/\*

(1) Creazione del DID implicitamente (chiamata da una pagina a caso; si assume che l'utente ce l'abbia già) 🡪 fatto

(2) Meccanismo di challenge-response di generazione di prova: numero casuale che viene cifrato con la chiave privata dell'utente e che viene decifrato con la chiave pubblica

(decidere se mandare un "riferimento" alla chiave pubblica utilizzata per la cifratura oppure se dall'altra parte li prova tutti) con inserimento dei dati anagrafici e successiva aggiunta del DID (si assume che l’utente abbia già il suo DID in fase di ) - Registrazione

(3) In fase di login, l'utente accede al sito presentando solo il DID e attivo il meccanismo challenge-response nuovamente (accesso con identità digitale)

(4) Quando voglio vedere un certo contenuto (es. più di 14 anni), l'utente deve presentare la VC (che contiene l'età) e la prova che è lui (che è il DID)

La Verifiable Credential ha un certo issuer; qui viene verificata l'intera catena di fiducia e si verifica che mi fido di almeno uno di questi nella catena

// Successivamente, capire se l'informazione blockchainAccountId è bastevole per dimostrare secondo ZKP è valida o meno e se

// la tipologia di firma EcdsaSecp256k1RecoverySignature2020 è prevista dallo standard ZKP; vedere e capire lo standard ZKP 🡪 Sembra che non la supporti, perché fa parte di VC, mentre l’altro specifica solo le parti presenti e le parti in gioco

All’avvio dell’applicazione

1. Prendo un account qualsiasi da quelli di test e registro il DID chiamando createDid() o eventualmente un nuovo account

Meccanismo di registrazione (protocollo) – da 2 a 11 è l’effettiva autenticazione

1. Inserisco i dati e il DID del passo prima
2. Recupero l’account dell’utente tramite web3 accounts (utente)
3. Cifro il numero con il sign di Web3 (utente):
   1. <https://web3js.readthedocs.io/en/v1.9.0/web3-eth.html#sign>
4. Genero una prova contenente il metodo di verifica, un valore di proof e il proof purpose con valore “authentication” (utente):
   1. <https://w3c.github.io/vc-data-integrity/#example-a-dataintegrityproof-example-using-a-nist-ecdsa-2022-cryptosuite>
5. L’utente invia tutto ciò
6. Mi recupero da verificationMethod mi recupero il didUrl per verificarlo
   1. <https://w3c.github.io/vc-data-integrity/#example-a-dataintegrityproof-example-using-a-nist-ecdsa-2022-cryptosuite>
7. Con questo didUrl chiamo il metodo getAuthentication di SSI.sol
8. Ciò che ritorna è un oggetto contenente blockchainAccountId
9. Uso poi recover con parametri il numero di prima, come signature il campo proof di quanto ha inviato l’utente (attenzione: NON è prefissato di suo) (massimo metti false)
   1. <https://web3js.readthedocs.io/en/v1.9.0/web3-eth-accounts.html#recover>
10. Confronto il risultato che ritorna recover con il blockchainAccountId
11. Se sono uguali, riesco ad entrare (login)/fare la registrazione

Per tutti gli issuer, mi dovrò certamente creare degli account di test da cui dipende l’emissione delle VC (la root o il DID viene fatta nello stesso modo dell’utente normale), mentre tutti gli altri enti che fanno riferimento all’altro come root, verranno creati con childTrustedDid; il processo è ricorsivo, partendo dalla root, andando avanti e quindi, nel campo from specifico di volta in volta l’address.

Flusso logico (setup) 🡪 fatto

Mi creo una serie di issuers che si fidano gli uni degli altri (3/4)

Il primo fa il createDid() e poi gli altri me li creo con il createChildTrusted.

Già pronti all’avvio

Alla verifica:

* Richiedere la Verifiable Credential all’accesso del contenuto (rilasciata dall’issuer); tramite un server tramite semplice endpoint in locale e poi ottenerla 🡪 fatto
  + Provo a vedere discorso richiesta; per il momento, data la non semplicità della cosa, me la genero in locale e simulo un delay giusto per rendere l’applicazione più credibile
  + Dentro il codice ho inserito tutti gli appunti logici/per tesi/di riferimento di tutto l’ambaradan realizzato
* L’utente in locale si deve generare la VP con ZKP all’interno (dipende dall’età) 🡪 fatto
* Il cinema verifica la proof (in questo caso, opera da verifier sul tipo di firma usato per ZKP ed estrae gli opportuni dati)
  + Nella fase di verifica della proof, ho la verifica dell’issuer (qui il resolveChain)

ZKP.

1. First, you need to verify the integrity of each Verifiable Credential contained within the Verifiable Presentation. To do this, you need to perform the following steps for each Verifiable Credential:

- Verify the digital signature of the Verifiable Credential using the public key of the issuer. You can use a library like `jsonld-signatures` for this purpose.

- Verify that the Verifiable Credential was issued to the same subject as the Verifiable Presentation holder.

- Verify that the Verifiable Credential contains the required claims, in this case, the age claim.

- Verify that the Verifiable Credential is not revoked or expired, if this information is available.

2. Once you have verified the Verifiable Credentials, you need to verify the Zero Knowledge Proofs contained within each Verifiable Credential. To do this, you need to perform the following steps for each Zero Knowledge Proof:

- Verify the proof using the public parameters of the corresponding credential schema.

- Verify that the proof demonstrates the validity of the required claims, in this case, the age claim.

3. Finally, you need to verify that the Verifiable Presentation contains all the necessary Verifiable Credentials, and that the holder of the Verifiable Presentation is the same as the holder of the Verifiable Credentials.

Per creare una Verifiable Presentation basata su un insieme di Verifiable Credentials, l'utente deve disporre di un issuer che ha emesso una Verifiable Credential in modo che l'utente possa derivare una prova dalla Verifiable Credential originariamente emessa. Ciò implica che l'utente può dimostrare la validità della firma dell'issuer senza rivelare i valori che sono stati firmati o rivelando solo determinati valori selezionati.

Per utilizzare una Verifiable Presentation zero-knowledge, è necessario che la Verifiable Credential contenga una Proof che consente all'utente di derivare una Verifiable Presentation che rivela solo le informazioni che l'utente intende rivelare. Inoltre, se viene utilizzata una definizione di credenziale, la definizione di credenziale deve essere definita nella proprietà credentialSchema, in modo che possa essere utilizzata da tutte le parti per eseguire varie operazioni crittografiche in zero-knowledge.