python + django

mongo DB

Piattaforma Multichain di scambio token ERC20 e deposito/ritiro da Smart Contract "Salvadanaio".

LOCAL BLOCKCHAIN GANACHE & TESTNET GOERLI

PATOKEN ERC20 PLATFORM

- Utilizzando l'IDE **VISUAL STUDIO CODE** ed avvalendosi della distribuzione **PyMongo** all'interno framework **Django** per l'archiviazione documentale su database non SQL **MongoDb**, si è realizzata un'interfaccia grafica che, tramite libreria **WEB3.PY**, assegna al primo User Amministratore creato in fase di migrazione DB e setting iniziale da Terminale (**python manage.py createsuperuser**) il ruolo di Deployer di un contratto in Standard ERC20 e di Faucet dei relativi Token distribuiti in numero casuale ai successivi Utenti standard che si registrano dall' home menù raggiungibile, dopo avvio del server, all'Url http://127.0.0.1:8000/

Nota avvio Server:
Al fine di preservare la riservatezza delle private Key degli EoA, la modalità DEBUG è disattivata di Default in setting.py rendendo inaccessibili anche i file Statici e contenuti CSS.
Avviare dunque il server secondo la modalità:
python manage.py runserver --insecure

PATOKEN ERC20 PLATFORM

- Il numero di Accounts che si sceglie di impostare in fase di avvio workspace Ganache (*default 10*) determinerà il numero massimo di utenti registrabili all'interno della piattaforma.

- All'accesso dell'Admin o di qualsiasi utente creato successivamente direttamente all'interno del contesto, verrà automaticamente Deployato uno esotico Token in standard ERC20 chiamato "PATOKEN" e verranno mintati 10000 TOKEN.

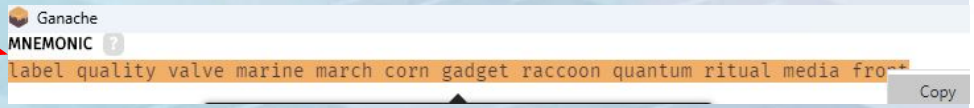
- Ad ogni nuovo utente registrato la piattaforma attribuirà un numero casuale di token compreso tra 100 e 200. L'attribuzione delle quantità di token verrà prima archiviata in una collection MongoDb e da essa verranno inizialmente propagate delle effettive TX su Ganache LocalBC verso gli address associati ai diversi account, poi eventualmente ripetibili su Goerli BC dopo deploy relativo contratto.

PATOKEN ERC20 PLATFORM

- All'utente non viene mai richiesto di scegliere,copiare ed incollare uno specifico Address account presente sulla GUI Ganache né di recuperarne la relativa Private Key.

- La scelta è motivata dalla volontà di rendere "user friendly" la piattaforma rendendo le funzionalità offerte trasparenti rispetto alle complessità intrinseche delle operazioni su Blockchain.

- L'utente dovrà semplicemente inserire la "MNEMONIC PHRASE" del WORKSPACE appena avviato sulla GUI GANACHE in esecuzione locale.



DEFAULT SETTINGS:
RPC SERVER HTTP://127.0.0.1:8546
NETWORK ID 5777

PATOKEN ERC20 PLATFORM

- Con o senza il login utente, l'homepage modifica dinamicamente il suo aspetto per indicare l'avvenuto Deploy contratto, il numero massimo di account registrabili o , a login avvenuto, le diverse funzionalità offerte dalla piattaforma:

DESCRIZIONE FUNZIONALITA'

—--Funzioni disponibili per tutti gli utenti —

- **Mostra TOTAL SUPPLY** : in fase di progettazione SmartContract e scrittura in Solidity del file PATOKEN.SOL vengono definite caratteristiche e funzionalità che avrà il Token in Standard ERC20 incluso il numero massimo di token circolanti.
  La funzione consente di interrogare il contratto deployato presentandone il relativo dato.

- **Trasferisci "PatokenErc20" ad un altro Address o deposita al MoneyBox**:
  Accedendo a questa pagina l'utente potrà individuare tra gli Address Account presenti sulla GUI Ganache cui è sincronizzato grazie alla Mnmonic Phrase, un indirizzo diverso dal proprio a cui sarà possibile trasferire una certa quantità di Token, segnalando l'eventuale inserimento di indirizzi errati (o non in piattaforma) o quantità non disponibili nel proprio bilancio.
  Oltre al contratto ERC20, l'Admin eseguirà al primo accesso il Deploy di un ulteriore SmartContract "MoneyBox" a cui gli utenti potranno inviare, dopo TX di inserimento in whiteList e TX di Approvazione spesa, i propri token avendo tale contratto la funzionalità di "Salvadanaio".
  Una volta che l'Admin avrà per eseguito l'accesso all'ulteriore sezione linkata con l'immagine Salvadanaio, le stesse funzionalità saranno disponibili anche nella BC Goerli.

DESCRIZIONE FUNZIONALITA'

—--Funzioni disponibili per tutti gli utenti —

- **Visualizza saldo PATOKEN “Current EoA” depositato sul MoneyBox** :
  Su entrambe le chain richiamando l’ABI del contratto MoneyBox precedentemente generata dalla compilazione tramite Framework “Brownie” del relativo file Solidity, usando gli address di Deploy archiviati come campi del Faucet, si è realizzata una view che interroga il contratto e ne restituisce il Bilancio associato allo specifico EoA autenticato in piattaforma.

- **Ritira Patoken precedentemente depositati sul MoneyBox SC dal EoA autenticato:**

  Analogamente alla precedente funzione, si realizza un'interfaccia che consente di propagare delle TX che, modificando lo stato della Blockchain , richiamano la funzione withdrawERC20ForUser dello SmartContract riaccreditando l’EoA dei token precedentemente depositati.

DESCRIZIONE FUNZIONALITA'

—--Funzioni disponibili per utente Admin—

- **Visualizza saldi Patoken** : successivamente al recupero del bilancio associato ciascun Address, tramite interrogazione ciclica dell' istanza dello SmartContract ERC20 deployato , viene presentata una vista che mostra tutti i saldi.

- **Analizza gli eventi di Trasferimento Token**: da analisi e manipolazione delle Transaction susseguitesi nei blocchi a seguito di trasferimenti o depositi token tra account o verso MoneyBox, verranno recuperati i dati relativi agli eventi "Transfer" la cui definizione ed implementazione è ereditata dallo standard Erc20 a cui il contratto Patoken fa riferimento sul file Solidity. Tali eventi verranno archiviati all'interno di una collection MongoDb

DESCRIZIONE FUNZIONALITA'

- **Contatore Eventi Trasferimento Token in Piattaforma**:
  Su entrambe le chain vengono implementati dei contatori tra loro indipendenti che dinamicamente reagiscono ad ogni evento di Trasferimento o deposito token tra EoA o verso MoneyBox

MECCANISMI CRITTOGRAFICI PROTEZIONE PK

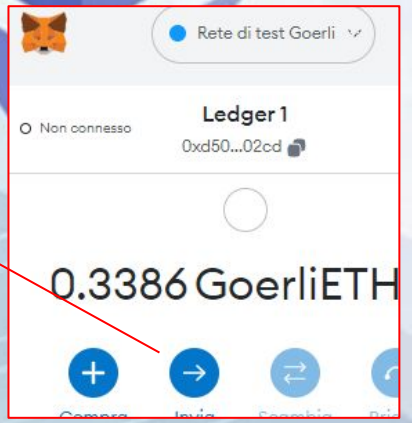
- Per garantire la sicurezza delle chiavi Private attraverso cui gli utenti gestiscono i propri fondi internamente la piattaforma centralizzata , una volta generati ed associati ad ogni EoA l'address e la Private Key quest'ultima viene archiviata sulla collection Mongo solo a seguito di processo di cifratura della stessa tramite le seguenti librerie:

```
from cryptography.fernet import Fernet
import pickle
```

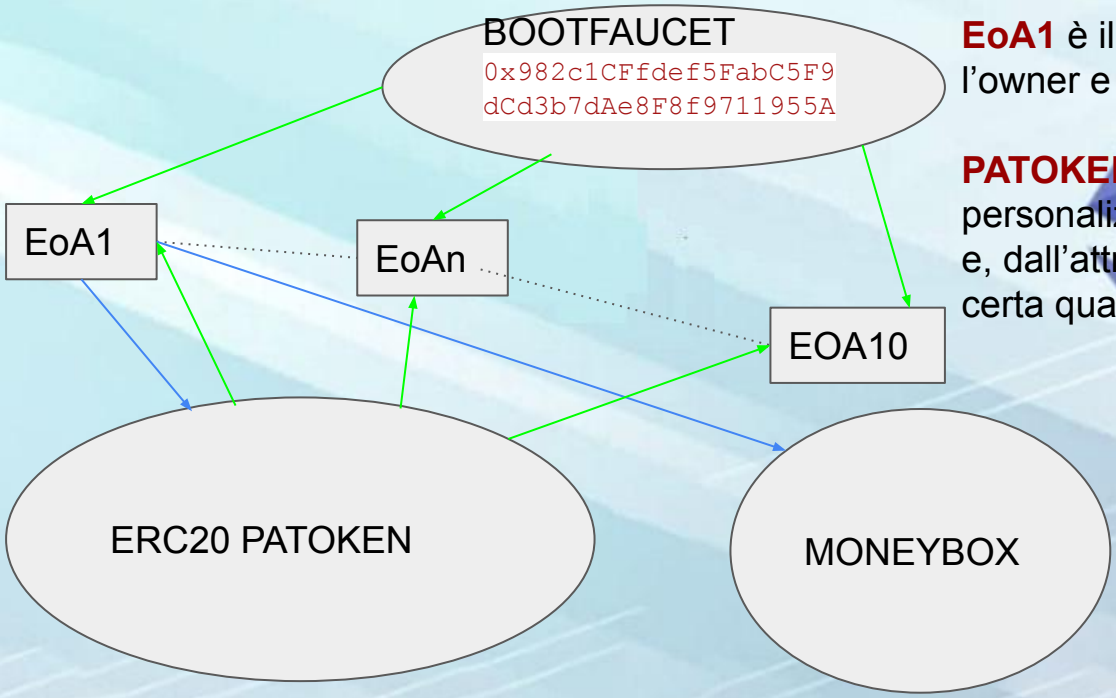
Utilizzando le stesse attraverso le funzioni **createCryptographicKey(),manageCryptPk(), goCript(),goDecript**() ed un file Binario **systemFernet.pickle** custode del segreto, le chiavi di utilizzo fondi risultano archiviate ma non codificate in chiaro ed accessibili solo dall'account autenticato nel contesto

MECCANISMI SPECIFICI SU TESTNET PUBBLICA GOERLI

- Per effettuare il primo accesso alle funzionalità della piattaforma su Goerli BC con conseguente deploy del contratto su tale rete, occorre prima alimentare l'Address Randomicamente generato ed associato all'Admin, Faucet della piattaforma, con degli Ether di Test inviate dal Proprio Metamask



SINOTTICO SMART CONTRACT PIATTAFORMA



EoA1 è il superUser creato in fase di setting iniziale; è l'owner e deployer dei contratti Patoken e MoneyBox.

PATOKEN è lo Smartcontract che istanzia il Token personalizzato definendone la total supply(10000) e, dall'attribuzione randomica iniziale, ne assegna una certa quantità ai diversi EoA

BOOTFAUCET è lo SmartContract deployato esternamente dal Developer all'address **0x982c1CFfdef5FabC5F9dCd3b7dAe8F8f9711955A** e rifornito con fondi adeguati per garantire il recharge di tutti i potenziali EoA o per situazioni di congestione anomale.
Per accedere a tali fondi è comunque necessario eseguire almeno il 1°deposito

Environment post install requirements

Strumenti ed applicativi per lo sviluppo

- Python 3.9.0
- Visual Studio Code
- MongoDB
- Node.js
- Ganache

Librerie ed ulteriori upgrade da richiamare da terminale successive all'installazione del requirements.txt

- npm install -g ganache-cli
- brownie pm install OpenZeppelin/openzeppelin-contracts@4.0.0
- pip install --upgrade pymongo==3.12

REPOSITORY GIT:

- **https://github.com/antopat1/ProgettoEthereumWeb3diAntoninoPaterno**