Scientific Research Publishing

Journal of Computer and Communications, 2021, 9, 38-55
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Highly Secure Residents Life Event Management System Based on Blockchain by Hyperledger Fabric

Ragouguelaba Agoda Koussema, Hirohide Haga

Graduate School of Science and Engineering, Doshisha University, Kyotanabe, Japan
Email: jeanpierre.agoda.koussema@gmail.com, hhaga@mail.doshisha.ac.jp

How to cite this paper: Koussema, R.A. and Haga, H. (2021) Highly Secure Residents Life Event Management System Based on Blockchain by Hyperledger Fabric. *Journal of Computer and Communications*, **9**, 38-55.
https://doi.org/10.4236/jcc.2021.99003

Received: August 9, 2021
Accepted: September 14, 2021
Published: September 17, 2021

Abstract

This article describes the design and implementation of Residents Life Event Management System (hereinafter called as RLEMS) with high level reliability and security by blockchain technology. The data access environment provided by blockchain is highly secure and trustworthy. In Blockchain system, some data fragments are grouped into one piece called as *blocks*, and all blocks are connected to create a chain of blocks in database. When blocks are connected, hash value is used to connect blocks properly. Blockchain technology enables highly secure and reliable data management system under relatively poor ICT environment. For example, developing countries such as African countries do not have sufficient ICT environment. Therefore adopting blockchain technology is suitable for such countries. Based on this consideration, we have started to build RLEMS on the blockchain system. In previous work, we used the MultiChain as a blockchain platform. However, as MultiChain platform is mainly for private blockchain system, it is not suitable for government-level data management system. Therefore, we tried to use another blockchain framework. We selected Hyperledger Fabric which was developed by Linux Foundation. It enables to implement all styles of blockchain system. This article describes the design and implementation of RLEMS by using Hyperledger Fabric. Furthermore, to provide the best user experience, we also built the web application interface with Java web application framework named PrimeFace. The implementation of a prototype revealed that the Hyperledger Fabric blockchain technology is more suitable than MultiChain.

Keywords

Blockchain, Secure Database, Life Event Management, Web Application

1. Introduction

The principal subject of our research is to design and implement highly secure and reliable data management system under poor ICT (Information and Communication Technology) environment.

Maintaining trusted data about people, organizations, properties, and activities is an essential in all organizations such as government. Local, states, and national authorities are responsible for keeping track of details such as birth and death dates, marital status, business licenses, property transactions, and illegal activity of people. Even in advanced countries, managing and utilizing these data properly are difficult. Some documents are only available on paper, and residents must often visit appropriate official sections to update his/her data.

Traditional data protection solutions such as passwords, firewalls, and data encryption are designed to keep hackers out of organizations and data stores. But how can we keep our data, especially most valuable assets (contracts, property titles, account statements, and so on) from being altered or even removed by people who can access our systems illegally? Especially residents' life event records are one of the most fundamental data in any country. Some organizations such as governments in developing countries are facing the difficulty of data management problem. They don't have enough communication infrastructures such as high-speed data communication lines. Data exchange system cannot work properly with poor communication infrastructure. Some advanced technologies exist to implement a highly secure and reliable system for data management. But under relatively poor ICT environment, implementing such advanced data management system requires new approach for realization.

In order to tackle these problems, we start our research about highly secure and reliable data management system under poor ICT environment. Adopted technologies are blockchain and web-based interface. Even in developing countries, wireless communication networks such as cell-phone and smartphones are widely used due to cheap implementation cost. Therefore, using web-based interface on smartphones enables people to access data management system easily. In a previous work [1], we reported the implementation of the web-based RLEMS with blockchain framework. In previous system, we used the MultiChain [2] as a platform to implement blockchain function. MultiChain is easy-to-use and light-weight framework for implementing blockchain system. However, MultiChain mainly supports private blockchain network and this kind of blockchain network is not always suitable for building nation-level data management system. Therefore, we decided to select another framework to implement the nation-level RLEMS. Based on the several consideration, we have selected Hyperledger Fabric [3] as a platform of revised system.

This article consists as follows: Section 2 describes the basic understanding of blockchain, Section 3 is about brief introduction of Hyperledger Fabric which we adopted in the revised system. In Section 4 we describe the proposed system. In Section 5, we summarize this article.

2. Basics of Blockchain

2.1. Basic Concept

Roughly speaking, blockchain is a database which contains various kinds of data such as text, images, motion pictures and others. Any type of data can be stored in blockchain. However, there are several essential differences between conventional database and blockchain. From a theoretical perspective, a blockchain is regarded an append-only, distributed time-stamped data structure. Distributed peer-to-peer networks are made possible using blockchains, in which non-trusting users may communicate with one other, which is verified without the requirement for a trusted authority. Blockchain is distinct from standard relational and new "Not only SQL" (NoSQL) databases. In conventional databases, we can execute so-called CRUD, where CRUD means "Create, Read, Update, and Delete." However, blockchain simply allows for "Create" and "Read." Users may read them, but they will never change or remove anything once created. All transactions and activities on the blockchain are only added and are recorded on the database with exact time stamps.

Figure 1 is a conceptual illustration of blockchain system.

The fundamental concept of blockchain is relatively simple; it is a linked list-based database. As shown in Figure 1, there are several blocks in one blockchain. Each block contains some information such as

- Hash value of previous block,
- Data to be stored in the current block.

And

- Specific value named *nonce*.

Among them, only nonce is computable, which must hold the specific condition of the hash value of each block. Once block is connected to the specific blockchain, all data are unchangeable. Hash value of each block is computed using all data in the block. When users want to connect one block to existing blockchain, hash value of current block is computed. As hash value of current block is computed using all stored data in the block, once some part of data is

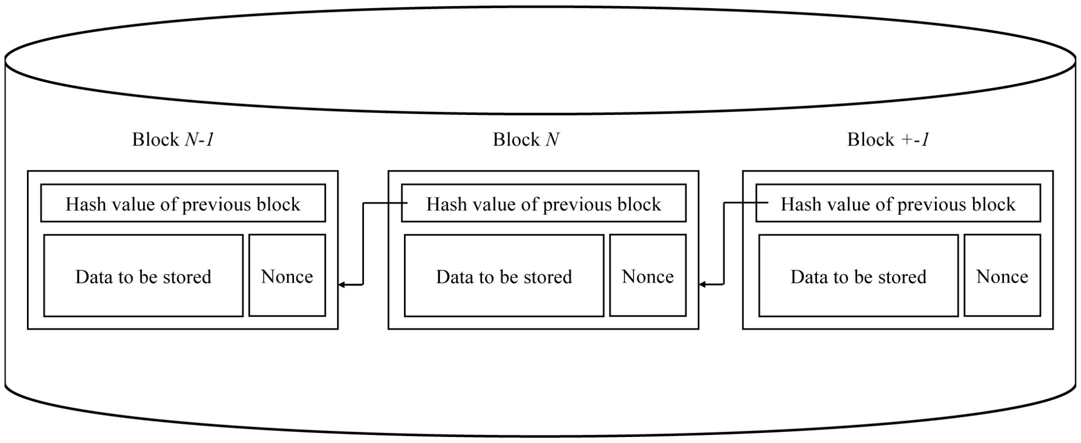


Figure 1. Conceptual illustration of blockchain.

modified, hash function will generate completely difference hash value and therefore it is very easy to find the falsification of data. In order to generate each block, the user must select appropriate value of *nonce* in each block and the adjustment of *nonce* requires huge amount of computing power. Therefore, falsification is virtually impossible. This is the main reason why blockchain provides highly secure and reliable data management. And blockchain has an attractive function which is very different from the traditional databases. Blockchain system is constructed from several nodes and every node stores all blockchain. In other words, data and its copies are stored in all nodes. This situation is often called as "Distributed Ledger" or **DL**. Therefore, even some accidents such as power failure or cracker's attack destroy the database of one node in the network, all data will be recovered by using the data stored in the other nodes. This is why blockchain has high robust nature.

2.2. How Blockchain Works

When a user wants to add data in the blockchain, user firstly packs several data fragments into one data package. For example, in case of Bitcoin, each transaction such as transferring money from one user to another is a data fragment. Certain amount of basic data fragments, say 20 fragments, will be packed into one package. After the finish of constructing one package, block construction will start. Each block contains several data such as data package, nonce and hash value of previous block. Each block must be connected to existing blockchain. To connect new block to exiting blockchain, hash value of each block must satisfy the specific condition. For example, if hash value is represented by 256 bits (32 bytes), hash value of first 64 bits (8 bytes) must be 0. To satisfy the condition, user must compute specific nonce value. To get such hash value, users will set appropriate *nonce* value. However, as hash function generates virtually random value, user must repeat computing hash value by assigning specific value to *nonce* for huge amount of times. It requires millions of computing time and power. Once a specific value of *nonce* which satisfies the condition of hash value is found, newly created block will be connected to existing blockchain. The reason why blockchain has highly secure and reliability is the amount of computing time. As mentioned above, to connect one block to blockchain, user must compute appropriate nonce value. For example, let n be the length of blockchain (the number of blocks in the blockchain), and the attacker modified the data in k-th block. When data in k-th block is falsified by someone, hash value of k-th block will change drastically. Therefore, the connection of blocks will be broken. If malicious attacker tries to connect modified block to blockchain, he/she has to find new nonce value which satisfies the condition of hash value. The change of hash value of k-th block affects following all $(n - k + 1)$ blocks; attacker must compute $(n - k + 1)$ nonce value again to connect all blocks in the blockchain and this re-computation requires quite a huge amount of computation. Therefore, falsification of blockchain is virtually impossible. This is why the blockchain system provides highly secure and reliable data.

2.3. Types of Blockchain

There are mainly three types of blockchains: public, private, and consortium blockchain. The features of each type can be summarized as shown below.

1) Public Blockchain: This type of blockchain is opened to public and anyone can participate as a node in the network. The users may or may not be rewarded for their participation. They are not owned by anyone and are publicly open for anyone to participate in. The best example of public blockchain is Bitcoin. Whenever a user transfers a transaction, it is reflected on everyone's copy of the block. Bitcoin uses a public blockchain.

2) Private Blockchain: As the name implies, private blockchain is private and is opened only to a consortium or group of individuals or organizations that have decided to share the ledger among themselves. Only the owner makes any changes to it because he has a right. Federated blockchains are faster (high scalability) and provide more transaction privacy.

3) Consortium Blockchain: Consortium blockchain is basically a hybrid of public and private blockchains. There are federated blockchains which are operated under the leadership of a group. In a group, there are two or more administrative nodes. Opposite to public blockchains, no one without the approval by administrative nodes is allowed to participate in the process of verifying transactions. For example, in the banking sector, this type of blockchains is mostly used.

Additional functionality may be offered to blockchains using the computer interface. From a practical perspective, a blockchain keeps a history of all transactions, which, for example, enables the computation of each user's balance. These are complicated states, which, e.g., transition from one to another once specified requirements are satisfied.

Network-controlled public, private, and federated blockchains all create three different categories based on their network rights. Anyone may join a public blockchain as a new user or node miner. In private blockchains, it is often preferable to utilize a whitelist of approved individuals with specified attributes and rights. A federated blockchain is a blend of public and private blockchain technologies with a collection of nodes that validate transaction processes instead of a single organization. We also examine issues such as the time it takes to approve a transaction, as well as security elements such as anonymity.

2.4. Sample Practical Application of Blockchain

Blockchain and digital ledger has been used to protect trusted records by financial institutions. It has been used, too, in the healthcare system to share and manage data. Healthcare data are highly sensitive, there is a need to protect it from an illegal access. Estonia, for example, uses a technology called Keyless Signature Infrastructure (KSI) [4] to keep all government data safe from any illegal access. In KSI, the concept of hash value plays an essential role in protecting data in Information System. KSI uses hash values, which are numeric values that uniquely reflect large quantities of data. The hash values are spread through a private network of government computers and stored in a blockchain [5]. In the work the authors implemented a decentralized record management system to handle electronic health records using blockchain technology [6]. Some authors recommended the use of smart contracts to handle clinical trial authorization information on a permissioned Ethereum blockchain and a private IPFS network to store the data structure [7].

3. Short Introduction to Hyperledger Fabric

3.1. Overall of Hyperledger Fabric

Hyperledger Fabric is an open source enterprise-grade blockchain platform. By using Hyperledger Fabric, users can implement distributed ledger technology (DLT). Hyperledger fabric can be used to develop from small private-level system to large enterprise-level system. It is developed by Linux Foundation. Now Linux Foundation manages the development and release of it. Hyperledge Fabric support mainly Consortium and Private blockchain. It can be used for general-purpose system development.

There are some features of the Hyperledger Fabric that distinguish it from other blockchain frameworks. This is what we need to know to keep things simple. Hyperledger Fabric is an approved blockchain. It is not intended to be accessible to the world, but as many individuals as you like can be added. It promotes smart contracts, much as Ethereum does. In Hyperledger Fabric, smart contract is called *chaincode* (the specific term for smart contract code from Hyperledger Fabric). The smart contracts of Hyperledger Fabric are written in Go language, Java and JavaScript. It is an approved blockchain. It is not intended to be accessible to the world, but as many individuals as you like can be added. It promotes smart contracts, much as Ethereum does.

There is a concept of "channels" in Hyperledger Fabric, where parties that are part of a blockchain may privately establish separate transactions and then send the final state to the main blockchain to be registered. In other blockchains, this is not unlike state networks, but there is an extra layer of privacy. All respondents have recognized identities, preserved by what Hyperledger Fabric calls "Membership Service Providers" (MSP). If you authorize a group of 10 hospitals to participate in your blockchain, the network will be aware of each of the 10 hospitals. This is Hyperledger Fabric's main feature that makes it work well with enterprise solutions.

To reach consensus, Hyperledger Fabric does not use traditional Proof of Work (PoW) or Proof of Stake (PoS) processes. Instead, it uses a sequence of checked transactions, since it is highly approved. Chaincode can only be used to approve a transaction if it is also checked by two parties who execute a transaction, each signing it and then peers who take the transaction. In short, what you need to think about for now is that for them to get included in the ledger, multiple checked participants need to sign transactions. This is a perfect match for blockchains focused on companies that don't have many participants. There is a natural barrier to malicious activity since the participants are familiar to each other. Here, read more. This is a basic flow diagram for transactions. For now, you do not need to comprehend all the specifics. Figure 2 shows the conceptual structure of Hyperledger Fabric blockchain network.

In this figure, the same color components correspond to each organization. There are several basic components in Hyperledger Fabric:

- **Peer:** Each peer contains chaincode and DB. Chaincode describes the smart-contract program.
- **Chaincode:** Chaincode is a kind of *application program* which implements the smart contract.
- **Orderer:** Orderer controls the order of transaction on the network.
- **CA:** CA stands for "Certificate Authority". This component gives the certificate of each transaction.
- **Client App:** This is an application program of each organization which uses the blockchain network.

Note that the difference of "chaincode" and "client app" is that chaincode describes the program to implement the smart contract which maintains the transaction logic. By using chaincode, each transaction must be certificated by the consensus of all peers. No certificated transaction will be added to DB. On the other hand, "client app" is an end-user application which uses the data in the blockchain database. This app is almost similar to the usual database application program. More details should be referred to [3] or other resources.

3.2. Hyperledger Fabric Workflow

Hyperledger fabric works as following order:

1) Create transaction proposal: The client application in the peer creates a proposal of transaction. This proposal is submitted to blockchain network.

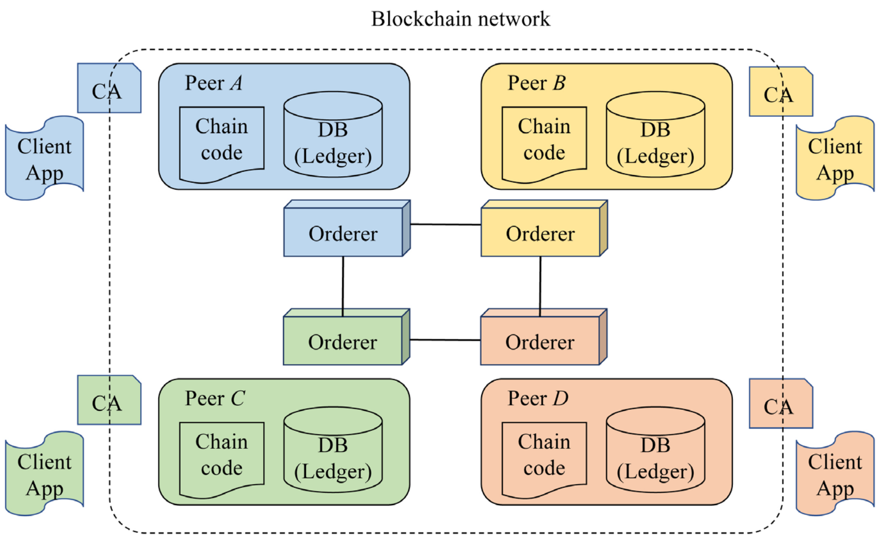


Figure 2. Conceptual structure of hyperledger fabric.

2) Consensus of transaction: After submitting the proposal of transaction to blockchain network, each peer checks if this transaction is valid or not by using chaincode. The results of all peers' chaincode are returned to the client.

3) Submission to orderer: After being approved by all peers, the proposal of transaction is sent to the orderer which orders the transaction into a block within the database.

4) Commitment of transaction: After finishing the ordering by orderer, the transaction will be sent to all peers to add the ledger.

Figure 3 is an illustrated explanation of Hyperledger Fabric workflow. As Hyperledger Fabric clearly separates the consensus algorithm and end-user application, it enables flexible system configuration.

3.3. Installing Hyperledger Fabric

Hyperledger Fabric can be installed on many kinds of machines. Support OS includes Linux, Windows, and macOS. Instruction steps are described many support sites such as [8]. After following few instructions, the implementor has nothing further to do. Some hardware and software requirements must be fulfilled to successfully install Hyperledger Fabric. Requirements are shown as follow:

- **OS:** Ubuntu 14.04/16.04LTS (both 64-bit), or macos 10.12 or greater version
- **Software tools:**

- cURL: the latest version
- git: the latest version
- Docker engine: version 17.06.2-ce or greater
- Docker-compose: version 1.14 or greater
- Go language: version 1.13.x
- Node: version 8.9 or later. Note that version 9 is not supported, and version 10 is supported from 10.15.30

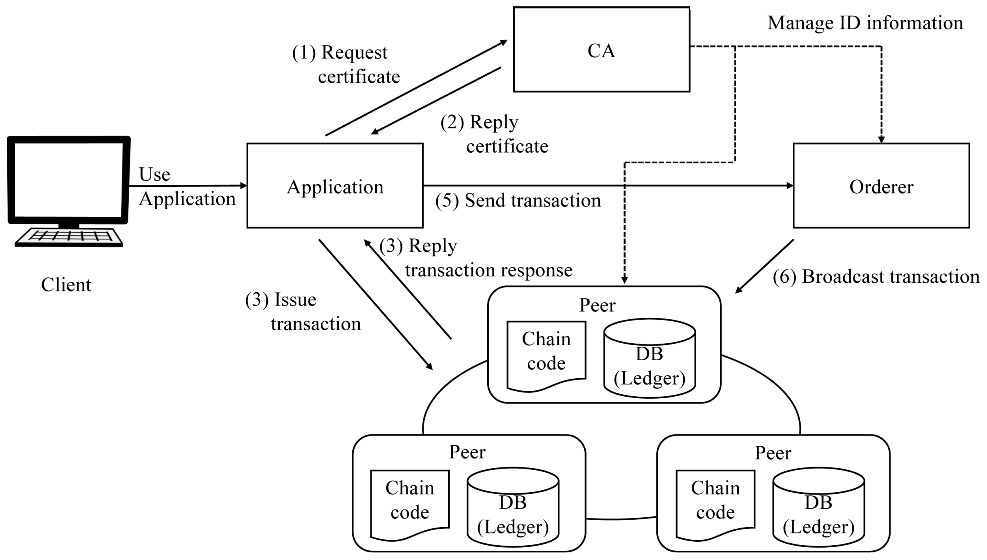


Figure 3. Illustration of hyperledger fabric workflow.

- npm: version 5.x
- Python: 2.7.x (3.x.x is not appropriate)

Install steps of Hyperledger Fabric on the Linux machine is as follow:

1) Install curl and the golang:

- sudo apt-get install curl
- sudo apt-get install golang

The package golang installs the Go distribution to /usr/local/go. The package should put the /usr/local/go/bin directory in your PATH environment variable. You may need to restart any open Terminal sessions for the change to take effect.

- export GOPATH=$HOME/go
- export PATH=$PATH:$GOPATH/bin

2) Install node.js, npm and Python:

- sudo apt get install nodejs
- sudo apt get install npm
- sudo apt get install python

3) Install Samples, Binary and docker images:

- curl -sSL https://bit.ly/2ysbOFE | bash -s
- curl sSL https://bit.ly/2ysbOFE | bash -s - fabric_version fabric-ca_version thirdparty_version
- curl sSL https://bit.ly/2ysbOFE | bash -s - 2.0.1 1.4.6 0.4.18

4) Export binary path:

- Go to FabricSamples directory
- Go to bin directory
- Add this path to $PATH variable in ~/.bash_profile
- Export PATH:$PATH:/Users/<your user name>/fabric-sampple_2.0/fabric-samples/bin

3.4. Invoking and Connecting Network with Hyperledger Fabric

In Hyperledger Fabric, blockchain is implemented as a virtual network of several nodes. A blockchain network offers a variety of infrastructure services to various applications, such as accounting and contracts. Primarily, smart contracts (chaincode) are used to record transactions on every peer node's ledger. End users may utilize either client applications or the blockchain network itself as clients. After installation, we start by implementing the first network which will have the structure shown in **Figure 4**.

There is just one node in Orderer at the level of nodes. In Org1 and Org2, each organization has two peers. Peers Peer0 and Peer1 are each known as peer. There are thus just five nodes in the First Network. When installed locally, certain components inside First Network are implemented as containers. Docker images are produced from Hyperledger Fabric's Docker images. In the localhost, we initially have five containers running. Typically, a Docker Compose-based container configuration is established using a suitable Docker Compose YAML

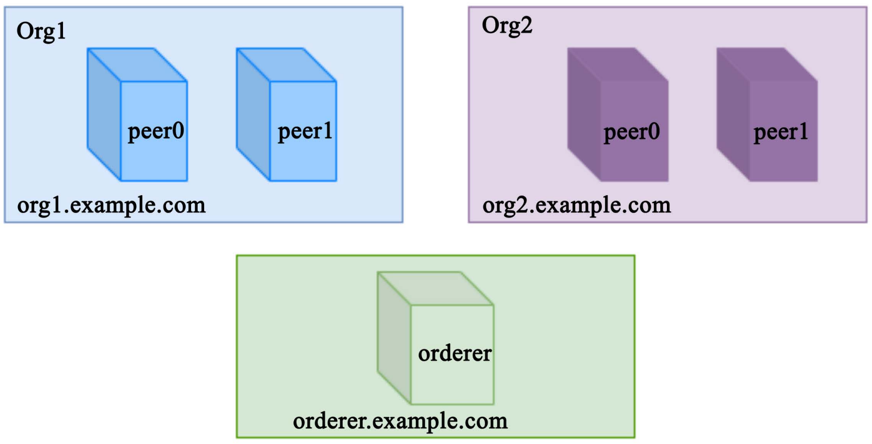


Figure 4. First network structure.

file. When we start working on the chaincode, there are more containers running.

In order to use blockchain network, the user must create the "channel". The channel is created such that each new channel starts with block 0. The Orderer creation has finished, and block 0 has been formed. Block 0 is created as a file called mychannel.block after the execution of create-artifacts.sh command shown in **Figure 5**.

With the mychannel.block, the peers join the channel. The command peer channel join foin for each peer joining the channel is shown in **Figure 6**.

Next step is anchoring peers. Anchoring means the allocation of "anchor" to each organization. Anchor is an access points between different organizations in blockchain network. Hyperledger Fabric provides shell script to allocate anchor. **Figure 7** is an execution result of shell-script.

The final step is an activation of network. After restarting machine, Hyperledger Fabric automatically awakes the network shown in **Figure 8**.

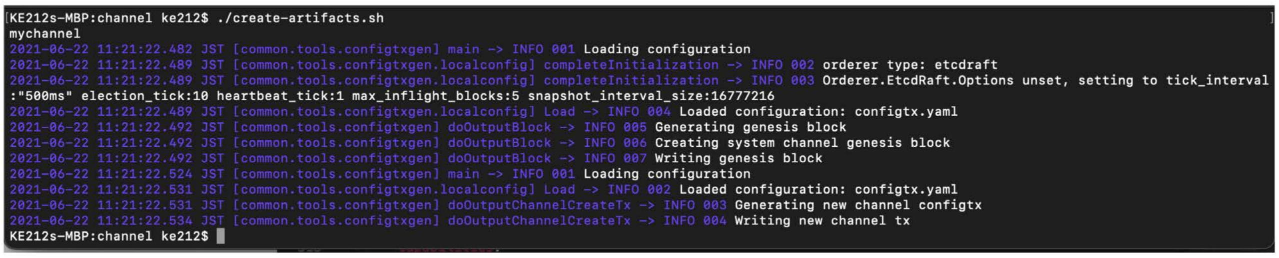
After installing Hyperledger Fabric platform, it is necessary to activate network and connect the external Java web application. To make this connection, we will use the Hyperledger Fabric Gateway SDK [9] which allows applications to interact with a Hyperledger Fabric blockchain network. **Figure 9** is the complete code sample from the official website. **Figure 9** is a part of sample code written in Java.

By executing this sample code, you can verify whether your installation is succeeded or not.

4. Design and Implementation of Revised RLEMS with PrimeFaces and MySQL

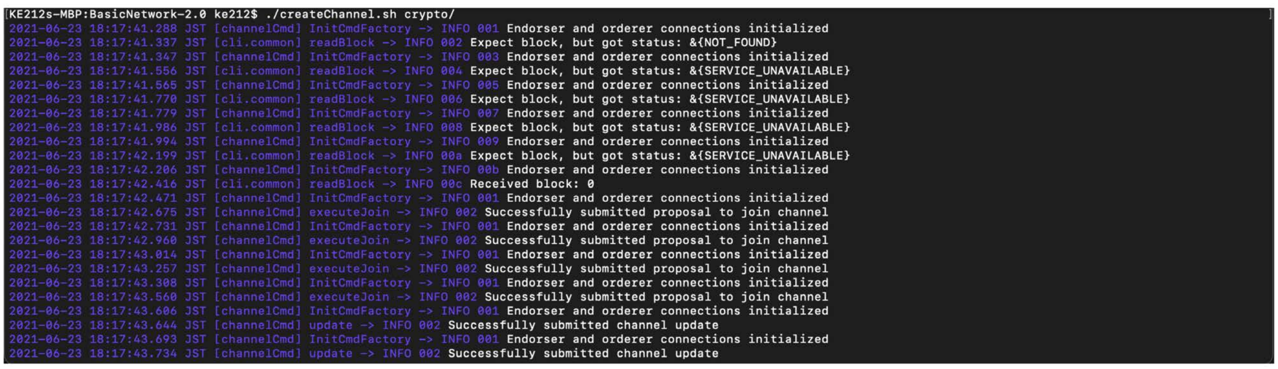
4.1. Overview

To launch our prototype system as a web application, we had to try to build it. We used Hyperledger Fabric blockchain, MySQL database system, and Java web framework PrimeFaces to develop our RLEMS as a web application. **Figure 10**



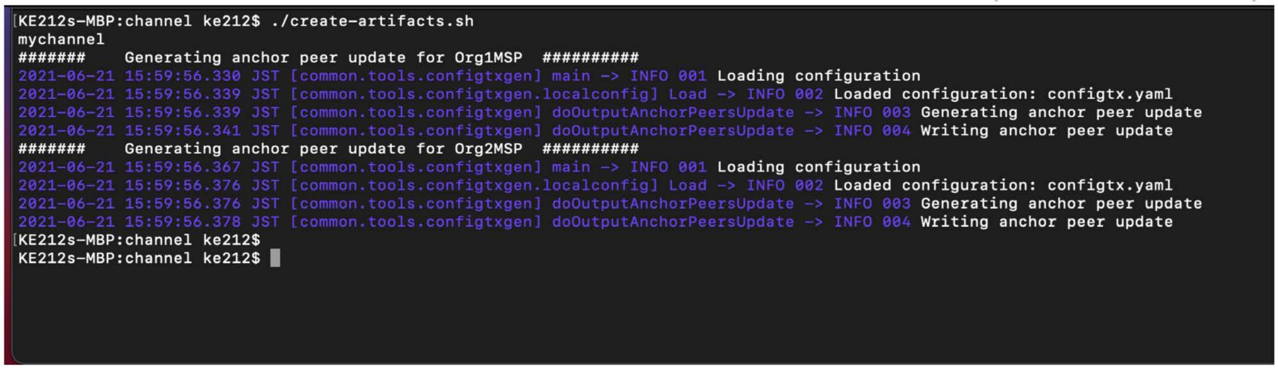
```
[KE212s-MBP:channel ke212$ ./create-artifacts.sh
mychannel
2021-06-22 11:21:22.482 JST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-06-22 11:21:22.489 JST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2021-06-22 11:21:22.489 JST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset, setting to tick_interval
:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2021-06-22 11:21:22.489 JST [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: configtx.yaml
2021-06-22 11:21:22.492 JST [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2021-06-22 11:21:22.492 JST [common.tools.configtxgen] doOutputBlock -> INFO 006 Creating system channel genesis block
2021-06-22 11:21:22.492 JST [common.tools.configtxgen] doOutputBlock -> INFO 007 Writing genesis block
2021-06-22 11:21:22.524 JST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-06-22 11:21:22.531 JST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: configtx.yaml
2021-06-22 11:21:22.531 JST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx
2021-06-22 11:21:22.534 JST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
KE212s-MBP:channel ke212$
```

Figure 5. Creating channel.



```
[KE212s-MBP:BasicNetwork-2.0 ke212$ ./createChannel.sh crypto/
2021-06-23 18:17:41.288 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:41.337 JST [cli.common] readBlock -> INFO 002 Expect block, but got status: &{NOT_FOUND}
2021-06-23 18:17:41.347 JST [channelCmd] InitCmdFactory -> INFO 003 Endorser and orderer connections initialized
2021-06-23 18:17:41.556 JST [cli.common] readBlock -> INFO 004 Expect block, but got status: &{SERVICE_UNAVAILABLE}
2021-06-23 18:17:41.565 JST [channelCmd] InitCmdFactory -> INFO 005 Endorser and orderer connections initialized
2021-06-23 18:17:41.770 JST [cli.common] readBlock -> INFO 006 Expect block, but got status: &{SERVICE_UNAVAILABLE}
2021-06-23 18:17:41.779 JST [channelCmd] InitCmdFactory -> INFO 007 Endorser and orderer connections initialized
2021-06-23 18:17:41.986 JST [cli.common] readBlock -> INFO 008 Expect block, but got status: &{SERVICE_UNAVAILABLE}
2021-06-23 18:17:41.994 JST [channelCmd] InitCmdFactory -> INFO 009 Endorser and orderer connections initialized
2021-06-23 18:17:42.199 JST [cli.common] readBlock -> INFO 00a Expect block, but got status: &{SERVICE_UNAVAILABLE}
2021-06-23 18:17:42.206 JST [channelCmd] InitCmdFactory -> INFO 00b Endorser and orderer connections initialized
2021-06-23 18:17:42.416 JST [cli.common] readBlock -> INFO 00c Received block: 0
2021-06-23 18:17:42.471 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:42.675 JST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
2021-06-23 18:17:42.731 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:42.960 JST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
2021-06-23 18:17:43.014 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:43.257 JST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
2021-06-23 18:17:43.308 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:43.560 JST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
2021-06-23 18:17:43.606 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:43.644 JST [channelCmd] update -> INFO 002 Successfully submitted channel update
2021-06-23 18:17:43.693 JST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-23 18:17:43.734 JST [channelCmd] update -> INFO 002 Successfully submitted channel update
```

Figure 6. Peer joining the channel.

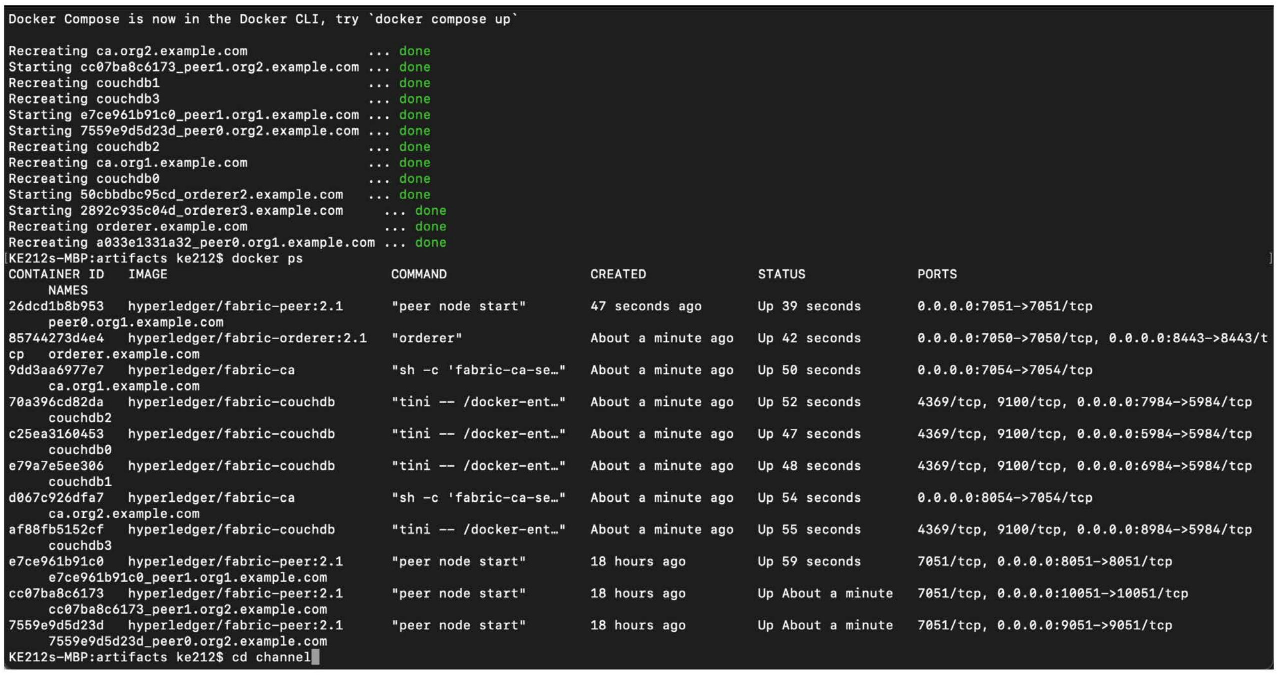


```
[KE212s-MBP:channel ke212$ ./create-artifacts.sh
mychannel
#######    Generating anchor peer update for Org1MSP  ##########
2021-06-21 15:59:56.330 JST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-06-21 15:59:56.339 JST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: configtx.yaml
2021-06-21 15:59:56.339 JST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2021-06-21 15:59:56.341 JST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
#######    Generating anchor peer update for Org2MSP  ##########
2021-06-21 15:59:56.367 JST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-06-21 15:59:56.376 JST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: configtx.yaml
2021-06-21 15:59:56.376 JST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2021-06-21 15:59:56.378 JST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
[KE212s-MBP:channel ke212$
KE212s-MBP:channel ke212$
```

Figure 7. Anchoring peers.

represents the overall structure of revised system. The structure of revised system is almost same as previous version.

Overall structure of this system is almost same as previous system. In a previous system, we used MultiChain as a framework of blockchain. In this new system, we used Hyperledger Fabric as a framework of blockchain. MySQL database system, and Java web framework PrimeFaces [10] to develop our RLEMS as a web application. In a previous system, we also used the PrimeFaces framework to implement web service interface. Current system also uses the same framework to enable the user interface implementation without changing it. Furthermore, thanks to the adoption of same DBMS and web framework, the number of new codes decreased. We only need to implement blockchain platform related software and some interface between web framework and Hyperledger Fabric.



```
Docker Compose is now in the Docker CLI, try `docker compose up`

Recreating ca.org2.example.com                ... done
Starting cc07ba8c6173_peer1.org2.example.com ... done
Recreating couchdb1                           ... done
Recreating couchdb3                           ... done
Starting e7ce961b91c0_peer1.org1.example.com ... done
Starting 7559e9d5d23d_peer0.org2.example.com ... done
Recreating couchdb2                           ... done
Recreating ca.org1.example.com                ... done
Recreating couchdb0                           ... done
Starting 50cbbdbc95cd_orderer2.example.com   ... done
Starting 2892c935c04d_orderer3.example.com     ... done
Recreating orderer.example.com                 ... done
Recreating a033e1331a32_peer0.org1.example.com ... done
KE212s-MBP:artifacts ke212$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED              STATUS              PORTS
     NAMES
26dcd1b8b953   hyperledger/fabric-peer:2.1        "peer node start"        47 seconds ago       Up 39 seconds       0.0.0.0:7051->7051/tcp
     peer0.org1.example.com
85744273d4e4   hyperledger/fabric-orderer:2.1     "orderer"                About a minute ago   Up 42 seconds       0.0.0.0:7050->7050/tcp, 0.0.0.0:8443->8443/t
cp   orderer.example.com
9dd3aa6977e7   hyperledger/fabric-ca              "sh -c 'fabric-ca-se…"   About a minute ago   Up 50 seconds       0.0.0.0:7054->7054/tcp
     ca.org1.example.com
70a396cd82da   hyperledger/fabric-couchdb         "tini -- /docker-ent…"   About a minute ago   Up 52 seconds       4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp
     couchdb2
c25ea3160453   hyperledger/fabric-couchdb         "tini -- /docker-ent…"   About a minute ago   Up 47 seconds       4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
     couchdb0
e79a7e5ee306   hyperledger/fabric-couchdb         "tini -- /docker-ent…"   About a minute ago   Up 48 seconds       4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp
     couchdb1
d067c926dfa7   hyperledger/fabric-ca              "sh -c 'fabric-ca-se…"   About a minute ago   Up 54 seconds       0.0.0.0:8054->7054/tcp
     ca.org2.example.com
af88fb5152cf   hyperledger/fabric-couchdb         "tini -- /docker-ent…"   About a minute ago   Up 55 seconds       4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp
     couchdb3
e7ce961b91c0   hyperledger/fabric-peer:2.1        "peer node start"        18 hours ago         Up 59 seconds       7051/tcp, 0.0.0.0:8051->8051/tcp
     e7ce961b91c0_peer1.org1.example.com
cc07ba8c6173   hyperledger/fabric-peer:2.1        "peer node start"        18 hours ago         Up About a minute   7051/tcp, 0.0.0.0:10051->10051/tcp
     cc07ba8c6173_peer1.org2.example.com
7559e9d5d23d   hyperledger/fabric-peer:2.1        "peer node start"        18 hours ago         Up About a minute   7051/tcp, 0.0.0.0:9051->9051/tcp
     7559e9d5d23d_peer0.org2.example.com
KE212s-MBP:artifacts ke212$ cd channel
```

Figure 8. Network up.

```java
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.concurrent.TimeoutException;

import org.hyperledger.fabric.gateway.Contract;
import org.hyperledger.fabric.gateway.ContractException;
import org.hyperledger.fabric.gateway.Gateway;
import org.hyperledger.fabric.gateway.Network;
import org.hyperledger.fabric.gateway.Wallet;
import org.hyperledger.fabric.gateway.Wallets;

class Sample {
    public static void main(String[] args) throws IOException {
        // Load an existing wallet holding identities used to access the network.
        Path walletDirectory = Paths.get("wallet");
        Wallet wallet = Wallets.newFileSystemWallet(walletDirectory);

        // Path to a common connection profile describing the network.
        Path networkConfigFile = Paths.get("connection.json");

        // Configure the gateway connection used to access the network.
        Gateway.Builder builder = Gateway.createBuilder()
                .identity(wallet, "user1")
                .networkConfig(networkConfigFile);

        // Create a gateway connection
        try (Gateway gateway = builder.connect()) {

            // Obtain a smart contract deployed on the network.
            Network network = gateway.getNetwork("mychannel");
            Contract contract = network.getContract("fabcar");

            // Submit transactions that store state to the ledger.
            byte[] createCarResult = contract.createTransaction("createCar")
                    .submit("CAR10", "VW", "Polo", "Grey", "Mary");
            System.out.println(new String(createCarResult, StandardCharsets.UTF_8));

            // Evaluate transactions that query state from the ledger.
            byte[] queryAllCarsResult = contract.evaluateTransaction("queryAllCars");
            System.out.println(new String(queryAllCarsResult, StandardCharsets.UTF_8));

        } catch (ContractException | TimeoutException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 9. Sample code of connecting Hyperledger Fabric to application.

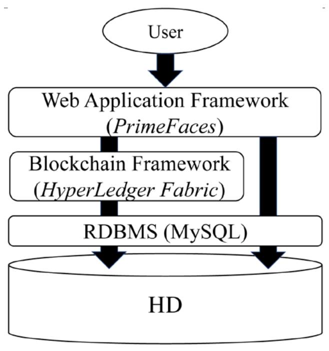

User
Web Application Framework (*PrimeFaces*)
Blockchain Framework (*HyperLedger Fabric*)
RDBMS (MySQL)
HD

Figure 10. Structure of revised system.

By using MySQL and blockchain technology, the data will transmit and receive from and to the user interface. We have our application, database, and blockchain platform all together in one system. The Java web framework is often used to do such tasks as inserting, updating, and deleting data on the blockchain. Data that must be sent to the blockchain technology and MySQL database may be sent using the Hyperledger Fabric framework's module. The data is encrypted and hashed to protect it from outside access. No need to fear since this data can be transferred over the internet. The information inside the data may only be given to the other entity that has an authorization from the blockchain network. The core part of this system is the web application built using the Java web platform. First, we try it out. A user must be logged in. After successfully logging in, the user is allowed to add, edit, or remove information. This new blockchain module is meant to be incorporated into Multichain, which will govern the information flows.

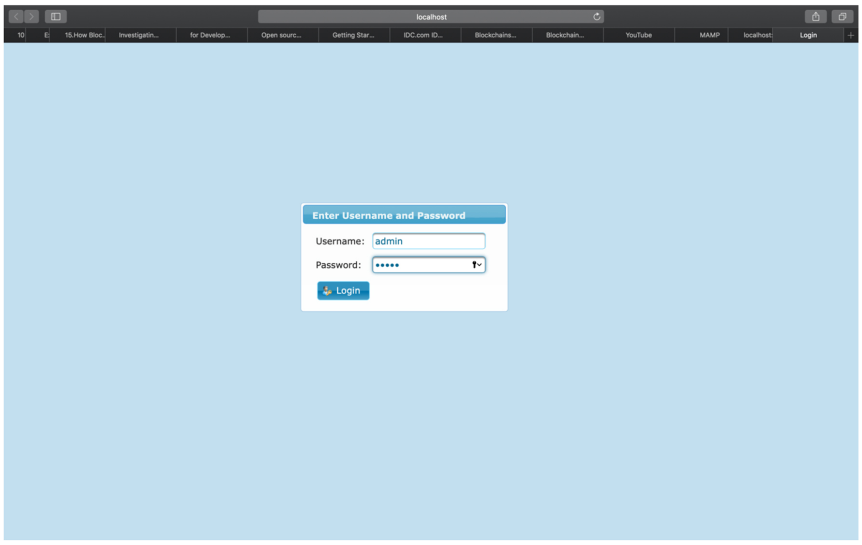
The tests below are examples of communication between the web application and MySQL database. Authentication must be established at the beginning of the web application. When we have the consideration that in mind, our intention is to get a registration for births in the national database. You'll need to provide information about the kind of hall, the hall's name, a prefecture, an area, and information about the infant to get a birth certificate at the end. A static of birth by area, by prefecture, and by hall will be created using the online application.

4.2. Implementation of Revised RLEMS

As we mentioned before, our system is implemented as a web application and we used the same web application framework. Therefore, from user's point of view, almost no change of user interface is found. For example, Figure 11 is a top page (authentication) of our system.

Figures from Figures 12-15 represent the workflow of the usage of revised system. Then we can get the official birth certificate document shown in Figure 16.

This workflow shows that our new system works well. As we keep using same web application framework, no change of user interface is found and it enables users to use new system smoothly.



Username: admin

Password: •••••

Figure 11. Authentication window.

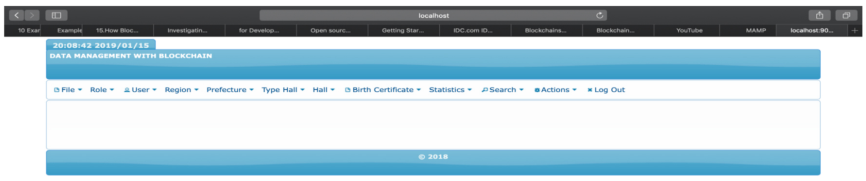
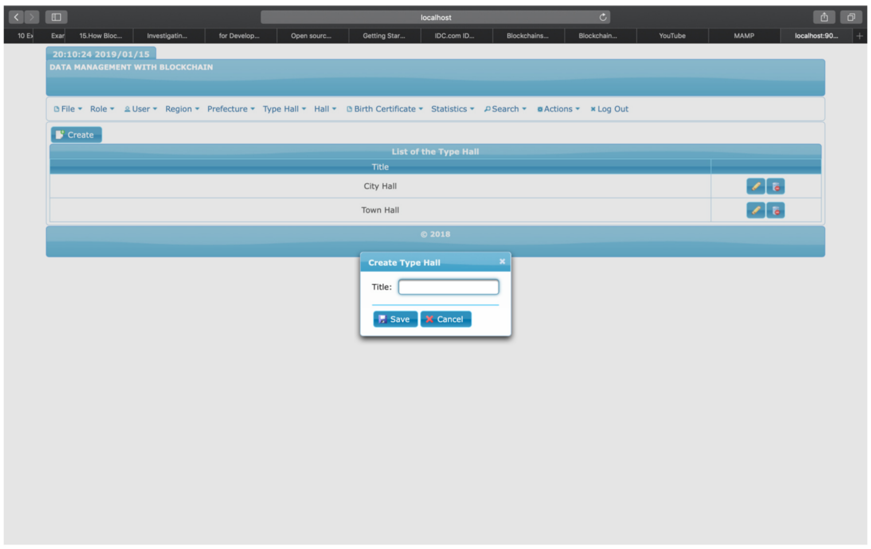


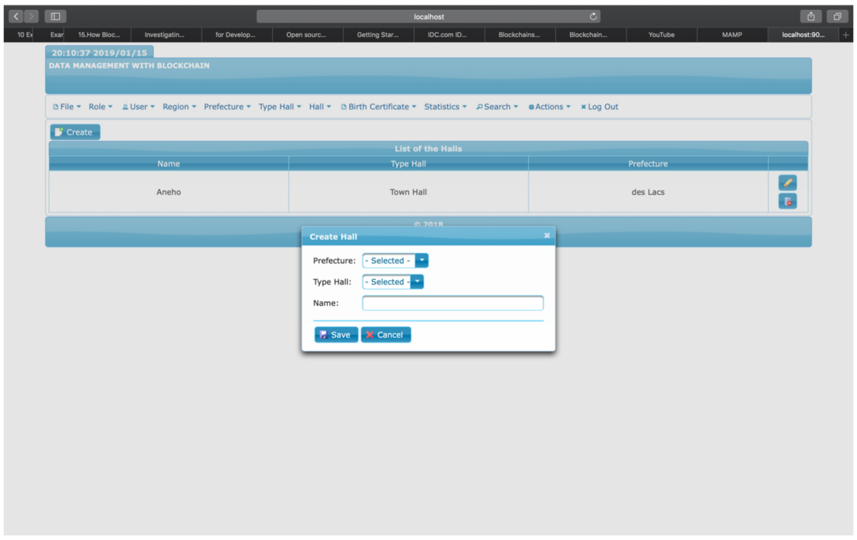
Figure 12. Main menu.



List of the Type Hall

Title
City Hall
Town Hall

Figure 13. Window for updating data in the database.

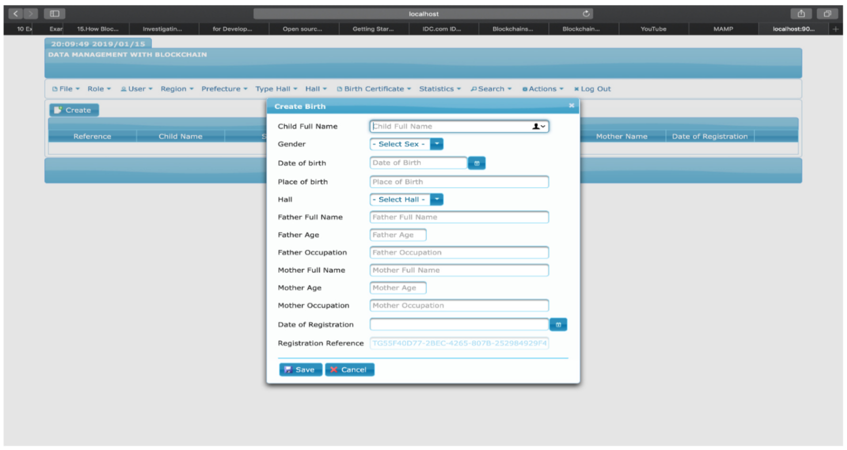

localhost
20:10:37 2019/01/15
DATA MANAGEMENT WITH BLOCKCHAIN
File Role User Region Prefecture Type Hall Hall Birth Certificate Statistics Search Actions Log Out
Create
© 2018
Create Hall
Save
Cancel

List of the Halls

Name	Type Hall	Prefecture
Aneho	Town Hall	des Lacs

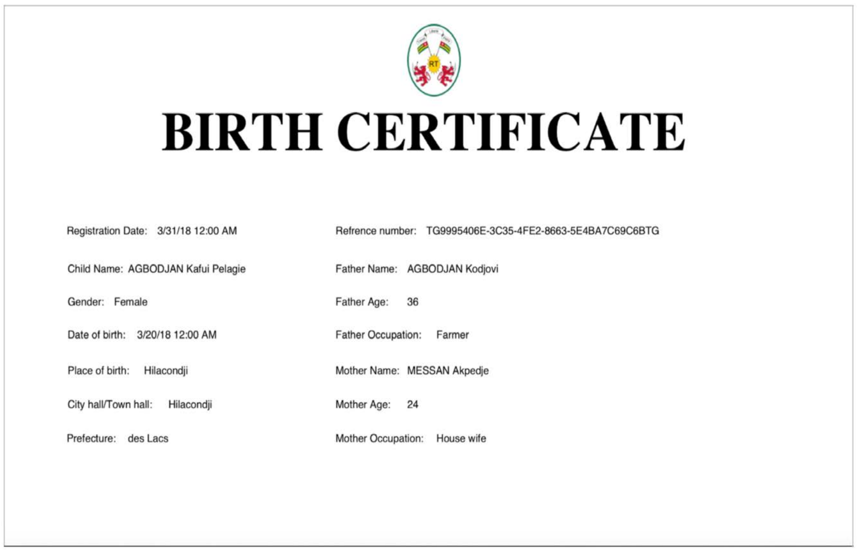
Prefecture: - Selected -
Type Hall: - Selected -
Name:

Figure 14. Window for updating data in the database.


localhost
20:09:49 2019/01/15
DATA MANAGEMENT WITH BLOCKCHAIN
File Role User Region Prefecture Type Hall Hall Birth Certificate Statistics Search Actions Log Out
Create
Reference
Child Name
Mother Name
Date of Registration
Create Birth
Save
Cancel

Child Full Name: Child Full Name
Gender: - Select Sex -
Date of birth: Date of Birth
Place of birth: Place of Birth
Hall: - Select Hall -
Father Full Name: Father Full Name
Father Age: Father Age
Father Occupation: Father Occupation
Mother Full Name: Mother Full Name
Mother Age: Mother Age
Mother Occupation: Mother Occupation
Date of Registration:
Registration Reference: TG55F40D77-2BEC-4265-807B-252984929F4

Figure 15. Entering user information to get user's official record.



BIRTH CERTIFICATE

Registration Date: 3/31/18 12:00 AM	Refrence number: TG9995406E-3C35-4FE2-8663-5E4BA7C69C6BTG
Child Name: AGBODJAN Kafui Pelagie	Father Name: AGBODJAN Kodjovi
Gender: Female	Father Age: 36
Date of birth: 3/20/18 12:00 AM	Father Occupation: Farmer
Place of birth: Hilacondji	Mother Name: MESSAN Akpedje
City hall/Town hall: Hilacondji	Mother Age: 24
Prefecture: des Lacs	Mother Occupation: House wife

Figure 16. Generated official birth certificate.

4.3. Chaincode of Our Application

As Hyperledger Fabric provides the flexibility of application development by using the "chaincode", which is a kind of application program description. By developing the appropriate chaincode for each application, users can implement their required application flexible on Hyperledger Frabric. Figure 17 describes the part of our chaincode written in Java. The total line of our chaincode is approximately 2000 lines.

4.4. Comparison of MultiChain and Hyperledger Fabric

Revised system uses Hyperledger Fabric as a platform of blockchain. Comparing to MultiChain, which was used in previous system, Hyperledger Fabric has following advantages:

1) Flexibility of organization structure: Basically MultiCahin is for the development of private blockchain system. On the other hand, Hyperledger Fabric is not restricted to private blockchain system. It supports consortium blockchain, which is suitable for our application. Furthermore, Hyperledger Fabric

```java
import org.hyperledger.fabric.contract.Context;
import org.hyperledger.fabric.contract.ContractInterface;
import org.hyperledger.fabric.contract.annotation.Contract;
import org.hyperledger.fabric.contract.annotation.Default;
import org.hyperledger.fabric.contract.annotation.Info;
import org.hyperledger.fabric.contract.annotation.Transaction;
import static java.nio.charset.StandardCharsets.UTF_8;

@Contract(
        name = "MyDataContract",
        info = @Info(
                title = "Data␣contract",
                description = "A␣very␣basic␣contract␣",
                version = "0.0.1-SNAPSHOT"))
@Default
public class MyDataContract implements ContractInterface {
    public  MyDataContract() {
    }
    @Transaction()
    public boolean myDataExists(Context ctx, String myAssetId) {
        byte[] buffer = ctx.getStub().getState(myDataId);
        return (buffer != null && buffer.length > 0);
    }
    @Transaction()
    public void createMyData(Context ctx, String myDataId, String value) {
        boolean exists = myAssetExists(ctx,myDataId);
        if (exists) {
            throw new RuntimeException("The␣data␣"+myDataId+"␣already␣exists");
        }
        MyData data = new MyData();
        data.setValue(value);
        ctx.getStub().putState(myDataId, asset.toJSONString().getBytes(UTF_8));
    }
    @Transaction()
    public MyData readMyData(Context ctx, String myDataId) {
        boolean exists = myDataExists(ctx,myDataId);
        if (!exists) {
            throw new RuntimeException("The␣data␣"+myDataId+"␣does␣not␣exist");
        }
        MyData newData = MyData.fromJSONString(new String(ctx.getStub().getState(myDataId),UTF_8));
        return newData;
    }
    @Transaction()
    public void updateMyData(Context ctx, String myDataId, String newValue) {
        boolean exists = myDataExists(ctx,myDataId);
        if (!exists) {
            throw new RuntimeException("The␣data␣"+myDataId+"␣does␣not␣exist");
        }
        MyData data = new MyData();
        data.setValue(newValue);
        ctx.getStub().putState(myDataId, data.toJSONString().getBytes(UTF_8));
    }
    @Transaction()
    public void deleteMyData(Context ctx, String myDataId) {
        boolean exists = myDataExists(ctx,myDataId);
        if (!exists) {
            throw new RuntimeException("The␣data␣"+myDataId+"␣does␣not␣exist");
        }
        ctx.getStub().delState(myDataId);
    }
}
```

Figure 17. Part of our chaincode.

clearly separate the blockchian system and application development. Especially "smart contract", which is the program triggered by the transaction in the blockchain or any information imported from outside of the system. Hyperledger Fabric support this "smart contract" by "chaincode", and MultiChain introduced the "smart filters" from version 2.0. The structure of "smart contract" of Hyperledger Fabric is clearer than that of MultiChain. Therefore, Hyperledger Fabric supports more flexible and easy to develop functions.

2) Flexibility of implementation language: Both of these two platforms support Java and JavaScript. Hyperledger Fabric supports Go language. But there is not so serious difference. Developing application with both of these platforms has no serious difference.

3) Richness of support information: Hyperledger Fabric is supported by IBM, Intel, NEC and other major venders. Therefore finding support information such as implementation, developing application, operations, and learning courses is easy. Comparing Hyperledger Fabric, MultiChain has less support information. Therefore, learning, implementing and operation is easier in Hyperledger Fabric.

5. Conclusion and Future Works

We implemented a high-reliability and secure data management system using blockchain in this article. With the help of blockchain technology, a dependable and secure data management system may be easily implemented. Our prototype attempted to find out how dependable and safe the blockchain technology is when it comes to data management. Furthermore, we used the user interface for the online application. There is no need to make any application installations on consumers' personal computers or cellphones when using a web application strategy. By using this strategy, the security of data management is maintained. Future work includes speeding up processing. Currently, the computational time required to link one block to an outgoing blockchain increases due to the necessity to search for an adequate nonce value. The system must handle more requests quicker than it does presently to keep up. GPU parallel processing will be possible with this. GPU computing allows massively parallel processing. Another future work option is to use Java to construct a blockchain from scratch.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Koussema, R.A. and Haga, H. (2020) Design and Implementation of Highly Secure Residents Management System Using Blockchain. *Journal of Computer and Communications*, **8**, 67-80. https://doi.org/10.4236/jcc.2020.89006

[2] Greenspan, G. (2015) Multichain Private Blockchain White Paper. https://www.multichain.com/download/MultiChain-White-Paper.pdf

[3] Baset, S., Desrosiers, L., *et al.* (2018) Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer. Packt Publishing, Birmingham.

[4] Nagasubramanian, G., Sakthivel, R.K., Patan, R., *et al.* (2020) Securing E-Health Records Using Keyless Signature Infrastructure Blockchain Technology in the Cloud. *Neural Computing & Applications*, **32**, 639-647. https://doi.org/10.1007/s00521-018-3915-1

[5] Cheng, S., Daub, M., Domeyer, A. and Lundqvist, M. (2017) Using Blockchain to Improve Data Management in the Public Sector. McKinsey Digital. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/using-blockchain-to-improve-data-management-in-the-public-sector

[6] Mayer, A.H., da Costa, C.A. and da Rosa Righi, R. (2019) Electronic Health Records in a Blockchain: A Systematic Review. *Health Informatics Journal*, **26**, 1273-1288. https://doi.org/10.1177/1460458219866350

[7] Steichen, M., Fiz, B., Norvill, R., Shbair, W. and State, R. (2018) Blockchain-Based, Decentralized Access Control for IPFS. 2018 *IEEE International Conference on Internet of Things* (*iThings*) *and IEEE Green Computing and Communications* (*GreenCom*) *and IEEE Cyber*, *Physical and Social Computing* (*CPSCom*) *and IEEE Smart Data* (*SmartData*), Halifax, NS, 30 July-3 August 2018, 1499-1506. https://doi.org/10.1109/Cybermatics_2018.2018.00253

[8] Chandran, J. (2020) Install and Configure Hyperledger Fabric v1.4 in Ubuntu 18.04.3 LTS. https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html

[9] https://github.com/hyperledger/fabric-gateway-java

[10] PrimeFace: Getting Started. https://www.primefaces.org/gettingstarted