

**NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS**

Faculty of Computer Science
Bachelor's Programme 'HSE University and University of London Double Degree
Programme in Data Science and Business Analytics'

Software Project Report (Final)

on the topic Implementation of a business object interface for blockchain application(EOS)

Fulfilled by the Student:

group #БПАД¹⁹³

03.06.2021

Date

D Sok
Signature

Sokolov Dmitrii Mikhailovich

Surname, First name, Patronymic, if any

Checked by the Project Supervisor:

Боломов Павел Дмитриевич

Surname, First name, Patronymic (if any), Academic title (if any)

старший программный архитектор

Job

АО Райффайзен Банк

Place of Work (Company or HSE Department)

Date 03.06. 2021

10

Grade according
to 10-point scale

Боломов

Signature

*Материалы проекта допущены НРА
Боломов / Боломов П.Д.*

Moscow 2021

Contents

1	Abstract	2
2	Key Terms and Definitions	2
2.1	Block chain related terms	2
2.2	Selected Technologies terms	3
2.3	Project Specific Terms	3
3	Selected/developed methods and algorithms	3
4	Project approaches and implementation	4
4.1	Tools and Instruments	4
4.2	Description of Project Implementation	4
4.2.1	Yandex Cloud	4
4.2.2	Database	5
4.2.3	Blockchain Network	6
4.2.4	Smart Contract	6
4.2.5	Adapters	7
4.2.6	Postgres Adapter	7
4.2.7	EOS Adapter	9
5	Additional Feature	9
5.0.1	User Application	9
6	Key conclusions	11
7	Field for Further Improvement	12
8	Bibliography	12

1 Abstract

Over the last ten years, Distributed Ledger Technology has grown at such a quick speed that it has completely revolutionized the way people do business, handle personal information, and keep track of it. As a result, this technology is gaining popularity, and it's a fantastic road to go with a lot of intriguing future possibilities.

DLT offers a lot of promise since, unlike a distributed database, there is no central administrator and all participants have a shared copy of the ledger. Because the ledger is shared, the complexity of transactions is greatly reduced, and transaction processing is much accelerated. It not only provides resilience since each member has a copy of the ledger, ensuring that there is no single point of failure, but it also provides transparency since updates require consent from a large number of other members inside the system.

Our objective was to create a DLT platform for conducting business activities on the EOSIO block chain platform.

2 Key Terms and Definitions

2.1 Block chain related terms

Block chain : It is a continuous sequential chain of blocks, built according to certain rules, containing information.

Smart Contract : Algorithm over the block chain, which is executed by the formation, control or provision of information about something within the system (the state of any data stored in the block chain).

Multi index structure : Structure which enables the construction of containers maintaining one or more indices with different sorting and access semantics.

cleos : cleos is a command line tool of eosio.

nodeos : Nodeos is the core EOSIO node daemon. Nodeos handles the blockchain data persistence layer, peer-to-peer networking, and contract code scheduling.

keosd : keosd is a key manager daemon for storing private keys and signing digital messages.

WASM : Web-Assembly Machine used to execute user-generated applications and code to blockchain.

2.2 Selected Technologies terms

DLT : Which stands for Distributed Ledger Technology is block chain communication technology for decentralized storage/modification of data.

EOSIO : A block chain platform to implement our project upon.

MVP : A product with minimal, but sufficient functions to satisfy the basic requirements.

MVC : Model-View-Controller is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user

2.3 Project Specific Terms

Business Process : Abstractly chosen work in system with various tasks for Raiffeisen/its partners.

DB Mirror : Basic Mirror API Database for connecting all parts of the project entirely.

DLT Adapter : A technical layer that moderates requests from the mirror database and DLT platform. They can be split into two:

- EOSIO Adapter
- Postgres Adapter

ABI : The Application Binary Interface (ABI) is a JSON-based description on how to convert user actions between their JSON and Binary representations.

3 Selected/developed methods and algorithms

In our project, we use three algorithms, one of which is the consensus algorithm, which is already employed by the EOSIO platform. It is based on the Delegated Proof of Stake algorithm (DPoS), which is a far superior alternative to the Proof of Stake method (PoS). The DLT uses this technique to find a consensus or widely acknowledged consensus among all member nodes.

This consensus can be used to configure digital wallet balances, token transfers, information being produced and saved, or identities/users being produced and saved. It is also considerably quicker and more efficient than PoS. Unlike PoS, where an attacker would need to obtain 51 % of the bitcoin within the system in order to attack it, the DPoS algorithm is significantly safer.

The second technique utilized in the project is the encryption algorithm, especially the Public Key Encryption method, which uses two keys: public and private keys to handle transactions for each user. In order to confirm transactions, public keys are exchanged and utilized in combination with respective private keys; this is done in the OpenSSL connection for file transference.

4 Project approaches and implementation

4.1 Tools and Instruments

psql : Terminal-based front-end for PostgreSQL.

pgAdmin 4 : A database management tool used for PostgresDB. It simplifies the creation, maintenance, and use of database objects by offering a clean and intuitive user interface. Using this with psql guarantees you the greatest experience and result. (we used psql for creating, editing, deleting databases and used pgAdmin 4 to see the final result)

EOS Studio : A graphic IDE for the easy creation of EOS based dApps (albeit the compiler is rather buggy).

Node.js : An open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside a web browser.

Yandex.Cloud : A cloud platform that provides private and corporate users with infrastructure services - Yandex Compute Cloud, Yandex Object Storage, etc.

Visual Studio Code : Code editor redefined and optimized for building and debugging modern web and cloud applications.

Xcode : Xcode is Apple's integrated development environment (IDE) for macOS, used to develop software for macOS, iOS, iPadOS, watchOS, and tvOS

4.2 Description of Project Implementation

4.2.1 Yandex Cloud

Raiffeisen supplied us with a server on Yandex.Cloud platform. The computer power is comprised of the following components: Intel Cascade Lake platform, 2 CPU cores, 8 gigabytes of RAM and 50 gigabytes of SSD drive. The operating system is Ubuntu 18.04.4 LTS. To obtain a remote access to our server via our local machine, one should set up an SSH connection. In order to do this, Oleg installed openSSH on a server. Then, have edited **sshd_config** file in order to allow password authentication. Finally, to connect we need to type

this command: `ssh host_ip_address` and we are in!

4.2.2 Database

Steve George Parakal and Oleg Ivanov created the Postgres database using pgAdmin4 in order to store data from deals in several tables such as members involved, deal status, and attributes to name a few (connections in the table were also created using the JOIN query).

Later on, this table would be backed up using the `pg_dump` command, and then Steve change the configuration files: `postgresql.conf` file to allow all addresses to be able to listen to the database instead of the default which is set to local host and also allowing remote TCP/IP connections, the next file which was changed was the `pg_hba.conf` file, Steve added some lines which would allow acceptance of remote connections from IPV4 addresses after providing a valid password (the md5 keyword), after restarting the Postgres service the remote connections were ready.

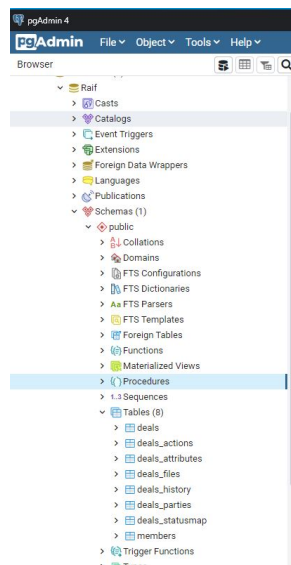


Figure 1: Database Tables

4.2.3 Blockchain Network

The foundation of that project is the blockchain network itself. It must be setted up on our Yandex cloud server. Process is quite simple, and require only working with linux terminal. First step, for me was to install EOSIO Binaries and EOSIO.CDT. Then set up wallet, start nodeos add necessary plugins, in our case there were history_plugin, http_plugin and chain_api_plugin, which allows connections for adapters on specified port , outputting blockchain's logs on specified port so adapters can fetch data from it, and for future monitoring, after that start keosd and add starting accounts to begin development.

4.2.4 Smart Contract

Regarding the smart contract, it consists of one header file and one C++ file, Smart-Contract deal combines the functionality of two smart-contracts: accounts and deals, account part is responsible for creating and adding (and declaring them) or deleting a user into/from the system. The deal part responsible for creating deals, adding/removing files from it, and update deal's status according to business logic of the application. List of users who is able to add deal and modify them is based on user's role which he is specifying when registering in system. Alongside with functions, 3 multi index tables were implemented to store data on blockchain which later will be mirrored in database : members(accounts registered in a system), deals and deal's history(stores changes in status of a deal).

The decision too combine functionality of two smart-contracts into one was made after I consider all possible variants:

- Best option, but the hardest one, is manual generation of ABI file out of contracts, which requires deep knowledge about EOS contract compilation, and is a rare practice.
- Worst option, is to deploy contracts to different accounts and set admin account's active to each account's active permission instead of key, which will go against business logic of platform and against security and whole purpose of blockchain in that project.

Choice of combining logic of two contracts into one, is justified that it's not going against business logic , it's easy and popular method to implement, and doesn't require spending countless time generating ABI file, if any changes to smart contract will be added.

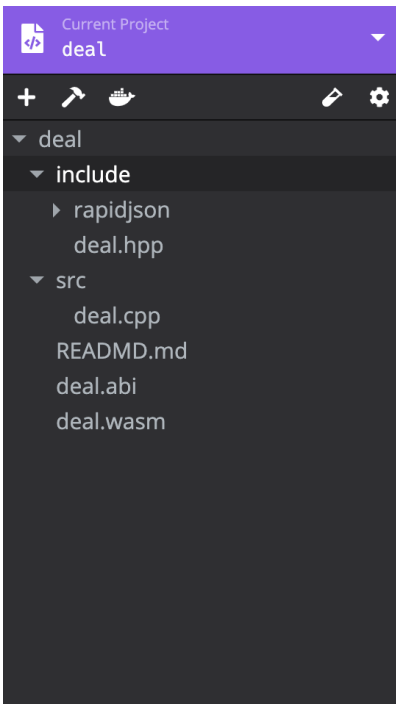


Figure 2: Smart Contract

4.2.5 Adapters

There are two adapters available: the Postgres adapter and the EOS adapter. The EOS adapter is primarily composed of three js files: `ObjectActionHandler.js` and two `index.js`. The former is responsible for receiving block data passed from an Action Reader and populating some external state deterministically derived from that data, as well as triggering some side effects or updates. The latter files create updaters and effects, while the other file instantiates the classes and combines the other two files. The Postgres adapter is intended to "listen" for and alert new deals by connecting to and accessing a database on a SQL server through javascript and making changes to data.


4.2.6 Postgres Adapter

The Postgres adapter was developed to connect to the Postgres database and use triggers. Triggers, as the name implies, are processes that are executed when an event happens in a database. One of the trigger functions was created to trigger a specific event, namely *notify_deal_event*, which is triggered when a new deal is added to the deals table, as the name implies. The trigger itself and the trigger function has been created using SQL queries, first the trigger function *notify_deal_event* and then the trigger itself *new_deal_trigger*:



```
1 CREATE OR REPLACE FUNCTION public.notify_deal_event()
2     RETURNS trigger
3     LANGUAGE plpgsql
4 AS $function$
5 BEGIN
6     PERFORM pg_notify('new_deal_event', row_to_json(NEW)::text);
7     RETURN NULL;
8 END;
9 $function$
10 |
```

Figure 3: The trigger function creation



```
1 CREATE TRIGGER new_deal_trigger AFTER INSERT ON deals
2 FOR EACH ROW EXECUTE PROCEDURE notify_deal_event();
```

Figure 4: The trigger creation

When a new deal is added to the deals table in database, the adapter receive a notification from database's trigger that it has been added along with a JSON of the new deal. Also ,Steve have created multiple other triggers and trigger functions for even more actions that are to be performed on tables in the database, for example adding of deal files, removing of deal files, updating of status.

I had created a javascript class called ApiService.js as seen in Figure 5(which is a part of Adapter) that works in tandem with my smart-contract on blockchain.

ApiService is a class consisting of async functions which accepts read values from the JSON Steve's code receives from triggers and perform actions on a blockchain basing on values it recieved like , which function to call on blockchain, who is active user, and values for smart-contract's functions.

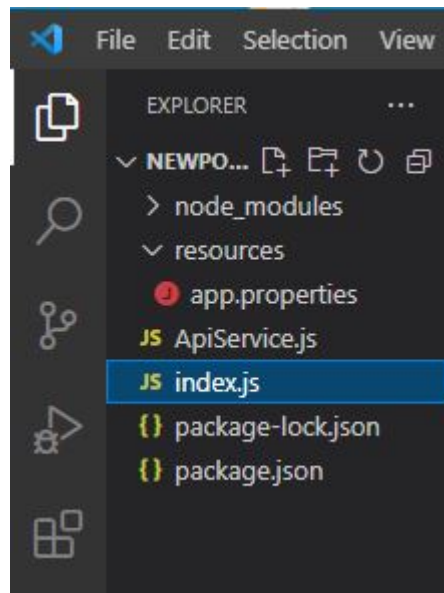


Figure 5: Postgres Adapter

4.2.7 EOS Adapter

Oleg Ivanov made the Eos adapter, which listens to nodes and parses data based on a certain action in blockchain. He used Demux-js together with their EOS blockchain class implementation: demux-js-eos. To explain how it works, if a user performs a transaction, the ActionReader class examines blocks being generated to determine if any transactions have occurred, and if so, it parses the action and passes the information onto the ActionHandler class object. Following that, his updaters and effect functions will be run. a newline a newline a newline a newline a newline a new The Updaters change the transaction's status, which the Effecters subsequently utilize to perform bespoke actions on the blockchain.

5 Additional Feature

5.0.1 User Application

To interact with DLT platform I developed simple IOS application, to allow users see list of their deals, add new one, and update it's status. MVC architecture was

chosen for that application , where MVC stands for Model-View-Controller.The application uses public library PostgresClientKit to establish connections to the Postgres Database server on Yandex Cloud as postgres has it's own specific protocols for connections , which are not included in any standard libraries.

Since the User Application is a front-end product, it does not reveal the technology we have used to create the DLT platform and hence a sample of the code used to create it can be shown.

In following screenshot presented part of Model of Application, which is responsible for establishing connection to server, pool/add/edit data in database, which is performed on background thread'.

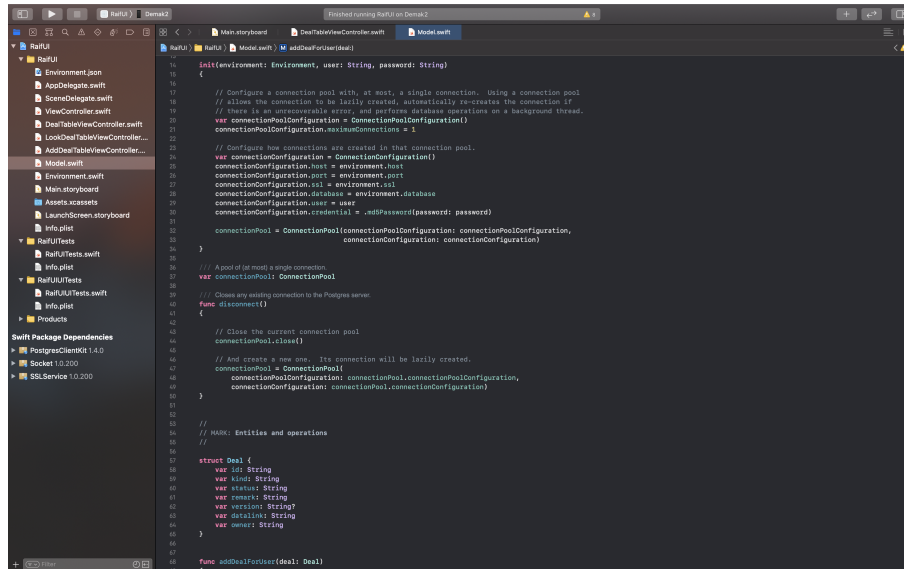


Figure 6: MVC

Figure 7 shows the Main storyboard (structure of user interface of the application), Please note that this is subject to change in future as this was made with meeting of basic features in mind, later on more functionality may be added to the app.

Figure 8 provides with real App view as for now , where main screen of the app (TableView) can be observed with list of deals fetched from the database , and on right side of the figure provided view with explicit details of the deal, which can be accessed by tapping the deal in TableView (NOTE : parameters of deal presented in figure 8 are not real, and made for demonstration purpose)



Figure 7: Main Story Board

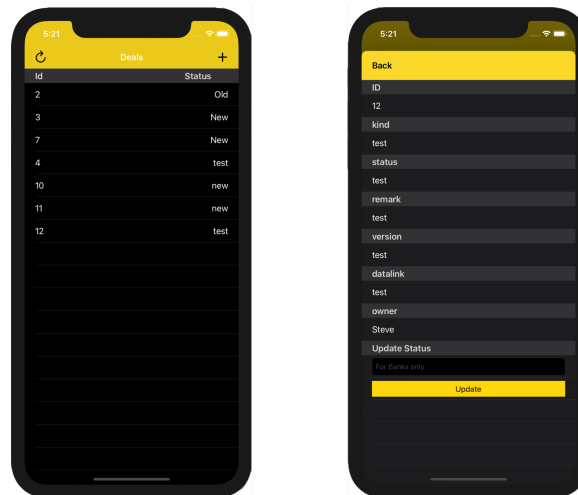


Figure 8: Application Interface

6 Key conclusions

Overall, a possible application of blockchain technology on a business platform

was implemented, almost all of the required procedures were followed including the creation of an extra additional feature that was created out of interest in seeing the changes of deals and information in one's own hand

7 Field for Further Improvement

In terms of further improvement to the platform, an IPFS system which stands for Interplanetary File System for a more decentralised way of sharing files (increasing security) could be implemented in the future and an OpenSSL connection for further security could be created to go along with it. With enough time, even a possible blockchain monitoring system can be created to view possible problems or changes of the network.

8 Bibliography

- [1] *EOSIO Technical White Paper v2*. URL: <https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md>.
- [2] *EOSIO Developer Portal*. URL: <https://developers.eos.io/welcome/latest/index>.
- [3] *Node.js Documentation*. URL: <https://nodejs.org/en/docs/>.
- [4] "Mirror Api Database Documentation". In: *Internal Raiffeisen System (confluence)* (2020).
- [5] *Docker Compose*. URL: <https://hard-skills.ru/notes/postgresql-install-docker-compose/>.
- [6] Brian Curran. *What is Delegated Proof of Stake Consensus? (DPoS) Complete Beginner's Guide*. 1 2018. URL: <https://blockonomi.com/delegated-proof-of-stake/>.
- [7] *PostgresClientKit Documentation*. URL: <https://codewinsdotcom.github.io/PostgresClientKit/Docs/API/Classes>.
- [8] *eosjs Documentation*. URL: <https://github.com/EOSIO/eosjs>.
- [9] *demux Documentation*. URL: <https://github.com/EOSIO/demux-js>.
- [10] *demux eosjs Documentation*. URL: <https://github.com/EOSIO/demux-js-eos>.