Smart Contacts as APIs

Athanasios Priftis
Information Systems Department
HESSO / HEG-GE,
Geneva, Switzerland
e-mail: athanasios.priftis@hesge.ch

Joël Israel
Information Systems Department
HESSO / HEG-GE,
Geneva, Switzerland
e-mail: joel.israel@hesge.ch

Jean-Philippe Trabichet
Information Systems Department
HESSO / HEG-GE,
Geneva, Switzerland
e-mail: jean-philippe.trabichet@hesge.ch

Abstract— **Although blockchain protocols have existed for some time now, a focused analysis on smart contracts, as Application Programming Interfaces (APIs) for user driven and web based applications, is clearly missing. APIs as abstract interfaces can inspire us in designing smart contract based applications and information infrastructures. Such an approach has an impact both on the architecture and coding of applications. In this article, we will use our pilot on managing building rights within the City of Geneva to demonstrate how the architecture, design and implementation of smart contracts can be advanced. Initiating the creation of new applications and services based on the smart contracts characteristics, such as forced temporality and immutability and transparency, comes with new opportunities and challenges. Blockchain could be more than an innovative technology, a building block of new forms of social applications and infrastructures through the design of smart contracts as APIs.**

Keyword-Smart contracts; Blockchain pilot; APIs; Web APIs; information infrastrucure and application; user-generated content.

I. INTRODUCTION

The main purpose of our article is to examine a blockchain - smart contract infrastructure, inspired by APIs, in a real life pilot. This research and application effort, launched late 2018, is still in progress within the Geneva City administration. We will start the article presenting smart contracts in relation to the concepts of information infrastructures, in particular Application Programming Interfaces (API). An analysis will follow, presenting the work that is taking place, involving various actors: a research group of the University of Applied Sciences in Geneva, collaborating with several public administration departments of the Geneva State in the area of building rights management and house development, notably the Cantonal Office of Housing and Urban Planning (DALE) and selected private entities. The goal of this cross-organizational, action oriented, research effort is to co-produce a set of smart contracts, developed as APIs, facilitating the open and transparent execution of urban planning processes, while designing a multi-stakeholder governance infrastructure of smart contracts. This information infrastructure could set the basis for initiating hybrid, public – private, services in the future. Finally, we will discuss the importance of coding smart contracts as APIs. We will make appear some crucial characteristics of smart contracts as key elements both in the area of building rights management and the smart contracts' themselves.

This is how our paper is structured. In section II, we describe in more detail what a standard API is and discuss the sociotechnical aspect of information infrastructures, mainly in terms of public governance. In section III, we present how smart contracts create applications and spaces of social decision. Finally, in sections IV and V, we describe the, API driven, architecture of our information infrastructure, related to our pilot application, and discuss further work and challenges.

II. UNDERSTANDING APIS AS INFORMATION INFRASTRUCTURES

As already demonstrated in previous research efforts [1], we cannot rely on the modern disciplinary methods and frameworks of knowledge in order to think and interpret the transformative effect which new technology is having on our culture. It is precisely these methods and frameworks that modern technology requires us to rethink. Smart contracts as APIs can intersect the current state of opacity in application development and contribute to our understanding of semantic rules to user created applications.

An API can be understood as an abstract interface establishing parameters for computational exchange. These parameters can be accessed and incorporated for the creation of any number of possible interfaces. In other words, it acts as an interface, mainly by representing and defining the possible functions of the exposed information elements, in the form of tools that express, and make available, certain functions of these elements. In this way, an API creates a standardized method to facilitate forms of exchange between various information elements and computational agents to make them interoperable and independent of their respective implementations [2]. As mentioned, this is done through an API's establishing of specified procedures, typically through establishing parameters of access through the assigning of various identifiers, priorities and restrictions that can be operated upon within API-facilitated exchanges.

 ISBN: 978-1-61208-757-3

In regards to which information elements can and cannot be surfaced and shared, an API can be seen as both an entry point into the black box of a particular computational service, but also as a clearly defined possibility towards other possible exchanges with this service. At present, the term API refers specifically to the category of APIs known as Web-based APIs. A Web API encapsulates and specifies all of the valid messages that two or more computational parties can request and accept while communicating via network protocols [3].

Bucher [4] provides important understandings to the sociotechnical questions at play in API practices. They pertain to API-supported fields such as application ecosystems and social media platforms. They are better understood when combining insights from fields, such as software studies with ethnographic approaches into how developers produce and make sense of code in their work with APIs. The importance of APIs as both practical connective enablers and abstract infrastructures for networked computational practices is a key element of our analysis. By focusing on smart contracts as an API implementation in information infrastructures, we aim to give a few suggestions for how anyone working with them might think openness and terms of inclusivity set upon practices of sharing, participation and exchange.

Information infrastructures are closely linked to social innovation. They are considered as a significant part of Information and Communication Technologies (ICT) innovations, the development and study of which comprises both the technological components, as well as, the social aspects. The analysis of these information infrastructures includes technological characteristics, capabilities, interactions and negotiations between actors involved in their development. Information infrastructures have to be developed through a collaborative approach, as actors have to give up control some over their data and systems to realize mutual benefits, supported by governance mechanisms making this possible. The entire setting in which actors operate may change because of a social innovation [5]. This requires organizations to develop advanced social and collaborative capabilities, to be able to realize new modes of public governance. Social factors affect the development, adoption, change, operations, and stability of information infrastructures, as well as, the application and services linked to them [6].

In this context, learning from APIs while developing smart contracts, can be extremely useful in the following areas: a) designing and deploying the overall architecture of our application, b) understanding and explaining, the unique possibility of each smart contract as an API, serving a larger, user oriented information infrastructure and c) establishing user driven parameters negotiating the relationship between transparency, openness, business model and integration of systems. There areas are answering to questions such as what data is closed, what is open, what is made accessible, what is kept internal to a system, what is open to edition and how this possibility to edit is, actually, taking place.

Our approach provides more experience and results at this exact point. Smart contracts as APIs become an important element to give some sense on how data is being circulated, made accessible and inaccessible. Even more, they allow us to (re)think, and at times intervene, to the overall rules of governance of platforms and applications around us.

III. ON SMART CONTRACTS

Understanding smart contracts as applications and spaces of social decision making, needs a more detailed analysis. This is what this section attempts to do.

A. Smart Contracts and their design as applications

The term smart contract was introduced by Nick Szabo back in 1997 [7]. Smart contracts are self-executing computer programs that implement a set of functionalities. They are based on business rules and contractual agreements. Smart contracts, very much like APIs, can automate business logic by embedding, verifying, and enforcing the contractual clauses of an agreement without intervention from intermediaries. The main characteristics of smart contracts include machine readability and distributed code running on a blockchain platform. Smart contracts, similarly to APIs, can be part of an application program, but can also act autonomously for a predefined period distributed [8].

Blockchain technology established the ground for the implementation of smart contracts as pieces of code that consist of executable functions and state variables. Specifically the execution of a function changes the state of the variables according to related logic implementation. Nowadays, the Ethereum blockchain protocol [9], is the most widely used technological platform for the development of the smart contracts, using Solidity, an object oriented high-level language, as the implementation language.

The design of a smart contract consists of their conceptual and technical part. The latter requires the setup of the blockchain nodes, the definition of the business functions, the description of the processes between the users and the application template design for the definition of the smart contract. The conceptual design consists of the description and classification of business rules that will be extracted from information carriers (e.g., documents or code). The specification of conversion of extracted rules to smart contract functionalities using domain knowledge, formally represented as ontologies [10]. The extracted information gains semantic meaning from the exploitation and usage of standardized knowledge representation, such as ontologies and semantic rules. Adopting semantic rules incorporated into, and enforced by a smart contract, can be facilitated by using smart contract templates. The templates can serve as the skeleton for generating the final smart contract to be used in the blockchain network.

B. On Contracts, Smart Contracts and Social Decision making

As Dupont and Maurer argue, blockchain technologies differ from traditional social systems that validate, maintain and enforce contracts between people (e.g., accountancy and legal systems), because crypto-contracts tend to build social and functional properties within the system [11]. In other

words, where lawyers and judges are needed to enforce legal regulations and notaries are needed to validate certain legally binding contracts, the blockchain allows for the validation of smart contracts and their enforcement in its own right without the necessity for arbitrating third parties. This implies that in contrast with conventional contract laws, which are necessarily coupled with their human validators and enforcers, blockchain technologies are capable of establishing and maintaining forms of political organization that are (at least in the virtual realm) self-sustaining [12].

The decentralized enforcement of smart contracts "dematerializes" or rather depersonalizes the auditing authority: it eradicates the need for human arbitrators such as notaries or accountants. While traditional contracts can be described as textually expressed voluntary agreements between two or more contracting parties that require human arbitration to be validated, audited and enforced, a smart contract appears as a mechanism that can be made binding by means of computational scrutiny, without human interference. However, the work of contracting remains embedded in social interactions, namely the act of consenting to a specific contractual reality. The aspects that are delegated to the technology are the validation, storing and enforcement of the contractual clauses.

As an initial governance method for our case study, linked to smart contract and information infrastructure challenges, we envisage a consensus-driven approach. As Klievink and Janssen note, the consensus process is well suited for a society where technological and economic progress is within reach [13]. This approach has two clear advantages over existing alternatives. First, the initiation of an ongoing discussion with interested parties, as to achieve an early alignment on governance rules, mainly through a proposal and voting possibilities. Second, pilot participants gradually develop strong incentives to resolve conflicts early in the process to secure the viability of the application.

IV. PILOT: OPEN REGISTERS FOR BUILDING RIGHTS

In this section, we will examine the context of our pilot and describe the, API driven, architecture of our information infrastructure.

A. Context

The DALE (Office cantonal du Logement et de la Planification Foncière, State of Geneva), in collaboration with the University of Applied Sciences in Geneva (HEG-GE) are co-producing a public register with set of smart contracts facilitating the open and transparent execution of urban planning processes. This existing initiative attempts to: a) test and authenticate the execution of a process between several entities of the domain, through the application of smart contracts and b) make proposals around a multi-stakeholder governance of smart contracts for public services.

A recent study by Credit Suisse [14] describes the challenging situation around an "affordable" housing-to-buy in Geneva, highlighting, the urbanism consequence: transportation, pollution, environment, moving of population to other countries - areas. Thus, new solutions, services and policies around building rights and urban planning are becoming urgent. The existing informational infrastructure includes a detailed analysis of the business process around building permissions. The overall view of how the building permission is processed today, includes: a) public administration actors - departments' participation and roles, b) decision process and status of a building permit and c) rules of when and how are building rights are calculated.

There are few selected information and consultation initiatives, driven by the public administration, with professional and local populations before the opening of new building zone. This process is set in order to generate building rights in specific areas with some kind of citizen participation. Evaluating this situation, we concluded that the process is largely opaque to the outside and confined within the public administration actors. At the same time, opening a new building zone can last up to four years, depending on various circumstances. Moreover, the implementation process of the accepted projects, in a specific building area, is not available to the public. These initial elements justify the main goal of this pilot: managing the building rights process in a more open, educated and collective way.

B. An API driven architecture

The architecture of our application tries to serve the need for more openness, transparency and collective management by the following core socio-technical elements: a) decentralized, permissive and editable storage of all data collected within our application, b) easy and transparent smart contract deployment and scrutiny for the related administrative processes and c) possibility for users to create proposals and vote for the proposals of others.

We are using the InterPlanetary File System's (IPFS) API to interact with it as our storage system. IPFS is a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system [15]. Similar to a torrent, IPFS allows users to not only receive but host and edit content.

As opposed to a centrally located server IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. The main method is the following: a hash is obtained based on the image file's binary codes. The file is retrieved by searching for it with its hash. It is not possible to replace an image, with another one, because the file is changing when its hash changes. The hash code is immutable on the Ethereum Blockchain and the file is immutable on IPFS.

For our application to interact with IPFS, we are using the method, presented in Figure 1. An IPFS node dials to other application instances using WebRTC:

 ISBN: 978-1-61208-757-3

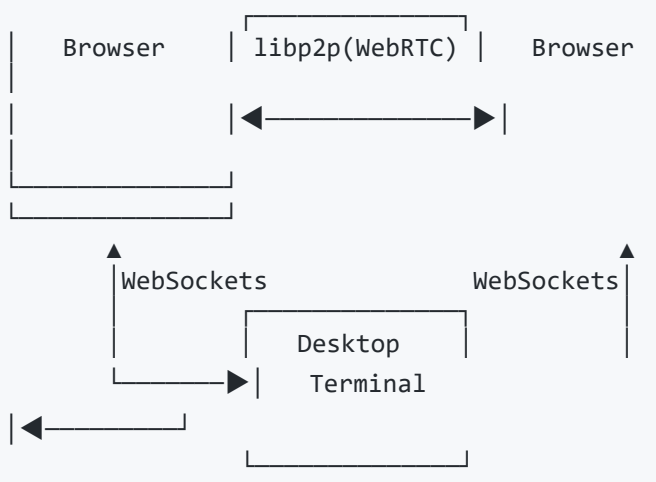

Figure 1. Application diagram with an IPFS node that dials to other instances using WebRTC

We are using the Nuxt.js framework, ideal for creating Vue.js applications and abstracting away the client/server distribution. The two important functions deployed are the ADD function (which takes a message ready to be posted and returns the information of the created hash) and the READ function read (which takes a hash and returns the message created).

The next step is to add the discussion with Ethereum, here the logic of the mechanism. We first check if we have local data to be taken into consideration through the user's browser: if no, we collect the data by reading the smart contract, pointing us to the file that needs to be recovered. If yes, we check with the smart contract that the referenced data are still valid, we update the smart contract and notify all the users that this change took place, followed by the online documentation. This process is facilitated by an appropriate middleware known as MetaMask. MetaMask is an application acting as a bridge that allows to run Ethereum apps from the user's browser without running a full Ethereum node.

The result of these architectural choices is an early prototype, published in June 2019, validated as a minimum viable product from the actors of the pilot. It follows the architecture described above and gives access to read and edit an initial building rights' table (DAB) describing the number of building rights allocated to a specific construction area, as well as, the exact parts of land associated to them. In Figure 2, we present a view of the main page of the DAB prototype:

Figure 2. The DAB prototype accessible at https://proto3.ynternet.org/

This initial table describing building rights in a specific construction area is at the core of the case study. It is the main register allowing for building applicants (site developers, architects, private entities) to interact with the public administration and claim their rights to execute a building project in a selected area. Below follows a more detailed view of this, now, decentralised and blockchain validated table. Figure 3 demonstrates how the editing of the initial building rights table in the DAB is presented:

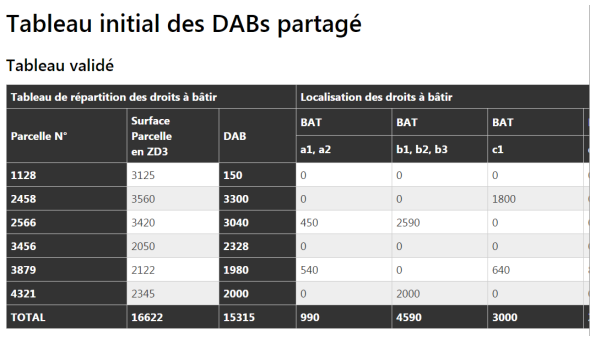

Tableau de répartition des droits à bâtir			Localisation des droits à bâtir		
Parcelle N°	Surface Parcelle en ZD3	DAB	BAT a1, a2	BAT b1, b2, b3	BAT c1
1128	3125	150	0	0	0
2458	3560	3300	0	0	1800
2566	3420	3040	450	2590	0
3456	2050	2328	0	0	0
3879	2122	1980	540	0	640
4321	2345	2000	0	2000	0
TOTAL	16622	15315	990	4590	3000

Figure 3. Editing the initial building rights table in the DAB prototype accessible at https://proto3.ynternet.org/

Regarding proposal creation of the application, our architecture implements the following general idea: when a user wants to send a new data through a smart contract, or even create a new one, we add his/her proposal on a stack of proposals with a dedicated function (addProposal). When we accept a proposal, we empty the stack and replace the data with the one in the proposal (acceptProposal that calls updateHash). This point brings us to the semantic data structure of the application. As already mentioned, IPFS allow us to post whatever type of data we wish, raw data or encrypted data. The application is now saving data in the JavaScript Object Notation (JSON), with a history of all edits made upon this data. The final version of the application will generate several JSON data models based on different processes, for example the table of distribution of user's building- rights in the application: one can create a specific field data, mapped as an object of which each key is another object, as demonstrated in Figure 4:

```
{
 "data": {
  "parcelle_id_1": {
   "bat_1": "dab_1",
   "bat_2": "dab_2",
  }
  "parcelle_id_2": {
   "bat_2": "dab_4",
  }
 }
 "created_at": "xxxx",
 "created_by": "xxxx",
 "previousVersion": "xxxx"
}
```

Figure 4. Data structure for the distribution of user's building rights

 ISBN: 978-1-61208-757-3

This modular and decentralized architecture, described in summary above, is thought out itself as an API. It acts as an interface for semantic data to be stored and then executed, as part of, or new smart contracts. These data are organised and can be accessed through a central smart contract, an oracle, which will serve as a registry table for other smart contracts. The oracle keeps a key - value of an identifier to the address of a smart contract tracing back to all data related to it. This process is called oraclize and provides a way to get outside data from any API onto the blockchain. This point allows us to proceed to the next area and examine the actual coding structure of smart contracts as APIs.

C. Coding Smart Contracts as APIs

Coding Smart Contracts as APIs allows us to design and deploy them, with the four characteristics described in the following paragraphs. The first one consists of making public a register with its data and add a read and write function available to its users. Figure 5 details how this code operates.

```
// This is the object structure representing a record
    struct record {
      address created_by;
      address updated_by;
      address smartContractAddress;
      bool exists;
    }
    // the mapping representing the register
    mapping(string => record) internal register;
    // Public function that write a record into the register
    function write(string memory _identifier, address
_smartContractAddress) public returns (bool) {
        address creator = msg.sender;
        if (register[_identifier].exists) {
            creator = register[_identifier].created_by;
        }
        register[_identifier] = record({smartContractAddress:
_smartContractAddress, exists: true, updated_by: msg.sender, created_by:
creator});
        return true;
    }
    // Public function that read a record value from the register
    function read(string memory _identifier) public view returns
(address) {
        require(register[_identifier].exists, "This record is empty");

        return register[_identifier].smartContractAddress;
    }
```

Figure 5. Creation of a blockchain register with a read and write functions

The second characteristic is linked to the modularity smart contracts as APIs, particularly for assigning of various identifiers, priorities and rules. This includes adding the canUpdateExistingRecord parameter, set with the smart contract deployment. This parameter is stating if an existing record can be updated. In Figure 6, we present the initiation a smart contract as an API.

```
// Can you update an existing record ?
    bool internal canUpdateExistingRecord;
  // At Smart Contract deployment you must say if an existing record can
be updated
    constructor (bool _canUpdateExistingRecord) public {
      canUpdateExistingRecord = _canUpdateExistingRecord;
    }
    ...
    function write(string memory _identifier, address
_smartContractAddress) public returns (bool) {
      // Before writing into the register, check whether you are about to
update an existing record and if you have the right to do so => otherwise we
send an exception stating "Existing records can't be updated"
      require(!register[_identifier].exists || canUpdateExistingRecord,
"Existing records can't be updated");
      ...
    }
```

Figure 6. Initiating a smart contract as an API

The third characteristic is about initiating a continuous interoperability between, user-driven, applications. This is initiated by adding an event, emitted every time someone writes on the register. The data of this event are describing its full internal process and are open and reusable to all blockchain users. In Figure 7, we demonstrate the code creating events allowing for more interoperability.

```
// event that can be transmitted and followed on the blockchain
    event writeRegister(
      string _identifier,
      address _smartContractAddress
    );
    ...
    function write(string memory _identifier, address
_smartContractAddress) public returns (bool) {
      ...
      // when we write into the register, we emit the event
      emit writeRegister(_identifier, _smartContractAddress);
      ...
    }
```

Figure 7. Creating events allowing for more interoperability

The final characteristic regards the possibility for a collaborative edition of the smart contracts based on transparent rules and constant user driven improvement. This take place by adding a function that changes the value of the canUpdateExistingRecord parameter. The code presented in Figure 8 introduces a propositions' mechanism for all users of the application and will be, at a later stage of this pilot, associated to a voting function.

```
  // an event to be sent when someone change the
canUpdateExistingRecord value
    event updateCanUpdateExistingRecord(
      bool _oldValue,
      bool _newValue,
    );
    // You can modify the canUpdateExistingRecord value
    function setCanUpdateExistingRecord(bool
_canUpdateExistingRecord) public {
      emit updateCanUpdateExistingRecord(canUpdateExistingRecord,
_canUpdateExistingRecord);
      canUpdateExistingRecord = _canUpdateExistingRecord;
    }
```

Figure 8. Towards user driven collaboration and improvement

As with all public smart contracts, the code presented above can be traced online with all of its actual transactions [16]. The demonstrated characteristics showed how a smart contract designed as an API, can be implemented in a modular and open way.

 ISBN: 978-1-61208-757-3

V. CONCLUSIONS AND FUTURE WORK

Application development, in a blockchain and smart contracts context, seems like a novel opportunity to create platforms with less opacity and great collaboration for their participants. However, blockchain(s) remain protocol(s) and as we already learned in the last twenty, or so, years, decentralized protocols are neither participative, nor more democratic by default: control and centralization can take place in various levels and spheres [17]. Through this article, we tried to use APIs as an overall concept both of the architecture and the smart contracts of our pilot. Particularly, the function the oracle is cutting through existing APIs and smart contracts as applications.

Developing an API culture is a prerequisite to use any information infrastructure, particularly the ones allowing for the deployment of smart contract enabled applications. Smart contracts' immutability and forced temporality are crucial. At the time of execution of an application, there are elements that demand explicit attention and negotiation with involved stakeholders. Moreover, they need to be designed and thought our as APIs, exposing for the very beginning their objectives, rules of operation and governance.

The early experience from our building rights' management pilot is teaching us that, smart contracts adoption and administration, need to be tightly linked to specific skills within the responsible organizations. The main driver for public sector blockchain pilot initiatives, like our Geneva based pilot described in this article, is mostly based in the principles of transparency and efficiency, particularly in business process and data monitoring. Blockchain and smart contracts are unique elements for information infrastructures serving such principles.

ACKNOWLEDGMENT

This paper is possible thanks to the SCODES research project: an applied research project on blockchain protocols and smart contracts, involving five Hautes Ecoles from French-speaking Switzerland (University of Applied Sciences HES-SO). Its goal is to develop knowledge within this particular field studying and developing practical cases of use and transferring the acquired knowledge to the regional economic actors.

REFERENCES

[1] A. Priftis and J.-P. Trabichet, "The CoWaBoo protocol and applications: towards the learnable social semantic web" International Journal on Advances in Software, vol. 11, pp. 6-17, 2018.

[2] T. Espinha, A. Zaidman and H.-G. Gross, "Web API growing pains: Loosely coupled yet strongly tied," Journal of Systems and Software, vol. 100, pp. 27–43, 2015.

[3] Wikipedia. *Application programming interface*, [Online]. Available from: https://en.wikipedia.org/wiki/Application_programming_interface, [retrieved: September 2019].

[4] T. Bucher, "Objects of intense feeling: The case of the Twitter API," Computational Culture, number 3, 2013. Available from: http://computationalculture.net/objects-of-intense-feeling-the-case-of-the-twitter-api/ [retrieved: September 2019].

[5] D. Rüede and K. Lurtz, K., "Mapping the various meanings of social innovation: Towards a differentiated understanding of an emerging concept" EBS Business School Research Paper Series 12–03, Oestrich-Winkel, 2012.

[6] P. Constantinides, "The development and consequences of new information infrastructures: the case of mashup platforms", Media, Culture & Society, vol. 34(5), pp. 606-622, 2012.

[7] N. Szabo, "Formalizing and Securing Relationships on Public Networks". First Monday, vol. 2, no. 9, 1997, doi: http://dx.doi.org/10.5210/fm.v2i9.548 [retrieved: September 2019].

[8] R. O'Shields, "Smart Contracts: Legal Agreements For The Blockchain," 21 N.C. Banking Inst. pp. 180-181, 2017.

[9] V. Buterin, V. "A next-generation smart contract and decentralized application platform." Ethereum 1–36 (2014) [Online]. Available from: http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf [retrieved: September 2019].

[10] M. Wöhrer and U. Zdun, "Smart contracts: Security patternsin the ethereum ecosystem and solidity," in1st InternationalWorkshop on Blockchain Oriented Software Engineering@ SANER 2018, 2018. [Online]. Available from: https://eprints.cs.univie.ac.at/5433/7/sanerws18iwbosemain-id1-p-380f58e-35576-preprint.pdf [retrieved: September 2019].

[11] Q. Dupont and B. Maurer, B "Ledgers and Law in the Blockchain." Kings Review [Online]. Available from: http://kingsreview.co.uk/magazine/blog/2015/06/23/ledgers-andlaw-in-the-blockchain/ [retrieved: September 2019].

[12] A. Wright and P. De Filippi, "Decentralized Blockchain Technology and the Rise of Lex Cryptographia." Social Science Research Network 2580664 (2015). Available from: http://papers.ssrn.com/abstract=2580664 [retrieved: September 2019].

[13] B. Klievink and M. Janssen, M., "Developing multi-layer information infrastructures: Advancing social innovation through public–private governance". Information Systems Management, 31(3), pp. 240-249, 2014.

[14] Investment Solutions & Products Swiss Economics, "Location, location, floor plan for the Swiss Real Estate Market in 2019" [Online]. Available from: https://www.credit-suisse.com/media/assets/private-banking/docs/ch/privatkunden/eigenheim-finanzieren/swiss-real-estate-market-2019.pdf [retrieved: September 2019].

[15] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)" [Online]. Available from: https://ipfs.io/ipfs/QmV9tSDx9UiPeWExXEeH6aoDvmihvx6jD5eLb4jbTaKGps [retrieved: September 2019].

[16] Smart Contracts as API: the Oracle. Full code accessible at https://ropsten.etherscan.io/address/0xf083a44e157b9ac4b7dea543bd67547ecf57d00a#code [retrieved: September 2019].

[17] A. Galloway, Protocol: How Control Exists After Decentralization, MIT Press, 2004