Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Investigating the suitability of blockchain for managing patients consent in clinical trials

Boris Flesch

August 31, 2021

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MSc in Computer Science (Intelligent Systems)

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Signed: _____ Date: _____

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Signed: _____ Date: _____

Abstract

Clinical trials aim at finding new treatments to improve patients lives. These trials rely on patients clinical data which is particularly sensitive and raises privacy concerns, especially in the context of rare disease such as Motor Neuron Disease (MND) which motivates this research project. An approach to empower patients gaining control over their data consists in requiring their consent with particular clauses regarding data (e.g. what data is shared, with who, for what purpose). This dissertation investigates the usage of blockchain to manage patients consent in clinical trials.

An iterative and multidisciplinary literature review is conducted to determine methods, models and regulations in place for consent management in clinical trials with a focus on the European Union (EU). State-of-the-art usage of blockchain in clinical trials is also reviewed and results in the identification of a knowledge gap that exists. While intrinsic properties of blockchain makes it interesting to address patients consent management (e.g. decentralised, privacy- and integrity-preserving, tamper-proof), this approach currently lacks of evaluation regarding its compliance with regulations such as the GDPR. Furthermore, it is unclear from previous work what exactly should be the scope of the blockchain for managing patients consent in clinical trials. The design-science framework and the underlying notion of *wicked problems* are followed to design an artifact which addresses the sole problem of patients consent management while maximising its interoperability with other artifacts.

In this dissertation, a prototype of a Blockchain as a Service (BaaS) for consent management is designed and implemented, and its compliance with regulation is evaluated with the example of the GDPR. It results in a functional BaaS which demonstrates feasibility as well as fulfilment of the requirements (i.e. implementation of consent clauses and compliance with the GDPR). The prototype provides with encouraging results, although it raises limitations that are discussed. This dissertation paves the way for future work and investigation to solve related (sub)problems by designing further artifacts.

Acknowledgements

First of all, I would like to thank Prof. Gaye Stephens who supervised this dissertation project. She has been involved in every step of the project, from the identification of the problem to the several proofreading iterations, and provided me with an invaluable support and guidance which helped me to conduct this dissertation project.

I would also like to thank all members of the MND Team of the ADAPT Centre at Trinity College Dublin. Their research about Motor Neuron Disease has been crucial to motivate this project and to set its context.

I am also thankful to all the professors and staff of Trinity College Dublin who contributed to make the learning experience fascinating and enriching throughout this Master's degree.

Finally, my gratitude goes to all my family and friends for their perpetual support and motivation. Particularly, I would like to acknowledge my father, Dr. Gérard Flesch, for his in-depth proofreading and his valuable insights.

Contents

List of Figures

List of Tables

Abbreviations

ABI	Application Binary Interface
API	Application Programming Interface
BaaS	Blockchain as a Service
BCT	Blockchain Technologies
CT	Clinical Trial
CTMS	Clinical Trial Management Software
dApps	Decentralised Applications
DeFi	Decentralised Finance
EU	European Union
EVM	Ethereum Virtual Machine
GDPR	General Data Protection Regulation
HIQA	Irish Health Information and Quality Authority
ICT	Information and Communications Technology
JS	JavaScript
JSON	JavaScript Object Notation
MND	Motor Neuron Disease
OOP	Object-Oriented Programming
PoA	Proof-of-Authority
PoW	Proof-of-Work
RPC	Remote Procedure Call
TTP	Trusted Third Party
Tx	Transaction
TXH (TxHash)	Transaction Hash
URI	Uniform Resource Identifier

1 Introduction

Information Communication Technologies (ICT) have an increasingly prominent role in healthcare (9). Various technologies that aim to improve patients' lives are emerging (e.g. AI-assisted diagnostics, personalised healthcare, drugs side-effects prediction, large-scale clinical studies). Most of these new ICT-based techniques require large amounts of patients' data to build relevant and reliable solutions (10).

An important use of patients' data is for clinical trials which aim to find new treatments to cure diseases. The particular context of rare diseases introduces scarcity in patients' data, from which derives further privacy concerns. For instance, the scarcity of data makes re-identification of patients with rare diseases easier than others. An approach for enhancing patients' data privacy is to empower them to gain control over their data. This is usually done through the collection of patients consent. The aim of this project is to investigate the usage of blockchain to manage patients consent in clinical trials.

As shown in the literature, there are many perspectives to consent management in general and in healthcare in particular. Using an ICT enabled process to manage consent adds another perspective. The application of ICT to healthcare is not a trivial process. Healthcare is a domain involving complex processes and sub-processes, stakeholders from many disciplines and a reliance on team based activity. This context is important to consider when choosing a method to undertake the investigation and to situate the work with what has gone before and any future work.

1.1 Motivation

The context of rare disease and, more specifically, Motor Neurone Disease (MND) which motivates this research project provides an interesting background to investigate the notion of patients consent and compliance with regulations. For instance, the scarcity of patients can lead to refined clauses and level of granularity of their consent (7). Conversely to more prevalent diseases, the increased patients willingness to participate to clinical trials combined with their scarcity requires an extensive control of the sharing of their data.

MND is an uncommon condition for which no cure currently exists. The condition affects the brain and nerves and causes deterioration of patients health sometimes over many years. As it is a rare condition, few local clinical data (e.g. demographic information, diagnosis, treatments, patients monitoring) is available to properly conduct clinical studies and trials related to MND. For that reason, there is a real need to share patients data in rare disease. Gathering data from patients within all countries in the European Union (EU) could be a great opportunity to further research which focuses on curing MND and improving lives of patients diagnosed with that condition.

Exchanging health-related information in a highly reliable and secure way facilitates the establishment of clinical trials and research based on patients' data shared across many regions of the world. However, threats to patients data privacy still exist and there is room for improvement in that domain. For instance, patients data exchange is highly beneficial to rare diseases like MND where there is limited data available in some regions.

Many regulations including the General Data Protection Regulation (GDPR) aim to protect patients' data and, more generally, any usage of personal information shared on the internet. As the GDPR classifies Health data as sensitive data, particular attention should be given to prevent unauthorised access and sharing of this type of data.

1.2 Research question

The research question of this dissertation is the following: "*To what extent can blockchain be used to manage patients consent for clinical trials?*"

1.3 Objectives

This dissertation investigates the suitability of blockchain – a decentralised ledger further defined in section 2.1.5 – for managing consent in clinical trials. It focuses on the evaluation of compliance of blockchain usage with regulations such as the GDPR and clauses inherent to patients' consent in clinical trials. This investigation is performed through an evaluation of feasibility and compliance with regulations. The following research objectives have been defined:

(a) Analysis of the state-of-the-art methods and models for consent management in clinical trials

(b) Analysis of regulations in place for data privacy in the context of consent management for clinical trials within the EU

(c) Creation of a Blockchain as a Service (BaaS) prototype for consent management in

clinical trials

(d) Evaluation of the compliance with regulations of using blockchain for consent management

(e) Evaluation of the feasibility of implementing a blockchain for consent management

1.4 Methodology

This project is based on the design-science research framework which relies on the design of artifacts to solve wicked problems, i.e. complex problems that have unstable requirements and constraints (11). It focuses on iteratively determining what the problem really is in order to design a solution while maximising interoperability with other solutions to related problems. In that sense, the investigation conducted within this research project does not follow a *thesis, antithesis, synthesis* structure. Instead, it should be seen as an exploration process which draws upon an iterative and empirical approach to identify and solve the wicked problem of consent management in clinical trials. An artifact or knowledge moment will result from this investigation. In the future, when enough related knowledge moments come together, a comparison based on these explorations could be done.

1.5 Organisation of this dissertation

This dissertation is organised in six chapters as follows:

Chapter 1 introduces the dissertation project.

Chapter 2 provides relevant background information including literature review of related work as well as discussions regarding the design-science methodology followed within this dissertation.

Chapter 3 focuses on research design and methods by explaining how design-science guidelines are concretely applied to this research project and describing the evaluation methods considered.

Chapter 4 details the design and the implementation of the Blockchain as a Service (BaaS), which is the main artifact resulting from this dissertation.

Chapter 5 evaluates the resultant implementation of the BaaS and describes limitations that have been identified.

Finally, chapter 6 draws conclusions and suggests ideas for future work which relates to this project and could require further investigation.

2 Background

This chapter presents background information which is essential to identify and understand the knowledge gap that exists within the usage of blockchain technologies (BCT) for consent management in clinical trials. In the first section, the essential terms are defined to ease the understanding of the subsequent sections. The second section discusses design-science approach. Then, the third section establishes a literature review of related work. Thereafter, the knowledge gap that exists is stated and, finally, conclusions are drawn from background research.

2.1 Essential terms

In the following subsections, the concepts of clinical trials, privacy, security, consent and blockchain are described.

2.1.1 Clinical trials

According to the World Health Organization (WHO), conducting clinical trials is a way to test and evaluate a new drug, treatment, procedure or other health-related product with a set of patients volunteer to take part of the trial (12). These trials are meant to evaluate both the effectiveness of the treatment, potential side effects and overall effects on patients health. After being rigorously designed and approved by competent authorities, clinical trials are divided into four phases of studies as follows:

- Phase I studies test the new product on a small group of people to identify main side effects and determine a safe dosage.
- Phase II studies are conducted on a larger group of people after the new product has been proven to be safe in Phase I.
- Phase III studies are conducted on a group again larger than Phase II and considering patients from larger populations (e.g. in different regions of the world).
- Phase IV studies occur after a product has been approved nation-wide and needs

testing over a longer timeframe.

Friedman et al. define clinical trials as "a prospective study comparing the effects and value of intervention(s) against a control in human beings" (13). Furthermore, they state that clinical trials stand amongst the best experimental techniques to assess the relevance and usefulness of an intervention (e.g. a new drug or treatment). Due to the small number of subject, rare diseases can imply different studies design. However, trial designs "must meet the same rigorous standards as those for trials for more prevalent diseases" (14).

2.1.2 Privacy

The notion of privacy appears not to have a universally accepted definition. Its definition rather differs depending on individuals as well (15). For instance, in information systems, privacy is defined as the degree of control an individual has over their data, i.e. what data is used, for what purpose and who has access to it. Similarly, in law, the GDPR defines data privacy as a way to "empower users to make their own decisions about who can process their data and for what purpose" (16). From a medical perspective, privacy is often similarly defined and merged to security and confidentiality concerns regarding patients data (17).

In this dissertation, the definition of privacy will focus on the one given by the GDPR as a way to allow users (i.e. patients) to decide whether or not to share their health data, at what level of granularity, to who they are willing to share it and, finally, for what purpose (i.e. in which data processing or aims). To concretely apply this definition of privacy, the GDPR relies on the collection of users or patients informed and explicit consent.

2.1.3 Security

Security and privacy are two complementary notions which are often discussed dependently from one another (10, 18). This dissertation essentially focuses on privacy, hence the following definition helps understanding what is *not* in the scope of this project, although some privacy issues are addressed by using security techniques.

In information systems, security consists in protecting any data from intrusion, breach, unauthorised action or unauthorised access. The RFC3552 states that security does not consist in a single property or block but rather "a series of related but somewhat independent properties" (19) which are divided into two main categories (i.e. goals): communication security and systems security.

2.1.4 Consent

The Cambridge Dictionary defines consent as an agreement or a permission given by an individual to do something[1].

In a medical context, only volunteer patients can participate to medical trials and their consent is systematically required prior to an intervention such as a surgical operation or when a new treatment is tested. That is an essential notion in all ethics guidelines and principles for research on human subjects, such as the Declaration of Helsinki (2013) (20), the Belmont Report (1979) (21) and the Nuremberg Code (1947) : "The voluntary consent of the human subject is absolutely essential." (22). It is therefore necessary to collect patients consent on (i) their willingness to participate to the clinical trials, (ii) what data they are willing to share and (iii) for what purpose. Furthermore, it is required to gather patients *informed* and *explicit* consent.

Informed consent

In healthcare, a consent is considered *informed* when a patient has enough autonomy to evaluate the implications of what they are consenting to. In other terms, the patient is informed of what will be done to them and what will not prior to giving their consent (23). It is part of medical ethics standards to ensure that an individual with decision making capability can give or not their consent.

Explicit consent

An *explicit* consent requires the patient to perform a particular action to grant their consent. In other terms, an explicit consent follows an 'opt-in' model and cannot be defaulted as granted.

An interesting analogy can be made with cookies consent banners which appeared on websites over the past few years in accordance with the GDPR (24). Informed consent can be obtained by presenting to the visitor a comprehensive list of cookies which are used by the website and explaining what they are used for. The notion of explicit consent is characterised by preventing any cookies to be activated until the user has clicked, for instance, on an "Accept cookies" button (25). In other terms, the fact that a visitor continues their navigation on a website cannot be considered as an explicit consent.

[1] Definition of 'consent' by the Cambridge Dictionary, https://dictionary.cambridge.org/dictionary/english/consent

2.1.5 Blockchain

A chain of blocks

A blockchain is a secure, transparent and distributed ledger. Unlike traditional databases, it is an append-only registry which stores the complete history of transactions in an tamper evident and tamper resistant way, hence blockchains are often referred to as immutable ledgers. This is made possible by cryptographic functions used to create hashes of blocks to link them together (26).

In a blockchain, a block is composed of two main sections: header and data. The header usually contains metadata such as the block number, the hash of the data and the hash of the previous block's header. Data contains transactions (Tx) that have been processed through the blockchain. A block can either contain no transaction, one or several transactions.

Public, private or permissioned

A blockchain can either be public, private or permissionned. For instance, Bitcoin and Ethereum are two public (or 'permissionless') blockchains particularly known for their application in decentralised banking and finance (DeFi) (27). On a public blockchain, anyone can join the blockchain and act as a node of the decentralised network. A private blockchain, on the other hand, allows only authenticated participants to join the network. A permissioned blockchain takes the advantage of both public and private blockchains by providing specific rights to some users (i.e. nodes) (28).

The principle of consensus

Another key concept of blockchain is the principle of consensus. As a blockchain is distributed across many nodes, when a new block is added to the chain, the majority of nodes needs to reach an agreement to effectively add the block to the chain. Otherwise, the block is rejected. This agreement on the state of the blockchain is based on a consensus algorithm (29).

The most widely used consensus algorithm is the Proof-of-Work (PoW), which is also referred to as 'mining'. For instance, this algorithm is used by Bitcoin. It relies on computational power to solve algorithmic puzzles. More precisely, it consists in finding a value which hash starts with a required number of zero bits (30). It is a non-trivial problem and requires high computational resources as no formula exists to determine such value.

Other algorithms can be used, such as the Proof-of-Authority (PoA) which relies on nodes identity in the network. With this algorithm, some nodes are granted higher authority in the

network and can act as 'validators' to confirm new transactions and blocks within the blockchain (31).

Smart contracts

In addition to connecting nodes in a decentralised network, it is possible to deploy Smart Contracts on some blockchains. Sometimes referred to as chaincode, these contracts are programs that are executed on the blockchain and which can be used as trusted and decentralised third-parties or arbiters when processing is made over the blockchain (32). For example, a smart contract could be setup to handle voting in an election. These contracts are usually written in Solidity, an Object-Oriented Programming (OOP) language which has some similarities with JavaScript and C++.

2.2 Design-science

This section introduces the design-science framework. First, design-science guidelines and concepts are presented. Then, the idea of *wicked problems* is discussed. In the third subsection, the genres of inquiry in design-science are introduced. Finally, a focus is made on how design-science can be applied to the domain of clinical trials.

2.2.1 Design-science guidelines and concepts

Design-science is a research paradigm which consists of designing artifacts to help resolve complex problems. Hevner et al. describe this paradigm as a framework including clear guidelines to support design of information systems research projects (33). This framework defines design-science as a "build-and-evaluate loop" consisting in building and evaluating an innovative approach to solve a problem, improving the understanding of the problem and repeating the process; until eventually reaching a final design artifact.

The design-science research framework establishes the following seven guidelines which make possible to break a large problem into smaller problems and to build rigor in the research approach:

1. Design as an Artifact
2. Problem Relevance
3. Design Evaluation
4. Research Contributions
5. Research Rigor
6. Design as a Search Process

7. Communication of Research

The framework also proposes a wide range of design evaluation methods: observational, analytical, experimental, testing and descriptive.

2.2.2 Solving wicked problems

Design-science is particularly suitable for solving *wicked problems*, defined by Introne et al. as "problems for which no single computational formulation of the problem is sufficient" (11). In the design-science research framework for information systems, these problems are characterised by "unstable requirements and constraints based upon ill-defined environmental contexts" and implying "complex interactions among subcomponents of the problem and its solution" (33).

The notion of wicked problems has originally emerged from the domain of social science (34). It is now applied to a wide range of problems, including healthcare which domain is considered as a wicked problem in its entirety (35).

2.2.3 Genres of inquiry

In addition to the research framework itself, Baskerville et al. define a segmentation of design-science research in four distinct genres of inquiry, based on a interchangeable combination of Nomothetic/Idiographic and Design/Science (5). Practically, 'nomothetic' defines general classes of problems while 'idiographic' refers to more specific ones. The notion of 'design' alludes to solutions such as theories or frameworks, contrarily to 'science' which relates to concrete, systematic evaluation and validation of a solution, i.e. a design.

The four genres of inquiry can be summarised as follows:

1. *Nomothetic Design*, aims in providing design, frameworks or theories as solutions to a wide class of problems (i.e. generalised solutions).
2. *Nomothetic Science* can be seen as an application of an artifact issued from Nomothetic Design to one or more specific population(s) in order to proceed to a systematic evaluation or validation.
3. *Idiographic Design* provides knowledge (e.g. design, frameworks, theories) within the scope of a specific problem, conversely to Nomothetic Design which is more general. It seeks for turning "an existing situation into a preferred one" (5).
4. *Idiographic Science* refers to concrete research and validation of a solution when applied to a specific problem or artifact.

These genres of inquiry help in identifying where knowledge gap exists and from which perspective a wicked problem could be addressed to complement artifacts that already exist in related work.

2.2.4 Applying design-science to clinical trials

The characteristics of design-science framework make it particularly suitable for research in information systems for healthcare and, more specifically, clinical trials. Indeed, the health industry can be segmented in a large amount of wicked problems which require to be addressed separately and in a flexible enough way to enable interoperability of the proposed solutions; that is, designing as artifacts to solve subproblems.

To identify opportunities of application of blockchain technologies to healthcare, Yaqoob et al. segmented healthcare into the following seven industries: Improved Drug Traceability, Patient Record Management, Clinical Trials and Precision Medicine, Maintaining Consistent Permissions, Protecting Telehealth Systems, Optimising Health Insurance Coverage and Medical Billing Systems (1). As shown on Figure 2.1, each of these segments has its own existing problems that blockchain technologies could help to solve according to the authors. From the perspective of design-science, each industry and each subproblem can be perceived as a wicked problem for which ascertaining all background and contextual information does not seem realistic. Each of these subproblems could also result in the development of an independent but interoperable artifact.

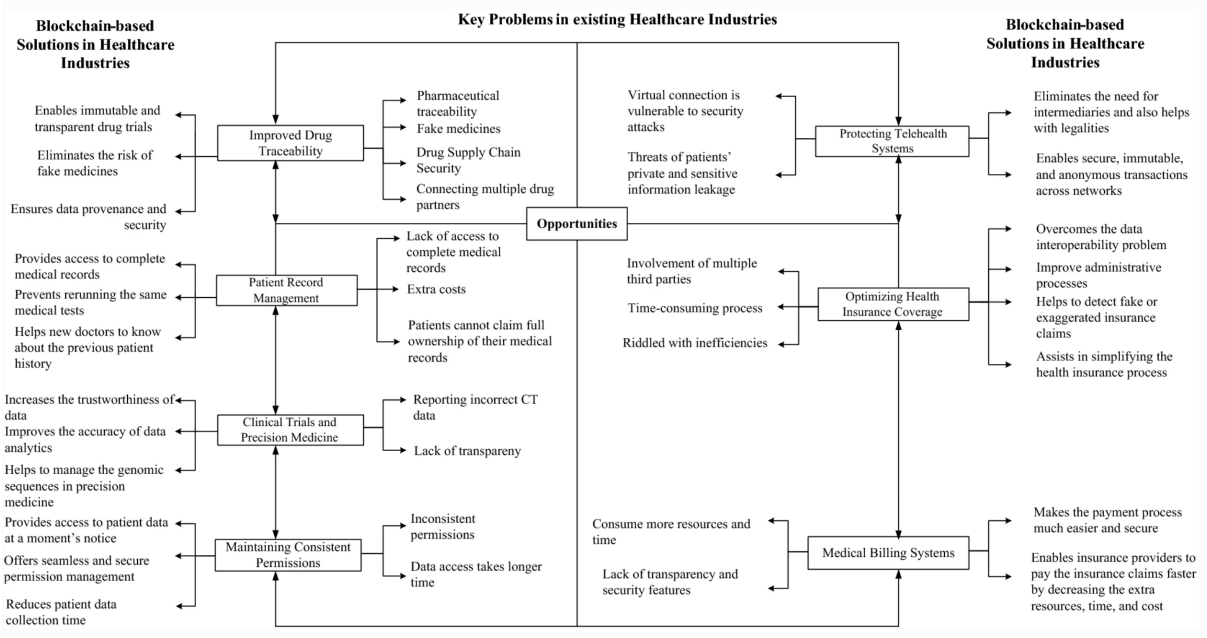

Figure 2.1: Blockchain opportunities in healthcare industries by Yaqoob et al. (1)

2.3 Literature review

Due to the wicked nature of the problem addressed, literature review is essential not only in understanding the problem but also in identifying what the problem is. Therefore, background research has been conducted following an iterative and empirical approach which considers the multifaceted concepts and stakeholders involved in consent management for clinical trials. This research section introduces background knowledge and an overview of the state-of-the-art regarding patients data, clinical trials, legislation and blockchain technologies.

First a description of patient data will be presented followed by the enrolment of patients in clinical trials. Then, consent models that are in place within the European Union are introduced. In the fourth subsection, a focus will be made on consent forms which are used in the EU. Thereafter, as one of the research objectives concerns compliance with regulation, GDPR-compliant consent management will be discussed from the perspective of linked data. Finally, an exploration of the state-of-the-art usage of blockchain technologies in clinical trials will be conducted.

2.3.1 Patients data

This section describes how clinical data is currently handled specifically for clinical trials and more generally for support the provision of healthcare. First, the notion of Electronic Health Records and their usage in clinical research is introduced. Secondly, the ownership of patients data is discussed. Finally, as this dissertation project focuses on the example of rare diseases with MND, particular concerns for patients with rare disease are considered.

Electronic Health Records and clinical research

Patients data is a key component to properly conduct clinical trials. In healthcare, patients data is usually stored within Electronic Health Records (EHRs) – also known as Electronic Medical Records (EMRs). These records allow doctors to store patients data, for instance within a hospital information system. For instance, patient data can be gathered by doctors to help provide a more accurate diagnosis. For that reason, it is referred to as the *primary use* of patient data (i.e. using the data for its original purpose; for the reason it has been originally collected). This data could be reused, for instance to conduct clinical trials. In that context, it is referred to as the *secondary use* of data (36). However, it is worth noting that these two definitions are often confused in the context or rare diseases where only few patients and data is available and clinical research is seen as part of the care process.

Instead of considering EHRs and data gathering for clinical studies as two disjoint entities, Cowie et al. emphasise how using patients health records has the potential to leverage

clinical research (37). That approach could therefore be used as a basis for gathering patients data (i.e. both EHRs and clinical research data) for prospective clinical trials. In addition to gathering and storing patients data within EHRs, collecting such data inevitably requires discussion about ownership.

Ownership of patients data

Although the investigation of patients data ownership is out of the scope of this dissertation, having an idea of the current state of data ownership helps understanding the stakes and consequences of giving patients more control over their data. A frequent question is to determine who patients' data belongs to, for example, the patients themselves, the caretakers and/or the researchers. In addition to ownership, Charlotte J. Haug raises the question of who should have control over such data. While the answer remains unclear, she reports the importance of using current technologies to handle dynamic consent management as well as "to stay in touch with trial participants and seek their renewed consent" (38).

In 2014, Blumenthal and Squires proposed a comparative discussion on either letting caretakers or the patients themselves have full control of the patient data (39). Doctors insist on the fact that having access to all patients health information can help them provide better care. Conversely, patients have the right to control their health and are supposed mentally competent to accept or reject doctors recommendations. The risk of empowering patients to control their health information could be that relevant information is missing on the doctors end to establish a correct diagnosis. Therefore, patients should assume the consequences of their decisions if they choose – purposefully or not – to hide some data. Indeed, "the ways in which caretakers use information are often non-linear and unpredictable" (39) and an ongoing medical treatment could cause side-effects or have severe consequences if taken along another treatment. Finally, the same paper also describes the lack of comprehension, both from patients and doctors perspectives, on what exactly EHRs contain and who has access to them.

Particular concerns for patients with rare diseases

As shown by Schwartz et al., only a minority of patients choose to restrict access to their health data by their primary care providers (40). However, patients with rare disease could tend to share their data more easily while being less concerned about their privacy. Indeed, research tends to demonstrate that health data is a key component for finding new drugs and treatments in healthcare; as less data is available in a rare disease context, patients are generally more willing to share their data in order to help research finding a treatment to cure their disease (41).

An important threat to privacy in health data is the concept of deanonymisation: the

re-identification of a patient based on a subset of their data. For instance, "99.98% of Americans can be reidentified from a database with less than 15 demographic attributes" (41). In the context of rare disease, even less demographic attributes could lead to the re-identification of a patient as the entropy in patients health data decreases with the scarcity of the disease. In other terms, it is easier to re-identify someone within a small region or population. That is an important threat to patients privacy, especially in the case of rare diseases where patients willingness to participate to such trials is significant, often regardless of their own privacy.

In this section, how patients data is used in clinical research was discussed. Then, it has been identified that there is no evidence in determining who really owns patients data, which is an important concern from a privacy and consent perspective. Finally, another concern is the smaller importance given to privacy by patients with rare diseases. The following will focus on the usage of patients data in clinical trials, for instance in the process of patients enrolment.

As the enrolment of patients in clinical trials requires their consent, it is crucial to understand how consent is gathered and what model and regulations are relevant.

2.3.2 Patients enrolment in clinical trials

Beside the typical processes in clinical trials such as its four phases of studies described in section 2.1.1, an important step regarding patients consent is the enrolment process, also referred to as patients recruitment. To take part in clinical trials, patients have to fulfil a set of eligibility criteria which are specific to each clinical trial. The CONSORT 2010 Group proposed a statement to improve the reporting of randomised controlled trials which is depicted in Figure 2.2 (2). Although the statement has been established in 2010, it is still a reference in the process and remains adopted within recent clinical trials. The key steps of this flow diagram are the following:

1. Patients enrolment after their eligibility has been assessed
2. Patients randomisation
3. Allocation of the patients to intervention
4. Follow-up of the patients
5. Analysis of the results

These steps outline the necessity of collecting patients consent for their enrolment within the clinical trial. As patients can consent to use only a subset of their clinical data within the clinical trial and for a specific purpose, it is crucial to consider the notion of dynamic consent, i.e. consent that could evolve with the clinical trial if additional data appears to be

needed or if data has to be used for a different purpose than the one initially planed in the clinical trial description.

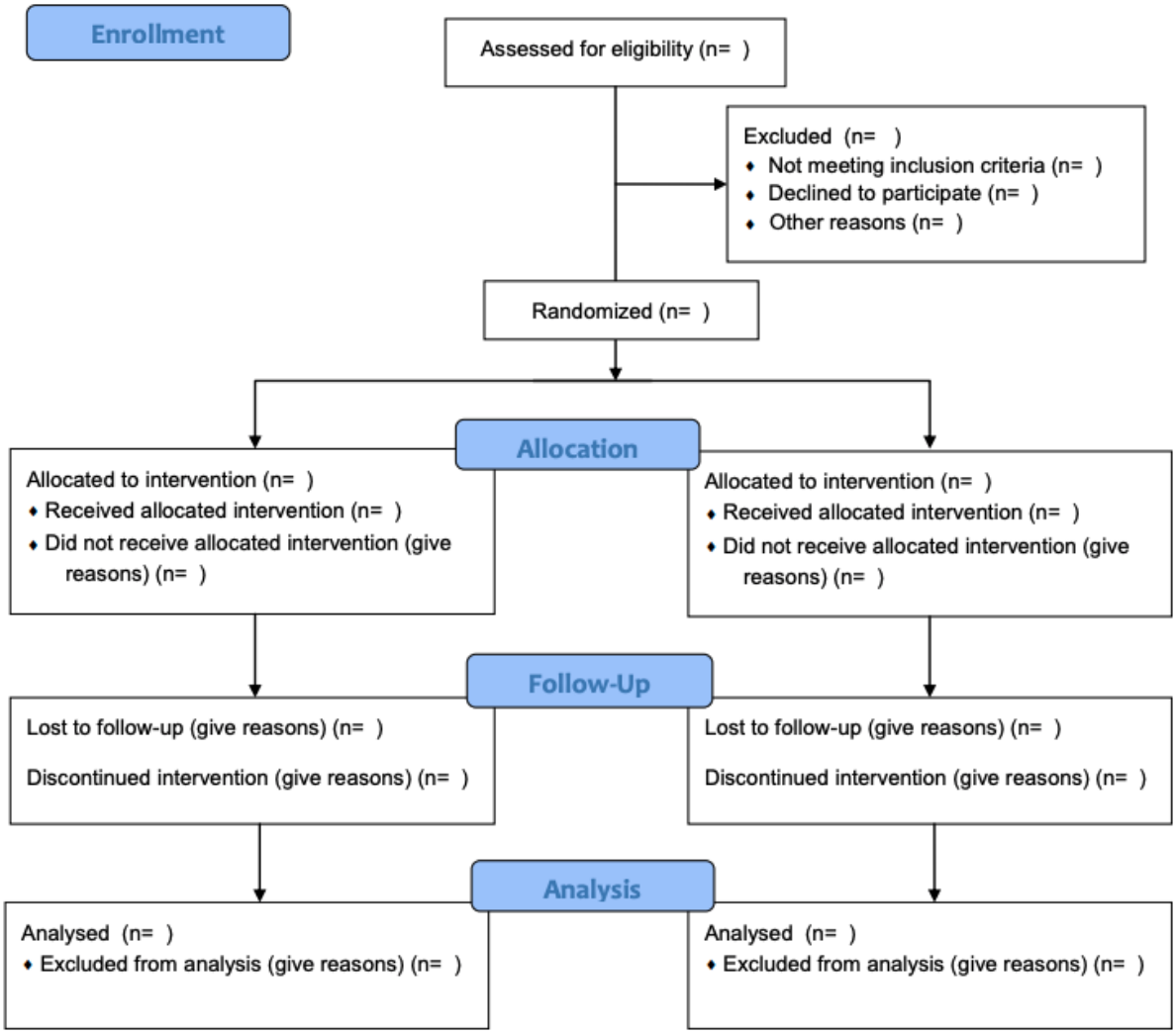

Figure 2.2: CONSORT 2010 Flow Diagram (2)

For instance, McCroary described the enrolment of patients in clinical trials for MND as a very slow process, essentially due to data access restrictions which makes it difficult to quickly interrogate the databases (42). Furthermore, the current enrolment process relies on paper-based data capture which is then stored in digital spreadsheets; while relying on the knowledge of one dedicated expert. In order to speed up the process, McCroary proposes to connect all MND patients within a single database in his investigation.

2.3.3 Consent models within the European Union

In February 2020, the Irish Health Information and Quality Authority (HIQA) published an International review of consent models for the collection, use and sharing of health information (6). The paper focuses on consent models, eHealth initiatives and applicable

regulations within each country. The following countries in the EU have been included in the paper: Ireland, Northern Ireland, Estonia, Finland and Denmark.

The HIQA particularly distinguishes two types of consent: explicit and implied or, respectively, opt-in and opt-out consent models. While the notion of explicit consent has already been previously defined; the HIQA insists on the active user engagement that is implied by explicit consent. Conversely, implied consent refer to consent given without requiring an action from the user, although it can be opted-out.

To summarise key findings of this paper, all EU countries implied have a national health identifier in place. Moreover, most of these countries have either EHR systems in place or under development. As of now, only Estonia, considered as a leader in eHealth systems and technologies, pioneered in using blockchain technologies within its nation-wide eHealth infrastructure.

From a consent models perspective, no country seems to require explicit patients consent for providing direct care. These countries rely on implied consent for individual care. In Finland and Estonia, the explicit consent of the patient is only required for sharing data with external healthcare. For the secondary use of patients data, both have authorities in charge of delivering permits for its usage by third-parties, after patients data has been anonymised. Regarding Northern Ireland, explicit consent of the patients is required "if identifiable data is used for research purposes" (6), while no consent is required for the usage of anonymised data.

Finally, the HIQA review also provides the following list of GDPR statements that allow processing health-related data:

- the data subject has given explicit consent to the processing of those personal data for one or more specified purposes
- processing is necessary for the purposes of carrying out the obligations and exercising specific rights of the controller
- processing is necessary to protect the vital interests of the data subject or of another natural person where the data subject is physically or legally incapable of giving consent
- processing is necessary for reasons of substantial public interest
- processing is necessary for the purposes of preventive or occupational medicine, for the assessment of the working capacity of the employee, medical diagnosis, the provision of health or social care or treatment or the management of health or social care systems and services on the basis of Union or Member State law or pursuant to contract with a health professional

- processing is necessary for reasons of public interest in the area of public health.

The current context of COVID-19 pandemic provides concrete examples of the application of the latest statement, authorising the processing of health data for reasons of public interest or public health. On the one hand, Cosgriff et al. pointed out the lack of publicly available COVID-19 data in May 2020 (43). On the other hand, as synthesised by Shuja et al. in September 2020 (44), the public health context of COVID-19 has been largely utilised to build various COVID-19 datasets including medical images, speech and textual data and methods.

Table 2.1 summarises the legislation (rows) in place for each EU country (column) studied by the HIQA. A checkmark symbol '✓' means that the legislation is applied in the country it refers to. A cross ('✗') means that the legislation is not applied. Laws which implement or are based on the GDPR are written in bold. For instance, the "Privacy and Electronic Communications Regulations 2003" is applied in Northern Ireland. As this law appears in bold in the table, it means that it implements the GDPR; hence "GDPR" is also marked as applied in Northern Ireland. Other laws such as the "Health and Social Care Act 2016" are also applied alongside the GDPR implementation in Northern Ireland.

This table shows evidence that each country essentially relies on a set of national regulations. However, all countries studied except Estonia have at least one regulation that implements the GDPR (e.g. Privacy and Electronic Communications Regulations in Northern Ireland, Data Protection Act in Finland).

Therefore, it appears that most of the countries regulations regarding consent models are based on the GDPR, in addition to country-wide regulations and authorities. For that reason and to narrow further the scope of the wicked problem we address, the following of the background research related to regulations will essentially focus on the GDPR.

	Ireland	Northern Ireland	Estonia	Finland	Denmark
Health Services Organisation Act, 2001	✗	✗	✓	✗	✗
Data Protection Acts 1988-2018	✓	✓	✗	✓	✗
Health Research Regulations 2018	✓	✗	✗	✗	✗
GDPR	V	V	X	V	V
Health Identifiers Act 2014	✓	✗	✗	✗	✗
Health (Provision of Information) Act 1997	✓	✗	✗	✗	✗
Freedom of Information Act 2000	✗	✓	✗	✗	✗
Privacy and Electronic Communications Regulations 2003	✗	✓	✗	✗	✗
Health and Social Care (Data Processing) Act (Northern Ireland) 2016	✗	✓	✗	✗	✗
Code of Practice on Protecting the Confidentiality of Service User Information	✗	✓	✗	✗	✗
Personal Data Protection Act, 2007	✗	✗	✓	✗	✗
Regulation on the System of Security Measures for Information Systems 2007	✗	✗	✓	✗	✗
Law of Obligations Act, 2001	✗	✗	✓	✗	✗
Regulation on Conditions and Procedure for the Issue of Prescriptions for Medicinal Products and for the Dispensing of Medicinal Products by Pharmacies and the Format of Prescriptions 2005	✗	✗	✓	✗	✗
Penal Code, 2001	✗	✗	✓	✗	✗
Act on handling Customer Data in Health and Social Care (159/2007)	✗	✗	✗	✓	✗
Data Protection Act (Tietosuojalaki)	✗	✗	✗	✓	✗
Act on Electronic Prescription	✗	✗	✗	✓	✗
Act on Secondary Use of Health and Social data 2019	✗	✗	✗	✓	✗
Act on Processing of Personal Data (Persondataloven)	✗	✗	✗	✗	✓
Danish Act of Health (Sundhedloven)	✗	✗	✗	✗	✓
Ministry of Health and the Elderly's Data Protection policy	✗	✗	✗	✗	✓
Act on Research Ethics Review of Health Research Projects (no. 593, 14 June 2011)	✗	✗	✗	✗	✓

Table 2.1: Overview of legislation in place within EU countries implied in the study conducted by the HIQA (6) (with regulations relying on the GDPR in bold)

2.3.4 Consent forms within the European Union

General consent clauses and granularity

The background research and literature review has not revealed any forms which are used or suggested within the EU. Rather, consent forms are designed on a per-clinical trial basis and are expected to provide patients with enough material so that they can give their informed and explicit consent.

Already in 1998, Edwards et al. were investigating what could be the best possible method for collecting patients informed consent (23). They especially focused on the meaning of *informed* consent and tried to define the appropriate level of detail and granularity of information to provide patients with. Their findings revealed links between the quantities of information and consent rate, understanding and anxiety of the patients. For instance, while providing more information leads to more understanding from the patients, letting too much delay for patients to reflect seems to decrease the consent rate. There also seem to be an optimal level of detail to provide patient with in order to inform them appropriately while preventing an overburden of information, which could be a source of overthinking and anxiety for some patients. Ideally, the authors state that "autonomy is the foundation of informed consent" (23), although it is hardly feasible nor verifiable in reality.

The European Data Protection Board (EDPB) has published guidelines on consent under the EU 2016/679 regulation (i.e. GDPR) (45) which defines a set of minimum requirements that any informed consent should include to be valid. These requirements are the following:

1. the controller's identity
2. the purpose of each of the processing operations for which consent is sought
3. what (type of) data will be collected and used
4. the existence of the right to withdraw consent
5. information about the use of the data for automated decision-making where relevant
6. information on the possible risks of data transfers due to absence of an adequacy decision and of appropriate safeguards

Furthermore, the EDPB guidelines outline the high granularity and specificity that is expected regarding patients data and processing purposes in consent forms. For instance, a patient should not have to consent to a group of non-atomic processing purposes and should rather chose the specific purpose(s) they accept. Depending on the data and processing purpose, "several consents may be warranted to start offering a service" (45).

Particular clauses for rare disease research

The data of patients affected by rare disease is more subject to global sharing due to the scarcity of the cases. In addition to that, such data raises more important privacy issues as it is easier to re-identify patients with rare diseases. For that reason, the IRDiRC-GA4GH MCC Task Force met in order to determine which consent clauses are important within the context of rare diseases (7). This Model Consent Clauses Task Force has resulted from an alliance between the International Rare Diseases Research Consortium and the Global Alliance for Genomics and Health. Table 2.2 lists all clauses for both generic and rare disease consent which emerged from their work. Despite some additional specific clauses, it is interesting to see that consent clauses for rare diseases largely rely on generic clauses. In other terms, the clauses of a consent form for rare diseases does not appear to differ much from common diseases.

2.3.5 GDPR consent management in linked data

With the emergence of the GDPR and the many rules it implies regarding the management and processing of personal data, research has already been conducted in ICT in order to facilitate the compliance with such regulations. It appears that the compliance of consent with the GDPR has already been widely explored in the domain of linked data and ontologies.

Research conducted by the W3C's Data Privacy Vocabulary and Controls Community Group resulted in the creation of the Data Privacy Vocabulary, a vocabulary specific to the compliant handling of personal data (46) and aiming in enhancing interoperability in this domain. The notion of consent, as defined by the GDPR, is also a key constituent of this vocabulary. Furthermore, this high-level ontology has been created based on the Data Privacy Vocabulary and incorporates classes such as Processing, Purpose, Recipient and so on, i.e. classes directly related to notions established by the GDPR.

Such work has facilitated the creation of more specific ontologies, such as the consent ontology proposed by Fatema et al. which is based on the five requirements of the GDPR for a consent to be considered valid (i.e. freely given, specific, informed, unambiguous and with parental permission for child below 16 years old) (47). This consent ontology has been further investigated and iterated on by Pandit et al. to design the GConsent ontology (3) which resulted from the extension of the Provenance Ontology (PROV-O). The aim of the GConsent ontology, which is shown in Figure 2.3, is to be as compliant as possible with all facets of the GDPR regulation.

In addition to the definition of the aforementioned vocabularies and ontologies, Pandit et al. focused on evaluating their compliance with the GDPR (48). Based on existing consent ontologies, this particular article proposes a test-driven approach to evaluate the degree of

Generic clauses	Rare disease specific clauses
General information/Introduction (name of researchers, hospital/ institution, funders/sponsors, etc.)	Rare Disease Research Introductory Clause Familial Participation
Nature and objectives of the study Voluntariness of participation	Audio/Visual Imaging
Procedures involved in participation (what will happen during the study) /types of data and samples that will be collected	Collecting, storing, sharing of rare disease data Recontact for matching
Possibility of large scale genome-wide sequencing techniques Potential physical, psychological, social and informational risks Potential benefits of participation	Data Linkage
Protections in place [locally] to ensure security/privacy/confidentiality Duration/place of data/sample storage	Return of Results to Family Members Incapacity/Death
Hosting of data in an open access database	Risks and Benefits
Access to data/samples for research purposes (who will have access, types of access, governance framework, procedures in place – ex. data access committee), including access by pharma/industry, if applicable	
Access to data/samples for purposes of auditing, validation, control, etc.	
Return of research results/incidental findings (processes and potential inclusion in medical records)	
Withdrawal procedures (sample/data retrieval, destruction, no further contact, no further access, unlink, no further use, etc.)	
Compensation/reimbursement	
Prospects for commercialization and intellectual property procedures Study dissemination/publication	
Assent (where applicable)	
Re-contact	
Study oversight (IRB/REC/REB)	

Table 2.2: Generic and rare disease specific consent clauses defined by the IRDiRC-GA4GH MCC Task Force (7)

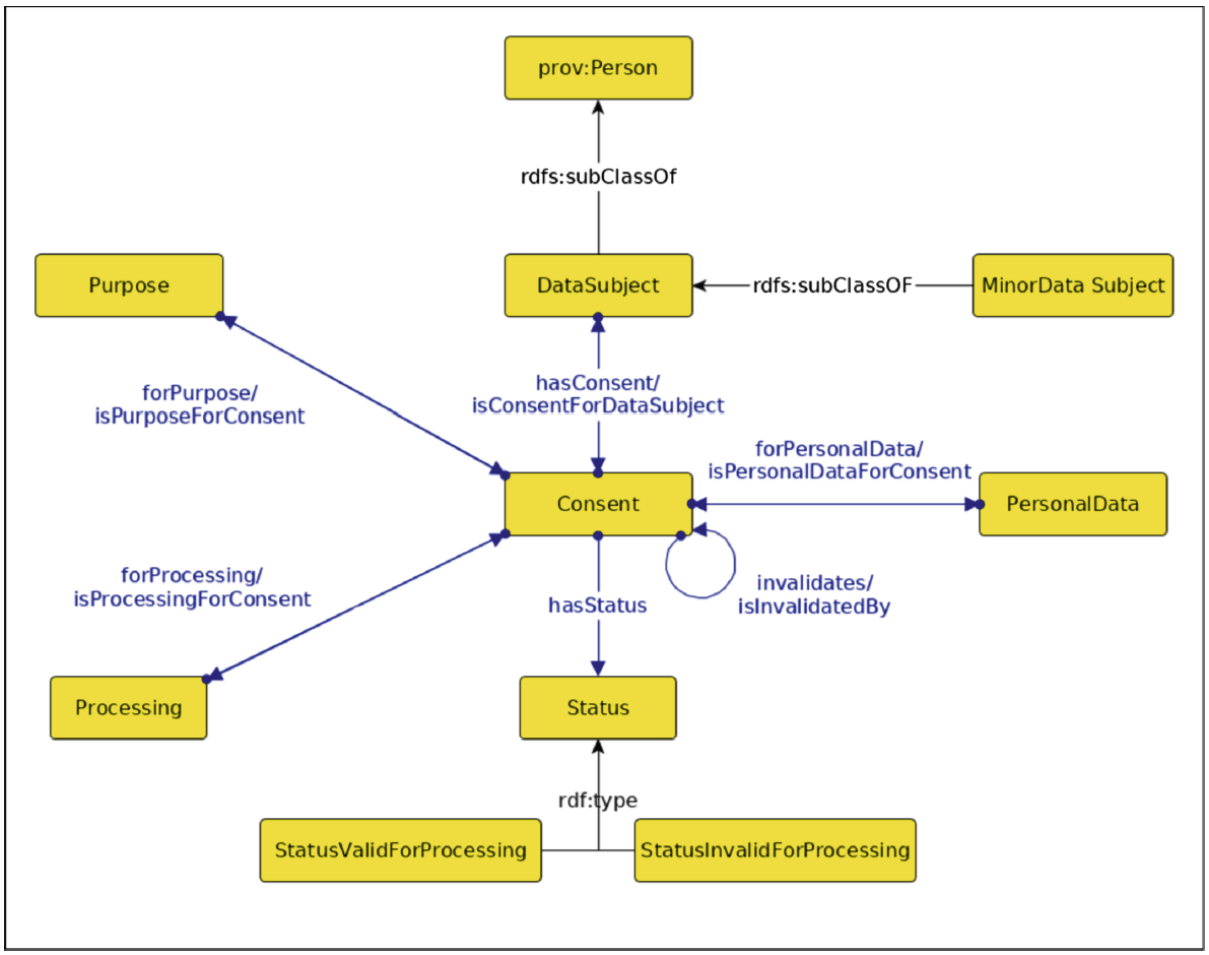

Figure 2.3: Overview of GConsent ontology created by Pandit et al. (3)

compliance of an organisation using personal data towards GDPR. Interestingly, the authors draw a list of constraints based on requirements of the GDPR. The test-driven approach consists in evaluating each constraint (i.e. with a pass/fail outcome); the more constraints pass, the more the organisation is compliant with the GDPR.

2.3.6 Blockchain technologies in clinical trials

Blockchain technologies have already been largely investigated in healthcare. However, fewer applications of these technologies have been conducted within the specific scope of clinical trials. This section establishes a review of the state-of-the-art usage of blockchain technologies in clinical trials. First, an overview of blockchain application to clinical trials is proposed. Thereafter, the usage of smart contracts is discussed. Finally, a list of limitations that currently exist is drawn.

Is blockchain really needed?

When using blockchain to address a particular use-case, one of the first question is to determine whether it is a relevant choice. For that purpose, Wüst and Gervais designed a

straightforward decision framework (4) which is displayed in Figure 2.4.

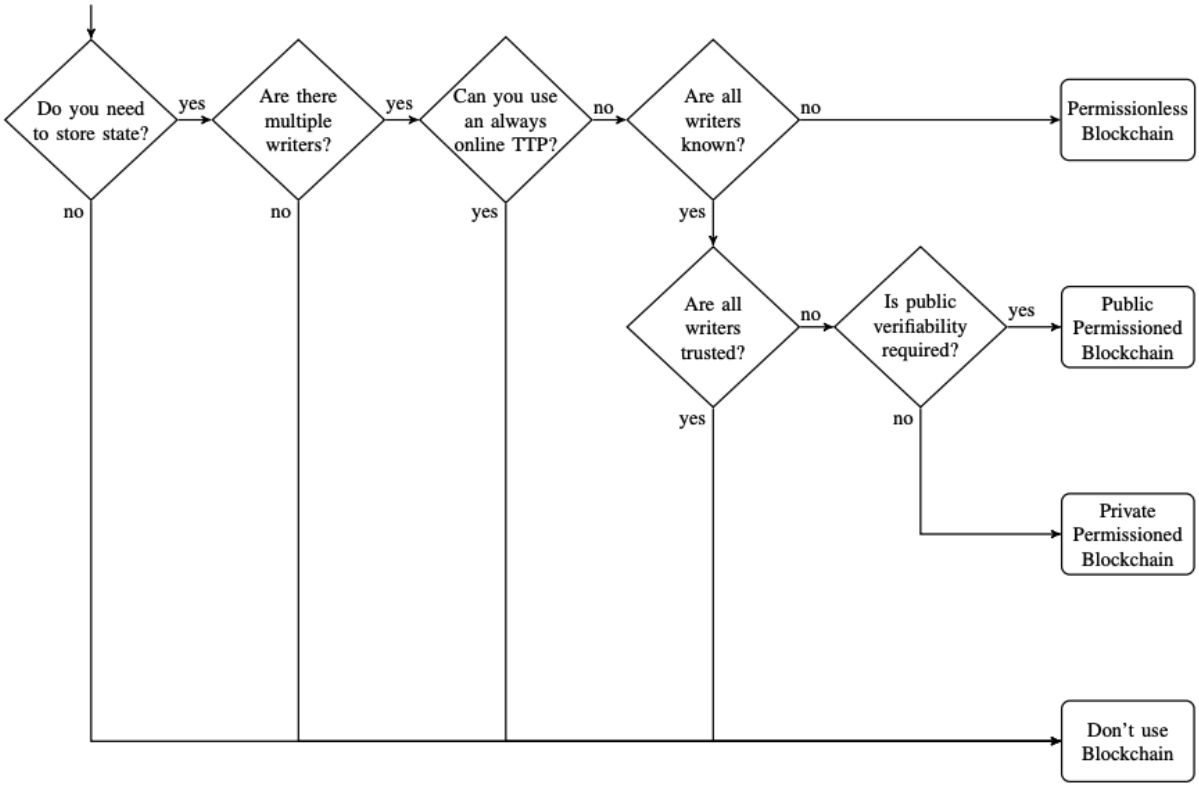

Figure 2.4: Blockchain decision framework created by Wüst and Gervais (4)

The questions of this decision process can be answered within the context of consent management in clinical trials:

- Do you need to store state? **Yes**, state of patients consent has to be stored.
- Are there multiple writers? **Yes**, multiple patients could write data (i.e. give their consent), as well as researchers and/or doctors.
- Can you use an always online trusted third party (TTP)? **No**, although it would be technically feasible; it remains one reason for applying blockchain, i.e. provide a transparent register of patients consent based on a consortium instead of a single and centralised TTP.
- Are all writers known? **Yes**, in the sense that a patient should be known by the blockchain (i.e. authenticated or authorised) to grant their consent.
- Are all writers trusted? **No**, although patients should be authenticated, they cannot be considered as a trusted authority when writing data.
- Is public verifiability required? **No**, verifiability is required at least between patients and researchers involved in the clinical trial. Data should also be verifiable in case of an audit, which could be conducted internally (i.e. not publicly).

Although this framework remains simple and its result cannot be taken for granted regarding the complexity of the problem of managing patients consent in clinical trials, it provides an orientation towards the establishment of a 'Private Permissioned Blockchain'.

Blockchain for dynamic consent management in clinical trials

The dynamic consent management using blockchain technology proposed by Albanese et al. in February 2020 appears to be amongst the most up-to-date and relevant articles that has been published until now (49). This paper essentially focuses on the usage of BCT to address the lack of control over granularity that currently exists in consent management as well as the evolution of a consent in time, i.e. dynamic consent. To address these issues, the authors leverage a wide range of features of blockchains such as decentralised trust management, immutability of records, authentication (i.e. permissioned blockchain), global consistency updates within the ledger and integrity of transactions performed on the blockchain. The solution is implemented and evaluated using Hyperledger Fabric, an open source and permissioned blockchain maintained by the Linux Foundation and designed for enterprise applications (50).

Furthermore, Albanese et al. have taken the initiative of designing their Blockchain as a Service (BaaS) in order to address the heterogeneity and incompatibility of Clinical Trial Management Software (CTMS) (e.g. REDCap, OpenClinica, Phoenix CTMS, proprietary systems). Within their research, they noticed that paper-based consent forms are the norm. Once a consent form is signed by a patient, the status of the consent is usually reported within the CTMS in standard database fields. Their work therefore focuses on evaluating the feasibility of using blockchain to ease the process of collecting patients consent by replacing forms with a BaaS.

According to the system design proposed in (49), data insertion and access is also meant to be handled by the blockchain. This conception choice extends the scope of the BaaS and results in a wider artifact from a design-science perspective. Considering the heterogeneity of patients data, focusing on BaaS for consent management could potentially facilitate its integration within existing systems which already handle patients data (e.g. EHRs, databases, spreadsheets). Furthermore, this work leads to a centralised dashboard on which patients can handle their consent; without making explicit whether this consent management dashboard could be implemented within existing systems using the proposed BaaS.

Patients data sharing

Shah et al. discusses patients consent to share their data. Their article focuses on an in-depth evaluation of smart contracts which support data sharing between 'patients', 'providers' and 'viewers'. In this paper, the definition of patients consent is not taken from a

regulation perspective such as the GDPR. Rather, consent is considered as basic approvals which are explicitly given by patients. Therefore, no clauses specific to regulations are considered in the paper (e.g. data processing purpose, consent validity time frame, clinical trial name).

The authors aim in "restoring ownership over medical data to the patients themselves" (51) by creating an ecosystem for data sharing between patients, healthcare providers and researchers. For this purpose, the proposed smart contracts handle data creation, viewer authorisation and data transfers; while providing patients settings to control how their data is shared using group-based access rights (e.g. providing access to a subset of their data to a group of viewers, providing full access and writing rights to their doctors). Finally, the authors also propose an approach using Proof-of-Work consensus with the idea to reward patients to encourage them sharing their data.

In (51), an in-depth evaluation of smart contracts usage for medical data sharing from a technical perspective is conducted, with an implementation relying on the public blockchain Ethereum. For instance, gas consumption, throughput and latency are thoroughly evaluated in networks composed of more than 500 nodes. However, there is a lack of compliance with regulations as consent depends on 'permission strings' determined by the patients and stored within the contracts.

Nevertheless, this paper raises the interesting question of determining where patients data should be stored. In the prototype, data creation and storage is delegated to a SQL database to which are referring queries stored in the blockchain (i.e. off-chain data storage). While this choice can be justified by technical limitations, Millard focuses on the compatibility between blockchain and law from the perspective of data protection (52). In his paper, he identifies a citation from Jan Philip Albrecht "who played a prominent role in the development and finalisation of the EU's [GDPR]" (52) and which clearly designates BCT as not suitable to store personal data because of their non-compliance with the GDPR's subjects rights. For instance, blockchains would not provide patients the ability to delete their data because of the immutable intrinsic design of blockchains from which no transaction can be deleted.

Limitations

In addition to proposing two different approaches to apply blockchain technologies in consent management in healthcare, (49) and (51) also identify limitations.

For instance, there is an evidence that the state-of-the-art work aiming in applying BCT to consent management in clinical trials lack of evaluation regarding compliance with regulations such as the GDPR. Current BCT implementations in this domain seem to rely on privacy which is implicitly offered in blockchain technologies 'by design', without further

evaluation.

Finally, it remains unclear from conclusions drawn in previous work what exactly should be the scope of BCT application to consent management in clinical trials. For instance, whether data should be handled by the BaaS as pointers to databases or if this should be considered out of the scope of the consent management system to maximise interoperability.

2.4 Knowledge gap

Owing to the literature review and background research, this section aims in clearly stating the knowledge gap that has been identified.

There is a need to allow patients gain more control over their health data, or at least knowing what such data contains and who has access to their data. Reliable, transparent and secure consent management seems to be a great enabler for patients privacy by allowing them to choose which data to share, to whom, and for what purpose. Properly done, such a mechanism could enforce both privacy on the patients end, but also security of data by limiting its usage by unauthorised or untrustworthy third parties.

By its intrinsic properties, there is an evidence that blockchain could help address the problem of consent management for clinical trials. However, despite the endeavours to shape blockchains for clinical trials, there is still a lack of evaluation regarding compliance with regulations such as the GDPR. While the approach of Blockchain as a Service appears to be suitable, most of the state-of-the-art implementations merge consent management with other subproblems such as data sharing, patients rewarding or storage of patients data. Relying on the design-science framework helps focusing on a very single subproblem – consent management in clinical trials – which includes regulation adherence. This particular focus encourages the design of an artifact which provides stronger interoperability.

2.4.1 Genre of inquiry to explore

As described in section 2.2.3, four genres of inquiry are defined in design-science. Figure 2.5 approximates where some of the most important papers referenced in background research situate from the perspective of their genre of inquiry.

It is important to note that the genres of inquiry are not absolute classes. It rather helps in situating references relatively to one another. For instance, this dissertation specifically addresses patients consent management in clinical trials, which could situate it within the idiographic genre. However, other related work such as Smarter Smart Contracts' paper (51) addresses even more specifically the idea of data sharing from technical perspectives. In comparison, this dissertation addresses a larger class of problems. There is no global scale or

granularity from which this diagram should be perceived; it rather supports further discussion for refinement of the scope of the problem being addressed.

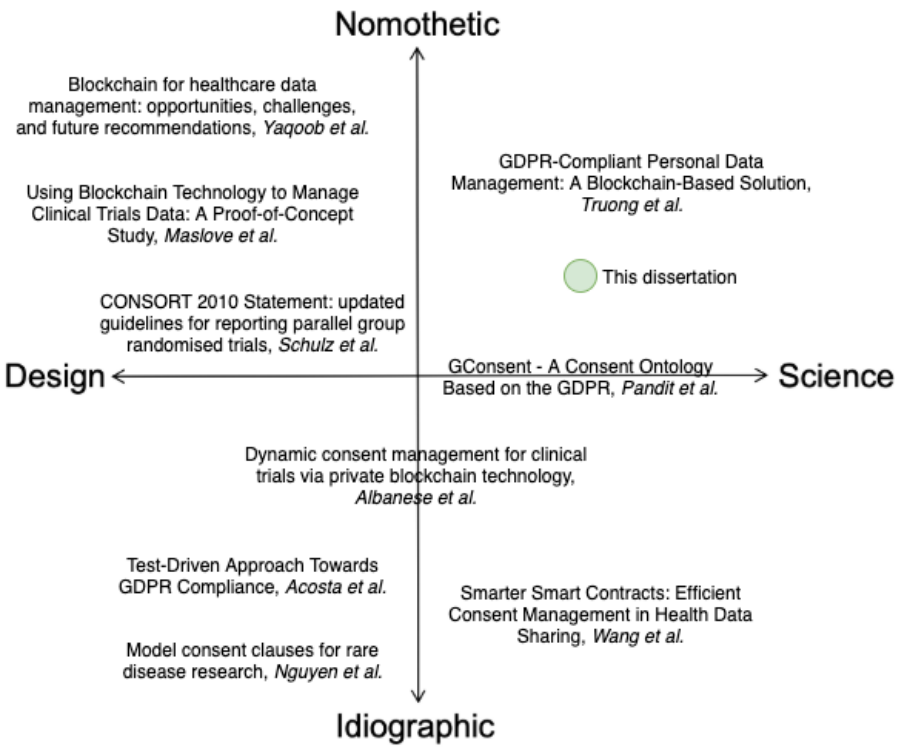

Figure 2.5: Overview of the genre of inquiry of related work, modified from (5)

Although genres of inquiry are not strict classes, there is an evidence that few research has been conducted in both nomothetic design and science for the specific context of applying blockchain to consent management in clinical trials. Blockchain-related papers rather seem to address specific problems in consent management, leading them mostly in the idiographic genre. Related work regarding GConsent ontology is mixed between nomothetic and idiographic science as it proposes a specific answer to a generalised class of problems while proceeding to its concrete evaluation.

This dissertation aims in focusing on the nomothetic science genre by proposing a solution to the general problem of consent management in clinical trials. It differs from previous work by offering a solution which maximises interoperability, while applying concrete and testable evaluation methods to address the knowledge gap which has been previously identified.

2.5 Conclusion

Having defined the essential terms and introduced key concepts of the design-science research framework, background research helped understanding the current state-of-the-art of consent management for clinical trials. Furthermore, background research emphasises the

multifaceted nature of clinical trials and, more generally, healthcare; which makes relevant the approach of designing as artifacts and resolving wicked problems.

The iterative approach in which the literature review has been conducted led to the identification of a knowledge gap that exists for managing patients consent in clinical trials. Using a BaaS to address patients consent management for clinical trials seems appropriate. However, this approach currently lacks of evaluation regarding its compliance with regulations (e.g. the GDPR). Furthermore, there is a need to design an artifact which addresses the sole problem of consent management (i.e. without considering related subproblems) while maximising its interoperability with other artifacts.

In the next chapter, research design and methods are described.

3 Research design and methods

This chapter describes the research design and methods that will be used. First, explanations are given about how design-science guidelines are applied to this project. In the second section, details are provided on the evaluation methods. Finally, conclusions of the chapter are drawn.

3.1 Following design-science guidelines

The design-science framework proposes a set of guidelines presented in section 2.2. To some extent, all of these guidelines are followed in this project:

3.1.1 Guideline 1 – Design as an Artifact

The implementation of a BaaS for consent management in clinical trials is the main artifact designed within this project. The evaluation process, especially regarding compliance with regulations, could be considered as another relevant artifact which could be applied for further evaluations (e.g. with other regulations or local implementations of the GDPR).

3.1.2 Guideline 2 – Problem Relevance

The initial problem comes from the assumption that patients do not have enough control over their data. The iterative approach used within background research allowed to clearly identify the problem: patients do not have enough control over their data. It is therefore important to propose solutions that aim in empowering patients to regain this control. This is outlined by the knowledge gap that exists between BaaS for consent management and evaluation of their compliance with regulations.

3.1.3 Guideline 3 – Design Evaluation

The evaluation process, which could also be considered as an artifact, is based on the application of competency questions to blockchain. Although competency questions are

commonly used for evaluating ontologies, it is a new approach in the context of blockchains evaluation.

3.1.4 Guideline 4 – Research Contribution

The artifact designed within this research project is described in details. The source code of the implementation is also publicly available on GitHub (see Appendix A1.1). Furthermore, the provided artifact builds upon previous work in many complementary domains.

3.1.5 Guideline 5 – Research Rigor

This research relies on the analysis of previous, state-of-the-art work within the domain of blockchain usage for consent management in clinical trials as well as related topics (i.e. BCT, consent management in clinical trials, regulations).

3.1.6 Guideline 6 – Design as a Search Process

The aforementioned rigor has been required to properly conduct background research with an iterative and empirical approach. This search process especially helped in identifying and refining the wicked problem addressed. Therefore, the design of the artifact benefits from this search process and will be implemented following a similar build-and-evaluate approach described in section 3.2.3.

3.1.7 Guideline 7 – Communication of Research

The communication of research is made through this dissertation report as well as the open-source availability of the source code on GitHub.

3.2 Evaluation

This section describes the evaluation process that is used for this research project. As the usage of blockchain for consent management in clinical trials is a cross-disciplinary problem, a multifaceted evaluation will be conducted with a strong focus on compliance with regulations. In consequence, the first section focuses on evaluating compliance with both regulations and clinical trials consent clauses. Thereafter, the evaluation of feasibility is described. Finally, the notion of evaluation by implementation is introduced.

3.2.1 Consent form clauses

The consent form clauses are borrowed from (7) (see 2.2). The aim of these clauses is to ensure that patients are given enough information when offered to enrol within the clinical

trial and to autonomously take the decision to either accept or reject the consent request.

3.2.2 Compliance with regulations

According to background research, the GDPR is the regulation from which country-wide implementations derive the most. Hence, for the purpose of evaluating compliance with regulations in this dissertation, the GDPR will be taken as a basis. (48) and (3) respectively draw lists of constraint and competency questions which ensure compliance with the GDPR. Although competency questions are usually used to evaluate ontologies, we propose to apply these questions to blockchain in order to evaluate its compliance with regulations. For the purpose of this dissertation, the following competency questions established by Pandit et al. for their evaluation of the GConsent ontology (3) are used to evaluate the compliance of the BaaS implementation:

Questions about consent:

- C1 Who is the consent about?
- C2 What type of Personal Data are associated with the Consent?
- C3 What type of Purposes are associated with the Consent?
- C4 What type of Processing are associated with the Consent?
- C5 What is the Status of Consent?
- C6 Is the current status valid for processing?
- C7 Who is the consent given to?

Questions about the context of how consent was created/given/invalidated:

- T1 What is the location associated with consent?
- T2 What is the medium associated with consent?
- T3 What is the timestamp associated with the consent?
- T4 What is the expiry of the consent?
- T5 How was the consent acquired/changed/created/invalidated?
- T6 What artefacts were shown when consent was acquired/changed/created/invalidated?

Questions related to Third Party associated with the consent:

- D1 Is the purpose or processing associated with a third party?

- D2 What is the role played by the third party in the purpose or processing?

3.2.3 Evaluation by implementation

The design-science relies on the idea of a build-and-evaluate loop which aims to improve the artifact on each iteration. This loop underlies the usage of an iterative and incremental development model were each development cycle focuses on a specific set of features (e.g. use-cases, requirements) until all requirements are met.

These model and principle of design-science also imply an evaluation by implementation, i.e. a perpetual evaluation which is part of the development cycles. Design-science describes this as a creative process in which "the design-science researcher must be cognisant of evolving both the design process and the design artifact as part of the research" (33).

3.3 Conclusion

The BaaS is evaluated against consent form clauses, compliance with regulations and is implemented following the build-and-evaluate loop proposed by design-science. Therefore, the main outcomes of this evaluation process are (i) an evaluation of the feasibility of the implementation of the BaaS and (ii) an evaluation of compliance with regulations through the example of the GDPR regulation.

From a design-science perspective, this evaluation is at the border of a controlled experiment – evaluating the BaaS in a controlled environment – and structural testing which is performed through "coverage testing of some metric [...] in the artifact implementation" (33); in this case the coverage of competency questions.

The next chapter details the design and implementation of the BaaS.

4 BaaS design and implementation

This chapter provides in-depth information regarding the design of the Blockchain as a Service (BaaS) and its implementation. First, use-case scenarios are described to specify the scope of the BaaS. In the second section, (non-)functional requirements addressing both use-cases and competency questions are described. In the third section, technical choices are explained. Thereafter, the architecture of the BaaS is described. In the fourth section, details on the implementation of the BaaS are given and, finally, conclusions are drawn.

4.1 Use-case scenarios

This section describes all use-case scenarios that should be supported by the implementation of the BaaS. First, the creation of a clinical trial within the BaaS is described. Then, the request of a consent from a patient is presented. From a patient perspective, the acceptance, rejection and revocation of consents are introduced. Finally, the update of a clinical trial by a researcher is discussed. These use-cases are summarised within Figure 4.1.

All use-cases involve two possible actors: 'researcher' and 'patient'. The 'researcher' actor is provided as an example and could interchangeably represent a whole organisation (e.g. hospital, research center) in real-world scenario. As a grammatical note, the plural notion of 'consents' is introduced within the implementation as a patient can have multiple consents related to different clinical trials. In that context, a patient consent could be seen as a single form or document that is virtually possessed by the patient on the blockchain.

In addition to the use-cases described below, it could be interesting to consider patients consent given on their own initiative. For instance, allowing a patient to purposefully give consent to share all their clinical data for any purposes. However, as the implementation of the GDPR usually requires high granularity of statements for data sharing, definitions of processing and purposes, this use-case does not seem realistic. It is therefore not considered as part of the ones described below. Rather, the direct exchange between a researcher and a patient seems to be the most relevant approach within this context: (i) the researcher creates the clinical trial within the blockchain, (ii) the researcher asks a patient to give a

consent and (iii) the patient can either accept or reject the consent request.

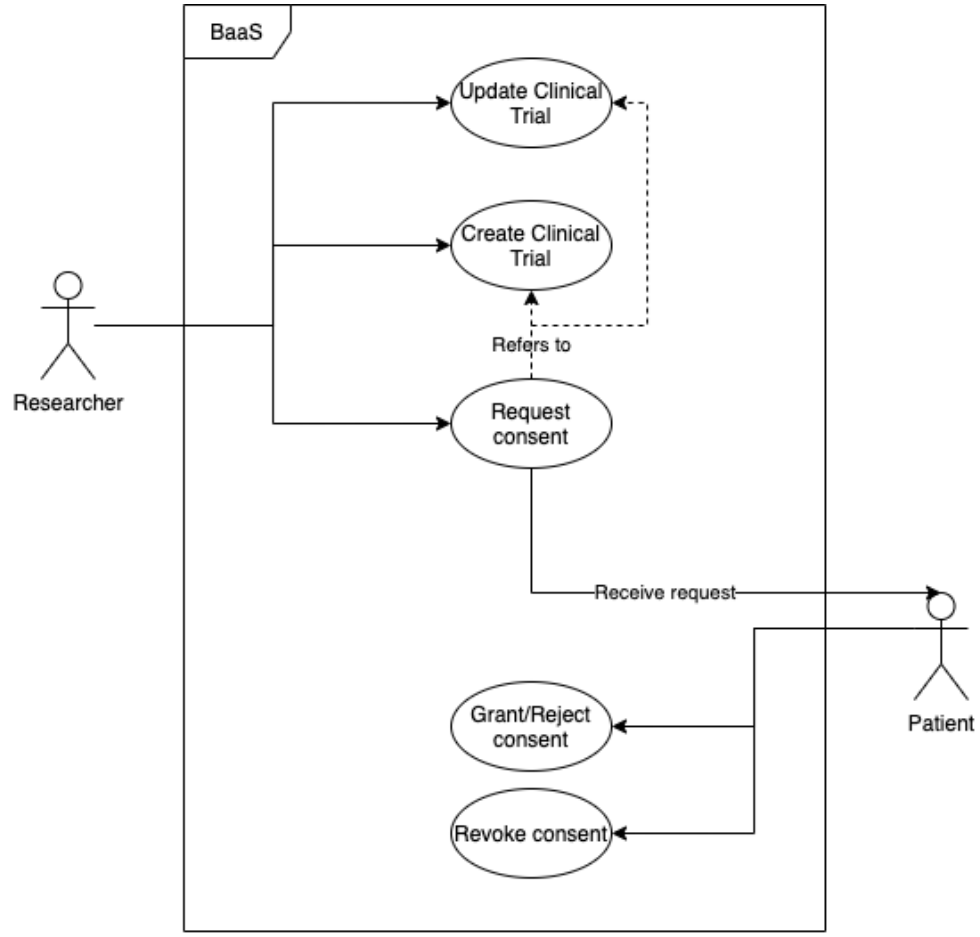

Figure 4.1: Use cases diagram

4.1.1 UC1 – Create a clinical trial

Use-case	Create (i.e. register) a clinical trial in the blockchain
Actors	Researcher
Pre-conditions	Have researcher rights within the smart contract
Post-conditions	N/C
Scenario	1. The researcher specifies all information relative to the trial 2. The request is sent to the Smart Contract to register the clinical trial in the blockchain 3. Once the transaction has been registered, its transaction hash (TXH) is returned
Alternative scenario	In case information about the clinical trial are missing or if any other error occurs, the transaction is reverted

Table 4.1: Use-case 1 – Create a clinical trial

4.1.2 UC2 - Request consent from patient

Use-case	Request consent from patient (e.g. for enrolment in a clinical trial)
Actors	Researcher and patient
Pre-conditions	Patient address must be known
Post-conditions	N/C
Scenario	1. The researcher specifies all information relative to the consent request 2. The request is sent to the Smart Contract to register the consent request in the blockchain 3. The consent request is stored within patient's consents with the status 'requested' 4. Once the transaction has been registered, its transaction hash (TXH) is returned
Alternative scenario	In case information about the consent request are missing or if any other error occurs, the transaction is reverted

Table 4.2: Use-case 2 – Request consent

4.1.3 UC3 - Consent request acceptance/rejection by patient

Use-case	Acceptance/rejection of consent request by patient
Actors	Patient
Pre-conditions	Patient has received a consent request that has not already been accepted or rejected
Post-conditions	N/C
Scenario	1. The patient indicates the reference to the consent in their request and their choice to accept or reject it 2. The request is sent to the Smart Contract to update the status of the consent indicated by the patient 3. The patient consent is updated with the status 'granted' (if the request is accepted) or 'rejected' otherwise 4. Once the transaction has been registered, its transaction hash (TXH) is returned
Alternative scenario	In case the consent is not found or if any other error occurs, the transaction is reverted and the consent remains in 'requested' status

Table 4.3: Use-case 3 – Consent request acceptance/rejection by patient

4.1.4 UC4 - Update a clinical trial

When a clinical trial is updated (e.g. researchers, associated data, processing, purposes or any other clause), the GDPR requires patients to explicitly renew their consent. Therefore,

all consents associated with a clinical trial that is updated should be put 'on hold' until the patient has re-consented to the new terms of the clinical trial. The third point of this use-case scenario (see Table 4.4) forces the early expiration of granted consents associated with the updated clinical trial and creates a request similar to UC2 (section 4.1.2).

Use-case	Clinical trial update
Actors	Researcher
Pre-conditions	Researcher has already created a clinical trial
Post-conditions	If consents are associated with the updated clinical trial, they are automatically revoked and new consent requests are made to the patients in order to accept new terms of the clinical by re-consenting
Scenario	1. The researcher indicates the clinical trial to update and all information that have to be updated within their request 2. The request is sent to the Smart Contract to update the clinical trial 3. All consents associated with the updated clinical trial are set with status 'expired' 4. New consents associated with the update clinical trial are created with the status 'requested' 5. Once the transaction has been registered, its transaction hash (TXH) is returned
Alternative scenario	In case the clinical trial is not found or if any other error occurs, the transaction is reverted

Table 4.4: Use-case 4 – Clinical trial update

4.2 Requirements

Table 4.5 describes both functional and non-functional requirements that have been defined in order to address the aforedescribed use-cases and meet competency questions requirements. Non-functional requirements also relate to the principle of design-science of design as an artifact, i.e. designing a self-sustainable artifact which answers a specific problem while maximising its interoperability with other artifacts.

4.3 Technical choices

This section describes all technical choices that have been made in order to address the use-cases, functional requirements and competency questions previously stated. The first subsection introduces Hyperledger Besu, the blockchain which is used in the project. Then, the Smart Contract programming language Solidity is presented. Thirdly, the choice of Node.js is explained for the development of the 'as a Service' part of the blockchain

Functional requirement	Non-functional requirement
F1. API interface to create and update clinical trials	NF1. Interoperability
F2. API interface to create consent requests	NF2. Privacy and security
F3. API interface to accept or reject consent requests	NF3. Integrity and accountability
F4. API interface to read clinical trials and patients consents	NF4. Ease of usage and deployment
F5. Abstraction of the Blockchain layer when interacting with the API	
F6. Segmentation of rights for researchers and patients	

Table 4.5: Requirements for the BaaS

implementation. Finally, Truffle Framework and Web3 are successively introduced to give details about how interactions with the blockchain are performed.

4.3.1 Hyperledger Besu

Hyperledger Besu is an open-source blockchain project developed by the Linux Foundation. It is the latest blockchain released within the Hyperledger Project (53), after having been announced in august 2019. The Hyperledger Project is an ecosystem which aims in providing open-source and entreprise-grade level blockchains to ease the building of BCT-based frameworks and applications.

Besu is a Java-based Ethereum client which has the particularity to operate on both public and permissioned blockchains [1]. Besu also enables the development of application under private instances of blockchains based on Ethereum. Furthermore, as Ethereum relies on gas to perform transactions (i.e. transaction fees are paid with Ethereum gas), Besu proposes a gas-free configuration to facilitate blockchain implementation wihtin systems which do not require transaction fees.

Furthermore, unlike other projects such as Hyperledger Fabric or Sawtooth, Hyperledger Besu handles a wide range of consensus algorithms including Proof-of-Work (PoW) as well as Proof-of-Authority (PoA) consensus (50). Besu permissionning schemes have been specifically developed to be used within consortium environments [2].

For these reasons, Hyperledger Besu seems to be interesting within the context of consent management for clinical trials. For instance, relying on a private blockchain rather than a

[1] Announcing Hyperledger Besu, https://www.hyperledger.org/blog/2019/08/29/announcing-hyperledger-besu

[2] Hyperledger Besu, https://www.hyperledger.org/use/besu

public one ensures the confidentiality of patients consents by keeping them inside a hermetic network, while enabling transparency of transactions. PoA consensus is also more relevant than PoW as transactions could be handled by private nodes distributed across research organisations and hospitals.

While alternatives exist such as Quorum, another Ethereum-based open-source blockchain project devleoped by ConsenSys [3], it is worth noting that the focus on evaluation of compliance with regulations could be performed similarly with other blockchain solutions, for instance Hyperledger Fabric which has been used in (49).

4.3.2 Solidity

Solidity is the programming language which is used to create smart contracts in Ethereum blockchains. It is an OOP language influenced by C++, Python and JavaScript [4]. Once compiled, a smart contract can be deployed and stored within the blockchain. It has its own address, which can be called to execute public methods defined within the contract. Smart contracts can be executed by anyone else who has access to the blockchain, although smart contracts' methods can specify additional requirements (e.g. user verification based on their address). The 'Hello World' contract snippet below issued from Solidity by Example [5] gives a brief overview of Solidity's syntax:

```
1 // SPDX-License-Identifier: MIT
2 // compiler version must be greater than or equal to 0.7.6 and less than
   ↪  0.8.0
3 pragma solidity ^0.7.6;
4
5 contract HelloWorld {
6     string public greet = "Hello World!";
7 }
```

4.3.3 Node.js

JavaScript (JS) is a programming language based on ECMAScript specification. It is a multi-paradigm language which is particularly used in web applications (e.g. most of front-end web frameworks rely on JS). Node.js – often abbreviated Node – is a JavaScript runtime which allows to run JavaScript outside of web browsers. Leveraging a non-blocking, callback-based approach, Node.js enables to use JavaScript on the back-end and to build a large variety of scalable programs [6].

[3] ConsenSys Quorum, https://consensys.net/quorum/
[4] Solidity 0.8.7 documentation, https://docs.soliditylang.org/en/v0.8.7/
[5] https://solidity-by-example.org/hello-world/
[6] About Node.js, https://nodejs.org/en/about/

Combined with frameworks such as Express, a minimalist and flexible web framework for Node, Node.js is fairly convenient to build web applications, including APIs [7]. The following example demonstrates how to create a basic HTTP server using Node.js and Express:

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => { res.send('Hello World!') });
app.listen(port, () => {
    console.log('Listening at http://localhost:' + port);
});
```

4.3.4 Truffle Framework

Truffle Suite is a set of tools that ease development on blockchains. The main tool, Truffle, provides an environment for smart contracts development, testing and deployment through the Ethereum Virtual Machine (EVM). Other tools such a Ganache and Drizzle are respectively designed to provide a local development blockchain instance to test smart contracts and to provide front-end libraries for decentralised applications (dApps) development [8].

Truffle is especially useful for smart contracts as it eases their deployment on the blockchain. The deployment of a smart contract requires the following steps that are handled by Truffle once configured properly, allowing to rapidly deploy new contracts on the blockchain:

1. Compiling the contract to obtain (i) its binary code and (ii) its Application Binary Interface (ABI), a description of all methods (and their arguments) that can be called to interact with the contract
2. Establishing a connection with the blockchain using a client (e.g. web3.js)
3. Creating a new transaction containing the smart contract binary code and its ABI

4.3.5 web3.js

Web3.js is an Ethereum client which allows to interact with a blockchain through a node, either local or remote, using either HTTP, IPC or WebSocket. The client provides many libraries and modules to perform a wide range of operations on the blockchain such as connecting to the blockchain, performing transactions (i.e. with users or smart contracts) or reading information about blocks and transactions that have been performed on the blockchain.

[7] Express, https://expressjs.com/
[8] Truffle Suite, https://www.trufflesuite.com/

It is worth noting that Truffle relies on web3.js to perform interactions with the blockchain. While transactions and interactions with the blockchain could be performed at a low-level using web3.js, Truffle and the related library Truffle-contract ease these interactions by providing higher-level interfaces, which are used in the implementation proposed in this dissertation.

4.4 Architecture

This section describes the architecture of the BaaS. The diagram shown on Figure 4.2 illustrates the difference between consent management without BaaS (paper-based consent forms) and with BaaS. One clear added feature is the notion of dynamic consent, i.e. a consent which evolves along with the clinical trial. However, this feature is not specific to BaaS and could be implemented with other information systems, for instance a standard SQL database. The purpose of this diagram is rather to situate where the BaaS artifact should be implemented, while important concepts of blockchain which are being evaluated are its privacy-preserving mechanisms (e.g. consensus, data integrity, storage in smart contract) against regulations.

The first subsection discusses the technical architecture of the blockchain. Thereafter, the data model is presented.

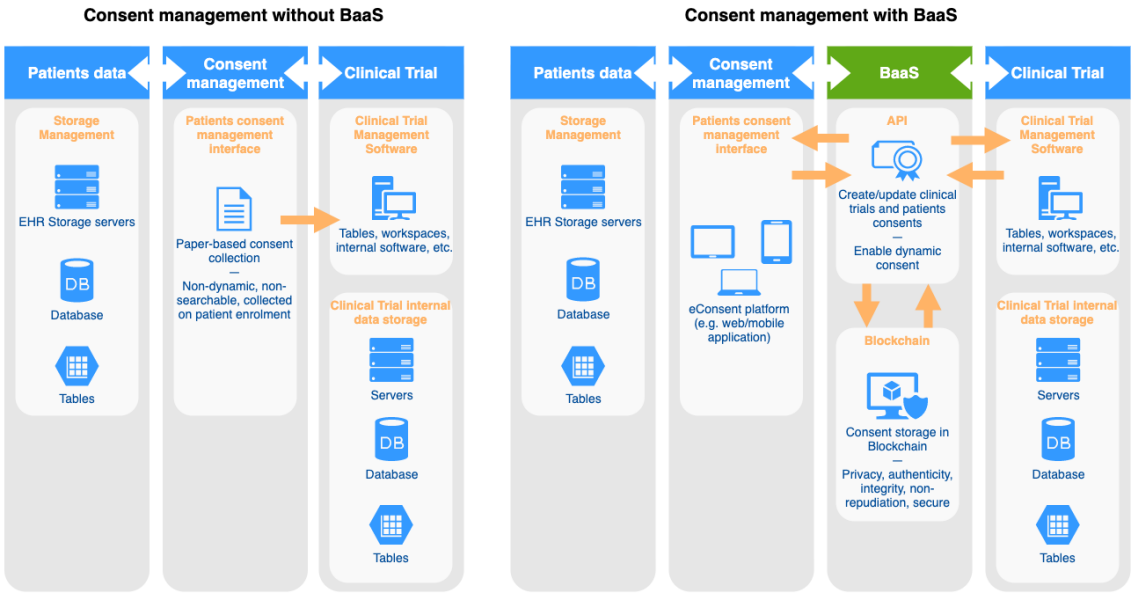

Figure 4.2: Consent management without/with BaaS

4.4.1 Technical architecture

Figure 4.3 depicts where each technology described in section 4.3 stands within the proposed architecture.

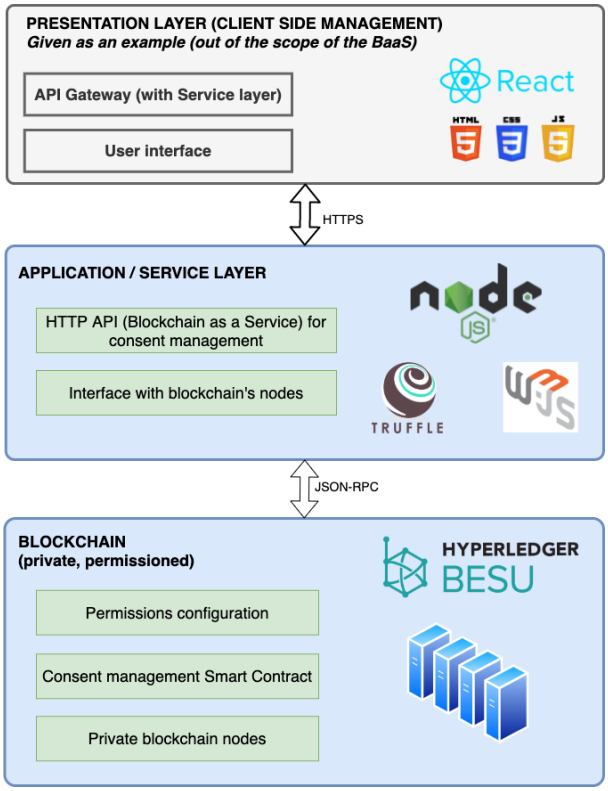

Figure 4.3: BaaS technical architecture diagram

From bottom to top, the first layer is composed of the blockchain, which relies on a private instance of Hyperledger Besu. This blockchain project has been chosen as it is particularly convenient for consortium environments implementing PoA consensus algorithms. Based on EVM (Ethereum Virtual Machine), it also allows to deploy a permissioned blockchain including smart contracts written in Solidity.

The service layer provides an API to facilitate interaction with the blockchain by clients. In other terms, this API creates an abstraction of all operations related to the blockchain (e.g. interaction with smart contracts, necessity of knowing contracts' ABIs, connection with a blockchain client) so that client applications which are connected to the blockchain can interact with it using simple HTTP requests (i.e. without requiring knowledge of the underlying blockchain implementation). This layer is developed in Node.js using the Express framework and implements Truffle-contract as well as web3.js – JavaScript clients for Ethereum blockchains – to interact with Hyperledger Besu.

Finally, the first layer is given as an example as it is out of the scope of this dissertation. This presentation layer is expected to provide users (i.e. researchers and patients) with an interface to manage their clinical trials and consents. It could take, for example, the form of a web-application based on languages such as HTML, CSS and JS. This layer can interact with the BaaS using basic HTTP requests as designed within the service layer. The reason for not providing a presentation layer is that any existing EHR management software or

CTMS can implement its own user interface with the BaaS for managing consents.

4.4.2 Data model

The data model proposed in Figure 4.4 essentially relies on the clauses provided by the MCC Task Force (7).

The model also proposes to segregate clinical trials and consents. This distinction allows to bind many consents to the same clinical trial. Therefore, if any clauses of the clinical trial change (e.g. the requirement of additional patients data), all consents bound to this trial can automatically be 'expired' in order to ask patients to renew their consent for the new clauses.

A 'parent ID' has also been introduced in both clinical trial and consent entities. This ID refers to a previous version of the same entity, with '0' meaning that no previous version exists. For instance, if a new clinical trial is created within the blockchain, its parent ID will be 0 and its ID *N*. If this trial is edited, a new version will be created in the blockchain with the parent ID *N*, i.e. the ID of its previous version.

For the purpose of this dissertation, it is considered that all patients are asked for the same clauses. Hence clauses are mostly part of the clinical trial model. For instance, that means that a patient A cannot be asked for different data than patient B if both participate to the same clinical trial, as their respective consents refer to the same clinical trial. If this happens not to be true in a real case usage, the clauses could be interchangeably placed either in the 'clinical trial' or the 'consent' without affecting the BaaS.

4.5 Implementation details

This section provides details about the implementation of the BaaS. The first subsection focuses on the blockchain configuration. The second describes the development and deployment of the smart contract. Finally, the last subsection considers the development of the service layer (i.e. the API).

4.5.1 Blockchain configuration

The chosen blockchain, Hyperledger Besu, offers an important flexibility through its configuration options. The configuration which has been used for this project is described in this section. All configuration options are defined in three places:

- The genesis file: a configuration file which stores information about the genesis block (i.e. the first block) of the blockchain. Each node which joins the blockchain needs to

have information about this genesis block (see Appendix A1.4.1 for complete configuration file).

- The node's configuration file: a configuration file which is proper to each node joining the network and specifies information such as network ports to use or the 'bootnode' (i.e. another node to connect with and which is already in the network) (see Appendix A1.4.1 for complete configuration file).
- The node command line interface: additional configuration options than the ones specified in the node's configuration file can directly be appended to the start command of the node.

Private blockchain

Ethereum is a public permissionless blockchain which relies on mining (i.e. PoW consensus algorithm). For this project, the blockchain setup is a private Ethereum-based instance (i.e. based on Ethereum's technical implementation; not the public blockchain). The aim of a private blockchain is to restrict it so that only users or nodes within the same network can access the blockchain. For the purpose of this dissertation, all tests are performed in a local environment as they require no third-party access to the blockchain to conduct the proposed evaluation. Furthermore, rights for researchers and patients are handled within smart contracts. In real case, an alternative would be to make the blockchain permissioned, which allows the blockchain to be publicly visible while restricting access to a list of authorised users and nodes based on their public key.

Consensus algorithm

The blockchain is configured using the IBFT 2.0 consensus algorithm, which implements Proof-of-Authority (PoA). Conversely to PoW, PoA algorithms define nodes that have the capability to participate to the creation of new blocks and validating transactions. This is particularly power- and time-efficient in the context of private blockchains which do not require mining and calculation power. IBFT (Istanbul Bizantin Fault Tolerance) is one of the algorithms that implements the concept of PoA with the advantages of being highly fault-tolerant (e.g. in case of a hardware failure or the failure of a node of the blockchain) and efficient in terms of time between blocks that are created [9].

Gas usage

Another configuration choice is to make the blockchain gas-free. Blockchains such as Ethereum have the ability to add a fee to each transaction processed on the blockchain

[9] Scaling Consensus for Enterprise: Explaining the IBFT Algorithm, https://consensys.net/blog/enterprise-blockchain/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm/

depending on the required computational power. This fee is paid in gas, a "unit that measures the amount of computational effort required" [10]. In the context of a private blockchain in which nodes are handled, for instance, by research organisations, there is no requirement for establishing a fee on transactions. However, gas consumption could be implemented in the future depending on the context of application of the BaaS. In practice, in a gas-free blockchain, gas is still associated to every transaction, but the value of the gas is set to zero so that no fee is applied to any transactions.

Nodes in the network

A blockchain is composed of many nodes which purpose are to create new blocks and validate transactions that are stored in these blocks in a decentralised manner. For the IBFT 2.0 algorithm to be completely Byzantine fault tolerant, it is required to run at least four nodes within the blockchain. As described by Hyperledger Besu, a Byzantine-fault-tolerant blockchain ensures that a consensus can be reached "despite nodes failing or propagating incorrect information to peers" [11].

Blockchain accounts

In addition to the nodes in the network, three default accounts have been created in the blockchain. These accounts refer to users and are composed of a key pair (public and private keys); public key from which derives an address [12]. Therefore, each user of the blockchain can be identified by their address and their transactions can be signed using their private key. For this project, three internal user accounts have been allocated: one for a researcher and two for patients. These 'internal' accounts are created along with the blockchain and specified in the genesis configuration file. External accounts can also be created using third-party account providers such as MetaMask [13]. As the blockchain is private, any external account which has access to the private network is allowed to access the blockchain. For comparison, in a permissioned network, each account should be listed in an authorisation file (i.e. whitelist) to access the blockchain.

Once the blockchain has been properly configured, each node can be executed in a terminal as shown in Figure 4.5. In its commande line interface, Hyperledger Besu displays information about each node added to the blockchain with the following information:

- Whether the block has been 'produced' (i.e. created by this node) or 'imported' (i.e. created by another node in the network)

[10] Gas and fees - Ethereum, https://ethereum.org/en/developers/docs/gas/

[11] IBFT 2.0 - Hyperledger Besu, https://besu.hyperledger.org/en/stable/HowTo/Configure/Consensus-Protocols/IBFT/

[12] Technical background of version 1 Bitcoin addresses, https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses

[13] MetaMask, https://metamask.io/

- The block's number following the '#' symbol
- The number of transactions (tx) in that block
- The number of pending transactions
- The gas used (i.e. fee) in that block
- The block's address (*0x...*)

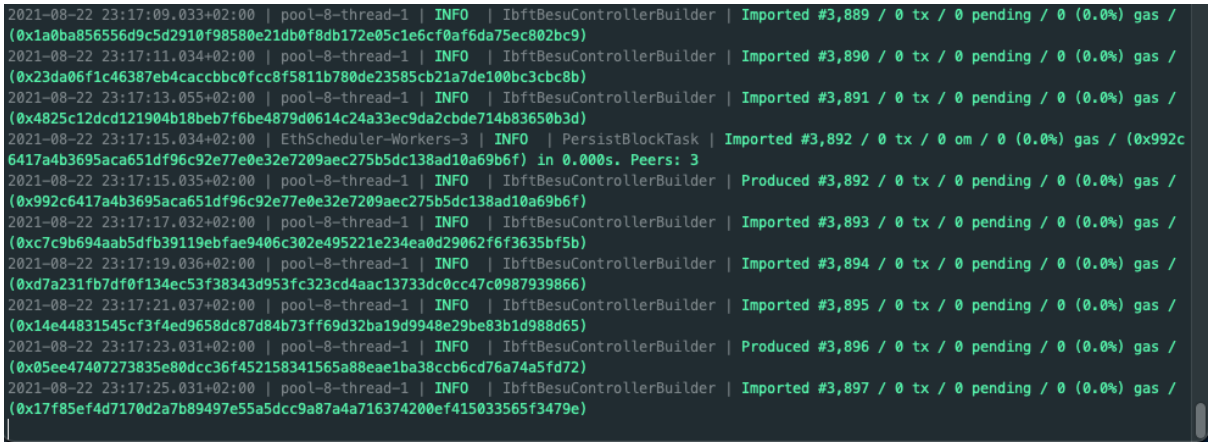

Figure 4.5: Execution of a blockchain's node in a terminal

4.5.2 Smart contract

Development

The management of patients consent and clinical trials is handled by a single smart contract developed in Solidity and named *ConsentsManager*. The smart contract implements two structures [14] which respectively define the models of a clinical trial and a consent. A list of clinical trials is maintained within the smart contract as well as a list of patients (identified by their address) and their respective consents.

Methods defined within the smart contract allow to create a clinical trial, to request a consent, to update a clinical trial and to accept, reject or revoke a consent on the patient's end. A list of researchers addresses is also maintained within the contract to determine whether a user is authorised to perform an action or not, for instance only researchers can create clinical trials or request consents to patients.

The *grandConsent* method shown in Figure 4.6 gives examples of Solidity's syntax and some of its features (see Appendix A1.4.1 for complete source code). For instance, the *msg* variable refers to the message that is being sent on the blockchain and which is calling this method; *msg.sender* refers to the address of the sender of the message. The *require*

[14] Type Structs - Solidity, https://docs.soliditylang.org/en/v0.8.7/types.html#structs

function is essential in Solidity to require conditions to be true. If a requirement is not met, the second parameter specifies the error message to return and the execution of the method is stopped.

```
function grantConsent(uint _consentId) public {
    PatientConsents storage patientConsents = patientsConsents[msg.sender];
    require(_consentId >= 1 && _consentId <= patientConsents.count, "Invalid consent ID");
    require(patientConsents.consents[_consentId].status == ConsentStatus.REQUESTED, "Consent
    not in 'REQUESTED' status");
    patientConsents.consents[_consentId].status = ConsentStatus.GRANTED;
}
```

Figure 4.6: Example of the *grantConsent* method in Solidity

Clarifications on the notion of 'immutable' blockchain

Having defined the blockchain as an 'append-only' and 'immutable' ledger, it could be confusing to see within the last line of the method that the status of an existing patient's consent is edited. In fact, this behaviour is normal as the entity which is changed is data stored in the smart contract. What remains immutable and append-only are calls that are made to reach this state (i.e. transactions). For instance, if a consent is changed from status 'REQUESTED' to 'GRANTED', the blockchain allows to browse the history of transactions to find the one that originated the status change, as well as who initiated the request (i.e. same as *msg.sender*) and at what time.

Deployment

When the smart contract has been developed, the framework Truffle handles its compilation and deployment in the blockchain. Truffle's migration script generates the ABI of the smart contract as well as its binary. A transaction is then performed on behalf of a user account configured in Truffle. Once the contract is deployed in the blockchain, the transaction receipt is given by Truffle as demonstrated in Figure 4.7.

Amongst all the information contained in the transaction receipt, the 'contract address' is particularly important as it is the address that will be callable in order to interact with the smart contract. In a blockchain, a smart contract has its own address and acts similarly as a standard user account, i.e. by sending transactions.

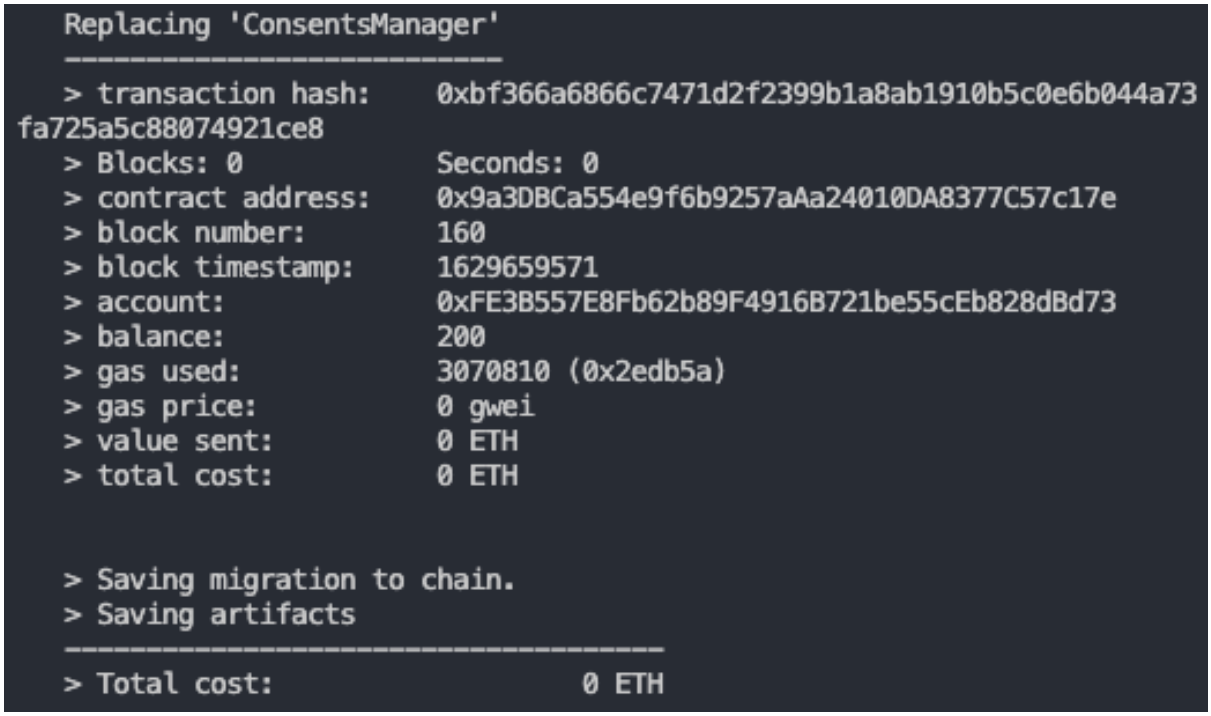

Figure 4.7: Truffle transaction receipt after smart contract deployment

4.5.3 Service layer implementation

The service layer of the BaaS relies essentially on Node.js, Truffle-contract and web3.js.

API Design

The API is the entry point to the underlying blockchain. Therefore, a robust API needs to be proposed to interact with the BaaS in an efficient manner. For that reason, the design of the API follows the principles of REST API development (8). A key concept of REST APIs is the usage of actions based on types of HTTP requests (i.e. HTTP verbs) which define CRUD operations (Create / Read / Update / Delete) as specified in Table 4.6.

HTTP Verb	Action
GET	Access a resource in read-only mode
POST	Create a new resource in the server
PUT	Update an existing resource in the server
DELETE	Delete an existing resource in the server

Table 4.6: Main HTTP verbs for REST APIs, modified from (8)

Another key concept of REST APIs is the usage of 'resource identifier' in endpoints (i.e. Unique Resource Identifier, URI). For instance, the following API requests could be defined

to interact with clinical trials within the blockchain (and the same reasoning applies to consents):

- POST /clinical-trials – Create a new clinical trial in the blockchain
- PUT /clinical-trials/:id – Update the clinical trial having the ID given in the URI

Finally, the API will use JSON as a data exchange format as (i) it is the most popular data exchange format for APIs with XML and (ii) it is a default choice for REST APIs, especially the ones developed in Node.js as JSON can be interpreted in JavaScript without any parsing – unlike XML –, which makes JSON more efficient.

HTTP Server

The HTTP server which allows to query the API is created in Node.js using the framework Express. Self-proclaimed as a minimal framework for web applications, Express provides a simple library of methods to route HTTP queries. For instance, the code snippet shown on Figure 4.8 handles any HTTP POST request on /clinical-trials, providing the request and response objects (*req* and *res*), respectively containing information about the request (e.g. query, parameters, body) and the response (i.e. what data to return once processing has finished).

```
app.post('/clinical-trials', async (req, res) => {
    // Processing...
});
```

Figure 4.8: Example of defining a POST request endpoint with Node.js and Express

Interaction with the blockchain

Hyperledger Besu's nodes can allow JSON-RPC HTTP API calls to interact with the blockchain. This allows clients such as web3.js to interact with the blockchain through its nodes. As Truffle is used to deploy smart contracts in the private blockchain, it is possible to rely on Trufle's configuration and the usage of the Truffle-contract package which eases the interaction with smart contracts. Indeed, methods of the smart contracts are called using classic transactions on the blockchain. However, the data of these transactions needs to exactly match the ABI of the smart contract (i.e. the specification of the smart contract). Truffle-contract acts as a bridge between web3.js and the blockchain by performing additional checks and data-binding.

4.6 Conclusion

Having defined all use-case scenarios, functional and non-functional requirements, technical choices and architecture of the BaaS, this chapter has provided in-depth explanations regarding the implementation and configuration of the BaaS while following the aforedescribed requirements.

That resulted in the implementation of a BaaS capable of handling clinical trials and their related consents. To provide client applications with a simple interface, the BaaS offers an HTTP API which is based on REST principles to interact with resources (e.g. create or update a clinical trial, request a consent, accept/reject a consent request). The BaaS is divided in two layers: a blockchain layer which relies on Hyperledger Besu and a service layer (i.e. API), developed in Node.js, which abstracts the underlying blockchain.

The next chapter discusses the evaluation and limitations of this implementation.

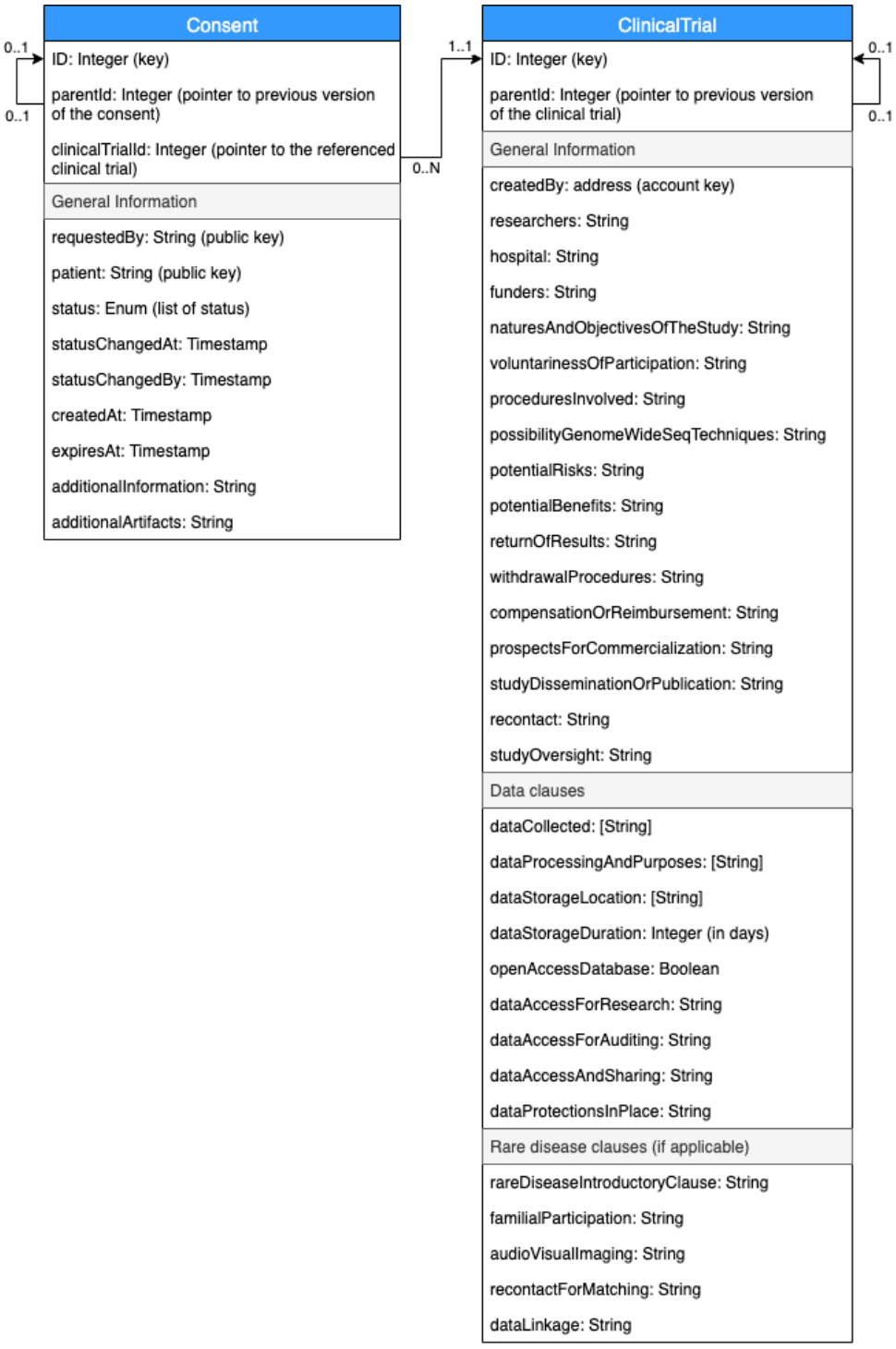

Figure 4.4: Data model of the BaaS

5 Evaluation and limitations

This chapter is dedicated to the evaluation of the BaaS implementation. The first section describes results that have been obtained. The second section details testing that have been performed on the BaaS. Thereafter, requirements fulfilment is discussed. In the fourth section, in-depth explanations are given about the limitations that have been raised. Finally, the last section draws conclusions of the chapter.

5.1 Results

A complete Blockchain as a Service (BaaS) system for patients consent management in clinical trials has resulted from the established requirements and the iterative development process. It is possible to interact with the BaaS through HTTP requests which design is based on REST APIs. The service layer creates a complete abstraction of the underlying blockchain (i.e. Hyperledger Besu). Furthermore, the implementation has been used to evaluate compliance with regulations, specifically the GDPR, through the application of competency questions borrowed from the domain of linked data in which this evaluation method is commonly used.

5.2 Testing

Functional testing of the BaaS has been conducted from the perspective of the API as well as the blockchain itself. The following subsections respectively describe API testing and blockchain testing.

5.2.1 API testing

For conducting functional testing of the API, all its endpoints have been implemented in Postman [1], a complete platform for API testing. Figure 5.1 gives an example of Postman's interface with the testing of the API endpoint for requesting patient's consent. Four zones are highlighted with numbers on the figure:

[1] https://www.postman.com/

- Number 1 designates the selector for the type of HTTP request, i.e. GET, POST, PUT, DELETE.
- Number 2 is the text input for the endpoint to use.
- Number 3 shows the request body, which contains data sent in the HTTP request. In the 'Request Consent' HTTP request, data regarding the consent should be provided including patients address (i.e. their identifier on the blockchain), the ID of the clinical trial the consent refers to, the expiry timestamp of the consent, the parent ID of the consent (if applicable, i.e. if the requested consent is a new version of an existing one) and any additional information specific to this consent.
- Number 4 shows the response body, i.e. all data which is returned after the request has been performed. For POST and PUT requests, which imply registering data in the blockchain, a transaction receipt is usually returned. It contains information such as the transaction hash and the block number in which the transaction has been registered.

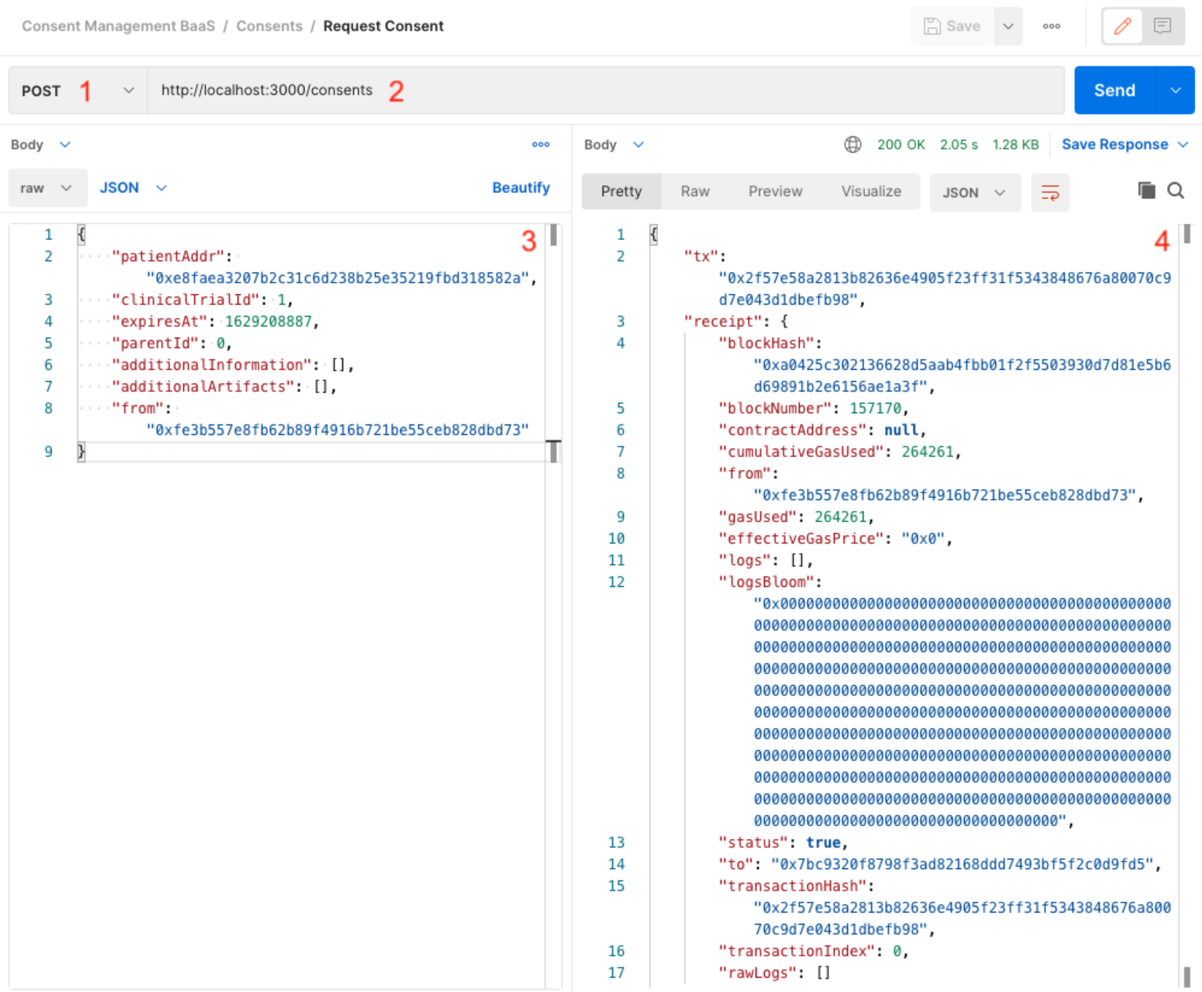

Figure 5.1: BaaS API testing with Postman

5.2.2 Blockchain testing

The blockchain has essentially been tested to verify that transactions are properly registered and that data contained within transactions can be easily retrieved, as it is crucial in ensuring transparency claimed by blockchain technologies. It also plays an important role in data integrity and non-repudiation as no transaction can be deleted from the blockchain (i.e. an immutable and append-only registry).

A trivial approach to determine if a transaction is registered is to verify the logs provided by Hyperledger Besu. A block which contains any transactions will contain a value greater than zero before 'tx', as highlighted in Figure 5.2 where block #11,108 contains one transaction.

Figure 5.2: Block containing a transaction shown in Hyperledger Besu's logs

In addition to that, blockchain explorers can be used to further explore any block, transactions or addresses (i.e. user account, smart contract) that is registered in the blockchain. For instance, Alethio Ethereum Lite Explorer [2] is an open source blockchain explorer for Ethereum blockchains. Once installed and configured locally, it provides an interface capable of searching information within the blockchain. For instance, Figure 5.3 shows the results of a research based on a transaction hash. From top to bottom, it displays:

- The transaction hash (TXH)
- The block in which the transaction has been registered and its timestamp
- The position of the transaction in the block, as a block can contain several transactions (starting from 0)
- Next to the position, the *nonce* refers to the number of transactions sent by the same address (e.g. user)
- The addresses of both sender and receiver of the transaction (e.g. a user or a smart contract)
- Information about gas usage and fees, in this example gas price is 0 as the blockchain is configured gas-free

[2] https://github.com/Alethio/ethereum-lite-explorer

- Input data in hexadecimal format (e.g. the arguments given to a smart contract's method)

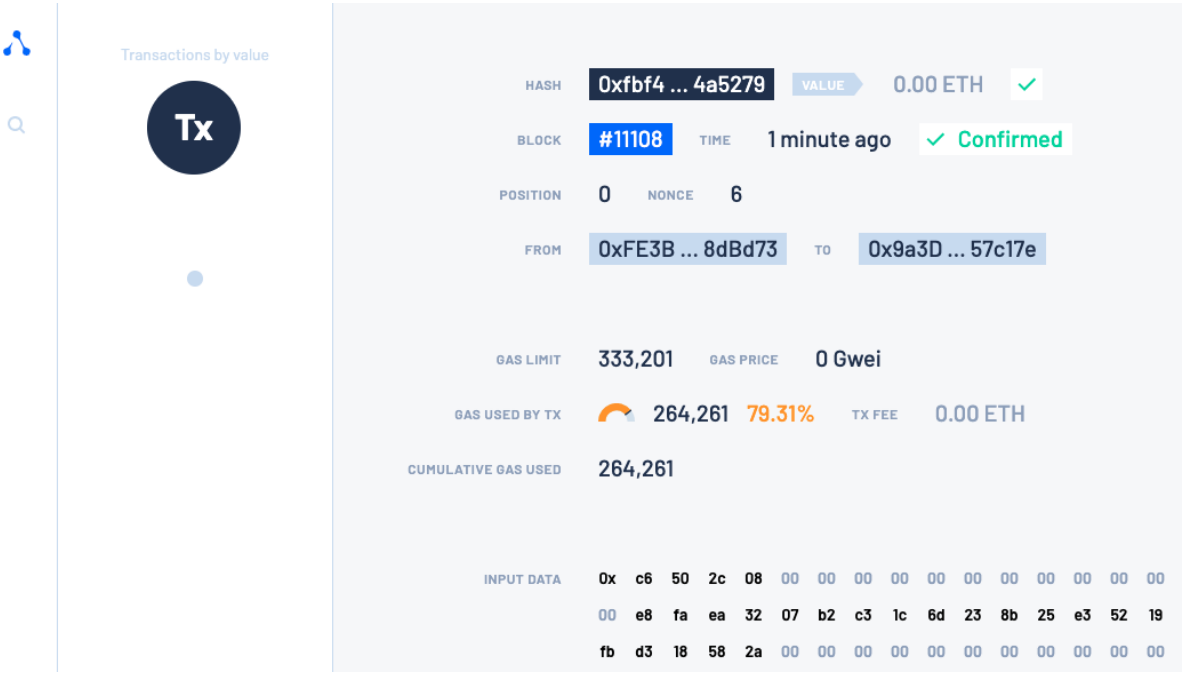

Figure 5.3: Transaction exploration with Alethio Ethereum Lite Explorer

Finally, the usage of web3.js *getTransaction* method [3] combined with an ABI decoder [4] allows to retrieve a transaction in JavaScript and to match its input data with the parameters of a smart contract's method (see Appendix A1.4.2 for source code). Figure 5.4 shows an example of the application of this process to a consent request transaction from which all method's parameters have been retrieved.

```
{
  name: 'requestConsent',
  params: [
    {
      name: '_patientAddr',
      value: '0xe8faea3207b2c31c6d238b25e35219fbd318582a',
      type: 'address'
    },
    { name: '_clinicalTrialId', value: '1', type: 'uint256' },
    { name: '_parentId', value: '0', type: 'uint256' },
    { name: '_expiresAt', value: '1629208887', type: 'uint256' },
    { name: '_additionalInformation', value: '', type: 'string' },
    { name: '_additionalArtifacts', value: '', type: 'string' }
  ]
}
```

Figure 5.4: Transaction input retrieved in details using web3.js and abi-decoder in JavaScript

[3] https://web3js.readthedocs.io/en/v1.2.2/web3-eth.html#id59
[4] https://github.com/ConsenSys/abi-decoder

5.3 Requirements fulfilment

The iterative development process and the testing of the BaaS demonstrate a functional system which fulfils all requirements. The BaaS implements both generic and rare disease consent clauses established by the MCC Task Force. Furthermore, the implementation complies with both functional and non-functional requirements as detailed in tables 5.1 and 5.2.

Functional requirement	**Requirement fulfilment**
F1. API interface to create and update clinical trials	Endpoints: POST or PUT /clinical-trials
F2. API interface to create consent requests	Endpoint: POST /consents
F3. API interface to accept or reject consent requests	Endpoint: POST /consents/:patient/:consentId/grant or POST /consents/:patient/:consentId/revoke
F4. API interface to read clinical trials and patients consents	Endpoints: GET /clinical-trials and GET /consents/:from (where :from is the address of the patient)
F5. Abstraction of the Blockchain layer when interacting with the API	Development of a service layer in Node.js which abstracts the underlying blockchain
F6. Segmentation of rights for researchers and patients	List of researchers addresses stored within the ConsentsManager smart contract

Table 5.1: Functional requirements fulfilment

Non-functional requirement	**Requirement fulfilment**
NF1. Interoperability	Design of an API which can be used by any third-party system
NF2. Privacy and security	Deployment of a private blockchain (i.e. not publicly accessible) and which supports private transactions
NF3. Integrity and accountability	Intrinsic to blockchain, especially due to its immutable and append-only properties
NF4. Ease of usage and deployment	API based on REST principles which facilitates its usage; possible deployment of new nodes within the blockchain network

Table 5.2: Non-functional requirements fulfilment

Finally, the BaaS fully complies with the GDPR according to the competency questions that have been issued from the GConsent ontology (3). Table 5.3 details the compliance which implies clauses defined by the MCC Task Force as well as fields that have been added as part of the BaaS data model to ensure its compliance.

Competency question	Question fulfilment
C1 Who is the consent about?	patient field (patient address)
C2 What type of Personal Data are associated with the Consent?	Data clauses: dataCollected, dataProcessingAndPurposes, dataStorageLocation, dataStorageDuration, dataAccessForResearch, dataAccessForAuditing, dataAccessAndSharing, dataProtectionsInPlace, openAccessDatabase
C3 What type of Purposes are associated with the Consent?	dataProcessingAndPurposes clause
C4 What type of Processing are associated with the Consent?	dataProcessingAndPurposes clause
C5 What is the Status of Consent?	status field
C6 Is the current status valid for processing?	If status is 'GRANTED'
C7 Who is the consent given to?	requestedBy field (researcher address)
T1 What is the location associated with consent?	hospital clause (which refers to the hospital or institution and, implicitly, its location) and dataStorageLocation clauses
T2 What is the medium associated with consent?	Blockchain is systematically associated with a consent granted through the BaaS
T3 What is the timestamp associated with the consent?	createdAt field
T4 What is the expiry of the consent?	expiresAt field
T5 How was the consent acquired/changed/created/invalidated?	naturesAndObjectives clause ("Specifies the activity that was responsible for the consent instance." [5])
T6 What artefacts were shown when consent was acquired/changed/created/invalidated?	All clauses of the referenced clinical trial; additionalInformation and additionalArtifacts clauses (if applicable)
D1 Is the purpose or processing associated with a third party?	dataAccessAndSharing clause
D2 What is the role played by the third party in the purpose or processing?	dataAccessAndSharing clause (i.e. access by third-party should be motivated by specific purpose or processing within the same clause)

Table 5.3: Competency questions fulfilment

While evidence demonstrate the feasibility and compliance with the GDPR of managing patients consent for clinical trials using a BaaS, this technique raises limitations which are considered in the next section.

[5]https://openscience.adaptcentre.ie/ontologies/GConsent/docs/ontology

5.4 Limitations

This section details the limitations of the BaaS. First, a focus is made on the implementation of the data model in Solidity. Thereafter, Solidity's language limitations are discussed. Finally, the implementation of the API by third-parties is considered.

5.4.1 Data model in Solidity

Many data models could be suitable to store patients consents within the blockchain. For instance, consents could be merged with clinical trials' data model which has been proposed in this implementation. However, Solidity and smart contracts are not meant to store data as a database does (e.g. SQL, NoSQL). Rather, it relies on arrays and mappings [6] which imply limitations in terms of data model complexity (e.g. no native notion of tables, indexes and relation between data models).

Furthermore, it is not evident that a 'consent history' (i.e. the many versions of a patient's consent) should be retrieved (i) by iterating over blockchain transactions (54) or (ii) by storing the evolution of the consent within the smart contract storage (which requires more computational power). In this dissertation, (ii) is implemented by providing each consent and clinical trial a 'parent ID' which can refer to a previous version of the consent. As it is out of the scope of this dissertation, further investigation would be required to determine what approach is the most suitable and optimised from the blockchain's perspective.

5.4.2 Solidity language limitations

While Solidity's syntax is developer-friendly, it still presents many limitations, for instance regarding its stack, data types and error handling.

One of the main limitations that have been faced during the implementation of the BaaS is the 'Stack too deep' error. Alongside 'error handling' and 'mappings', it is one of the most criticised aspect of Solidity by developers according to the Solidity Summit survey of 2020 [7], results of which are shown in Figure 5.5.

[6] https://docs.soliditylang.org/en/latest/style-guide.html#mappings

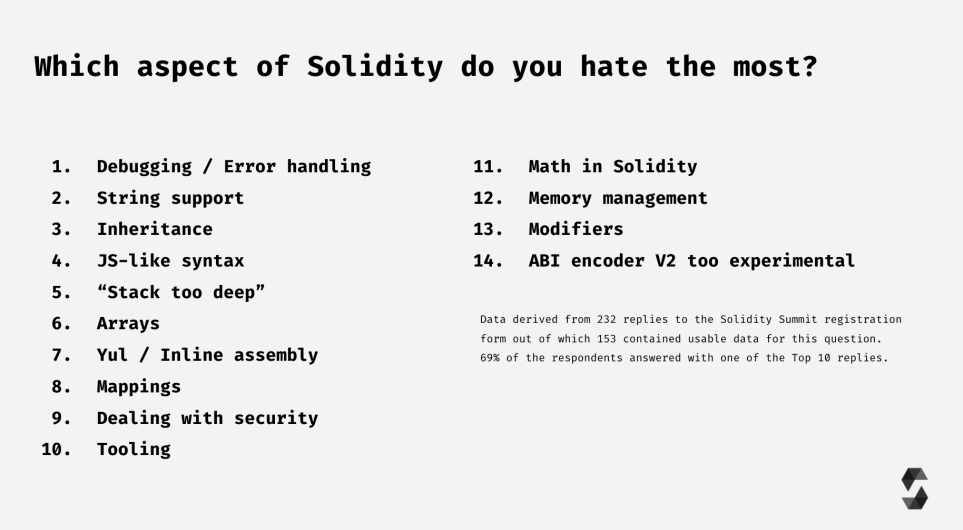

Figure 5.5: Most 'hated' aspects of Solidity according to their 2020 developer survey

For instance, the 'Stack too deep' error can occur when too many local variables are used within a block (e.g. a function) or when a structure contains too many variables. When parameters and/or variables used within the scope of a block exceed approximately a dozen, the error is raised. This limitation has been particularly disturbing as consents require a large number of clauses. To avoid it, it has been required to 'compress' the storage of clinical trials' parameters within the smart contract. In other terms, some clauses described in the data model have been merged in the clinical trial's structure to be added to the smart contract. This is why a single *generalInformation* field is responsible for containing all general consent clauses (i.e. 15 clauses) within the clinical trial structure.

Therefore, the API (i.e. the service layer) translates (i.e. encodes) JSON data to arrays of strings that Solidity handles and stores as single variables. Not only does the API facilitate the implementation with the blockchain, it also resolves limitations imposed by Solidity. However, it is worth noting that the API also has its limitations as discussed below.

5.4.3 Third-party implementation of the API

While the BaaS meets all criteria for compliance with the GDPR regulation, the service layer relies on thorough implementation made by third-parties (i.e. CTMS, user-interfaces for patients). For instance, there is no certainty that a user-interface will display all information about the consent that are stored within the smart contract prior to sending the patient acceptance request to register it on the blockchain through the BaaS API. While offering a centralised user interface as proposed by Albanese et al. (49) could address this issue, it would largely hamper interoperability. An interesting trade-off could be to handle, within the service layer of the BaaS, a strict flow for presenting information to the patients and

[7] https://twitter.com/solidity_lang/status/1258432541226401793/photo/1

researchers prior to accepting requests registered within the blockchain (e.g. through an embedded web service).

It is worth noting that the aforementioned limitations focus on blocking points that have been encountered throughout the implementation of the BaaS. As this dissertation addresses a wicked problem, the proposed implementation could not address all the scope of the related disciplines and artifacts. Therefore, further limitations which are due to the limited scope of this dissertation are discussed in section 6.4 from the perspective of future work.

5.5 Conclusion

This chapter has provided results obtained from the implementation of the BaaS. It has described the testing processes that have been used, the requirements fulfilment and the limitations of the implementation.

The BaaS implementation demonstrates feasibility as well as a complete fulfilment of the requirements that have been established. Results show that the BaaS can implement all required consent clauses and that it is compliant with the GDPR, according to the defined set of competency questions. These results are encouraging, although many limitations have been raised (most of which are due to Solidity).

The next chapter concludes this dissertation and suggests future work.

6 Conclusion and future work

This chapter concludes the dissertation project. First, the main outcomes of this dissertation are summarised. Then, the fulfilment of research objectives is detailed. In the third section, faced challenges are discussed. Finally, future work is suggested.

6.1 Main outcomes

As stated by the research question of this dissertation in section 1.2, the aim of this work is to conduct an investigation of the usage of blockchain to manage patients consent for clinical trials. The iterative background research, identification and refinement of the wicked problem, design of evaluation criteria and implementation of the project led to an operational Blockchain as a Service (BaaS) for patients consent management for clinical trials.

The main outcomes of this project are (i) the prototype of a BaaS which has been made accessible on GitHub (see Appendix A1.1) and (ii) the full coverage of GDPR-related competency questions by the BaaS which demonstrates blockchain's capabilities to comply with regulations within that specific context.

Furthermore, the proposed solution especially focuses on consent management with a strong attention to interoperability to support future explorations and artifacts.

6.2 Fulfilment of research objectives

This section details how research objectives defined in section 1.3 have been fulfilled.

Objectives (a) and (b) have been satisfied through the iterative background research which helped to determine consent models, forms and applied regulations. This led to a particular focus on the GDPR which is implemented in the vast majority of EU countries and serves as a basis for other country-wide regulations.

Objective (c) has been fulfilled through the development of the BaaS prototype which has been evaluated against a set of competency questions to ensure its compliance with the

GDPR, hence fulfilling (d). Finally, (e) has been confirmed with the testing of the BaaS both at the level of the service layer and the underlying blockchain technology.

6.3 Challenges faced

During this dissertation project, a number of challenges have been faced from different perspectives. This section briefly describes some of these challenges. First, challenges raised by the multidisciplinary approach are mentioned, followed by organisational challenges. Then, technical challenges are discussed.

6.3.1 Multidisciplinary approach

One of the main challenge that has been faced throughout this dissertation project is the constant multidisciplinary approach which is required to address the problem of patients consent management. This project implies knowledge in law (i.e. regulations), strong focus on methodology (i.e. design-science and especially design as a search process, wicked problems, iterative approach) and computer science (i.e. particularly BCT).

This approach sets the context for a particular challenge – which could be perceived as a limitation – in the process of design-science: the implementation of an artifact in a domain implying many disciplines can barely be conducted by a single person. Therefore, design-science fosters working with other people who specialise in related disciplines. For this dissertation, the inputs from members of the MND Team at ADAPT Centre (TCD) helped understand some facets of the problem such as the global context of clinical trials for MND and processes implied for collecting patients consent.

6.3.2 Organisation

The main organisational challenge derives from the iterative approach for which no stopping rule exists. It has been difficult to know where to stop in the iterative background research and to exactly determine the problem to address. The same applies to the scope of the artifact designed. From a personal perspective, this approach is completely new and does not relate to any previous research project.

6.3.3 Technical challenges

Many technical challenges have been faced through the implementation of the BaaS. Most of these challenges come from the fact that Hyperledger Besu is a very recent blockchain project which benefits from less documentation, community and support than others such as Hyperledger Fabric. For instance, properly configuring the local blockchain instance led to

exchanges with Hyperledger Besu's community chat [1]. Interacting with the blockchain through the JavaScript (Node.js) client for the service layer of the BaaS has also been more complex than expected; again rather in terms of configuration and 'good practices' than algorithmically.

The fact that this project is a first hands-on exploration of blockchain technologies has also made the implementation more difficult. For instance, most of the technical limitations implied by Solidity are not common to other languages and therefore often unexpected.

6.4 Future work

While implying several domains of research (i.e. ICT, regulations, research methods), this work paves the way for future investigation by raising multidisciplinary concerns which are described in this section. First, the specific scenario of voluntary patients' consent is discussed. Then, compliance with other regulations is mentioned. In the third subsection, data vocabulary heterogeneity is discussed, followed by clauses data validity. Thereafter, further research that would be required on authentication is mentioned. Finally, prospective performance evaluations and optimisations are proposed.

6.4.1 Voluntary patients' consent

The GDPR regulation requires a low level of granularity in terms of required data as well as processing and purposes associated with that data. On the one hand, these requirements help to preserve patients privacy by letting them decide who to share their data with and for what reason. On the other hand, it could also hinder voluntary initiatives taken by patients for sharing their data. In case a patient is willing to share their data globally, for any purposes and without any restrictions, it could be interesting to determine to what extent this voluntary approach remains compliant with the GDPR (or related regulations) and how it should be implemented.

6.4.2 Compliance with other regulations

This dissertation focused on EU regulations and particularly the GDPR. The proposed approach for evaluating compliance based on competency questions could be tested against other regulations, for instance the FDA and HIPAA which are often referred to in the USA.

[1] Hyperledger Chat, https://chat.hyperledger.org/channel/besu

6.4.3 Data vocabulary heterogeneity

The clauses that have been implemented rely on paper-based consent forms and usually contain paragraphs of text. Investigations could be conducted on vocabulary, as a heterogeneous vocabulary could highly improve interoperability between the BaaS and other artifacts or systems (e.g. CTMS). For instance, *dataProcessingAndPurposes* clause can contain an array of strings such as "The data will be used within AI processing system". Standardising the possible values could help automating exchange of information with the BaaS, e.g. by replacing the previous example with values such as "AI_PROCESSING_IMPLIED".

6.4.4 Clauses data validity

The implemented BaaS can ensure that the input formats are respected (e.g. strings or arrays are given). However, it cannot determine whether or not the content of a specific clause is clear enough for the consent to be considered 'informed' from the patient's perspective. Along with a more homogeneous vocabulary for writing clauses within the blockchain, this concern could also motivate further research on Natural Language Processing (NLP) techniques for compliance verification.

6.4.5 Authentication

The blockchain handles authentication and authorisations based on a key pair that each user possesses. However, it would be possible that either the API or third-party software acts as an authentication service to communicate with the blockchain on behalf of the user logged in. For instance, a third-party application could require the patient to log in and, then, communicate with the BaaS using the patient's key pair. This concern leads to another one regarding the delegate usage of the blockchain, i.e. to authorise a trusted person to act on the patient's behalf in case they are in incapacity of taking a decision. These concerns would require further investigation to determine the best approach to properly authenticate both patients and researchers.

6.4.6 Performance evaluations and optimisations

The prototype implementation demonstrates encouraging results regarding the feasibility of using blockchain for managing patients consent. However, tests have been performed in a local network with only four nodes connected to form the blockchain. Further evaluation of the efficiency and throughput of the solution in a real environment would be required, as well as optimisations within the smart contract implementation (e.g. opting for consent history through transactions as discussed in section 5.4.1).

Bibliography

[1] Ibrar Yaqoob, Khaled Salah, Raja Jayaraman, and Yousof Al-Hammadi. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. *Neural Computing and Applications*, January 2021. ISSN 0941-0643, 1433-3058. doi: 10.1007/s00521-020-05519-w. URL http://link.springer.com/10.1007/s00521-020-05519-w.

[2] Kenneth F Schulz, Douglas G Altman, and David Moher. CONSORT 2010 Statement: updated guidelines for reporting parallel group randomised trials. page 8, 2010.

[3] Harshvardhan J. Pandit, Christophe Debruyne, Declan O'Sullivan, and Dave Lewis. GConsent - A Consent Ontology Based on the GDPR. In Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J.G. Gray, Vanessa Lopez, Armin Haller, and Karl Hammar, editors, *The Semantic Web*, volume 11503, pages 270–282. Springer International Publishing, Cham, 2019. ISBN 978-3-030-21347-3 978-3-030-21348-0. doi: 10.1007/978-3-030-21348-0_18. URL http://link.springer.com/10.1007/978-3-030-21348-0_18. Series Title: Lecture Notes in Computer Science.

[4] Karl Wust and Arthur Gervais. Do you Need a Blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, Zug, June 2018. IEEE. ISBN 978-1-5386-7204-4. doi: 10.1109/CVCBT.2018.00011. URL https://ieeexplore.ieee.org/document/8525392/.

[5] Richard L. Baskerville, and Mala Kaul, Veda C. Storey, and and and. Genres of Inquiry in Design-Science Research: Justification and Evaluation of Knowledge Production. *MIS Quarterly*, 39(3):541–564, March 2015. doi: 10.25300/misq/2015/39.3.02. URL https://doi.org/10.25300/misq/2015/39.3.02. Publisher: MIS Quarterly.

[6] International review of consent models for the collection, use and sharing of health information, February 2020. URL https://www.hiqa.ie/sites/default/files/2020-02/International-Review%E2%80%93consent-models-for-health-information.pdf.

[7] on behalf of the IRDiRC-GA4GH Model Consent Clauses Task Force, Minh Thu Nguyen, Jack Goldblatt, Rosario Isasi, Marlene Jagut, Anneliene Hechtelt Jonker, Petra Kaufmann, Laetitia Ouillade, Fruszina Molnar-Gabor, Mahsa Shabani, Eric Sid, Anne Marie Tassé, Durhane Wong-Rieger, and Bartha Maria Knoppers. Model consent clauses for rare disease research. *BMC Medical Ethics*, 20(1):55, December 2019. ISSN 1472-6939. doi: 10.1186/s12910-019-0390-x. URL https://bmcmedethics.biomedcentral.com/articles/10.1186/s12910-019-0390-x.

[8] Pro REST API Development with Node.js - Fernando Doglio - Google Books, . URL https://books.google.fr/books?hl=en&lr=&id=kjUwCgAAQBAJ&oi=fnd&pg=PR7&dq=rest+api&ots=f249NvcKtd&sig=DiKhEhTvmxEI4UkZ_Yh9rPKqaXU&redir_esc=y#v=onepage&q=rest%20api&f=false.

[9] Arni Ariani, Allya P. Koesoema, and Soegijardjo Soegijoko. Innovative Healthcare Applications of ICT for Developing Countries. In Hassan Qudrat-Ullah and Peter Tsasis, editors, *Innovative Healthcare Systems for the 21st Century*, pages 15–70. Springer International Publishing, Cham, 2017. ISBN 978-3-319-55773-1 978-3-319-55774-8. doi: 10.1007/978-3-319-55774-8_2. URL http://link.springer.com/10.1007/978-3-319-55774-8_2. Series Title: Understanding Complex Systems.

[10] Harsh Kupwade Patil and Ravi Seshadri. Big Data Security and Privacy Issues in Healthcare. In *2014 IEEE International Congress on Big Data*, pages 762–765, Anchorage, AK, June 2014. IEEE. ISBN 978-1-4799-5057-7. doi: 10.1109/BigData.Congress.2014.112. URL https://ieeexplore.ieee.org/document/6906856/.

[11] Joshua Introne, Robert Laubacher, Gary Olson, and Thomas Malone. Solving Wicked Social Problems with Socio-computational Systems. *KI - Künstliche Intelligenz*, 27(1): 45–52, December 2012. doi: 10.1007/s13218-012-0231-2. URL https://doi.org/10.1007/s13218-012-0231-2. Publisher: Springer Science and Business Media LLC.

[12] WHO - Clinical Trials, 2021. URL https://www.who.int/health-topics/clinical-trials/. Publication Title: Who.int.

[13] Lawrence M Friedman, Curt D Furberg, David L DeMets, David M Reboussin, and Christopher B Granger. *Fundamentals of Clinical Trials*. Springer International Publishing, 2015.

[14] Robert C. Griggs, Mark Batshaw, Mary Dunkle, Rashmi Gopal-Srivastava, Edward Kaye, Jeffrey Krischer, Tan Nguyen, Kathleen Paulus, and Peter A. Merkel. Clinical

research for rare disease: Opportunities, challenges, and solutions. *Molecular Genetics and Metabolism*, 96(1):20–26, January 2009. ISSN 10967192. doi: 10.1016/j.ymgme.2008.10.003. URL https://linkinghub.elsevier.com/retrieve/pii/S1096719208002539.

[15] Committee on Health Research and the Privacy of Health Information: The HIPAA Privacy Rule, Board on Health Sciences Policy, Board on Health Care Services, and Institute of Medicine. *Beyond the HIPAA Privacy Rule: Enhancing Privacy, Improving Health Through Research*. National Academies Press, Washington, D.C., February 2009. ISBN 978-0-309-12499-7. doi: 10.17226/12458. URL http://www.nap.edu/catalog/12458. Pages: 12458.

[16] A guide to GDPR data privacy requirements, February 2019. URL https://gdpr.eu/data-privacy/. Library Catalog: gdpr.eu Section: GDPR Compliance.

[17] Medical privacy, May 2021. URL https://en.wikipedia.org/w/index.php?title=Medical_privacy&oldid=1024999530. Page Version ID: 1024999530.

[18] Ajit Appari and M. Eric Johnson. Information security and privacy in healthcare: current state of research. *International Journal of Internet and Enterprise Management*, 6(4):279, 2010. ISSN 1476-1300, 1741-5330. doi: 10.1504/IJIEM.2010.035624. URL http://www.inderscience.com/link.php?id=35624.

[19] E. Rescorla and B. Korver. Guidelines for Writing RFC Text on Security Considerations. URL https://tools.ietf.org/html/rfc3552#section-1.1.

[20] World Medical Association Declaration of Helsinki: Ethical Principles for Medical Research Involving Human Subjects. *JAMA*, 310(20):2191, November 2013. ISSN 0098-7484. doi: 10.1001/jama.2013.281053. URL http://jama.jamanetwork.com/article.aspx?doi=10.1001/jama.2013.281053.

[21] The Belmont Report. page 10, .

[22] Nuremberg Code. The Nuremberg Code. *Trials of war criminals before the Nuremberg military tribunals under control council law*, 10(1949):181–2, 1949.

[23] Sarah J. L. Edwards, Richard J. Lilford, Jim Thornton, and Jenny Hewison. Informed consent for clinical trials: in search of the "best" method. *Social Science & Medicine*, 47(11):1825–1840, 1998. ISSN 0277-9536. doi: https://doi.org/10.1016/S0277-9536(98)00235-4. URL https://www.sciencedirect.com/science/article/pii/S0277953698002354.

[24] GDPR.eu - Data Privacy, 2021. URL https://gdpr.eu/data-privacy/. Publication Title: GDPR.eu.

[25] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (Un)Informed Consent: Studying GDPR Consent Notices in the Field. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 973–990, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 978-1-4503-6747-9. doi: 10.1145/3319535.3354212. URL https://doi.org/10.1145/3319535.3354212. event-place: London, United Kingdom.

[26] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. October 2018. doi: 10.6028/nist.ir.8202. URL http://dx.doi.org/10.6028/NIST.IR.8202. Publisher: National Institute of Standards and Technology.

[27] Dirk A Zetzsche, Douglas W Arner, and Ross P Buckley. Decentralized Finance. *Journal of Financial Regulation*, 6(2):172–203, September 2020. ISSN 2053-4833, 2053-4841. doi: 10.1093/jfr/fjaa010. URL https://academic.oup.com/jfr/article/6/2/172/5913239.

[28] Marko Vukolić. Rethinking Permissioned Blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 3–7, Abu Dhabi United Arab Emirates, April 2017. ACM. ISBN 978-1-4503-4974-1. doi: 10.1145/3055518.3055526. URL https://dl.acm.org/doi/10.1145/3055518.3055526.

[29] Shubhani Aggarwal and Neeraj Kumar. Chapter Eleven - Cryptographic consensus mechanismsIntroduction to blockchain. In Shubhani Aggarwal, Neeraj Kumar, and Pethuru Raj, editors, *Advances in Computers*, volume 121 of *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, pages 211–226. Elsevier, January 2021. doi: 10.1016/bs.adcom.2020.08.011. URL https://www.sciencedirect.com/science/article/pii/S0065245820300668.

[30] L. M. Bach, B. Mihaljevic, and M. Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550, Opatija, May 2018. IEEE. ISBN 978-953-233-095-3. doi: 10.23919/MIPRO.2018.8400278. URL https://ieeexplore.ieee.org/document/8400278/.

[31] Roberto Saltini and David Hyland-Wood. IBFT 2.0: A Safe and Live Variation of the IBFT Blockchain Consensus Protocol for Eventually Synchronous Networks. *arXiv:1909.10194 [cs]*, September 2019. URL http://arxiv.org/abs/1909.10194. arXiv: 1909.10194.

[32] Lin William Cong and Zhiguo He. Blockchain Disruption and Smart Contracts. *The Review of Financial Studies*, 32(5):1754–1797, May 2019. ISSN 0893-9454, 1465-7368. doi: 10.1093/rfs/hhz007. URL https://academic.oup.com/rfs/article/32/5/1754/5427778.

[33] Hevner, March, Park, and Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75, 2004. doi: 10.2307/25148625. URL https://doi.org/10.2307/25148625. Publisher: JSTOR.

[34] Andrejs Skaburskis. The Origin of "Wicked Problems". *Planning Theory & Practice*, 9 (2):277–280, June 2008. ISSN 1464-9357, 1470-000X. doi: 10.1080/14649350802041654. URL http://www.tandfonline.com/doi/abs/10.1080/14649350802041654.

[35] Gloria F. Donnelly. The Transformation of Healthcare: A Wicked Problem. *Holistic Nursing Practice*, 20(5):215–216, September 2006. ISSN 0887-9311. doi: 10.1097/00004650-200609000-00001. URL http://journals.lww.com/00004650-200609000-00001.

[36] D. R. Schlegel and G. Ficheur. Secondary Use of Patient Data: Review of the Literature Published in 2016. *Yearbook of Medical Informatics*, 26(1):68–71, August 2017. ISSN 0943-4747. doi: 10.15265/IY-2017-032. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6250993/.

[37] Martin R. Cowie, Juuso I. Blomster, Lesley H. Curtis, Sylvie Duclaux, Ian Ford, Fleur Fritz, Samantha Goldman, Salim Janmohamed, Jörg Kreuzer, Mark Leenay, Alexander Michel, Seleen Ong, Jill P. Pell, Mary Ross Southworth, Wendy Gattis Stough, Martin Thoenes, Faiez Zannad, and Andrew Zalewski. Electronic health records to facilitate clinical research. *Clinical Research in Cardiology: Official Journal of the German Cardiac Society*, 106(1):1–9, January 2017. ISSN 1861-0692. doi: 10.1007/s00392-016-1025-6.

[38] Charlotte J. Haug. Whose Data Are They Anyway? Can a Patient Perspective Advance the Data-Sharing Debate? *New England Journal of Medicine*, 376(23):2203–2205, June 2017. ISSN 0028-4793, 1533-4406. doi: 10.1056/NEJMp1704485. URL http://www.nejm.org/doi/10.1056/NEJMp1704485.

[39] David Blumenthal and David Squires. Giving Patients Control of Their EHR Data. *Journal of General Internal Medicine*, 30(S1):42–43, January 2015. ISSN 0884-8734, 1525-1497. doi: 10.1007/s11606-014-3071-y. URL http://link.springer.com/10.1007/s11606-014-3071-y.

[40] Peter H. Schwartz, Kelly Caine, Sheri A. Alpert, Eric M. Meslin, Aaron E. Carroll, and William M. Tierney. Patient Preferences in Controlling Access to Their Electronic

Health Records: a Prospective Cohort Study in Primary Care. *Journal of General Internal Medicine*, 30(S1):25–30, January 2015. ISSN 0884-8734, 1525-1497. doi: 10.1007/s11606-014-3054-z. URL http://link.springer.com/10.1007/s11606-014-3054-z.

[41] Adrian Thorogood. International Data Sharing and Rare Disease: The Importance of Ethics and Patient Involvement. In Zhan He Wu, editor, *Rare Diseases*. IntechOpen, March 2020. ISBN 978-1-83880-023-9 978-1-83880-024-6. doi: 10.5772/intechopen.91237. URL https://www.intechopen.com/books/rare-diseases/international-data-sharing-and-rare-disease-the-importance-of-ethics-and-patient-involvement.

[42] Evan McCroary. Investigation into ICT support for clinical trial enrolment of patients with Motor Neuron Disease. page 79, April 2021.

[43] Christopher V. Cosgriff, Daniel K. Ebner, and Leo Anthony Celi. Data sharing in the era of COVID-19. *The Lancet Digital Health*, 2(5):e224, May 2020. ISSN 2589-7500. doi: 10.1016/S2589-7500(20)30082-0. URL https://www.thelancet.com/journals/landig/article/PIIS2589-7500(20)30082-0/abstract. Publisher: Elsevier.

[44] Junaid Shuja, Eisa Alanazi, Waleed Alasmary, and Abdulaziz Alashaikh. COVID-19 open source data sets: a comprehensive survey. *Applied Intelligence*, 51(3):1296–1325, March 2021. ISSN 0924-669X, 1573-7497. doi: 10.1007/s10489-020-01862-6. URL http://link.springer.com/10.1007/s10489-020-01862-6.

[45] edpb (European Data Protection Board). Guidelines 05/2020 on consent under Regulation 2016/679, May 2020. URL https://edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_202005_consent_en.pdf.

[46] Harshvardhan J. Pandit, Axel Polleres, Bert Bos, Rob Brennan, Bud Bruegger, Fajar J. Ekaputra, Javier D. Fernández, Roghaiyeh Gachpaz Hamed, Elmar Kiesling, Mark Lizar, Eva Schlehahn, Simon Steyskal, and Rigo Wenning. Creating a Vocabulary for Data Privacy: The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG). In Hervé Panetto, Christophe Debruyne, Martin Hepp, Dave Lewis, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems: OTM 2019 Conferences*, volume 11877, pages 714–730. Springer International Publishing, Cham, 2019. ISBN 978-3-030-33245-7 978-3-030-33246-4. doi: 10.1007/978-3-030-33246-4_44. URL http://link.springer.com/10.1007/978-3-030-33246-4_44. Series Title: Lecture Notes in Computer Science.

[47] Kaniz Fatema, Ensar Hadziselimovic, Harshvardhan Pandit, Christophe Debruyne, Dave Lewis, and Declan O'Sullivan. Compliance through Informed Consent: Semantic Based Consent Permission and Data Management Model. page 16.

[48] Harshvardhan J. Pandit, Declan O'Sullivan, and Dave Lewis. Test-Driven Approach Towards GDPR Compliance. In Maribel Acosta, Philippe Cudré-Mauroux, Maria Maleshkova, Tassilo Pellegrini, Harald Sack, and York Sure-Vetter, editors, *Semantic Systems. The Power of AI and Knowledge Graphs*, volume 11702, pages 19–33. Springer International Publishing, Cham, 2019. ISBN 978-3-030-33219-8 978-3-030-33220-4. doi: 10.1007/978-3-030-33220-4_2. URL http://link.springer.com/10.1007/978-3-030-33220-4_2. Series Title: Lecture Notes in Computer Science.

[49] Giuseppe Albanese, Jean-Paul Calbimonte, Michael Schumacher, and Davide Calvaresi. Dynamic consent management for clinical trials via private blockchain technology. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):4909–4926, November 2020. ISSN 1868-5137, 1868-5145. doi: 10.1007/s12652-020-01761-1. URL http://link.springer.com/10.1007/s12652-020-01761-1.

[50] Dan Brown. Review of Five popular Hyperledger DLTs- Fabric, Besu, Sawtooth, Iroha and Indy, February 2021. URL https://training.linuxfoundation.org/announcements/review-of-five-popular-hyperledger-dlts-fabric-besu-sawtooth-iroha-and-indy/. Library Catalog: training.linuxfoundation.org.

[51] Mira Shah, Chao Li, Ming Sheng, Yong Zhang, and Chunxiao Xing. Smarter Smart Contracts: Efficient Consent Management in Health Data Sharing. In Xin Wang, Rui Zhang, Young-Koo Lee, Le Sun, and Yang-Sae Moon, editors, *Web and Big Data*, volume 12318, pages 141–155. Springer International Publishing, Cham, 2020. ISBN 978-3-030-60289-5 978-3-030-60290-1. doi: 10.1007/978-3-030-60290-1_11. URL https://link.springer.com/10.1007/978-3-030-60290-1_11. Series Title: Lecture Notes in Computer Science.

[52] Christopher Millard. Blockchain and law: Incompatible codes? *Computer Law & Security Review*, 34(4):843–846, August 2018. ISSN 02673649. doi: 10.1016/j.clsr.2018.06.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0267364918302437.

[53] Vikram Dhillon, David Metcalf, and Max Hooper. The Hyperledger Project. In *Blockchain Enabled Applications*, pages 139–149. Apress, Berkeley, CA, 2017. ISBN 978-1-4842-3080-0 978-1-4842-3081-7. doi: 10.1007/978-1-4842-3081-7_10. URL http://link.springer.com/10.1007/978-1-4842-3081-7_10.

[54] Ahmed Afif Monrat, Olov Schelen, and Karl Andersson. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access*, 7: 117134–117151, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2936094. URL https://ieeexplore.ieee.org/document/8805074/.

A1 Appendix

The appendix includes relevant information related to the project. First, information are given about the GitHub repository and then about the API documentation. Thereafter, useful resources are mentioned, especially regarding the technical usage of Hyperledger Besu, programming in Solidity and Node.js. Finally, the source code of the project is provided.

A1.1 GitHub repository

The BaaS implementation is open-source and available in the following GitHub repository: https://github.com/borisflesch/baas-consent-management. A detailed 'README' file is provided with instructions to install and execute the BaaS in a local environment.

In addition to the GitHub repository, the most important parts of the source code are also available in appendix A1.4.

A1.2 API Documentation

A Postman [1] collection has been created to exhaustively describe the BaaS API. A Postman collection can be tested within the software or exported as a standalone documentation as shown in figure A1.1. An export of the collection is available at the root of the GitHub repository (*postman-collection.json*) and can be imported in any Postman instance to test the API [2].

[1] https://www.postman.com/

[2] https://learning.postman.com/docs/getting-started/importing-and-exporting-data/

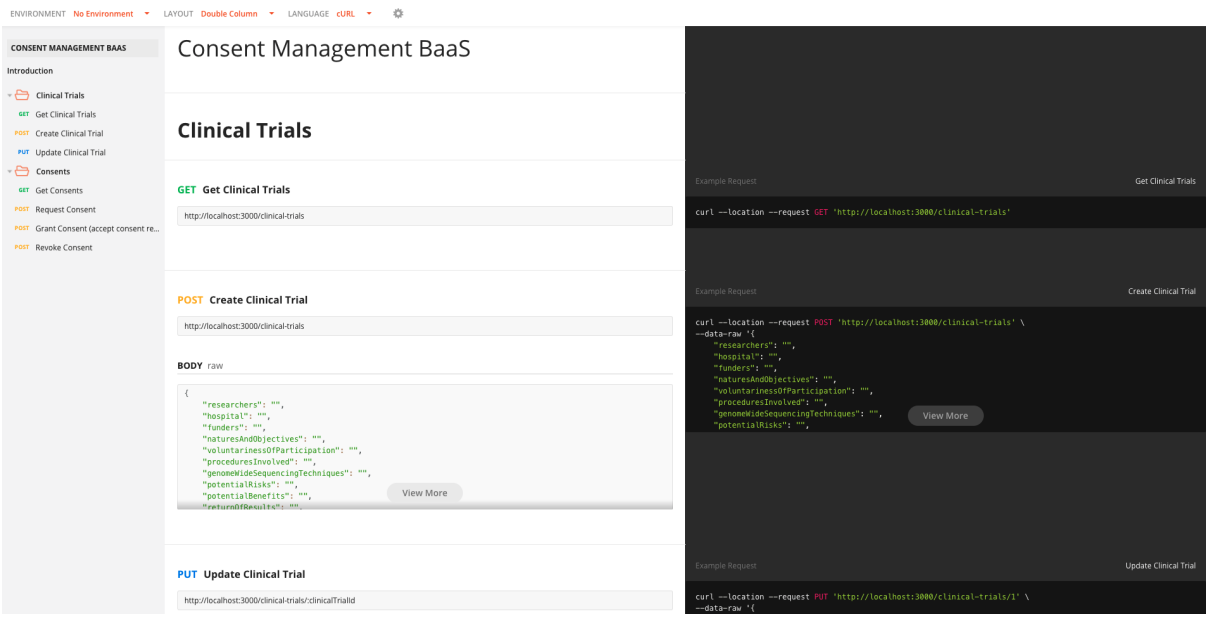

Figure A1.1: API documentation generated with Postman

A1.3 Useful resources

This section lists useful resources that have been used in this dissertation with a strong focus on the technical facet of the project. The aim is to provide any readers enough material to get started. The first subsection provides essential resources that have been used for the implementation of Hyperledger Besu. The second one focuses on Solidity programming languages. Finally, relevant resources used for the service layer implementation are listed.

A1.3.1 Hyperledger Besu

- Hyperledger Besu home page, provides high-level information about the blockchain project
  https://www.hyperledger.org/use/besu
- Hyperledger Besu Wiki which includes more details about the key characteristics of the blockchain
  https://wiki.hyperledger.org/display/BESU/Hyperledger+Besu
- Guide to create a private network with Hyperledger Besu
  https://besu.hyperledger.org/en/stable/Tutorials/Private-Network/Create-IBFT-Network/
- Blockchain's genesis block configuration file
  https://besu.hyperledger.org/en/1.4.2/HowTo/Configure/Genesis-File/

- Documentation on transactions with external account using MetaMask
  Note: a "ready-to-use" example is also available via a Docker container, however, it appears to be less flexible in terms of configuration than using the aforementioned, lower-level guide.
  https://besu.hyperledger.org/en/1.4.2/Tutorials/Examples/Private-Network-Example/#create-a-transaction-using-metamask

- Free gas network configuration
  https://besu.hyperledger.org/en/1.4.2/HowTo/Configure/FreeGas/

- Deploying a smart contract
  This Hyperledger Besu's tutorial can be useful for understanding the principle of deploying a contract. However, it may be confusing as the code snippets provided are not following the same example and require changes to work properly. It is rather an interesting background knowledge when using contract deployment frameworks such as Truffle described in the next bullet point.
  https://besu.hyperledger.org/en/stable/Tutorials/Contracts/Deploying-Contracts/

- Using Truffle to deploy a smart contract
  Both links are interesting to explore in parallel: the first provides a guide on how to use Truffle to deploy a smart contract on an Ethereum blockchain. It provides a complete example off the shelf that can be followed to discover each step of the process. The second link describes the configuration of Truffle to adopt when using it with Hyperledger Besu.
  https://www.trufflesuite.com/docs/truffle/quickstart
  https://besu.hyperledger.org/en/stable/HowTo/Develop-Dapps/Truffle/

A1.3.2 Solidity

- YouTube – Solidity Tutorial - A Full Course on Ethereum, Blockchain Development, Smart Contracts, and the EVM, by freeCodeCamp.org
  This 1h30mins long video provides a strong first overview of Solidity programming language. For a beginner with some background knowledge in another object-oriented programming language and having some theoretical knowledge about what smart contracts can be used for, it is an interesting first hands-on tutorial.
  https://www.youtube.com/watch?v=ipwxYa-F1uY

- Solidity Tutorial (Tutorials Point)
  This tutorial offers a very convenient organisation by technical points of Solidity (e.g. types, loops, arrays, mappings). It is absolutely convenient to use as a glossary when searching for specific technical points along with code examples.
  https://www.tutorialspoint.com/solidity/index.htm

- Solidity documentation
  For a more in-depth understanding of Solidity's principles, its official documentation remains the most precise and detailed resource.
  https://docs.soliditylang.org/en/v0.8.7/

- Ethereum Stack Exchange
  Ethereum is one of Stack Exchange's communities where members frequently ask and answer questions related to Ethereum and all its derivatives, including smart contracts, Solidity and a wide range of other technologies or frameworks. The forum often provides useful answers to specific questions, which are especially appreciated when references to official sources or documentations are provided to verify the answer's statement and/or search for further information in the same topic.
  https://ethereum.stackexchange.com/

A1.3.3 Service layer (Node.js and modules)

- There are plenty of JavaScript tutorials and books freely available on the Internet. The ones from W3Schools and Tutorials Point are particularly well structured and provide plenty of information regarding JS.
  https://www.w3schools.com/js/
  https://www.tutorialspoint.com/javascript/index.htm

- Tutorials Point proposes another tutorial which focuses essentially on Node.js features. It also includes chapters dedicated to the Express.js framework and RESTful APIs.
  https://www.tutorialspoint.com/nodejs/index.htm

- Express.js also provides very useful code snippets for configuring the entry point of the application and its routes (i.e. to handle HTTP requests).
  http://expressjs.com/en/guide/routing.html

- The Truffle framework has "Boxes" which are boilerplates developed by the community to get started when developing applications. One of these boxes focuses on providing API endpoints with Express.js to interact with smart contracts using Truffle. Unfortunately, it is not maintained and quite outdated as most of the code lasts for either 2017 or 2019. However, it provides a very interesting approach to 'bind' Express.js routes with a smart contract via Truffle-contract and web3.js. To be properly used, it requires updates of its dependencies (packages), JS syntax and web3.js configuration.
  https://www.trufflesuite.com/boxes/express-box

A1.4 Source code

This section provides all relevant source code of the implementation of the BaaS. The complete source code is available on GitHub (see section A1.4).

A1.4.1 Blockchain

Genesis block configuration

```
{
  "config" : {
    "chainId" : 1337,
    "muirglacierblock" : 0,
    "ibft2" : {
      "blockperiodseconds" : 2,
      "epochlength" : 30000,
      "requesttimeoutseconds" : 4
    }
  },
  "nonce" : "0x0",
  "timestamp" : "0x58ee40ba",
  "gasLimit" : "0x47b760",
  "difficulty" : "0x1",
  "mixHash" :
  ↪  "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
  "coinbase" : "0x0000000000000000000000000000000000000000",
  "alloc" : {
    "fe3b557e8fb62b89f4916b721be55ceb828dbd73" : {
      "privateKey" :
      ↪  "8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",
      "comment" : "private key and this comment are ignored.  In a real
      ↪  chain, the private key should NOT be stored",
      "balance" : "0xad78ebc5ac6200000"
    },
    "627306090abaB3A6e1400e9345bC60c78a8BEf57" : {
      "privateKey" :
      ↪  "c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
      "comment" : "private key and this comment are ignored.  In a real
      ↪  chain, the private key should NOT be stored",
      "balance" : "90000000000000000000000"
```

```
    },
    "f17f52151EbEF6C7334FAD080c5704D77216b732" : {
      "privateKey" :
      ↪  "ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",
      "comment" : "private key and this comment are ignored.  In a real
      ↪  chain, the private key should NOT be stored",
      "balance" : "900000000000000000000000"
    }
  },
  "extraData" : "0xf87ea000000000000000..."
}
```

IBFT Network configuration

```
{
 "genesis": {
   "config": {
      "chainId": 1337,
      "muirglacierblock": 0,
      "contractSizeLimit": 2147483647,
      "ibft2": {
        "blockperiodseconds": 2,
        "epochlength": 30000,
        "requesttimeoutseconds": 4
      }
    },
    "nonce": "0x0",
    "timestamp": "0x58ee40ba",
    "gasLimit": "0x1fffffffffffff",
    "difficulty": "0x1",
    "mixHash":
    ↪  "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
    "coinbase": "0x0000000000000000000000000000000000000000",
    "alloc": {
       "fe3b557e8fb62b89f4916b721be55ceb828dbd73": {
         "privateKey":
         ↪  "8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",
         "comment": "private key and this comment are ignored.  In a
         ↪  real chain, the private key should NOT be stored",
```

```
            "balance": "0xad78ebc5ac6200000"
        },
        "627306090abaB3A6e1400e9345bC60c78a8BEf57": {
          "privateKey":
          ↪  "c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
          "comment": "private key and this comment are ignored.  In a real
          ↪  chain, the private key should NOT be stored",
          "balance": "90000000000000000000000"
        },
        "f17f52151EbEF6C7334FAD080c5704D77216b732": {
          "privateKey":
          ↪  "ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",
          "comment": "private key and this comment are ignored.  In a real
          ↪  chain, the private key should NOT be stored",
          "balance": "90000000000000000000000"
        }
      }
  },
  "blockchain": {
    "nodes": {
      "generate": true,
        "count": 4
    }
  }
}
```

Nodes configuration

```
genesis-file="../genesis.json"
data-path="data"

bootnodes=["enode://928385df356fde1776...@127.0.0.1:30303"]

p2p-port=30304

rpc-http-enabled=true
rpc-http-api=["ETH","NET","IBFT"]
host-allowlist=["*"]
rpc-http-cors-origins=["all"]
```

```
rpc-http-port=8546

miner-enabled=true
min-gas-price=0
miner-coinbase="0xe8FAeA3207b2C31C6d238b25E35219Fbd318582a"
```

Notes:

- All nodes are configured with similar parameters, except (i) the first node which has no "bootnodes" and (ii) the "p2p-port" and "rpc-http-port" which must be different for each node.
- Although the blockchain is gas-free, Hyperledger Besu requires the "miner-coinbase" address to be specified for the parameter "min-gas-price" to be taken into account.

ConsentsManager smart contract

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract ConsentsManager {

    enum ConsentStatus{ REQUESTED, GRANTED, REJECTED, REVOKED, EXPIRED }

    struct Consent {
        uint id;
        uint parentId;
        uint clinicalTrialId;

        address requestedBy;
        address patient;
        ConsentStatus status;
        uint statusChangedAt;
        address statusChangedBy;
        uint createdAt;
        uint expiresAt;
        string additionalInformation;
        string additionalArtifacts;
    }

    struct ClinicalTrial {
```

```
25          uint id;
26          uint parentId;
27          address createdBy;
28          uint createdAt;
29
30          // General information
31          string[] generalInformation;
32
33          // Data
34          string[] dataCollected;
35          string[] dataProcessingAndPurposes;
36          string[] dataClauses;
37
38          // Rare diseases specific clauses
39          string[] rareDiseaseClauses;
40      }
41
42      struct PatientConsents {
43          uint count;
44          /* Consent[] consents; */
45          mapping(uint => Consent) consents;
46      }
47
48      uint patientsCount = 0;
49      mapping(uint => address) public patients;
50      mapping(address => PatientConsents) public patientsConsents;
51
52      uint public clinicalTrialsCount = 0;
53      mapping(uint => ClinicalTrial) public clinicalTrials;
54
55      address[] public researchers =
        ↪ [0xFE3B557E8Fb62b89F4916B721be55cEb828dBd73]; // Address of
        ↪ researchers
56
57      /**
58       * Check if the msg sender address is in smart contract's stored
    ↪ researchers
59       */
60      function isResearcher() private view returns (bool) {
```

```
61          for (uint i = 0; i < researchers.length; i++) {
62              if (researchers[i] == msg.sender) {
63                  return true;
64              }
65          }
66          return false;
67      }
68
69      /**
70       * Create a new clinical trial
71       */
72      function createClinicalTrial(
73          string[] memory _generalInformation, string[] memory
            ↪ _dataCollected,
74          string[] memory _dataProcessingAndPurposes, string[] memory
            ↪ _dataClauses,
75          string[] memory _rareDiseaseClauses
76      ) public {
77          require(isResearcher(), "Only researchers can create a clinical
            ↪ trial");
78
79          clinicalTrialsCount++;
80          clinicalTrials[clinicalTrialsCount] = ClinicalTrial({
81              id: clinicalTrialsCount,
82              parentId: 0,
83              createdBy: msg.sender,
84              createdAt: block.timestamp,
85              generalInformation: _generalInformation,
86              dataCollected: _dataCollected,
87              dataProcessingAndPurposes: _dataProcessingAndPurposes,
88              dataClauses: _dataClauses,
89              rareDiseaseClauses: _rareDiseaseClauses
90          });
91      }
92
93      /**
94       * Update an existing clinical trial
95       */
96      function updateClinicalTrial(
```

```
97          uint _clinicalTrialId,
98          string[] memory _generalInformation, string[] memory
            ↪  _dataCollected,
99          string[] memory _dataProcessingAndPurposes, string[] memory
            ↪  _dataClauses,
100         string[] memory _rareDiseaseClauses
101     ) public {
102         require(isResearcher(), "Only researchers can create a clinical
            ↪  trial");
103         require(_clinicalTrialId >= 1 && _clinicalTrialId <=
            ↪  clinicalTrialsCount, "Invalid clinical trial ID");
104
105         /*ClinicalTrial storage oldClinicalTrial =
            ↪  clinicalTrials[_clinicalTrialId];*/
106
107         // Create new clinical trial with old version as 'parentId'
108         clinicalTrialsCount++;
109         clinicalTrials[clinicalTrialsCount] = ClinicalTrial({
110             id: clinicalTrialsCount,
111             parentId: _clinicalTrialId,
112             createdBy: msg.sender,
113             createdAt: block.timestamp,
114             generalInformation: _generalInformation,
115             dataCollected: _dataCollected,
116             dataProcessingAndPurposes: _dataProcessingAndPurposes,
117             dataClauses: _dataClauses,
118             rareDiseaseClauses: _rareDiseaseClauses
119         });
120
121         // Early expiration of all consents associated with the previous
            ↪  clinical trial version
122         for (uint i = 0; i < patientsCount; i++) {
123             PatientConsents storage patientConsents =
                ↪  patientsConsents[patients[i+1]];
124             for (uint j = 0; j < patientConsents.count; j++) {
125                 Consent storage consent = patientConsents.consents[j+1];
126                 if (consent.clinicalTrialId == _clinicalTrialId) {
127                     consent.status = ConsentStatus.EXPIRED;
128                     consent.statusChangedAt = block.timestamp;
```

```
129                     consent.statusChangedBy = address(this);
130
131                     // Request new consent
132                     requestConsent(patients[i+1], clinicalTrialsCount,
                        ↪  j+1,
133                                 consent.expiresAt,
                                    ↪  consent.additionalInformation,
134                                 consent.additionalArtifacts);
135                 }
136             }
137         }
138     }
139
140
141     /**
142      * Request consent to a patient (as a researcher)
143      */
144     function requestConsent(
145         address _patientAddr, uint _clinicalTrialId, uint _parentId,
146         uint _expiresAt, string memory _additionalInformation,
147         string memory _additionalArtifacts
148     ) public {
149         require(isResearcher(), "Only researchers can require consent");
150         require(_clinicalTrialId >= 1 && _clinicalTrialId <=
            ↪  clinicalTrialsCount, "Invalid clinical trial ID");
151
152         PatientConsents storage patientConsents =
            ↪  patientsConsents[_patientAddr];
153
154         if (patientConsents.count == 0) {
155             patientsCount++;
156             patients[patientsCount] = _patientAddr;
157         }
158
159         patientConsents.count++;
160         patientConsents.consents[patientConsents.count] = Consent({
161             id: patientConsents.count,
162             parentId: _parentId,
163             clinicalTrialId: _clinicalTrialId,
```

```
164             requestedBy: msg.sender,
165             patient: _patientAddr,
166             status: ConsentStatus.REQUESTED,
167             statusChangedAt: block.timestamp,
168             statusChangedBy: msg.sender,
169             createdAt: block.timestamp,
170             expiresAt: _expiresAt,
171             additionalInformation: _additionalInformation,
172             additionalArtifacts: _additionalArtifacts
173         });
174     }
175 
176     /**
177      * Grant consent as a patient
178      */
179     function grantConsent(uint _consentId) public {
180         PatientConsents storage patientConsents =
            ↪ patientsConsents[msg.sender];
181         require(_consentId >= 1 && _consentId <= patientConsents.count,
            ↪ "Invalid consent ID");
182         require(patientConsents.consents[_consentId].status ==
            ↪ ConsentStatus.REQUESTED, "Consent not in 'REQUESTED'
            ↪ status");
183         patientConsents.consents[_consentId].status =
            ↪ ConsentStatus.GRANTED;
184     }
185 
186     /**
187      * Reject consent as a patient
188      */
189     function rejectConsent(uint _consentId) public {
190         PatientConsents storage patientConsents =
            ↪ patientsConsents[msg.sender];
191         require(_consentId >= 1 && _consentId <= patientConsents.count,
            ↪ "Invalid consent ID");
192         require(patientConsents.consents[_consentId].status ==
            ↪ ConsentStatus.REQUESTED, "Consent not in 'REQUESTED'
            ↪ status");
```

```
193             patientConsents.consents[_consentId].status =
                ↪ ConsentStatus.REJECTED;
194         }
195
196         /**
197          * Revoke consent as a patient
198          */
199         function revokeConsent(uint _consentId) public {
200             PatientConsents storage patientConsents =
                ↪ patientsConsents[msg.sender];
201             require(_consentId >= 1 && _consentId <= patientConsents.count,
                ↪ "Invalid consent ID");
202             patientConsents.consents[_consentId].status =
                ↪ ConsentStatus.REVOKED;
203         }
204
205         /**
206          * Fetch all clinical trials registered
207          */
208         function getClinicalTrials() public view returns (ClinicalTrial[]
        ↪ memory) {
209             ClinicalTrial[] memory ret = new
                ↪ ClinicalTrial[](clinicalTrialsCount);
210             for (uint i = 0; i < clinicalTrialsCount; i++) {
211                 ret[i] = clinicalTrials[i+1];
212             }
213             return ret;
214         }
215
216         /**
217          * Fetch all consents attached to the given clinical trial
218          */
219         function getClinicalTrialConsents(uint _clinicalTrialId) public view
        ↪ returns (Consent[] memory) {
220             require(isResearcher(), "Access not allowed");
221             require(_clinicalTrialId >= 1 && _clinicalTrialId <=
                ↪ clinicalTrialsCount, "Invalid clinical trial ID");
222
223             // Determine the number of consents (non-dynamic array)
```

```
224         uint consentsNb = 0;
225         for (uint i = 0; i < patientsCount; i++) {
226             PatientConsents storage patientConsents =
                ↪ patientsConsents[patients[i+1]];
227             for (uint j = 0; j < patientConsents.count; j++) {
228                 Consent storage consent = patientConsents.consents[j+1];
229                 if (consent.clinicalTrialId == _clinicalTrialId) {
230                     consentsNb++;
231                 }
232             }
233         }
234 
235         // Store all consents in an array that will be returned
236         Consent[] memory consents = new Consent[](consentsNb);
237         uint consentCount = 0;
238         for (uint i = 0; i < patientsCount; i++) {
239             PatientConsents storage patientConsents =
                ↪ patientsConsents[patients[i+1]];
240             for (uint j = 0; j < patientConsents.count; j++) {
241                 Consent storage consent = patientConsents.consents[j+1];
242                 if (consent.clinicalTrialId == _clinicalTrialId) {
243                     consents[consentCount++] = consent;
244                 }
245             }
246         }
247 
248         return consents;
249     }
250 
251     /**
252      * Fetch all consents associated to a patient
253      */
254     function getPatientConsents(address _patientAddr) public view returns
        ↪ (Consent[] memory) {
255         require(_patientAddr == msg.sender || isResearcher(), "Access not
            ↪ allowed");
256 
257         Consent[] memory consents = new
            ↪ Consent[](patientsConsents[_patientAddr].count);
```

```
258         for (uint i = 0; i < patientsConsents[_patientAddr].count; i++) {
259             consents[i] = patientsConsents[_patientAddr].consents[i+1];
260         }
261 
262         return consents;
263     }
264 }
```

A1.4.2 API (Service layer)

Truffle framework configuration

```
1  const PrivateKeyProvider = require("@truffle/hdwallet-provider");
2  const privateKeys = [
3    "0x8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",
     ↪ "0xc87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
4    "0xae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",
5    "0xd39b6bd9bf5a5bd15d212bf5787000e5099dc97aff78ee3dcdd09c8202a808bb" //
     ↪ EXTERNAL, METAMASK
6  ];
7  const privateKeyProvider = new PrivateKeyProvider(privateKeys,
   ↪ "http://127.0.0.1:8545", 0, 4);
8  
9  module.exports = {
10   networks: {
11     besu: {
12       provider: privateKeyProvider,
13       network_id: "*",
14       gasPrice: 0,
15       gas: "0x47b760",
16     }
17   },
18   compilers: {
19     solc: {
20       version: "^0.8.6"
21     }
22   }
23 };
```

Application server

```
const express = require('express');
const app = express();
const port = 3000 || process.env.PORT;
const Web3 = require('web3');
const consentConnect = require('./connection/ConsentsManager.js');
const truffleConfig = require('./truffle-config');

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.get('/getAccounts', (req, res) => {
    consentConnect.start((answer) => {
        res.send(answer);
    });
});

app.get('/clinical-trials', async (req, res) => {
    res.send(await consentConnect.getClinicalTrials());
});

app.get('/consents/:patient', async (req, res) => {
    res.send(await
    ↪ consentConnect.getPatientConsents(req.params.patient));
});

app.post('/consents', async (req, res) => {
    res.send(await consentConnect.requestConsent(req.body));
});

app.post('/consents/:patient/:consentId/grant', async (req, res) => {
    res.send(await consentConnect.grantConsent({
        from: req.params.patient,
        consentId: req.params.consentId,
    }));
});

app.post('/consents/:patient/:consentId/revoke', async (req, res) => {
```

```
    res.send(await consentConnect.revokeConsent({
        from: req.params.patient,
        consentId: req.params.consentId,
    }));
});

app.post('/clinical-trials', async (req, res) => {
    res.send(await consentConnect.createClinicalTrial(req.body));
});

app.put('/clinical-trials/:clinicalTrialId', async (req, res) => {
    res.send(await consentConnect.updateClinicalTrial({
        ...req.body,
        clinicalTrialId: req.params.clinicalTrialId,
    }));
});

app.listen(port, () => {
    console.log("Application listening at http://localhost:" + port);
    consentConnect.web3 = new Web3(truffleConfig.networks.besu.provider);
});
```

Connection with the smart contract

```
const contract = require('@truffle/contract');

const consentmanagerArtifact =
↪  require('../build/contracts/ConsentsManager.json');
const ConsentManager = contract(consentmanagerArtifact);

const apiToBaasClinicalTrialData = (data) => {
  const _generalInformation = [
    data.researchers,
    data.hospital,
    data.funders,
    data.naturesAndObjectives,
    data.voluntarinessOfParticipation,
    data.proceduresInvolved,
    data.genomeWideSequencingTechniques,
```

```
    data.potentialRisks,
    data.potentialBenefits,
    data.returnOfResults,
    data.withdrawalProcedures,
    data.compensationOrReimbursement,
    data.prospectsForCommercialization,
    data.studyDisseminationOrPublication,
    data.recontact,
    data.studyOversight,
  ];
  const _dataCollected = data.dataCollected;
  const _dataProcessingAndPurposes = data.dataProcessingAndPurposes;
  const _dataClauses = [
    data.dataStorageLocation,
    data.dataStorageDuration,
    data.dataAccessForResearch,
    data.dataAccessForAuditing,
    data.dataAccessAndSharing,
    data.dataProtectionsInPlace,
    data.openAccessDatabase,
  ];
  const _rareDiseaseClauses = [
    data.isRareDisease,
    data.rareDiseaseIntroductoryClause,
    data.familialParticipation,
    data.audioVisualImaging,
    data.recontactForMatching,
    data.dataLinkage,
  ];

  return [_generalInformation, _dataCollected,
  ↪ _dataProcessingAndPurposes, _dataClauses, _rareDiseaseClauses];
};

const baasToApiClinicalTrialData = (data) => {
  return {
    researchers: data._generalInformation[0],
    hospital: data._generalInformation[1],
    funders: data._generalInformation[2],
```

```
    naturesAndObjectives: data._generalInformation[3],
    voluntarinessOfParticipation: data._generalInformation[4],
    proceduresInvolved: data._generalInformation[5],
    genomeWideSequencingTechniques: data._generalInformation[6],
    potentialRisks: data._generalInformation[7],
    potentialBenefits: data._generalInformation[8],
    returnOfResults: data._generalInformation[9],
    withdrawalProcedures: data._generalInformation[10],
    compensationOrReimbursement: data._generalInformation[11],
    prospectsForCommercialization: data._generalInformation[12],
    studyDisseminationOrPublication: data._generalInformation[13],
    recontact: data._generalInformation[14],
    studyOversight: data._generalInformation[15],
    dataCollected: data._dataCollected,
    dataProcessingAndPurposes: data._dataProcessingAndPurposes,
    dataStorageLocation: data._dataClauses[0],
    dataStorageDuration: data._dataClauses[1],
    dataAccessForResearch: data._dataClauses[2],
    dataAccessForAuditing: data._dataClauses[3],
    dataAccessAndSharing: data._dataClauses[4],
    dataProtectionsInPlace: data._dataClauses[5],
    openAccessDatabase: data._dataClauses[6],
    isRareDisease: data._rareDiseaseClauses[0],
    rareDiseaseIntroductoryClause: data._rareDiseaseClauses[1],
    familialParticipation: data._rareDiseaseClauses[2],
    audioVisualImaging: data._rareDiseaseClauses[3],
    recontactForMatching: data._rareDiseaseClauses[4],
    dataLinkage: data._rareDiseaseClauses[5],
  }
};

module.exports = {
  start: function(callback) {
    ConsentManager.setProvider(this.web3.currentProvider);

    // Get the initial account balance so it can be displayed.
    this.web3.eth.getAccounts((err, accs) => {
      if (err != null) {
        console.log("There was an error fetching your accounts.");
```

```
92          return;
93        }
94
95        if (accs.length == 0) {
96          console.log("Couldn't get any accounts! Make sure your Ethereum
            ↪  client is configured correctly.");
97          return;
98        }
99        this.accounts = accs;
100       this.account = this.accounts[2];
101
102       callback(this.accounts);
103     });
104   },
105   getClinicalTrials: async function() {
106     ConsentManager.setProvider(this.web3.currentProvider);
107     try {
108       const instance = await ConsentManager.deployed();
109       const res = await instance.getClinicalTrials.call();
110       return res.valueOf().map(e => {
111         return {
112           id: e[0],
113           parentId: e[1],
114           createdBy: e[2],
115           createdAt: e[3],
116           ...baasToApiClinicalTrialData({
117             _generalInformation: e[4],
118             _dataCollected: e[5],
119             _dataProcessingAndPurposes: e[6],
120             _dataClauses: e[7],
121             _rareDiseaseClauses: e[8],
122           }),
123         };
124       });
125     } catch (e) {
126       console.log(e);
127       return "Error 404";
128     }
129   },
```

```
130   createClinicalTrial: async function(data) {
131     ConsentManager.setProvider(this.web3.currentProvider);
132     try {
133       const instance = await ConsentManager.deployed();
134       const res = await instance.createClinicalTrial(
135         ...apiToBaasClinicalTrialData(data),
136         { from: data.from });
137       console.log(res);
138       return res.valueOf();
139     } catch (e) {
140       console.log(e);
141       return "Error";
142     }
143   },
144   updateClinicalTrial: async function(data) {
145     ConsentManager.setProvider(this.web3.currentProvider);
146     try {
147       const instance = await ConsentManager.deployed();
148       const res = await instance.updateClinicalTrial(
149         data.clinicalTrialId,
150         ...apiToBaasClinicalTrialData(data),
151         { from: data.from });
152       console.log(res);
153       return res.valueOf();
154     } catch (e) {
155       console.log(e);
156       return "Error 404";
157     }
158   },
159   getPatientConsents: async function(from) {
160     ConsentManager.setProvider(this.web3.currentProvider);
161     try {
162       const instance = await ConsentManager.deployed();
163       const res = await instance.getPatientConsents.call(from, { from });
164       console.log(res);
165       return res.valueOf().map(e => {
166         return {
167           id: e[0],
168           parentId: e[1],
```

```
169             clinicalTrialId: e[2],
170             requestedBy: e[3],
171             patient: e[4],
172             status: e[5],
173             statusChangedAt: e[6],
174             statusChangedBy: e[7],
175             createdAt: e[8],
176             expiresAt: e[9],
177             additionalInformation: e[10],
178             additionalArtifacts: e[11],
179           }
180         });
181       } catch (e) {
182         console.log(e);
183         return "Error 404";
184       }
185     },
186     requestConsent: async function(data) {
187       ConsentManager.setProvider(this.web3.currentProvider);
188       try {
189         const instance = await ConsentManager.deployed();
190         const res = await instance.requestConsent(
191           data.patientAddr, data.clinicalTrialId,
192           data.parentId, data.expiresAt,
193           data.additionalInformation, data.additionalArtifacts,
194           { from: data.from });
195         return res.valueOf();
196       } catch (e) {
197         console.log(e);
198         return "Error 404";
199       }
200     },
201     grantConsent: async function(data) {
202       ConsentManager.setProvider(this.web3.currentProvider);
203       try {
204         const instance = await ConsentManager.deployed();
205         const res = await instance.grantConsent(
206           data.consentId,
207           { from: data.from });
```

```
208         return res.valueOf();
209       } catch (e) {
210         console.log(e);
211         return "Error 404";
212       }
213     },
214     revokeConsent: async function(data) {
215       this.web3.eth.handleRevert = true;
216       ConsentManager.setProvider(this.web3.currentProvider);
217       try {
218         const instance = await ConsentManager.deployed();
219         const res = await instance.revokeConsent(
220           data.consentId,
221           { from: data.from });
222         return res.valueOf();
223       } catch (e) {
224         console.log(e);
225         return "Error 404";
226       }
227     },
228
229
230 }
```

Transaction decoder script

```
1  const Web3 = require('web3');
2  const truffleConfig = require('./truffle-config');
3  const abiDecoder = require('abi-decoder');
4
5  async function main() {
6      const web3 = new Web3(truffleConfig.networks.besu.provider);
7      const txh = process.argv[2];
8      web3.eth.getTransaction(txh, (error, result) => {
9          if (error) {
10             console.log(error);
11             return;
12         } else if (!result) {
13             console.log("No result.");
```

```
14              return;
15          } else {
16              abiDecoder.addABI(
                ↪  require('./build/contracts/ConsentsManager.json').abi);
17              console.log(abiDecoder.decodeMethod(result.input));
18          }
19
20          process.exit();
21      });
22  }
23
24  main();
```