Seminar on BlockChain Technology

Report of the project ChainUrRents, presented by Luciano García-Bañuelos

Authors: Andreas Ellervee
Orlenys López Pintado

University of Tartu
January, 2017

General Flow Description

In this project, we had to implement a blockchain-based market application that enables people to rent properties online. The system stores and manages the information about all the properties that should be added by their corresponding owners. A client who wants to rent a property will access the list of available properties and he can pick the desired one. Both, user and owner must be registered in our system for taking part in contracts or for adding properties, in the case of owners.

Once a client selects a property, he sends a contract request to the owner who may accept or reject the proposal. When the owner accepts, the client must pay the rent of the first month and a deposit (50% of rental cost), followed by owners payment who sends a deposit on the same amount as the client. Before the owner's transaction, the contract may be canceled by both without fee (e.g. if the counterpart is slow to answer). After that, they also have 72 hours to cancel, but now a 50% of the deposit will be sent to the other part as compensation.

If no cancellation occurs, the system keeps waiting for the meeting where the client will receive the keys of his new house. When any of involved part incurs in a violation on the meeting data (e.g, some of them do not assist), the contract will finish and the affected part will receive the deposit from the other one. A third role was added in our system to act as referee who solves discrepancies between client and owner (deciding who is right), in cases as the previously described. If the meeting was a success, the application transacts the rental cost from the client to the owner who also gets his deposit back. The system preserves the client's deposit until he leaves the property when the rental period finishes. The user receives his deposit back when the contract ends, if everything was fine during the rental period, otherwise the deposit will go to the owner. At this point our referee role may act to solve any disagreement.

Application Description

Our solution was structured on two back-ends and one front-end applications.

The first back-end application was designed to guarantee the reliability of the transactions. We modeled the flow of the rental contract using BPMN 2.0[1] (see Figure 1) and we deployed the resulting model as a smart contract in an Ethereum Blockchain Network. Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference[2]. We implemented the smart contract in Solidity, a contract-oriented high level language designed to runs into the Ethereum Virtual Machine[3]. Each node in our model becomes a function in our smart contract. As exception, consecutive service activities (designed to be executed automatically) were joined into the same function to avoid ether consumption on unnecessary function calls. The smart contract validates and executes all the operations that involve money transactions between client and owner, offering the reliability of the Blockchain Network.

For testing our blockchain application we have to run testrpc that is a Node.js based Ethereum client for testing and development. It uses ethereumjs to simulate full client behavior and to develop Ethereum applications[4]. The test server offers 10 blockchain addresses with a default amount of ether, so client and owner must select one of these for the money transactions.

The second back-end application, was designed to store all the information about the users, roles, properties and contract requests handled by the system. For this application, we used MongoDB that is an open-source document database that provides high performance, high availability and automatic scaling[5]. This application avoids storing and managing information not related to money transactions, into the Blockchain Network, reducing the ether consumption (and transaction fees) only to the operations involved in the money transactions.

The web front-end application guarantees the interaction of the users with our rental system. It was implemented in AngularJS that offers a structural framework for dynamic web applications. It allows to use HTML 5 as template language and extend HTML syntax to express the application's components clearly and succinctly[6]. In addition, the implementation of the the controllers and services of the front-end

[1] http://www.bpmn.org/.

[2] https://www.ethereum.org/.

[3] https://solidity.readthedocs.io/en/develop/.

[4] https://github.com/ethereumjs/testrpc.

[5] https://www.mongodb.com/.

application was done in JavaScript. We utilized the web3 object provided by the JavaScript web3.js library to interact with our smart contract in Blockchain. This library works with any Ethereum node, which exposes an RPC layer, it communicates to a local node through RPC calls[7]. We used Angular's $resource for our API calls to the MongoDB back-end application. These services retrieve data in JSON format and then we can use it in our Angular controllers.

Code Repository

The code of the resulting application, instructions for running it and the BPMN model can be downloaded from:

https://bitbucket.org/andreasellervee/blockchain-seminar/overview

[6] https://angularjs.org/

[7] https://github.com/ethereum/wiki/wiki/JavaScript-API.

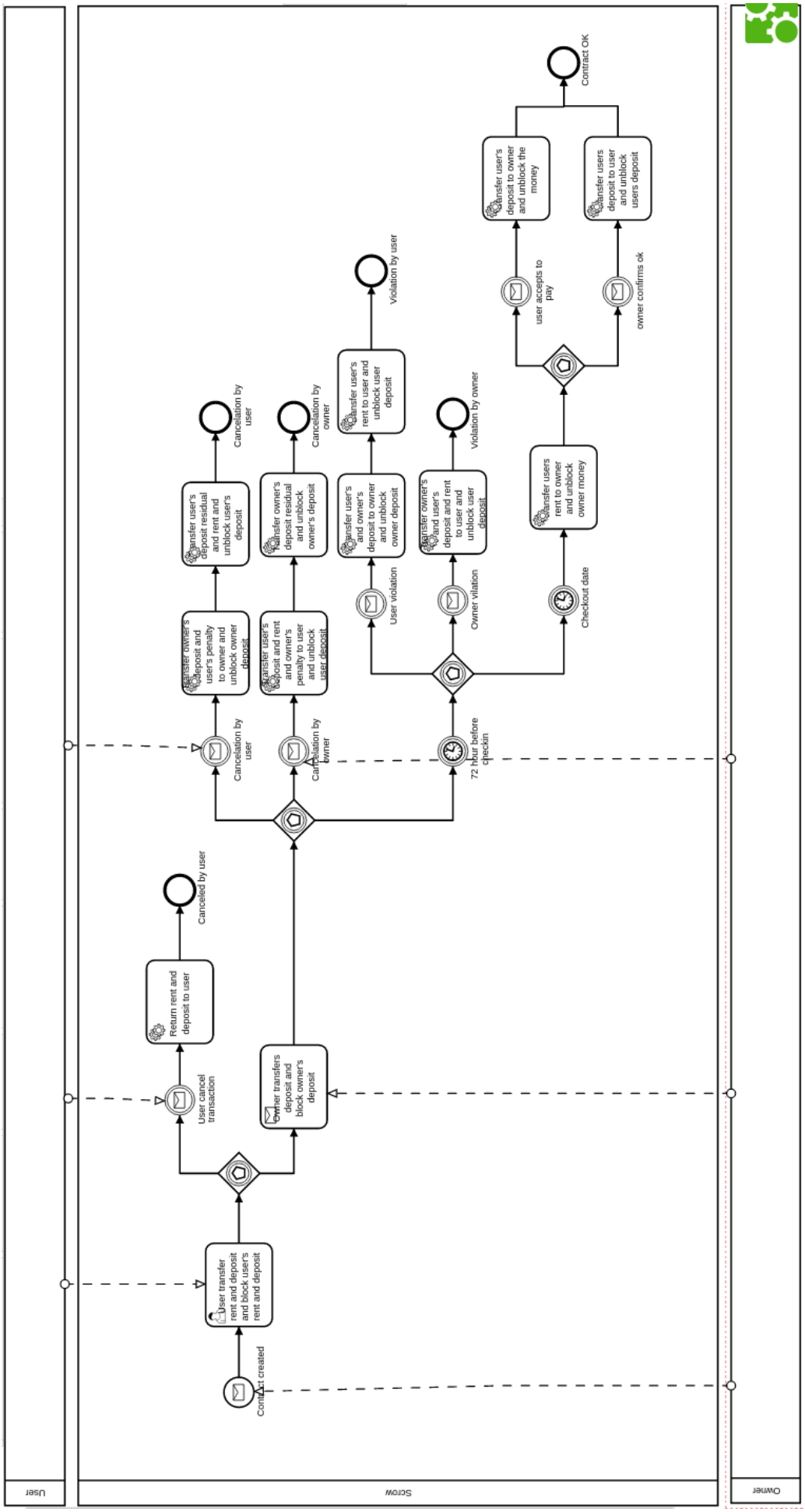

Figure 1: BPMN model of the smart contract.