Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Identità digitale e servizi web decentralizzati: interoperabilità e single sign-on

Tesi di laurea triennale

Relatore

Prof. Davide Bresolin

Correlatore

Dr. Mattia Zago

Laureanda

Martina Garon

Matricola

1170566

Anno Accademico 2022-2023

Per loro nulla può cambiare, perché loro stessi non possono cambiarsi.

— Ende, Michael. La storia infinita. Stoccarda, Longanesi & C., 1981

Dedicato a chi c'è ed a chi c'è stato.

Abstract

Questo documento descrive il lavoro che ho svolto durante il periodo di *internship*, presso l'azienda Athesys srl. L'*internship* è durato 300 ore circa, dal 13 Marzo 2023 al 26 Maggio 2023 e sono stati approfonditi temi riguardanti l'innovazione e la ricerca applicata in ambito di identità decentralizzata e di *Self-Sovereign Identity* (SSI).

Attualmente, la *privacy* delle informazioni relative le identità digitali non è completamente tutelata: spesso, è necessaria una figura intermedia che li memorizza e li gestisce. In questo modo, l'utilizzo delle informazioni da parte dell'utente proprietario dell'identità è sempre condizionato dalla figura intermedia.

Nasce quindi la necessità di una nuova gestione delle informazioni personali, che restituisca all'utente il controllo della propria identità digitale. *Self-Sovereign Identity* (SSI) è un recente modello ideato per rimuovere la figura intermedia e garantire all'utente la gestione completa dei propri dati.

La soluzione si colloca in ambito *Identity and Access Management* (IAM) e può essere abbinata alle diverse tipologie di *login*, quale *Single Sign-On* (SSO). La tecnologia è sostenuta dalla fondazione Hyperledger ed utilizza le *blockchain*.

Aries JS è un *framework* che implementa il nuovo modello ed ho creato un prototipo, affinché Athesys srl possa valutarne l'integrazione con il suo applicativo Monokee SSO.

Gli altri obbiettivi da raggiungere durante il tirocinio consistevano in una formazione adeguata in merito all'identità decentralizzata, un approfondimento delle tecnologie (ad esempio: Indicio TestNet, TypeScript, Node.JS ed Express JS), l'analisi, la progettazione e lo sviluppo di un PoC dimostrativo, in forma di API, per la generazione di credenziali.

"È meglio essere povero che incolto: il povero, infatti, manca solo di denaro, l'incolto di umanità"

— Citazione di Aristippo.
Laerzio, Diogene. Vite dei filosofi a cura di Gigante Marcello. Editori Laterza, 1983

Ringraziamenti

Vorrei esprimere la mia gratitudine al Prof. Davide Bresolin, relatore della mia tesi, per aver accettato di accompagnarmi in questo percorso. Ricordo i professori del corso di Laurea: sono riusciti a trasmettermi la passione per un'Informatica "umana".
Un pensiero particolare va a Dr. Mattia Zago, Dr. Matteo Midena e Dr. Roberto Griggio di Athesys, per avermi introdotto in un settore che non conoscevo!
Desidero ringraziare con affetto i miei genitori Donatella e Giancarlo, che assieme a mia sorella Diana hanno ascoltato i racconti delle difficoltà incontrate.
Ringrazio Giacomo e la sua famiglia, per avermi sostenuta sin da subito e per avermi accolta!
Ricordo con affetto nonna Carla, per le innumerevoli preghiere per chiedere "aiuto dall'Alto". La mia gratitudine è per le zie e gli zii, per il grande coraggio che mi hanno sempre infuso e la grinta con cui mi hanno sostenuto. Non posso dimenticare i miei cugini: con loro, sono tornata in aula! Ringrazio i miei colleghi, che da anni seguono l'andamento accademico con la stessa passione del campionato. Ricordo tutti gli amici, che attendono la festa di laurea da troppo tempo!
Infine, ricordo anche chi non incoraggiava il mio percorso accademico: mi ha fornito un motivo in più per terminarlo.
Grazie a Voi, la scommessa "con me stessa" è stata vinta.

Padova, Luglio 2023 Martina Garon

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Si descrivono il contesto delle attività di tirocinio, la soluzione sviluppata e la struttura del testo.

1.1 Contesto

L'identità è un fatto abilitante nella società ed il riconoscimento di una persona è requisito fondamentale per accedere ai servizi. In ambito digitale, persone fisiche e giuridiche hanno numerosi strumenti per identificarsi con differenti livelli di sicurezza. Le soluzioni federate sono comunemente utilizzate per l'identificazione e dispongono di un ventaglio di protocolli ciascuno con le proprie peculiarità. La gestione federata delle identità prevede che un utente registri i propri dati personali presso un ente, il quale può abilitare l'utente stesso ad altri servizi, ad esempio *ecommerce* e *blog*. L'ente federato, però, risulta essere un accentratore di informazioni personali collezionando dati sensibili e strettamente legati all'utente. L'ente accentratore funge anche da controllore in quanto abilita o meno l'utilizzo dell'identità presso servizi terzi.

Si cercano quindi nuovi modelli per restituire all'utente il controllo dei suoi dati. Da qualche anno, l'identità decentralizzata riscuote notevole interesse e potrebbe migliorare le attuali modalità di autenticazione, aumentandone *privacy* e sicurezza. L'identità decentralizzata prevede che i dati degli utenti siano presso i loro dispositivi e che

l'utente li condivida volontariamente verso servizi terzi.

Le due modalità differiscono principalmente nello stoccaggio dei dati: i dati sono presso un ente centrale nella modalità federata, mentre rimangono presso l'utente con l'identità decentralizzata. Inoltre, l'identità decentralizzata utilizza strutture progettate per soddisfare requisiti elevati di *privacy* e sicurezza, che non sono garantiti nella modalità federata: infatti, la presenza dei dati utente presso l'ente federato ne abilita potenzialmente l'uso non autorizzato.

Si studia quindi il nuovo modello di identità decentralizzata *Self-Sovereign Identity* (SSI), che sfrutta l'immutabilità e la sicurezza della *blockchain* per garantire la disponibilità e la trasparenza delle informazioni.

1.2 Gestione delle identità

Le criticità relative alle identità digitali sono strettamente connesse alle loro modalità di utilizzo. *Identity and Access Management* (IAM) è la disciplina che si occupa della gestione delle identità, spaziando tra le modalità di *login*, la cessione dei privilegi e la gestione complessiva degli *account*.

Una delle principali modalità di *login* è il *Single Sign-On* (SSO), che abilità l'accesso a servizi diversi con un'unica credenziale. I protocolli maggiormente diffusi sono *Security Assertion Markup Language* (SAML), *Open Authorization* (OAuth) ed *OpenID Connect* (OIDC). La cessione dei privilegi ad un *account* è fondamentale per garantire un livello di sicurezza elevato: allo scopo, è fortemente consigliato un approccio *Zero Trust* (in cui si diffida degli utenti non riconosciuti), applicato al paradigma *Just-in-Time* (JIT), che applica i privilegi solo per il tempo necessario.

Comunemente, la gestione complessiva dell'*account* avviene tramite identità centralizzate. Tra l'utente ed il servizio, viene interposto un "ente federato", con il compito di gestire i dati di identità ed autorizzare l'uso dell'identità digitale presso il servizio. L'utente condivide la gestione del proprio *account* con l'ente federato.

Si è studiata una modalità alternativa, dove il controllo dell'identità è detenuto solamente dall'utente stesso: è l'"identità decentralizzata". Il modello principale è la SSI, fondata sulla *Distributed Ledger Technology* (DLT). Quest'ultima è una tecnologia

che distribuisce i dati in più luoghi, mantenendoli allineamenti tra di loro, allo scopo di rendere i dati fruibili agilmente e di ridurre i punti di rottura. SSI utilizza una tipologia di DLT: la *blockchain*. Essa garantisce una struttura ideale per conservare dati sensibili, quali i dati utente, con proprietà di immutabilità, crittografia e concatenazione delle informazioni.

SSI prevede tre attori: *Issuer*, *Holder* e *Verifier*. *Issuer* è colui che emette un documento che testimonia un fatto riguardante *Holder*. *Holder* è l'intestatario del documento e l'utente "utilizzatore" della propria identità: gestisce i suoi dati direttamente dai suoi dispositivi, grazie ad un *wallet* digitale. *Verifier* è colui che richiede il documento per fornire un servizio. I tre attori instaurano una relazione di fiducia, chiamata "triangolo della fiducia", in cui riescono a verificare i dati forniti in autonomia.

1.3 Scopo del tirocinio

Athesys srl si occupa di ricerca in ambito di identità digitale e di IAM. Ha sviluppato l'applicativo Monokee SSO, che permette il SSO su diverse piattaforme. Ha composto un *team* di ricerca dedicato all'identità decentralizzata ed è molto attenta ai nuovi *software* presenti nel mercato.

Hyperledger è una fondazione dedicata alla diffusione delle *blockchain* ed ha creato progetti specifici per l'identità decentralizzata, tra cui Hyperledger Aries.

Athesys srl sta valutando l'integrazione nell'applicativo Monokee SSO del *framework* Aries JS, che è una delle implementazioni disponibili di Hyperledger Aries.

L'obiettivo del tirocinio è uno studio di fattibilità dell'integrazione di Aries JS in Monokee SSO. Allo scopo, sono state approfondite le tecnologie citate ed è stato creato un *Proof of Concept* (PoC) dimostrativo delle potenzialità di Aries JS, in forma di API.

1.4 Soluzione sviluppata

Durante il tirocinio, ho approfondito le tematiche in cui opera Monokee SSO. In particolare, ho svolto ricerche sui temi collegati all'identità decentralizzata, ad esempio

SSI ed Hyperledger. Successivamente, ho studiato come avviene la gestione delle identità (detta anche IAM), la modalità di *login* SSO ed alcuni protocolli specifici, come SAML, OAuth ed OIDC. Tutto il materiale raccolto è stato integrato alla documentazione aziendale.

Successivamente, ho studiato il Aries JS, per poi progettare e sviluppare un prototipo che ne valutasse le potenzialità. Il PoC è un *software* suddiviso in tre API, ciascuno che prevede funzionalità dedicate ad un utente specifico.

Il PoC è stato necessario ad Athesys srl per valutare se integrare Aries JS nel suo applicativo. Le funzionalità esposte nella documentazione del *framework* sono state testate dal prototipo e ritenute utili dal *team* di sviluppo. Purtroppo, la documentazione è scarna e non presenta molti casi d'uso, ma quelli previsti sono ben illustrati, rendendo agevole l'implementazione di quanto descritto. Si conclude che Aries JS abbia un ottimo potenziale, ma richiede studi più approfonditi.

1.5 Struttura del testo

Il testo illustra le conoscenze apprese durante lo studio e lo sviluppo del PoC. Il testo è stato strutturato in capitoli.

Nel secondo capitolo si presenta il progetto, con gli obiettivi da raggiungere e le modalità di svolgimento.

Il terzo capitolo illustra quanto appreso durante le attività di formazione. L'approfondimento è stato necessario per la comprensione del contesto ed il raggiungimento degli obiettivi.

Il quarto capitolo illustra i requisiti necessari per lo sviluppo dell'API.

Il quinto capitolo descrive la progettazione dell'API sviluppata.

Nel sesto capitolo si discutono le problematiche riscontrate e le opportunità delle tecnologie incontrate.

Nel settimo capitolo si traggono le conclusioni e si espongono i risultati raggiunti.

Relativamente al documento, sono state adottate le seguenti convenzioni tipografiche:

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- I termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*;
- Le fonti sono raccolte nella bibliografia.

Capitolo 2

Descrizione del tirocinio

Athesys srl è l'azienda ospitante del tirocinio. Durante l'esperienza, è stata fatta una formazione adeguata e la creazione di un PoC dimostrativo.

2.1 L'azienda

Athesys srl è un *system integrator* impegnato principalmente nello studio e sviluppo soluzioni di *Identity and Access Management* (IAM). La sede è situata a Padova, in Via Giacinto Andrea Longhin, 79.

L'azienda è stata fondata nel 2010 dalla sinergia di affermati professionisti del settore IT ed offre consulenze specializzate in ambito *System Integration*, *Database Management*, Sicurezza Applicativa, *Governance Cloud Platform*, *Hyperconvergenza* e Sviluppo *Software* con metodologia Agile. Le tecnologie a disposizione dell'azienda sono numerose e le consentono di sviluppare per *mobile*, *web*, *desktop*, con relativa archiviazione dati.

Figura 2.1: Il logo di Athesys srl. Immagine scaricata dal sito ufficiale di Athesys srl nel luglio 2023.

2.2 L'idea del progetto

Monokee SSO è un *software* sviluppato dalla *startup* Monokee srl, in collaborazione con Athesys srl. Monokee SSO è una soluzione IAM per le identità digitali centralizzate e decentralizzate, che permette il *login* tramite *Single Sign-On* (SSO).

L'idea del tirocinio nasce dall'esigenza di integrare il *software* Monokee SSO con le credenziali emesse con Aries JS. Queste credenziali sono di tipo decentralizzato e la tecnologia è affine ad SSI di *Hyperledger Indy*. Attualmente, l'applicativo non contempla questa funzionalità e l'azienda necessitava di un *test* di fattibilità.

Figura 2.2: Il logo di Monokee srl. Immagine scaricata dal sito ufficiale di Monokee srl nel luglio 2023.

2.3 Requisiti ed obiettivi

Un requisito necessario per completare il prodotto era un'adeguata conoscenza dei sistemi decentralizzati e di SSI. A tal proposito, è stato previsto in periodo di formazione.

Lo scopo finale del progetto era di coiadiuvare il *team* di sviluppo ad integrare un sistema di *login* decentralizzato. Gli obbiettivi da raggiungere erano molteplici. In primo luogo, era richiesta una formazione individuale sui temi di identità digitale, identità decentralizzata, *Identity and Access Management* (IAM), i principali protocolli federati e *Self-Sovereign Identity* (SSI). Successivamente, è stato approfondito il *framework* Aries JS, utilizzato per l'implementazione dello scambio delle credenziali. Terzo ed ultimo obbiettivo era l'integrazione di Monokee SSO all'utilizzo di credenziali di tipo SSI.

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- «ob» per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- «de» per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- «op» per i requisiti opzionali, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito. Nel dettaglio, si è previsto il raggiungimento dei seguenti obiettivi:

- Obbligatori:
  - **ob01**: implementazione attraverso la piattaforma aziendale Monokee SSO dello scambio di credenziali verificabili utilizzando un *agent* Hyperledger Aries;
  - **ob02**: integrazione attraverso la piattaforma aziendale Monokee SSO di due servizi esterni utilizzando Monokee SSO come *Identity Provider* e sia SAML che OIDC come protocolli SSO.

- Desiderabili:
  - **de01**: integrazione PoC accesso SP via SAML / OIDC utilizzando identità effimere ottenute da una specifica VC;
  - **de02**: integrazione PoC accesso SP via SAML / OIDC utilizzando identità effimere ottenute da una VP generica contenente molteplici VC;
  - **de03**: *mapping* ruolo utente SSI su *claim* SSO via anoncreds.
- Opzionali:
  - **op01**: sviluppo documentazione architetturale, flussi dati e flussi di controllo per gli strumenti utilizzati.

2.4 Pianificazione

Il tirocinio è stato suddiviso in due macro periodi: il primo è stato dedicato alla formazione, mentre il secondo allo sviluppo dei prodotti documentali e *software*. Durante il periodo di formazione sono stati approfonditi dei temi specifici:

- Formazione sulle tecnologie per l'identità digitale SSI (20 ore), durante il quale si è studiato il principale paradigma tecnologico per la SSI (Hyperledger Indy);
- Formazione sulle tecnologie per l'identità digitale federata (20 ore), durante il quale si sono studiati i principali protocolli tecnologici per l'integrazione di sistemi SSO: SAML ed OIDC.

Nel secondo periodo sono stati sviluppati i prodotti richiesti:

- *Deliverable 1*, composto da:
  - presentazione di Hyperledger e dei protocolli SAML e OIDC per le identità federate. Il *deliverable 1* doveva essere completato sotto forma di articoli nella *Knowledge Base* aziendale a completamento della documentazione esistente;
  - Studio di fattibilità ed analisi dei requisiti (40 ore): durante questa fase si è studiato il *framework* Hyperledger Aries ed il relativo utilizzo di Hyperledger

Indy come libreria per interagire con la *Distributed Ledger Technology* (DLT). (Ob. 1, 2).

- *Deliverable 2*, composto da:
  - Pianificazione requisiti e funzionalità per integrazione di credenziali verificabili con identità effimere SAML o OIDC (sia da SSI a SSO che viceversa);
  - tre cicli di sviluppo con metodologia Agile (2 settimane), con una struttura generale di iterazione così definita:
    * Studio e progettazione funzionalità: identificazione attributi e componenti minime richieste dai protocolli (sia federati che decentralizzati) e proposta di soluzione per ottenere questi dalla controparte tecnologica;
    * *Meeting* interno e condivisione stato avanzamento linee lavoro: presentazione dei risultati e delle opportunità associate con la soluzione candidata in esame;
    * Implementazione: integrazione API *agent* SSI e SSO;
    * Documentazione: realizzazione *deliverable* e contribuzione alla documentazione aziendale rispetto ai temi sviluppati nella fase 1 e durante il ciclo attivo.
- *Deliverable 3*: progettazione architetturale e implementativa delle integrazioni realizzate.

Il totale del lavoro pianificato ammonta a 304 ore.

2.5 Modalità di svolgimento

L'attività di *stage* è stata svolta in regime di *smart working*, con differenti orari lavorativi, come mostrato nella figura 2.3:

- dal 13/03/2023 al 14/04/23 dalle ore 14:00 alle ore 18:00;
- dal 17/04/2023 al 25/07/2023 dalle ore 09:00 alle ore 18:00, con un'ora di pausa pranzo.

Il lavoro è stato svolto sul *computer* personale, senza vincoli da parte di Athesys srl sul luogo di lavoro. Athesys srl ha messo a disposizione spazi e risorse presso il suo HQ, qualora si fosse preferito lavorare fisicamente in presenza. Una riunione settimanale è stata pianificata con il *tutor* aziendale, nonché Correlatore, ed il *management* per allineamento sullo stato avanzamento lavori (indicativamente il lunedì pomeriggio).

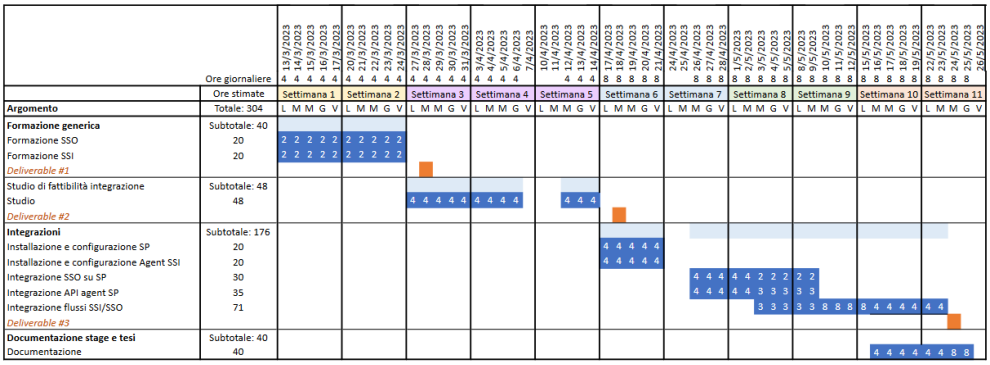

Figura 2.3: Il calendario del tirocinio. Immagine proveniente dal Piano di Progetto.

2.6 Processi e metodologie

In fase di "avvio" del tirocinio, il *tutor* ha illustrato il *business case* del progetto. Athesys srl sta valutando l'integrazione di Monokee SSO con le credenziali verificabili in SSI. Aries JS ha numerose potenzialità per raggiungere lo scopo e necessita di una dimostrazione di funzionamento. Il principale vantaggio è l'ampliamento della gamma di credenziali utilizzabili in Monokee SSO, che gli permetterebbe di affermarsi come *leader* di mercato.

Il principale risultato che ci si aspetta dal progetto è la dimostrazione della facilità di utilizzo e l'integrazione con i componenti offerti da Hyperledger. Il successo del progetto è testimoniato da un PoC funzionante, corredato da documentazione adeguata.

La pianificazione del progetto è indicata nel Piano di Progetto ed ha suddiviso le attività in due periodi. Il primo periodo è stato dedicato all'approfondimento dell'identità decentralizzata e delle tecnologie collegate.

Le attività sono state suddivise in macroaree:

- Ricerca delle informazioni in *internet* e con colloqui con il *team* di Athesys srl;
- Redazione di un testo riassuntivo delle ricerche;
- Integrazione della *Knowledge Base* aziendale di Athesys srl, con le informazioni raccolte.

Il secondo periodo è stato dedicato allo studio di fattibilità del *framework* Aries JS. Le attività si possono suddividere in aree:

- Studio del framework Aries JS, con approfondimento dei *tutorial* forniti;
- Definizione dei requisiti del PoC, con redazione del documento "Analisi dei Requisiti";
- Progettazione del PoC, con redazione de documento "Progettazione Architetturale";
- Sviluppo del PoC;
- Presentazione dei risultati al *team* di Athesys srl.

Le *milestone* da raggiungere corrispondono agli obiettivi di periodo.

Athesys srl ha coinvolto parte del suo *team*, affiancando il *tutor* e lo sviluppatore per la configurazione delle tecnologie ed il supporto. È stata adottata una metodologia Agile.

Il monitoraggio delle attività è stato costante. Durante il primo periodo di approfondimento, numerosi sono stati gli incontri di discussione degli argomenti. Durante il secondo periodo, alcune difficoltà tecniche sono state risolte grazie al supporto del *team*.

La chiusura del progetto è avvenuta con la presentazione dei risultati al *tutor* Dr. Mattia Zago, allo sviluppatore Dr. Matteo Midena, al Presidente Dr. Roberto Griggio ed al tutto il *team* di sviluppo. I presenti sono stati soddisfatti e sono propensi ad integrare Aries JS all'applicativo Monokee SSO.

Capitolo 3

Background

3.1 Identità analogiche e digitali

L'identità è verificata quotidianamente. Le modalità di "identità digitale" hanno subito numerose evoluzioni nel tempo, aumentandone sicurezza e privacy*.*

L'identità è il complesso dei dati personali, caratteristici e fondamentali che consentono l'individuazione di un soggetto e ne garantiscono l'autenticità: riguarda qualsiasi caratteristica o fatto che possa identificarlo.

Un soggetto è un'entità con dati identificativi. Esempi di soggetto sono: persone fisiche e giuridiche, servizi e sistemi.

Quotidianamente, le identità sono certificate da documenti. Esempi di documenti sono:

- documenti personali (come carta d'identità o passaporto);
- certificazioni (come attestati scolastici o di corsi specializzati);
- documenti che identificano l'appartenenza ad un gruppo (si citano ad esempio la tessera di una palestra, la *fidelity card* di un negozio o il *badge* lavorativo).

Nell'era digitale, identificare gli individui è un requisito per accedere ai servizi. Si citano due comuni modalità di gestione delle identità:

- **Identità a validità limitata**: ogni *service provider* definisce e gestisce le proprie modalità di registrazione e identificazione. L'utente è dotato di credenziali di accesso al servizio;
- **Identità federata**: un intermediario è incaricato di confermare l'identità di coloro che tentano l'accesso. Questa modalità impatta negativamente sulla *privacy*, poiché l'intermediario riceve informazioni riguardo l'utilizzo del sito terzo, tracciando l'attività dell'utente. In Italia, gli intermediari più comuni sono: Google, Facebook e SPID.

I sistemi di identità tradizionali centralizzano l'emissione, la manutenzione ed il controllo degli identificatori.

Recentemente, è stato ideato un modello alternativo, che sopperisce ai difetti delle altre modalità: la *Self-Sovereign Identity* (SSI).

3.2 *Identity Access Management* (IAM)

IAM amministra le autorizzazioni degli utenti. Un approccio JIT, concede le autorizzazioni quando necessarie, mentre con un'ottica Zero Trust riduce le autorizzazioni permanenti.

Identity and Access Management (IAM) si colloca nel settore della *cyber security* ed è l'insieme delle tecnologie, criteri e procedure che determinano le autorizzazioni di accesso agli applicativi di un'organizzazione. Si applica nella gestione del ciclo di vita *end-to-end* delle identità e dei diritti degli utenti per tutte le risorse di un'organizzazione, sia nei *data center* sia nel *cloud*.

La strategia *Zero Trust* prevede che l'*account* dell'utente sia privo di autorizzazioni. A seguito dell'autenticazione, saranno concesse le autorizzazioni essenziali, per la durata minima necessaria.

IAM permette la "gestione delle identità", suddivisa in: *governance* delle identità, gestione degli accessi e servizi di *directory*. La *governance* delle identità include attività specifiche:

- Gestione di *provisioning* e *deprovisioning* degli utenti;
- *Intelligence* delle identità, che corregge rapidamente le autorizzazioni degli utenti ad alto rischio;
- Esaminare continuamente l'azienda per identificare e correggere le *policy* che influenzano la segregazione dei compiti.

La gestione degli accessi consente la flessibilità di accesso alle infrastrutture, integrando identità e sistemi. Generalmente, è riconoscibile nelle modalità di autenticazione (vedi figura 3.1), ad esempio *Single Sign-On* (SSO) e *Multi-factor authentication* (MFA).

I servizi di *directory* raggruppano le applicazioni ed i servizi accessibili dall'utente. Forniscono flessibilità e ottimizzazione dell'architettura, velocizzano i progetti di gestione delle identità e le implementazioni delle applicazioni stesse.

I vantaggi di IAM sono notevoli:

- **Automazione**: consente una maggiore efficienza operativa, riducendo il tempo e le risorse necessarie per gestire manualmente l'accesso degli utenti e limitando il rischio di errore umano;
- **Sicurezza**: maggiore controllo sull'accesso degli utenti, riducendo il rischio di violazioni dei dati;
- ***Governance***: consente una collaborazione più forte in tutta l'infrastruttura;
- **Agilità**: consente alle organizzazioni di mantenere la sicurezza e la *compliance* anche all'interno di modelli di *business* in evoluzione.

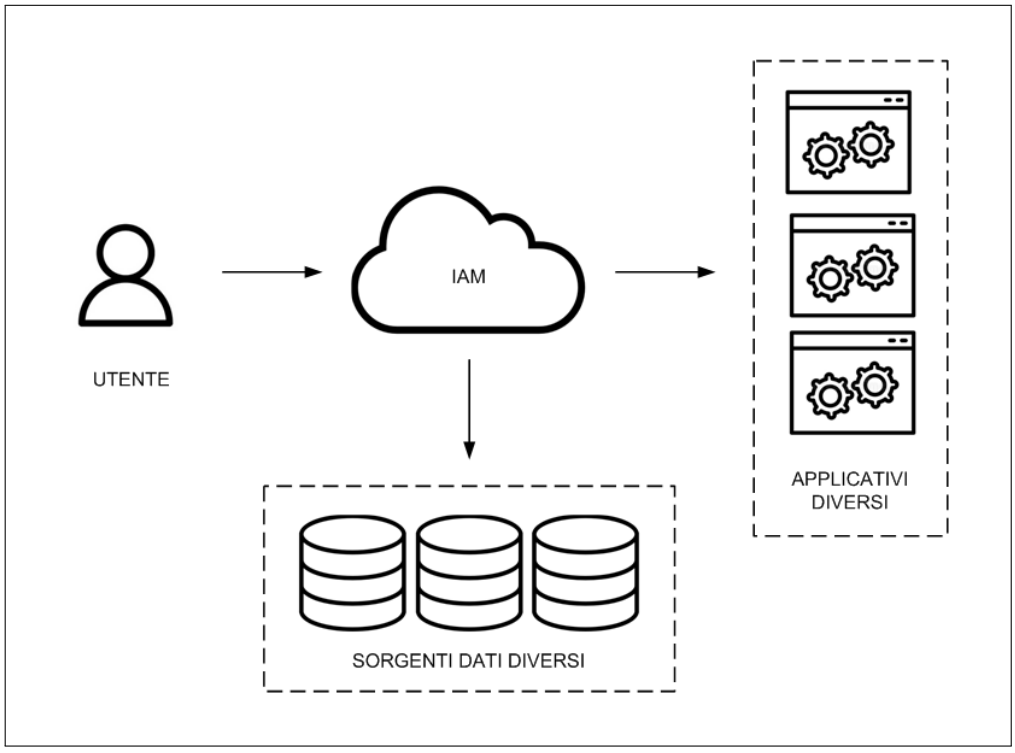

Figura 3.1: L'utente effettua il *login* presso un'infrastruttura. Grazie ad IAM, avrà accesso personalizzato alle risorse.

Il modello *Zero Trust* si avvale del paradigma *Just-in-Time* (JIT), che abilita l'utente ad un accesso privilegiato, in tempo reale. L'accesso JIT garantisce che le attività privilegiate vengano condotte in conformità delle *policy* per la gestione di

identità e accessi (IAM), alla gestione dei servizi IT (ITSM), alla gestione degli accessi privilegiati (PAM) ed alle relative credenziali e flussi di lavoro.

JIT è applicabile grazie a differenti tipologie di accesso:

- **Intermediazione e rimozione degli accessi**: l'accesso è abilitato a fronte di una giustificazione adeguata e sarà abilitato per un periodo di tempo definito;
- **Account effimeri**: l'accesso avviene grazie ad *account* monouso;
- **Elevazione temporanea**: le autorizzazioni "permanenti" sono ridotte e possibilmente innocue. Autorizzazioni più elevate e specifiche verranno concesse a richiesta e per un periodo limitato.

Il paradigma JIT aiuta le organizzazioni a migliorare il proprio approccio alla sicurezza informatica. Inoltre, rende lineari i processi di cessione dei *grant*, contribuendo alla semplificazione del lavoro degli amministratori.

3.3 Protocolli SSO

Single Sign-On (SSO) consente il login a più servizi a fronte di una credenziale, semplificando le attività di IAM.

Il *Single Sign-On* (SSO) è un processo di autenticazione che consente ad un utente di accedere a più applicazioni con un solo *set* di credenziali di accesso, come illustrato in figura 3.2. Questa modalità migliora la *User Experience*, poiché non è necessario effettuare il *login* ad ogni applicativo. L'autenticazione SSO, quindi, facilita l'utilizzo delle risorse di rete senza soluzione di continuità.

Gli attori coinvolti sono due:

- ***Provider* di identità** (IdP): è l'archivio delle identità, che accetta o rifiuta le richieste di autenticazione;
- ***Provider* di servizi** (SP, detto anche "*Service Provider*" o "*Relying Party*"): è l'applicazione che richiede ad IdP di autenticare l'utente.

Fondamentalmente, l'autenticazione con SSO si basa su una relazione di fiducia tra domini. Quando l'utente vuole autenticarsi presso un SP, si verifica se l'utente è autenticato presso IdP. Se la verifica ha esito positivo, SP permette l'accesso; diversamente, viene richiesto il *login* presso l'IdP. I dati di verifica dell'autenticazione assumono la forma di un *token*.

I vantaggi dell'autenticazione SSO sono diversi:

- Riduce il numero di credenziali in possesso dell'utente;
- Semplifica la gestione delle credenziali, da parte degli amministratori delle applicazioni;
- Migliora la protezione dell'identità, introducendo altre modalità di autenticazione, quali l'autenticazione a due fattori (2FA) e l'autenticazione a più fattori (MFA).

SSO deve affrontare anche delle difficoltà:

- La violazione di un *account* SSO crea criticità a tutto il sistema di autenticazione ed alle applicazioni abilitate;

- Un disservizio di SSO impedisce l'utilizzo delle applicazioni;
- La configurazione di SSO può richiedere settaggi specifici per ogni ambiente implementato;
- Alcuni IdP possono essere dei *social network*, che possono essere bloccati da *governance* di rete locali: infatti, la *governance* ha visibilità e potere di organizzazione di un sistema, essendo ad un livello superiore del *management*;
- Alcuni SP possono fornire i dati utente a entità di terze parti.

SSO è una funzionalità delle identità federate ed è considerato una soluzione IAM. Inoltre, permette un approccio *Zero Trust*.

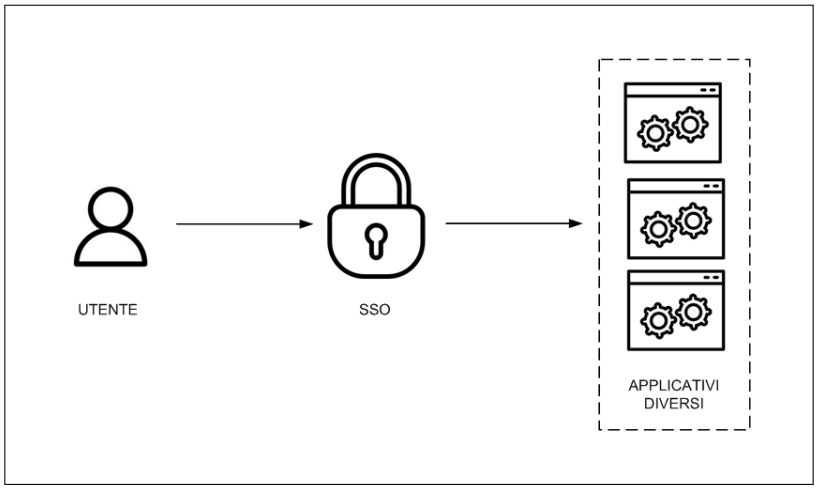

Figura 3.2: Schematizzazione del processo di *login* con SSO.

SSO può essere implementato con diversi i protocolli di autenticazione, tra cui:

- *Security Assertion Markup Language* (SAML);
- *OpenID Connect* (OIDC), creato sulla base di *Open Authorization* (OAuth).

***Security Assertion Markup Language* (SAML)** è uno *standard* di federazione che consente ad IdP di autenticare gli utenti e di trasferire il *token* di autenticazione a SP. Si utilizza il linguaggio XML per codificare le informazioni e può coprire anche vari messaggi e profili di protocollo che fanno parte dello *standard*. Sono due le funzioni di sicurezza principali di SAML:

- **Autenticazione**: confermare che gli utenti sono chi dicono di essere;
- **Autorizzazione**: passaggio dell'autorizzazione utente alle applicazioni per l'accesso a determinati sistemi o contenuti.

Un'"asserzione SAML" è un documento XML che IdP invia ad SP, contenente lo stato di autorizzazione dell'utente. Ne esistono diverse tipologie:

- **Asserzione di autenticazione**: fornise i dati di *login* dell'utente ed il metodo di autenticazione utilizzato;
- **Asserzione di attributo**: contiene informazioni specifiche sull'utente;
- **Asserzione di autorizzazione**: indica se un utente è autorizzato a utilizzare un servizio.

Figura 3.3: Il logo di SAML. Immagine scaricata dal sito internet ufficiale di SAML nel luglio 2023.

***Open Authorization* (OAuth)** è un protocollo *standard* che scambia i dati di autorizzazione tra le applicazioni senza mostrare i dati relativi all'autenticazione utente. Ad esempio, OAuth permette ad un *social media* di suggerire dei possibili collegamenti nella rete *social*, grazie ai contatti delle *email*. I tipi di *grant* più utilizzati sono il *Client Credentials grant* e l'*Authorization Code grant*:

- ***Client Credentials grant***: è utilizzato principalmente per autorizzare un'applicazione, anziché un utente. Solitamente, è applicato a servizi *batch*. All'applicazione autorizzata si assegnano *client ID* e *client secret*;
- ***Authorization Code grant***: è impiegato da applicazioni con interazione dell'utente. L'applicazione vien preventivamente registrata presso l'*Authorization Server*, fornendo almeno un nome e un *redirect URI* (necessario per informare l'applicazione dell'autorizzazione). Durante la registrazione, il *provider* assegna all'applicazione *client ID* e *client secret*, che verranno scambiati durante il processo di autorizzazione.

OAuth è stato progettato per essere estensibile. Un'evoluzione è "PKCE", progettata per rendere più sicuro l'*Authorization Code grant* e limitare le probabilità che l'*Authorization Token* venga intercettato. Il vantaggio di questo tipo di *grant* è che non è necessario l'intervento dell'utente per generare un nuovo *Access Token*.

Figura 3.4: Il logo di OAuth. Immagine scaricata dal sito internet ufficiale di OAuth nel luglio 2023.

Un altro protocollo è ***OpenID Connect* (OIDC)**, che utilizza API REST e *token* di autenticazione JSON per consentire l'accesso in modalità SSO. Consente di verificare l'identità dell'utente in base all'autenticazione eseguita da un IdP, nonché di ottenere informazioni di base sul profilo dell'utente finale in modo interoperabile e simile a REST. La *suite* di specifiche è estensibile, consentendo ai partecipanti di utilizzare funzionalità opzionali come la crittografia dei dati di identità.

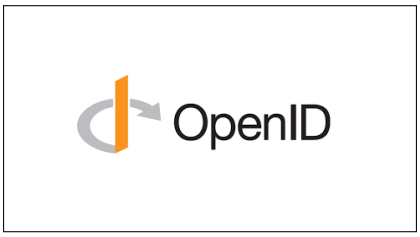

Figura 3.5: Il logo di OIDC. Immagine scaricata dal sito internet ufficiale di OIDC nel luglio 2023.

OIDC è un *layer* sopra al protocollo OAuth. Secondo il Request for Comments number 6749 (RFC6749), OAuth definisce le modalità per l'emissione di un *access token*, grazie ad una *web* API. Nel dettaglio, l'API prevede il campo mandatorio `response_type`, valorizzabile con una delle due opzioni previste: `code` o `token`.

OIDC riprede da OAuth il campo `response_type` e ne espande le funzionalità, prevedendo i valori: `code`, `token` ed `id_token`, le loro combinazioni ed il valore `none`.

Inoltre, OIDC è flessibile verso OAuth. Infatti, le richieste in OIDC prevedono che il campo `scope` sia valorizzato con il valore `openid`. Se così non fosse, la richiesta viene elaborata secondo le modalità previste da OAuth.

I flussi di dati previsti sono molteplici e possono essere rappresentati con un Diagramma delle Attivita: la figura 3.6 mostra l'esempio in cui `response_type` sia uguale a `code`.

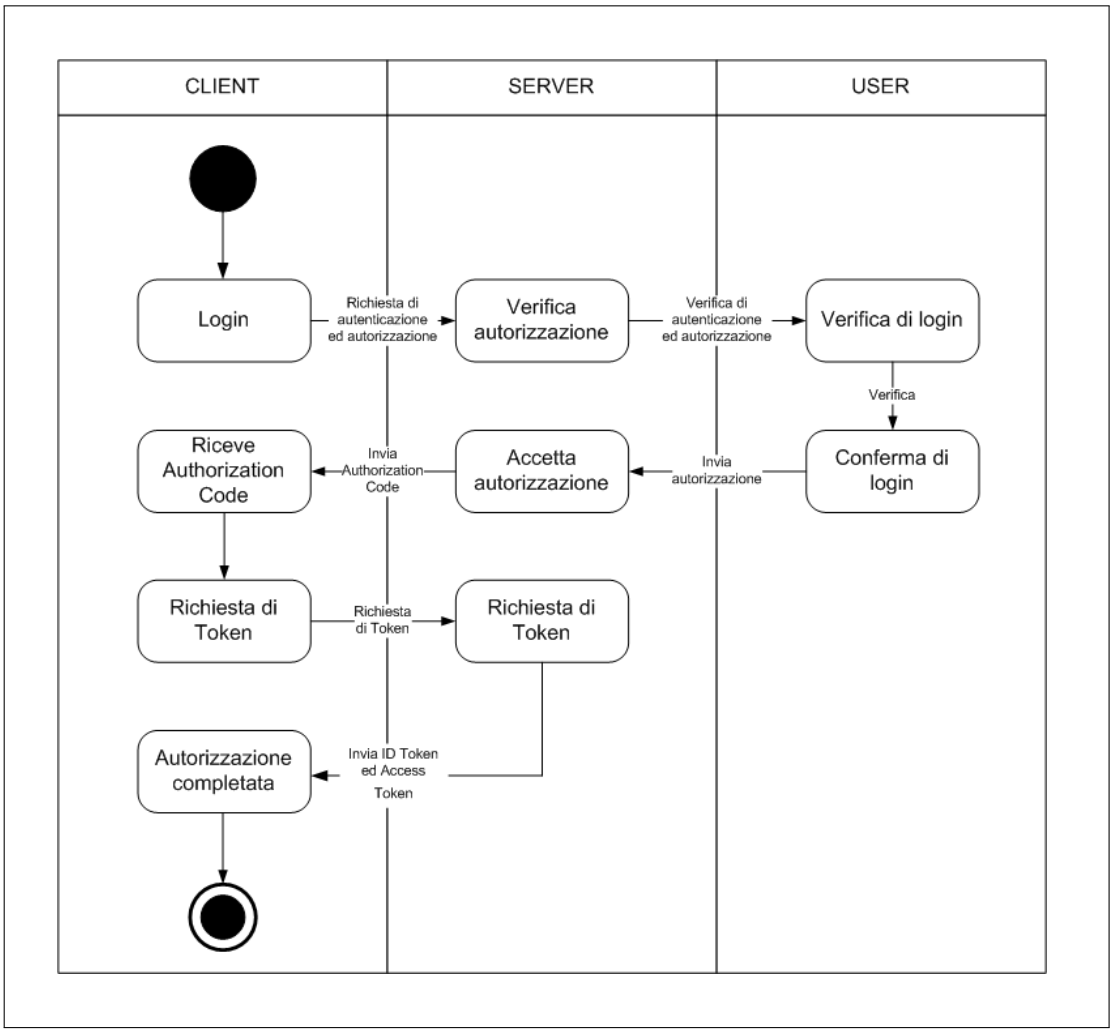

Figura 3.6: Diagramma delle Attivita di OIDC, per il caso in cui `response_type` impostato a `code`.

3.4 *Distributed Ledger Technology* (DLT)

DLT è un sistema digitale per registrare dati in modo distribuito, dove ogni nodo elabora e verifica i record. *La tecnologia può essere applicata in svariati settori, ad esempio in* Self-Sovereign Identity *(SSI).*

Distributed Ledger Technology (DLT) può essere tradotto in "Tecnologia di Registro Distribuito" e si riferisce all'infrastruttura tecnologica ed ai protocolli che consentono l'accesso, la convalida e l'aggiornamento simultaneo dei *record*. DLT garantisce la sicurezza dei dati, grazie alla crittografia, mentre la sua struttura assicura la loro immutabilità.

Come mostrato in figura 3.7, in DLT il registro è distribuito, quindi presente in ogni nodo della rete. La distribuzione garantisce fiducia e trasparenza, perché ogni nodo può visualizzarlo, modificarlo e verificarlo. Modelli diversi di registri sono chiamati *Centralized Ledger*: si caratterizzano con la presenza di un ente accentratore federato, che mantiene il registro e ne controlla gli accessi.

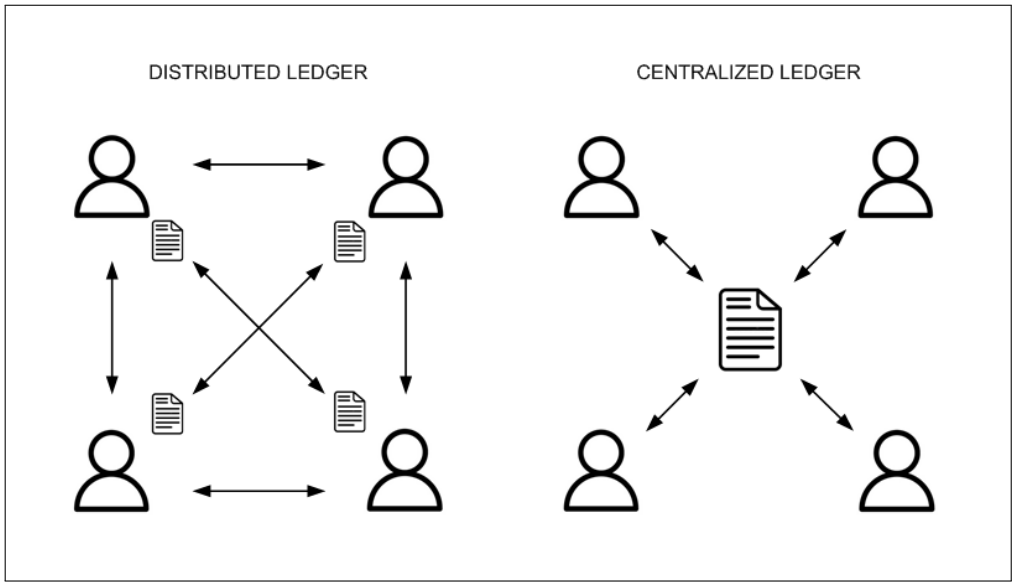

Figura 3.7: Confronto tra *Distributed Ledger* e *Centralized Ledger*. In un *Distributed Ledger*, i dati sono presenti presso tutti i nodi. In un *Centralized Ledger*, i dati sono presenti in un luogo comune, al quale tutti i nodi accedono.

La *blockchain* è una tipologia di *Distributed Ledger Technology* (DLT). Ciascun *record* di dato è inteso come un blocco immutabile. Ciascun blocco contiene i riferimenti del blocco precedente, creando una concatenazione di blocchi, come illustrato in figura 3.8. Il criptaggio del registro garantisce l'**immutabilità** e la **riservatezza** dei dati.

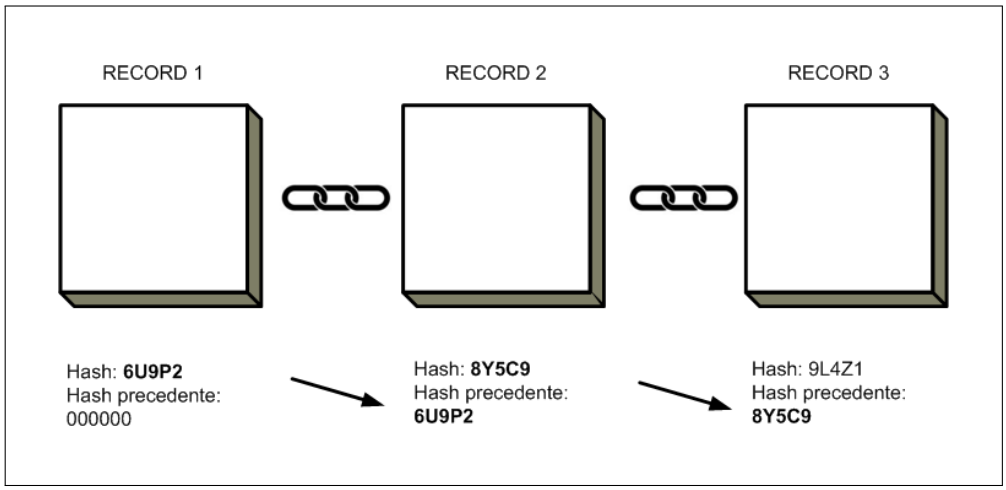

Figura 3.8: Raffigurazione di una *blockchain*. Ciascun blocco è identificato con un codice *hash* e contiene il codice *hash* del blocco precedente. I blocchi risultano quindi collegati, similmente ad una catena.

DLT porta notevoli vantaggi agli archivi di dati, riducendo gli errori ed i tempi di manutenzione:

- Aumento di visibilità e trasparenza dei dati contenuti nel registro;
- Riduzione dei costi operativi, grazie all'eliminazione di un'autorità centrale che governa il registro;
- Aumento di velocità di transazione, grazie alla riduzione dei tempi di aggiornamento dei registri;
- Riduzione di rischi di attività fraudolenta, manomissione e manipolazione;
- Aumento di affidabilità e resilienza agli errori, grazie alla mancanza di un ente accentratore;
- Aumento dei livelli di sicurezza.

La tecnologia, pur facilitando i processi di lettura e scrittura, necessita di procedure di verifica. Il controllo dei registri e l'allineamento con le copie decentralizzate richiedono risorse e tempi elevati, che possono influire negativamente sulle prestazioni.

Il concetto di registro è nato secoli orsono, basti pensare ai primi registri contabili. DLT ha ricevuto un notevole interesse a partire dal 2009, con l'introduzione nel mercato della criptovaluta *bitcoin*, che utilizza la *blockchain*. La nuova moneta ha dimostrato le potenzialità della *blockchain*, rimanendo comunque sicura. La tecnologia *blockchain* si sta diffondendo in numerosi settori, grazie all'attività di Hyperledger. La sua applicazione per l'identità digitale sfocia in *Self-Sovereign Identity* (SSI), garantendo sicurezza, immutabilità e riservatezza.

3.5 *Self-Sovereign Identity* (SSI)

Self-Sovereign Identity *(SSI) è un nuovo modello per la gestione delle identità digitali e dei documenti che ne provano l'esistenza. Nasce per garantire sicurezza e* privacy *agli utenti.*

Self-Sovereign Identity (SSI) si traduce in "identità autonoma" ed è un modello per la gestione delle identità digitali, dove le entità intestatarie di un'identità digitale hanno il controllo completo dei dati.

In altri modelli, è presente un ente intermediario, che riconosce l'identità e ne autorizza l'utilizzo. L'ente intermediario esercita potere sull'utilizzo dell'identità digitale e ne raccoglie informazioni, nonostante siano dati personali. In SSI, l'ente intermediario non è presente: i dati dell'identità possono essere archiviati sui dispositivi dell'utente, che può fornirli a richiesta.

SSI non sarebbe possibile senza due elementi:

- il *World Wide Web Consortium* (W3C), incaricato di uniformare il *format* delle credenziali della firma digitale;
- le *blockchain* pubbliche, che sfruttano le tecnologie di distribuzione.

SSI vuole garantire l'**autenticità** e l'**integrità** del documento che attesta l'identità: un documento è autentico se è valido e regolare, mentre è integro se è stato emesso da qualcuno di autorevole. Inoltre, la custodia ed il controllo delle informazioni personali rimangono all'utente proprietario.

La definizione di SSI è stata coniata da Christopher Allen, esperto di *security*, *privacy* e identità *online*, nel 2016. Nella sua pubblicazione, si teorizzano i principi di un nuovo modello di gestione dell'identità digitale:

1. **Esistenza**: un'identità digitale deve essere sempre associata ad un utente fisico;
2. **Controllo**: l'utente ha il dominio completo dell'identità digitale;
3. **Accesso**: l'utente deve essere in grado di recuperare agilmente i propri dati, in qualsiasi momento;

4. **Trasparenza**: gli algoritmi alla base di SSI devono essere gratuiti, *open source*, ben noti ed indipendenti, per garantire trasparenza sul loro funzionamento ed aggiornamento;
5. **Persistenza**: idealmente, l'identità dovrebbe durare per sempre o comunque fino a che l'utente desidera;
6. **Portabilità**: le informazioni sull'utente e sulla sua identità non devono essere vincolate ad un'entità digitale né ad una specifica giurisdizione;
7. **Interoperabilità**: le identità digitali dovrebbero essere utilizzabili a livello globale e non limitate a determinate attività o settori;
8. **Consenso**: la condivisione dei dati identificativi con altri attori deve avvenire esclusivamente con il consenso dell'utente;
9. **Minimizzazione**: la condivisione dei dati deve essere ridotta al minimo indispensabile;
10. **Protezione**: i diritti e le libertà delle persone hanno la priorità sulle esigenze della rete a supporto del modello SSI.

In SSI, si identificano tre ruoli utente:

- *Issuer*: autorità che emette il documento;
- *Holder*, detto anche "*Subject*": persona intestataria del documento / credenziali;
- *Verifier*: colui che ha richiesto il documento e vuole verificarlo, visionarlo o utilizzarlo (ad esempio, per evadere una pratica).

Tra i tre ruoli, si instaura un legame di fiducia, chiamato "triangolo della fiducia" (schematizzato in figura 3.9), dove:

- *Issuer* si fida di *Holder*, poiché ha accertato i dati al momento dell'emissione del documento;
- *Holder* si fida di *Verifier*, poiché fornisce il documento da verificare;
- *Verifier* e *Holder* si fidano di *Issuer*, perché *Issuer* è autorevole ad emettere credenziali;

- *Verifier* si fida *Holder*, per proprietà transitiva: dato che *Issuer* si è fidato di *Holder*, anche *Verifier* si fida.

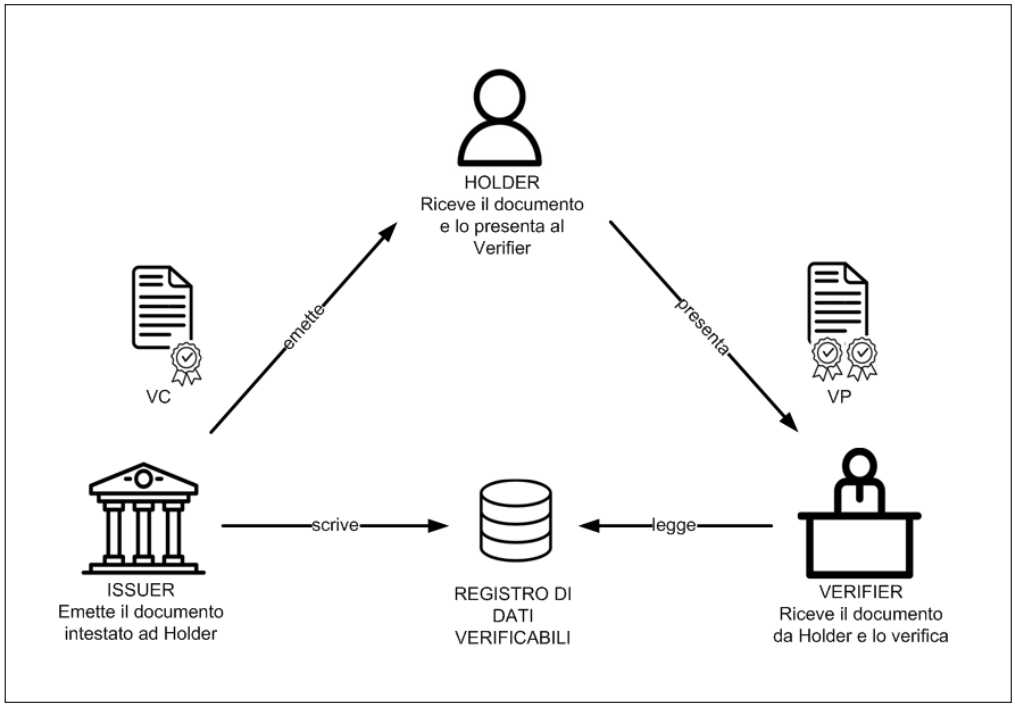

Figura 3.9: Il "triangolo della fiducia": il rapporto che si instaura tra *Issuer*, *Holder* e *Verifier* è fondamentale per la validità dei documenti.

I tre ruoli sono identificabili anche nella quotidianità. Ad esempio, il Comune rilascia la carta di identità e ottiene il ruolo di *Issuer*, perché ha emesso il documento. Per ritirare una raccomandata all'ufficio postale, si deve presentare la carta d'identità: la Posta ottiene il ruolo di *Verifier*, perché richiede un documento. L'addetto postale effettua delle verifiche sulla carta d'identità e dopo aver superato i controlli, la raccomandata viene consegnata. In questo processo, l'operatore non verifica che il Comune abbia effettivamente emesso il documento all'individuo, ma suppone che ciò sia vero, poiché il documento stesso è risultato integro.

Allo stesso modo, nel modello SSI, un *Issuer* firma e rilascia una credenziale digitale (VC) che attesta un fatto relativo alla persona (*Holder*) e la invia al suo *wallet* digitale. Per far accedere un *Holder* ad un servizio *online*, il *Verifier* verifica uno o più fatti relativi all'utente, le cui attestazioni sono nel suo *wallet*. Le credenziali vengono presentate dall'*Holder* al *Verifier*: quest'ultimo ne verifica l'autenticità attraverso il

controllo della firma. Il tutto avviene senza richiedere il permesso all'autorità che ha emesso il documento VC (*Issuer*).

Il modello SSI è composto da quattro elementi:

- *Decentralized Identifier* (DID);
- *DID Document*;
- *Verificable Credential* (VC);
- *Verificable Presentation* (VP).

Il *Decentralized Identifier* (DID) è un identificativo, composto da un codice alfanumerico basato su un sistema a doppia chiave crittografica, memorizzato su *blockchain*. Il suo scopo è di identificare univocamente un'entità *online*. Essendo un codice, non contiene informazioni aggiuntive. I DID sono emessi, detenuti e controllati dagli individui. Il DID è una stringa, composta da 3 elementi:

- *scheme*;
- *method*;
- *payload*, detto anche "*method-specific Identifier*".

Il DID appare come segue: `scheme:method:payload`. La struttura è simile a quella di un dominio *internet*, ad esempio: `com.google.gmail`.

Il *DID Document* è un *file* JSON, che contiene i dati relativi all'identità. Di seguito un esempio di *DID Document*.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
```

```
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Una *Verificable Credential* (VC) è un qualsiasi tipo di attributo collegato a un'entità. Alcuni corrispettivi fisici di una VC sono, per esempio, la patente di guida o il diploma di laurea. In SSI, i VC sono digitali, immodificabili e verificabili. Il modello dei dati per le VC è definito dal W3C "*Verifiable Credentials Data Model v1.1*", pubblicato il 3 marzo 2023. Un VC è formato da quattro elementi:

- *Schema*;
- *Holder*: DID di *Holder*, corrispondente all'intestatario del documento;
- *Issuer*: DID di *Issuer*, corrispondente all'emittente del documento;
- *Proof*: firma di *Holder* ed *Issuer*, firmata con chiave pubblica di *Issuer*.

Verificable Presentation (VP) è un documento che contiene la VC ed altri dati. VP è emesso dall'*Holder* per presentare un documento ad un *Verifier*. Una VP è così composta:

- VC, composta a sua volta da:
  - *Schema*;
  - *Holder*: DID di *Holder*, corrispondente all'intestatario del documento;
  - *Issuer*: DID di *Issuer*, corrispondente all'emittente del documento;
  - *Proof*: la firma di *Holder* ed *Issuer*, firmata con chiave pubblica di *Issuer*.
- Codice per dimostrare delle proprietà;
- Firma di *Holder*.

Dato il DID, è possibile risolverlo per ottenere il *DID Document*.

La risoluzione avviene grazie all'*Universal Resolver* ed il processo può essere descritto sommariamente come segue:

1. Dato lo *scheme*, si identifica il risolutore adatto. Nella fattispecie di un modello SSI, il risolutore dovrà essere specifico per i DID;
2. Grazie al risolutore, dallo *scheme* si ottiene il *method*;
3. Se il risolutore riconosce il *method*, esso fornirà il *DID Document*. Il *DID Document* contiene una serie di parametri, tra cui il *payload*.

Quando il *Verifier* necessita di un documento di un utente, deve richiederlo ad *Holder*. A fronte della richiesta, l'*Holder* invia il documento VC, contenuto all'interno di un documento VP. Il *Verifier* non accetta di *default* quanto ricevuto, ma attua delle verifiche di autenticità ed integrità. Tramite questo processo, si instaura il "triangolo della fiducia". Il processo di verifica che attua il *Verifier* può essere suddiviso in *step*:

1. Il *Verifier* riceve la VP dall'*Holder*;
2. Il *Verifier* deve controllare se la VP è stata inviata realmente dall'*Holder*. Il controllo specifico effettuato è il seguente: se la chiave pubblica che ha firmato la VP corrisponde alla chiave pubblica dell'*Holder*, allora chi ha emesso VP è l'*Holder*;
3. Il *Verifier* deve controllare se chi ha emesso la VC (contenuta all'interno della VP) è autorevole. Il controllo specifico effettuato è il seguente: se la chiave pubblica che ha firmato la VC corrisponde alla chiave pubblica dell'*Issuer*, allora chi ha emesso VC è l'*Issuer*;
4. Se entrambi i controlli sono superati con successo, *Verifier* "si fida" di *Holder* ed accetta la VP e la VC. La fiducia verso *Holder* è ottenuta per proprietà transitiva dalla fiducia che ha verso *Issuer*.

In SSI è assente l'ente intermediario, portando notevoli vantaggi. Infatti, non esiste una collezione di informazioni sensibili e manca il tracciamento delle informazioni personali. In merito alla sicurezza, i dati dell'identità sono locati presso i dispositivi dell'utente. Il processo di utilizzo dei dati è più efficiente rispetto ad quello di altri modelli. Inoltre, SSI è compatibile con il regolamento europeo "GDPR", poiché

implementa *by design* le logiche che tutelano i dati dei cittadini. La sicurezza è garantita dalla tecnologia *blockchain*, creando fiducia tra le diverse parti e fornendo garanzie crittografiche a dimostrazione della validità delle attestazioni. Inoltre, la distribuzione della *blockchain* rende i dati portabili e utilizzabili tramite gli identificatori.

Nonostante i notevoli vantaggi, sono presenti alcuni inconvenienti che non possono essere sottovalutati. Gli utenti sono responsabili della propria sicurezza e spesso quelli meno avvezzi potrebbero non comprendere a fondo i rischi di un uso inappropriato. Inoltre, i dati personali e le loro origini potrebbero essere molteplici, rendendo complesso il tracciamento delle autorizzazioni.

3.6 Hyperledger

Hyperledger sviluppa la blockchain specifiche per diversi settori. Per SSI, sono stati creati progetti dedicati.

"*Hyperledger Foundation*" è un consorzio di aziende impegnate nella diffusione della tecnologia *blockchain* in settori non prettamente informatici, offrendo innovazione ed incrementando la sicurezza e la *privacy* dei dati. Hyperledger lavora per creare *blockchain* specifiche per ogni settore, valorizzando le peculiarità dei *business*.

Creato nel dicembre 2015 dalla Linux Foundation, Hyperledger inizialmente contava trenta membri. Attualmente, comprende più di duecento aziende *leader* nei settori della finanza, banche, Internet delle cose (IoT), gestione della catena di approvvigionamento, produzione e tecnologia. Tra loro vantano aziende rinomate come: Bosch, IBM, Samsung, Microsoft, Hitachi, American Express, JP Morgan e Visa, oltre a un gran numero di *startup* basate su *blockchain*, come Blockforce e ConsenSys.

Figura 3.10: Il logo di *Hyperledger Foundation*. Immagine scaricata dal sito internet ufficiale di Hyperledger nel luglio 2023.

Le attività di Hyperledger promuovono diffusione, sviluppo e supporto delle *blockchain*. Ad esempio, le principali attività si possono riassumere in quattro punti:

- Creare ***framework open source***, che sfruttino DLT a livello aziendale, fornendo basi di codice a supporto delle transazioni;
- Fornire un'**infrastruttura neutrale**, aperta e guidata dalla comunità supportata da una gestione tecnica e commerciale;
- Creare **comunità tecniche** per sviluppare *blockchain* e PoC di registro condiviso, casi d'uso, percorsi sul campo e implementazioni;
- **Educare il pubblico** sulle opportunità di mercato per la tecnologia *blockchain*.

Generalizzando, la fondazione sfrutta le potenzialità delle *blockchain* per costruire una nuova generazione di applicazioni transazionali che stabiliscano fiducia, responsabilità e trasparenza, snellendo i processi aziendali ed i vincoli legali.

Hyperledger è impegnato nella promozione della SSI, grazie a tre progetti dedicati, allo scopo di garantire sicurezza, *privacy* e fiducia tra le parti. Relativamente all'identità digitale, si citano:

- Hyperledger Indy;
- Hyperledger Aries;
- Hyperledger Ursa.

Hyperledger Indy è stato il primo *framework* sviluppato dalla fondazione dedicato all'identità. Fornisce strumenti, librerie e componenti riutilizzabili per la gestione dell'identità decentralizzata digitale. Utilizza DLT per creare un registro distribuito, accompagnato da strumenti, librerie e componenti riutilizzabili per la creazione e l'utilizzo delle identità digitali, radicate su *blockchain*. In questo modo, le identità sono interoperabili tra domini amministrativi ed applicazioni differenti. I casi d'uso considerano attentamente alcuni requisiti fondamentali, tra cui prestazioni, scalabilità ed affidabilità. In particolare, la *privacy* è requisito di fondamentale importanza per un registro pubblico delle identità, che potenzialmente può scalare globalmente. Evernym è stato il creatore e tuttora uno dei principali contributori di Hyperledger Indy, costribuendo con il codice sviluppato per la rete Sovrin, da cui nacque Hyperledger Indy.

Figura 3.11: Il logo di Hyperledger Indy. Immagine scaricata dal sito internet ufficiale di Hyperledger Indy nel luglio 2023.

Hyperledger Ursa è una libreria crittografica condivisa utilizzabile sia dai progetti Hyperledger sia da qualsiasi altro *software* che necessiti di una base crittografica solida e controllata. La libreria è un *repository opt-in* per utilizzare la crittografia.

Figura 3.12: Il logo di Hyperledger Ursa. Immagine scaricata dal sito internet ufficiale di Hyperledger Ursa nel luglio 2023.

Hyperledger Aries fornisce un *toolkit* progettato per le soluzioni incentrate sulla creazione, trasmissione, archiviazione ed utilizzo di credenziali verificabili. Sostanzialmente, Hyperledger Aries è un'infrastruttura per interazioni *peer-to-peer* (P2P) radicate su *blockchain*. Molti sono i *framework* che lo implementano: durante il progetto, è stato approfondito Aries JS.

Figura 3.13: Il logo di Hyperledger Aries. Immagine scaricata dal sito internet ufficiale di Hyperledger Aries nel luglio 2023.

3.7 Aries JS

In Hyperledger, la creazione e la comunicazione delle credenziali sono compiti assegnati ad Hyperledger Aries. Diversi sono i framework che lo implementano: è stato studiato Aries JS.

Hyperledger Aries è un *software* sviluppato dalla *Hyperledger Foundation* che si occupa della creazione, trasmissione, archiviazione ed utilizzo di credenziali verificabili. Aries JS è un *framework open source* che rende disponibili le funzionalità di Hyperledger Aries tramite JavaScript.

Aries JS sfrutta il supporto crittografico fornito da Hyperledger Ursa per garantire la gestione sicura dei codici *secret* e la gestione decentralizzata delle chiavi. Il codice sorgente è disponibile *online* nella *repository* pubblica in GitHub, mentre la documentazione è consultabile nel sito ufficiale. Il *framework* è distribuito grazie l'*Apache License Version* 2.0 (Apache-2.0).

Il *framework* in JavaScript è nato per rendere conosciuta ed accessibile la *Self-Sovereign Identity* (SSI). Ciò è agevolato grazie ad un linguaggio diffuso: si sfrutta quindi la popolarità di JavaScript per propagare l'identità decentralizzata. Inoltre, la flessibilità di JavaScript lo rende adattabile a molte esigenze, dato che è utilizzabile anche per applicativi *server-side*. Il *framework* è compatibile con React Native e con Node.JS. Durante lo *stage*, è stato utilizzato Node.JS.

Aries JS si occupa fondamentalmente di creare un canale di comunicazione crittografato tra *Issuer* ed *Holder*, emmettere delle credenziali ed inviarle. Tutto ciò avviene grazie ad alcuni componenti specifici:

- *agent*;
- DIDComm;
- *wallet*.

L'***agent*** è il componente che permette la connessione con altri servizi. Un *agent* Aries ha tre caratteristiche essenziali:

- Agisce come fiduciario per conto di un singolo proprietario di identità (*Holder*);
- Contiene chiavi crittografiche che incarnano in modo univoco la sua autorizzazione delegata;
- Interagisce utilizzando protocolli DIDComm interoperabili.

Pertanto, un *agent*:

- Agisce per conto dell'*Holder* e dell'*Issuer* per creare connessioni, emettere credenziali, inviare messaggi ed altro;
- Dispone di un *toolkit* crittografico con cui può operare in modo univoco, sicuro e verificabile;
- Interagisce con altre entità (ad esempio, un altro *agent*) tramite i protocolli DIDComm.

DIDComm è un protocollo che permette la comunicazione sicura, privata ed indipendente dal trasporto tra *Issuer* ed *Holder*. Il DID è un codice alfanumerico, che deve essere risolto per ottenere informazioni aggiuntive (per ottenere il *DID Document*). Ad esempio, la conversione di un *Decentralized Identifier* (DID) in *DID Document* avviene grazie al protocollo DIDComm.

Issuer ed *Holder* sono dotati di un *wallet*, installato sul proprio dispositivo. Complessivamente, abilita la gestione delle identità digitali e l'interazioni con altri attori. Ad esempio, *Issuer* emetterà le credenziali dal *wallet*, mentre *Holder* lo userà per collezionare i dati dell'identità digitale ed i documenti relativi.

Issuer ed *Holder* necessitano di un canale di comunicazione crittografato per poter colloquiare in totale sicurezza. Sono necessarie le seguenti azioni per stabilire un canale:

1. Configurazione di un *wallet* e di un *agent* sia per *Issuer* che per *Holder*;
2. Creazione di un invito al collegamento (chiamato "*invitation*") da parte di *Issuer* e successivo invio ad *Holder*;

3. Ricezione dell'*invitation* da parte di *Holder*, che lo accetta e risponde positivamente ad *Issuer*;
4. Ricezione del riscontro verso *Issuer*.

In questo modo, i due soggetti hanno stabilito un canale di comunicazione crittografato, in cui possono scambiarsi documenti e credenziali in modo sicuro.

Una credenziale è un documento emesso dal *wallet* dell'*Issuer* verso il *wallet* dell'*Holder*. L'emissione di una credenziale si svolge come di seguito descritto:

1. Realizzazione di un canale di comunicazione tra *Issuer* ed *Holder*;
2. Registrazione di uno *scheme* da parte di *Issuer*, equivalente al "progetto" della credenziale, e la sua associazione ad un emittente specifico, chiamata "definizione della credenziale";
3. Nel frattempo, *Holder* è in ascolto di eventuali credenziali in entrata;
4. Emissione della credenziale, da parte di *Issuer*;
5. Ricezione della credenziale, da parte di *Holder*, il quale la accetta e risponde positivamente all'*Issuer*.

La definizione della credenziale esplicita l'*Issuer*, assicurando che l'emittente della credenziale sia chi è dichiarato nella definizione: ciò è un requisito fondamentale per l'instaurarsi del triangolo della fiducia.

3.8 Indicio

Indicio promuove la diffusione delle identità digitali, basate su blockchain. Le credenziali verificabili sono generate e gestite grazie a Hyperledger Indy ed Hyperledger Aries.

Indicio è una società fondata a Seattle, Washington, nel 2020, che sviluppa soluzioni di identità digitale. Lo sviluppo è impostato sull'utilizzo di *blockchain* e *provider* di rete decentralizzati affidabili. Indicio è *leader* di mercato nello sviluppo di ecosistemi digitali attendibili ed *open source*, per la condivisione di dati verificabili e lo sviluppo di relazioni affidabili e sicure, grazie anche ad un approccio *Zero Trust*. Promuove l'identità decentralizzata come bene pubblico, per consentire il controllo dei dati delle proprie identità digitali.

Indicio ha realizzato quattro *network* per lo sviluppo delle identità digitali, costruite ed operative con i *framework* Hyperledger Aries. Le reti sono supportate da diverse *community* dislocate globalmente.

Una rete indispensabile per il *testing* di nuove integrazioni è la Indicio TestNet. La *network* è gratuita ed è dedicata allo svillupo di PoC che utilizzano Hyperledger Indy ed Hyperledger Aries.

Hyperledger Indy ed Indicio operano assieme per diffondere lo scambio di credenziali verificabili.

Figura 3.14: Il logo di Indicio. Immagine scaricata dal sito internet ufficiale di Indicio nel luglio 2023.

Capitolo 4

Analisi dei requisiti del PoC

Data la crescente esigenza di Self-Sovereign Identity (SSI), si vuole integrare il suo utilizzo nell'applicativo Monokee SSO, sviluppato da Athesys srl. Si stilano i requisiti necessari alla creazione di un PoC di una API, che utilizzi il framework Aries JS.

L'applicativo Monokee SSO è una completa soluzione IAM. Athesys srl necessita di un'API REST, che utilizzi Hyperledger Aries per l'emissione e la gestione di credenziali di tipo Hyperledger. Allo scopo, esistono diversi *framework*: si è scelto di utilizzare Aries JS. Durante lo *stage*, è stato ideato e sviluppato un PoC, per valutare la fattibilità e le potenzialità di Aries JS.

L'API deve essere di tipo REST. Sostanzialmente, il prodotto finale deve contenere i comandi descritti dal *tutorial* di Aries JS[1], che dovranno essere richiamati tramite specifiche chiamate HTTP GET e POST. L'API deve essere suddivisa in tre servizi, con le seguenti caratteristiche:

- "**Servizio *Server***": si pone come l'utilizzatore dell'API. Utilizza "Servizio *Issuer*" e "Servizio *Holder*" per dimostrarne le funzionalità;
- "**Servizio *Issuer***": insieme di funzionalità che istanziano i componenti necessari ad un utente *Issuer*;

[1] *Aries JS, 2023, Tutorials, ultima visita in luglio 2023.* URL: https://aries.js.org/guides/0.4/tutorials.

- "**Servizio *Holder***": insieme di funzionalità che istanziano i componenti necessari ad un utente *Holder*.

La suddivisione permette di separare nettamente le funzionalità riservate ad attori specifici.

Gli utenti utilizzatori dell'API hanno una buona conoscenza di: SSI, identità digitale decentralizzata, JavaScript. Generalmente, è un utente che ha necessità di implementare un servizio, utilizzando le credenziali fornite da Hyperledger Aries.

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. Un Diagramma dei Casi d'Uso (in inglese "*Use Case Diagram*") è un diagramma di tipo *Unified Modeling Language* (UML), dedicato alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono con il sistema stesso. Le funzionalità del prodotto sono suddivise tre servizi, ciascuno destinato ad un attore specifico:

- **Attore "Sviluppatore"**: attore che utilizza il servizio "Servizio *Server*";
- **Attore "*Issuer*"**: attore che utilizza il servizio "Servizio *Issuer*" e che emetterà la credenziale;
- **Attore "*Holder*"**: attore che utilizza il servizio "Servizio *Holder*" e che riceverà la credenziale.

Gli *use case* sono raffigurati in diagrammi nelle figure 4.1, 4.2 e 4.3.

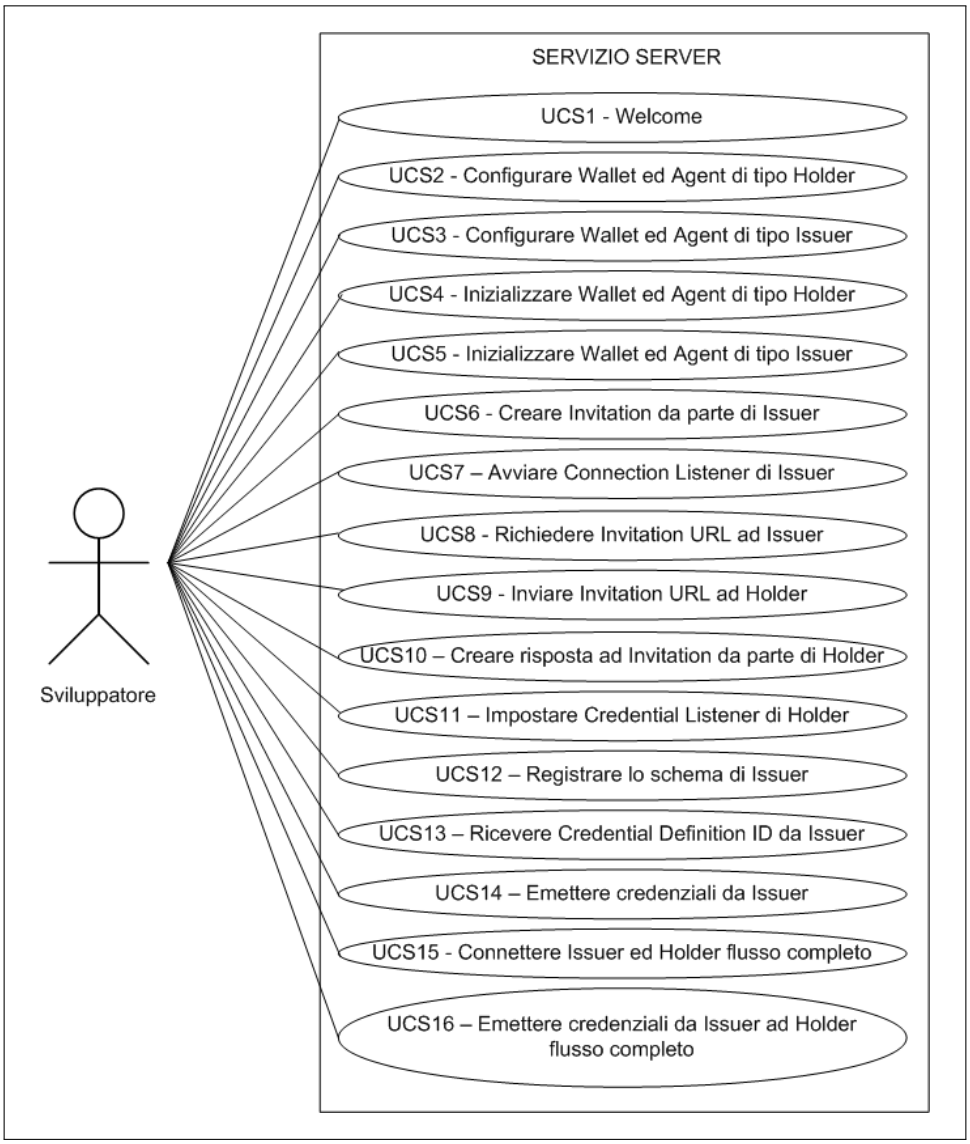

Figura 4.1: *Use case* del servizio *Server*.

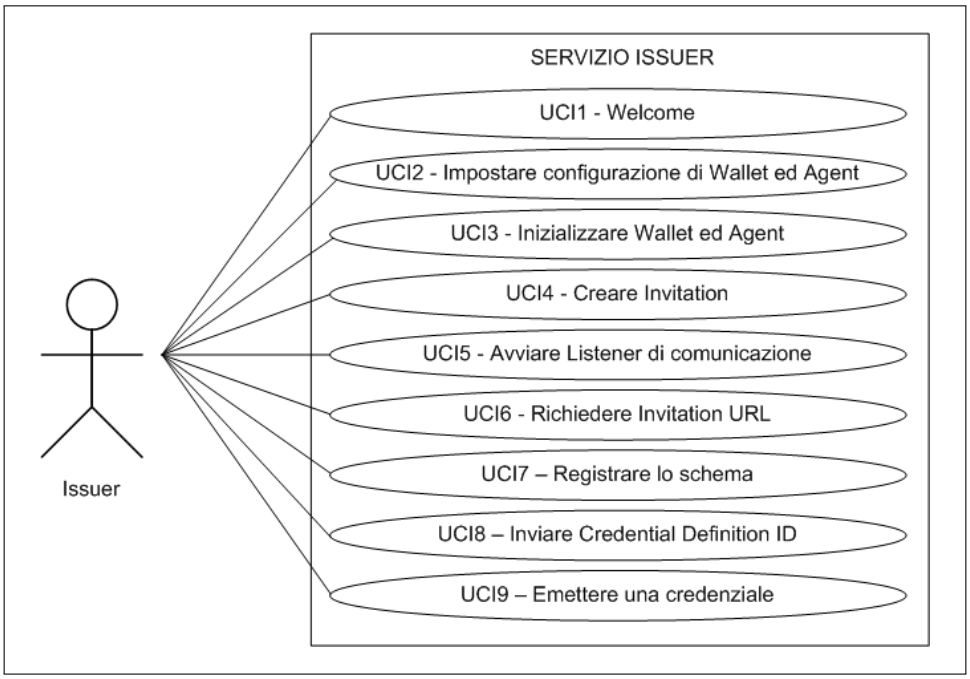

Figura 4.2: *Use case* del servizio *Issuer*.

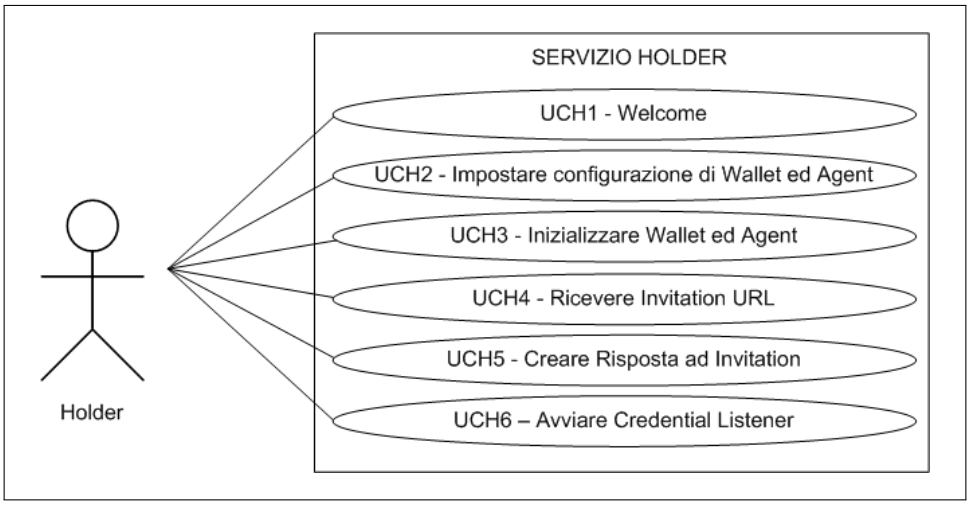

Figura 4.3: *Use case* del servizio *Holder*.

UCS1: Welcome

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" è attivo.

Descrizione: il "Servizio *Server*" riceve una richiesta generica e risponde con una stringa di benvenuto.

Postcondizioni: l'utente visualizza una stringa di benvenuto.

UCS2: Configurare *Wallet* ed *Agent* di tipo *Holder*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di configurare *wallet* ed *agent* di *Holder*. La richiesta viene inoltrata al "Servizio *Holder*". Il "Servizio *Holder*" ritorna l'esito dell'operazione.

Postcondizioni: Il "Servizio *Server*" conosce lo stato di configurazione di *wallet* ed *agent* di tipo *Holder*.

UCS3: Configurare *Wallet* ed *Agent* di tipo *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di configurare *wallet* ed *agent* di *Issuer*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato di configurazione di *wallet* ed *agent* di tipo *Issuer*.

UCS4: Inizializzare *Wallet* ed *Agent* di tipo *Holder*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di inizializzare *wallet* ed *agent* di

Holder. La richiesta viene inoltrata al "Servizio *Holder*". Il "Servizio *Holder*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato di inizializzazione di *wallet* ed *agent* di tipo *Holder*.

UCS5: Inizializzare *Wallet* ed *Agent* di tipo *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di inizializzare *wallet* ed *agent* di *Issuer*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato di inizializzazione di *wallet* ed *agent* di tipo *Issuer*.

UCS6: Creare *Invitation* da parte di *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di creare una *Invitation* da parte dell'*Issuer*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato dell'*Invitation* generata dall'*Issuer*.

UCS7: Avviare *Connection Listener* di *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di avviare il *Listener* di richieste di *Issuer*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato della risposta generata dall'*Issuer*.

UCS8: Richiedere *Invitation URL* ad *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di conoscere l'*Invitation URL*. Il dato è conosciuto da *Issuer*, quindi viene richiesto al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'*Invitation URL*.

Postcondizioni: il "Servizio *Server*" conosce l'*Invitation URL*.

UCS9: Inviare *Invitation URL* ad *Holder*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta di inviare l'*Invitation URL* ad *Holder*. Il dato viene inviato al "Servizio *Holder*". Il "Servizio *Holder*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" conosce l'*Invitation URL*.

UCS10: Creare risposta ad *Invitation* da parte di *Holder*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta che *Holder* crei una risposta ad una *Invitation*. La richiesta viene inoltrata al "Servizio *Holder*". Il "Servizio *Holder*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato della risposta generata dall'*Holder*.

UCS11: Impostare *Credential Listener* di *Holder*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta che *Holder* avvii il *Credential Listener*. La richiesta viene inoltrata al "Servizio *Holder*". Il "Servizio *Holder*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato d'avvio del *Credential Listener*.

UCS12: Registrare lo *schema* di *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta che *Issuer* registri lo *scheme*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato della registrazione dello *scheme*.

UCS13: Ricevere il *Credential Definition ID* da *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta che *Issuer* fornisca il *Credential Definition ID*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce il *Credential Definition ID*.

UCS14: Emettere credenziali da *Issuer*

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*" ed il "Servizio *Issuer*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta che *Issuer* emetta una credenziale verso *Holder*. La richiesta viene inoltrata al "Servizio *Issuer*". Il "Servizio *Issuer*" ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Server*" conosce lo stato dell'invio della credenziale.

UCS15: Connettere *Issuer* ed *Holder* flusso completo

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*", il "Servizio *Issuer*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta connettere *Issuer* ed *Holder* tra loro, provvedendo a fare tutti i settaggi necessari. Vengono fatte le opportune richieste a "Servizio *Issuer*" e "Servizio *Holder*". Entrambi i servizi ritornano l'esito delle operazioni.

Postcondizioni: il "Servizio *Server*" ha connesso tra loro "Servizio *Issuer*" e "Servizio *Holder*".

UCS16: Emettere credenziali da *Issuer* ad *Holder* flusso completo

Attori Principali: utente sviluppatore.

Precondizioni: il "Servizio *Server*", il "Servizio *Issuer*" ed il "Servizio *Holder*" sono attivi.

Descrizione: il "Servizio *Server*" riceve la richiesta emettere una credenziale da *Issuer* ad *Holder*, provvedendo a tutti i settaggi necessari. Vengono fatte le opportune richieste a "Servizio *Issuer*" e "Servizio *Holder*". Entrambi i servizi ritornano l'esito delle operazioni.

Postcondizioni: il "Servizio *Server*" ha connesso tra loro "Servizio *Issuer*" e "Servizio *Holder*". Inoltre, *Issuer* ha emesso una credenziale, che *Holder* ha accettato.

UCI1: Welcome

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve una richiesta generica e risponde con una stringa di benvenuto.

Postcondizioni: l'utente visualizza una stringa di benvenuto.

UCI2: Impostare configurazione di *Wallet* ed *Agent*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve i dati per configurare *wallet* ed *agent*. Il "Servizio *Issuer*" configura *wallet* ed *agent* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" conosce lo stato di configurazione di *wallet* ed *agent*.

UCI3: Inizializzare *Wallet* ed *Agent*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve l'ordine di inizializzare *wallet* ed *agent*. Il "Servizio *Issuer*" inizializza *wallet* ed *agent* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha attivi *wallet* ed *agent*.

UCI4: Creare *Invitation*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve l'ordine di creare un *Invitation*. Il "Servizio *Issuer*" inizializza un *Invitation* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha creato un *Invitation*.

UCI5: Avviare *Listener* di comunicazione

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve l'ordine di avviare il *Listener* ad eventuali richieste in entrata. Il "Servizio *Issuer*" inizializza un *Listener* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha attivato un *Listener*.

UCI6: Richiedere *Invitation URL*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve la richiesta di comunicare l'*Invitation URL*. Il "Servizio *Issuer*" invia l'*Invitation URL* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha comunicato l'*Invitation URL*.

UCI7: Registrare lo *schema*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve la richiesta di registrare lo *scheme*. Il "Servizio *Issuer*" lo registra e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha registrato lo *scheme*.

UCI8: Inviare *Credential Definition ID*

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve la richiesta di fornire il *Credential Definition ID*. Il "Servizio *Issuer*" lo fornisce e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha comunicato il *Credential Definition ID*.

UCI9: Emettere una credenziale

Attori Principali: utente *Issuer*.

Precondizioni: il "Servizio *Issuer*" è attivo.

Descrizione: il "Servizio *Issuer*" riceve la richiesta di inviare una credenziale ad *Holder*. Il "Servizio *Issuer*" invia la credenziale e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Issuer*" ha inviato una credenziale.

UCH1: Welcome

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve una richiesta generica e risponde con una stringa di benvenuto.

Postcondizioni: l'utente visualizza una stringa di benvenuto.

UCH2: Impostare configurazione di *Wallet* ed *Agent*

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve i dati per configurare *wallet* ed *agent*. Il "Servizio *Holder*" configura *wallet* ed *agent* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" conosce lo stato di configurazione di *wallet* ed *agent*.

UCH3: Inizializzare *Wallet* ed *Agent*

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve l'ordine di inizializzare *wallet* ed *agent*. Il "Servizio *Holder*" inizializza *wallet* ed *agent* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" ha attivi *wallet* ed *agent*.

UCH4: Ricevere *Invitation URL*

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve l'*Invitation URL*. Il "Servizio *Holder*" imposta l'*Invitation URL* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" ha ricevuto l'*Invitation URL*.

UCH5: Creare risposta ad *Invitation*

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve l'ordine di creare una risposta ad un eventuale *Invitation*. Il "Servizio *Holder*" inizializza una risposta e successivamente ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" ha attiva una risposta.

UCH6: Avviare *Credential Listener*

Attori Principali: utente *Holder*.

Precondizioni: il "Servizio *Holder*" è attivo.

Descrizione: il "Servizio *Holder*" riceve l'ordine di avviare il *Credential Listener* per eventuali credenziali in entrata. Il "Servizio *Holder*" inizializza un *Listener* e ritorna l'esito dell'operazione.

Postcondizioni: il "Servizio *Holder*" ha attiva un *Listener*.

4.2 Requisiti

Da un'attenta analisi dei requisiti e degli *use case* effettuata sul progetto, è stata stilata la tabella che traccia i requisiti in rapporto agli *use case*.
Sono stati individuati diversi tipi di requisiti, sono stati categorizzati ed identificati.

Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli *Use Case* delineati in fase di analisi.

Tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	**Descrizione**	**Fonte**
RFN-1	L'API deve essere suddivisa in tre servizi: "Servizio *Server*" (per dimostrare e pilotare le funzionalità degli altri servizi), "Servizio *Issuer*" (che agisce come utente *Issuer*) e "Servizio *Holder*" (che agisce come utente *Holder*)	SAL 2023/05/09
RFN-2	L'API permette di ricevere la configurazione di *agent* e *wallet* di *Holder* dal "*Servizio Server*"	UCS2
RFN-3	L'API permette di ricevere la configurazione di *agent* e *wallet* di *Issuer* dal "*Servizio Server*"	UCS3
RFN-4	L'API permette di inizializzare *agent* e *wallet* di *Holder* dal "*Servizio Server*"	UCS4
RFN-5	L'API permette di inizializzare *agent* e *wallet* di *Issuer* dal "*Servizio Server*"	UCS5
RFN-6	L'API permette di definire un *Invitation* dal "*Servizio Server*" per l'*Issuer*	UCS6
RFN-7	L'API permette di avviare il *Connection Listener* dal "*Servizio Server*" per l'*Issuer*	UCS7
RFN-8	L'API permette di richiedere l'*Invitation URL* dal "*Servizio Server*" all'*Issuer*	UCS8
RFN-9	L'API permette di inviare l'*Invitation URL* dal "*Servizio Server*" all'*Holder*	UCS9
RFN-10	L'API permette di creare una risposta all'*Invitation* da parte di *Holder*, tramite "*Servizio Server*"	UCS10
RFN-11	L'API permette di impostare il *Credential Listener* di *Holder*, tramite "*Servizio Server*"	UCS11
RFN-12	L'API permette di registrare lo *schema* di *Issuer*, tramite "*Servizio Server*"	UCS12

Requisito	**Descrizione**	**Fonte**
RFN-13	L'API permette ad *Issuer* di fornire il *Credential Definition ID*, tramite "*Servizio Server*"	UCS13
RFN-14	L'API permette ad *Issuer* di emettere credenziali, tramite "*Servizio Server*"	UCS14
RFN-15	L'API permette ad *Issuer* ed *Holder* di connettersi tra loro, tramite un canale di comunicazione pilotato dal "*Servizio Server*"	UCS15
RFN-16	L'API permette ad *Issuer* di emettere una credenziale verso *Holder*, tramite un canale di comunicazione pilotato da "*Servizio Server*"	UCS16
RFN-17	L'API permette di definire un *wallet* per l'*Issuer*	UCI2
RFN-18	L'API permette di definire un *agent* per l'*Issuer*	UCI2
RFN-19	L'API permette di inizializzare il *wallet* di *Issuer*	UCI3
RFN-20	L'API permette di inizializzare l'*agent* di *Issuer*	UCI3
RFN-21	L'API permette all'*Issuer* di creare una *Invitation*	UCI4
RFN-22	L'API permette di avviare il *listener* di comunicazione tra *Issuer* ed *Holder*	UCI5
RFN-23	L'API permette ad *Issuer* di esporre l'*Invitation URL*	UCI6
RFN-24	L'API permette ad *Issuer* di registrare uno *scheme*	UCI7
RFN-25	L'API permette ad *Issuer* di fornire il *Credential Definition ID*	UCI8
RFN-26	L'API permette ad *Issuer* di emettere una credenziale	UCI9
RFN-27	L'API permette di definire un *agent* per l'*Holder*	UCH2
RFN-28	L'API permette di inizializzare il *wallet* di *Holder*	UCH3
RFN-29	L'API permette di inizializzare il *agent* di *Holder*	UCH3
RFN-30	L'API permette ad *Holder* di ricevere un *Invitation URL*	UCH4
RFN-31	L'API permette ad *Holder* di creare una risposta ad una eventuale *Invitation*	UCH5
RFN-32	L'API permette ad *Holder* di avviare un *Credential Listener*, affinché rimanga in ascolto di eventi	UCH6

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	**Descrizione**	**Fonte**
RQD-1	Le prestazioni dell'API deve garantire la giusta esecuzione dei *test*	-
RQD-2	La definizione degli oggetti deve essere consistente	-
RQD-3	I *test* devono essere replicabili	-
RQD-4	Il prodotto verrà progettato e sviluppato seguendo le norme dettate dall'Ingegneria del *Software*	-
RQD-5	Il prodotto sarà costantemente valutato dal *team* di Athesys srl	-

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	**Descrizione**	**Fonte**
RVN-1	Il prodotto deve essere una API REST	SAL 2023/04/17
RVN-2	Il prodotto deve essere una API del *framework* Aries JS	SAL 2023/04/17
RVN-3	Il prodotto deve essere scritto in linguaggio JavaScript	SAL 2023/04/17
RVN-4	Le chiamate al prodotto devono essere di tipo HTTP GET e POST	SAL 2023/04/17
RVN-5	Il prodotto deve essere sviluppato con Express JS	SAL 2023/04/17
RVN-6	Il prodotto deve utilizzare i comandi forniti da Aries JS ed esplicitati nella sua documentazione	SAL 2023/04/17
RVN-7	Le chiamate HTTP dell'Application Program Interface devono essere testate con il *software* Postman	SAL 2023/04/17
RVN-8	Il codice del prodotto deve essere salvato in una *directory* della macchina virtuale, fornita da Athesys srl	SAL 2023/04/17
RVN-9	Il collegamento alla macchina virtuale avviene tramite chiave privata via SSH	SAL 2023/04/17
RVN-10	Il canale di comunicazione utilizzato tra *Holder* ed *Issuer* deve essere crittografato	-
RVN-11	Il prodotto deve essere compatibile con i moderni *browser*	-
RVN-12	Per emettere le credenziali si deve utilizzare la Indicio TestNet	SAL 2023/05/16
RVN-13	Il prodotto deve replicare i *tutorial* presenti nella documentazione di Aries JS	SAL 2023/05/16

Capitolo 5

Progettazione e codifica del PoC

Progettazione del PoC, con suddivisione in sotto-servizi.

5.1 Tecnologie e strumenti

Per sviluppare il PoC ho utilizzato le seguenti tecnologie:

- Aries JS: *framework* JavaScript che permette l'emissioni di credenziali di tipo Hyperledger;
- Node.JS: ambiente in cui è possibile eseguire Aries JS;
- Express JS: *framework* di Node.JS, necessario alla libreria Aries JS;
- JavaScript: linguaggio per la scrittura degli *script* Express JS e linguaggio base di TypeScript;
- TypeScript: linguaggio in cui è scritto il *framework* Aries JS ed utilizzato nel prodotto;
- nodemon: *tool* che permette il riavvio automatico di un *server* Express JS;
- ts-node: *tool* che permette l'esecuzione dei file TypeScript, evitando l'invocazione del comando di compilazione;

- Postman: *software* per testare le chiamate HTTP, che esclude i problemi generati dal *testing* tramite *browser*;
- Visual Studio Code: *editor* con estensione per SSH;
- Macchina Virtuale di Athesys srl: a cui si accede in SSH tramite *tunnel* VPN;
- Indy CLI: interfaccia a linea di comando per interagire con il *ledger* Indy;
- Indicio TestNet: rete *open source* di *test* per la gestione e creazione di credenziali.

5.2 Progettazione

L'API è stata suddivisa in tre sottoservizi:

- "**Servizio *Server***": è una API che si occupa di interagire con "Servizio *Issuer*" e "Servizio *Holder*", a dimostrazione delle loro funzionalità;
- "**Servizio *Issuer***": è una API che contiene tutte le funzionalità necessarie ad un attore *Issuer*;
- "**Servizio *Holder***": è una API che contiene tutte le funzionalità necessarie ad un attore *Holder*;

Per integrare l'API nell'applicativo Monokee SSO, non sarà necessario utilizzare il "Servizio *Server*". "Servizio *Server*" infatti, costituisce una mera dimostrazione del funzionamento di "Servizio *Issuer*" e "Servizio *Holder*". I tre servizi colloquiano tra loro, come illustrato in figura 5.1. L'interazione principale avviene tra "Servizio *Issuer*" e "Servizio *Holder*".

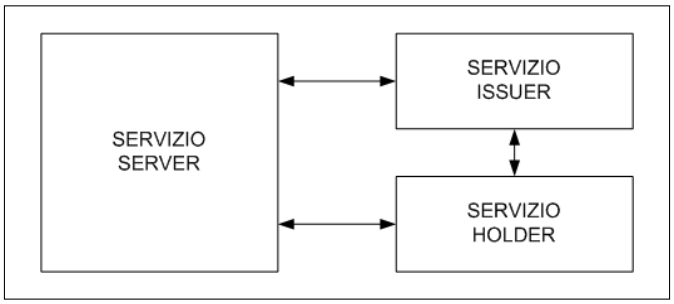

Figura 5.1: Colloquio tra i servizi di *Issuer*, *Holder* e *Server*.

5.3 Diagramma delle classi

Ogni servizio è una API, ciascuna costituita da una classe, come mostrato in figura 5.2. *Server* riceve le interazioni da parte dell'attore Sviluppatore e richiama le funzionalità di *Issuer* ed *Holder*, per dimostrarne il funzionamento. *Server* può invocare le singole attività, oppure invocare una sequenza di attività. Ad esempio, la creazione di un canale di comunicazione e l'emissione di credenziali coinvolgono più attività svolte da *Issuer* o *Holder*.

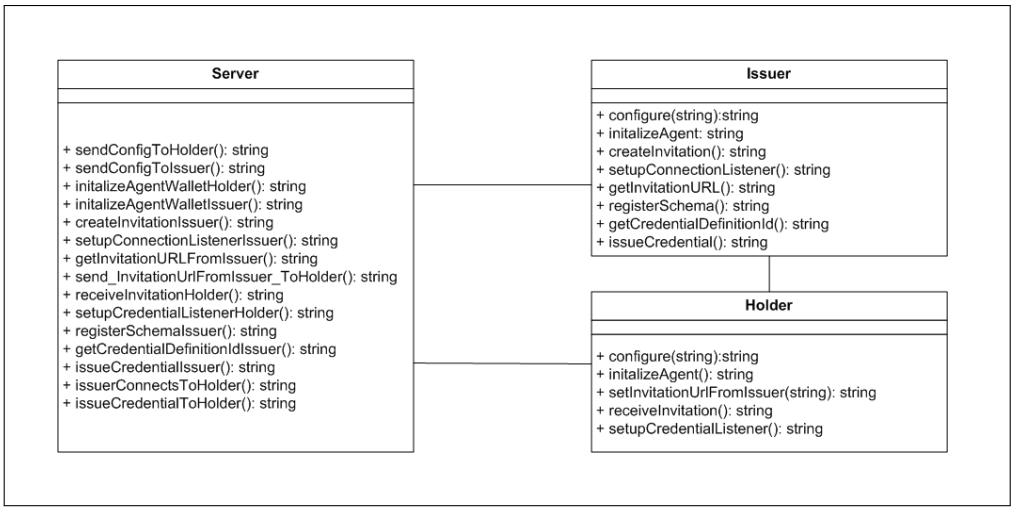

Figura 5.2: Diagramma delle classi.

5.4 Scrittura dati nella *chain*

L'API necessita di una *blockchain* per definire lo *scheme* da utilizzare per l'emissione delle credenziali. Allo scopo, si è utilizzato la Indicio TestNet, una *blockchain* di *test* predisposta per lo sviluppo di nuove reti.

5.5 Flusso di dati

Nell'API prodotta, esistono diversi flussi, mirati a far colloquiare le due entità *Issuer* ed *Holder* tramite un canale sicuro e criptato. Il Diagramma del Flusso dei Dati (DFD) schematizza i flussi di dati previsti dall'API e descritti dai *tutorial* di Aries JS.

Un DFD traccia il flusso di informazioni per qualsiasi processo o sistema: utilizza simboli definiti come rettangoli, cerchi e frecce, oltre a brevi etichette di testo, per mostrare *input* e *output* di dati, punti di memorizzazione e percorsi tra ciascuna destinazione.

5.5.1 Servizio *Server*

I flussi previsti dal "Servizio *Server*" pilotano "Servizio *Issuer*" e "Servizio *Holder*". "Servizio *Server*" sostituisce le entità che implementeranno gli altri due servizi. I flussi previsti dal "Servizio *Server*" sono illustrati in figura 5.3 e sono i seguenti:

- Configurare *wallet* ed *agent* di tipo *Issuer*: invia le configurazioni ad *Issuer*;
- Configurare *wallet* ed *agent* di tipo *Holder*: invia le configurazioni ad *Holder*;
- Inizializzare *wallet* ed *agent* di tipo *Issuer*: una volta ricevuti i settaggi, *wallet* ed *agent* possono essere inizializzati;
- Inizializzare *wallet* ed *agent* di tipo *Holder*: una volta ricevuti i settaggi, *wallet* ed *agent* possono essere inizializzati;
- Creare una *Invitation* da parte di *Issuer*: per creare un canale di comunicazione tra le due parti, una delle due deve preparare ed inviare una *Invitation*. Con questo flusso, *Issuer* prepara una *Invitation*;

- Creare risposta ad *Invitation* da parte di *Holder*: se *Holder* riceverà una *Invitation*, dovrà rispondere in modo adeguato;
- Avviare *Connection Listener* di *Issuer*: *Issuer* rimane in ascolto degli eventi relativi alle connessioni, grazie al proprio *Listener*;
- Inviare *Invitation URL* ad *Holder*: per poter iniziare una comunicazione, *Holder* deve ricevere un *Invitation URL*, generato da *Issuer*;
- Richiedere *Invitation URL* ad *Issuer*: per poter iniziare una comunicazione, *Issuer* deve generare un *Invitation URL*, che dovrà essere comunicato ad *Holder*;
- Impostare *Credential Listener* di *Holder*: *Holder* rimane in ascolto delle eventuali credenziali in entrata, grazie al proprio *Listener*;
- Registrare lo *scheme* di *Issuer*: *Issuer* deve creare uno *scheme* per poter emettere le credenziali;
- Ricevere il *Credential Definition ID* da *Issuer*: *Issuer* genera un *Credential Definition ID* per emettere le credenziali, che deve essere comunicato ad *Holder*, affinché possa ricevere delle credenziali;
- Emettere credenziali da *Issuer*: le credenziali vengono emesse ed inviate da *Issuer*;
- Connettere *Issuer* ed *Holder*: per poter creare un canale di comunicazione tra *Issuer* ed *Holder*, le due parti devono essere predisposte adeguatamente;
- Emettere credenziali da *Issuer* ad *Holder*: per poter emettere le credenziali, sono necessari un canale di comunicazione e settaggi adeguati ambo le parti.

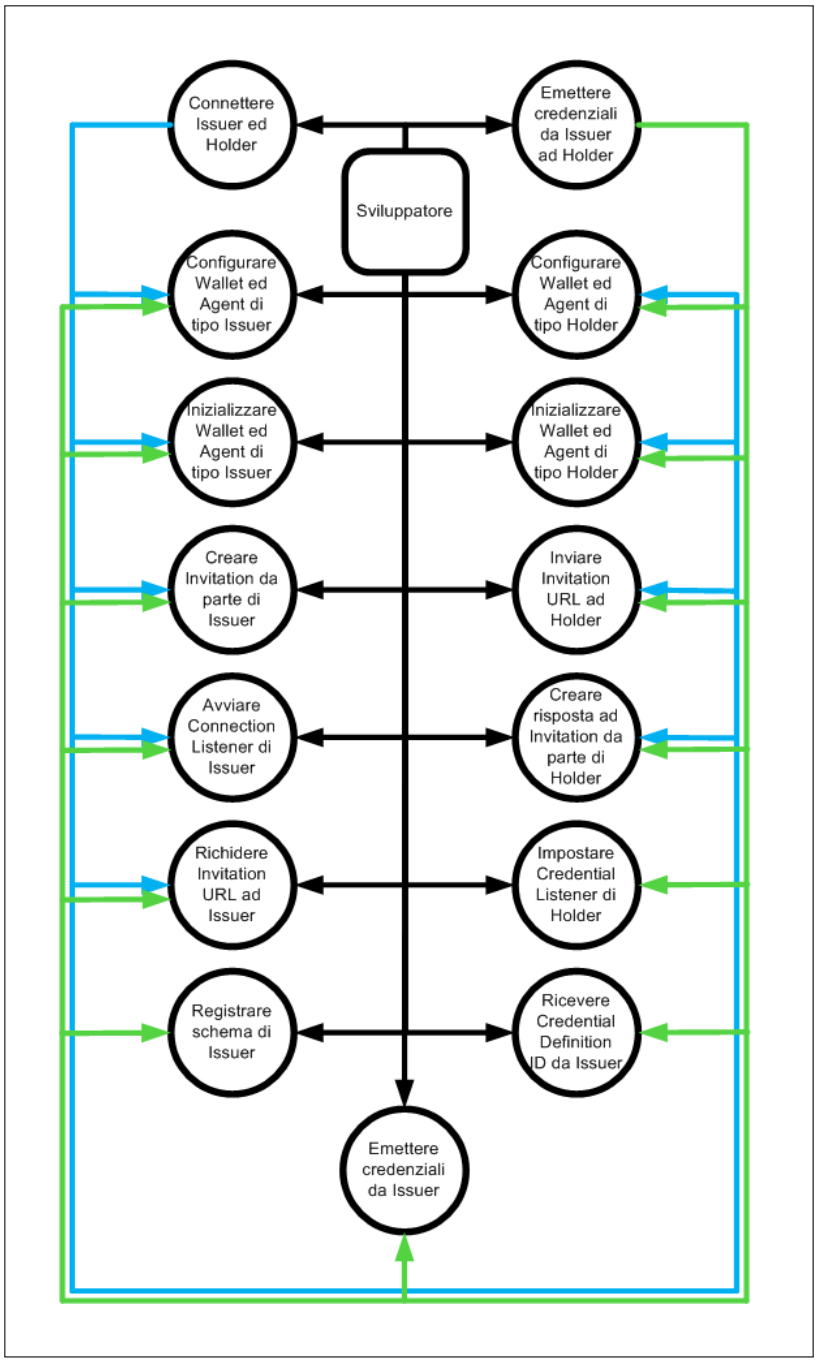

Figura 5.3: DFD del "Servizio *Server*".

5.5.2 Servizio *Issuer*

I flussi previsti per il "Servizio *Issuer*" comprendono tutte le attività che deve poter fare l'attore *Issuer*. I flussi previsti sono mostrati in figura 5.4 e sono i seguenti:

- Impostare la configurazione di *wallet* ed *agent*: *Issuer* riceve la configurazione dei suoi componenti e li configura;
- Inizializzare *wallet* ed *agent*: i due componenti *wallet* ed *agent* devono essere inizializzati, per poter essere utilizzati;
- Creare una *Invitation*: una *Invitation* è necessaria per iniziare il colloquio tra parti;
- Avviare il *Listener* di comunicazione: per reagire agli eventi, è necessario avviare un *Listener*;
- Richiedere *Invitation URL*: l'*Invitation URL* viene emesso da *Issuer* e deve essere fornito all'*Holder*. Nella fattispecie, deve poter essere comunicato ad un'entità terza, per poter pilotare il dialogo;
- Registrare lo *scheme*: lo *scheme* è necessario per emettere delle credenziali;
- Inviare il *Credential Definition ID*: il *Credential Definition ID* deve essere comunicato a chi riceverà le credenziali, prima di emetterle;
- Emettere una credenziale: l'emissione di una credenziale coinvolge anche la *chain*.

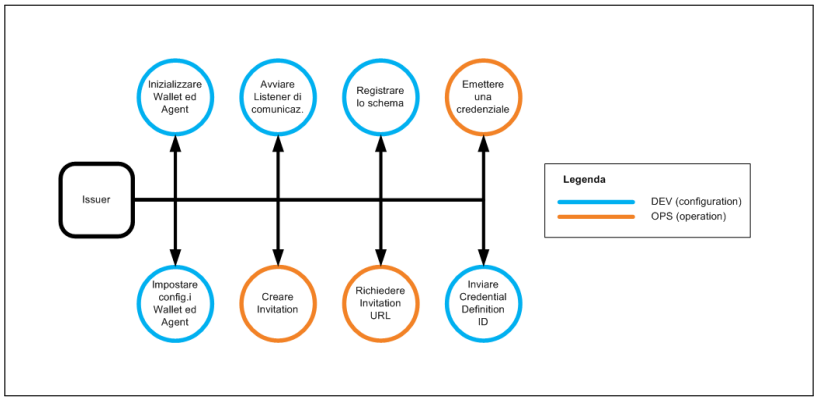

Figura 5.4: DFD del "Servizio *Issuer*".

5.5.3 Servizio *Holder*

I flussi previsti per il "Servizio *Holder*" comprendono tutte le attività che deve poter fare l'attore *Holder*. I flussi previsti sono schematizzati in figura 5.5 e sono i seguenti:

- Impostare la configurazione di *wallet* ed *agent*: *Holder* riceve la configurazione dei suoi componenti e li configura;
- Inizializzare *wallet* ed *agent*: i due componenti *wallet* ed *agent* devono essere inizializzati, per poter essere utilizzati;
- Creare una risposta ad una *Invitation*: se si riceve una *Invitation*, è necessario inviare una risposta adeguata;
- Avviare il *Credential Listener*: per reagire ad una credenziale ricevuta, è necessario avviare un *Listener*;
- Ricevere *Invitation URL* da *Issuer*: *Holder* deve conoscere l'*Invitation URL* generato da *Issuer*.

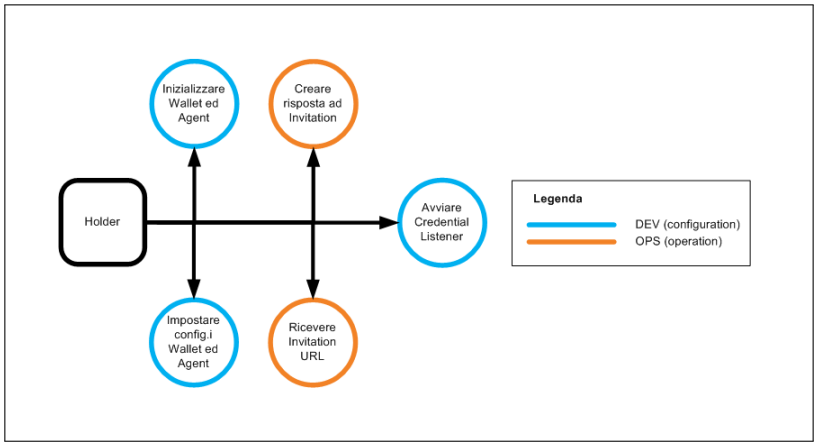

Figura 5.5: DFD del "Servizio *Holder*".

5.6 Dettagli tecnici

Le API devono essere raggiungibili tramite chiamata HTTP: pertanto, ciascuna è presente in un *server* Node.JS dedicato. Ciascuna chiamata ritorna l'esito, in formato JSON.

5.7 Verifica e validazione

Il *testing* è stato eseguito costantemente, al fine di individuare errori.

Durante l'approfondimento delle tematiche collegate ad SSI, gli incontri formativi accertavano che i concetti principali fossero stati appresi ed erano spunto per discussioni su alcuni particolari.

Durante l'analisi e la progettazione del PoC, è stata costante l'interazione con Athesys srl, in modo da raccogliere i requisiti ed essere guidati nella definizione della specifica tecnica.

Lo sviluppo del PoC è stato suddiviso in tre servizi, in forma di API. Ciascun servizio è costituito da funzioni *middleware* di Express JS. Le funzioni attivano gli *hook* offerti da Aries JS e non aggiungono altre funzionalità. Pertanto, il PoC stesso può essere inteso come un *test* del *framework* Aries JS.

Il PoC è stato testato utilizzato diversi approcci:

- Invio di chiamate HTTP all'API, grazie al *software* Postman;
- Verifiche dei dati scritti nella *chain*;
- *Test* di accettazione da parte del *team* di sviluppo.

Dall'approfondimento delle tematiche è nata della documentazione integrativa della *Knowledge Base*, mentre dallo sviluppo si è ottenuto un PoC che dimostra le potenzialità e la netta separazione tra gli attori. Entrambi i prodotti sono stati validati, controllando che rispettassero le richieste iniziali. Infine, sono stati presentati ed approvati da Athesys srl.

Capitolo 6

Discussione

Si discute quanto appreso durante il tirocinio.

6.1 SSI

L'identità decentralizzata ha le potenzialità per rivoluzionare la gestione dei dati personali, perché restituisce il controllo delle informazioni al loro *owner*, garantendo elevati livelli di sicurezza. In questo modo, soddisfa due bisogni degli utenti utilizzatori: il controllo completo delle proprie informazioni e la garanzia di una struttura affidabile. SSI elimina quindi la diffusione impropria di informazioni personali ed rimuove attori superflui nella relazione tra utente e servizio. Pertanto, può impattare nella società apportando sensibili miglioramenti nella gestione delle identità.

I fattori che possono ostacolare l'adozione di SSI sono principalmente di carattere strutturale: infatti, devono essere previsti delle DLT pubbliche e *wallet* idonei. Le grandi *software house* non dovranno tentare di privatizzare le *blockchain*, ma dovranno essere promotori dell'identità decentralizzata.

6.2 Aries JS

Aries JS è un *framework* leggero ed il linguaggio diffuso non ostacola il suo utilizzo. Per comprenderne l'utilizzo, sono comunque necessarie adeguate conoscenze in ambito SSI, poiché la documentazione si limita alle spiegazioni relative al *framework*.

Attualmente, Aries JS risulta essere una delle librerie più recenti presenti nel mercato. Altri *framework* sono poco mantenuti e sono caratterizzati da una ripida curva di apprendimento.

Si nota che il *framework* è stato progettato per essere il più semplice possibile e con particolare attenzione alla destinazione delle funzionalità. Ciò permette di separare nettamente le responsabilità tra applicativi destinati ad attori diversi. La configurazione degli *agent* è semplice e sono state previste modalità DEV ed OPS. La *community* è attiva e sono frequenti i rilasci di nuove funzionalità. Purtroppo, la documentazione risulta essere ridotta all'essenziale ed i casi d'uso sono limitati, risultando spesso incompleta.

Inoltre, Aries JS e le altre librerie non sono ancora allineati alle tecnologie centralizzate e viceversa, anche se si notano degli avanzamenti nell'integrazione. Per esempio, SAML non prevede integrazioni con tecnologie decentralizzate, mentre *OpenID Foundation* sta sviluppando nuovi protocolli dedicati all'identità decentralizzata (OIDC4VP e OIDC4VCI), che ad oggi sono in via di sviluppo.

Per questi motivi, Aries JS è una valida libreria, ma sembra essere prematura per essere adottata in ambito *enterprise*.

6.3 Sviluppo del PoC

Il PoC è stato necessario per dimostrare che Aries JS potesse separare nettamente le funzionalità dedicate ad *Issuer* da quelle destinate ad *Holder*. I due attori, infatti, agiscono in SSI con applicazioni e *device* diversi. La suddivisione netta delle funzionalità tra attori risulta quindi requisito fondamentale per la valutazione del *software*.

Il PoC non implementa tutte le funzionalità di un applicativo completo. Ad esempio, non sono stati affrontati alcuni temi, come il ritiro di una credenziale e l'integrazione

con un *wallet*. Sarebbe molto interessante sviluppare prototipi che affrontino queste tematiche.

Lo sviluppo del PoC è stato occasione per conoscere Express JS. In questa fase, il problema principale non è stato lo studio dei comandi di Aries JS, ma bensì il colloquio tra servizi. Le scarse conoscenze iniziali hanno reso difficili i primi colloqui tra le parti, ma approfondimenti mirati hanno sanato le lacune. Ora, apprezzo Express JS e la sua flessibilità per il *routing* delle richieste.

Inoltre, ho potuto applicare le mie conoscenze nella programmazione asincrona e riconosco che è fondamentale per gli applicativi con interazione utente.

Infine, il PoC mi ha permesso di conoscere TypeScript. Avendo conoscenze basilari di JavaScript, non ho trovato particolari difficoltà ad utilizzare il nuovo linguaggio.

6.4 Sviluppi futuri

Secondo la mia esperienza, SSI può rivoluzionare la quotidiana gestione delle credenziali ed il modo in cui un utente fornisce le proprie informazioni. Il triangolo della fiducia può creare il *layer* di fiducia tra le parti sopra la comunicazione in rete, abilitando SSI all'utilizzo in settori dove la *privacy* è requisito fondamentale: ad esempio, nell'ambito bancario e sanitario. Secondo me, SSI potrebbe diventare un nuovo *standard* per la gestione dei dati.

Anche per Aries JS si prospetta un'ampia diffusione e la scelta del linguaggio JavaScript lo rende facilmente utilizzabile. La documentazione di Aries JS diventerebbe più completa se includesse più *use case*, ad esempio ritiro di una credenziale emessa o la cancellazione di un *Issuer*. Infatti, le credenziali possono essere emesse potenzialmente da qualsiasi ente, ad esempio un'azienda privata. L'ente emittente svolge il ruolo di *Issuer* e può accadere che termini la sua attività, con il conseguente ritiro dei documenti emessi. Questi casi d'uso sono rilevanti poiché riflettono le attività necessarie per una gestione completa dei documenti (digitali o analogici che siano).

Secondo quanto appreso, posso affermare che l'identità decentralizzata è un'evoluzione necessaria nella gestione delle identità digitali. Non è più accettabile la presenza di un ente federato che abiliti la facoltà di accedere a servizi, accentrando le informazioni

ed utilizzandole per lucrare. Attualmente, uno dei vantaggi dell'identità federata è la comodità di accedere a molteplici servizi con un solo *account*, a patto di condividere l'attività dell'utente con l'ente federato. Per l'utente medio, la comodità è un aspetto che solitamente vince su altri fattori, quali la violazione della *privacy*. Diventa necessario ristabilire le priorità in termini di sicurezza, con un'adeguata cultura digitale. Le nuove generazioni di utenti sono in grado di comprendere i rischi e le strutture che compongono modelli come SSI e potenzialmente saranno i primi utilizzatori delle "credenziali verificabili".

La diffusione dell'identità decentralizzata può essere frenata dall'associazione della *blockchain* alle criptovalute, perché spesso vengono ritenute inaffidabili ed incontrollate.

Le tre figure *Issuer*, *Holder* e *Verifier* corrispondo ai ruoli presenti nelle comuni attività "analogiche" di rilascio e verifica dei documenti. In SSI, ciascuno può verificare le attività degli altri grazie alla firma pubblica. La verifica però è un processo complesso che incrocia le firme digitali e può risultare complesso.

Ritengo che l'identità decentralizzata sarà utilizzata nella quotidianità, magari grazie all'adozione da parte delle istituzioni o di grandi aziende. Secondo me, l'identità federata è ormai superata, perché gli utenti sono sensibili nei confronti dei loro dati e non è più necessario un ente accentratore, che li tutela da utilizzi inappropriati. Il modello SSI è ben definito, ma non si è ancora raggiunto uno *standard* di mercato da adottare nella società. Allo stato attuale, ritengo siano necessarie ancora attività di ricerca nel settore, per raggiungere l'evoluzione della gestione delle identità. Ad esempio, si potrebbe approfondire come strutturare il dato nelle *blockchain* e standardizzare i metodi di rilascio e ritiro di una credenziale.

Capitolo 7

Conclusioni

Si traggono le conclusioni dell'esperienza di tirocinio.

7.1 Identificazione del contesto

L'identità decentralizzata è l'evoluzione dell'identità digitale. L'innovamento nel settore è richiesto da utenti finali ed amministratori, che auspicano una buona usabilità dei prodotti ed elevati livelli di sicurezza. SSI restituisce il controllo dei dati all'utente *owner* ed innova il settore dell'IAM. In un mondo che vuole essere sempre più digitalizzato, ritengo che i tempi siano maturi affinché l'identità decentralizzata sia considerata come nuovo modello di gestione delle identità. Gli *account* ed i dati associati non sono solamente un tecnicismo necessario per accedere ad un servizio, ma identificano la persona all'interno di un contesto digitale. In quanto indentificativo, l'*account* deve essere tutelato come i dati personali.

L'intento di Monokee SSO è di soddisfare entrambe le necessità e di ampliare la gamma di servizi offerti. Credo che l'integrazione con l'identità decentralizzata permetterebbe all'applicativo di essere in linea con le esigenze del mercato, perché gli utenti richiedono la tutela dei loro dati e l'identità decentralizzata offre la soluzione cercata.

Aries JS è una valida soluzione per l'implementazione ed il tirocinio ne ha dimostrato

la fattibilità.

7.2 Raggiungimento degli obiettivi

Per poter comprende a fondo il settore delle identità digitali, è necessaria un'introduzione in IAM ed al SSO. Successivamente, si è approfondito Hyperledger ed i suoi progetti, con particolare attenzione ad Hyperledger Indy. Dopo uno studio approfondito di Aries JS, è stato ideato e sviluppato un PoC, a dimostrazione della fattibilità di integrazione con Monokee SSO.

Gli obiettivi di tirocinio sono stati soddisfatti in parte:

- Obbligatori:
  - **ob01**: implementazione attraverso la piattaforma aziendale Monokee SSO dello scambio di credenziali verificabili utilizzando un *agent* Hyperledger Aries.
    L'obiettivo è stato raggiunto poiché è possibile emettere le credenziali utilizzando Aries JS.
  - **ob02**: integrazione attraverso la piattaforma aziendale Monokee SSO di due servizi esterni utilizzando Monokee SSO come *Identity Provider* e sia SAML che OIDC come protocolli SSO.
    L'obiettivo è stato raggiunto settando la configurazione necessaria in Monokee SSO.
- Desiderabili:
  - **de01**: integrazione PoC accesso SP via SAML / OIDC utilizzando identità effimere ottenute da una specifica VC.
    L'obiettivo è stato raggiunto con la creazione di un PoC specifico.
  - **de02**: integrazione PoC accesso SP via SAML / OIDC utilizzando identità effimere ottenute da una VP generica contenente molteplici VC.
    L'obiettivo non è stato raggiunto, poiché Aries JS non è ancora allineato alle tecnologie centralizzate, il che richiede ulteriori studi e approfondimenti. Per esempio, SAML non prevede integrazioni con tecnologie decentralizzate,

quindi andrebbe studiato un convertitore da VC a SAML *response*, lasciando aperto il problema della creazione di una SAML *request*. Nel caso di OIDC, la *OpenID Foundation* sta lavorando su OIDC4VP e OIDC4VCI, ma ad oggi si trovano ancora in uno stato preliminare.

  - **de03**: *mapping* ruolo utente SSI su *claim* SSO via anoncreds.
    L'obiettivo non è stato raggiunto poiché avrebbe richiesto un approfondimento dedicato. Per un utilizzo confortevole di SSI, è necessario prevedere che gli accessi possano avvenire via SSO. Anoncreds potrebbe gestire gli accessi in SSO, ma la tecnologia non è ancora abbastanza stabile per definire un *mapping* del ruolo utente. Uno dei principali ostacoli è che il substrato tecnologico non è ancora *plug and play* ed avrebbe richiesto studi specifici. Pertanto, ho preferito dedicarmi agli obiettivi obbligatori.

- Opzionali:

  - **op01**: sviluppo documentazione architetturale, flussi dati e flussi di controllo per gli strumenti utilizzati.
    L'obiettivo è stato raggiunto ed il PoC è accompagnato da documentazione.

La maggior parte degli obiettivi sono stati raggiunti con successo. Alcuni obiettivi non sono stati soddisfatti principalmente perché le tecnologie non sono sufficientemente stabili da garantire integrazioni agili. Molte risorse sono state dedicate allo studio del *background* di SSI, allo scopo di creare una solida conoscenza per gli sviluppi.

7.3 Conoscenze acquisite

Il tirocinio ha ampliato le mie conoscenze in ambito di identità: in particolare, ho compreso che il settore dell'identità decentralizzata è attivo e ricco di spunti di sviluppo. Gli approfondimenti hanno permesso una comprensione maggiore dell'argomento e l'utilità che può avere nella società. SSO ed i protocolli che ne permettono l'attuazione erano poco conosciuti ed il tirocinio ha permesso di comprendere l'utilità e gli aspetti fondamentali.

Il tirocinio è stato occasione per consolidare le conoscenze, come Express JS e TypeScript.

Inoltre, lavorare per obiettivi è stato molto soddisfacente e costruttivo, poiché permette un'organizzazione migliore e rende tangibile il prodotto durante lo sviluppo.

7.4 Valutazione personale

L'esperienza è stata un'ottima opportunità per ampliare le mie conoscenze ed ora sono più sensibile al tema della sicurezza dei dati personali. Le conoscenze teoriche apprese hanno arricchito il mio bagaglio culturale ed hanno generato notevole interesse nel settore. Le tecnologie a cui sono stata introdotta hanno numerose potenzialità e continuerò ad utilizzarle in autonomia. Avrei apprezzato avere una formazione più dettagliata prima del tirocinio in ambito JavaScript e *blockchain*, ma sono riuscita a colmare le lacune.

Athesys srl valorizza l'aspetto umano di ogni collaboratore: pone particolare attenzione alle esigenze personali e ritiene che sia il *team* a dare valore all'azienda stessa. Essendo una realtà giovane, dinamica ed innovativa, il clima è accogliente e le nuove idee sono sempre ben accette. Lo sviluppo personale e professionale viene promosso nella varietà degli ambiti di azione. Ho potuto apprezzare personalmente come le mie idee fossero ben accettate e diventassero spunto per colloqui di approfondimento.

Tutte le attività sono state svolte in *smartworking* e ne ho apprezzato la flessilità. Il timore principale era che l'isolamento fisico dal *team* potesse influire negativamente sulle attività, ma ciò non è accaduto: il *tutor* e lo sviluppatore di Athesys srl sono stati sempre disponibili a guidarmi e sostenermi.

Alla luce di ciò, sono molto soddisfatta del tirocinio e non mi pento della scelta fatta.

Acronimi e abbreviazioni

API Application Program Interface. iii, 3, 4, 10, 22, 41, 42, 55–58, 60–62, 66, 67, 78, 83

DFD Diagramma del Flusso dei Dati. ix, 62, 64–66, 79

DID Decentralized Identifier. 30–32, 38, 79, 82–85

DLT Distributed Ledger Technology. 2, 3, 10, 24–26, 34, 35, 68, 79, 81

HQ headquarters. 11, 80

HTTP *HyperText Transfer Protocol*. 41, 58, 66, 67, 80

IAM Identity and Access Management. iii, viii, 2–4, 6–8, 15–19, 41, 72, 73, 78, 81, 83, 84

JIT Just in Time. 2, 15–17, 82

JS JavaScript. 82

JSON JavaScript Object Notation. 22, 30, 79, 82, 83

MFA Multi-factor authentication. 15, 82

OAuth Open Authorization. viii, 2, 4, 19, 21, 22, 83

OIDC OpenId Connect. viii, 2, 4, 8–10, 19, 22, 23, 73, 74, 83

P2P peer-to-peer. 36, 83

PoC Proof of Concept. iii, 3, 4, 6, 9, 11, 12, 34, 40, 41, 59, 67, 69, 70, 73, 74, 83

REST Representational state transfer. 22, 41, 58, 83

RFC6749 Request for Comments: 6749. 22, 83

SAML Security Assertion Markup Language. viii, 2, 4, 8–10, 19, 20, 69, 73, 74, 84

SPID Sistema Pubblico di Identità Digitale. 14, 84

SSH Secure Shell. 58, 60, 85

SSI Self-Sovereign Identity. iii, 2–4, 7–11, 14, 24, 26–32, 34, 35, 37, 41, 42, 67–72, 74, 78, 80, 81, 84, 85

SSO Single Sign-On. iii, viii, 2–4, 7–10, 15, 18, 19, 22, 73, 74, 83, 84

UML Unified Modeling Language. 42, 79, 84

VC Verificable Credential. 9, 29–32, 73, 74, 85

VP Verificable Presentation. 9, 30–32, 73, 85

VPN *Virtual Private Network*. 60, 85

W3C World Wide Web Consortium. 27, 31, 85

XML eXtensible Markup Language. 20, 80

Glossario

agent in informatica, è un'applicazione che si connette a un processo, solitamente in esecuzione su un'altro *host*. 8, 10, 37, 38, 45, 46, 50, 52, 55, 56, 62, 65, 66, 69, 73, 79

API in informatica, si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un *set* di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'*hardware* e il programmatore o tra *software* a basso e quello ad alto livello, semplificando così il lavoro di programmazione. 58, 76

Aries JS Aries JS (denominato anche "Hyperledger Aries Framework JavaScript") è un *framework* JavaScript che permette l'emissione e lo scambio di credenziali SSI. Aries JS utilizza il *framework* Hyperledger Aries. Hyperledger Aries Framework JavaScript è distribuito tramite *Apache License Version* 2.0 (Apache-2.0). iii, 3, 4, 7, 8, 11, 12, 36, 37, 41, 58, 59, 62, 67, 69, 70, 72, 73, 78, 79

Athesys srl azienda ospitante dello *stage* ed è un *system integrator* impegnato principalmente nello studio e sviluppo soluzioni IAM. iii, viii, 3, 4, 6, 7, 11, 12, 41, 57, 58, 60, 67, 75, 82

blockchain il termine inglese *blockchain* può essere tradotto in italiano come "libro mastro" o "registro". La *blockchain* nasce per essere un archivio, che contiene i dati relativi a risorse ed alle relative transazioni ed un registro decentralizzato e distribuito. iii, ix, 2, 3, 25–27, 30, 33–36, 40, 62, 68, 71, 75, 78–81

credenziali coppia di "nome utente" e "*password*", utilizzate per identificarsi presso un sistema. 84

crittografia branca della "Crittologia" che tratta i metodi per rendere un messaggio non comprensibile/intelligibile a persone non autorizzate a leggerlo, garantendo così, in chiave moderna, il requisito di confidenzialità o riservatezza tipico della sicurezza informatica. Un tale messaggio si chiama comunemente "crittogramma" e i metodi usati sono detti "tecniche di cifratura". 3, 22, 24, 36, 81

DID il DID (*Decentralized Identifier*) è un codice alfanumerico basato su un sistema a doppia chiave crittografica, memorizzato su *blockchain*, che consente di identificare univocamente un'entità *online*. 76

deprovisioning in informatica, contrario di *provisioning*: processo di distruzione e configurazione di un'infrastruttura IT, che comprende le procedure necessarie per limitare o annullare l'accesso di utenti e sistemi a varie risorse. 15

DFD sta per "Diagramma di Flusso dei Dati" ed è un diagramma che traccia il flusso di informazioni per qualsiasi processo o sistema. Utilizza simboli definiti come rettangoli, cerchi e frecce, oltre a brevi etichette di testo, per mostrare *input* e *output* di dati, punti di memorizzazione e percorsi tra ciascuna destinazione. 76

Diagramma dei Casi d'Uso in inglese "*Use Case Diagram*", diagramma dedicato alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. 42

Diagramma delle Attivita un tipo di diagramma UML che permette di descrivere un processo attraverso dei grafi, in cui i nodi rappresentano le attività e gli archi l'ordine con cui vengono eseguite. viii, 22, 23

DID Document *file* JSON, che costituisce un elemento aggiuntivo, contenente ulteriori informazioni sull'entità identificata. Dato il DID, è possibile ricavare il DID Document. 30–32, 38, 79, 82, 83, 85

DIDComm protocollo di comunicazione. Utilizzato in Aries JS come protocollo di comunicazione tra *agent*. 37, 38

DLT acronimo di *Distributed Ledger Technology* (tradotto in "tecnologie di registro distribuito"): sistema digitale per la registrazione di "transazione di attività", in cui le transazioni e i relativi dettagli sono registrati in più luoghi contemporaneamente. 76

Evernym *leader* mondiale nello sviluppo di SSI, ha realizzato la rete Sovrin ed è uno dei maggiori fondatori di Hyperledger Indy. 35

Express JS *framework* web per Node.JS. iii, 58, 59, 67, 70, 74, 83

XML metalinguaggio per la definizione di linguaggi di *markup*, ovvero un linguaggio basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo. 77

Facebook impresa statunitense che controlla i servizi di rete sociale "Facebook" e "Instagram", i servizi di messaggistica istantanea "WhatsApp" e "Messenger" e sviluppa i visori di realtà virtuale "Oculus Rift". Dal 2021 è conosciuta come "Meta Platforms". 14

framework in informatica, si intende un'architettura logica a supporto di altre architetture. Il *framework* viene utilizzato per lo sviluppo *software*, allo scopo di facilitarne l'implementazione. iii, 3, 4, 8, 9, 12, 34–37, 41, 58, 59, 67, 69, 78, 80, 81

Google azienda informatica statunitense che offre servizi *online*. Tra la grande quantità di prodotti o servizi offerti, troviamo il motore di ricerca "Google", il sistema operativo "Android", il sistema operativo "Chrome OS" e servizi *web* quali "YouTube", "Gmail", "Play Store", "Google Maps" e molti altri. 14

HQ sede centrale. 76

Holder In SSI, è l'entità a cui è intestata un documento. Può essere chiamato anche "*Subject*". ix, 3, 28, 29, 31, 32, 37–39, 41, 42, 45–49, 52, 53, 55, 56, 58, 60–63, 65, 66, 69, 71, 84, 85

HTTP protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni in un'architettura *client-server*. 76

Hyperledger Hyperledger è un progetto, a cui partecipano più di 200 aziende. Il progetto ha lo scopo di promuovere la tecnologia *blockchain* in tutti i settori, sviluppando *software* *open source*. iii, 3, 4, 9, 11, 26, 34–37, 41, 59, 73, 81

Hyperledger Aries è un *software* che fornisce un *kit* di strumenti condiviso, riutilizzabile e interoperabile progettato per iniziative e soluzioni incentrate sulla creazione, la trasmissione e l'archiviazione di credenziali digitali verificabili. ix, 3, 8, 9, 35–37, 40–42, 73, 78, 81

Hyperledger Indy *framework* ideato appositamente per gestire l'identità decentralizzata, basato su *blockchain*. ix, 7, 9, 35, 40, 73, 80, 81

Hyperledger Ursa libreria crittografica da utilizzare nel progetto Hyperledger, per implementare crittografia solida e controllata. ix, 35–37

IAM in informatica, si intendono i sistemi integrati di tecnologie, criteri e procedure in grado di consentire alle organizzazioni di facilitare e controllare gli accessi degli utenti ad applicazioni e dati critici, proteggendo contestualmente i dati personali da accessi non autorizzati. 76

identità complesso dei dati personali, caratteristici e fondamentali che consentono l'individuazione di un soggetto o ne garantiscono l'autenticità. 1, 13–15, 28, 74, 81

identità decentralizzata in informatica, per identità decentralizzata si ha quando ciò che definisce un'identità è decentralizzato. In un sistema ad identità centralizzata, le identità sono raccolte in un unico punto. Con l'identità decentralizzata, le identità sono presso i dispositivi dei proprietari dell'identità stessa: in questo modo, non esiste un punto accentratore delle identità. iii, 1–3, 8, 11, 35, 37, 40, 68–72, 74, 81

Indicio società che sviluppa soluzioni di identità digitale basate su *blockchain* e DLT affidabili. ix, 40, 81

Indicio TestNet *network* sviluppata da Indicio utilizzabile per il *testing* di nuove soluzioni. Utilizza Hyperledger Indy ed Hyperledger Aries. iii, 40, 58, 60, 62

Indy CLI interfaccia a linea di comando per interagire con il *ledger* Indy. 60

Invitation in SSI, è la richiesta inviata per iniziare una comunicazione. 62, 63, 65, 66

Issuer In SSI, è l'entità (persona fisica o giuridica) che emette un documento. ix, 3, 28–32, 37–39, 41, 42, 45–52, 55, 56, 58, 60–63, 65, 66, 69–71, 84

JS in informatica, linguaggio di programmazione multi paradigma, orientato agli eventi. 37, 42, 58, 59, 70, 75, 76, 78, 82–84

JSON semplice formato per lo scambio di dati. 66, 76

JIT in informatica, si intende l'insieme delle regole che abilitano gli utenti ad accedere agli *account* e agli *asset* privilegiati esclusivamente quando tale accesso è necessario. 76

Linux Foundation organizzazione *no-profit* che sostiene lo sviluppo del *kernel* Linux, elaborando anche un insieme di *standard* al fine di uniformare le caratteristiche dei vari sistemi operativi Linux. Fornisce un *hub* affidabile e neutrale per sviluppatori e organizzazioni per codificare, gestire e ridimensionare progetti ed ecosistemi di tecnologia *open source*. 34

method In un DID, indica il risolutore da usare per ottenere il DID Document. 30, 32

metodologia Agile nell'ingegneria del *software*, si indica un insieme di metodi di sviluppo del *software* fondati su principi comuni, direttamente o indirettamente derivati dai principi del "Manifesto per lo sviluppo agile del *software*". 6, 10, 12

Monokee srl *startup* che ha sviluppato il *software* Monokee SSO e collabora con Athesys srl. viii, 7, 82

Monokee SSO *software* sviluppato dalla *startup* Monokee srl in collaboraborazione con Athesys srl. Il prodotto finale del tirocinio sarà intergrato con l'applicativo Monokee SSO. iii, 3, 7, 8, 11, 12, 41, 60, 72, 73, 82

MFA acronimo di *Multi-factor authentication* (tradotto in "autenticazione a più fattori"): è un metodo di autenticazione elettronica in cui a un utente viene concesso l'accesso a un sito *web* o a un'applicazione solo dopo aver presentato con successo due o più elementi di prova (o fattori) ad un meccanismo di autenticazione. 76

Node.JS ambiente *open source*, *cross platform* ambiente *runtime* in JavaScript. iii, 37, 59, 66, 80

nodemon *tool* che permette il riavvio automatico di un *server* Express JS. 59

OAuth protocollo *standard* nel settore della IAM applicabile in ambito SSO. 76

open source in informatica, si indica un *software* distribuito sotto i termini di una licenza *open source*, che ne concede lo studio, l'utilizzo, la modifica e la redistribuzione. 28, 34, 37, 40, 60, 80, 82, 83

OIDC protocollo per l'autenticazione federata, che utilizza API REST e *token* di autenticazione JSON per consentire l'accesso in SSO. 76

payload detto anche "*method-specific Identifier*". In un DID, indica in modo specifico il DID Document. 30, 32

P2P in informatica, *peer-to-peer* (P2P) indica un modello di architettura logica di rete, in cui i nodi non sono gerarchizzati unicamente sotto forma di client o server fissi ("clienti" e "serventi"), ma anche sotto forma di nodi equivalenti o "paritari" (*peer*), potendo fungere al contempo da *client* e *server* verso gli altri nodi terminali (*host*) della rete. 76

Postman *software* che consente agli sviluppatori di progettare, costruire, testare e iterare le proprie API. 58, 60, 67

PoC *Proof of Concept* è traducibile in "prova di concetto" o "dimostrazione di fattibilità" e si intende la realizzazione incompleta o abbozzata di un determinato progetto o metodo, allo scopo di provarne la fattibilità o dimostrare la fondatezza di alcuni principi o concetti costituenti. Un esempio tipico è il prototipo. 77

provisioning in informatica, processo di creazione e configurazione di un'infrastruttura IT, che comprende le procedure necessarie per gestire l'accesso di utenti e sistemi a varie risorse. 15, 79

React Native libreria JavaScript per creare interfacce utente. 37

REST stile architetturale per sistemi distribuiti. 77

RFC6749 RCF è documento pubblicato dalla *Internet Engineering Task Force*, che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie dell'ambito informatico o, più nello specifico, di Internet. RFC6749 e la richiesta di commenti numero 6749, riguardante il protocollo OAuth. 77

scheme detto anche "*schema*". In un DID, è una definizione leggibile dalla macchina, della semantica di una struttura di dati; è utilizzato per definire gli attributi utilizzati in una o più *Credential Definition*. 30, 32, 39, 48, 51, 56, 62, 63, 65

SAML protocollo di federazione nell'ambito della IAM che permette l'autenticazione in SSO. 77

SSO acronimo di *Single Sign-On*, è la proprietà di un sistema di controllo d'accesso che consente ad un utente di effettuare un'unica autenticazione valida per più sistemi *software* o risorse informatiche alle quali è abilitato. 77

SPID credenziali che rappresenta l'identità digitale e personale di ogni cittadino italiano, con cui è riconosciuto dalla Pubblica Amministrazione per utilizzare in maniera personalizzata e sicura i servizi digitali. 77

Sovrin organizzazione *no profit* che amministra la "*Sovrin Network*", che promuove l'uso di SSI a livello globale ed in modo accessibile. 35, 80

SSI dall'inglese "identità auto-sovrana", si indica un modello per la gestione delle identità digitali, in cui individui o aziende hanno la proprietà esclusiva di controllare i propri *account* ed i dati personali. Gli individui con SSI possono archiviare i propri dati sui propri dispositivi e fornirli per la verifica e le transazioni senza la necessità di fare affidamento su un archivio centrale di dati. Con l'identità auto-sovrana, gli utenti hanno il controllo completo su come le loro informazioni personali vengono conservate e utilizzate. 77

triangolo della fiducia nel modello SSI, rapporto di fiducia che si instaura tra *Issuer*, *Verifier* e *Holder*. ix, 3, 28, 29, 32, 39, 70

ts-node *tool* che permette l'esecuzione dei *file* TypeScript, senza invocare il comando di compilazione. 59

TypeScript linguaggio di programmazione sviluppato da Microsoft. Si tratta di un'estensione di JavaScript, che rende tipizzati gli oggetti. iii, 59, 70, 74, 84

UML in Ingegneria del *Software*, è un linguaggio di modellazione e specifica basato sul paradigma *object-oriented*. L'UML svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti.

Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 77

Universal Resolver "risolutore universale" che risolve i DID per ottenere i *DID Document*. 32

VC un qualsiasi tipo di attributo collegato a un'entità. Alcuni corrispettivi fisici di una VC sono per esempio la patente di guida o il diploma di laurea. Nel modello SSI, però, le VC sono digitali, immodificabili e verificabili. 77

VP documento predisposto per essere verificato, che contiene la *Verificable Credential* (VC) ed altri dati. 77

Verifier è l'individuo, l'organizzazione, la società, il governo o altro ente a cui l'*Holder* deve dimostrare la legittimità e l'attendibilità delle informazioni (ad esempio, una banca con cui si desidera aprire un conto bancario). ix, 3, 28, 29, 31, 32, 71, 84

Visual Studio Code *editor* di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. 60

VPN VPN sta per *Virtual Private Network* ed è una rete privata, instaurata come connessione tra soggetti che utilizzano, come tecnologia di trasporto, un protocollo di trasmissione pubblico e condiviso. 77

SSH In informatica e telecomunicazioni, SSH (*Secure SHell*, *shell* sicura) è un protocollo che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica. 77

wallet In SSI, è il raccoglitore dei documenti e delle identità, necessario per emettere o ricevere credenziali. 3, 29, 37–39, 45, 46, 50, 52, 55, 56, 62, 65, 66, 68, 70

W3C organizzazione non governativa internazionale che ha come scopo quello di favorire lo sviluppo di tutte le potenzialità del *World Wide Web* e diffondere la cultura dell'accessibilità della Rete. 77

Zero Trust in informatica, modello di sicurezza che garantisce che dati e risorse siano inaccessibili per impostazione predefinita. 2, 15, 16, 19, 40

Bibliografia

Siti web consultati

AGID – Agenzia per l'Italia Digitale, 2023, SPID, ultima visita in luglio 2023. URL: https://www.spid.gov.it/.

Amazon Web Services, Inc. o società affiliate, 2023, Cos'è il SSO?, ultima visita in luglio 2023. URL: https://aws.amazon.com/it/what-is/sso/.

Aries JS, 2023, Aries Framework JavaScript Github repository, ultima visita in luglio 2023. URL: https://github.com/hyperledger/aries-framework-javascript.

Aries JS, 2023, Aries JavaScript Documentation "concepts", ultima visita in luglio 2023. URL: https://aries.js.org/guides/0.4/concepts.

Aries JS, 2023, Aries JavaScript Documentation, ultima visita in luglio 2023. URL: https://aries.js.org/.

Aries JS, 2023, Tutorials, ultima visita in luglio 2023. URL: https://aries.js.org/guides/0.4/tutorials (cit. a p. 41).

Athesys S.r.l, 2023, Athesys, ultima visita in luglio 2023. URL: https://athesys.it/.

Bitnovo.com, 2021, Cos'è Hyperledger?, ultima visita in luglio 2023. URL: https://blog.bitnovo.com/it/cose-hyperledger/.

Capterra, 2023, Onboarding, ultima visita in luglio 2023. URL: https://www.capterra.it/glossary/257/onboarding.

Cheqd, 2023, SSI: self-sovereign or decentralised identity explained, ultima visita in luglio 2023. URL: https://cheqd.io/ssi-self-sovereign-identity-explained/.

Decentralized Identity Foundation, 2022, Universal Resolver, ultima visita in luglio 2023. URL: https://dev.uniresolver.io/.

DID Resolution: Given a DID how do I retrieve its document? – Markus Sabadello – Webinar 13 - ssimeetup.org, ultima visita in luglio 2023. URL: https://ssimeetup.org/did-resolution-given-did-how-do-retrieve-document-markus-sabadello-webinar-13/.

DIDComm support, 2023, DIDComm, ultima visita in luglio 2023. URL: https://didcomm.org/.

Ethereum, 2023, Decentralized identity, ultima visita in luglio 2023. URL: https://ethereum.org/en/decentralized-identity/.

Google LLC, 2023, Google, ultima visita in luglio 2023. URL: https://www.google.com/.

Hyperledger Foundation, 2022, How To Make Decentralized Identity Easy: An Aries JavaScript Workshop, ultima visita in luglio 2023. URL: https://www.youtube.com/watch?v=eahEPX9qxsA.

Hyperledger, 2018, CLI for Indy-SDK, ultima visita in luglio 2023. URL: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/cli/README.html.

IBM, 2023, Cos'è Zero Trust?, ultima visita in luglio 2023. URL: https://www.ibm.com/it-it/topics/zero-trust.

IBM, 2023, IAM, Cos'è IAM?, ultima visita in luglio 2023. URL: https://www.ibm.com/it-it/topics/identity-access-management.

IBM, 2023, Single Sign On (SSO), ultima visita in luglio 2023. URL: https://www.ibm.com/it-it/topics/single-sign-on.

ICT&Strategy S.r.l., 2019, Single Sign-On, accesso facilitato alle risorse di rete: ecco come funziona, ultima visita in luglio 2023. URL: https://www.cybersecurity360.

it/nuove-minacce/single-sign-on-accesso-facilitato-alle-risorse-di-rete-ecco-come-funziona/.

ICT&Strategy S.r.l., 2022, OpenID Connect: cos'è, a cosa serve e perché è importante, ultima visita in luglio 2023. URL: https://www.cybersecurity360.it/soluzioni-aziendali/openid-connect-cose-a-cosa-serve-e-perche-e-importante/.

Indicio, 2023, Indicio TestNet, ultima visita in luglio 2023. URL: https://indicio.tech/indicio-testnet/.

Indicio, 2023, Indicio, ultima visita in luglio 2023. URL: https://indicio.tech/.

Internet Engineering Task Force (IETF), 2012, The OAuth 2.0 Authorization Framework, ultima visita in luglio 2023. URL: https://datatracker.ietf.org/doc/html/rfc6749.

JSON, 2023, JSON, ultima visita in luglio 2023. URL: https://www.json.org/.

Lucid Software Inc., 2023, Che cos'è un diagramma di flusso dei dati (DFD) e come crearne uno, ultima visita in luglio 2023. URL: https://www.lucidchart.com/pages/it/diagramma-di-flusso-dei-dati.

Mashub srl, 2022, La nuova frontiera dei dati personali? La Self-sovereign identity, che coniuga facilità d'uso e privacy, ultima visita in luglio 2023. URL: https://rinascitadigitale.it/magazine/la-nuova-frontiera-dei-dati-personali/.

Meta Open Source, 2023, React, ultima visita in luglio 2023. URL: https://react.dev/.

Meta Platforms Inc, 2023, Meta, ultima visita in luglio 2023. URL: https://www.meta.com/it/.

Microsoft, 2023, TypeScript, ultima visita in luglio 2023. URL: https://www.typescriptlang.org/.

Microsoft, 2023, Visual Studio Code, ultima visita in luglio 2023. URL: https://code.visualstudio.com/.

Monokee S.r.l, 2023, Gli Hyperledger dell'identità digitale, Indy, Aries e Ursa, ultima visita in luglio 2023. URL: https://monokee.com/hyperledger-aries/.

Monokee S.r.l, 2023, Monokee, ultima visita in luglio 2023. URL: https://monokee.com/.

npm, Inc, 2023, nodemon, ultima visita in luglio 2023. URL: https://www.npmjs.com/package/nodemon.

npm, Inc, 2023, ts-node, ultima visita in luglio 2023. URL: https://www.npmjs.com/package/ts-node.

OAuth, 2023, OAuth 2.0, ultima visita in luglio 2023. URL: https://oauth.net/2/.

OpenID Foundation, 2023, OpenID, ultima visita in luglio 2023. URL: https://openid.net/.

OpenJS Foundation, 2023, Node.JS, ultima visita in luglio 2023. URL: https://nodejs.org/en.

Oracle, 2023, Che cos'è l'Identity and Access Management (IAM)?, ultima visita in luglio 2023. URL: https://www.oracle.com/it/security/identity-management/what-is-iam/.

Oracle, 2023, Che cosa si intende per SAML (Security Assertion Markup Language)?, ultima visita in luglio 2023. URL: https://www.oracle.com/it/security/cloud-security/what-is-saml/.

Osservatorio Internet Media del Politecnico di Milano, 2022, Blockchain & Web3, ultima visita in luglio 2023. URL: https://www.osservatori.net/it/ricerche/osservatori-attivi/blockchain-web3.

Osservatorio Internet Media del Politecnico di Milano, 2022, Self-Sovereign Identity (SSI): i concetti di base e alcune applicazioni internazionali, ultima visita in luglio 2023. URL: https://blog.osservatori.net/it_it/self-sovereign-identity-spiegazione-applicazioni.

Postman, Inc., 2023, Postman, ultima visita in luglio 2023. URL: https://www.postman.com/.

Red Hat, 2023, Cos'è il provisioning?, ultima visita in luglio 2023. URL: https://www.redhat.com/it/topics/automation/what-is-provisioning.

Refsnes Data, 2023, HTTP Request Methods, ultima visita in luglio 2023. URL: https://www.w3schools.com/tags/ref_httpmethods.asp.

Sovrin Foundation, 2018, What Is Hyperledger Indy?, ultima visita in luglio 2023. URL: https://sovrin.org/faq/what-is-hyperledger-indy/.

Sovrin Foundation, 2023, Sovrin, ultima visita in luglio 2023. URL: https://sovrin.org/.

SSIMeetup, 2018, Overlays 1O1: Establishing Schema Definitions within the Self-Sovereign Identity (SSI) Ecosystem, Paul Knowles, Webinar 17, ultima visita in luglio 2023. URL: https://ssimeetup.org/overlays-1o1-establishing-schema-definitions-self-sovereign-identity-ssi-ecosystem-paul-knowles-webinar-17/.

StrongDM, 2023, The Difference Between SAML vs. OIDC, ultima visita in luglio 2023. URL: https://www.strongdm.com/blog/oidc-vs-saml.

StrongLoop, IBM, and other expressjs.com contributors, 2023, Express, ultima visita in luglio 2023. URL: https://expressjs.com/.

Takahiko Kawasaki for A Medium Corporation, 2017, Diagrams of All The OpenID Connect Flows, ultima visita in luglio 2023. URL: https://darutk.medium.com/diagrams-of-all-the-openid-connect-flows-6968e3990660.

Tech Target, 2021, distributed ledger technology (DLT), ultima visita in luglio 2023. URL: https://www.techtarget.com/searchcio/definition/distributed-ledger.

TeraNet, 2020, Che cos'è e come funziona il protocollo OAuth 2.0?, ultima visita in luglio 2023. URL: https://www.teranet.it/introduzione-ad-oauth-2.

The Linux Foundation, 2023, Homepage, ultima visita in luglio 2023. URL: https://www.linuxfoundation.org/.

The Linux Foundation, 2023, Hyperledger Foundation, ultima visita in luglio 2023. URL: https://www.hyperledger.org/.

The Linux Foundation, 2023, Hyperledger Indy, ultima visita in luglio 2023. URL: https://www.hyperledger.org/use/hyperledger-indy.

The Linux Foundation, 2023, Hyperledger Ursa, ultima visita in luglio 2023. URL: https://www.hyperledger.org/use/ursa.

The PostgreSQL Global Development Group, 2023, PostgreSQL, ultima visita in luglio 2023. URL: https://www.postgresql.org/.

W3C, 2022, Verifiable Credentials Data Model v1.1, ultima visita in luglio 2023. URL: https://www.w3.org/TR/vc-data-model/.

W3C, 2023, URL, ultima visita in luglio 2023. URL: https://url.spec.whatwg.org/.

W3C, 2023, W3C, ultima visita in luglio 2023. URL: https://www.w3.org/.