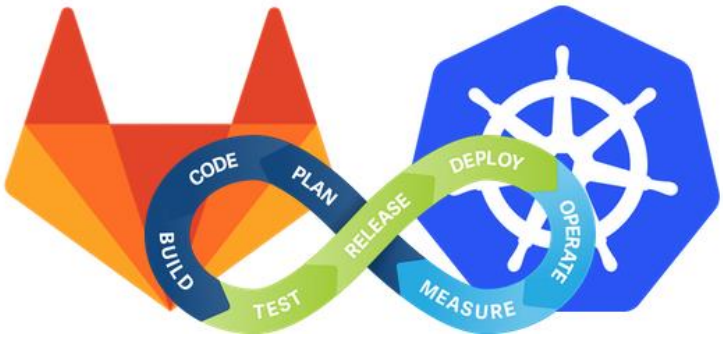

Figura 8 – Gitlab e Kubernetes

L'altro attore fondamentale della pipeline è **Kubernetes**, un sistema open-source di gestione di containers sviluppato da Google, il cui hardware si può rappresentare nei seguenti elementi:

- **Node** - è la più piccola unità computazionale di Kubernetes e rappresenta la singola macchina (o fisica in un datacenter, o virtuale in cloud) dentro il cluster. Ciò fornisce un livello di astrazione sulla macchina, definendola semplicemente come un set di risorse CPU e RAM che possono essere utilizzate: grazie a questa visione, qualsiasi macchina può essere sostituita da un'altra in cluster;
- **Cluster** - i nodi raggruppano le proprie risorse per formare una macchina più potente, il cluster: in questo modo quando si distribuiscono più programmi sul cluster, questo in modo intelligente distribuisce a sua volta il lavoro ai singoli nodi. Se viene

aggiunto o rimosso un nodo, il cluster provvederà automaticamente a distribuire il carico in modo bilanciato, astraendo il programmatore da quali singole macchine stiano effettivamente eseguendo il codice;

- **Persistent Volumes** - per memorizzare i dati in modo permanente, Kubernetes utilizza i persistent volumes. Mentre le risorse CPU e RAM di ogni nodo sono effettivamente raggruppate e gestite dal cluster, l'archiviazione non lo è - se si salvano dei dati su un nodo e poi quel nodo viene rimosso, si sono persi per sempre: per questo motivo, si possono collegare unità locali o cloud come un volume persistente, che può essere pensato come collegare un disco rigido esterno al cluster, senza essere associato a nessun nodo in particolare.

Lato software, invece, gli elementi fondamentali di Kubernetes sono:

- **Containers** - i programmi eseguiti su un cluster sono trasformati in container. In ognuno di essi possono essere inseriti più processi, ma è una best practice inserire un solo processo per container, per velocizzare l'implementazione, l'aggiornamento e la diagnostica dei problemi per ciascuno di essi;
- **Pod** - a differenza di altri sistemi per la gestione di containers, Kubernetes non esegue direttamente quest'ultimi ma li racchiude in una struttura di livello superiore: il pod. All'interno di esso, i containers condividono le risorse e la rete locale, facilitando la comunicazione fra essi come se si trovassero sulla stessa macchina, mantenendo allo stesso tempo un certo livello di isolamento dagli altri. Una singola instanza di un pod è in grado di mantenere un'applicazione, però al tempo stesso è standard avere più copie di un pod in esecuzione in qualsiasi momento in un sistema di produzione per consentire un migliore **load balancing** e **failure resistance**. Ogni pod viene ridimensionato (in giù o in su) in base alle esigenze: di conseguenza, tutti i suoi container devono scalare insieme, indipendentemente dalle loro esigenze individuali. Ciò può comportare degli sprechi, riducibili mantenendo ridotte le dimensioni del pod e accoppiando dei containers di supporto, generalmente indicati come **side-car**;
- **Deployments** - sebbene i pod siano l'unità di base di calcolo di Kubernetes, essi non vengono avviati direttamente sul cluster ma vengono gestiti da un ulteriore livello di astrazione: il **deployment**. Lo scopo principale di questa unità è di dichiarare quante repliche di un pod devono essere eseguite contemporaneamente. Ogni volta che viene aggiunto un deployment ad un cluster, verrà automaticamente lanciato e monitorato il numero richiesto di pod. Grazie al deployment, non è necessario gestire manualmente i pod: se uno di essi muore, sarà compito suo farne ripartire un altro;
- **Ingress** - Kubernetes di default fornisce l'isolamento tra i pod ed il mondo esterno. Se si desidera comunicare con un servizio in esecuzione in un pod, è necessario aprire un canale per la comunicazione: un **ingress**. Il metodo più semplice per aggiungere un ingress ad un cluster è tramite un controller ingress o tramite un load balancer.

Poiché tutti gli elementi della piattaforma vengono containerizzati grazie a Docker ed al relativo Dockerfile, è possibile collegare la repository ad un Kubernetes cluster, che si occuperà di tenere in vita i pod dell'applicativo e di aggiornarlo in caso di aggiornamenti.

La configurazione della pipeline CI/CD di Gitlab è gestita dal file .gitlab-ci.yml, di cui un esempio:

```
before_script:
    - echo "$REGISTRY_PASSWORD" | docker login -u
"$REGISTRY_USER" --password-stdin
build_image:
    script:
        - docker build --compress --force-rm --rm -t
    laserromae/whistleblowing:$CI_COMMIT_REF_NAME .
        - docker push
    laserromae/whistleblowing:$CI_COMMIT_REF_NAME
```

Le variabili $REGISTRY_USER e $REGISTRY_PASSWORD sono le credenziali di Gitlab, mentre la variabile $CI_COMMIT_REF_NAME fa riferimento al nome del branch della repository: master, development o test.

5.4 Testing

Il framework utilizzato per il testing della piattaforma è il modulo **unittest** di Python: questo framework supporta l'automazione, l'indipendenza e l'aggregazione in raccolte dei test.

Per fare ciò, unittest supporta alcuni concetti importanti nella programmazione orientata agli oggetti:

- **test fixture** - rappresenta la preparazione necessaria per eseguire uno o più test (con conseguente azione di cleanup). Ciò può comportare, ad esempio, la creazione di database temporanei, directories o l'avvio del server stesso;
- **test case** - la singola unità di test. Ha lo scopo di controllare una risposta specifica dato un particolare insieme di input; il framework fornisce una classe base, **TestCase**, che può essere utilizzata per creare nuovi test case;
- **test suite** - è una raccolta di test cases, test suites, o entrambi e viene utilizzata per aggregare tutti quei tests che dovrebbero essere eseguiti insieme;
- **test runner** – è il componente che orchestra l'esecuzione dei test e fornisce il risultato all'utente. Il runner può utilizzare un'interfaccia grafica, un'interfaccia testuale o restituire un valore speciale per indicare il risultato dell'esecuzione dei tests.

Tutti i test case della piattaforma sono presenti sotto la directory **tests/** e per essere eseguiti bisogna lanciare da linea di comando la seguente riga:

```
python -m unittest tests/run.py
```

Se si vuole aumentare il livello di verbosità, ovvero poter leggere maggiori dettagli sull'esecuzione dei test, basta inserire il flag **-v** (o l'equivalente --**verbose**) nel comando:

```
python -m unittest -v tests/run.py
```

5.5 Deployment

Allo stato dell'arte tecnologica, la virtualizzazione dei sistemi si può ottenere simulando delle risorse hardware (e quindi astraendo lo strato software dall'hardware reale) oppure suddividendo in maniera controllata le risorse reali.

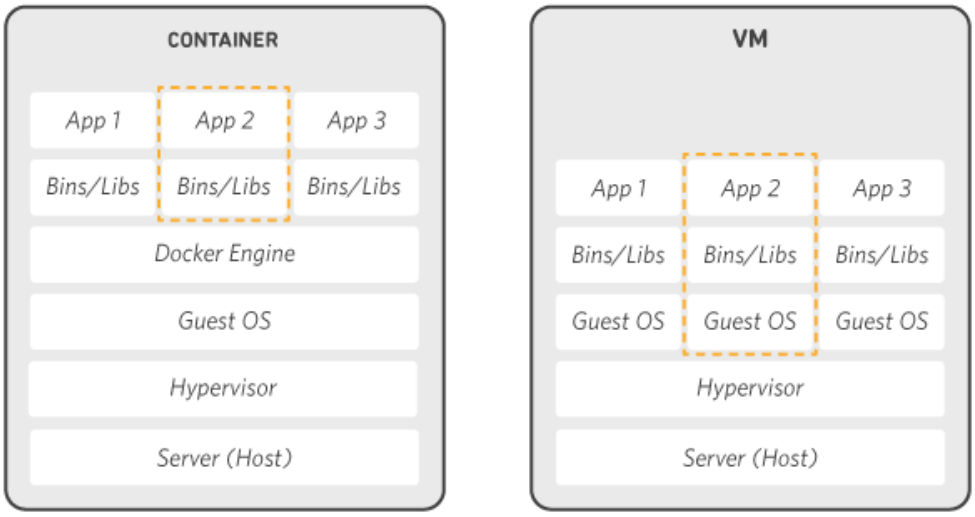

Figura 9 – Differenze container e virtual machine

Al fine di facilitare il deployment dell'applicazione in contesti eterogenei, si è scelto di utilizzare come strategia di deployment i container basati sul prodotto open-source Docker, che ricadono nella seconda modalità di virtualizzazione appena descritta.

Nel processo di containerizzazione, le risorse fisiche del sistema vengono suddivise mediante un meccanismo di sistema chiamato CGROUP (control group). Il demone Docker (Docker Engine), mediante i CGROUP, può impostare dei limiti per ogni container in esecuzione o lasciare l'utilizzo delle risorse in modalità concorrente. **Docker** è un progetto open-source che ha introdotto modifiche significative all'originario LXC, rendendo i container più portabili, offrendo al tempo stesso una flessibilità "**cloud-like**" a qualsiasi infrastruttura in grado di gestirli. Inoltre, fornisce delle utilities per gestire container **images**, che specificano come il container (o gruppi di container legati fra loro) deve essere fatto e come deve funzionare. Gli strumenti delle Docker images permettono allo sviluppatore di creare librerie di images, comporre images in una sola o lanciare applicazioni basate su di esse in un'infrastruttura locale o remota.

I Docker containers quindi tengono le applicazioni isolate dal sistema sottostante: ciò non solo rende il software stack più pulito, ma semplifica anche il modo in cui viene determinato l'utilizzo di risorse di sistema (CPU, GPU, memoria, I/O, networking, etc.) da parte di

un'applicazione containerizzata.

6 Security Assessment

6.1 Metodologie

Esistono diverse metodologie di security testing, il cui duplice scopo è prevenire comuni vulnerabilità e fornire all'utente una garanzia sulla sicurezza durante l'utilizzo dell'applicativo. Alcune delle tecniche utilizzate sono le seguenti:

- **Parameter tampering** - Tramite la modifica di query strings, parametri di POST requests e hidden fields, prova a guadagnare l'accesso non autorizzato a dati o funzionalità
- **Cookie poisoning** - Modifica i dati salvati nei cookies per testare la risposta dell'applicativo a valori non attesi
- **Session hijacking** - Tenta di acquisire i privilegi di un utente loggato
- **User privilege escalation** - Tenta di ottenere l'accesso non autorizzato ai privilegi dell'admin o degli utenti
- **Credential manipulation** - Modifica l'identificazione o l'autorizzazione delle credenziali al fine di ottenere l'accesso non autorizzato
- **Forceful browsing –** Sfrutta vulnerabilità sull'invio dei file all'utente da parte del web server per ottenere accesso a pagine private
- **Backdoors and debug options** – Identifica backdoors e codice di debugging (spesso eseguito con livello di accesso elevato) da sfruttare per ottenere un livello di accesso maggiore
- **Configuration subversion** – Sfrutta configurazioni errate del web server per ottenere accesso alle cartelle (directory browsing)
- **Input validation bypass** – Rimuove validazioni lato client per testare se i controlli sono effettuati anche lato server
- **SQL Injection** – Invia comandi SQL appositamente prodisposti nei campi di input per bypassare controlli
- **Cross-Site Scripting** – Inietta codice nella pagina web nel tentativo di farlo eseguire ad un utente

Tutte queste metodologie notificano l'utente della presenza di eventuali rischi, classificati da "low" ad "high" in base alla severità o criticità.

6.2 Il Framework OpenVAS

OpenVAS (Open Vulnerability Assessment System) è un framework gratuito per l'analisi e la gestione delle vulnerabilità che permette di effettuare la scansione di un sistema target (IP/HOSTNAME) basandosi su un range di porte ed una serie di policy. L'architettura è la seguente (divisa in tre livelli: **services**, **data**, **clients**):

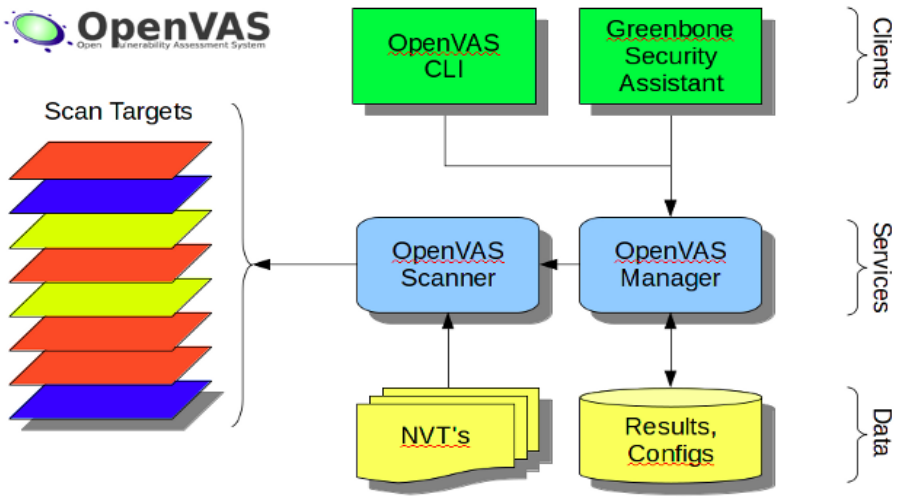

Figura 10 – Architettura framework OpenVAS

Il core è costituito dal livello services, diviso in:

- **scanner** – si occupa di eseguire i test di vulnerabilità verso i "target" (host su internet)
- **manager** – si occupa di interloquire con i client di OpenVas (fornito da GreenBone Networks), memorizzare i dati e gestire gli utenti del sistema

Il livello clients permette l'interazione con il manager e mette a disposizione:

- **Greenbone Security Assistant –** interfaccia grafica per la configurazione del sistema
- **CLI** – interfaccia a riga di comando (Command Line Interface)

Il livello data, invece, consiste in un database SQLite che contiene:

- **NVT** – la lista dei test effettuabili dallo scanner
- **configs, results** – configurazione del sistema, i dati sui risultati delle scansioni e i dati degli utenti del sistema

OpenVAS può essere installato, su una macchina Linux, tramite i seguenti comandi:

```
sudo apt-get update

sudo apt-get dist-upgrade

sudo apt-get install openvas

sudo openvas-setup

sudo openvas-start
```

I primi due comandi hanno lo scopo di aggiornare la distribuzione e, di conseguenza, il database degli NVT; il terzo comando installa il framework OpenVAS; il quarto comando effettua il setup; infine l'ultimo avvia il servizio, rendendolo disponibile all'indirizzo **https://127.0.0.1:9392**.

Come da procedura ANAC, il framework OpenVAS viene utilizzato per testare la sicurezza dell'applicazione, le cui features di sicurezza sono esposte nel prossimo paragrafo.

6.3 Sicurezza della piattaforma

Le applicazioni web, come la piattaforma descritta nell'elaborato, sono da tempo fra gli obiettivi preferiti dagli hacker, poiché forniscono accesso a informazioni sensibili e al tempo stesso sono relativamente facili da violare.

La piattaforma OpenWhistleblowing 2.0.0, oltre alla già citata pratica di cifratura del database tramite SQLCipher e la cifratura simmetrica delle segnalazioni su filesystem, prevede le seguenti misure di sicurezza grazie al framework Flask e all'implementazione di due librerie: **Flask-Talisman** e **Flask-SeaSurf**.

Flask prevede di default le seguenti soluzioni:

- **Cross-Site Scripting (XSS) security** – configura automaticamente Jinja2 per effettuare l'escape di tutti i valori (salvo altra indicazione) affinchè non contengano tags HTML. Per quanto riguarda l'XSS per iniezione di attributo, la soluzione è inserire tutte le espressioni Jinja fra apici singoli o doppi, rimuovendo la possibilità di inserire un handler JavaScript malevolo. L'unico tag che Jinja2 non protegge è **<a>** e l'attributo href, che può contenere codice JavaScript: per ovviare a ciò deve essere settato l'header **Content Security Policy** (**CSP**) a 'default-src: "self"';
- **JSON security** – nelle versioni precedenti di Flask, la funzione **jsonify()** non serializzava correttamente gli array in JSON per via di una vulnerabilità di sicurezza presente in ECMAScript 4. La versione successiva (ECMAScript 5) ha risolto questa vulnerabilità, quindi solo i browser datati sono vulnerabili, e la funzione jsonify() è stata modificata per supportare la corretta serializzazione degli array.

La libreria Flask-Talisman è una piccola estensione per Flask (creata dagli sviluppatori del Google Cloud Platform) che gestisce l'impostazione degli headers HTTP, risolvendo i problemi più comuni delle web applications. La configurazione di default è la seguente:

- Forza tutte le connessioni in https, eccetto se il server è in modalità debug;
- Abilita l'header **Strict-Transport-Security** (spesso abbreviato in HSTS) che comunica al browser che l'unico accesso possibile alla piattaforma è tramite https;
- Setta il cookie di sessione di Flask a **secure**, così che non può mai essere settato se l'applicazione viene acceduta (in qualche modo) in maniera non sicura;
- Setta il cookie di sessione di Flask a **httponly**, prevenendo l'accesso via JavaScript al suo contenuto;
- Setta l'header **X-Frame-Options** a **SAMEORIGIN** per evitare il clickjacking;
- Setta l'header **X-XSS-Protection** per abilitare un filtro anti-XSS per IE/Chrome;
- Setta l'header **X-Content-Type-Options** per prevenire un eventuale content type sniffing per Internet Explorer >= 9;
- Setta l'header **X-Download-Options** per prevenire l'apertura di file scaricati per Internet Explorer >= 8;
- Setta l'header **Content-Security-Policy** a **default-src: 'self'** per prevenire il precedentemente citato XSS sull'attributo href;
- Setta l'header **Referrer-Policy** a **strict-origin-when-cross-origin** che regola quali informazioni sui referenti devono essere incluse in ogni richiesta.

E' una best-practice affiancare a questa libreria l'estensione Flask-SeaSurf, che ha lo scopo di prevenire il Cross-Site Request Forgery (CSRF): questo tipo di vulnerabilità è molto diffuso ed è stato persino rinvenuto in grandi siti come YouTube; inoltre, è un attacco problematico poiché il suo meccanismo è relativamente facile da usare.

Questa libreria setta tutti gli attributi dell'applicazione relativi ai cookie con valori di sicurezza ed aggiunge a Jinja una funzione globale chiamata **csrf_token()**, che recupera il token corrente e tenta di "matcharlo" con il token della richiesta. In ogni template in cui sono previsti i metodi http POST, PUT e DELETE deve essere inserito il seguente codice:

```
<input type="hidden" name="_csrf_token" value="{{csrf_token()}}">
```

Che chiama la funzione globale precedentemente citata.

7 Conclusione e sviluppi futuri

Forte dell'alto valore tecnologico e coerente con le normative vigenti a livello italiano ed europeo, la piattaforma OpenWhistleblowing 2.0.0 si configura come un prodotto modulare e scalabile, con un'architettura solida ma al tempo stesso flessibile: queste caratteristiche consentono all'applicazione di essere distribuita ad un vasto numero di clienti con diverse necessità e requisiti, in modo da essere appetibile per tutte le Pubbliche Amministrazioni Centrali e Locali (e per tutti gli enti, privati e non, che collaborano con esse), per le banche e gli istituti finanziari, per le Autorità di vigilanza sul riciclaggio e sul terrorismo e, "last but not least", per l'Autorità Nazionale Anticorruzione.

Tramite un approccio Agile durante lo sviluppo, riassumibile in "early delivery/frequent delivery", e tramite l'utilizzo di piattaforme all'avanguardia come Gitlab, Kubernetes e Docker i processi di versioning, bugfixing, aggiornamento e deployment della piattaforma sono resi più efficienti e veloci; la forte struttura modulare, a differenza di quella monolitica del vecchio applicativo, fornisce la possibilità di integrare nuove tecnologie innovative a supporto dell'applicazione e/o di rimuovere quelle obsolete e non più richieste dal vasto parco clienti a cui la piattaforma fa riferimento.

Infine, grazie al risultato della fase di testing, il cliente ha la possibilità di esplorare e verificare tutti i test cases della piattaforma, potendone quindi verificare l'effettivo funzionamento; grazie ai risultati della fase di security assessment, invece, può assicurarsi che l'applicazione possegga quegli standard di sicurezza richiesti dalle normative vigenti per un applicativo che deve svolgere le mansioni descritte dal processo di whistleblowing.

Grazie all'esperienza lavorativa svolta presso la Laser Romae s.r.l., in particolare grazie all'attività di reingegnerizzazione descritta nell'elaborato, sono riuscito ad integrare le conoscenze acquisite durante il mio percorso di studi triennale con quelle necessarie per il mondo del lavoro.

Ho imparato ad analizzare ed interpretare i requisiti del cliente, progettando (ovvero individuando l'ambiente di sviluppo e definendo le metodologie di riferimento) e sviluppando l'applicativo definendone le procedure per la gestione e per la manutenzione, evidenziando anche eventuali rischi e proponendo soluzioni migliorative.

Ho ampliato le mie conoscenze riguardanti i linguaggi di programmazione e i paradigmi di programmazione sicura: ho appreso nuovi meccanismi per la cifratura delle informazioni ed ho approfondito quelli a me già noti, come le tecniche di cifratura **simmetrica** e **asimmetrica**; ho imparato a sviluppare RESTful API coerenti utilizzando i metodi HTTP; ho imparato ad utilizzare Gitlab, con relativo sistema di versioning, per aumentare il livello di efficienza durante il lavoro in team; ho appreso come creare un ambiente in sviluppo in cloud con annessa CI/CD Pipeline; ho imparato il funzionamento del sistema di containerizzazione e l'ho applicato tramite Docker containers; ho esplorato le tecniche dell'unittesting e le metodologie di security assessment.

L’azienda mi ha dato l’opportunità di lavorare per grandi clienti come l’ANAC e la Banca d’Italia: grazie al lavoro che ho effettuato con e per quest’ultima sono riuscito a rilevare la maggior parte delle criticità riscontrate nella vecchia piattaforma OpenWhistleblowing 1.0.1 (ed 1.0.0) e sulla base delle quali ho effettuato il lavoro di reingegnerizzazione. Per il mio lavoro effettuato ho ricevuto da G.S. e A.P. (rispettivamente un direttore ed un dipendente della Banca d’Italia con cui ho avuto il piacere di collaborare) le seguenti parole di encomio che mi riempiono di orgoglio:

> *“… da Paolo un eccellente contributo professionale, una grande disponibilità e, che non guasta mai, uno splendido lato umano”*

La progettazione e realizzazione di un prototipo per l’applicazione OpenWhistleblowing 2.0.0 è stato un lavoro complesso che di certo non si fermerà a quanto già fatto e descritto in questa relazione: il codice verrà revisionato, corretto e migliorato, anche su indicazione della stessa Autorità Nazionale Anticorruzione, per poter rilasciare per il riuso la miglior piattaforma possibile per la segnalazione degli illeciti.

8 Bibliografia

- L. 190/2012 (CD. LEGGE ANTICORRUZIONE): IL PRIMO APPROCCIO DEL LEGISLATORE ITALIANO AI WHISTLEBLOWING SCHEMES – Matteo Bascelli, CBA Studio Legale e Tributario, Milano

9 Sitografia

- https://www.html.it/
- https://github.com/globaleaks
- https://www.zetetic.net/sqlcipher/
- http://www.dirittobancario.it/

10 Allegati

10.1 Versioni software Openwhistleblowing 1.0.1

Lo sviluppo è stato effettuato su sistema operativo CentOS 7, mentre il rilascio su infrastruttura ANAC è stato effettuato su RedHat Linux 7.

10.1.1 Frontend

Linguaggio utilizzato: Javascript
Principali Framework: AngularJS + Bootstrap

Versioni con piattaforma di riferimento CentOS 7

Nome pacchetto	Versione
angular	1.6.10
angular-aria	1.5.5
angular-dynamic-locale	0.1.32
angular-filter	0.5.8
angular-i18n	1.5.5
angular-resource	1.5.5
angular-route	1.5.5
angular-translate	2.11.0
angular-translate-loader-url	2.11.0
angular-translate-loader-static-files	2.11.0
angular-ui-bootstrap-bower	1.3.2

angular-file-saver	1.1.1
angular-zxcvbn	3.2.0
bootstrap-inline-rtl	3.3.6
d3	3.5.17
flow.js	2.10.1
ng-flow	2.7.1
openpgpjs	2.3.0
scrypt-async	1.2.0
stacktrace-js	1.1.2
zxcvbn	4.3.0

10.1.2 Backend

Linguaggio utilizzato: Python
Principali Framework: Twisted + Cyclone

Versioni con piattaforma di riferimento CentOS 7

Nome pacchetto	Versione
cffi	0.8.2
cryptography	1.1
Cyclone	1.1

enum34	0.9.23
pdfrw	0.4
pyOpenSSL	0.14
pyasn1	0.1.7
psycopg2	2.4
pycparser	2.1.0
python_gnupg	0.3.6
python_docx	0.8.7
scrypt	0.6.1
six	1.5.2
storm	0.19
transaction	1.1.1
txsocksx	1.13.0.0
zope.component	4.0.2
zope.event	4.0.1
zope.interface	4.0.5
Parsley	1.2

Pillow	5.3.0
Twisted	13.2.0

10.1.3 Tor hidden service

Linguaggio utilizzato: Python
Principali Framework: Twisted + pyOpenSSL

Versioni con piattaforma di riferimento CentOS 7

Nome pacchetto	Versione
python-cffi	0.8.2
python-cryptography	0.8.2
python-enum34	0.9.23
python-openssl	0.14
python-parsley	1.2
python-pyasn1	0.1.7
python-pycparser	2.10
python-six	1.5.2
python-transaction	1.1.1
python-twisted-core	13.2.0
python-zope.interface	4.0.5

10.2 Versioni software Openwhistleblowing 2.0.0

Tutte le versioni software qua elencate fanno riferimento al prototipo in via di rilascio per l'ANAC e sono soggette a variazioni durante la fase di sviluppo.

10.2.1 Backend

Linguaggio utilizzato: Python
Principali Framework: Flask

Versioni con piattaforma di riferimento CentOS 7

Nome pacchetto	Versione
Flask	1.0.2
Flask-Cors	3.0.7
Flask-Hashing	1.1
Flask-SQLAlchemy	2.3.2
Flask-WTF	0.14.2
Pillow	5.4.1
celery	4.2.1
configparser	3.7.1
cryptography	2.5
fFlower	0.9.2
gunicorn	19.9.0
pdfrw	0.4

pyAesCrypt	0.4.2
pycrypto	2.6.1
python-docx	0.8.10
pysqlcipher	1.0.3
redis	3.1.0
requests	2.21.0

10.2.2 Frontend

Linguaggio utilizzato: Python
Principali Framework: Flask + Bootstrap 4.3.1

Versioni con piattaforma di riferimento CentOS 7

Nome pacchetto	Versione
Flask	1.0.2
Flask-Cors	3.0.7
Flask-Dropzone	1.5.3
Flask-SeaSurf	0.2.2
Flask-Talisman	0.6.0
Flask-WTF	0.14.2
configparser	3.7.1

gunicorn	19.9.0
requests	2.21.0