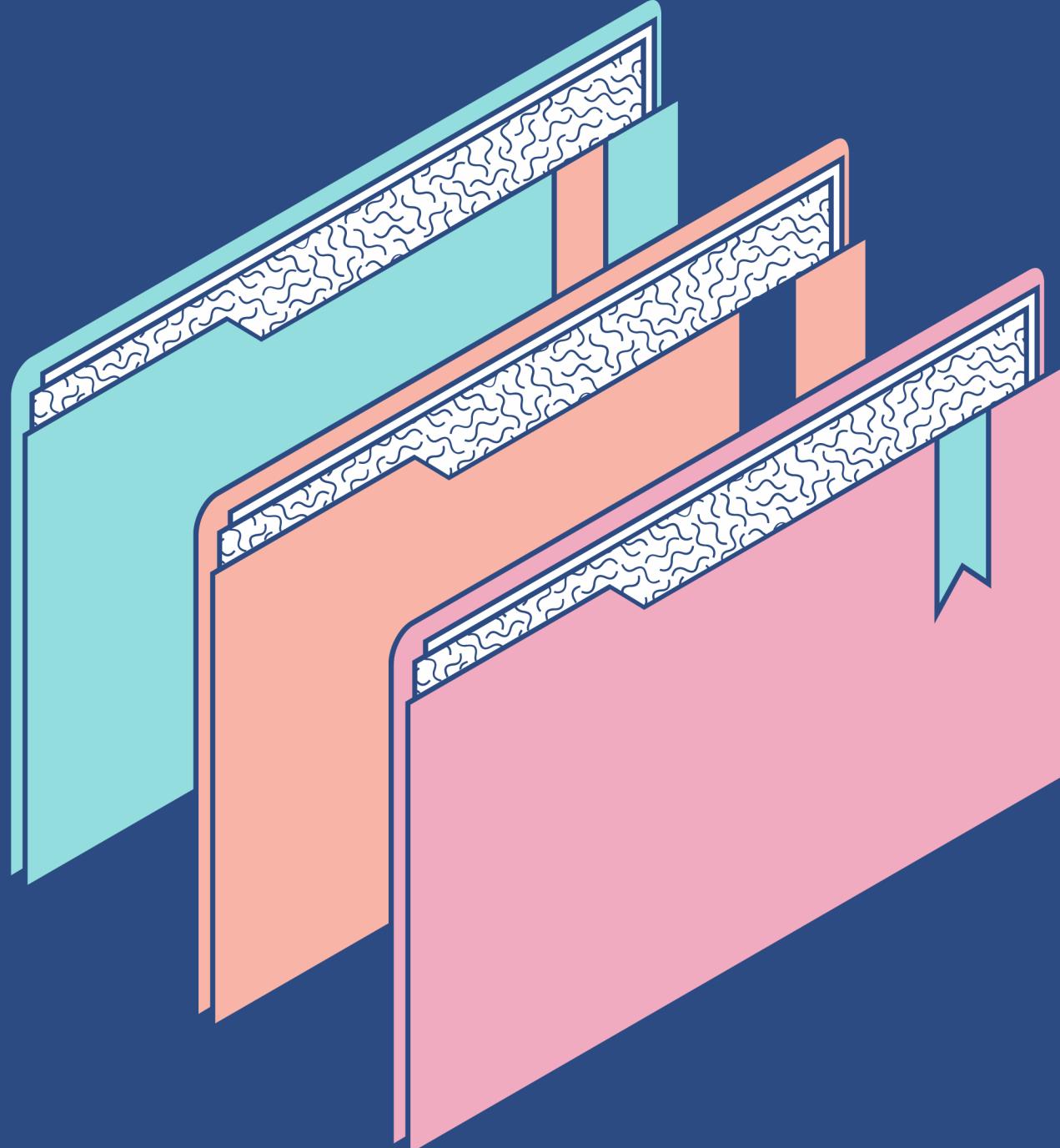




CO₂ APPLICATION

Progetto per l'esame di Software Security
and Blockchain

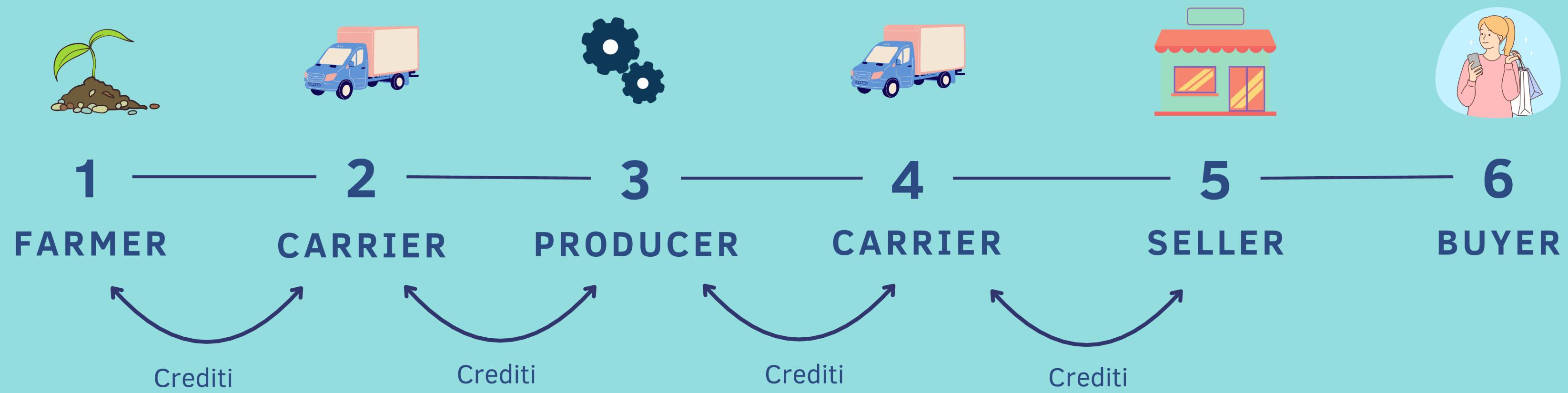
A CURA DI:
Gabriel Piercecchi
Tosca Pierro
Luca Pigliacampo
Caterina Sabatini

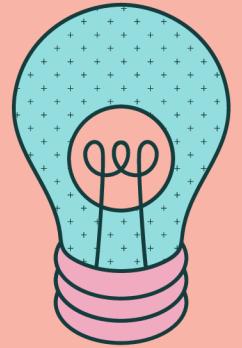


PUNTI CHIAVE

- Mostrare la catena di produzione dei singoli prodotti
- Registrare le emissioni di CO₂
- Dare *Bonus* o *Malus* in base alle emissioni di CO₂
- Software con parte on-chain e off-chain

Come funziona la filiera





Requisiti



BUYER

Visualizza tutta la filiera produttiva comprese le emissioni di Carbonio

FARMER/PRODUCER/SELLER

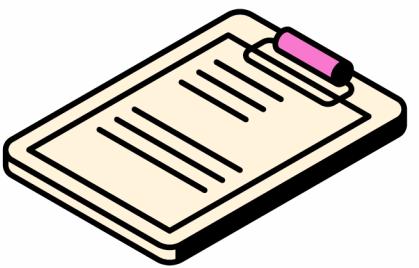
- Inserimento di prodotti
- Richieste di prodotti
- Richiesta di Coin

CARRIER

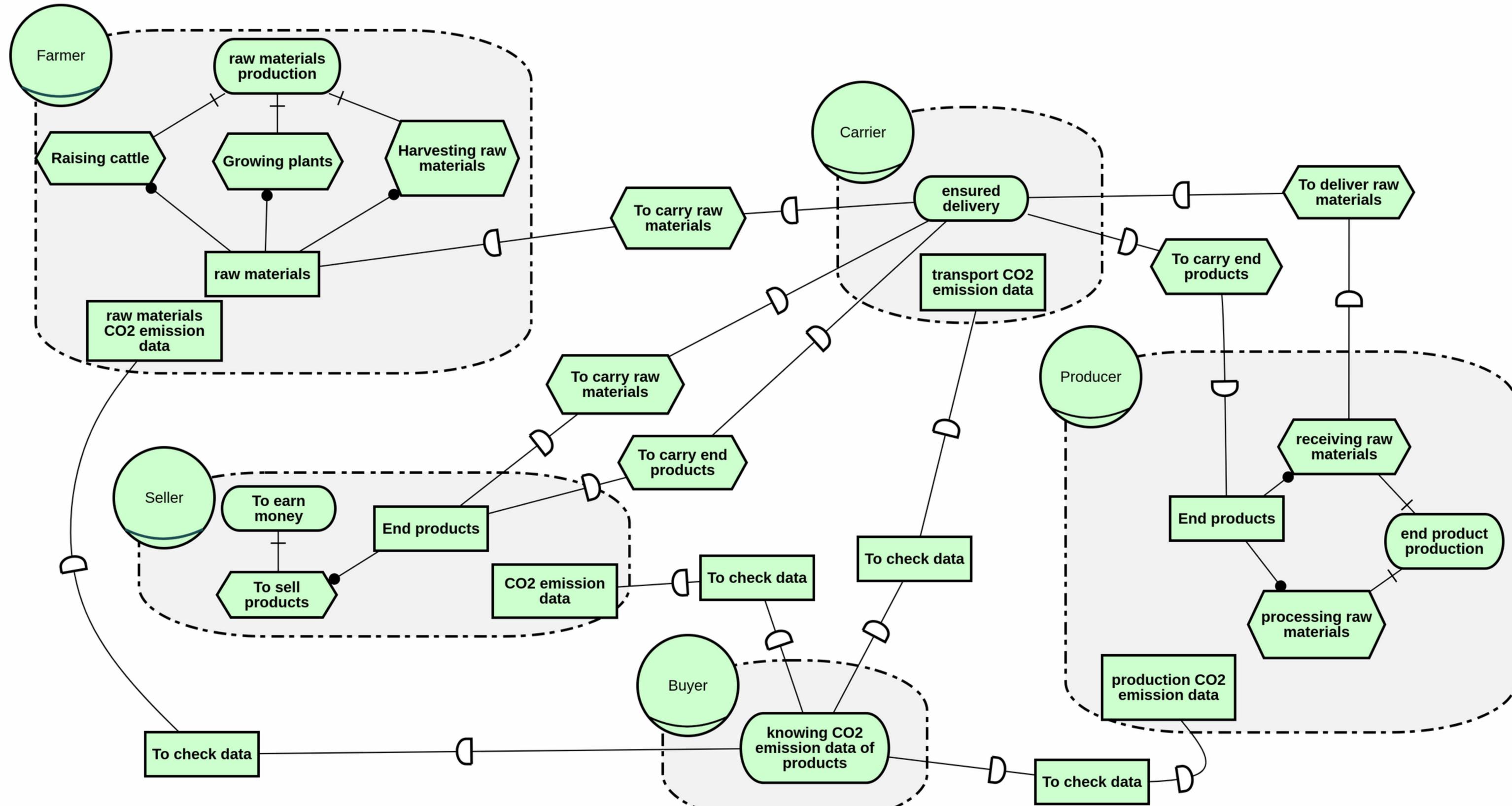
Invia i prodotti alle varie aziende

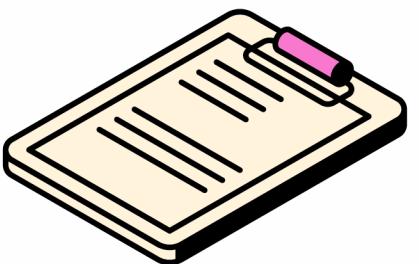
COIN

Scambiare Coin

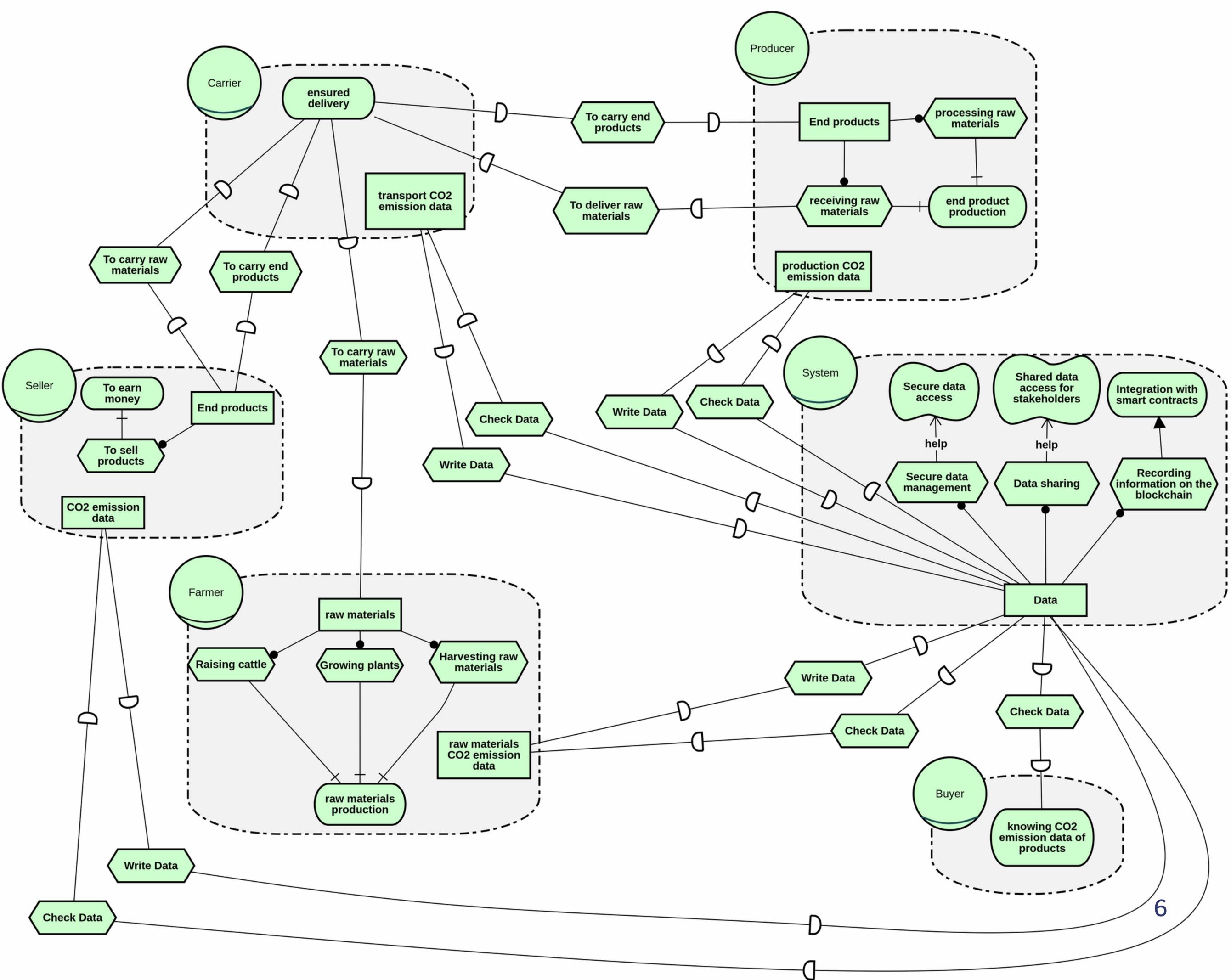


Progettazione Modello i* SENZA sistema





Progettazione Modello i* CON sistema



ASSETS

Asset	Value	Objective
Dati	Risorsa cardine del software di cui si deve garantire la sicurezza e la disponibilità	Integrity, Availability, Resilience
Scrittura dei dati	L'integrità e l'affidabilità e precisione dei dati delle emissioni di CO2 misurate immesse nel sistema	Integrity, Non-repudiation, Authentication, Authorization
Lettura dei dati	L'attendibilità del processo di verifica per garantire l'integrità, la qualità e la disponibilità dei dati	Integrity, Availability
Gestione sicura dei dati	Le operazioni di CRUD sui dati devono essere opportunamente protette	Confidentiality, Authentication, Integrity
Accesso ai dati in maniera condivisa per gli stakeholder	Gli stakeholder autorizzati devono poter accedere ai dati quando desiderano	Integrity, Non-repudiation
Misurazione dei parametri	L'attività di rilevazione e registrazione dei parametri deve essere continuamente protetta e monitorata poiché fondamentale per il calcolo dei Carbon credits	Integrity, Confidentiality, Reliability, Non-repudiation
Registrazione delle informazioni su blockchain	È necessario porre il caricamento delle informazioni sulla blockchain in quanto quest'ultima monitorerà l'accesso ai dati in sicurezza	Confidentiality, Integrity, Availability, Authentication, Authorization, Non-repudiation
Tracciamento delle azioni	Prassi per il monitoraggio dei log prodotti dal software	Integrity, Confidentiality, Non-repudiation
Integrazione con gli smart contract	Azione volta all'integrazione dell'applicativo off-chain con la sua controparte on-chain	Authentication, Availability, Non-repudiation
Politiche di autorizzazione	Policy e criteri per la gestione delle autorizzazioni di sistema	Authorization, Non-repudiation
Politiche di autenticazione	Policy e criteri per la gestione delle autenticazione di sistema	Authorization, Non-repudiation
Wallet	Portafoglio elettronico contenente le chiavi che permettono l'autenticazione sulla blockchain	Integrity, Confidentiality, Non-repudiation

Threat Model

Asset	Value (€)	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure	Attack	Inherent Probability	Inherent Risk	Control	Cost	Feasibility	Residual Probability	Residual Impact	Residual Risk	R.o.C	Overall Cost
Politiche di autenticazione	70.000-90.000	x		x	x						30.000 - 60.000	CAPEC-114: Authentication Abuse	9.00%	2.700 - 5.400	L'uso di autenticazione multifattore, la protezione contro attacchi di forza bruta, il monitoraggio delle attività sospette e la validazione continua delle credenziali.	6.500 - 7.500	Fattibilità: Elevata. Misure come la limitazione dei tentativi di accesso e l'autenticazione multifattore (MFA) sono ampiamente supportate dalle piattaforme moderne e rappresentano soluzioni efficaci per prevenire abusi nell'autenticazione. L'implementazione di tali misure riduce significativamente i rischi di accesso non autorizzato.	2%	30.000 - 60.000	600	-0,7	7.100
		x	x	x							30.000 - 60.000	CAPEC-115: Authentication Bypass	15.00%	4.500 - 9.000	L'uso di metodi di autenticazione robusti, la validazione rigorosa degli input, la protezione contro le vulnerabilità di accesso diretto ai dati e il monitoraggio delle attività sospette.	5.200 - 6.200	Fattibilità: Elevata. L'implementazione di HTTPS e la validazione dell'input sono misure di sicurezza standard e relativamente facili da integrare, con HTTPS che garantisce la crittografia delle comunicazioni e una protezione robusta contro intercettazioni e manipolazioni. La gestione sicura delle sessioni (session management) richiede un approccio più complesso.	3%	30.000 - 60.000	900	-0,3	6.100
		x	x	x							30.000 - 60.000	CAPEC-560: Use of Known Domain Credentials	25.00 %	7.500 - 15.000	Implementare l'autenticazione a più fattori, politiche di password forti, limitare i privilegi degli account e monitorare gli accessi anomali.	7.500 - 8.500	Fattibilità: Elevata. L'utilizzo di credenziali di dominio conosciute è una vulnerabilità comune, ma le misure di sicurezza per prevenirlo sono ben consolidate. Esistono diverse soluzioni facilmente integrabili nei sistemi di autenticazione esistenti, come l'adozione di autenticazione multifattore (MFA), la gestione sicura delle credenziali e il monitoraggio delle attività sospette.	5%	30.000 - 60.000	1.500	-0,2	9.000

Schemi Jacobson

Use case

Case Type	Use Case	Case ID	UC-3	
Case Name	Autorizzazione basata sui ruoli			
Actors	Oracle, Farmer, Carrier, Producer, Seller, Buyer, CO2 Application			
Description	Funzione che assegna e verifica i permessi di accesso alle funzionalità e ai dati in base al ruolo specifico di ciascun attore nella supply chain.			
Data	Tutti i dati relativi alle informazioni personali, ai prodotti e alle emissioni di CO2.			
Stimulus and preconditions	Gli attori coinvolti, che non sono Buyer, devono autenticarsi sul sistema per ottenere i permessi legati al loro ruolo. I ruoli sono predefiniti con accessi e autorizzazioni specifici. Il Buyer ha permessi minimi e non richiede autenticazione.			
Basic Flow	1) L'utente tenta di accedere a una funzione o a un dato specifico. 2) Il sistema verifica il ruolo e le autorizzazioni assegnate. 3) Se necessario il sistema richiede l'autenticazione da parte dell'utente 4) L'utente effettua la procedura di login 5) Il sistema verifica le credenziali 6) Una volta autenticato, il sistema recupera il ruolo associato all'utente. 7) Il sistema determina le autorizzazioni e le funzionalità accessibili in base al ruolo 8) L'utente esegue l'azione prefissata 9) Registrazione attività all'interno dei log			
Alternative Flow	AF-1: Login fallito 1) L'utente tenta di accedere a una funzione o a un dato specifico. 2) Il sistema verifica il ruolo e le autorizzazioni assegnate. 3) Se necessario il sistema richiede l'autenticazione da parte dell'utente 4) L'utente effettua la procedura di login 5) Il sistema verifica le credenziali 6) Le credenziali non sono valide 7) Il sistema mostra un messaggio di errore e richiede di reinserire le credenziali.			
Exception Flow	Il sistema non è in grado di verificare le credenziali a causa di un errore tecnico. Viene mostrato un messaggio di errore personalizzato.			
Response and Postconditions	L'utente ha accesso ai dati e alle funzionalità in base ai ruoli con cui sono autorizzati			
Non Functional Requirements	Integrità, Confidentialità, Affidabilità			

Schemi Jacobson

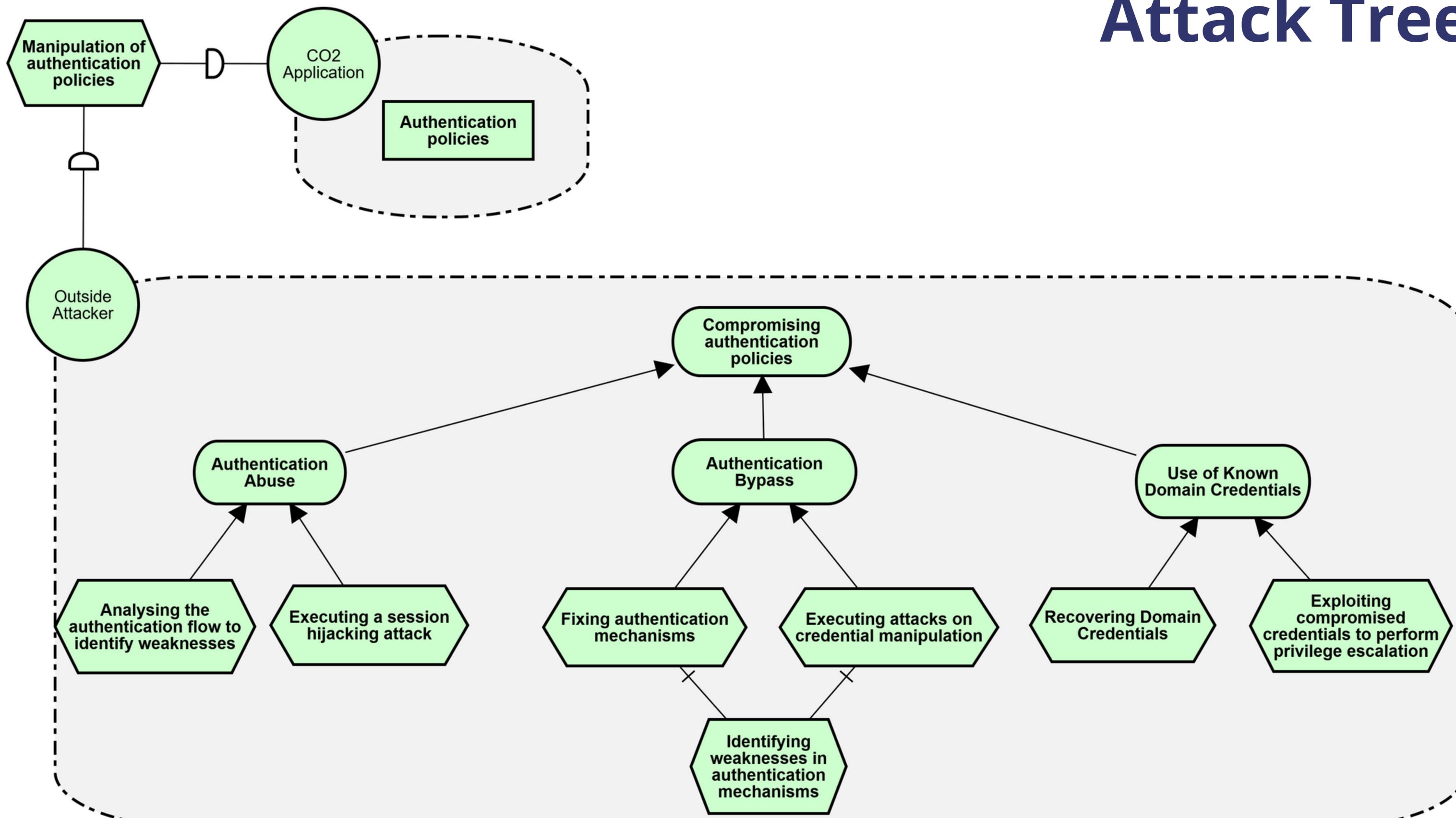
Abuse case

Misuse case

Case Type	Abuse Case	Case ID	AC-08
Case Name	Authentication Abuse		
Actors	Attaccanti, CO2 Application		
Description	Un attacco in cui un malintenzionato sfrutta vulnerabilità nel processo di autenticazione per ottenere accesso non autorizzato al sistema o alle risorse protette.		
Data	Credenziali di autenticazione		
Stimulus and preconditions	Le credenziali di accesso sono deboli o facilmente indovinabili (password deboli o predittive).		
Attack Flow 1	L'attaccante tenta di accedere al sistema utilizzando un nome utente e una password rubata o indovinata.		
Attack Flow 2	L'attaccante prova una serie di credenziali (password comuni o precedentemente esposte) per forzare l'accesso all'account.		
Attack Flow 3	-		
Response and Postconditions	Monitorare tentativi di login sospetti e anomali per identificare attacchi di tipo brute-force o credential stuffing. Avvisare l'utente di accessi non autorizzati e richiedere il cambio della password.		
Non Functional Requirements	Confidenzialità, Affidabilità, Resilienza, Disponibilità		
Mitigations	Forzare l'uso di password forti e la loro rotazione periodica.		
Comments	-		

Case Type	Misuse Case	Case ID	MC-01
Case Name	Authentication Bypass		
Actors	Attaccanti, Utenti, CO2 Application		
Description	Un utente ottiene l'accesso a dati o funzionalità riservate senza completare il processo di autenticazione, eludendo le procedure previste.		
Data	Tutti i dati riservati, non accessibili a utenti non autenticati.		
Stimulus and preconditions	L'utente non ha eseguito la procedura di autenticazione.		
Attack Flow 1	Nel tentativo di effettuare la procedura di login, l'utente ottiene l'accesso a dati riservati, senza completare correttamente il processo di autenticazione previsto.		
Attack Flow 2	Un attaccante, mediante l'ottenimento di un token o altre manipolazioni del software, induce il sistema a riconoscere erroneamente il completamento di una procedura di autorizzazione, senza averla realmente eseguita.		
Response and Postconditions	Accesso a dati privati da parte di utenti non autorizzati. Impossibilità di accesso a dati significativi per utenti con privilegi superiori.		
Non Functional Requirements	Affidabilità, Autenticazione, Confidenzialità, Non-ripudiazione		
Mitigations	Il software deve integrare meccanismi di autorizzazione robusti e prevedere un fail state che impedisca a un utente l'accesso a dati ai quali non dovrebbe poter accedere.		
Comments	-		

Attack Tree



Design

PER UN ADEGUATO
LIVELLO DI SICUREZZA

Catalogo Saltzer and Schroeder

- Psychological acceptability
 - "KISS" principle: il "Keep It Simple, Stupid"
 - Defense in depth
 - Open design
-

Catalogo Sommerville

- Log user actions
 - Avoid a single point of failure
 - Least common mechanism
-

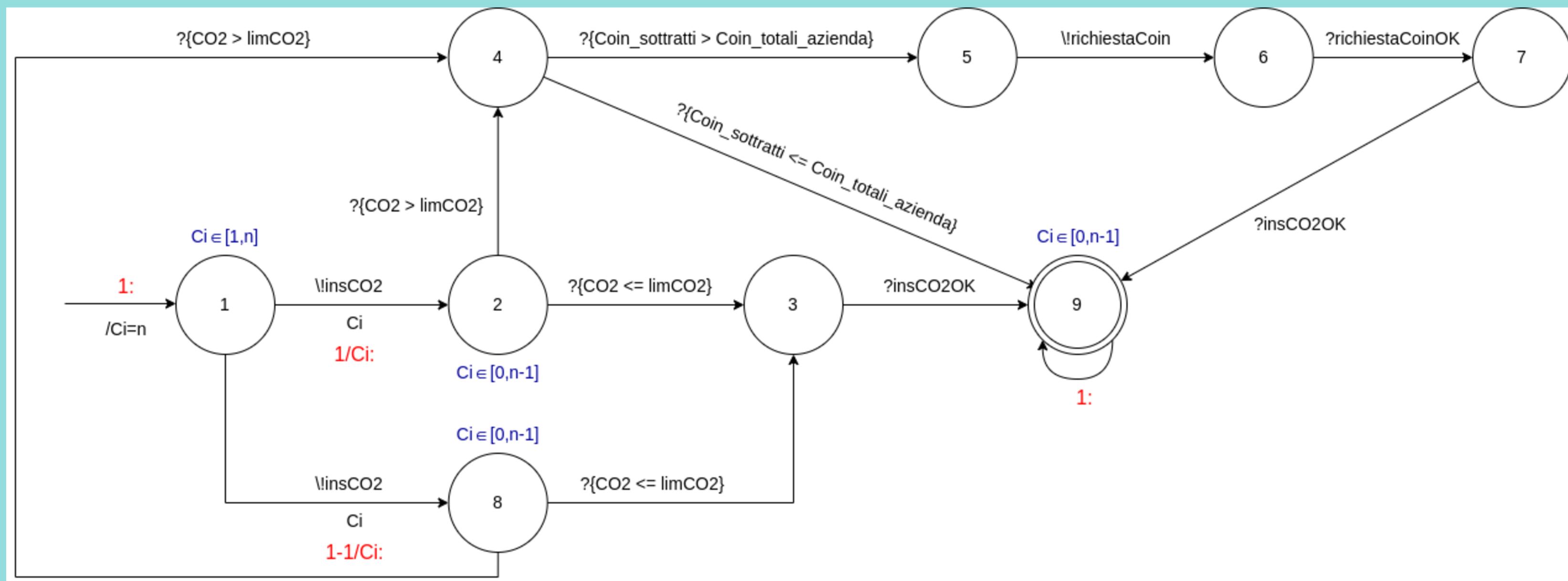
Catalogo OWASP

- Don't trust services
- Least privilege
- Fail securely

Markov Chain

MARKOV CHAIN RELATIVA ALL'INSERIMENTO DI CO₂

Per facilitare la lettura del grafico, gli elementi in nero rappresentano le etichette associate a ciascuna transazione, in rosso le probabilità associate ad alcune transazioni, mentre in blu, un'esemplificazione dei tentativi associati a ciascun inserimento.





Implementazione del Modello in PRISM

Verificare una proprietà di Safety e
una di Response

Codice prism

dtmc

```
const int n = 10;      // Numero di tentativi
const int limCO2 = 50; // Limite massimo di CO2
```

```
module ins_co2
```

```
stato : [1..10] init 1;    // Stato iniziale
ci : [0..n] init n;      // Numero di tentativi iniziali
CO2 : [1..limCO2+100] init 70; // Variabile CO2
Coin_sottratti : [0..30] init 10; // Coin sottratti in caso di Malus
Coin_azienda : [0..20] init 20; // Coin totali dell'azienda
```

```
// Stato iniziale: tentativo di inserire CO2
[insCO2] (stato=1) & (ci>0) -> (1/ci) : (stato'=2) & (ci'=ci-1) + (1-1/ci) : (stato'=8) & (ci'=ci-1);
```

```
// Bonus: Inserimento CO2 OK
[bonus] (stato=2) & (CO2<=limCO2) -> (stato'=3); // Successo se CO2 <= limCO2
[bonus] (stato=8) & (CO2<=limCO2) -> (stato'=3); // Successo nel caso di transizione da stato 9
```

```
// Malus: Inserimento CO2 KO
[malus] (stato=2) & (CO2>limCO2) -> (stato'=4); // Malus se CO2 > limCO2
[malus] (stato=8) & (CO2>limCO2) -> (stato'=4); // Malus nel caso di transizione a stato 9
```

Codice prism

```
// Transizione a stato di successivo tentativo
[OK] (stato=4) & (Coin_sottratti <= Coin_azienda)-> (stato'=9); //Raggiunto stato finale
[OK] (stato=3) -> (stato'=9); // Raggiunto stato finale

// Richiesta di coin
[ric_coin] (stato=4) & (Coin_sottratti > Coin_azienda) -> (stato'=5); // Transizione a stato 5 per richieste coin
[attesa] (stato=5) -> (stato'=6); // Transizione a stato 6 per attesa risposta

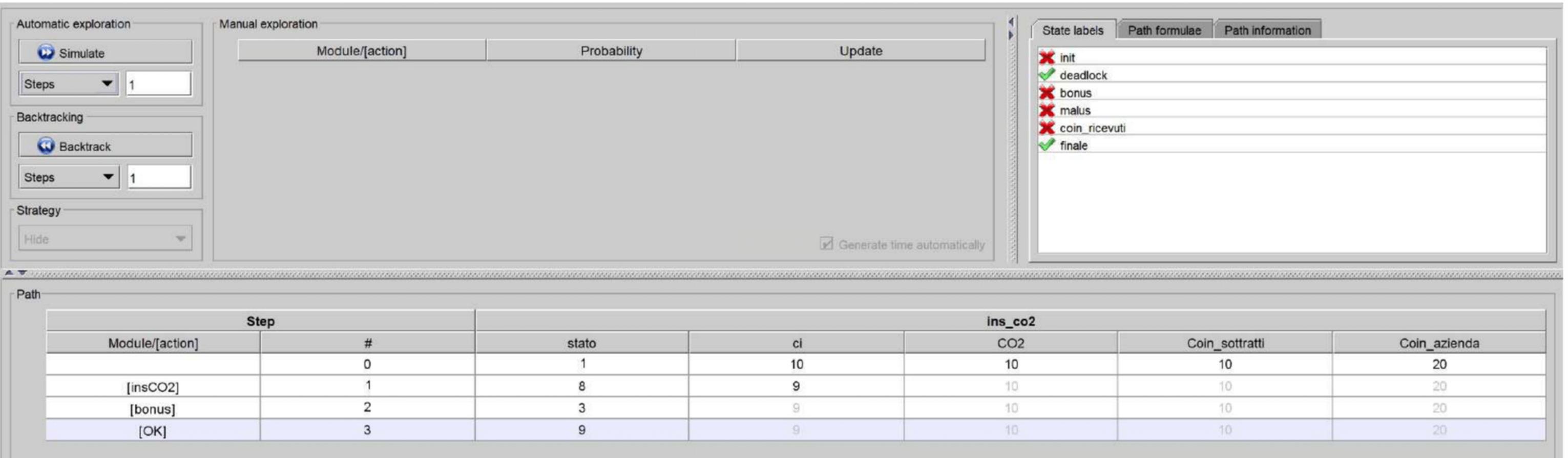
// Risultato della richiesta di coin
[ric_OK] (stato=6) -> (stato'=7); // Coin ricevuti, stato finale di successo

// Coin ricevuti: transizione a stato di successo finale
[OK] (stato=7) -> (stato'=9); // Successo, stato finale raggiunto

endmodule

// Etichette per stato finale
label "bonus" = (stato=3); // Etichetta bonus: stato 3
label "malus" = (stato=4); // Etichetta malus: stato 4
label "coin_ricevuti" = (stato=7); // Etichetta coin ricevuti: stato 7
label "finale" = (stato=9); // Etichetta stato finale di successo
```

Bonus



$\text{CO2} = 10$
 $\text{LimCO2} = 50$

Malus
CO2 = 70
LimCO2 = 50

Coin
sottratti con
successo

The screenshot shows a simulation interface with several panels:

- Automatic exploration:** Contains buttons for "Simulate" and "Backtrack", and dropdowns for "Steps" set to 1.
- Manual exploration:** A table with columns "Module/[action]", "Probability", and "Update".
- State labels:** A list including "init" (red X), "deadlock" (green checkmark), "bonus" (red X), "malus" (red X), "coin_ricevuti" (red X), and "finale" (green checkmark).
- Path:** A detailed table showing the state of variables over time steps 0 to 6. The variables are "stato", "ci", "CO2", "ins_co2", "Coin_sottratti", and "Coin_azienda".

Step		ins_co2				
Module/[action]	#	stato	ci	CO2	Coin_sottratti	Coin_azienda
	0	1	10	70	30	20
[insCO2]	1	8	9	70	30	20
[malus]	2	4	9	70	30	20
[ric_coin]	3	5	9	70	30	20
[attesa]	4	6	9	70	30	20
[ric_OK]	5	7	9	70	30	20
[OK]	6	9	9	70	30	20

Coin non
sufficienti

The screenshot shows a simulation interface with several panels:

- Automatic exploration:** Contains buttons for "Simulate" and "Backtrack", and dropdowns for "Steps" set to 1.
- Manual exploration:** A table with columns "Module/[action]", "Probability", and "Update".
- State labels:** A list including "init" (red X), "deadlock" (green checkmark), "bonus" (red X), "malus" (red X), "coin_ricevuti" (red X), and "finale" (green checkmark).
- Path:** A detailed table showing the state of variables over time steps 0 to 3. The variables are "stato", "ci", "CO2", "ins_co2", "Coin_sottratti", and "Coin_azienda".

Step		ins_co2				
Module/[action]	#	stato	ci	CO2	Coin_sottratti	Coin_azienda
	0	1	10	70	10	20
[insCO2]	1	8	9	70	10	20
[malus]	2	4	9	70	10	20
[OK]	3	9	9	70	10	20

Verifica proprietà di Safety

Tutti i dati devono essere inseriti
correttamente nel sistema

$P =? [F (\text{stato} = 3 \mid \text{stato} = 4)]$

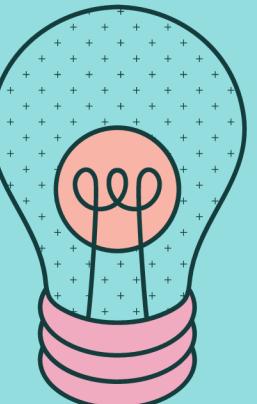
Verifica Response

Ogni volta che viene risparmiata
 CO_2 , l'organizzazione ottiene un
coin

$P =? [G (F (\text{CO}_2 \leq \text{limCO}_2 \& \text{stato} = 9))]$

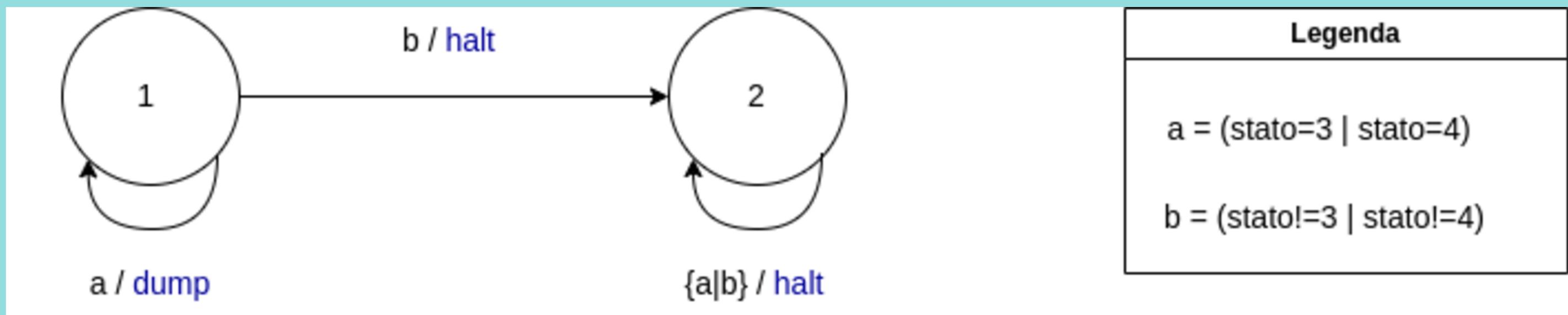
$\text{CO}_2 \leq \text{limCO}_2 = \text{Bonus}$

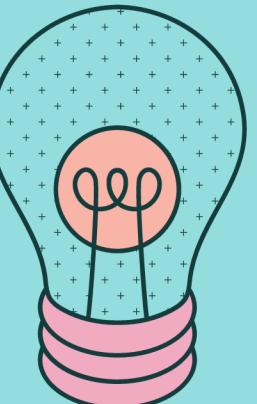
$\text{CO}_2 > \text{limCO}_2 = \text{Malus}$



Esempio di sintesi di un monitor di Runtime Enforcement

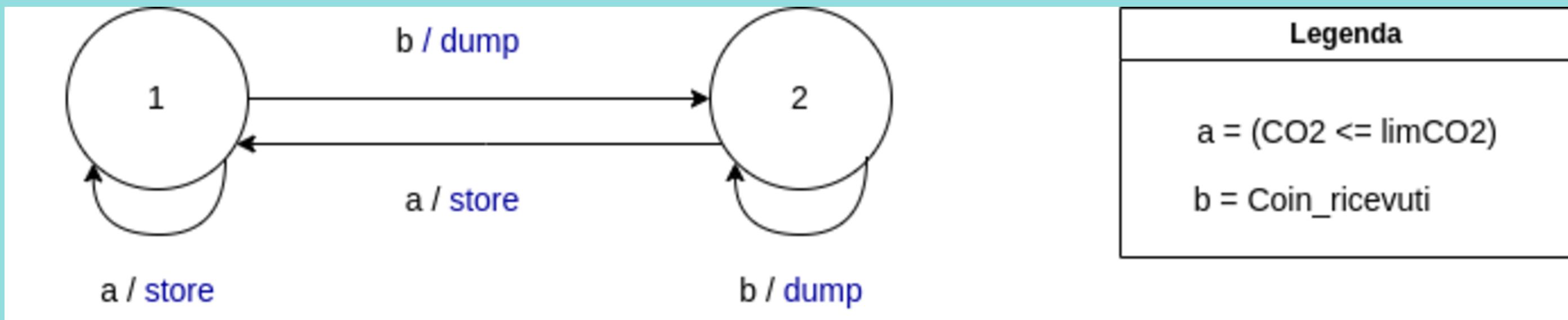
Monitor di RE relativo alla proprietà di Safety

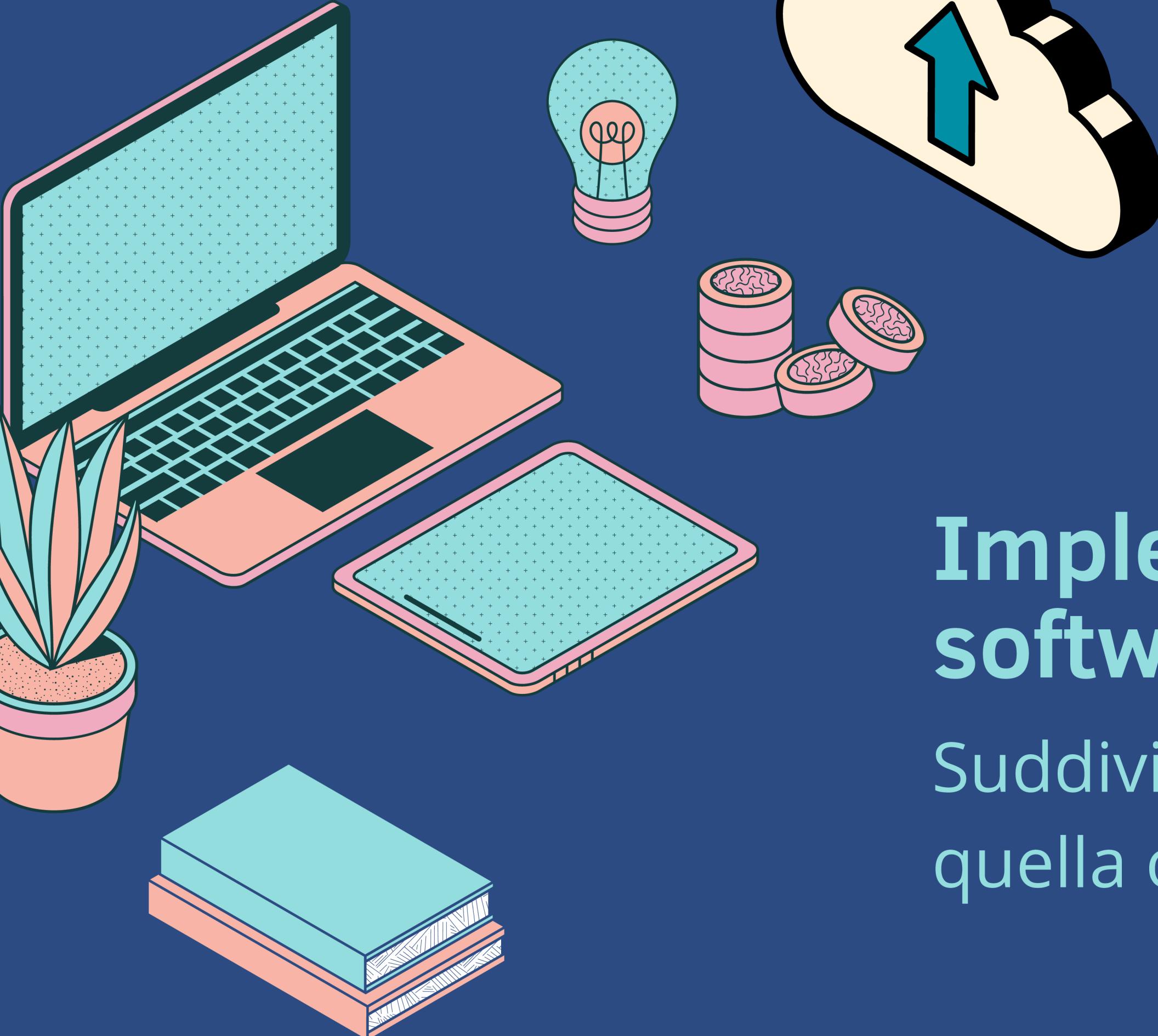




Esempio di sintesi di un monitor di Runtime Enforcement

Monitor di RE relativo alla proprietà di Response

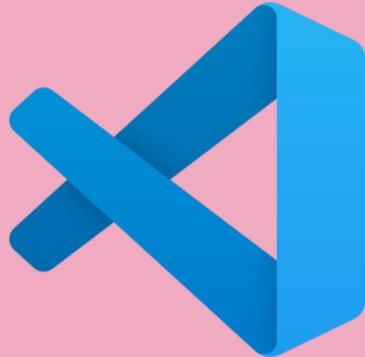
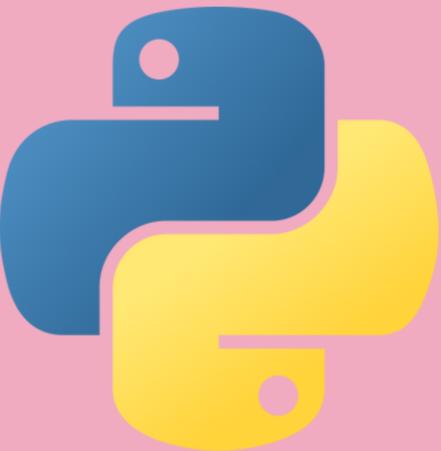




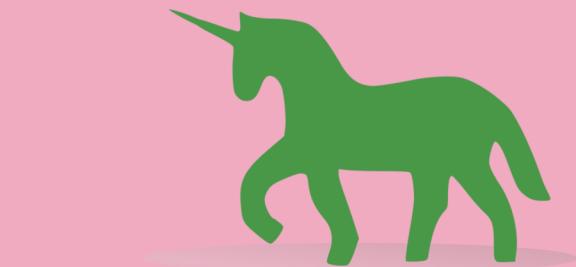
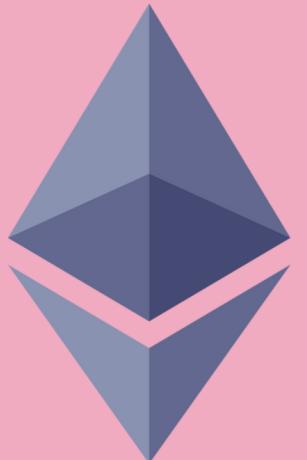
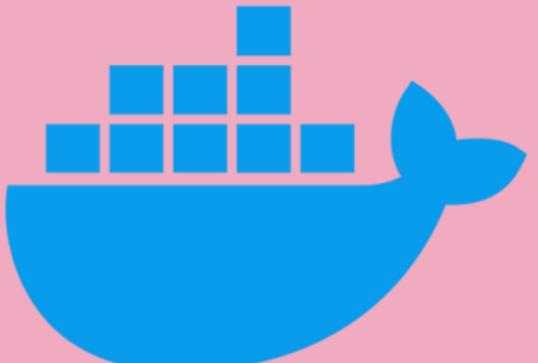
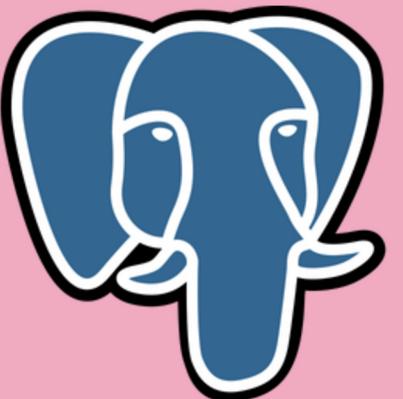
Implementazione del software

Suddividendo la parte off-chain da quella on-chain

Tecnologie utilizzate



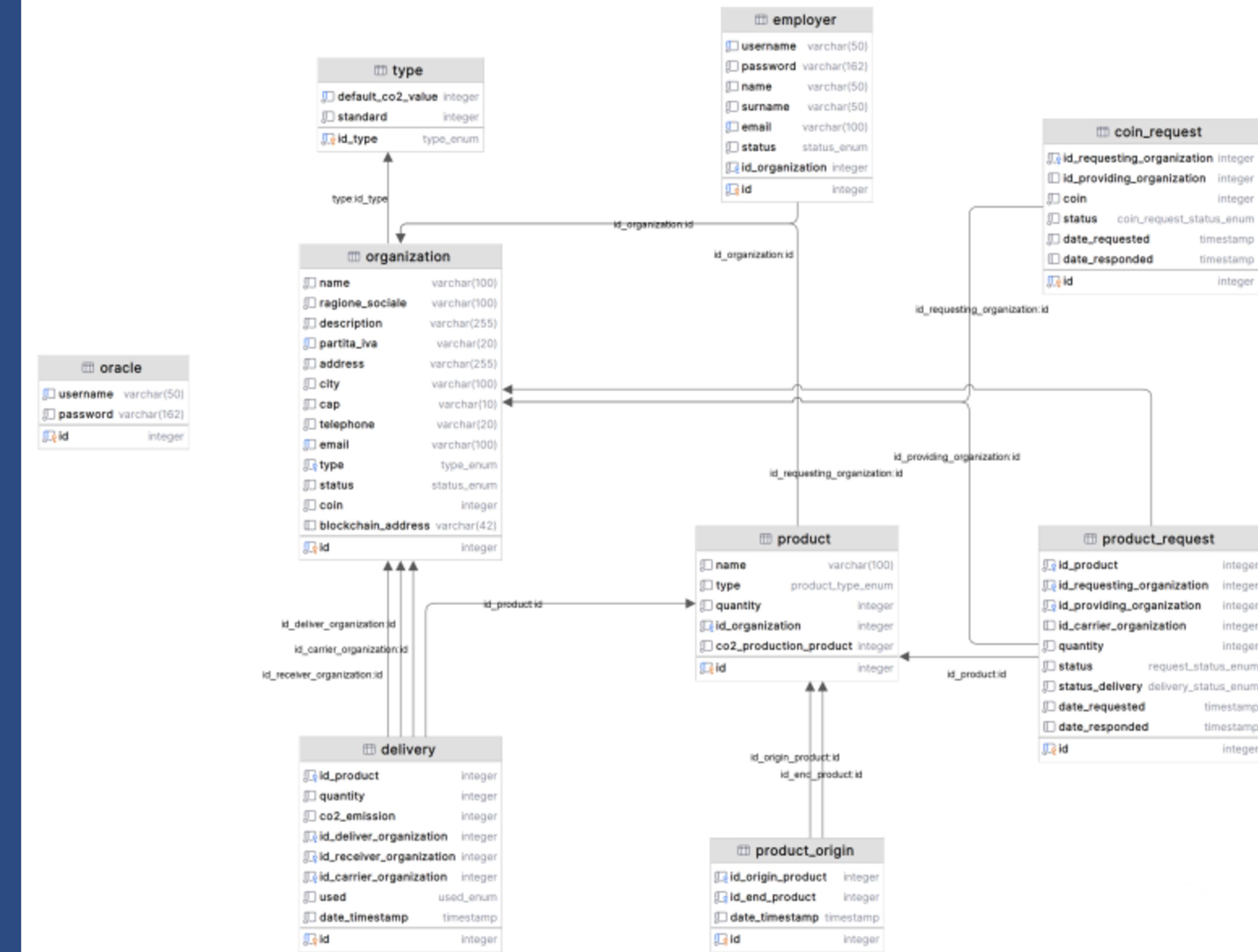
Off-chain



On-chain



Database



Blockchain

Quorum Explorer

The screenshot shows the Quorum Explorer interface running in a browser window titled "Quorum Explorer". The URL is "localhost:25000/explorer/explorer". The top navigation bar includes links for "Nodes", "Validators", "Explorer" (which is active), "Contracts", and "Wallets", along with a gear icon for settings.

The main content area is titled "Explorer". It displays a list of recent blocks under the "Blocks" section:

- Block 241: 0 Transactions, 1 seconds ago. Validator: 0x98c13344...4fed9c.
- Block 240: 0 Transactions, 15 seconds ago. Validator: 0x27a97c9a..da5f18.
- Block 239: 0 Transactions, 20 seconds ago. Validator: 0xce412f98...42d0ca.
- Block 238: 0 Transactions, 25 seconds ago. Validator: 0x98c13344..4fed9c.

Below this is a "Transactions" section. A green button labeled "Transaction" is visible. Two transactions are listed:

- Transaction hash: 0xb737877dd25088a295b982813878ab22e6163414916ae184304be872ef73f255. Block: 237. Hash: 0x6b29267dbb22fb41186bb5a1cf44f69782df54555e938b8e8ad1fcdb5fd7470. Gas: 0x1e8480. From: 0x627306090abab3a6e1400e9345bc60c78a8bef57.
- Transaction hash: 0xbb5f219e9abcbaed79f61e4894dd29006644aa6efa6a4f5f9e7a979ecc28870. Block: 224. Hash: 0x2ff6164066ac3b079a43608d4e247b70dc4717b55cdfffeb13391a5ebd3fca26. Gas: 0x1e8480. From: 0x627306090abab3a6e1400e9345bc60c78a8bef57.

A search bar for transactions is also present in the "Transactions" section.

Blockchain

Quorum Explorer Nodes

The screenshot shows the Quorum Explorer Nodes interface in a web browser. The title bar indicates the window is titled "Quorum Explorer" and the address bar shows "localhost:25000/explorer/nodes". The main navigation menu includes "Nodes" (selected), "Validators", "Explorer", "Contracts", and "Wallets". A gear icon for settings is also present.

The "Nodes" section displays the following metrics:

- Status: Running
- Blocks: 237
- Peers: 7
- Queued: 0

A detailed configuration panel for the current node is shown, listing the following parameters:

Client:	besu
Node ID:	86fcc16f4730fbfd238dc17ea552854c0943923bb1d5e886e5601b8d884fb0519060e0023f495dd24ffe60a65660fb7fdcdebfc eedd2b3673dfa63658825924b
Node Name:	besu/v23.4.1/linux-x86_64/openjdk-java-17
Enode:	enode://86fcc16f4730fbfd238dc17ea552854c0943923bb1d5e886e5601b8d884fb0519060e0023f495dd24ffe60a65660fb7fdcdebfc eedd2b3673dfa63658825924b@172.16.239.15:30303
RPC Url:	http://rpcnode:8545
IP Address:	172.16.239.15

At the bottom of the page, there is a "Peer Information" section which is currently empty.

- Gestione dei Saldi delle Organizzazioni
 - Il mapping balances associa ogni indirizzo di un'organizzazione al proprio saldo in "monete" digitali.
 - Le funzioni *updateCoins* e *updateCoinsBasedOnEmission* permettono di modificare il saldo in base a transazioni o emissioni di CO₂.
 - La funzione *transferCoins* consente di trasferire monete tra organizzazioni.
- Registrazione delle Transazioni
 - Ogni operazione viene registrata in un array transactions con dettagli come indirizzo, importo, timestamp e hash della transazione.
 - Le transazioni rifiutate vengono salvate in *rejectedTransactions* con informazioni aggiuntive come il motivo del rifiuto, il nome del prodotto e le emissioni di CO₂.

- **Monitoraggio dell'Origine dei Prodotti**
 - La struttura *ProductOriginTransaction* tiene traccia della trasformazione dei prodotti attraverso ID di origine e destinazione.
 - La funzione *registerProductOrigin* registra tali trasformazioni e ne emette un evento.
- **Eventi per il Tracciamento delle Modifiche**
 - *CoinUpdated*: notificato quando cambia il saldo di un'organizzazione.
 - *ProductOriginRegistered*: notificato quando viene registrata l'origine di un prodotto.
 - *TransactionRejected*: notificato quando una transazione viene rifiutata.

- Recupero delle Informazioni
 - *getBalance*: restituisce il saldo attuale di un'organizzazione.
 - *getTransactions*: permette di ottenere tutte le transazioni associate a un'organizzazione.
 - *getProductOriginTransactions*: fornisce dettagli sulle transazioni relative all'origine dei prodotti.

Smart contract

deploy.py

- Caricamento delle variabili d'ambiente:
 - La libreria dotenv carica le variabili d'ambiente dal file .env. In particolare, vengono utilizzate le informazioni relative all'indirizzo del nodo blockchain (BLOCKCHAIN_URL), l'indirizzo dell'amministratore (ADMIN_ADDRESS), e la chiave privata dell'amministratore (ADMIN_PRIVATE_KEY).
- Connessione alla blockchain:
 - Utilizza Web3 per connettersi alla blockchain tramite l'URL del nodo Ethereum (o POA - Proof of Authority) passato come variabile d'ambiente.

Smart contract

deploy.py

- **Compilazione del contratto Solidity:**
 - Se il contratto non è già stato compilato, la funzione `compile_contract` legge il codice Solidity da un file (`CoinContract.sol`), lo compila usando `solcx` e salva il codice compilato in un file JSON.
- **Caricamento del contratto compilato:**
 - Dopo che il contratto è stato compilato, il file JSON viene letto e vengono estratti l'ABI e il bytecode del contratto. Questi vengono poi utilizzati per creare un oggetto contratto in Web3, che può essere usato per interagire con il contratto distribuito sulla blockchain.

Smart contract

deploy.py

- Distribuzione del contratto sulla blockchain:
 - La funzione *deploy_contract* distribuisce il contratto sulla blockchain se non è già stato distribuito. Controlla se esiste un file che contiene l'indirizzo del contratto (*contract_address.txt*), e se non esiste, costruisce una transazione per il deploy del contratto.
 - La transazione viene firmata con la chiave privata e inviata alla rete. Dopo che la transazione è confermata, l'indirizzo del contratto viene salvato in un file di testo.
- Recupero del contratto da un indirizzo:
 - La funzione *get_contract* permette di ottenere un'istanza del contratto dato un indirizzo di contratto già distribuito. Viene caricato l'ABI dal file compilato e il contratto viene restituito utilizzando l'indirizzo e l'ABI.

Grazie per l'attenzione

