// Verifiable Presentation + Verifiable Credentials + Zero Knowledge Proof del DID che viene presentato dall'utente

// Definirmi l'insieme degli issuer che possono rilasciare la VC

/\*

(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.