Comprendre la blockchain

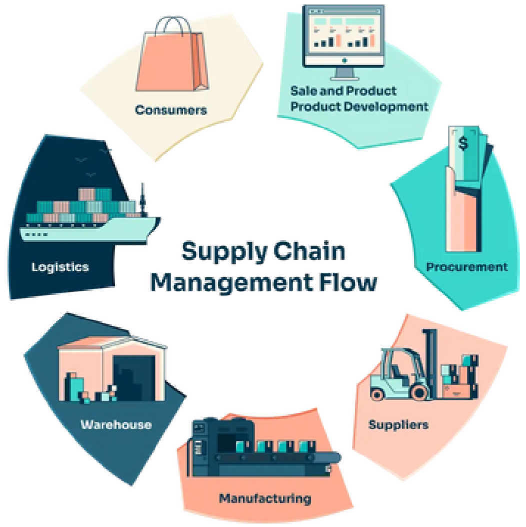
Blockchain and supply chain

by

Sofia Bellotti - sofia.bellotti@student-cs.fr
Beltran Bulbarella - beltran.bulbarella@student-cs.fr
Sara Bonino - sara.bonino@student-cs.fr
Lluc Lozano - lluc.lozano@student-cs.fr
Julia Pareto - julia.malhaes-pareto@student-cs.fr
Jorge Varadé - jorge.varade@student-cs.fr

Using blockchain in a Supply chain

Supply chains contain complex networks with a lot a parties



→ Blockchain simplifies workflows for all parties, regardless of network size, and gives auditors better visibility into activities across the value chain.

Goals:

- **Transparency –** register important information
- **Traceability –** improves operational efficiency
- **Tradeability –** transfer ownership without the physical asset changing hands.

Source: consensys.io

Benefits for the supply chain operation

01

Enhanced Security

- Role-based access and verified updates.
- Reduce fraud and counterfeiting while ensuring data integrity.

02

Increased Efficiency

- Automate processes, provide real-time updates, and minimize disputes.
- Save time by reducing manual work and streamlining operations.

03

Cost Savings & Consumer Trust

- Improve inventory management and build consumer confidence through product authenticity.

04

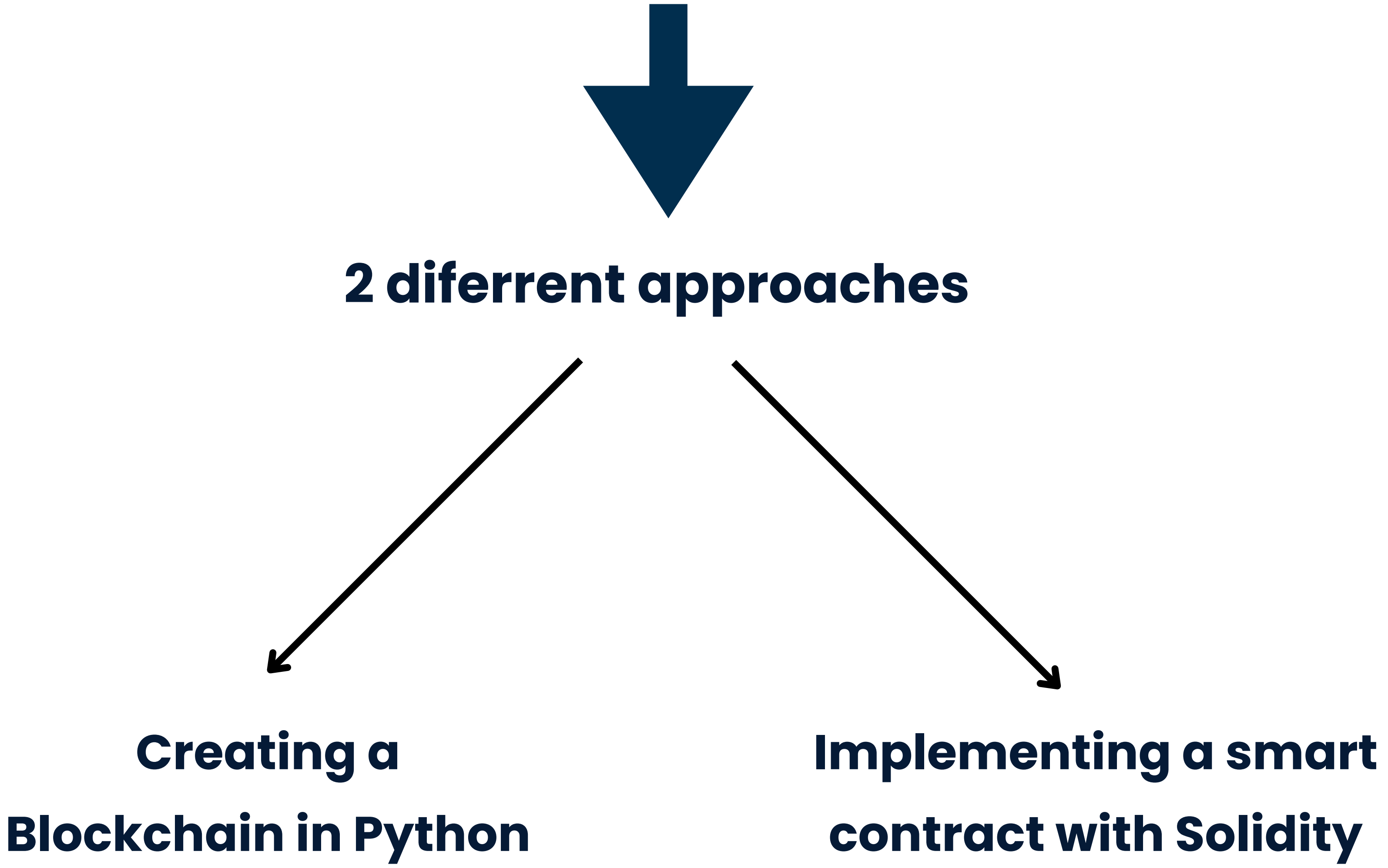
Scalability and Future-Ready Solutions

- Integrate with IoT devices for real-time updates based on real-world conditions.

PROOF OF CONCEPT (POC)

Objective:

Creation of a blockchain to accurately verify every step of the product's process, from its creation to its delivery to the customer, ensuring optimal traceability.



2 diferrent approaches

Creating a Blockchain in Python

Implementing a smart contract with Solidity

Supply Chain Blockchain

Roles

- **SUPPLIER –** Creates new products.
- **MANUFACTURER –** Updates product status to "Manufactured."
- **LOGISTICS PROVIDER –** Updates product status to "In Transit."
- **RETAILER –** Updates product status to "Available for Sale."
- **CONSUMER –** Scans QR codes to verify product authenticity and trace its journey.

Data structures

Product:

Represents each item in the supply chain.

Transaction History:

Logs all status updates and transfers of ownership.

Approach 1

Blockchain implementation in Python

Key Python Libraries:

- hashlib: For creating secure hashes.
- ecdsa: For signing transactions and verifying signatures.
- json: For managing transaction data.

Python in Blockchain Development

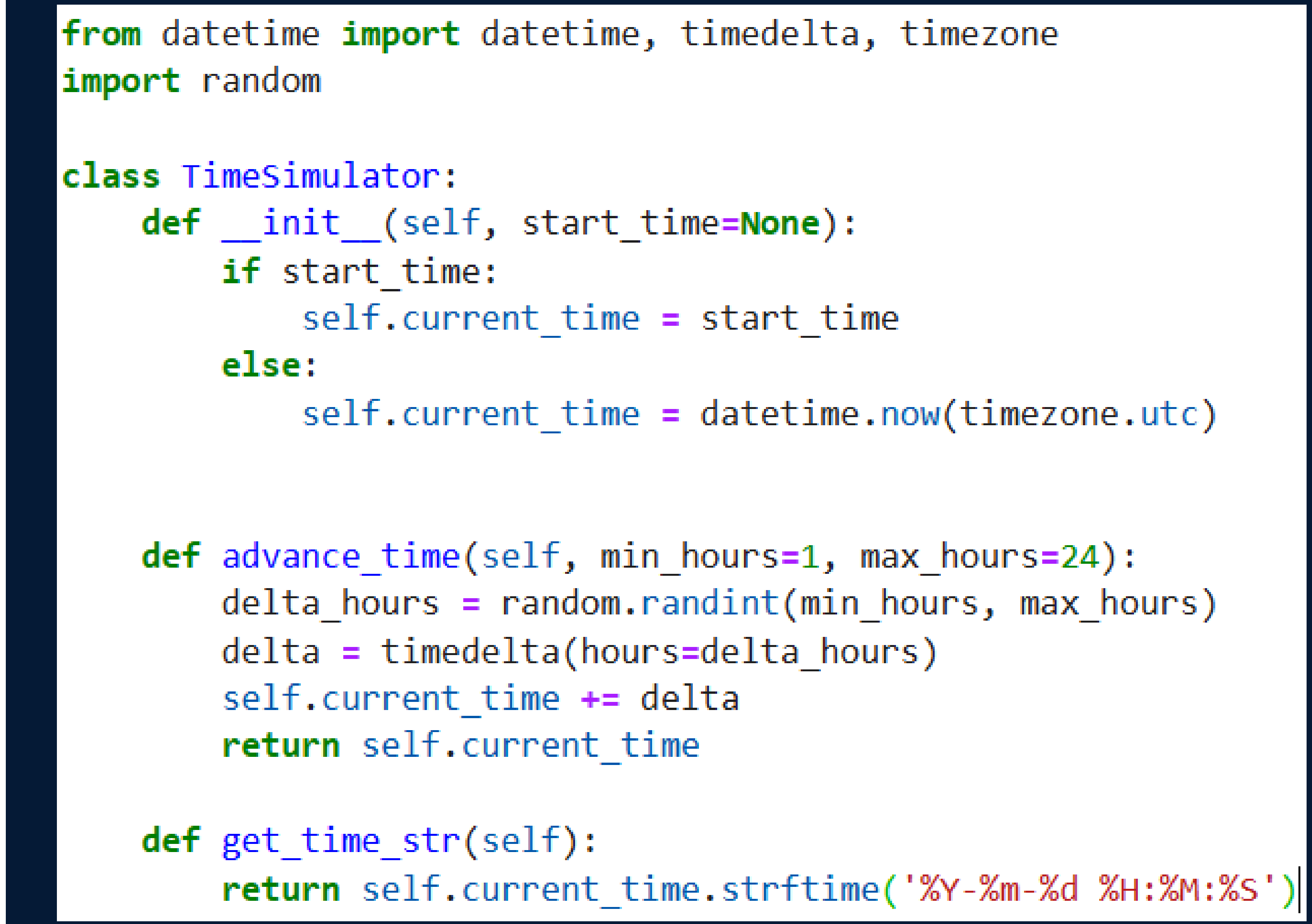
Why Use Python for Blockchain?

- Easy to implement
- Has robust libraries for cryptography and blockchain.
- Ideal for prototyping and deployment.
- Use of Object Oriented Programming

Blockchain Implementation

Modules

- utils
- config
- transaction
- encrypt_data
- block
- blockchain



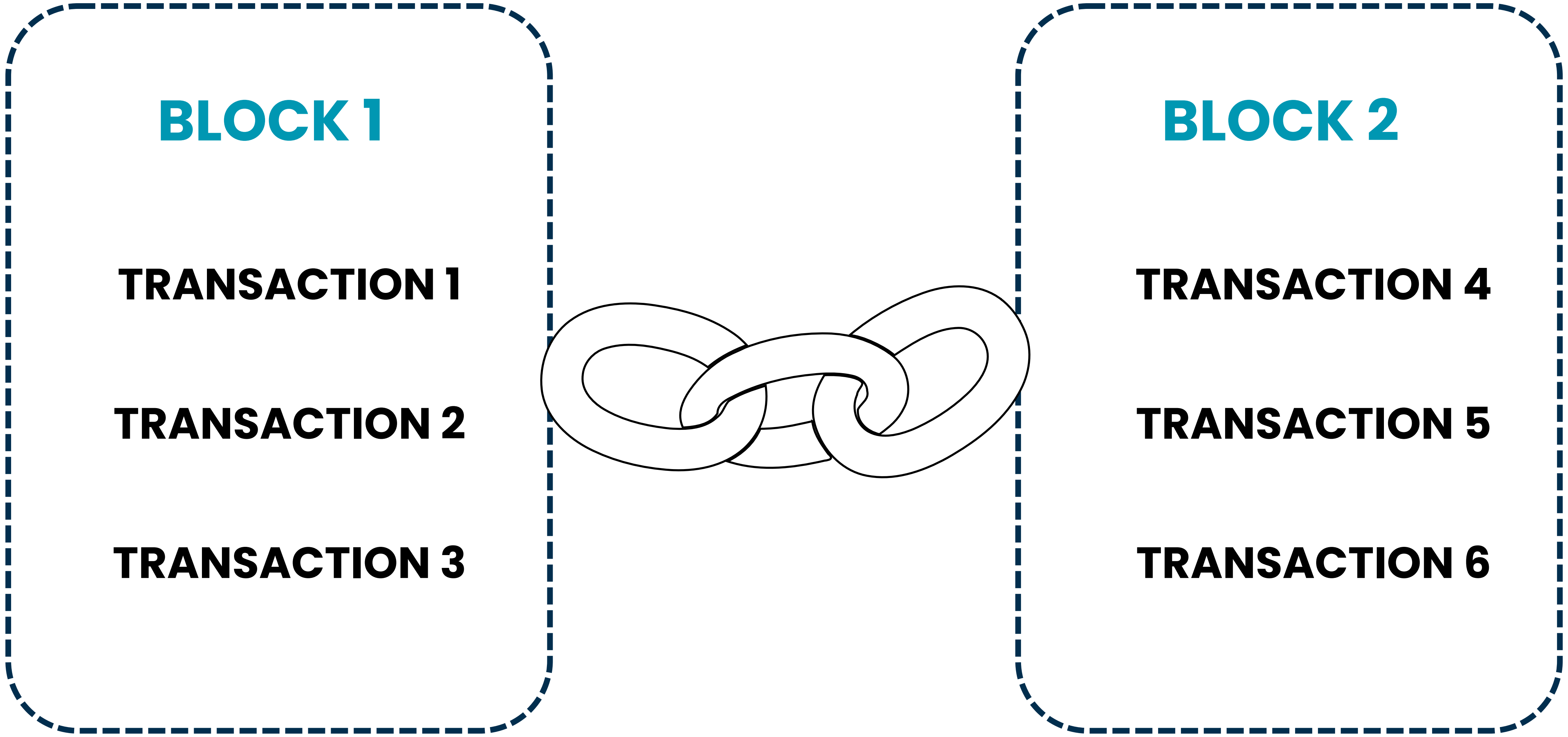
```python
from datetime import datetime, timedelta, timezone
import random

class TimeSimulator:
    def __init__(self, start_time=None):
        if start_time:
            self.current_time = start_time
        else:
            self.current_time = datetime.now(timezone.utc)


    def advance_time(self, min_hours=1, max_hours=24):
        delta_hours = random.randint(min_hours, max_hours)
        delta = timedelta(hours=delta_hours)
        self.current_time += delta
        return self.current_time

    def get_time_str(self):
        return self.current_time.strftime('%Y-%m-%d %H:%M:%S')
```

Block structure



BLOCK 1

TRANSACTION 1

TRANSACTION 2

TRANSACTION 3

BLOCK 2

TRANSACTION 4

TRANSACTION 5

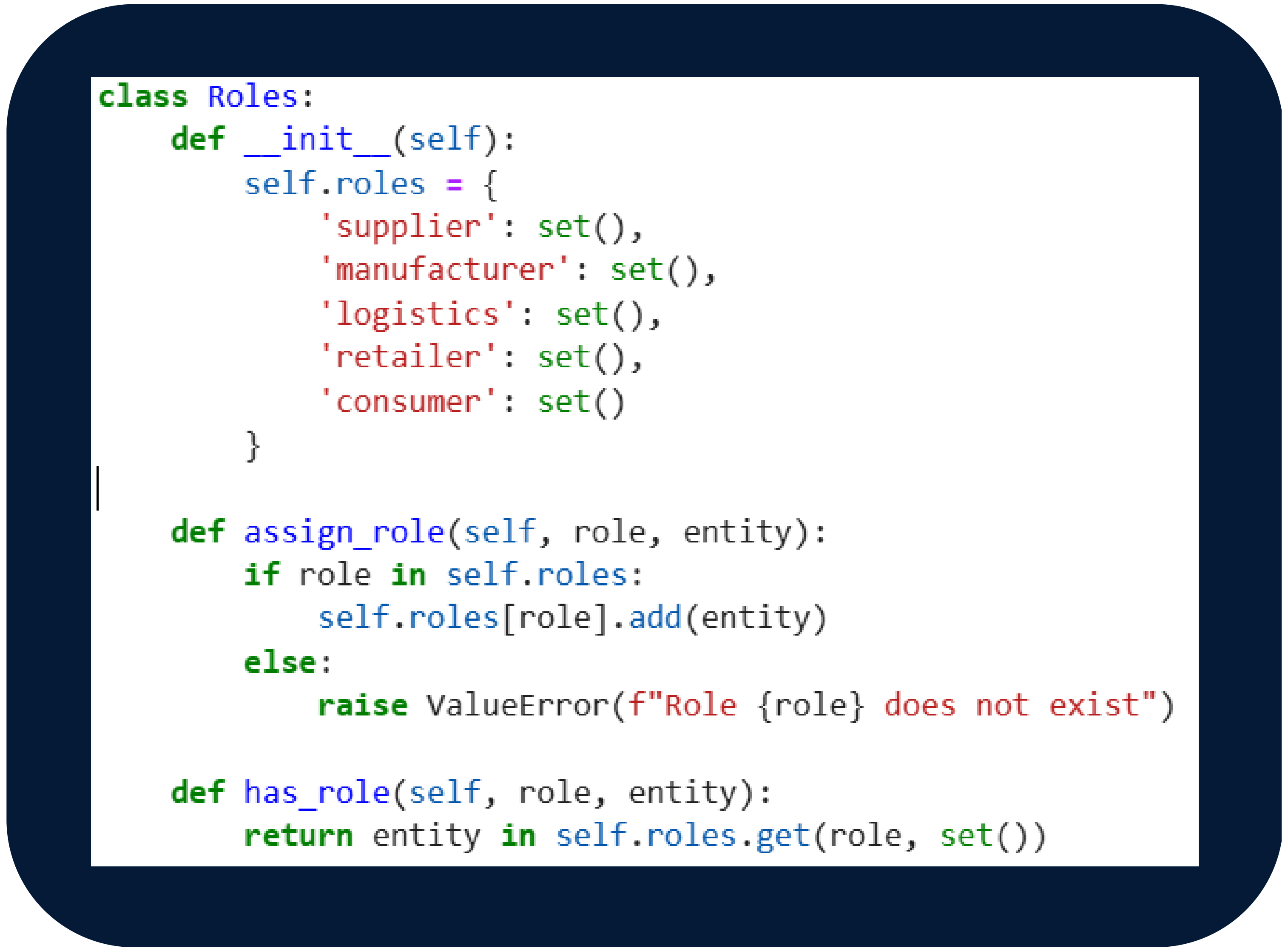
TRANSACTION 6

Supply Chain

Modules

- Product
- Roles
- Supply_chain

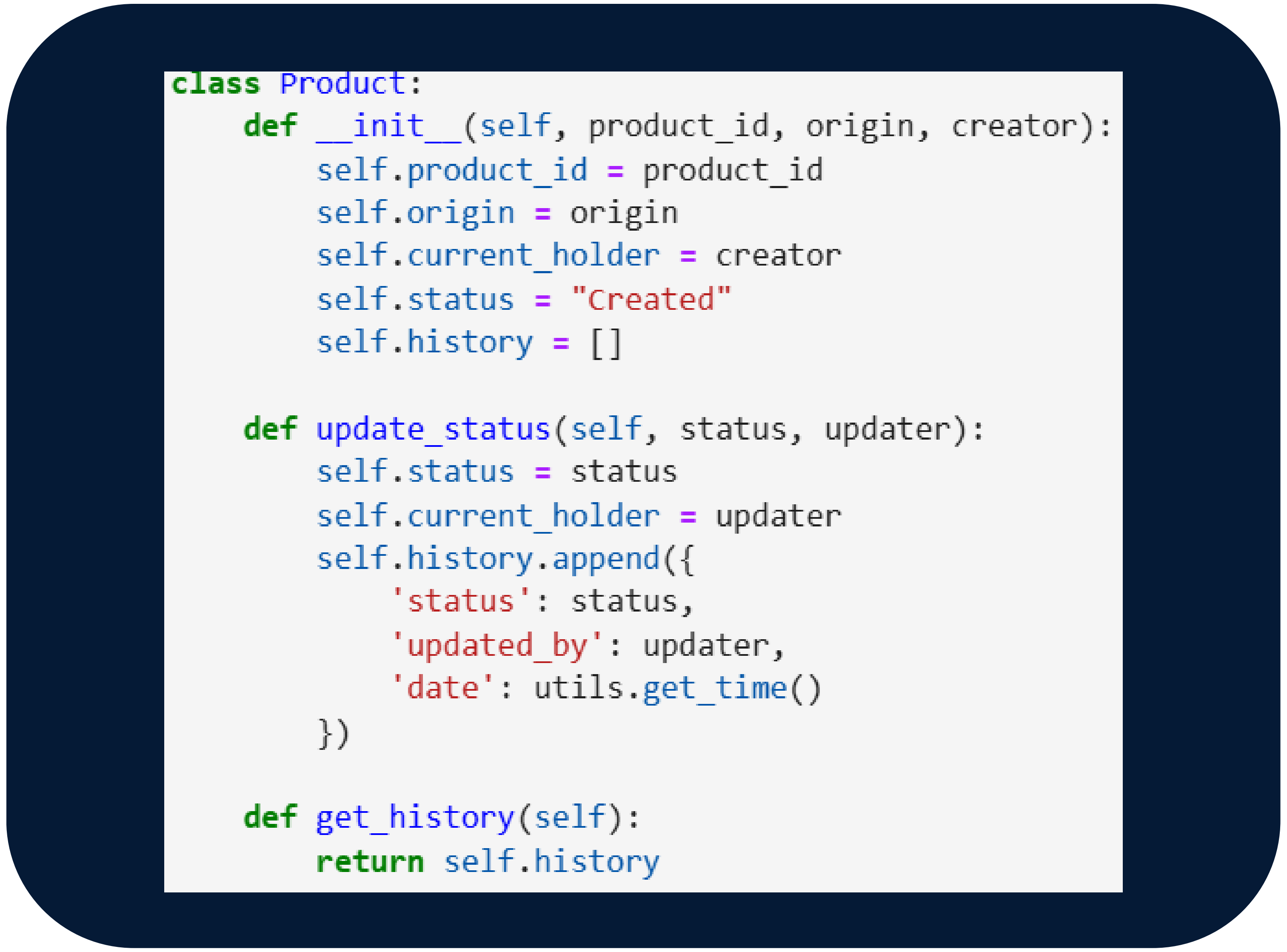
- Simulation
- Visualization
- App



```
class Roles:
    def __init__(self):
        self.roles = {
            'supplier': set(),
            'manufacturer': set(),
            'logistics': set(),
            'retailer': set(),
            'consumer': set()
        }

    def assign_role(self, role, entity):
        if role in self.roles:
            self.roles[role].add(entity)
        else:
            raise ValueError(f"Role {role} does not exist")

    def has_role(self, role, entity):
        return entity in self.roles.get(role, set())
```



```
class Product:
    def __init__(self, product_id, origin, creator):
        self.product_id = product_id
        self.origin = origin
        self.current_holder = creator
        self.status = "Created"
        self.history = []

    def update_status(self, status, updater):
        self.status = status
        self.current_holder = updater
        self.history.append({
            'status': status,
            'updated_by': updater,
            'date': utils.get_time()
        })

    def get_history(self):
        return self.history
```

Simulation

Objective of the Simulation:

- Create a model of product flow within a supply chain.
- Simulate random events that reflect the movement of products between different actors in the supply chain.

Simulated Events:

Product creation.

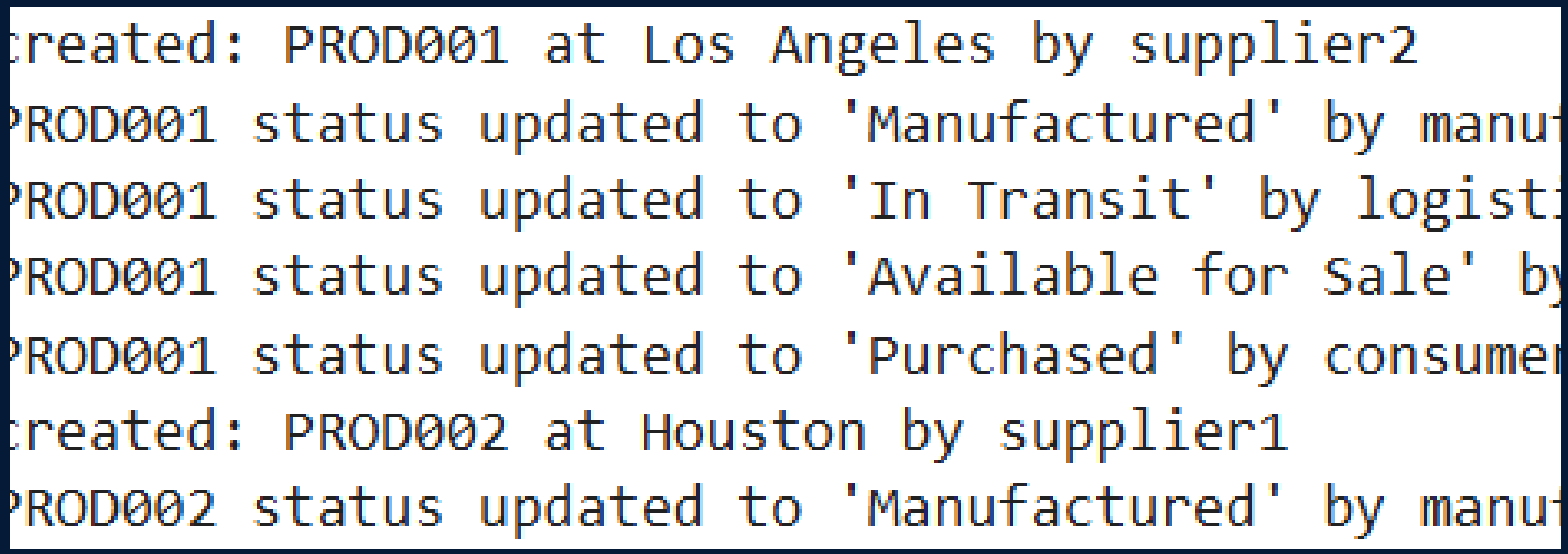
Status update.

Purchase by the end consumer.

Time-based Simulation:

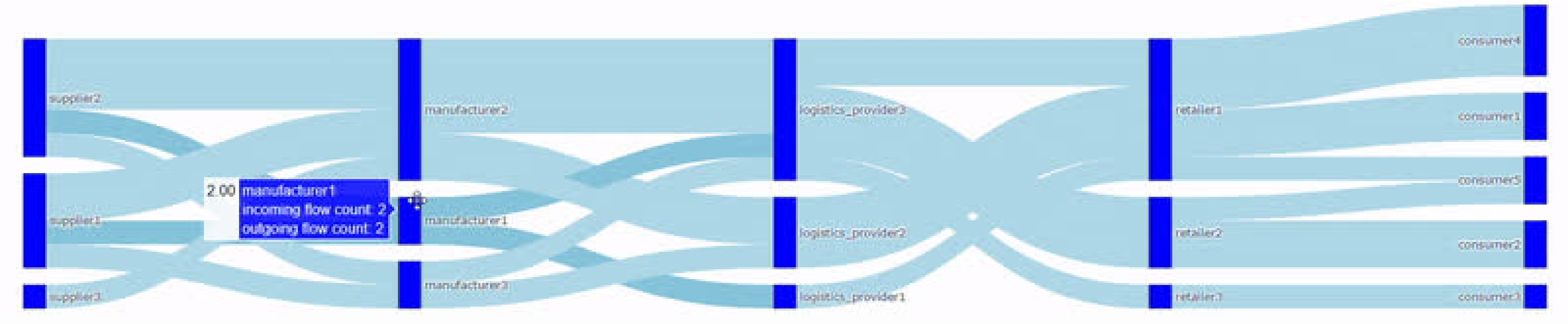
- **Each event occurs in a random time interval (between 1 and 24 hours).**
- **Each step between actors is determined by random distribution.**

Randomness: Events are determined through random values to simulate unpredictable flow.



```
created: PROD001 at Los Angeles by supplier2
PROD001 status updated to 'Manufactured' by manu
PROD001 status updated to 'In Transit' by logisti
PROD001 status updated to 'Available for Sale' by
PROD001 status updated to 'Purchased' by consumer
created: PROD002 at Houston by supplier1
PROD002 status updated to 'Manufactured' by manu
```

Product flow through the supply chain

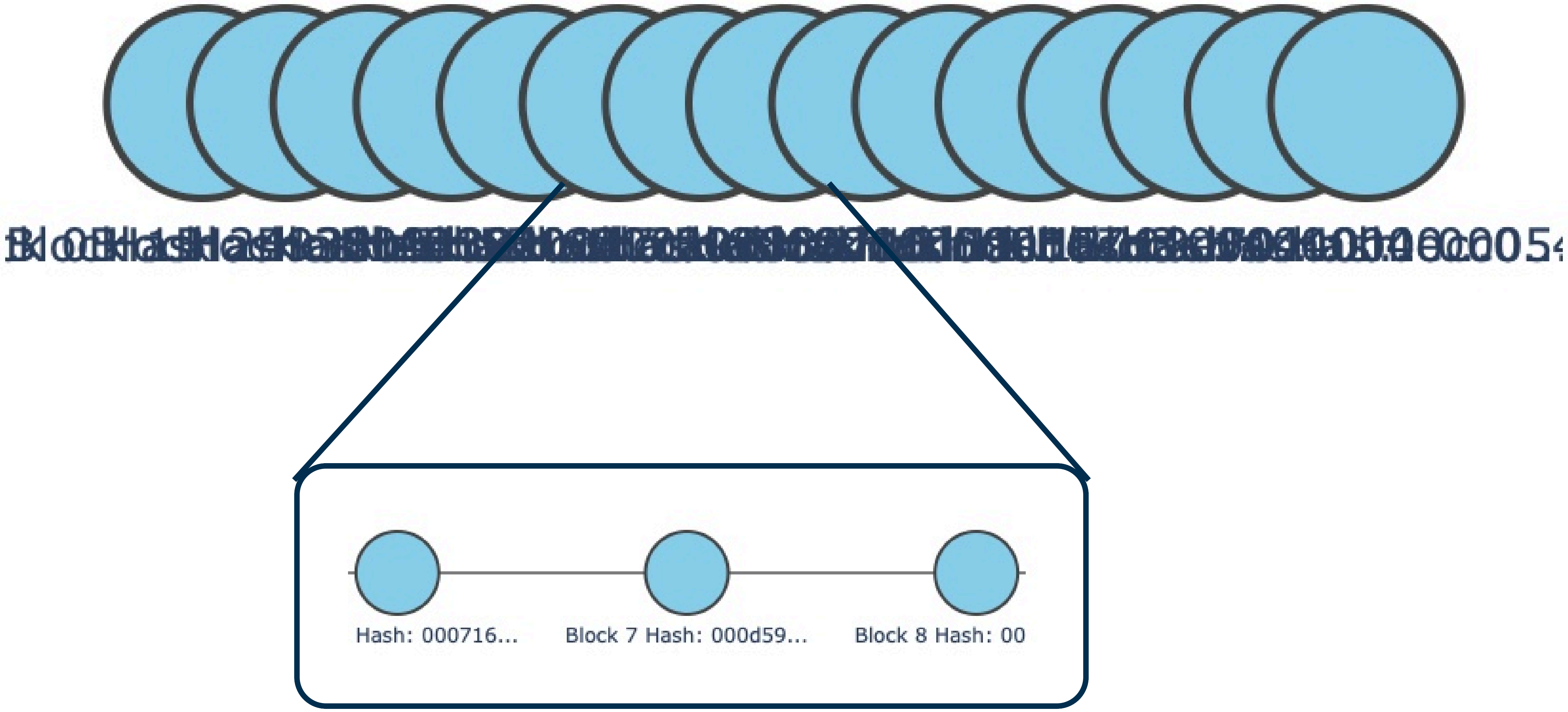


Suppliers **Manufacturers** **Logistics** **Retailers** **Consumers**

This is an example from a random simulation with

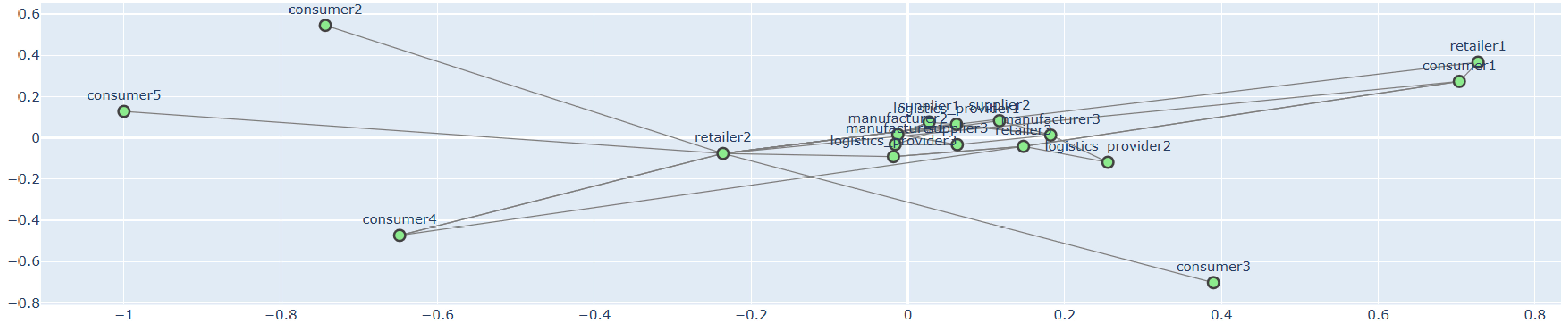
- 3 entities for each role in the chain;
- 5 clients.

Blockchain structure



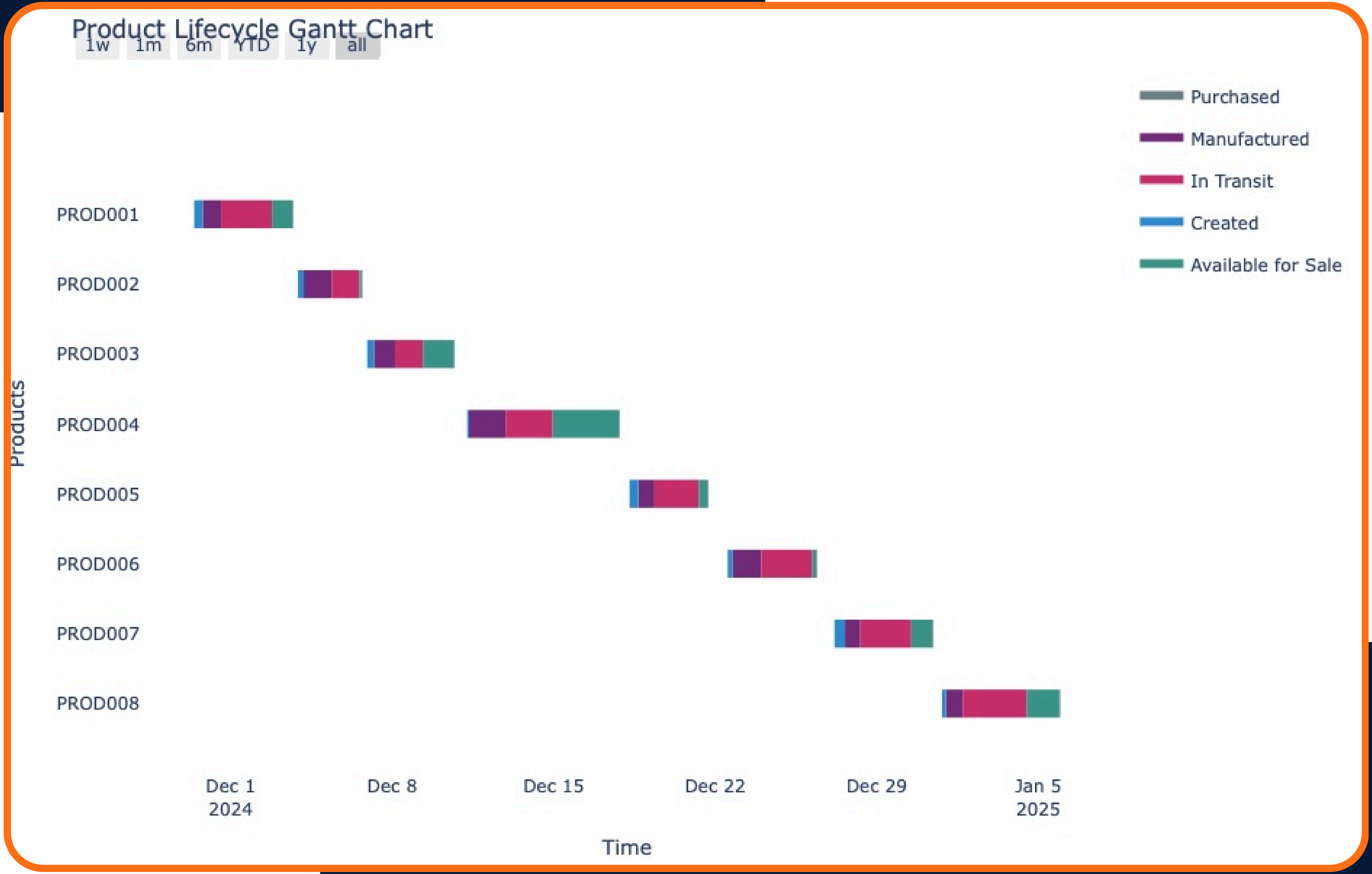
Linear structure

Network



consumer2
retailer1
consumer1
consumer5
supplier1
supplier2
logistics_provider1
manufacturer2
manufacturer3
manufacturer1
supplier3
retailer3
retailer2
logistics_provider3
logistics_provider2
consumer4
consumer3
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8

GANTT chart


Product Lifecycle Gantt Chart
1w
1m
6m
YTD
1y
all
Purchased
Manufactured
In Transit
Created
Available for Sale
Products
PROD001
PROD002
PROD003
PROD004
PROD005
PROD006
PROD007
PROD008
Dec 1
2024
Dec 8
Dec 15
Dec 22
Dec 29
Jan 5
2025
Time

Approach 2

Blockchain with Solidity

What is Solidity?

Object-oriented programming language specifically **designed for creating smart contracts for the Ethereum blockchain.**

Ethereum blockchain:

“the world’s programmable blockchain”

Solidity is therefore a contract-oriented language, which means that **smart contracts** are responsible for storing all of the **programming logic that transacts with the blockchain.**

Source: dappuniversity, investopedia

Methodology

Supply chain structure

The first step of the development was to determine the requisites and process of the supply chain we aimed to represent through the blockchain.

Smart contrat structure

We then transformed the requirements into the smart contract structure, establishing the requirements for each role and for each new added transaction.

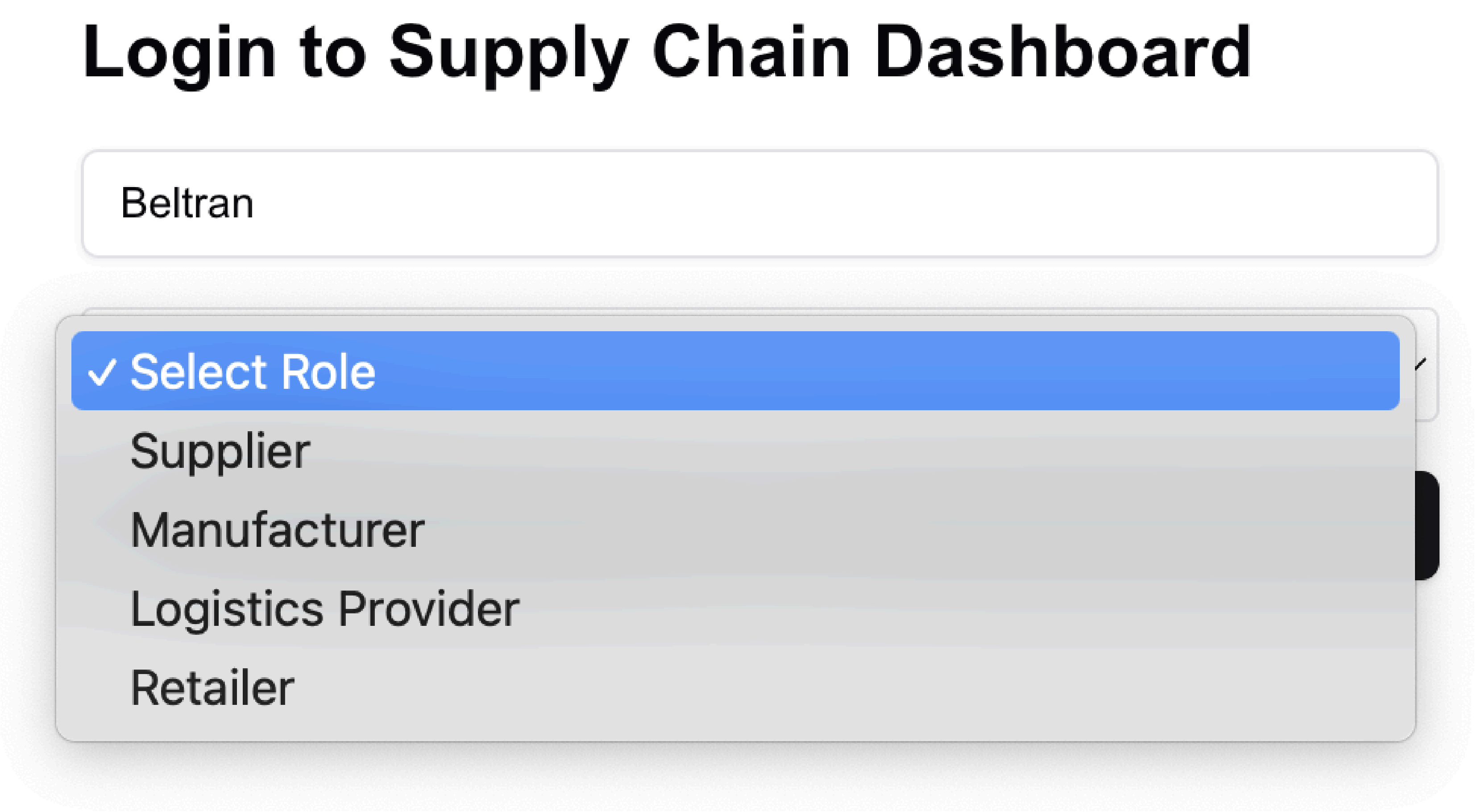
Platform development

We used HTML and Java to develop the frontend of our structure. The platform is used to allow the users to interact with the supply shain steps with specific roles.

Role selection

The user chooses his role in the supply chain. In the example, Beltran is a supplier.

Role-Based Access Control: the contract enforces strict **role-based permissions**, ensuring that only authorized entities can perform specific actions.



Login to Supply Chain Dashboard

Beltran

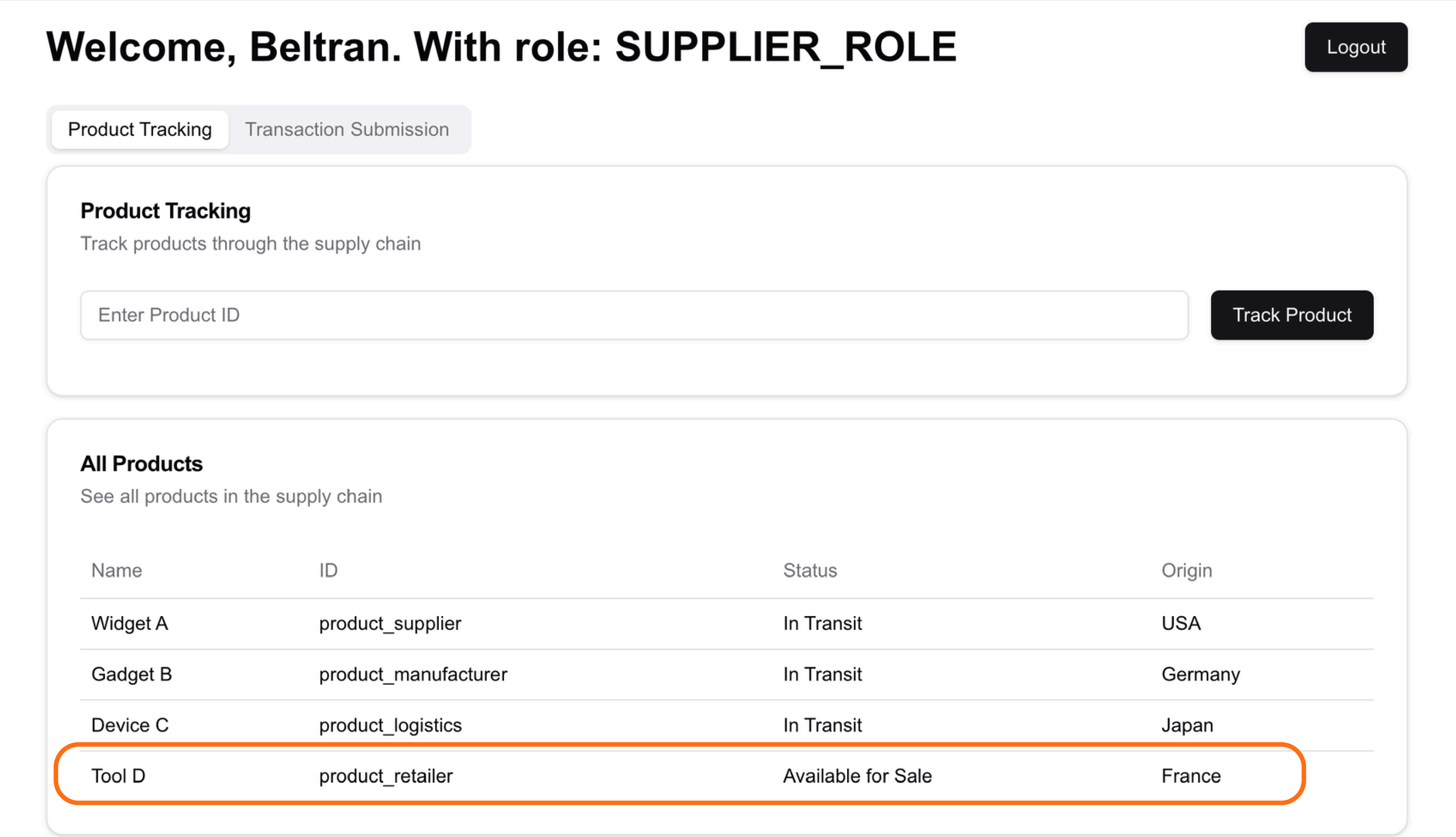
✓ Select Role
Supplier
Manufacturer
Logistics Provider
Retailer

Product tracking

Beltran can:

- Track a product;
- Submit a transaction of a product he owns.

He can see all the products in the supply chain and their status.



Welcome, Beltran. With role: SUPPLIER_ROLE

Product Tracking

Track products through the supply chain

All Products

See all products in the supply chain

Name	ID	Status	Origin
Widget A	product_supplier	In Transit	USA
Gadget B	product_manufacturer	In Transit	Germany
Device C	product_logistics	In Transit	Japan
Tool D	product_retailer	Available for Sale	France

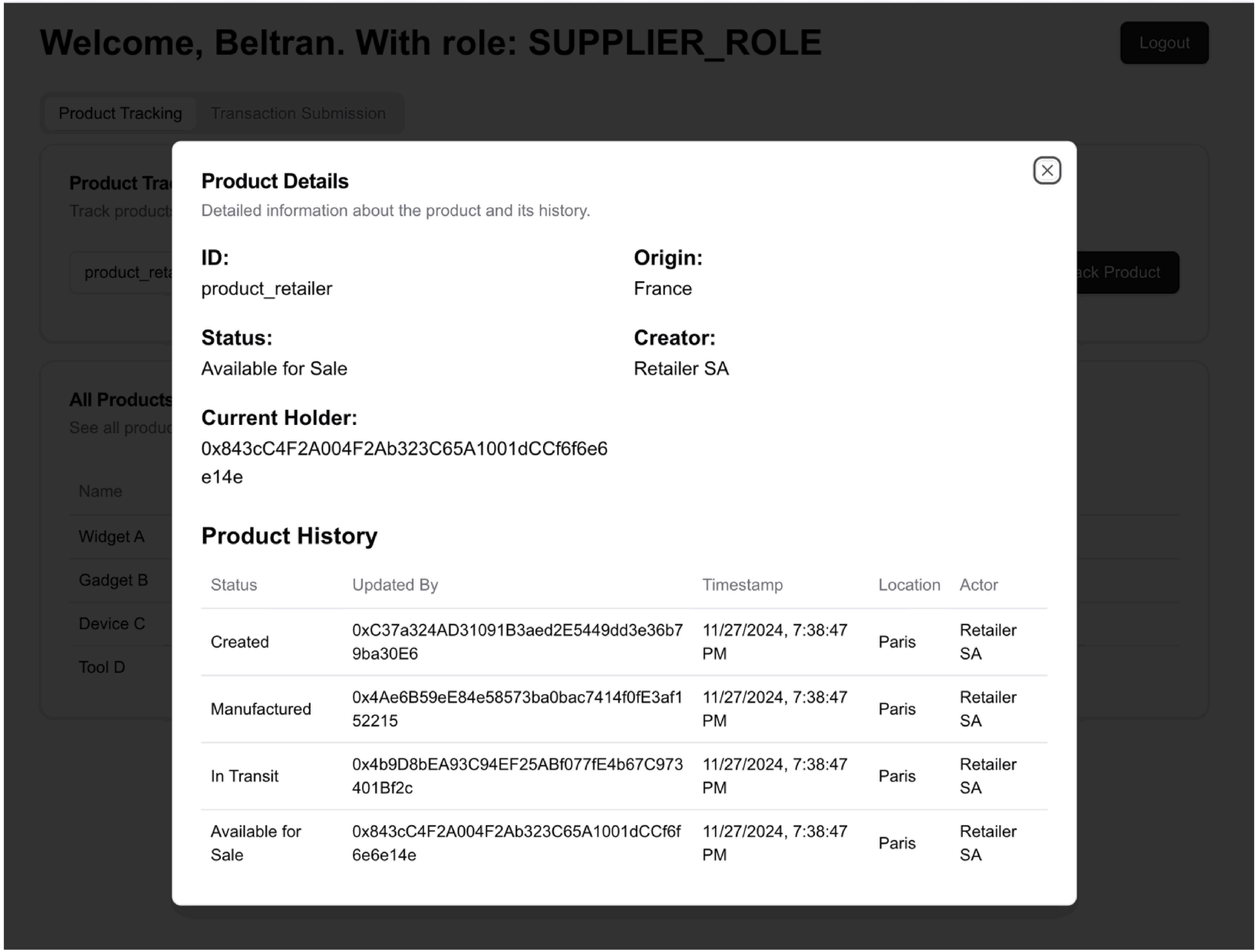
What if Beltran clicks on this product?

Solidity UI

Product tracking

He can click on every product to see the details and history about it.

Every product has **traceability at every step**. This makes it transparent, traceable, inmutable, secure and decentralized.



Product Details

Detailed information about the product and its history.

ID:
product_retailer

Origin:
France

Status:
Available for Sale

Creator:
Retailer SA

Current Holder:
0x843cC4F2A004F2Ab323C65A1001dCCf6f6e6e14e

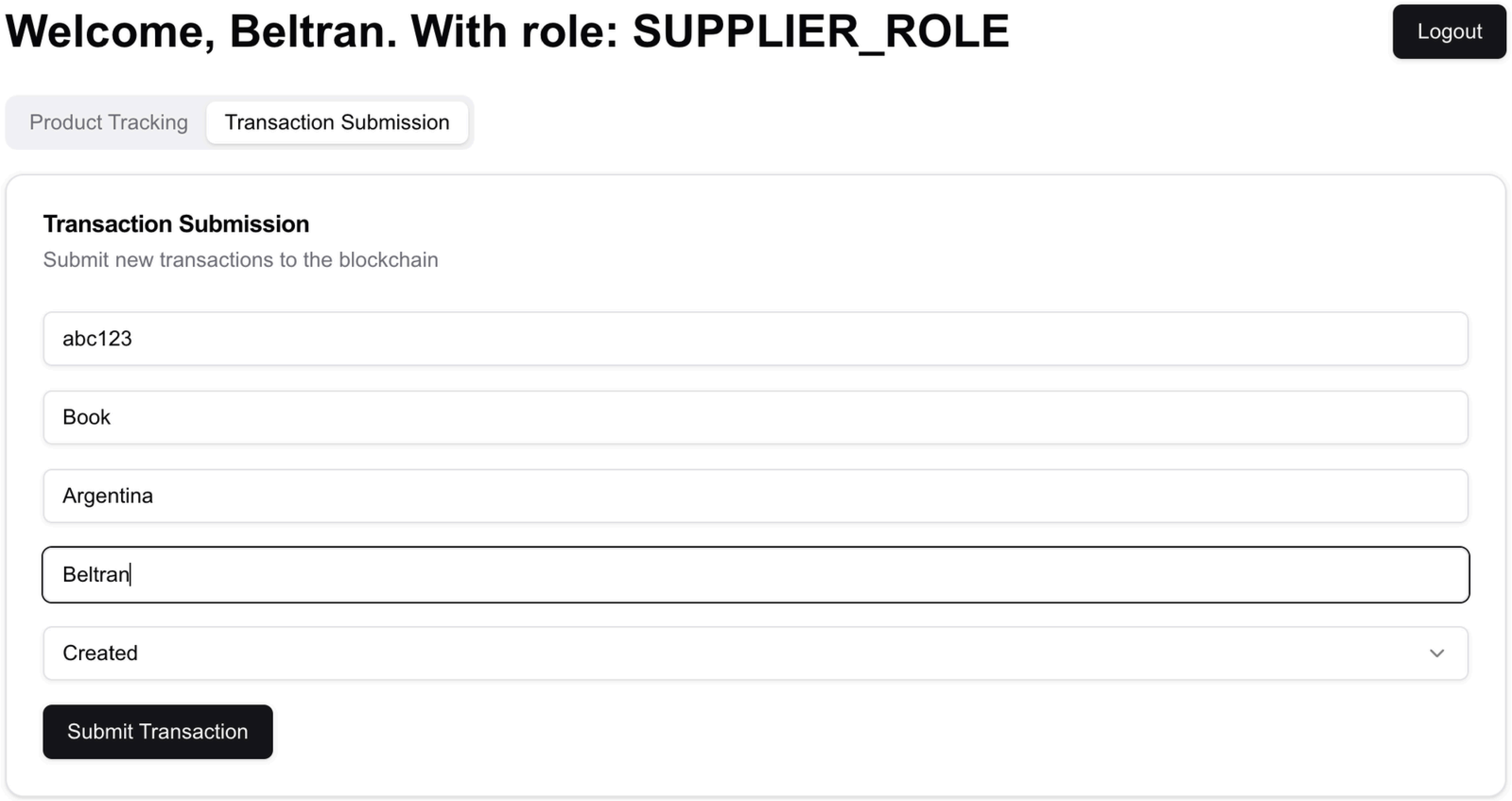
Product History

Status	Updated By	Timestamp	Location	Actor
Created	0xC37a324AD31091B3aed2E5449dd3e36b79ba30E6	11/27/2024, 7:38:47 PM	Paris	Retailer SA
Manufactured	0x4Ae6B59eE84e58573ba0bac7414f0fE3af152215	11/27/2024, 7:38:47 PM	Paris	Retailer SA
In Transit	0x4b9D8bEA93C94EF25ABf077fE4b67C973401Bf2c	11/27/2024, 7:38:47 PM	Paris	Retailer SA
Available for Sale	0x843cC4F2A004F2Ab323C65A1001dCCf6f6e6e14e	11/27/2024, 7:38:47 PM	Paris	Retailer SA

Transaction submission

Beltran can submit a transaction, but only in the “creation” mode (since he is a supplier). He specifies:

- ID;
- Product name;
- Origin;
- Username;
- Product state.



Welcome, Beltran. With role: SUPPLIER_ROLE

Transaction Submission

Submit new transactions to the blockchain

abc123

Book

Argentina

Beltran

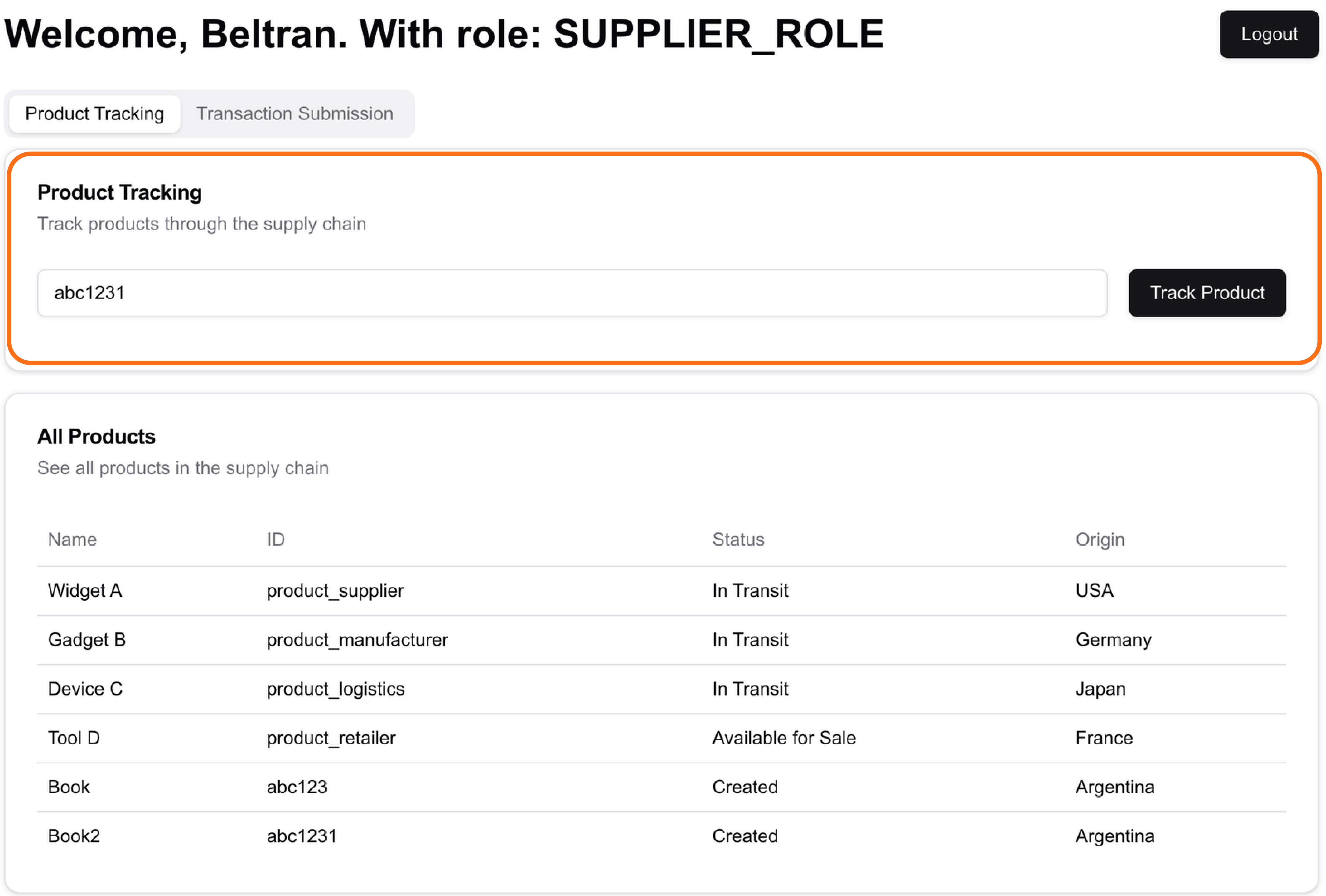
Created

Solidity UI

Product tracking

Now the product “Book” has been created and is visible in the supply chain.

We can also search for a product through its ID.



Welcome, Beltran. With role: SUPPLIER_ROLE

Product Tracking

Track products through the supply chain

abc1231

All Products

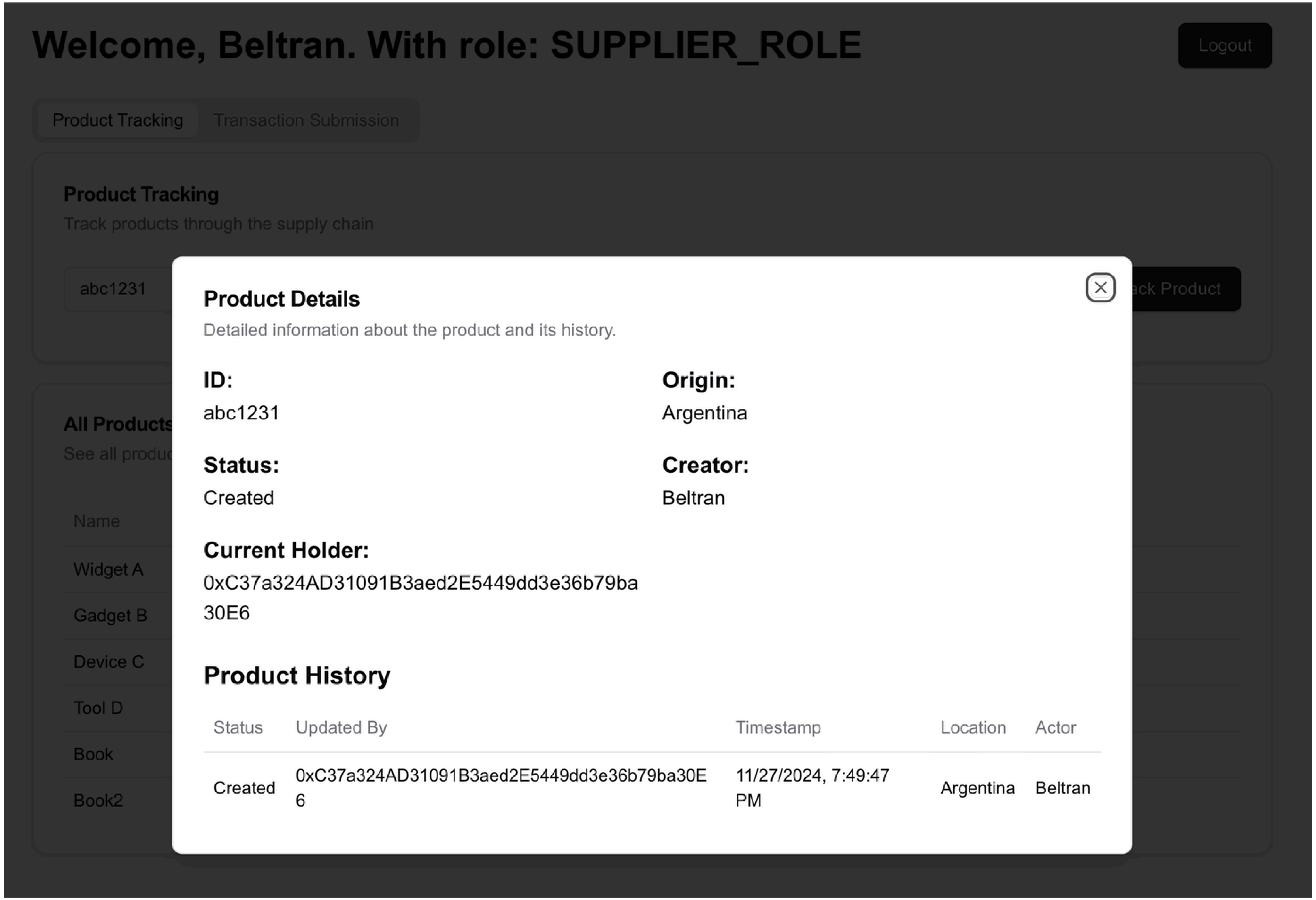
See all products in the supply chain

Name	ID	Status	Origin
Widget A	product_supplier	In Transit	USA
Gadget B	product_manufacturer	In Transit	Germany
Device C	product_logistics	In Transit	Japan
Tool D	product_retailer	Available for Sale	France
Book	abc123	Created	Argentina
Book2	abc1231	Created	Argentina

Solidity UI

Product tracking

By entering a product ID, he can see all the details about that product.



Welcome, Beltran. With role: SUPPLIER_ROLE

Product Tracking

Track products through the supply chain

All Products

See all produc

Name

Widget A

Gadget B

Device C

Tool D

Book

Book2

Product Details

Detailed information about the product and its history.

ID:
abc1231

Origin:
Argentina

Status:
Created

Creator:
Beltran

Current Holder:
0xC37a324AD31091B3aed2E5449dd3e36b79ba30E6

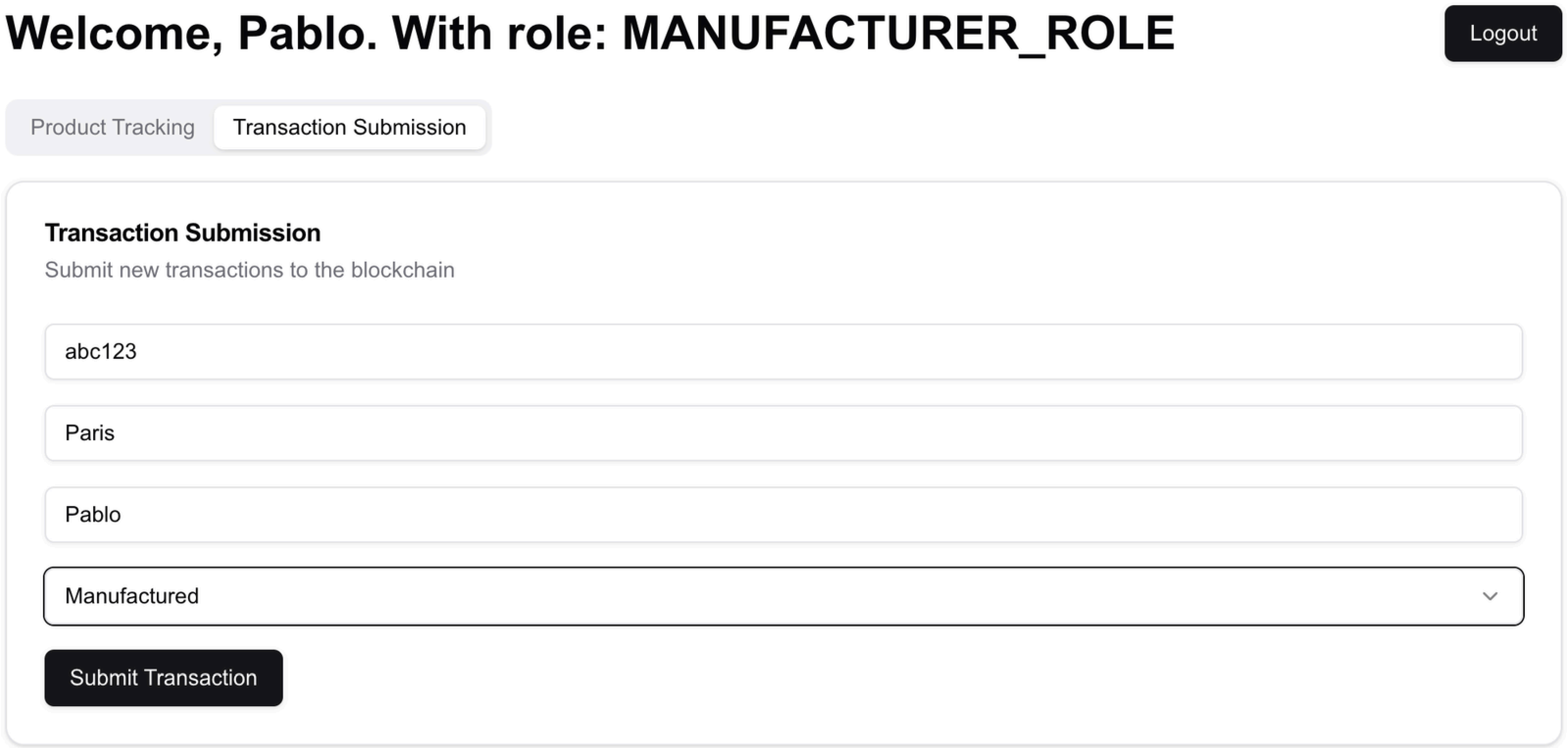
Product History

Status	Updated By	Timestamp	Location	Actor
Created	0xC37a324AD31091B3aed2E5449dd3e36b79ba30E6	11/27/2024, 7:49:47 PM	Argentina	Beltran

Manufacturer transaction submission

Pablo is a manufacturer: he works with the products given to him by suppliers.

Since the supplier Beltran created product “Book”, this product is ready to be manufactured.



Welcome, Pablo. With role: MANUFACTURER_ROLE

Transaction Submission

Submit new transactions to the blockchain

abc123

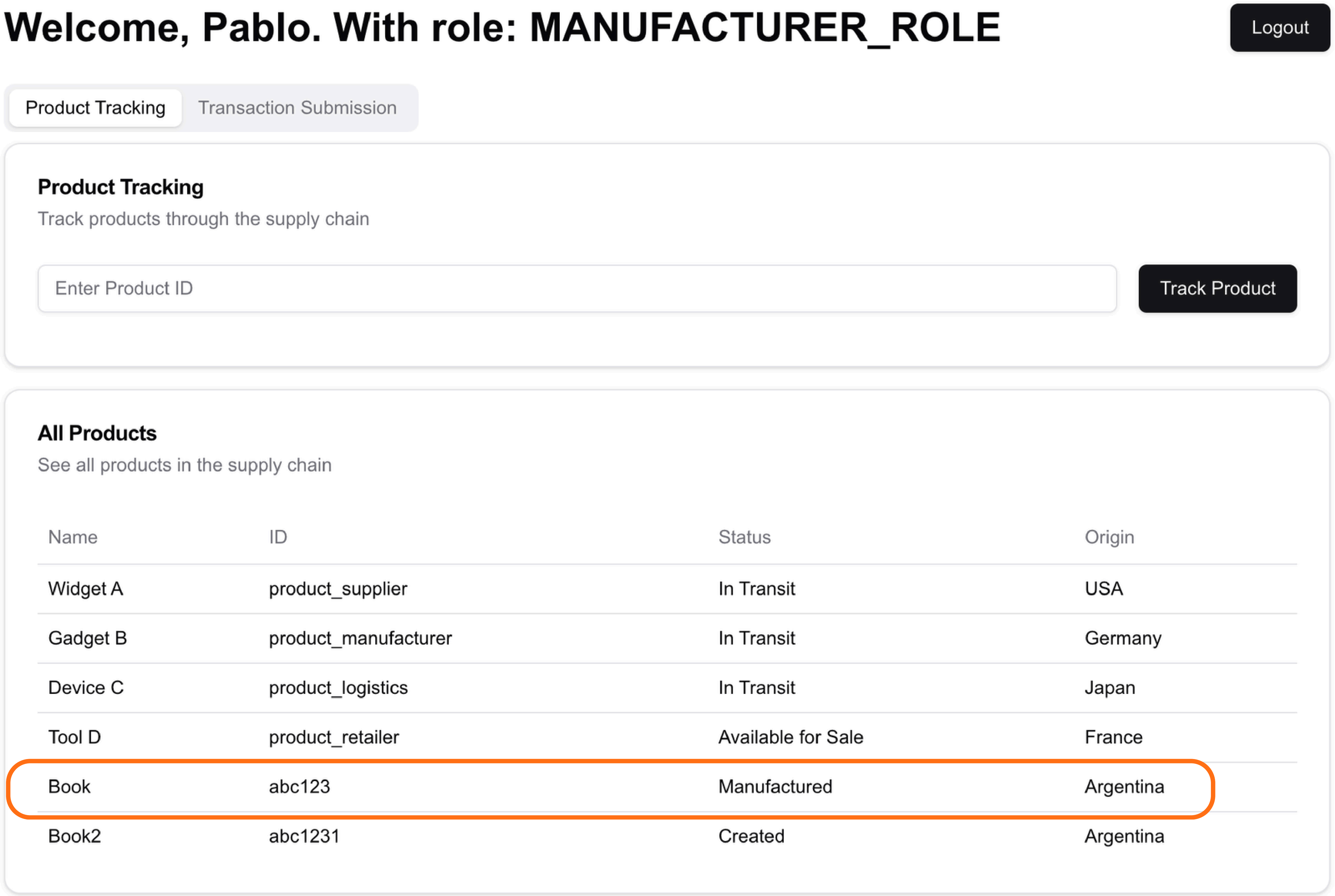
Paris

Pablo

Manufactured

Manufacturer transaction submission

Now Book's status has been updated from "Created" to "Manufactured".



Welcome, Pablo. With role: MANUFACTURER_ROLE

Product Tracking

Track products through the supply chain

Enter Product ID

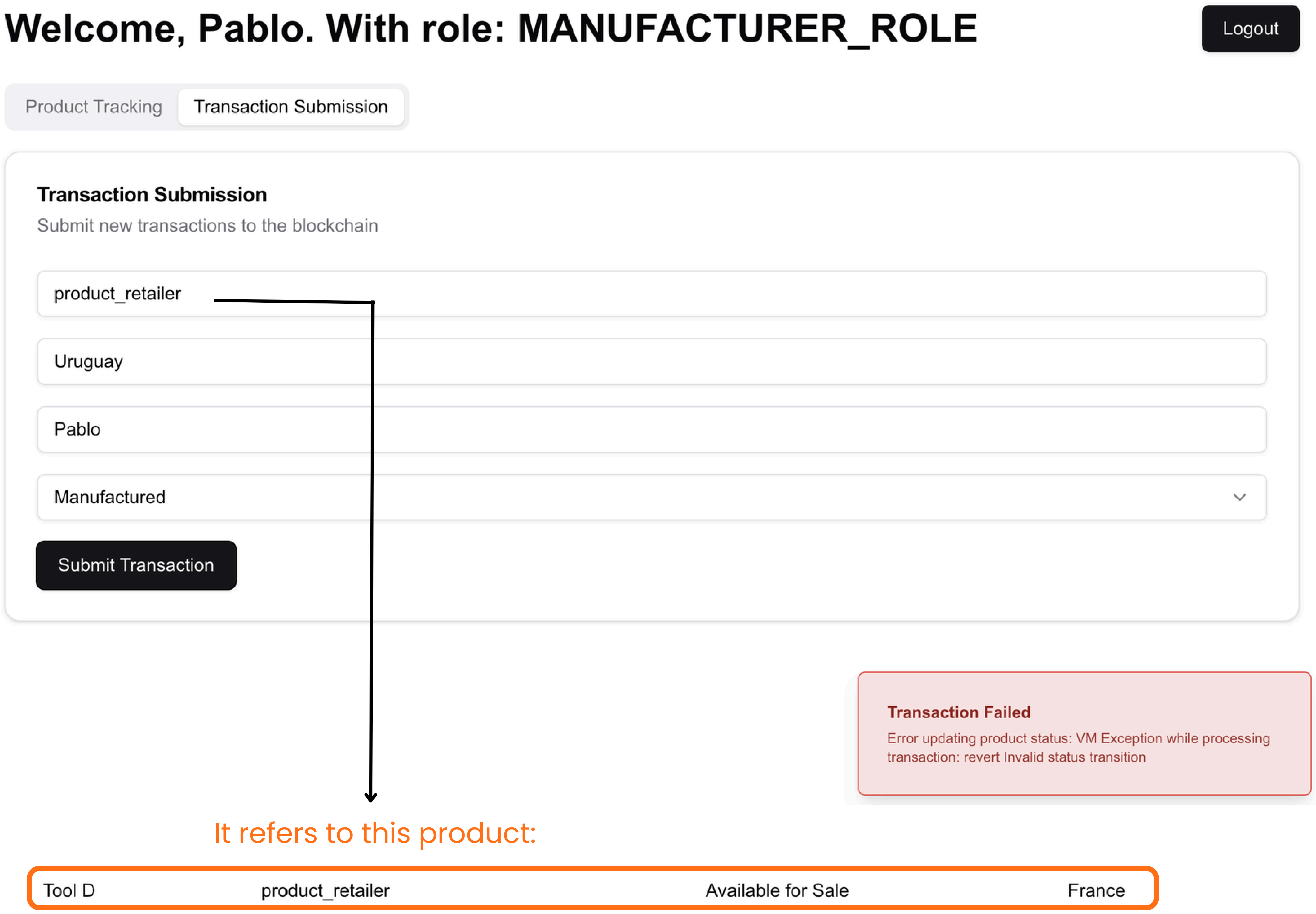
All Products

See all products in the supply chain

Name	ID	Status	Origin
Widget A	product_supplier	In Transit	USA
Gadget B	product_manufacturer	In Transit	Germany
Device C	product_logistics	In Transit	Japan
Tool D	product_retailer	Available for Sale	France
Book	abc123	Manufactured	Argentina
Book2	abc1231	Created	Argentina

Manufacturer transaction submission

If Pablo tries to manufacture a product owned by a retailer, the transaction fails.



Welcome, Pablo. With role: MANUFACTURER_ROLE

Transaction Submission

Submit new transactions to the blockchain

product_retailer

Uruguay

Pablo

Manufactured

Transaction Failed

Error updating product status: VM Exception while processing transaction: revert Invalid status transition

It refers to this product:

Tool D	product_retailer	Available for Sale	France

Conclusion

Python vs Solidity method

Python

ADVANTAGES

01 Provides more liberty and possibilities

02 Provides better understanding of the blockchain structure

DISADVANTAGES

01 Limited smart contract support

02 Provides more liberty and possibilities

Solidity

ADVANTAGES

01 The blockchain structure is more robust, since it has been broadly tested and refined

02 It requires less time do implement since most of the basic structure is already ready

DISADVANTAGES

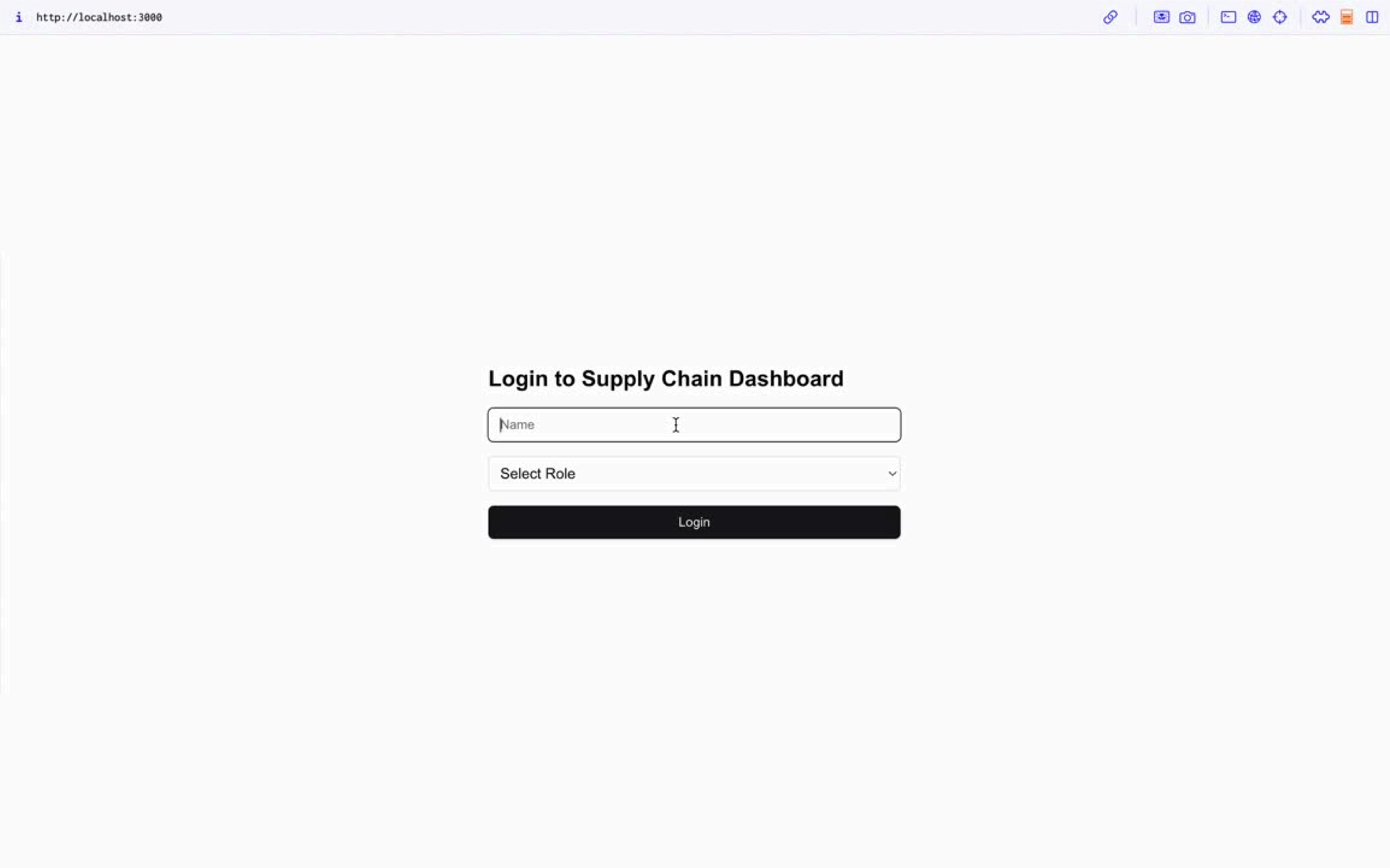
01 Smart contracts written in Solidity run on the Ethereum Virtual Machine (EVM) and require gas fees for execution

02 It requires developers to learn its syntax, nuances, and security practices

Comprendre la Blockchian

Thank You For Attention

Demo



http://localhost:3000

Login to Supply Chain Dashboard