

# Marketplace

---

Using Hyperledger Fabric

# Synopsis

- Purpose
- Architecture
- Demo
  - Web client
  - Rest API
- Problems faced
- Resources



# Purpose

---

# Purpose & Objective

- To implement a DApp to create a marketplace on Hyper Ledger Fabric Framework.
- The purpose of the DApp is to buy and sell commodities between the different accounts.
- Different Assets and Account holders can be created using the Front end interface.
- Also the event monitoring process in order to maintain the single ledger for transactions.

## Inspiration:

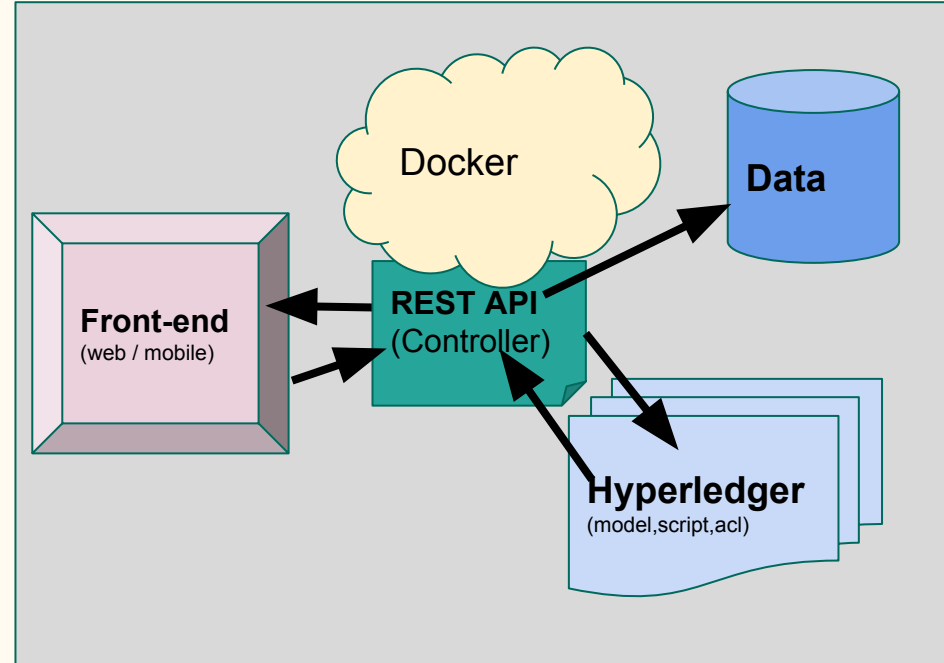
- Craigslist.
- The Distributed Ledger Technology to have single Ledger distributed over the various nodes. This single ledger helps in deriving the history of the commodity and its flow between the different ownerships.

# Architecture



# Architecture of the Market Place

- **Docker** - Run the Hyper Ledger Fabric Peer
- **Hyperledger Composer** - To define the Business Network definitions (Model files, Logic Javascript, Access Control List)
- **Composer Rest Server** - To create the REST API Interface to the Hyperledger Application
- **Angular Front End** - To Interact with expose REST API.



# Demo

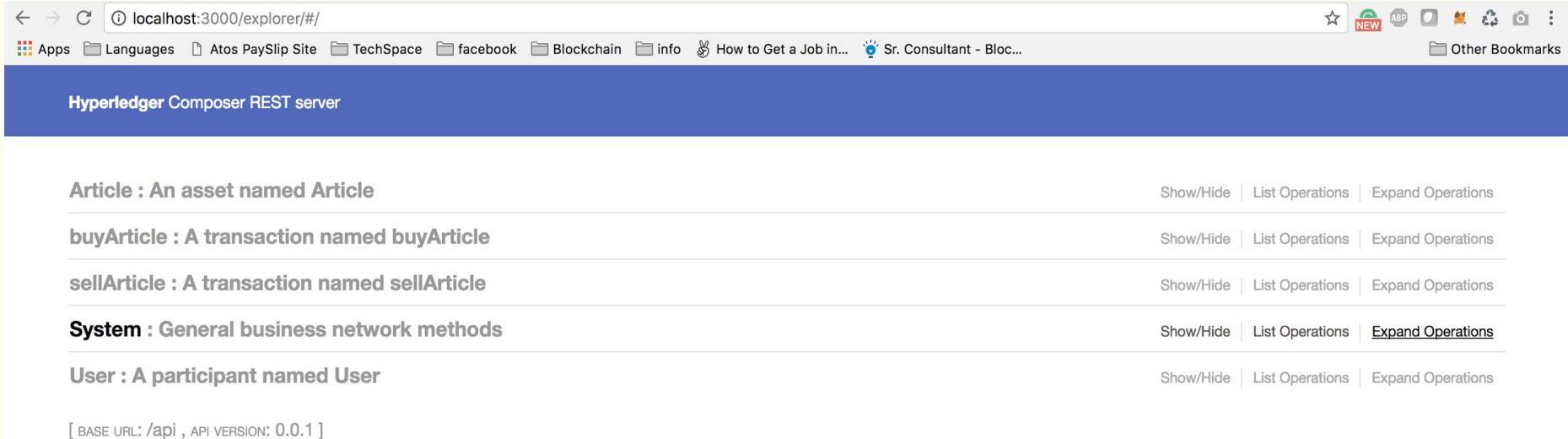
—

# REST Demo

Tools Used - Node.js, Composer Rest Network

URL - <http://localhost:3000/explorer/>

## Screen shot



The screenshot shows a web browser window with the address bar displaying `localhost:3000/explorer/#/`. The browser's bookmark bar includes links to 'Apps', 'Languages', 'Atos PaySlip Site', 'TechSpace', 'facebook', 'Blockchain', 'info', 'How to Get a Job in...', and 'Sr. Consultant - Bloc...'. The main content area of the browser shows the 'Hyperledger Composer REST server' interface. This interface has a blue header bar with the text 'Hyperledger Composer REST server'. Below the header, there is a list of resources, each with a title and three action links: 'Show/Hide', 'List Operations', and 'Expand Operations'. The resources are: 'Article : An asset named Article', 'buyArticle : A transaction named buyArticle', 'sellArticle : A transaction named sellArticle', 'System : General business network methods', and 'User : A participant named User'. At the bottom of the interface, there is a status bar that reads '[ BASE URL: /api , API VERSION: 0.0.1 ]'.

Resource	Show/Hide	List Operations	Expand Operations
Article : An asset named Article	Show/Hide	List Operations	Expand Operations
buyArticle : A transaction named buyArticle	Show/Hide	List Operations	Expand Operations
sellArticle : A transaction named sellArticle	Show/Hide	List Operations	Expand Operations
System : General business network methods	Show/Hide	List Operations	Expand Operations
User : A participant named User	Show/Hide	List Operations	Expand Operations

[ BASE URL: /api , API VERSION: 0.0.1 ]

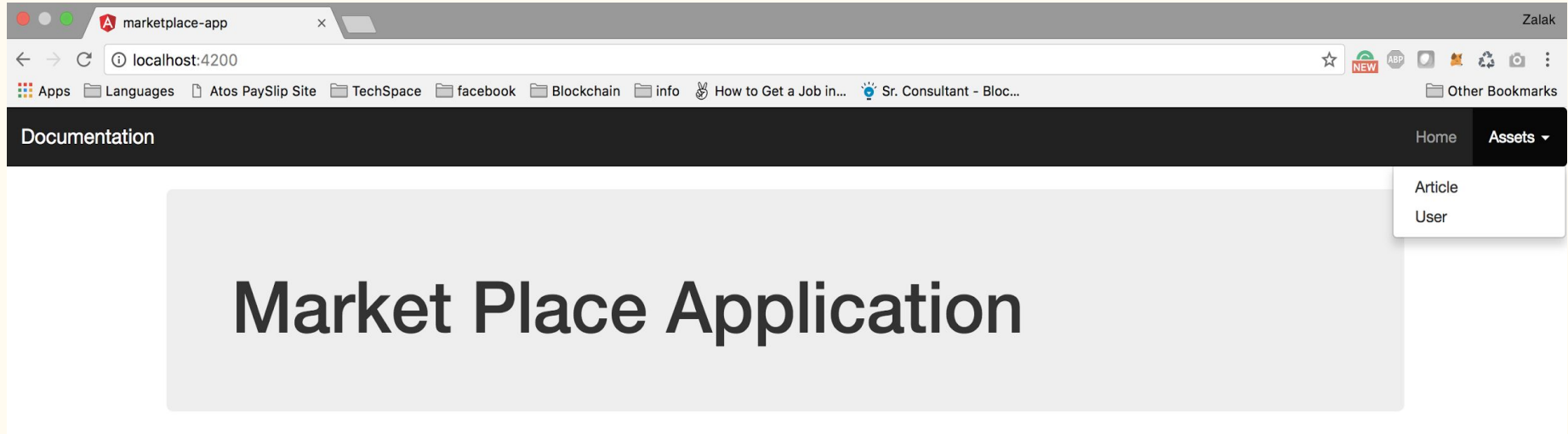


# Front-end Demo

Tools Used: Angular, Node, Composer

URL - <http://localhost:4200/>

- **Screenshot**



# Problems faced

—

# Research

- **Cloud setup and its Integration**
- **Security certification problems using local instances and docker images**
- **Docker Images management**
- **Angular Js Integration**

# Resources



# Research

- **HyperLedger Fabric Documentation**  
(<https://hyperledger-fabric.readthedocs.io/en/release/>)
- **Hyperledger Fabric Architecture**  
(<https://hyperledger-fabric.readthedocs.io/en/release/arch-deep-dive.html>)
- **Installation of Fabric using BlueMix** (<https://ibm-blockchain.github.io/setup/>)
- **Creation of Application on Hyperledger Fabric using Composer Docker Images**( <https://hyperledger.github.io/composer/introduction/introduction.html>)
- **Creation of Application using Local HyperLedger Fabric instances**([http://hyperledger-fabric.readthedocs.io/en/release/build\\_network.html](http://hyperledger-fabric.readthedocs.io/en/release/build_network.html))

# Conclusion

- Built Using: HyperLedger Fabric, Composer, Node.js, Angular.js, Docker, REST
- Next Steps:
  - Handling the Event notifications
  - Bring the Hyperlegder Blockchain Distributed Ledger on the front end.
  - Adding the Channels to the Blockchain
  - Error Handling

I Think this is  
awesome  
experience