Project Hygieia : A project based on Organ Donation

Here our main aim is to build a system to overhaul the current obsolete one. The aim of our current system is to make an Organ Donor list and an Organ Recipient list which will be store in a distributed data system i.e as blocks. It removes any middle men/ coordinators involved in the current system and thus removes human intervention

Getting started

`git clone <repo/branch name>`

`cd Hygieia`

`npm install`

`ng build`

After successful build, you should see a Hygieia folder inside the dist folder Open nginx.conf file in the repository, replace '/home/athil/Documents/Project_Hygieia/Hygieia/dist' with this folder path.

`apt-get install nginx`
After successful install, replace the nginx.conf file in /etc/nginx/nginx.conf with the one in previous step

(`sudo cp nginx.conf /etc/nginx/nginx.conf`)

Restart nginx using `/etc/init.d/nginx restart`
(if the port is still in use:
get PID using `sudo netstat -tulpn`
Do `sudo kill -2 <PID>` and then do the restart
)

`sudo docker-compose up`

The application should be available at `http://localhost:4200`

Sawtooth **Hygieia**
example of a Sawtooth application used in Healthcare

Introduction
This is a minimal example of a Sawtooth application,
with a javascript based transaction processor and corresponding client.

This example demonstrates a simple use case, wherein a user can add their name into a donor list and also a hospital has the ability to share a patient's details with the administrator by storing it in blocks.

Here the users can be of two types, they include:

1.Hospital side: Any number of patients stored as individual form, encompassing data includes patients name, age, blood group, specific organ they require, contact information.They are then grouped and added into individual blocks.

2.Donor List: Here the Web App inputs information by the independent willing members of society donating the organs. The information stored are grouped and the added into blocks so as to create blocks that are immutable and tamper proof.

All Hygieia transactions have the same 6 hex digit prefix, which is the first 6 hex characters of the SHA-512 hash of "hygieia" (that is, 'dcd58a').

The Hygieia count is stored at an 70 hex digit address derived from:

1. a 6-hex character prefix (the "hygieia" Transaction Family namespace) and
2. then by adding a individual independent hash of blood type and name of organ in case of donor.
3. Similarly in recipient side for creating addresses.

Components
The hygieia transaction family contains two parts :

I)Client Application ,written in Angular

II)Transaction Processor ,written in JavaScript

1. The client application has two parts:

* `angular application` : contains the client application that can make transactions to their validator through REST API

2. The Transaction Processor in javascript.

Transaction Processor is a generic class for communicating with a validator and routing transaction processing requests to a registered handler.

index.js has the Transaction Processor class.

The handler class is application-dependent and contains the business logic for a particular family of transactions.

HygieiaHandler.js and MatchHandler has the handler class.

The javascript transaction processor has the following files :

a)index.js (transaction processor class)
b)package.json
c)Dockerfile
d)HygieiaHandler.js (handler class)
d)MatchHandler.js (handler class)

Files which needs modification

a)docker-compose.yaml - the name of the transaction processor needs to be specified.
b)Dockerfile-the working directory of the tp needs to be specified.

Files newly included

a)package.json
b)index.js
c)HygieiaHandler.js
d) MatchHandler.js

Docker Usage
Prerequisites
This example uses docker-compose and Docker containers. If you do not have these installed please follow the instructions here: https://docs.docker.com/install/

OS Specifications
The preferred OS environment is Ubuntu Linux 16.04.3 LTS x64.
Although other Linux distributions which support Docker should work.

Building Docker containers
Before starting the project make sure the Docker service is up and running.
To start up the environment, perform the following tasks:

a)Open a terminal window.
b)Change your working directory to the same directory where you saved the Docker Compose file.
c)Run the following command:

$sudo docker-compose up --build

The `docker-compose.yaml` file creates a genesis block, which contain initial Sawtooth settings, generates Sawtooth and client keys, and starts the Validator, Settings TP, Hygieia TP, and REST API.

To stop the validator and destroy the containers, type `^c` in the docker-compose window, wait for it to stop, then type

```
$sudo docker-compose down
```