**Car-Auction On Blockchain Using HyperledgerFabric**

In this project one can list assets for sale (setting a reserve price), and watch as assets that have met their reserve price are automatically transferred to the highest bidder at the end of the auction.

* **Project Link**
* <https://github.com/manu461/ProjectsDemo/tree/master/BlockChainProjects/HyperLedgerComposerFabricProject/carauction-network>
* <https://github.com/manu461/ProjectsDemo/tree/master/BlockChainProjects/HyperLedgerComposerFabricProject/carauction-network/carauction-angular-app>
* **Project Description**

I created this project while completing **IBM Blockchain-Foundation Developer** course.

Car-Auction’s(this project’s) Business-network is developed using **Hyperledger Composer**, whereas the network been deployed locally using **Hyperledger Fabric**.

The complete package**(.bna)** containing **Business-Model**(.cto file), **script files**(.js file), **ACLs**(.acl file) and **Metadata**(.md file) is converted to an **Angular Application** using

**Yeoman** **generator**(YO code generator).

In this project one can list assets for sale (setting a reserve price), and watch as assets that have met their reserve price are automatically transferred to the highest bidder at the end of the auction.

This business network defines:

1. Participants
2. Member
3. Auctioneer
4. Assets
5. Vehicle
6. VehicleListing
7. Transactions:
8. Offer
9. CloseBidding

The **makeOffer** function is called when an **Offer** transaction is submitted. The logic simply checks that the listing for the offer is still for sale, and then adds the offer to the listing, and then updates the offers in the **VehicleListing** asset registry.

The **closeBidding** function is called when a **CloseBidding** transaction is submitted for processing. The logic checks that the listing is still for sale, sorts the offers by bid price, and then if the reserve has been met, transfers the ownership of the vehicle associated with the listing to the highest bidder. Money is transferred from the buyer's account to the seller's account, and then all the modified assets are updated in their respective registries.

\*\*NOTE: This is not my idea to develop such project, it is already provided as a template for business-network by IBM at their composer-playground website. I have studied the code and logics thoroughly and implemented it during completing IBM Blockchain-Foundation Developer course.

* **Tools Used**

1. Data-Modelling is done with **composer modelling language**. It is domain specific language for describing the nature of the business network. It is developed by IBM.
2. Business-logic is written in **JavaScript**.
3. **Composer-Playground** is used to do ideation and very quickly prototype composer solution.
4. Two main **npm** modules
5. Composer-client
6. Composer-Admin

which can be embedded in application for programmatic access.

1. **VSCode** is used as code-editor.
2. **CLI tools**
3. Complete package of Business network Archive (.bna) is converted into an Angular-Application using **Yeoman** **generator**(YO code generator).

* **Project Walkthrough**

1. **Starting the Application**
2. **Start Fabric**
3. Press **ctrl+alt+t** to open terminal.
4. Change directory to fabric-tools by typing **cd fabric-tools** on the terminal.
5. Then type this code on the terminal **./startFabric.sh**
6. **Start Business Network**
7. Type **cd ..**  and press enter on terminal to exit fabric-tools directory.
8. Change directory to carauction-network by typing **cd carauction-network** on the terminal.
9. Install the business network.

From the carauction-network directory, run the following command,

**composer network install --card PeerAdmin@hlfv1 --archiveFile carauction-network@0.0.1.bna**

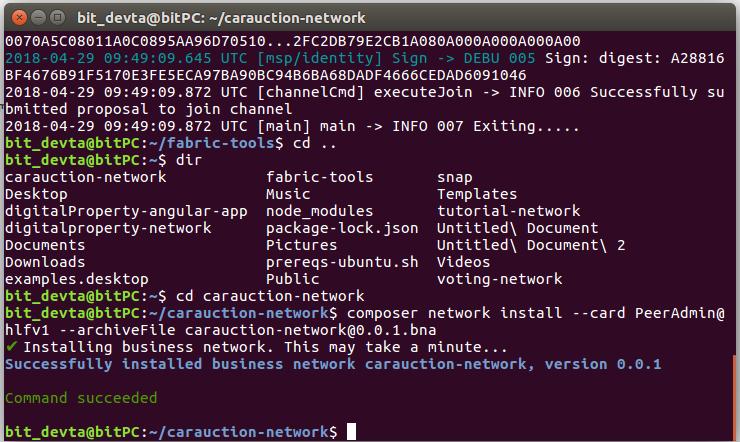


Figure 1. Install the business network

1. Start the business network

Do not change the directory and run the following command,

**composer network start --networkName carauction-network --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file PeerAdmin.card**

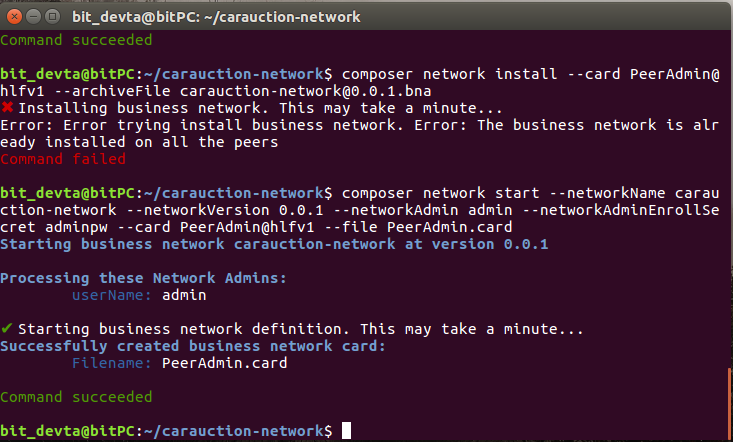


Figure 2. Start the business network

1. Check Ping

To check that the business network has been deployed successfully, run the following command to ping the network.

**composer network ping --card admin@carauction-network**

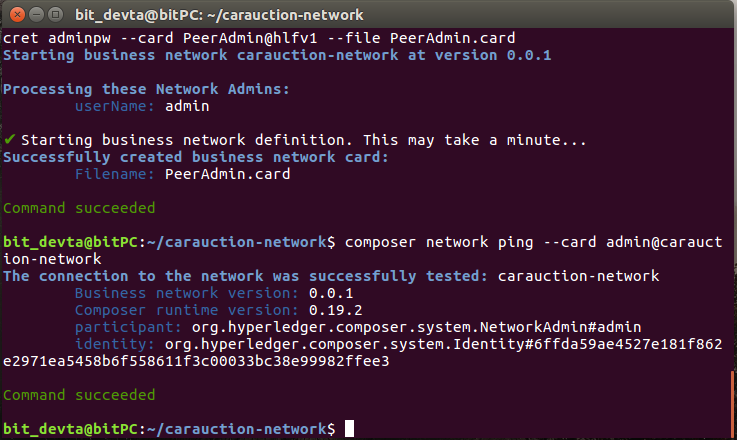


Figure 3. Ping check the Business network

1. **Start Rest API**

Type **composer-rest-server** on the terminal, and then,

1. Enter **admin@carauction-network** as business network card to use.
2. Select **never use namespaces**
3. Press **n** to specify if you want to enable authentication for REST API.
4. Press **y** to specify if you want to enable event publication.
5. Press **n** to specify if you want to enable TLS security.

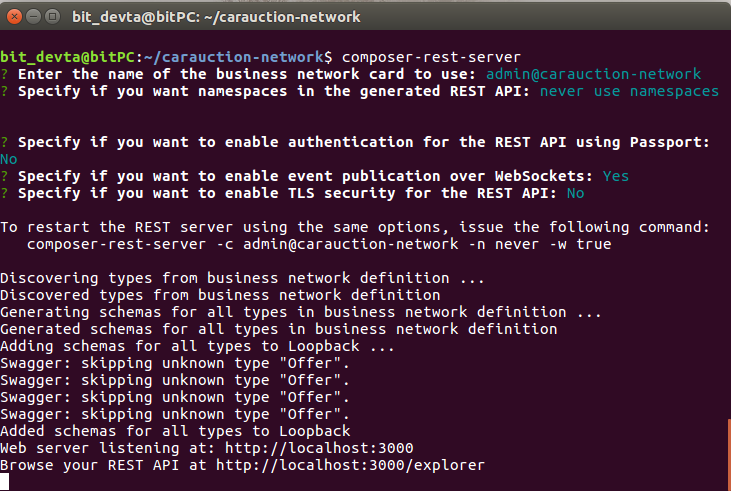


Figure 4. Start the rest API

1. **Start the Angular-App**
2. Open a new terminal using **ctrl+alt+t**
3. Change the directory using **cd carauction-network/carauction-angular-app**
4. Enter the following command to start the app **npm start**

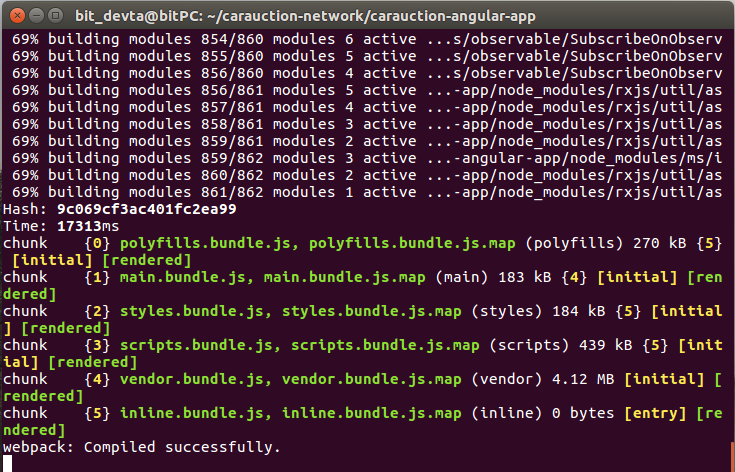


Figure 5. Start the angular-app

1. Angular-App is now started, open <http://localhost:4200>
2. **Application Walkthrough**
3. **Home Page**

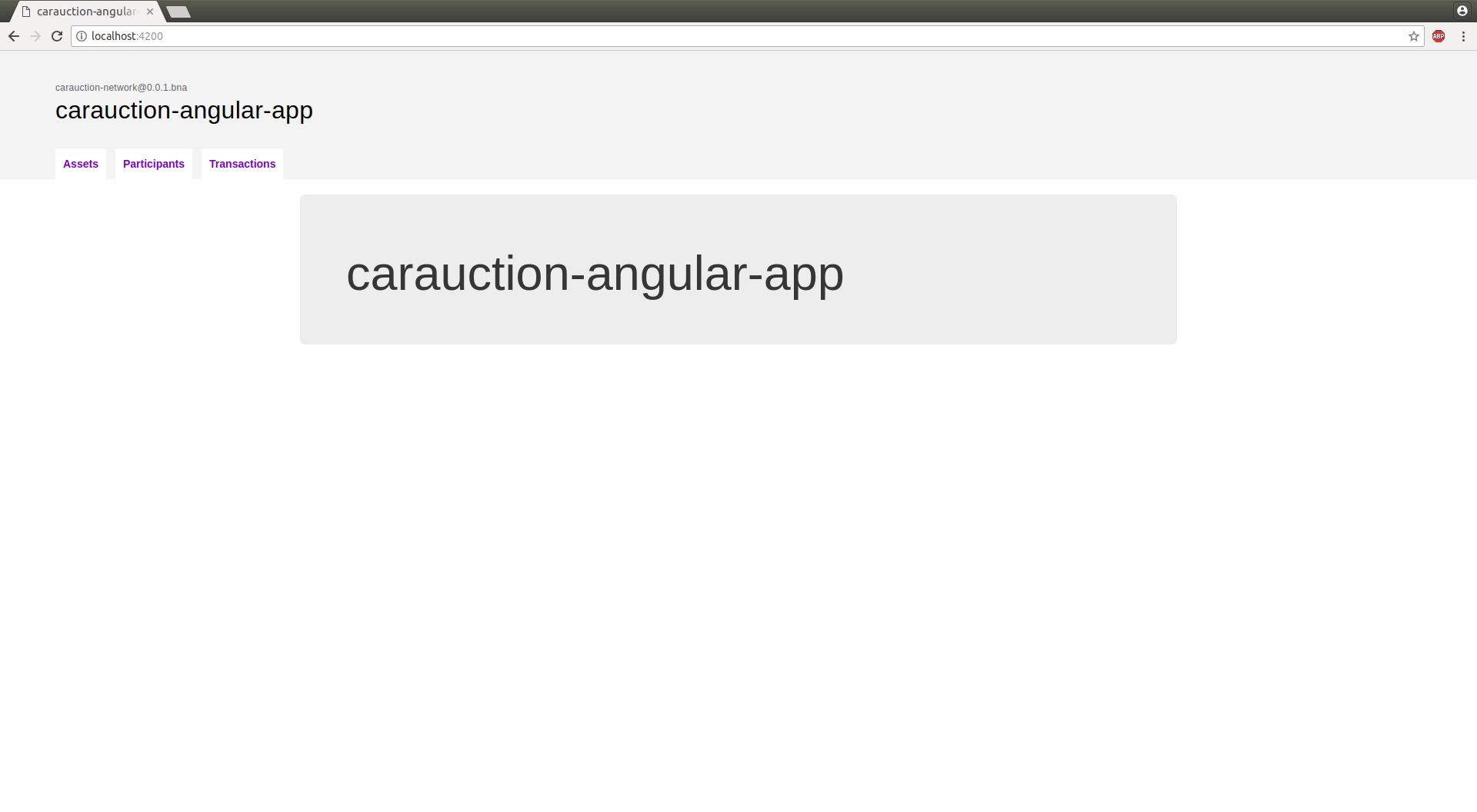


Figure 6. Home Page

1. **Assets Tab**

There are two types of assets:-

1. Vehicle

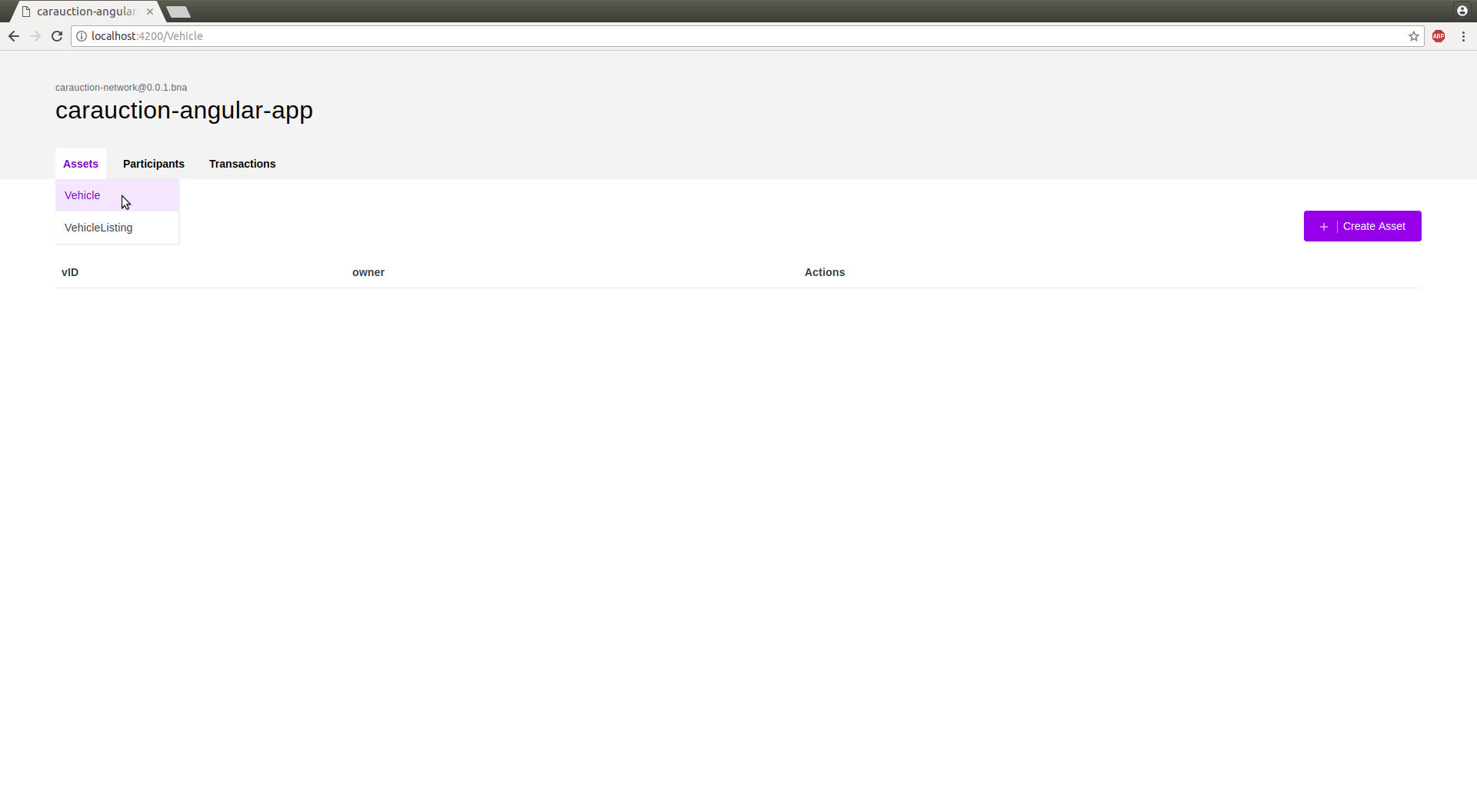


Figure 7. List of all Vehicle Asset

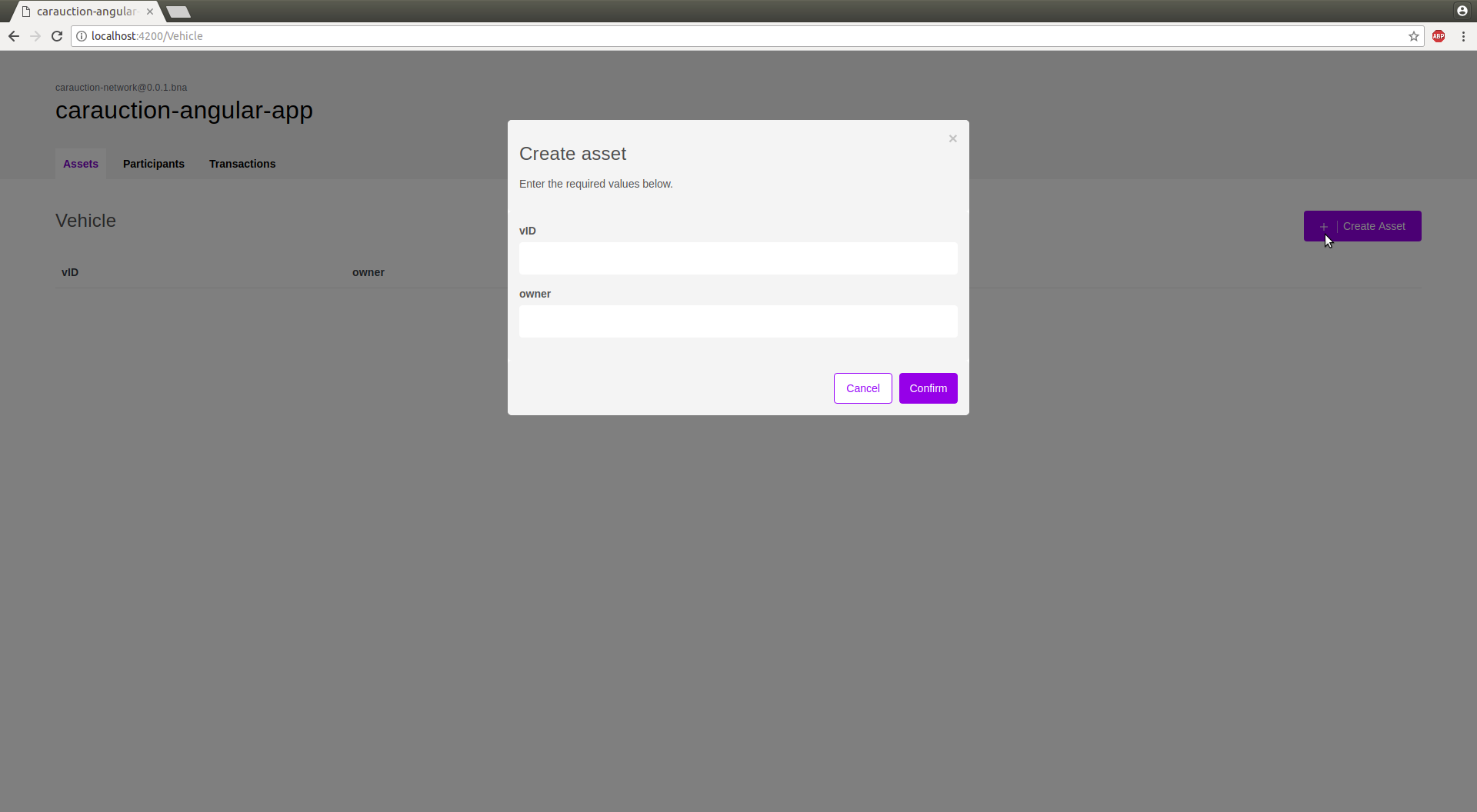


Figure 8. Create a new Vehicle Asset

Every vehicle is identified by a unique variable **vID**, and contains a reference to a member variable called **Owner**.

To create a new vehicle, one is required to insert a unique **vID** and a reference to a member who is supposed to be the **Owner** of the vehicle.

1. Vehicle Listing

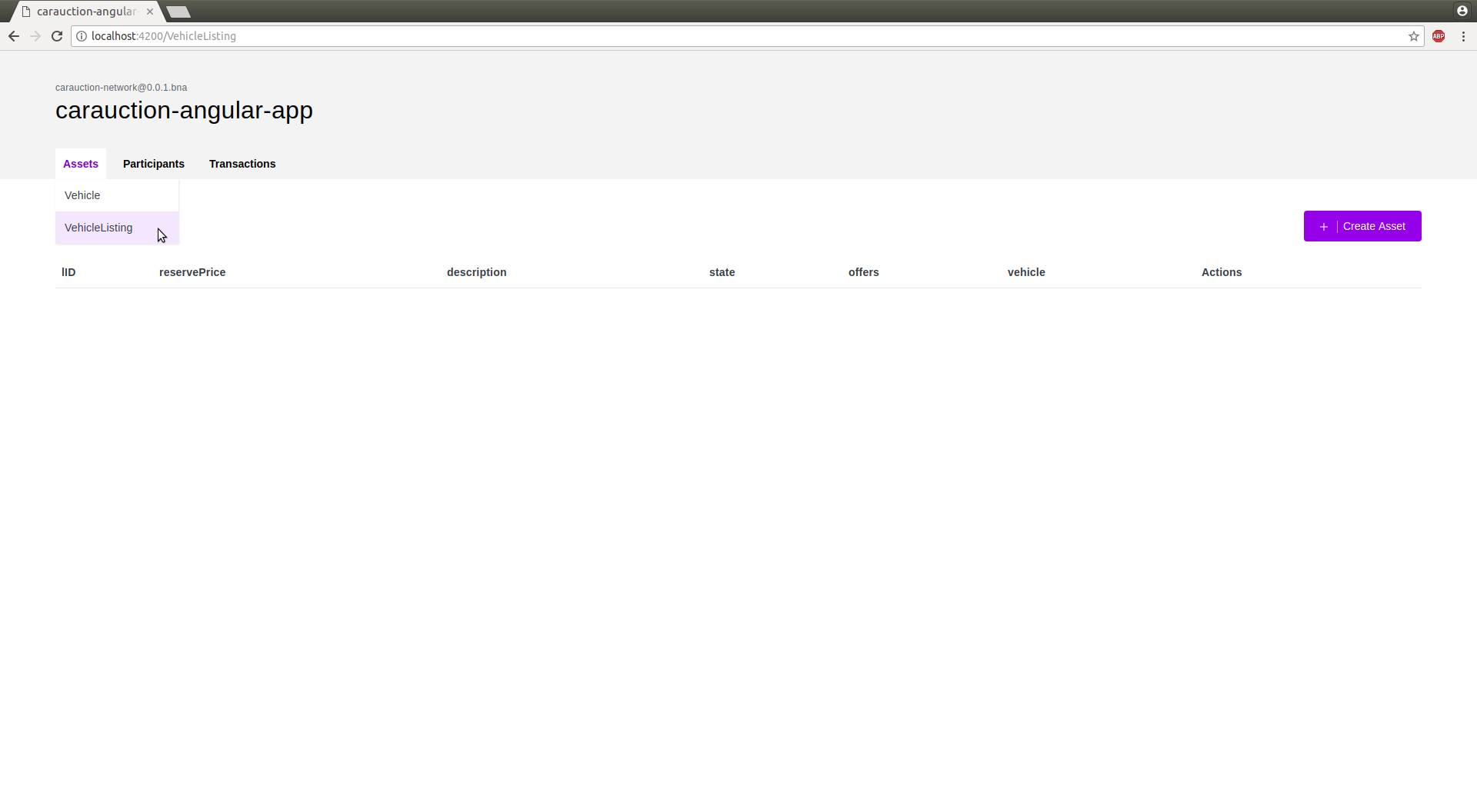


Figure 9. List of all VehicleListing

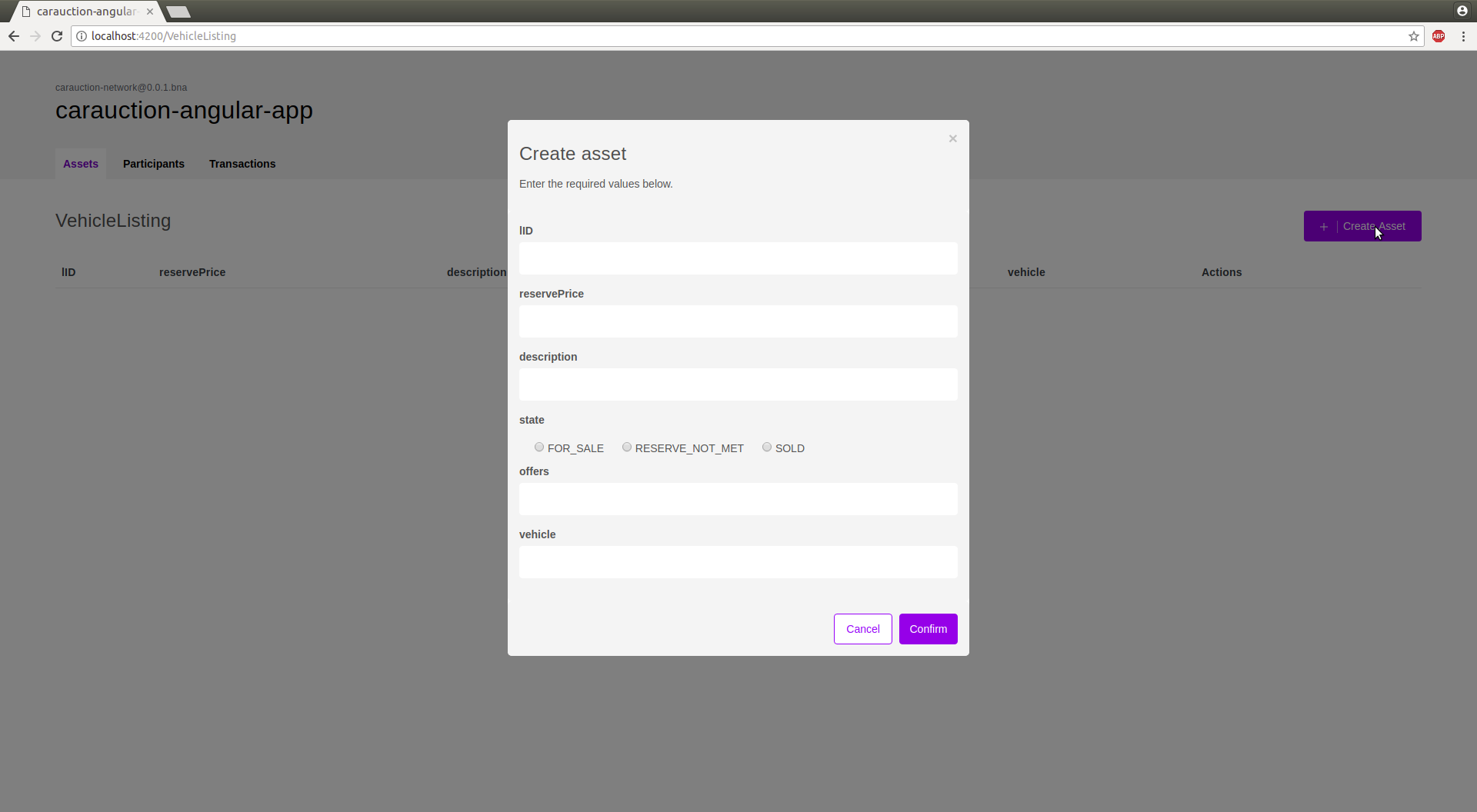


Figure 10. Create a new VehicleListing

Vehicle listing is used to list a vehicle for the auction.

It is identified by a unique variable **lID**. Every vehicle listing is associated with a vehicle using a reference to asset variable called **Vehicle**.

There is a threshold price known as **reserved price** associated, the bid price has to be more than reserved price in order to successfully buy the asset.

1. **Participants Tab**

There are two types of participants:-

1. Member

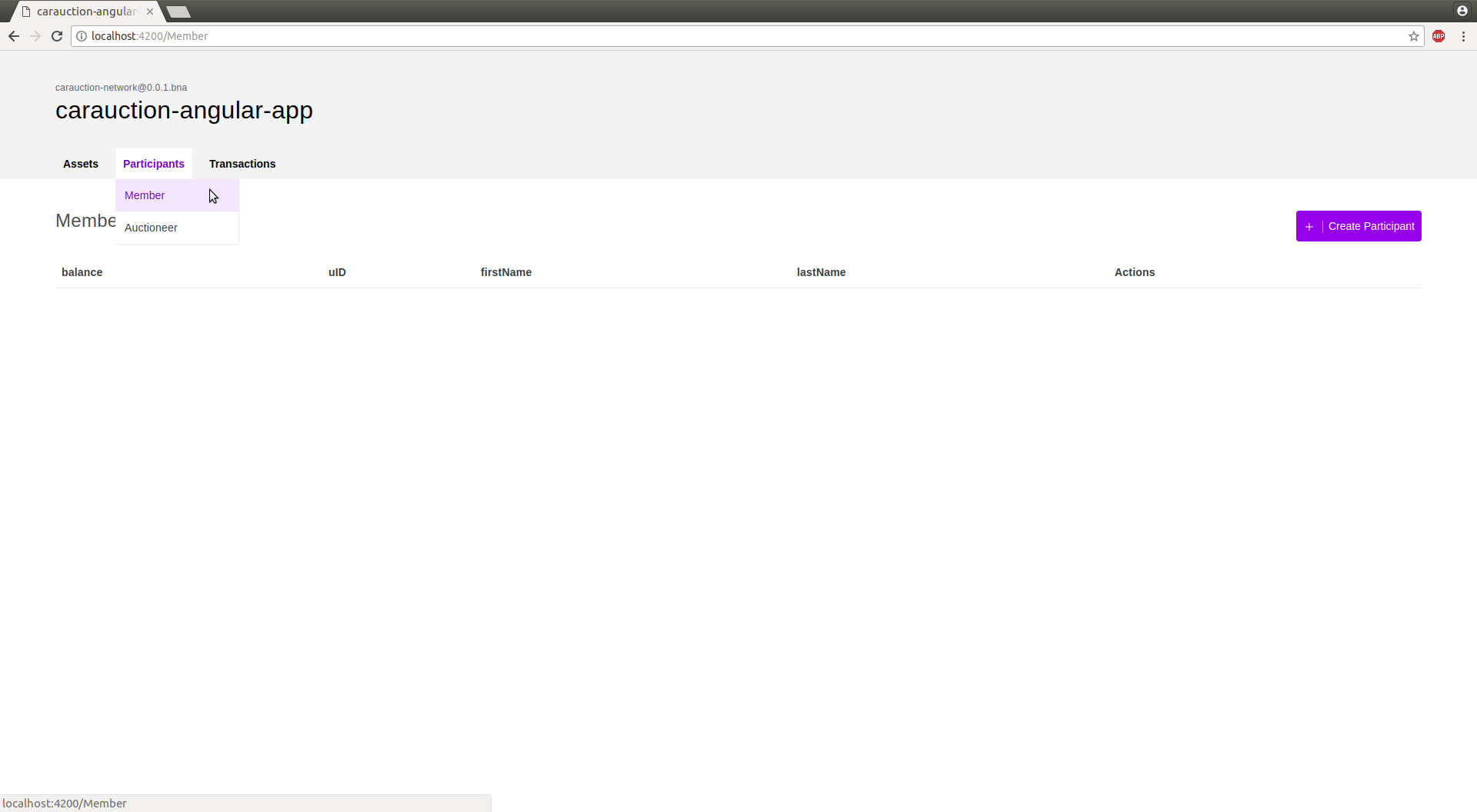


Figure 11. List of all members

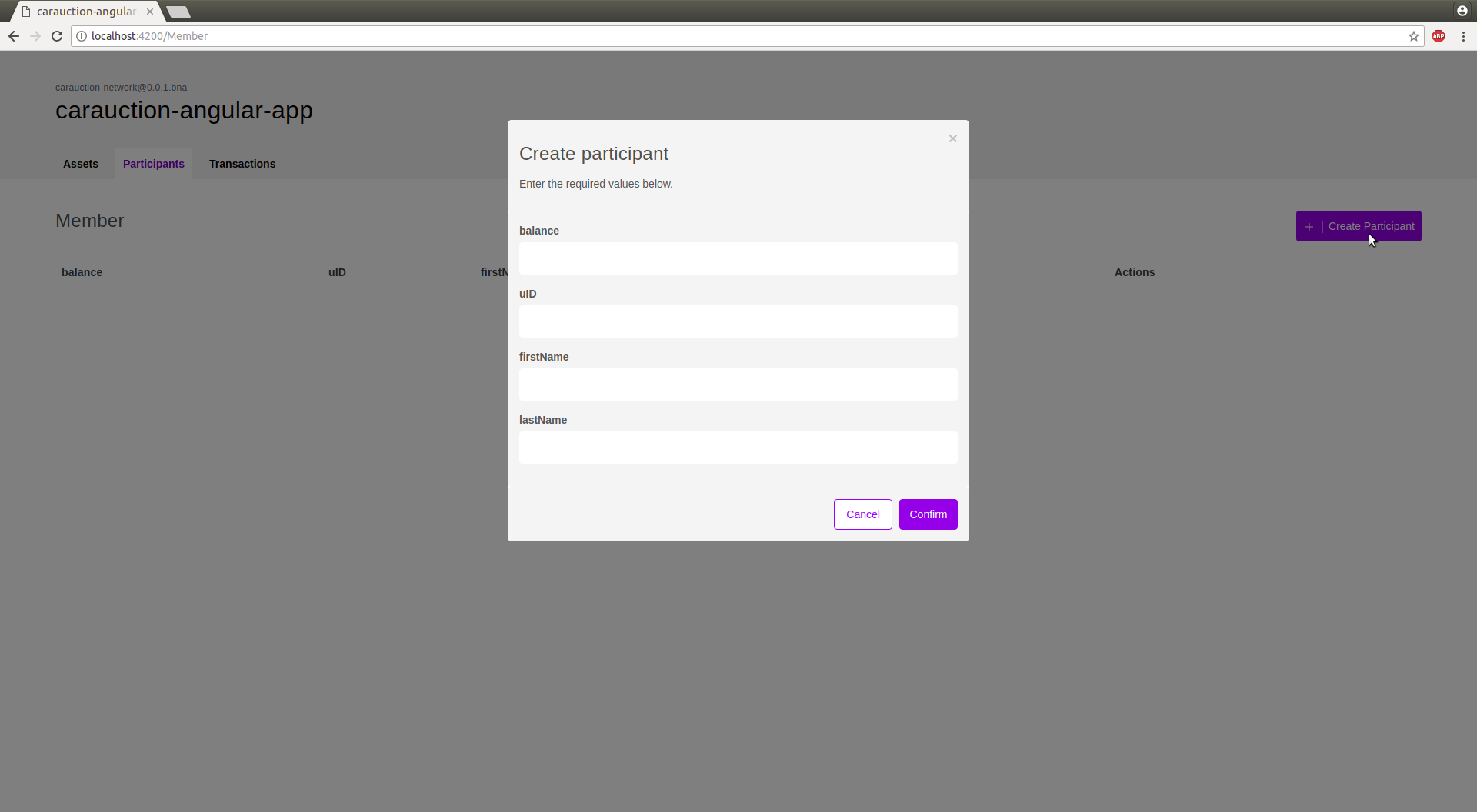


Figure 12. Add a new Member

Member is a participant of the auction, members are identified by unique member id **uID**. Every member has **Fist name**, **last name**, and **Balance** and is identified by a unique **uID**.

1. Auctioneer

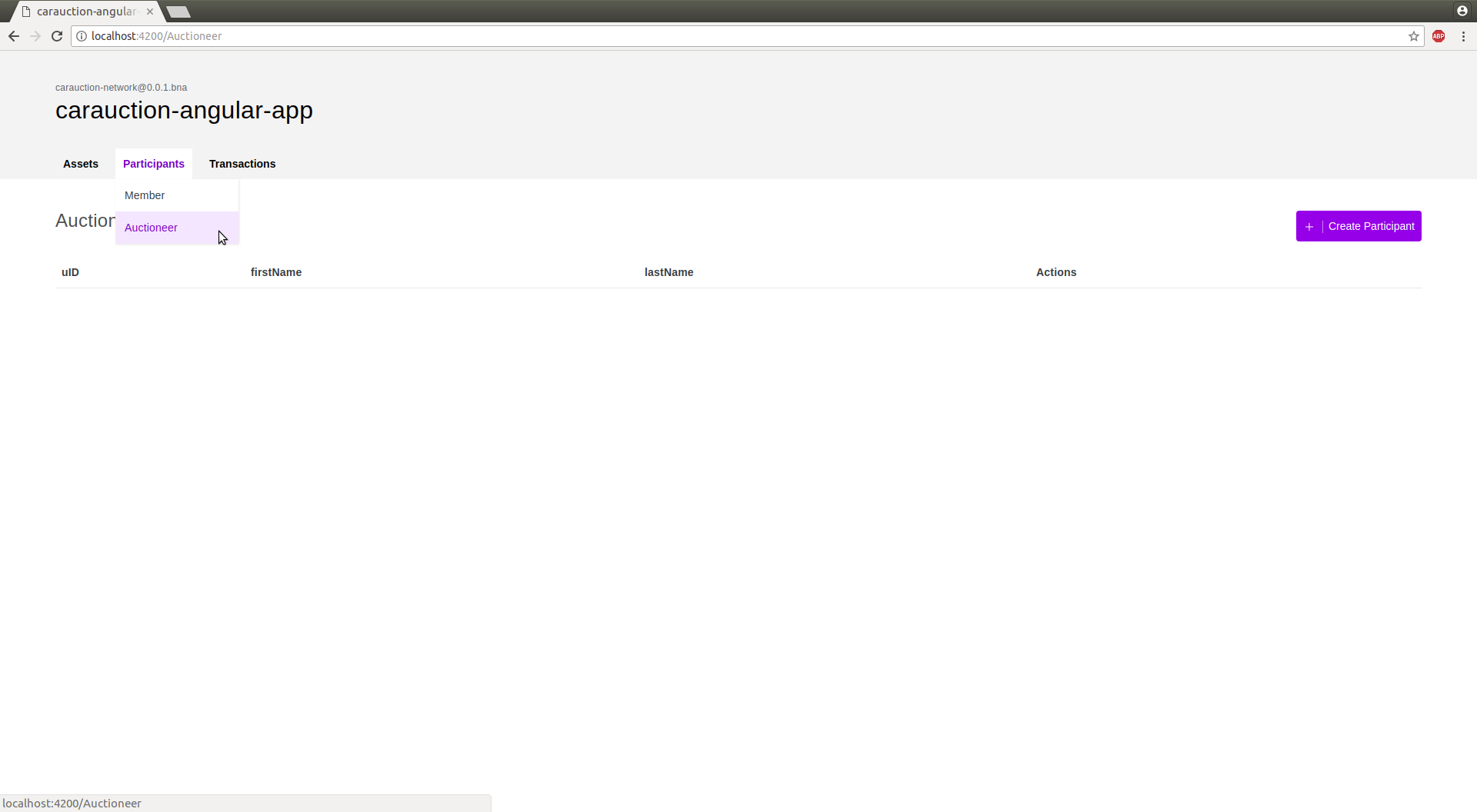


Figure 13. List of all Auctioneer

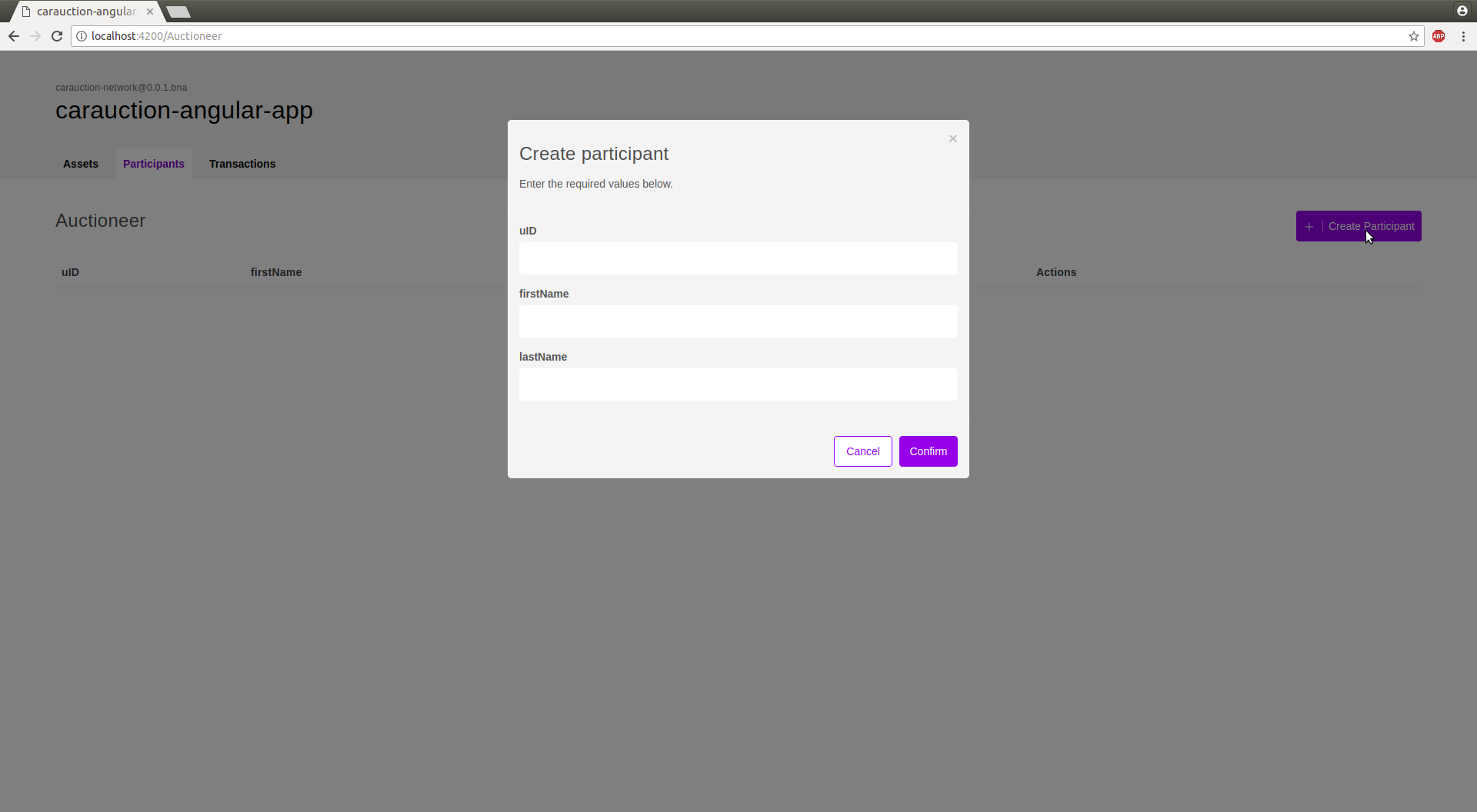


Figure 14. Add a new Auctioneer

Auctioneer is also a participant of the auction, but unlike member-auctioneer is used in order to provide oversight of the network.

Auctioneer is identified using a unique id **uID**. An auctioneer only has **First name** and **Last name** as its attribute.

1. **Transaction Tab**

There are two types of transactions:-

1. Offer
2. CloseBidding

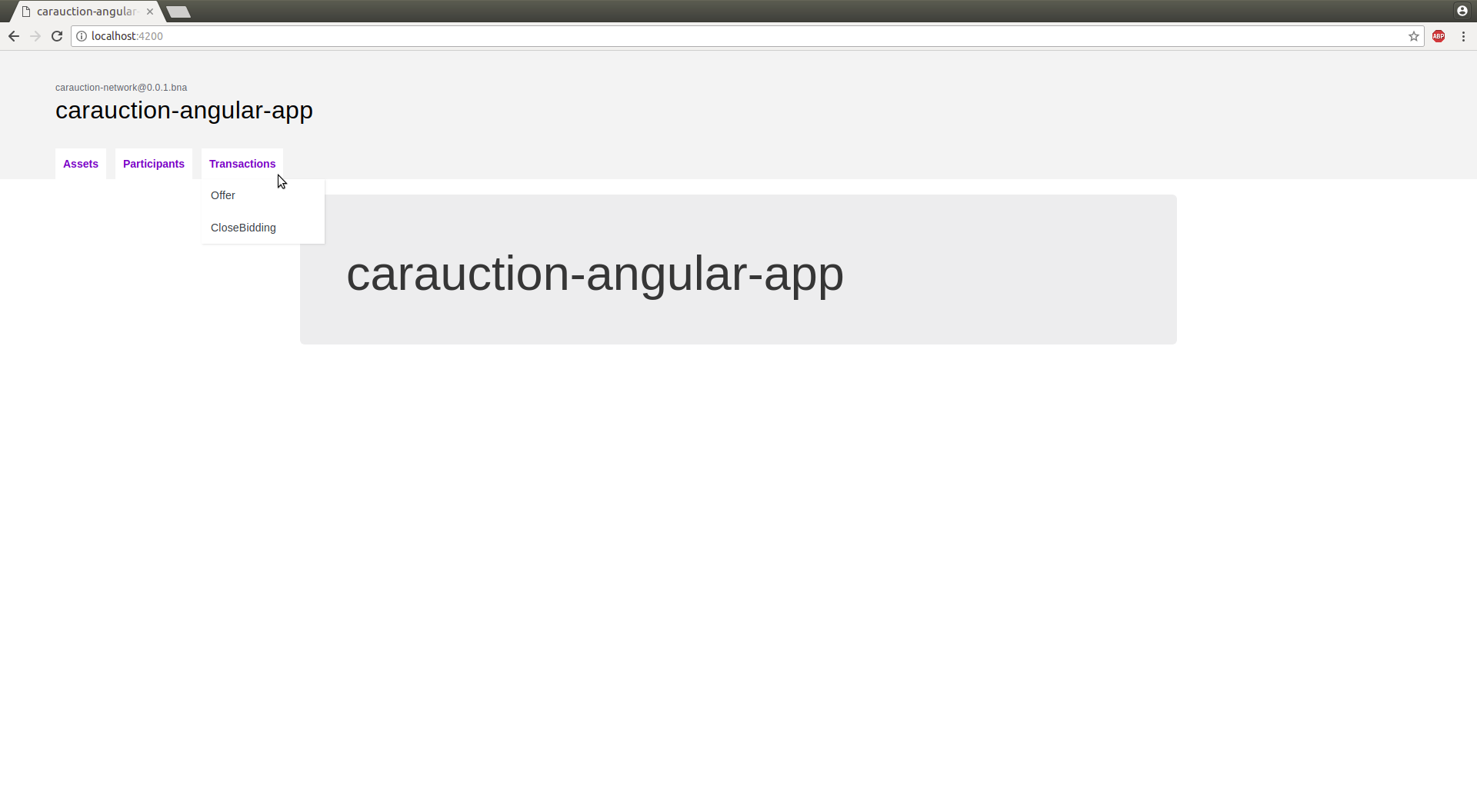


Figure 15. Transactions Tab

The **makeOffer** function is called when an **Offer** transaction is submitted. The logic simply checks that the listing for the offer is still for sale, and then adds the offer to the listing, and then updates the offers in the **VehicleListing** asset registry.

The **closeBidding** function is called when a **CloseBidding** transaction is submitted for processing. The logic checks that the listing is still for sale, sorts the offers by bid price, and then if the reserve has been met, transfers the ownership of the vehicle associated with the listing to the highest bidder. Money is transferred from the buyer's account to the seller's account, and then all the modified assets are updated in their respective registries.