Event Ticketing System

Documentation

Overview

The Event Ticketing System implemented on Hyperledger Fabric establishes a secure and transparent platform for managing event tickets. The system involves three key entities: Organiser, Reseller, and Attendee, each representing an organisation within a private blockchain network. The core logic is encapsulated in the smart contract named `Ticket-Contract`, offering a set of functions to create, read, transfer, sell, and delete tickets, as well as manage orders.

System Architecture

Participants

Organiser: Responsible for creating events and generating initial tickets.
Reseller: Facilitates ticket sales, transfers, and interactions with attendees.
Attendee: Purchases tickets and manages personal ticket inventory.

Smart Contract: Ticket-Contract

The `ticket-contracts.go` file is a critical component of the Event Ticketing System implemented on Hyperledger Fabric. This file contains the smart contract logic, defining the rules and functionalities that govern the behavior of the system on the blockchain.
The primary purpose of the `ticket-contracts.go` file is to define the smart contract, also known as chaincode, that will be deployed and executed on the Hyperledger Fabric network. This smart contract encapsulates the business logic of the Event Ticketing System, including functions for creating tickets, transferring ownership, selling tickets, managing orders, and other relevant operations.

Struct Definition:

- The smart contract logic is often encapsulated in a struct, defining the data structure and methods associated with the chaincode.

Ticket Struct:

```
type Ticket struct {

   AssetType         string `json:"AssetType"`

   TicketId             string `json:"TicketId"`

   EventId             string `json:"EventId"`

   EventDate string `json:"EventDate"`

   Price             string `json:Price`

   OwnedBy           string `json:"OwnedBy"`

   Event              string `json:"Event"`

   Status            string `json:"Status"`

}
```

The smart contract governs ticket-related operations and maintains a ledger of ticket transactions. Key functions include:

Create Ticket: Generates a new ticket with event details.

Read Ticket: Retrieves ticket information based on the ticket ID.

Get Ticket by Range: Retrieves tickets within a specified ID range.

Get Ticket History: Provides a historical record of a ticket's transactions.

Transfer Ticket: Transfers ownership of a ticket to another participant.

Sell Ticket: Facilitates the sale of a ticket by a reseller.

Delete Ticket: Removes a ticket from the system.

Create Order: Generates an order for multiple tickets.

Read Order: Retrieves information about a specific order.

Delete Order: Removes an order from the system.

Workflow

Ticket Creation: The Organiser initiates the process by creating event tickets using the `CreateTicket` function. These tickets are initially owned by the organiser.
Ticket Transfer and Sale: Ownership of tickets can be transferred or sold using the `TransferTicket` and `SellTicket` functions, respectively. Resellers play a crucial role in selling tickets to attendees.
Ticket Deletion: Unwanted or expired tickets can be deleted from the system using the `DeleteTicket` function.
Order Management: Attendees can place orders for multiple tickets using the `CreateOrder` function. The system also supports reading and deleting orders using the `ReadOrder` and `DeleteOrder` functions.
Querying Tickets: The system provides query functions such as `ReadTicket`, `GetTicketByRange`, and `GetTicketHistory` to retrieve specific ticket details.

Deployment

Hyperledger Fabric Network Setup

The system requires the deployment of a Hyperledger Fabric network with three organisations: Organiser, Reseller, and Attendee. The smart contract is installed and instantiated on the network to enable interaction.

Fabric Network

The components of the network is defined in **Fabric-Network** folder

Docker:

In this folder, two critical configuration files are present: "docker-compose-ca.yaml" and "docker-compose-3org.yaml."

The **"docker-compose-ca.yaml"** file orchestrates the deployment of the Certificate Authority (CA) components, essential for managing digital certificates within the network. This ensures secure communication and identity verification among the participants—Organisers, Resellers, and Attendees.

Meanwhile, the **"docker-compose-3org.yaml"** file defines the specifications for the three participating organisations within the network: Organiser, Reseller, and Attendee. This file orchestrates the deployment of the necessary peers, orderers, and other components, creating a cohesive and secure environment for the Event Ticketing System.

These Docker Compose files play a pivotal role in the system's deployment, ensuring the proper instantiation and configuration of the Hyperledger Fabric network components. They are integral to the overall network setup and facilitate the seamless interaction of participants and smart contracts within the blockchain ecosystem. For detailed execution, please refer to the accompanying command documentation, which outlines the specific commands for initiating the network and related components.

Core.yaml:

As an integral part of the network setup, the "core.yaml" file contributes to the proper functioning of peers, influencing aspects such as identity verification, encryption, and overall network communication. It is an essential component that aligns with the broader goal of establishing a secure and transparent environment for the Event Ticketing System.

Configtx.yaml

The contents of the "configtx.yaml" file are instrumental in establishing the network's key elements, including channels and participating organizations—Organiser, Reseller, and Attendee. It influences the creation of the Genesis Block, which initializes the blockchain, and governs the subsequent configuration updates.

startNetwork.sh and stopNetwork.sh

The `startNetwork.sh` script file has been written for initiating the Hyperledger Fabric network effortlessly. By executing this script through the command `./startNetwork`, users can efficiently start the network components and bring the Event Ticketing System to life.

This script encapsulates a series of essential commands, orchestrating the deployment of the network's components, including peers, orderers, and Certificate Authorities (CAs). By running this script, users can ensure a seamless and coherent launch of the Hyperledger Fabric network defined in the "Fabric-Network" directory.

For a more detailed understanding of the specific commands and operations performed by the `startNetwork.sh` script, it is recommended to refer to the script file directly. The accompanying command documentation provides a step-by-step guide for users to comprehend and execute the script successfully. This script significantly contributes to the ease of deployment for users, enabling them to effortlessly start the network and proceed with the operation of the Event Ticketing System on Hyperledger Fabric.

Similarly execute ./stopNetwork.sh to stop the network and the docker containers.

Client

The client application is a vital component of the Hyperledger Fabric Event Ticketing System, enabling users to interact with the blockchain network. The `client` folder encapsulates key functionalities for connecting to the network, creating, managing, and querying tickets, as well as handling user profiles. Here's an overview of the relevant files:

main.go

The `main.go` file serves as the entry point for the client application. It likely initializes configurations, establishes a connection to the Hyperledger Fabric network, and triggers the execution of essential functions. This file acts as the starting point for executing the client application.

client.go

The `client.go` file houses the main logic and functionalities of the client application. Functions within this file are likely responsible for interacting with the smart contracts deployed on the Hyperledger Fabric network. Examples of functionalities could include creating tickets, reading ticket details, transferring tickets, and managing orders. This file is crucial for orchestrating various actions within the ticketing system.

connect.go

In the `connect.go` file, the connection-related logic is likely implemented. This includes establishing a connection to the Hyperledger Fabric network, setting up the necessary configurations, and managing the communication between the client application and the blockchain network. This file plays a pivotal role in ensuring seamless interaction with the underlying fabric infrastructure.

profile.go

The `profile.go` file may handle user profiles within the client application. It could contain functions for creating and managing user profiles, associating users with their respective roles (organiser, reseller, attendee), and ensuring secure access to the ticketing system. Managing user profiles is essential for maintaining a secure and personalized experience within the ticketing application.

The collective functionality of these files in the `client` folder contributes to a robust and user-friendly client application that interacts seamlessly with the Hyperledger Fabric network for event ticketing. The modular structure allows for clear separation of concerns and ease of maintenance.

Events

The `events` folder houses the Event Ticketing System's event management application, responsible for creating, updating, and overseeing events within the ticketing platform. Below event is defined for the project.

```
chaincodeEventListener("organiser", "ticketchannel", "Final-Project")
```