Laurea Triennale in Informatica - Università di Salerno -
Sistema di autenticazione sulla rete Blockchain Ethereum

KRYPTOAUTH

Manuale delle Istruzioni

KryptoAuth

Riferimento	
Versione	1.0
Data	29/07/2022
Presentato da	Montefusco Alberto

Sommario

1. Introduzione

1.1 Scopo del Sistema

La realizzazione di KryptoAuth ha l'obiettivo di offrire maggiore sicurezza durante l'operazione di autenticazione sfruttando la tecnologia Blockchain Ethereum.

1.2 Scopo del Documento

Lo scopo di questo documento è di aiutare l'utente nell'installazione del sistema e, qualora mancanti, di tutte le componenti necessarie al suo funzionamento. In particolare, sarà mostrata la procedura di deploy dello Smart Contract sulla Blockchain di test Ganache, la creazione e la configurazione della rete per l'interfacciamento tra Metamask e Ganache e, infine, la traduzione dello Smart Contract scritto in Solidity in una classe Java mediante l'uso di solc.js e Web3j.

2. Requisiti per l'installazione

Requisiti lato client:

- web Browser;
- estensione Metamask;
- connessione ad Internet.

Requisiti lato server, necessari per l'uso di KryptoAuth:

- **Ganache**, requisito base per il funzionamento del sistema (**LINK**: Ganache);
- **Web3j**, con il quale si potrà convertire lo Smart Contract in linguaggio Java ed interfacciare la Web DApp con Ganache (**LINK**: Web3j);
- **Solc**, per generare l'abi e il bin dello Smart Contract (**LINK**: solc.js);
- **Truffle**, per effettuare il deploy dello Smart Contract su Ganache (**LINK**: Truffle).

2.1 Deploy dello Smart Contract

Per effettuare il deploy dello Smart Contract bisogna dapprima aprire l'applicazione Ganache: all'avvio possiamo creare un workspace personalizzato oppure avviarne uno di default tramite la sezione "Quickstart".

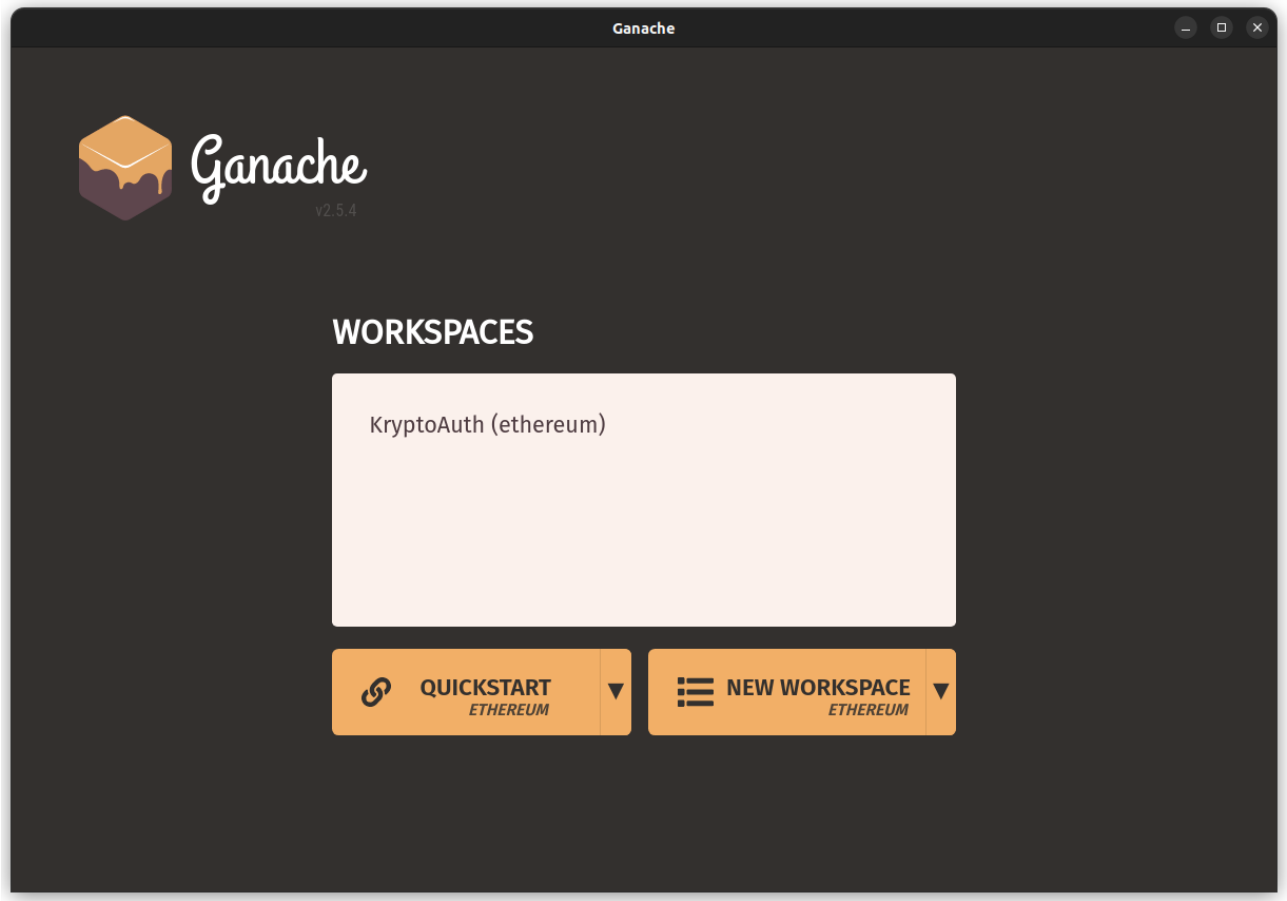

Figura 1. Homepage Ganache

Dopodiché, apriamo il nostro terminale e ci rechiamo nel package "smart contract" all'interno del progetto KryptoAuth. All'interno del terminale digitiamo:

- `truffle compile`, per verificare se ci sono errori sintattici all'interno dello Smart Contract;
- `truffle migrate`, per deployare lo Smart Contract (per effettuare un reset delle connessioni eseguiamo `truffle migrate --reset`).

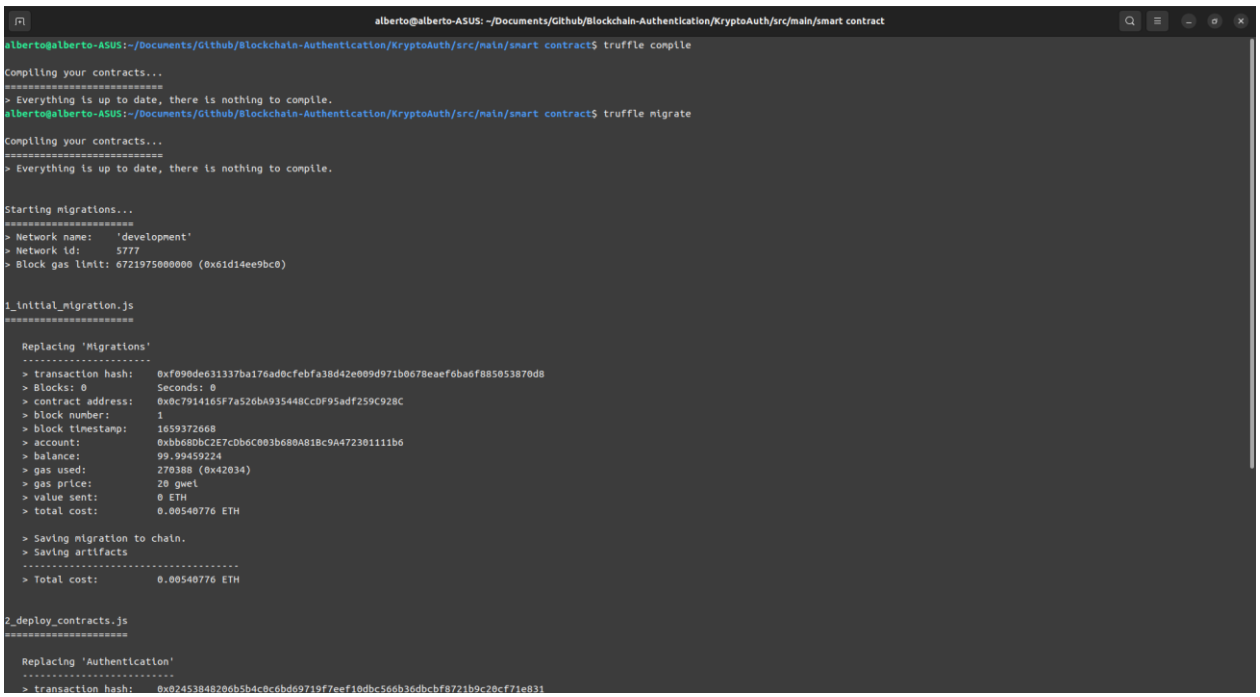

Figura 2. Compilazione Smart Contract

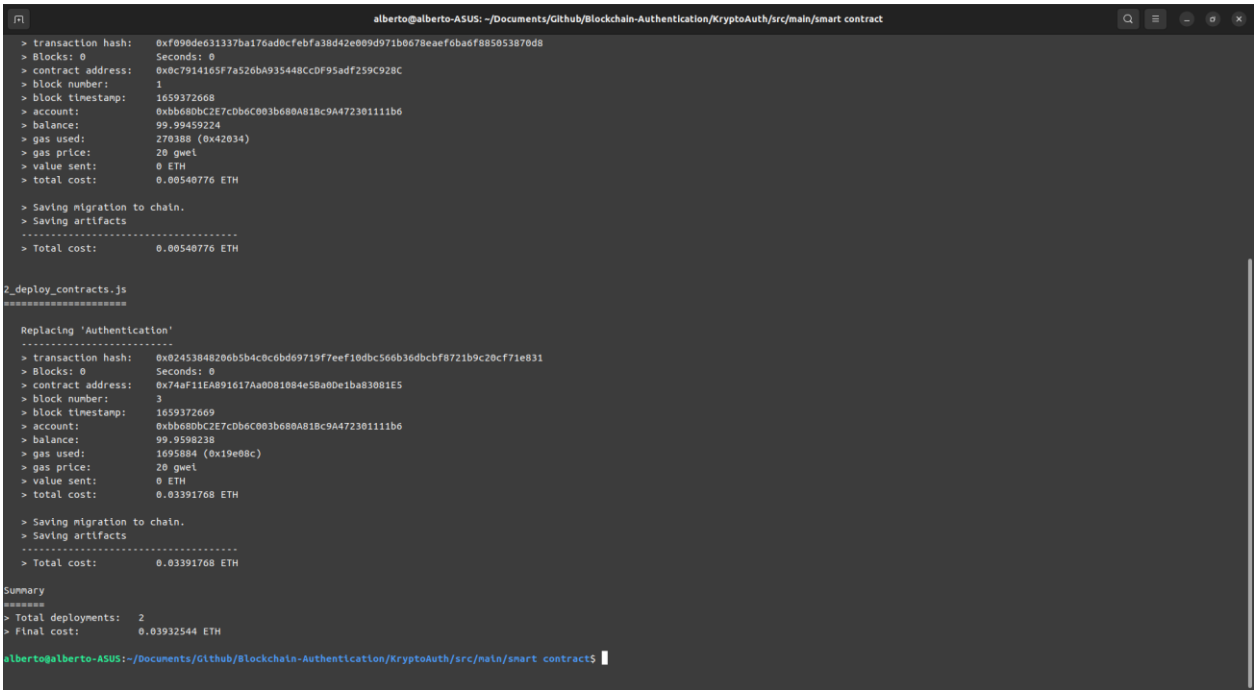

Figura 3. Deploy Smart Contract

Una volta completata la procedura il nostro contratto sarà deployato sulla Blockchain Ganache. Infatti, nella Figura 3., possiamo notare che il primo address ha speso 0.03932544 ETH per effettuare la seguente transazione.

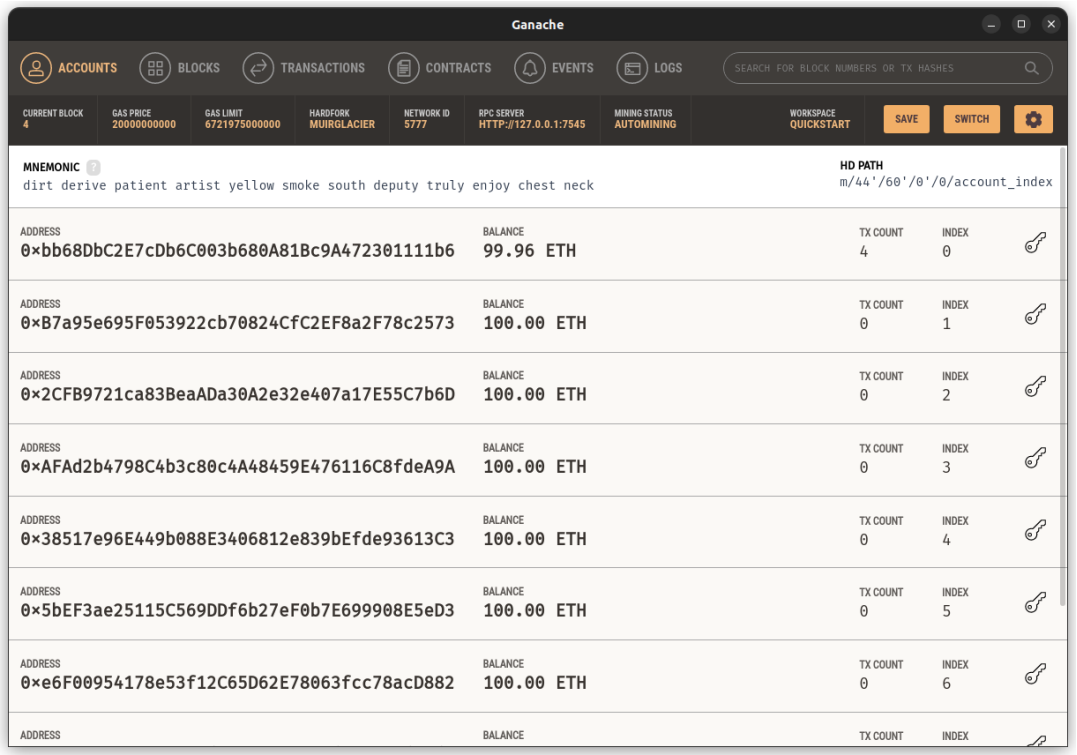

Figura 5. Accounts Ganache

Nella sezione “Contracts” ci sono gli indirizzi dei contratti deployati.

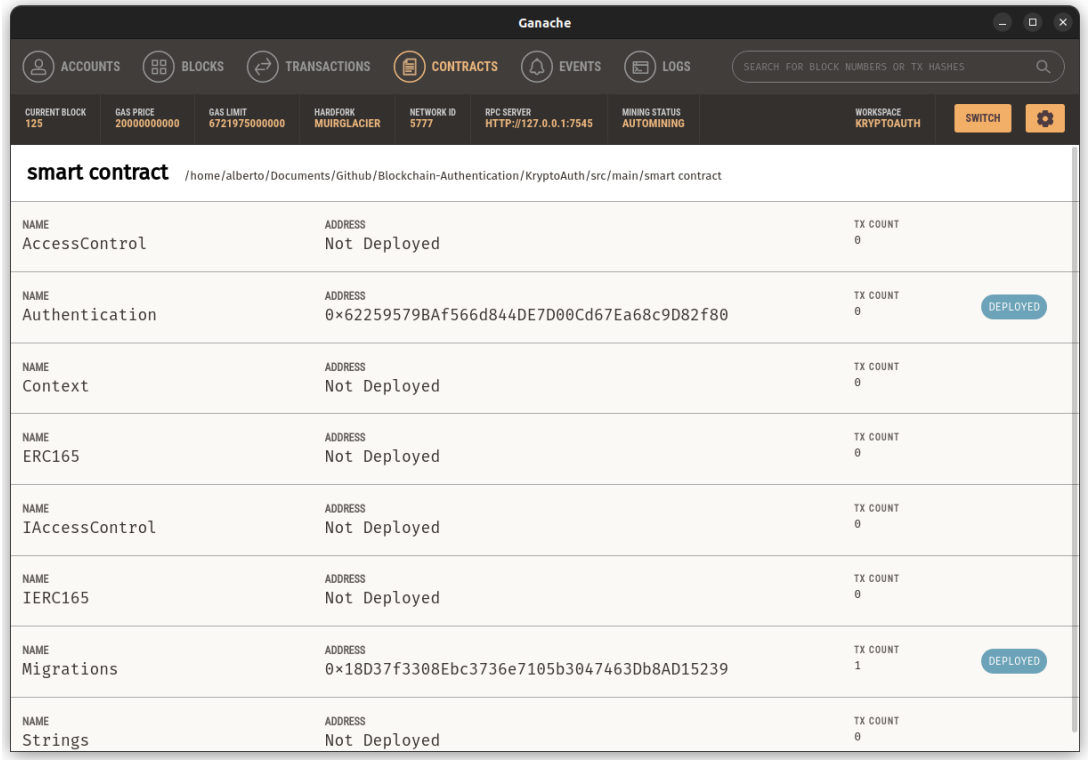

Figura 5. Contracts Address Ganache

2.2 Configurazione rete Metamask - Ganache

Per collegare l'estensione browser Metamask alla Blockchain Ganache i passaggi iniziali da seguire sono di installare e di registrarsi a Metamask; successivamente dalle impostazioni dell'estensione andiamo nella sezione "Aggiungi Rete" per creare una nuova rete.

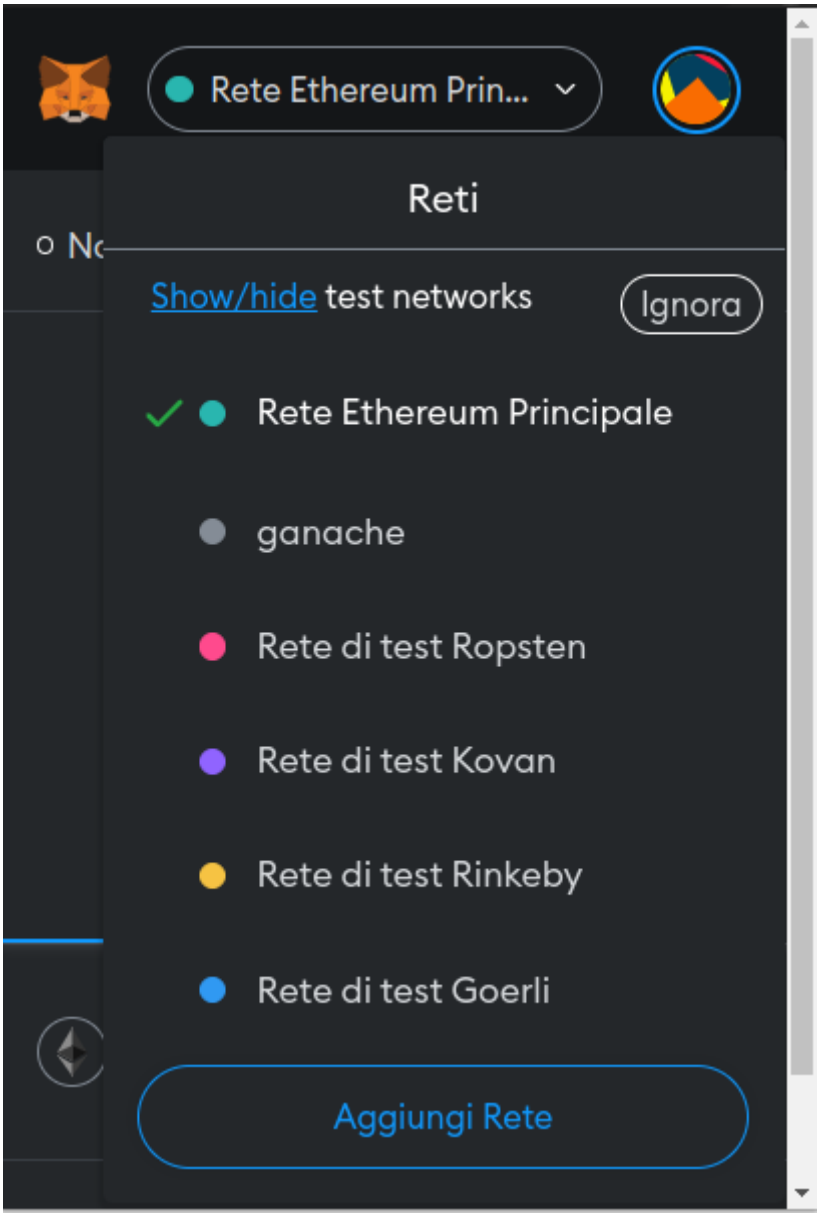

Figura 6. Aggiungi Rete

Le informazioni che devono essere inserite devono essere reperite da Ganache, in particolare:

- Nome rete: assegniamo il nome che vorremmo che abbia la nuova rete;
- Nuovo URL RPC: è l'indirizzo http di Ganache (default HTTP://127.0.0.1:7545);
- Chain ID: si deve inserire 1337 che corrisponde all'id di Ethereum;
- Currency Symbol: si deve inserire "ETH" se abbiamo una Blockchain Ethereum.

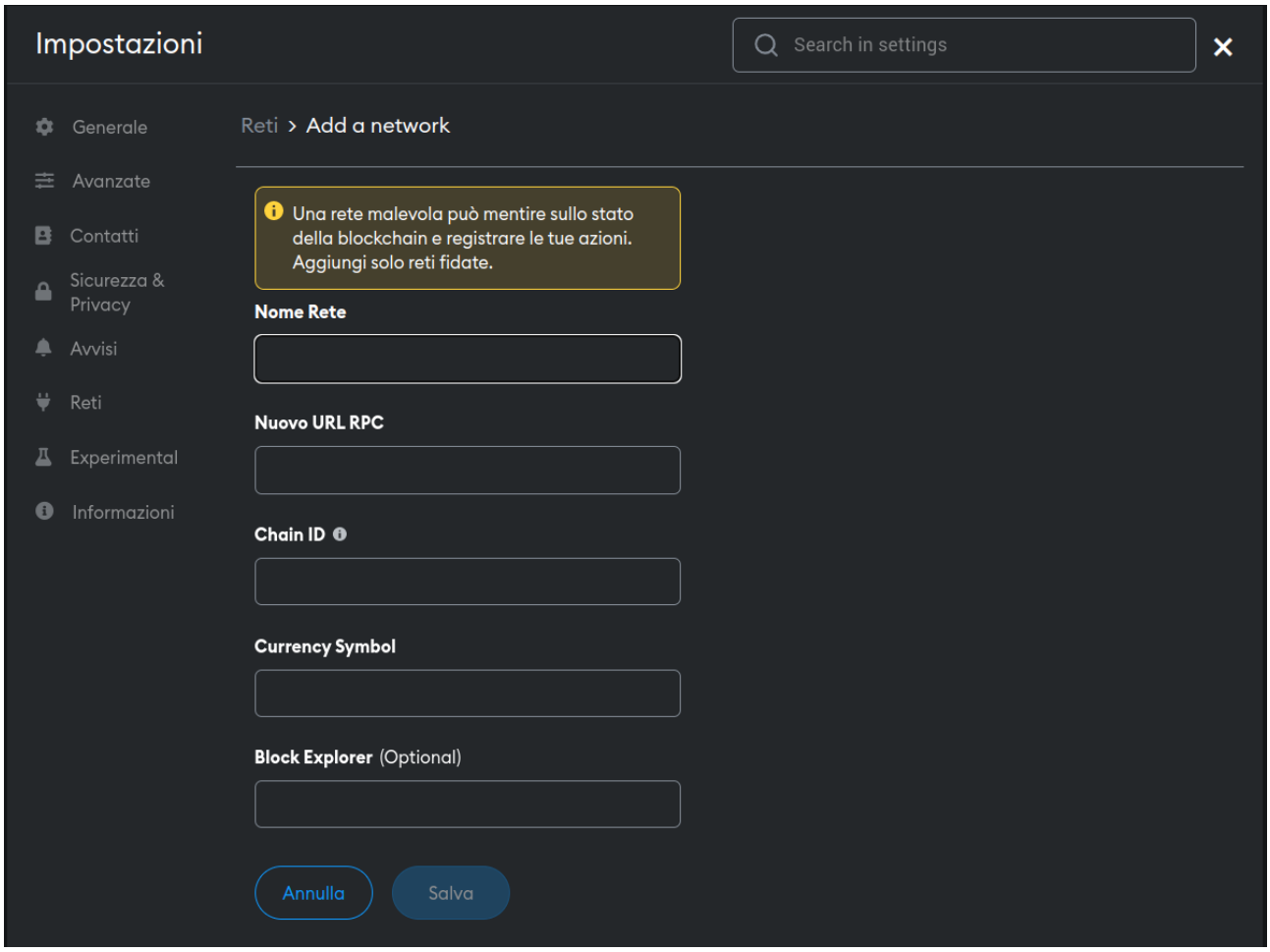

Figura 7. Configurazione Rete

Salviamo la nuova rete ed importiamo gli account da Ganache andando nella sezione "Importa Account" di Metamask.

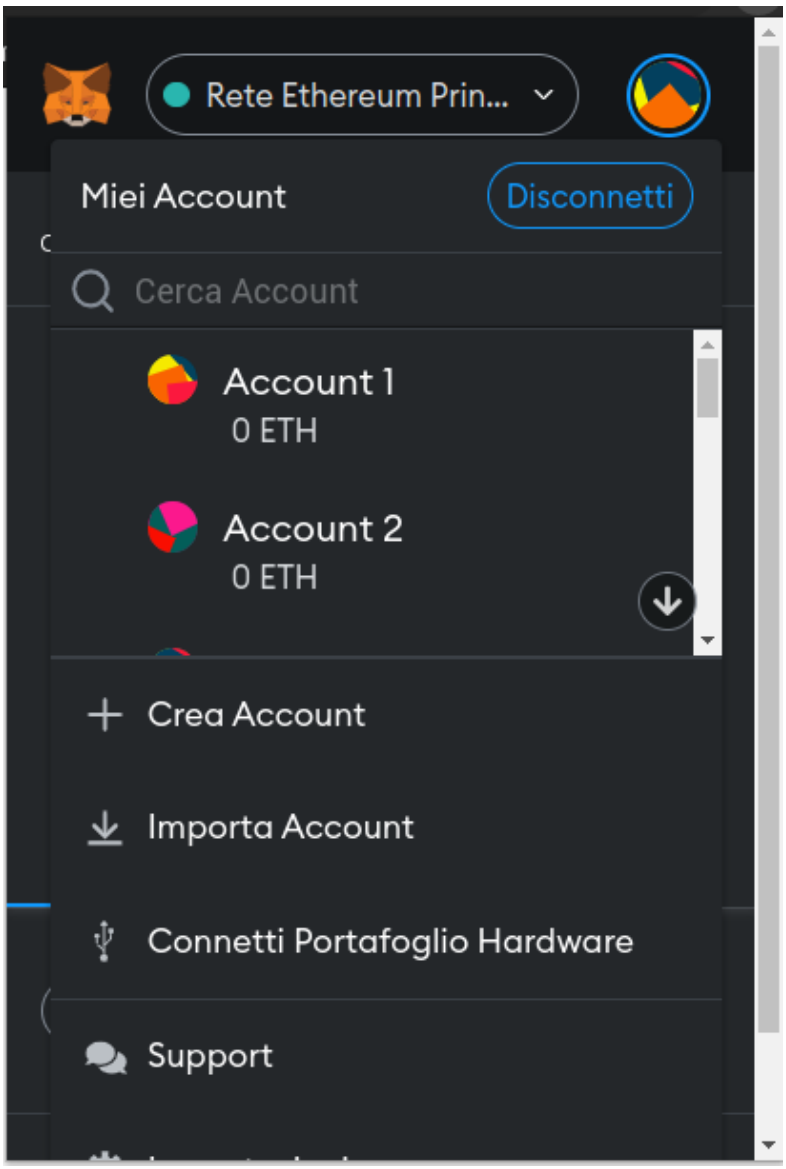

Figura 8. Importa Account

2.3 Installazione Web DApp

Per poter utilizzare la Web DApp KryptoAuth bisogna accedere dall'IDE JetBrains IntelliJ e avviare la classe main "KryptoAuth". Avviato il server Tomcat grazie a Spring Boot, accediamo al nostro Web Browser alla pagina http://localhost:8080/kryptoauth.

Nel caso in cui si volesse tradurre nuovamente lo Smart Contract scritto in Solidity in una classe Java, nel terminale digitiamo:

Figura 9. Generazione .abi e .bin

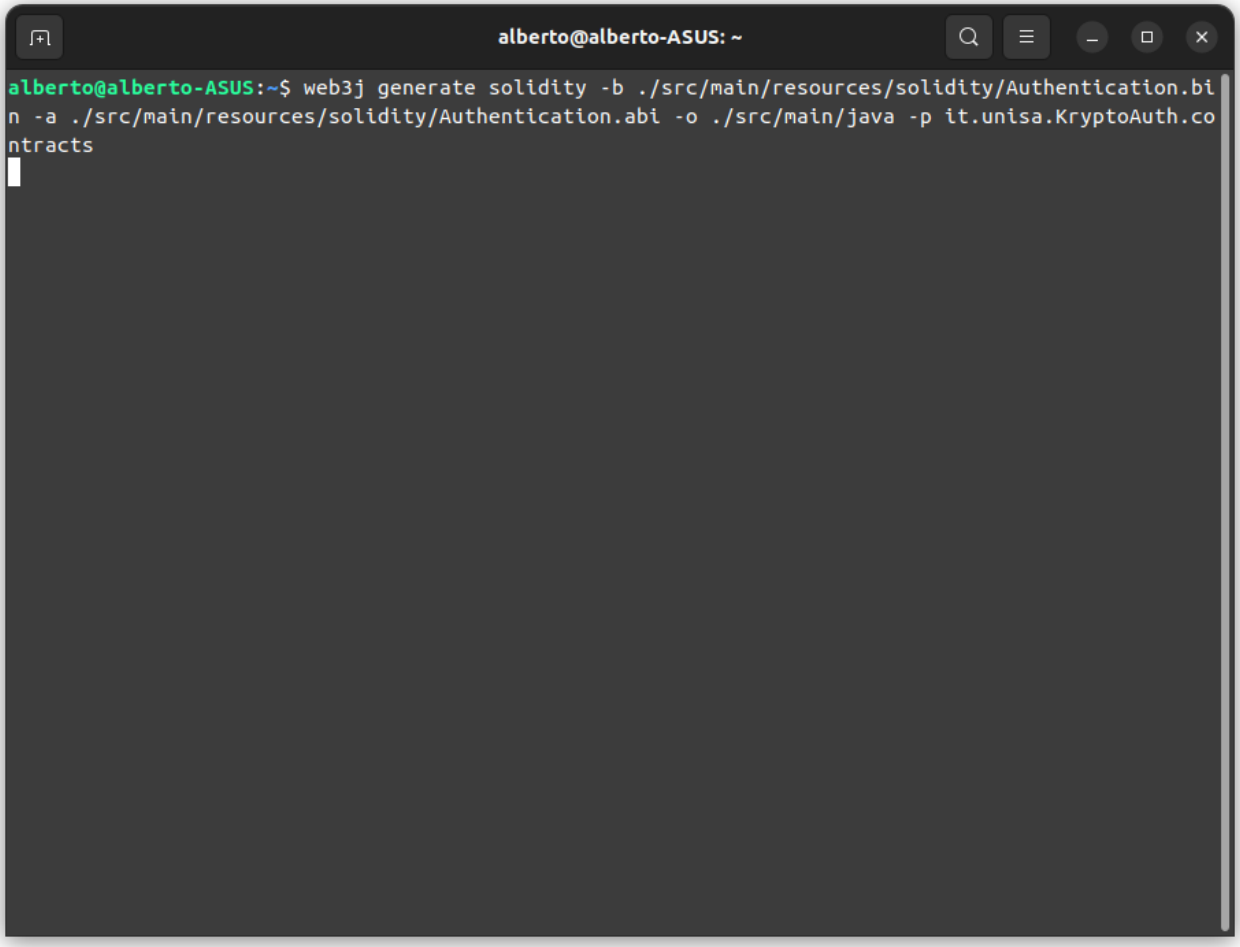

Figura 10. Generazione classe Java