Technical University of Crete

Department of Electrical and Computer Engineering

Methodology for designing GDPR compliant IoT applications

Christos Karageorgiou Kaneen
ckarageorgkaneen@gmail.com

Supervisor:
Euripides G.M Petrakis
petrakis@intelligence.tuc.gr

Thesis committee:
Vasilis Samoladas
vsam@softnet.tuc.gr

Antonios Deligiannakis
adeli@softnet.tuc.gr

December, 2019

Abstract

As of May 2018, the enforcement of the EU's General Data Protection Regulation (GDPR) has introduced new standards for organizations processing personal data of EU residents. With the purpose of giving people more control over their data, as well as protecting them from potential data breaches, proving compliance with GDPR requirements, to regulators who mandate it, has become an ever-increasing priority for most organizations, with steep fines to be paid for privacy violations. Due to the difficulty of analyzing a running system for evaluating its compliance, GDPR requirements must be taken into consideration during the system's design phase. In this work, we provide the methodology for analyzing these requirements and incorporating them into the design process of a Remote Patient Monitoring application. Since there is no universal methodology that covers all application domains and systems, we focus on a single such application domain: an IoT Service Oriented Architecture design for the cloud. By analyzing the dependencies between all system components (such as personal data, users, cloud services, etc.), we are able to create data-filled reports (related to the GDPR's personal data demands) that can be used for evaluating compliance. In order to show proof of concept, we apply the aforementioned analysis and represent our system's information of component properties, requirements and dependencies by means of a labeled-property graph in a graph database. The decision of whether the system is GDPR compliant can be reached once a series of questions (expressed as queries run upon the system graph) have been answered and analyzed. The rationale behind our approach deems it much easier to evaluate GDPR compliance once the designed system's graph has been constructed. In summary, we demonstrate how such a graph can be created by taking as input both: (a) design requirements and (b) GDPR requirements. We also demonstrate how the evaluation of GDPR compliance lies within analyzing the results of queries run upon the graph in a graph database.

Contents

Chapter 1

Introduction

The EU's General Data Protection Regulation (GDPR) was created with the purpose of regulating the way EU citizens' personal data is protected by the organizations that process it. All organizations that hold and process EU citizens' personal data are required to adapt to the new regulation. [1][2]

Considering the challenges of a rapidly evolving technological world that have up until now given rise to major data breaches (such as the recent Facebook–Cambridge Analytica scandal or the AOL search data leak more than a decade ago), the GDPR has become, since its implementation on May 18th 2018, a strict prerequisite for any organization dealing with EU personal data. Proving compliance with the GDPR requires having the ability to extensively document the processing procedures involving EU citizens' personal data, by evaluating its lawfulness, providing information on the security measures enforced for its protection and ensuring that sufficient processing agreements (i.e.consent) are in place. The importance of the regulation lies within its purpose: protecting EU citizens' rights by clarifying the actions organizations must take to safeguard them. [3][4]

1.1 Motivation

Due to the requirements posed by the GDPR (fundamental terms of which are described in Chapter 2), organizations must be able to comply with strict rules about how personal data (any data that can be used to identify a person) is manipulated within their systems. More specifically, the road to compliance involves tackling complex issues such as:

(a) How personal data is stored, secured, processed, transmitted and erased

(b) How the policies related to its retention are enforced [5]

(c) How proof of provided or revoked written consent, for its obtainment and usage, can be documented

Such issues lead to the question of what are the minimal steps that an organization must take to prove basic GDPR compliance, by taking into consideration the interactions between its systems' components and the personal data it processes on behalf of its users.

In recent years, the rapid expansion of growing domains such as Big Data, AI, Cloud computing and the IoT, among others, has brought to the forefront new challenges primarily linked to the personal data and trust, such as data transparency (the ability to easily access and work with data), data security and the clear presentation of terms of data processing activities. Such challenges have given rise to the problem of designing a system within relevant contexts (such as the IoT and the Cloud), by taking into account basic GDPR requirements (briefly described in 1.2 and more thoroughly in 2.1.3) and querying it for preliminary compliance evaluation. This problem is the one that we attempt to provide a solution to in the following chapters. [6]

1.2 Problem definition

Recently, Neo4j suggested that mastering GDPR requires prior knowledge in regards to the ways personal data is stored and managed in a system. More specifically, as part of its Privacy Shield solution, the company provides a comprehensive solution for compliance with the GDPR regulation, that is not, however, meant to directly answer whether a system is GDPR compliant or not but, rather, to provide the means (in graph form) of evaluating it, by representing how personal data is stored and managed. This approach, based on graph modeling and querying, attempts to connect personal data across all system components (i.e. parts such as services and databases) and track where and how personal information is stored, how it is used, how it moves to different locations or systems, who has access to this data and whether users have provided or revoked consent for doing so, among other activities more thoroughly described further on. [7][8]

By elaborating on the above, in order to answer whether a system meets the GDPR compliance criteria, the following questions must be answered:

0. **What are the fundamental data entities?**[1]
   The persons to whom the personal data belongs and user and device entities related to it

1. **What data do you have?**
   The personal data held within the organization's systems

2. **Where is the data stored?**
   The internal systems (services, databases, etc.) in which the personal data is stored and at what geographical location

3. **How and when did you obtain data?**
   The fundamental entities (Q0) the personal data is related to, the consent provided by persons (for its processing within the organization) and the time at which it was provided

4. **Why do you have the data?**
   The way in which the personal data is used (data usage) and the events of data processing that involve it

5. **Who has access to the data?**
   The entities (such as users and persons) that have access to the personal data

6. **Do you have permission to use the data? For what purposes?**
   Whether the organization's users have provided or revoked consents, as well as their respective data usages

7. **Is the data secure?**
   The security of personal data (at-rest and in-transit) within the components of the system. This question regards whether all entities that stored or process personal data employ the appropriate security mechanisms for its protection (OAuth2.0, encryption, etc.)

8. **How does the data travel through your systems?**
   Detailed information related to personal data lineage and data tracking: how and when personal data moves within the organization's internal systems [9]

[1] Question 0 was added for matters of convenience in our upcoming diagram design process (Chapter 3)

9. **Does the data ever cross international borders?**
   Whether the personal data crosses EU or non-EU borders into non-EU or EU countries, respectively

The information described in these answers needs to be explicitly represented in a formal way. Neo4j suggests that this information is best represented by means of a semantic graph, where each one of the questions in the above list can be formulated as a query on the graph whose results yield the data necessary for evaluating GDPR compliance. Judging whether a system is compliant is not an automated process (i.e. the decision cannot be made algorithmically). Instead, given the answers to the above questions (the results of executing the queries corresponding to each question), the final decision must be made by a human expert.

In our approach, we inspect the requirements of each and every one of the aforementioned questions and:

(a) Demonstrate how an IoT system that encompasses fundamental GDPR personal data requirements can be designed

(b) Answer the questions by querying the system, providing a means for evaluating GDPR compliance

The next section describes our approach to tackling this problem, in more detail.

1.3 Proposed solution

Solving the GDPR compliance problem in its generality is a difficult problem. Although our approach is inspired by Neo4J's Privacy Shield solution, it is not just an application of an existing methodology. In fact, no such as methodology exists. Although Neo4j's commercial solution provides guidelines for answering the GDPR compliance problems, it does not openly demonstrate in detail, how to apply them to an existing system.

To provide a means for evaluating GDPR compliance, compliance requirements must be taken into account in the system design phase. Presumably, the design of a system depends on a system's functional and non-functional requirements, which differ for systems in different application domains. The solution thus depends on system functionality and the specific answers to each one of the GDPR questions can be provided once the requirements of a system are specified. Thus, in order to tackle this difficult problem in its generality, we must take GDPR requirements into account from the beginning of the requirements analysis phase of the system under consideration. This is realized through a formal step-by-step procedure that involves the analysis and specification of the GDPR requirements encoded in the 10 aforementioned questions (1.2) that must be encompassed in the design of a remote patient monitoring IoT system, in a series of UML class diagrams. [10]

After designing a diagram for each question, we end up with our system's unified information class diagram that encapsulates all information related to the requirements necessary for the evaluation of basic GDPR compliance. Therefore, the problem of deciding upon the GDPR compliance of a system is transformed into designing the system's information class diagram by considering related requirements and using it to answer the preliminary compliance questions. To do so, the class diagram must be made queryable, a task that can be accomplished by transforming it into a semantic graph whose node and relationship properties represent the knowledge of the application.

Appropriate tools for this representation are the property graph database model and ontologies, such as OWL. In our approach, we decided to utilize the former for expressing and querying the contents of the graphs due to its simplicity in usage and wide range of available graph tools (e.g. databases) supporting it, opposed to the latter's high formality and lack of appropriate tools. [11][12][13][14]

Based on this approach, a system whose class diagram has been represented as a graph in a graph database (such as Neo4j) can be evaluated for GDPR compliance by analyzing the responses to the 10 GDPR queries run upon it. Since, however, a system's class diagram also encodes peculiarities related to its specific operation, the above approach must be applied on each different system separately (as, obviously, no universal class diagram exists for all system domains). In this work, to show proof of concept, we deal with providing GDPR compliance evaluation for the use case of an IoT and Cloud Service Oriented Architecture (SOA) of a remote patient monitoring system that involves the handling of sensitive health-related personal data, characteristic to a broad array of application domains wherein proving GDPR compliance is necessary. [15]

In short, the proposed solution employs an incremental approach. First, we design a Remote Patient Monitoring IoT System, based on SOA principles, by following a valid design approach based on UML and by also taking into account GDPR's personal data requirements reflected in the aforementioned questions (related to preliminary compliance). Then, we represent the designed system as a graph. Finally, having defined the knowledge of the system in a graph database, we answer the questions by appropriately querying the resulting graph.

1.4 Existing work

Although the academic literature on such a subject is still relatively scarce, work loosely related to ours includes:

- "**Designing Data Protection for GDPR Compliance into IoT Healthcare Systems**", a paper wherein researchers propose a data labeling model for supporting access control operations on privacy-critical patient data in the design of a GDPR compliant IoT healthcare system. This paper deals with a limited set of data transparency, access and privacy aspects of the regulation. [16]

- "**A Consent and Data Management Model**", a project that "aims to utilize semantic web technologies, such as OWL, in order to provide a common and cohesive framework for representing and aiding in the compliance of legislations like the GDPR. This research provides the integration of data management across different information systems specifically adhering to the GDPR and helping controllers to demonstrate compliance". [17]

For a more thorough description of the above work, in addition to how it compares to our own, see 2.3.

Outside the world of academia, a number of companies, such as Neo4j (as mentioned earlier), trust-hub and Cambridge Intelligence, provide commercial solutions primarily based on graph analytics tools, such as graph database visualization, for dealing with the problem of GDPR compliance. [18][19]

1.5 Thesis structure

In Chapter 2, we present the tools and concepts we utilized for both the design of our system, the construction of its graph representation and its querying, as well as further details on related academic work. In Chapter 3, we describe the system we are going to design, the steps for designing it based on GDPR principles and the methodology for importing it into a graph database and queries for answering the 10 compliance questions. In it, we also provide a demonstration of our method using the Neo4j graph database. Finally, in Chapter 4 we summarize our approach by stating our conclusions and mentioning future work.

Chapter 2

Background

2.1 The General Data Protection Regulation

Although the EU Parliament approved of the GDPR and adopted it in April 2016, the regulation actually took effect on the 25th of May, 2018. Within its scope it:

"1. Lays down rules relating to the protection of natural persons with regard to the processing of personal data and rules relating to the free movement of personal data.

2. Protects fundamental rights and freedoms of natural persons and in particular their right to the protection of personal data."

Its core aim being to protect EU/EEA citizens from privacy and data breaches by bringing laws around personal data, privacy and consent and by enforcing the obligation of privacy by design. [20]

Under GDPR, organizations must be able to prove that personal data is gathered legally. Furthermore, organizations collecting and processing personal data are required to protect it from exploitation and misuse.

2.1.1 Definitions

Important terms defined by the regulation that are prerequisites for fully-grasping our work include [21]:

- '**personal data**': any information relating to a data subject, meaning a natural person who can be identified in particular by reference to identifiers such as names, id numbers, location data, online ids or factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person
- '**biometric data**': personal data resulting from specific technical processing relating to the physical, physiological or behavioral characteristics of a natural or legal person, which allow or confirm the unique identification of that natural person
- '**data concerning health**': personal data related to the physical or mental health of a natural person, including the provision of health care services, which reveal information about his or her health status
- '**processing**': any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organization, structuring, storage, adaptation, alteration, retrieval, etc.

- '**pseudonymisation**': the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organizational measures to ensure that the personal data is not attributed to an identified or identifiable natural person
- '**controller**': the entity (natural or legal person, public authority, etc.) which, alone or jointly with others, determines the purposes and means of the processing of personal data
- '**processor**': an entity (natural or legal person, public authority, etc.) which processes personal data on behalf of the controller
- '**consent**': means any freely given, specific, informed and unambiguous indication of the data subject's wishes signifying agreement to the processing of their personal data
- '**third party**': an entity (natural or legal person, public authority, etc.) other than the data subject, controller, processor and persons that is authorized, under the direct authority of the controller or processor, to process personal data

2.1.2 Compliance

Being able to demonstrate GDPR compliance requires the implementation of measures meeting certain data protection criteria, on behalf of the data controller, that involve the collecting and processing personal data. According to *Art. 25*: data protection measures are required to be designed into the development of business processes for products and services (privacy-by-default considerations). Thus, the implementation of effective data protection measures in addition to the acts of being able to demonstrate the compliance of data processing activities become liabilities that lie with the controller. Significant data protection measures include the [22]:

- **Identification of personal data**: identifying the personal data being held and where it resides is the first step on the road to compliance
- **Protection of personal data**: protecting personal data means that all components of a system that process personal data must be designed with data protection considerations in mind and specific technical and organizational measures must be implemented to safeguard this information during processing. GDPR explicitly champions encryption and pseudonymization techniques that reduce the risks associated with data processing
- **Auditing of personal data**: the inspection of personal data for determining what data the organization should hold and should erase and how it is processed based on organizational needs and status of consent (e.g. withdrawn)
- **Management of consent and data usage**: managing consent requires it to be free, specific, informed and unambiguous. Furthermore, controllers must clearly state their intended data usage purposes and request explicit consent each time they do so

Therefore, some of the most impactful requirements an organization must be able to consider and measures they must implement for achieving basic GDPR compliance are:

- **Data Location & Consent**: Controllers are obliged to acquire explicit consent from users for the processing of their personal data for specific and clearly-stated purposes at specific locations, as well as for transferring it outside the EU

- **Right to access**: Controllers must provide the means for users to access their data at any time in a structured, machine-readable format
- **Data portability**: Data subjects should be able to transmit their personal data to another controller without hindrance from the controller to which the personal data has been provided
- **Right to rectification**: Data subjects should be able to correct any erroneous personal data that controllers store
- **Right to be forgotten**: Data subjects should be able at any time to prompt controllers into erasing all of their personal data, ceasing its distribution, and potentially inhibiting any processors from continuing to use it
- **Privacy by design**: Controllers and processors must design their services considering the "state of the art" privacy and security technology to protect their users' data. In the event of a data compromise, controllers and processors must notify data subjects

The abstract data model of Fig. 1 summarizes the prerequisite knowledge of personal data an organization must have for proving compliance: the processes that use it, the platforms that store and exchange it, the people who own it and have access to it and the places where it is stored at.

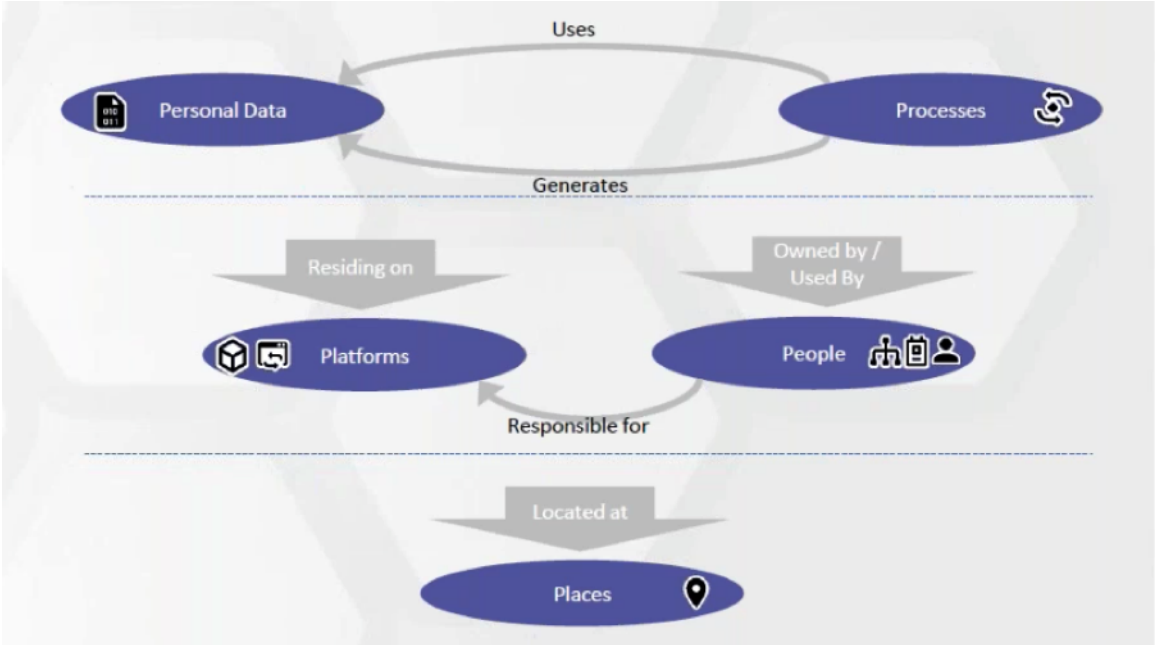

Figure 1: trust-hub's core personal data model. [23]

According to the proposed approach of Neo4j, an organization seeking to meet GDPR demands must be able to answer the following difficult questions of Fig. 2 in order to provide insight into the transparency and traceability of the personal information their system holds and thus a means for evaluating its compliance. In other words, these questions reflect the necessary knowledge an organization must be able to prove it holds, for satisfying the key personal data requirements related to GDPR compliance. These requirements involve being able to track personal data movement across all internal and external systems, applications and other components of a platform (mainly its data lineage: where underlying data originates from and how it flows through the components). More specifically:

- where the personal data was acquired
- whether consent for its usage was obtained and when
- to and from which systems it moves over time
- which geographical location it is stored at
- how it gets processed

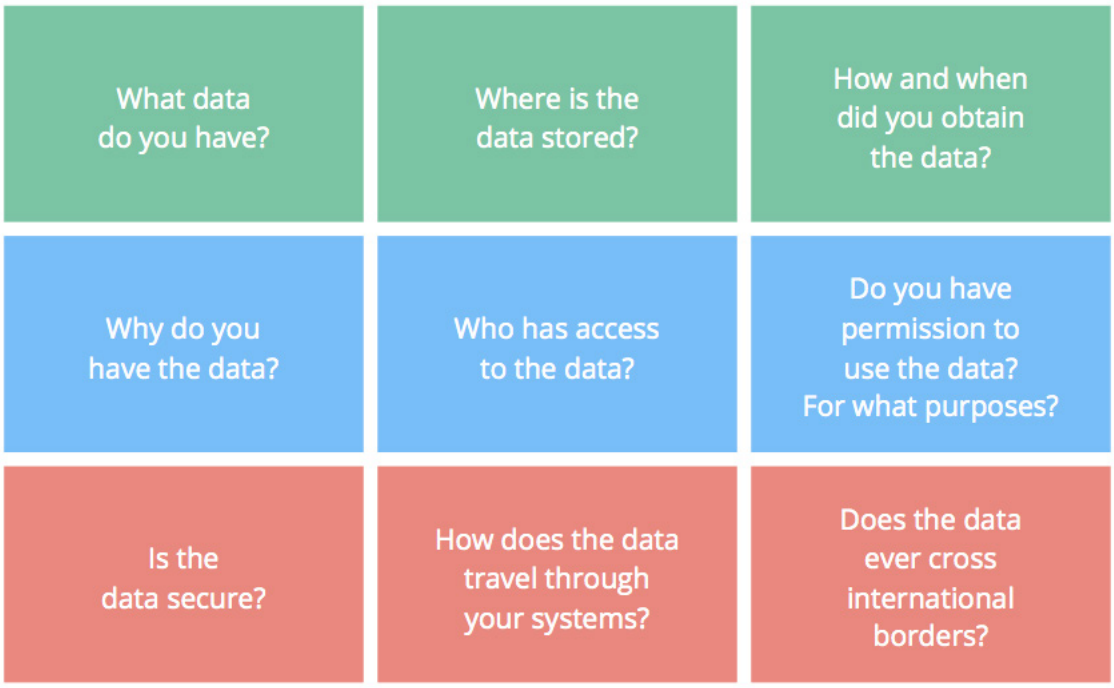

Figure 2: Neo4j's key GDPR requirements. [24]

Since the paths that personal data follows are complex and unpredictable, they are best visualized as graph structures, implemented using a graph database. This choice of structure and database best suits the nature of problems such as that of GDPR compliance in which data relationships of interconnected systems and other entities such as users, personal data, consent etc. are as important as the data itself. [25]

By creating a graph representation of a given platform in a graph database, answers about the personal data being held can be provided by means of querying upon the graph and extracting the required information (intertwined in nodes and relationships) in tabular form. The resulting answer tables can then be used for reaching the final decision on whether or not the system is GDPR compliant through means of evaluation and analysis by the following qualified subjects:

- EU Regulators demanding proof of compliance [26]
- Data Protection Officers and internal organization staff responsible for preserving privacy across all your systems [27]

Additionally, answers can also be provided to people such as users and consumers who want to find out what an organization knows about them and how their data is being used. The requirements thus associated with these questions will be analyzed in the next section and utilized in Chapter 3 for both the construction and querying of an IoT platform.

2.1.3 Compliance questions

0. **What are the fundamental data entities?**
   The data subjects the personal data belongs to and the user and device entities related to it

1. **What data do you have?**
   The personal data entities associated with the fundamental data entities

2. **Where is the data stored?**
   The database systems in which the personal data is stored and the geographical location where they are located.

3. **How and when did you obtain data?**
   The fundamental data entities it is related to, meaning the entities it was derived from (e.g. a device generating biometric personal data) and the time at which explicit consent of a specific type was provided by the data subject (the data belongs to), for enabling a data usage for its processing (see Q4).

4. **Why do you have the data?**
   The data processes involving the personal data and the purpose of the processing (data usage) that specific consent types enable.

5. **Who has access to the data?**
   The user (data owner) that is related to the data subject it belongs to and other users that have permission to access the data of the owner.

6. **Do you have permission to use the data? For what purposes?**
   Whether the data subject the personal data belongs to has provided and not revoked a consent for its usage.

7. **Is the data secure?**
   Whether the systems the personal data travels through (or could potentially travel through) and store it, and the fundamental data entities it is related to (see Q1) and have access to it (see Q5) employ proven security mechanisms such as OAuth2.0, master keys, and encryption, in-transit and at-rest.

8. **How does the data travel through your systems?**
   Detailed information on the tracking of personal data: the systems to and from which the personal data travels, from the time of its acquisition to its usage by data processes, in the context of given or revoked consents.

9. **Does the data ever cross international borders?**
   Whether the personal data ever crosses international borders refers the inspection of country information associated with the entities that exchange personal data in the context of data processing. Whether the personal data crosses EU or non-EU borders into non-EU or EU countries, respectively (see Q4 and Q8).

2.2 Concepts & Tools

Our approach follows the incremental design of a Remote Patient Monitoring System, that is based on previous work (described in 2.3). In this section, we present tools and technologies that we utilize throughout the design, implementation and demonstration of our solution.

2.2.1 UML

The information diagrams that form the basis of our System's graph model were created using a subset of UML and were drawn using the web diagramming software draw.io. In order to more clearly present the data contained in our designed System's diagrams and efficiently answer the aforementioned questions related to GDPR compliance by means of queries, we must create an equivalent graph (a set of nodes and relationships, instead of UML classes and associations) in a graph database. [28]

2.2.2 Property graphs

The property graph database model is represented by nodes (vertices), relationships (edges), properties and labels that abide by a set of basic rules:

1. Nodes can have multiple labels for grouping and categorization purposes
2. Relationships can have one label (usually called a relationship type)
3. Both nodes and relationships can store properties represented by key-value pairs
4. All relationships are directed edges, starting at one node and ending at another

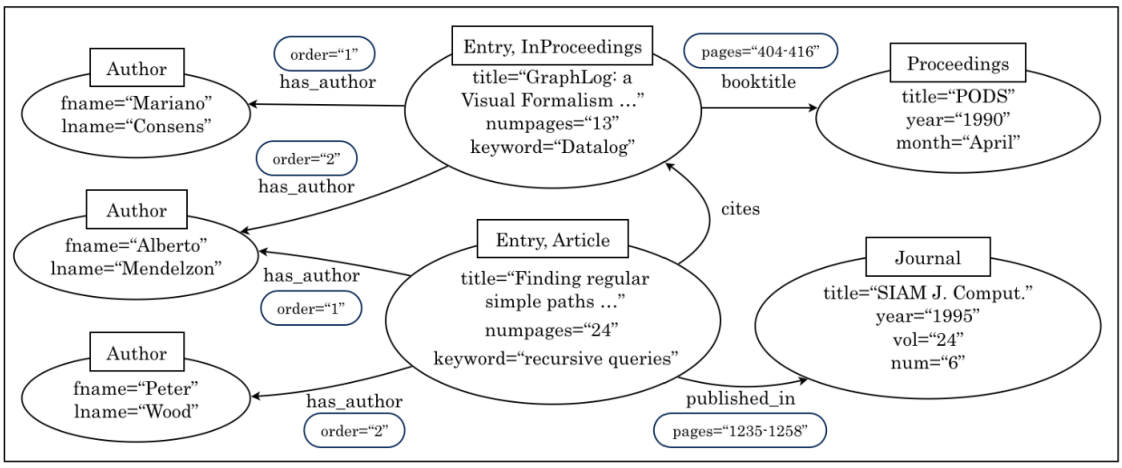

Figure 3: Example of a property graph representing bibliographic information [12]

2.2.2.1 Querying property graphs

In order to provide a solution that is general, we must define an abstract query language that will be used for import and answer querying purposes (see Chapter 3). For this reason we define *PseudoQL*, a highly-readable graph query pseudo-language, borrowing concepts from Neo4j's *Cypher* (2.2.3.2) and *SQL*. This language, created by us, is built upon the following set of commands (Tables 1 - 4) for reading, writing, returning and creating data in a property graph. [29]

Read commands	Description
SELECT	Specify patterns to find in the database
OPTIONAL SELECT	Specify patterns to find in the database and use null for patterns that do not exist

Table 1: PseudoQL read commands

Read sub-commands	Description
WHERE	Add constraints to patterns in a **SELECT** or **OPTIONAL SELECT** command
ORDER BY [**ASC** | **DESC**]	Specify that the output should be sorted in either ascending (default) or descending order (follows the **RETURN** command)

Table 2: PseudoQL read sub-commands

Write commands	Description
CREATE	Create nodes and relationships

Table 3: PseudoQL write commands

Return commands	Description
RETURN	Define what to include in the query return result

Table 4: PseudoQL return commands

PseudoQL's pattern constructs for reading and creating nodes and relationships in the graph are based on an intuitive and easy-to-read syntax (Tables 5 - 6).

Node syntax examples	Description
()	A plain node
(var)	A node that can be referred to through the var variable
(:Label1:Label2: . . . :LabelN)	A node with labels Label1, Label2, . . . and LabelN
(var:Label1:Label2: . . . :LabelN)	A node with labels Label1, Label2, . . . and LabelN that can be referred to through the var variable
(var:Label {key1: value1, key2: value2, . . . , keyN: valueN})	A node with label Label and properties: {key1: value1, key2:value2, . . . , keyN:valueN} that can be referred to through the var variable

Table 5: PseudoQL node syntax

Relationship syntax example	Description
−−	An undirected relationship
−−>	A directed relationship
-[var]->	A directed relationship that can be referred to through the var variable
-[var:TYPE]->	A directed relationship with type TYPE that can be referred to through the var variable
-[var:TYPE {key1: value1, key2: value2, . . . , keyN: valueN}]->	A node with label Label and properties: {key1: value1, key2:value2, . . . , keyN:valueN} that can be referred to through the var variable

Table 6: PseudoQL relationship syntax

Combining the syntax for nodes and relationships we can express patterns such as the following (Table 7). For a better understanding of PseudoQL, let's write two queries (1 and 2) and inspect their results.

Pattern example
(node1_var:Label1:Label2 {key1: value1, key2: value2}) -[rel_var:TYPE {key1: [1, "2", [3]]}]-> (node1_var:Label3:Label4 {key1: value1, key2: value2})

Table 7: *PseudoQL* pattern example

```
CREATE (author1:Author {fname: "Mariano", lname: "Consens"})
CREATE (author2:Author {fname: "Alberto", lname: "Mendelzon"})
CREATE (author3:Author {fname: "Peter", lname: "Wood"})
CREATE (entry1:Entry:InProceedings {
    title: "GraphLog: a Visual Formalism...",
    numpages:"13",
    keyword:"Datalog"})
CREATE (entry2:Entry:Article {title: "Finding regular simple paths ...",
    numpages: "24",
    keyword: "recursive queries"})
CREATE (proceedings:Proceedings {title: "PODS", year: "1990",
    month: "April"})
CREATE (journal:Journal {title: "SIAM J. Comput.", year: "1995",
    vol: "24", num: "6"})
CREATE (proceedings)<-[:booktitle{pages: "404-416"}]-(entry1)
    <-[:cites]-(entry2)-[:published_in {pages: "1235-1258"}]->(journal)
CREATE (author1)<-[:has_author {order: "1"}]-(entry1)
    -[:has_author {order: "2"}]->(author2)
CREATE (author2)<-[:has_author {order: "1"}]-(entry2)
    -[:has_author {order: "2"}]->(author3)
RETURN *
```

Listing 1: Create the bibliographic information property graph (Fig. 3)

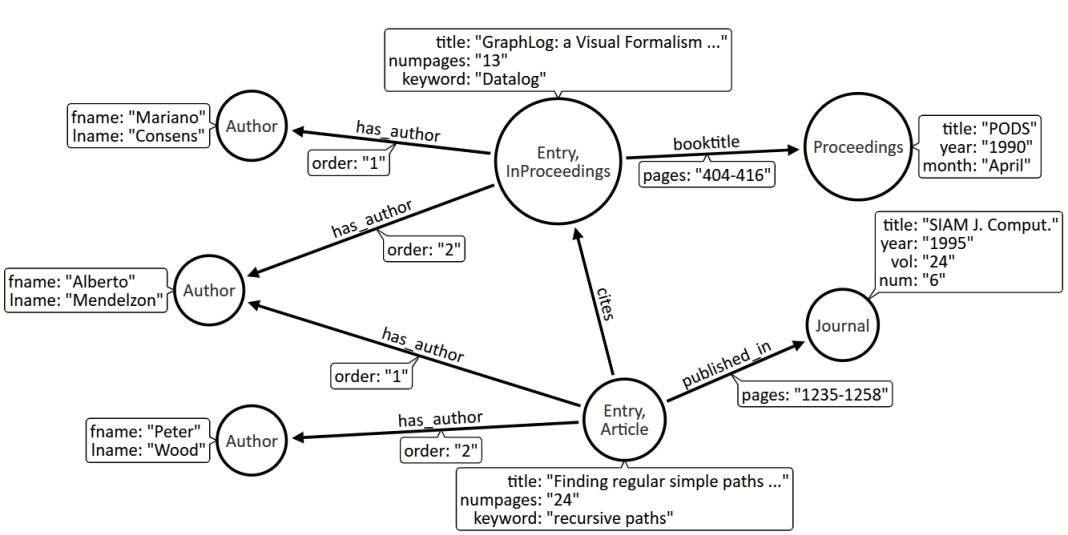

Figure 4: Resulting graph of *PseudoQL* query 1

```
SELECT (author:Author)<-[:has_author]-(:Article)
    -[pub_in:published_in]->(:Journal {title: "SIAM J. Comput."})
OPTIONAL SELECT (author)
    <-[:has_author]-(entry_in_proceedings:Entry:InProceedings)
RETURN author.lname as SIAMJComput_Author_Surname,
    pub_in.pages as PublishedAtPages,
    entry_in_proceedings.title as EntriesInProceedings
```

Listing 2: Query the bibliographic information graph (Fig. 3, 4)

- Query 1: Create and return the bibliographic information property graph (Fig. 3, 4)
- Query 2: In the same graph, find and return the last name of the authors whose article was published in the "SIAM J. Comput." journal, at which pages and any other entries in proceedings they may have. This query returns the data of Table 8, where the *EntriesInProceedings* field for *Wood* is *null* since he has not written any in-proceedings entry.

SIAMJComput_Author_Surname	PublishedAtPages	EntriesInProceedings
Wood	1235-1258	null
Mendelzon	1235-1258	GraphLog: a Visual Formalism...

Table 8: Query 2 result

2.2.2.2 *UML-to-property-graph* mapping

Answering the questions for evaluating GDPR compliance requires a graph representation of the UML class diagrams (resulting from the System's design process presented in Chapter 3). In other words, once the diagrams have been constructed and combined, they must be imported into a graph database as a single graph. This task can be accomplished by following the simple *UML-to-property-graph* mapping, proposed by us, as presented in Table 9.

UML	**Property Graph**
Class	Node
Class name	Node label
Association or Association class	Relationship
Association or Association class name	Relationship type
Attribute	Property
Class-subclass correlation [1]	Node with class and subclass names and attributes as labels and properties

Table 9: *UML-to-property-graph* mapping

As an example, the class diagram of Fig. 5 would be modeled, according to our mapping, as the property graph of Fig. 6.

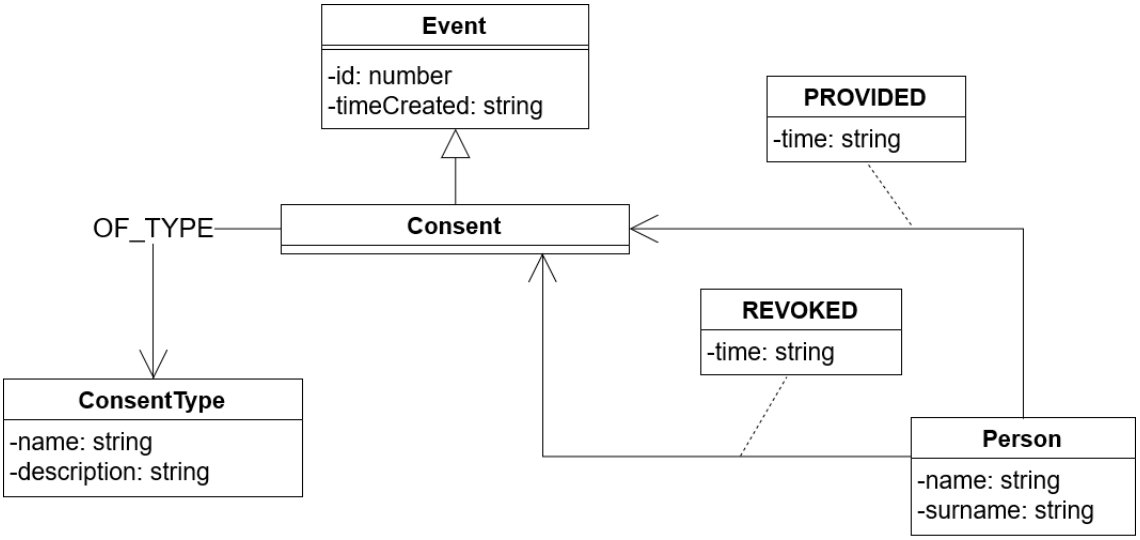

Figure 5: An example UML class diagram

[1] E.g. the class-subclass correlation: class A_N, which is subclass of A_{N-1}, ..., which is subclass of A_1 would be represented by a node with labels A_N, A_{N-1}, ... and A_1 containing the attributes of each class as properties.

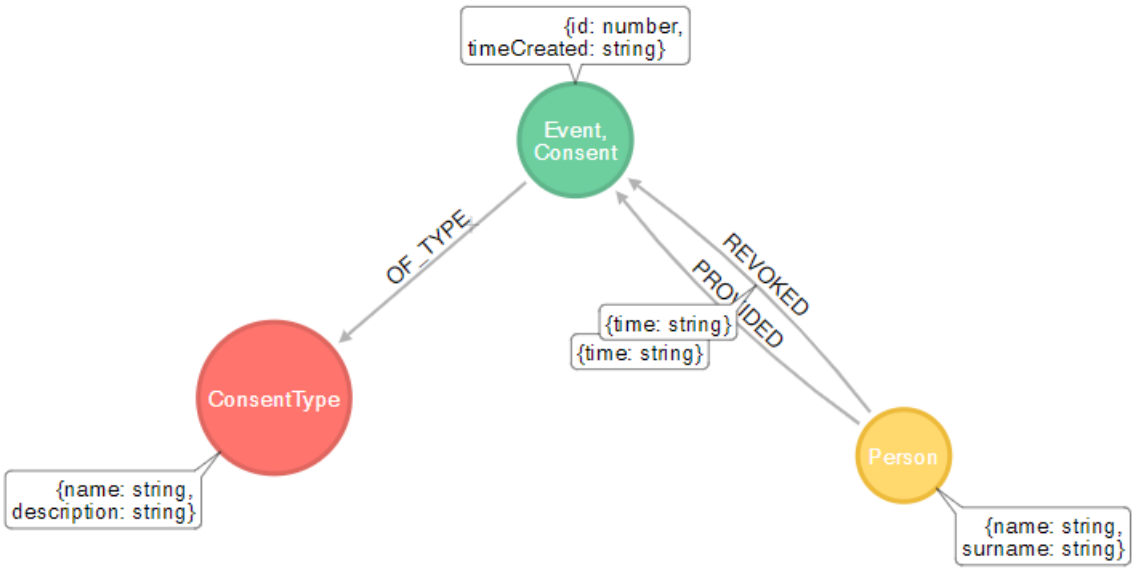

Figure 6: The property graph of the example UML class diagram (Fig. 5)

Notice how the UML *Event* class and the *Consent* subclass both get mapped unto a single property graph node with the class and subclass names as labels and the *id* and *timeCreated* attributes inherited as properties.

2.2.3 Graph databases

For comfortably manipulating a vast array of information modeled as nodes and relationships, we decided to adopt the graph database technology, for accomplishing tasks such as graph creation and querying, for answering the questions required for evaluating GDPR compliance. These NoSQL databases are inherently designed to handle complex graph constructs with relative simplicity and speed, in contrast to traditional relational databases, whose rigid schemas, strict table/column layout confinements and *JOIN* requirements deem them unsuitable for our task. [30]

2.2.3.1 Neo4j

The graph database we decided to adopt for our Demo (3.4) was Neo4j, a popular labeled-property graph database management system, whose widely-used platform and highly-expressive, well-document query language, Cypher, constitute reliable tools for querying graphs. [31]

2.2.3.2 Cypher

Cypher is "*a declarative graph query language that allows for expressive and efficient querying and updating of the graph <...>. Cypher is designed to be simple, yet powerful; highly complicated database queries can be easily expressed, enabling one to focus on your domain, instead of getting lost in database access.*". It is inspired by SQL, reusing part of its syntax (keywords such as *WHERE* and *ORDER BY*) while mixing it with a human-readable, ASCII-art-like query syntax equivalent to the one used in our *PseudoQL* (2.2.2.1). For example, running the query of Fig. 7 upon the data graph of Fig. 8 yields the result of Fig. 9. [32]

```
MATCH (r:Researcher)
OPTIONAL MATCH (r)-[:SUPERVISES]->(s:Student)
WITH r, count(s) AS studentsSupervised
MATCH (r)-[:AUTHORS]->(p1:Publication)
OPTIONAL MATCH (p1)<-[:CITES*]-(p2:Publication)
RETURN r.name, studentsSupervised,
       count(DISTINCT p2) AS citedCount
```

Figure 7: Example *Cypher* query (to be run on data graph of Fig. 8)

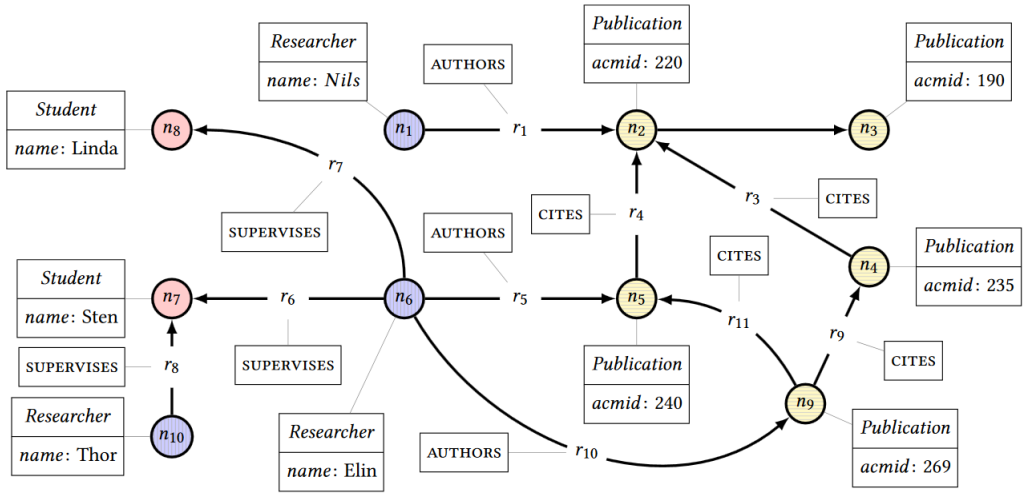

Figure 8: An example data graph showing supervision and citation data for researchers, students and publications

r.name	studentsSupervised	citedCount
Nils	0	3
Elin	2	1

Figure 9: Result of example *Cypher* query (Fig 7)

Query (Fig. 7) **explanation**: This query finds and returns the names of researchers, the number of students they supervise and the number of distinct publications that cite their own. For more details on Cypher querying, you may read the paper this example was taken from or the language's manual, which provides an introduction to the language in addition to full documentation. [33][34]

2.3 Related work

According to recent academic literature, the most interesting related work includes:

- "**Designing Data Protection for GDPR Compliance into IoT Healthcare Systems**", which focuses on the "implications of the General Data Privacy Regulation (GDPR) on the design of an IoT healthcare system": In this paper, researchers break down the regulation into requirements relevant to the technical design of information systems and show how to integrate the Decentralized Labelling Model (DLM) with Fusion/UML methods to produce a GDPR Compliant IoT Architecture. They propose a data labeling model to support access control for privacy-critical patient data and formally define four basic operations: *upload*, *delete* and *download* based on specific use cases of their healthcare application. Compared to our own work, this paper focuses exclusively on the technical requirement implications of sections 1-3 of GDPR's *Chapter III* [16][35]:

  - data controllers must give the data subject read access (1) to their data including any meta-data related to it
  - the system must enable deletion of data (2) and restriction of processing
  - the system functions must preserve the access rights of personal data (3)

  However, personal data requirements (summarized in our work's aforementioned questions (2.1.3) and used in Chapter 3 of our work) relating to where the personal data is stored, whether consent was provided, data usages enabled by consent types and tracking of the personal data, are not considered in this paper.

- "**A Consent and Data Management Model**", a project encompassing a collection of publications and applications aimed at utilizing semantic web technologies to provide a framework for aiding in GDPR Compliance. In this ongoing project, a group of researchers at Trinity College Dublin have to-date carried out extensive analysis of the GDPR and have quantified terms and obligations into the following linked data resources and ontologies [17]:

  - **GDPRtEXT** [36]: an RDF representation of the GDPR's text (at article-paragraph granularity) and a vocabulary of relevant terms and concepts
  - **GConsent** [37]: an OWL2 ontology for representing consent for GDPR compliance
  - **GDPRov** [38]: an ontology for "expressing provenance of consent and data lifecycles with a view towards GDPR compliance"
  - **Data Protection Rules Language** [39]: "An ontology to express data subject rights in relation to data protection regulations."

  Regarding compliance, they have created applications for:

  - **Evaluating GDPR Readiness** [40]: wherein the above mentioned metadata is exploited for evaluating GDPR readiness using an Irish government checklist (a technically broader set of requirements compared to our considered questions') with SPARQL queries. Though, not all questions of the checklist have yet been implemented. [41]
  - **Evaluating Compliance Data** [42]: wherein provenance information is validated using SHACL for ensuring organizations maintain all the required data prior to evaluation of compliance [43]

– **Test-driven approach for GDPR Compliance** [44]: wherein "tests can be generated that check for compliance using constraints gathered from requirements"

Compared to our work, this project attempts to take into account all of the regulation's detailed requirements and provide an automated means for checking compliance through a test-driven approach. Due to this ambitious undertaking's still ongoing and fragmentary nature, more concrete results are yet to be seen.

2.4 iXen

The design of our Remote Patient Monitoring System (3.2) was based on the iXen architecture due to its secure, open and interoperable nature which serves as a representative template for the design of a wide category of systems. Building upon previous work on IoT architecture design and implementation based on SOA and cloud micro-services, iXen is a secure-by-design, highly configurable, expandable and modular architecture that supports the generation of fully customizable applications by re-using devices and services. These re-usable services, implement fundamental functionality and offer a public interface allowing secure connections with other services (including third party ones). [45][46][47]

2.4.1 Users and functional requirements

It supports the following user groups and functional requirements:

- **Systems administrators**: they configure, maintain and monitor the cloud. In addition to their competence for providing cloud services, they are responsible for performing Create, Read, Update, Delete (CRUD) operations on (a) users (e.g. they can register new users to the system and define their access rights) and (b) devices. They are responsible for monitoring system operations at all times (e.g. monitoring user activity).

- **Infrastructure owners**: they subscribe to the cloud for a fee and are granted permission (by the cloud administrator) to register, configure, monitor or remove devices in their possession.

- **Application owners**: they subscribe to the cloud and to a set of devices for a fee. Once subscribed to devices, they can create applications using these devices. iXen provides query mechanisms for selecting devices of interest using device properties such as device type, geographic location, purpose etc. Application development in iXen is supported by flow-based programming web tools. An application is defined by wiring together the outputs of selected devices.

- **Customers**: they subscribe to applications for a fee. iXen provides query mechanisms to users for selecting applications available for subscriptions based on criteria such as purpose and functionality. Customers are only granted access rights to applications.

2.4.2 Architecture

iXen is designed as a composition of groups of autonomous RESTful services communicating with each other over HTTP, as shown in the architecture diagram of Fig. 10. [48]

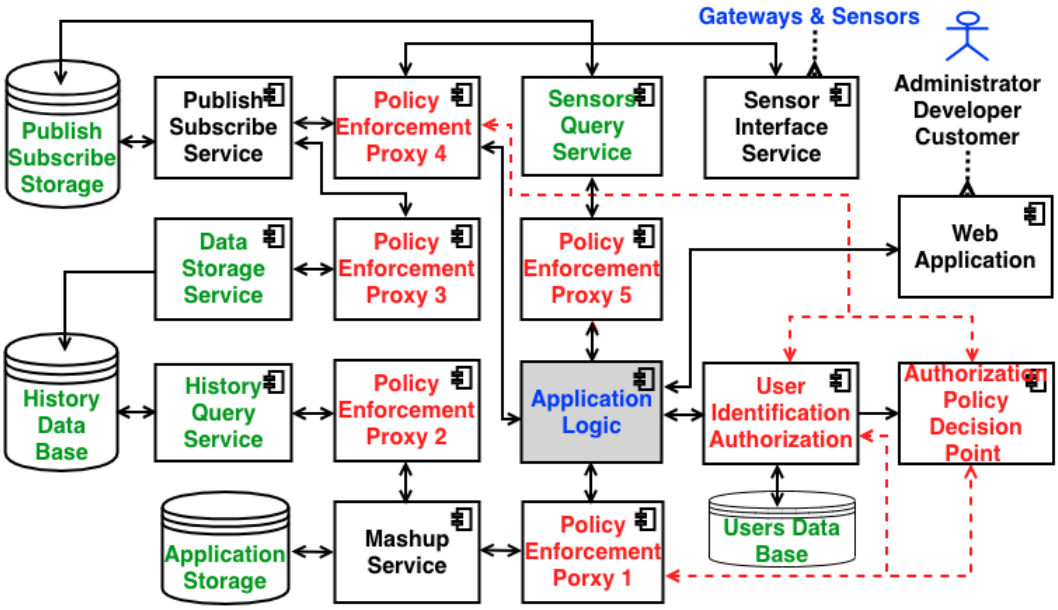

Figure 10: iXen Architecture

In the following chapter, we re-design the iXen architecture as a Remote Patient Monitoring System architecture by taking into account patient monitoring needs and the requirements posed by the 10 GDPR questions and we create UML class diagrams that will be imported into a graph database for creating answer queries for GDPR compliance evaluation.

Chapter 3

Solving the GDPR Compliance evaluation problem

3.1 Approach

Our method's general idea for tackling the problem of evaluating GDPR compliance involves designing a Remote Patient Monitoring System by considering compliance-related personal data requirements and querying it for results. Specifically, our solution is composed of two stages:

1. Creating the class diagram of a Remote Patient Monitoring System that encompasses information taken from the IoT healthcare and cloud micro-service contexts and fundamental GDPR requirements proposed by Neo4j's questions (described in 2.1.3)

2. Querying the class diagram's graph (populating a graph database) for answering the questions, which leads to the creation of data reports for evaluating GDPR compliance

3.2 Designing a Remote Patient Monitoring System

Our Remote Patient Monitoring System is founded on a re-design of the users, functional requirements and underlying architecture of iXen, an IoT system composed of RESTful cloud micro-services leveraging Service Oriented Architecture (SOA) principles with a strong emphasis on security. Every functionality constitutes autonomous services communicating with each other through RESTful interfaces over HTTP. This design follows a "Security by Design" approach, providing the necessary protection of services and data in the cloud and providing services only to authorized users or other services. Key components of this platform include services such as the *Device Interface* service to which patients' devices automatically transmit biometric data on a regular basis, as well as the *Publish-Subscribe* service that implements the doctor-patient subscription and publication mechanisms.

3.2.1 Users and Functional Requirements

Three types of users are supported:

1. **Patients**, who wear heart-monitoring sensor devices that independently send biometric data to a service in the cloud

2. **Doctors**, who have the ability to subscribe to a number of patients, monitor their conditions and access their personal data

3. **Administrators**, who hold the highest degree of access-right privileges, are able to access, change and delete personal data across users and services within the platform.

All users use a Web app interface to access their own data as well as various resources residing on different services, depending on their level of authorization. Four groups of components related to services are supported:

1. **Sensor**: Heart-rate-monitoring IoT devices are connected and continuously transmit data to the *Device Interface* service, which publishes it to the *Publish-Subscribe* service that sends it to subscribers and the *Data Storage* service.

2. **Storage**: The *Pub-Sub DB* (NoSQL database), which is part of the *Publish-Subscribe* service, holds the active devices transmitting data, active subscriptions and current device data (e.g. recently transmitted data). Heart-related health data, as received from devices and older historic data is stored in the *History DB* service (that contains a NoSQL database) through the *Data Storage* Service, whereas, the historic data can be accessed through the *Data History Recovery* service. Information related to user login credentials and authorization, such as roles and permissions, is stored in the *Relational DB*, that is part of the *User Identification – Auth* service.

3. **Application Logic**: The *Application Logic* service acts as the orchestrator in between the other services. Every time a user request is issued through the *Web App*'s interface, this service receives it and dispatches it to the appropriate service.

4. **Security**: Services related to security implement access control mechanisms based on user roles and access policies. On registration, users receive login credentials and an appropriate role related to their access-rights. On login, the *User Identification – Auth* service assigns users an OAuth2.0 token that encodes particular service authorization information by means of XACML, stored in the *Authorization Policy*

Decision Point service. For accessing resources, *Policy Enforcement Proxy* (PEP) services forward a user's OAuth2.0 token to the *User Identification – Auth* service to check for validity. If positive, the steps that follow are illustrated in Fig. 11. Regarding inter-service communication, services securely talk to each other with the help of a security key, referred to as a *Master Key*. In such a case, the *PEP* service stores the master key which is compared with the in-header key of a service's request. This simple mechanism is illustrated in Fig. 12. [49][50]

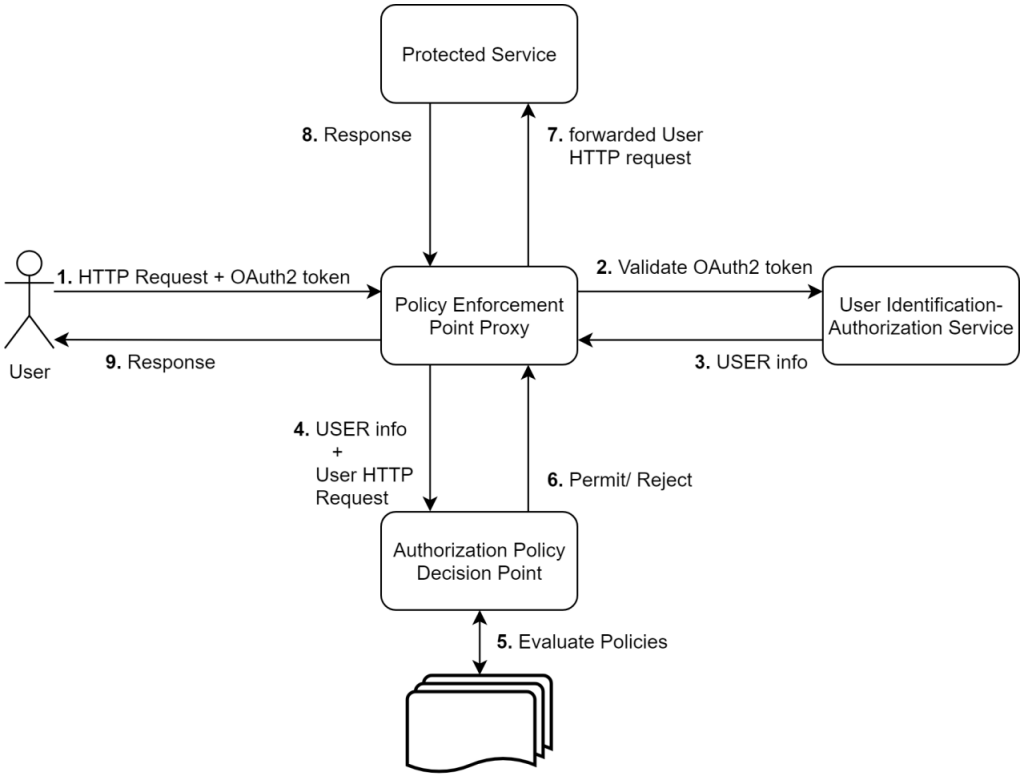

Figure 11: iXen OAuth2.0 security model

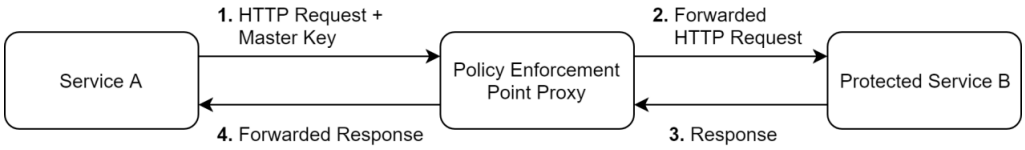

Figure 12: iXen Master Key security model

3.2.2 Architecture

The architecture of our Remote Patient Monitoring System, as illustrated in Fig. 13, is based on a redesign of the iXen architecture (Fig. 10) by including information specific to our users and functional requirements and removing those that are irrelevant to our application, such as application generation using mashups.

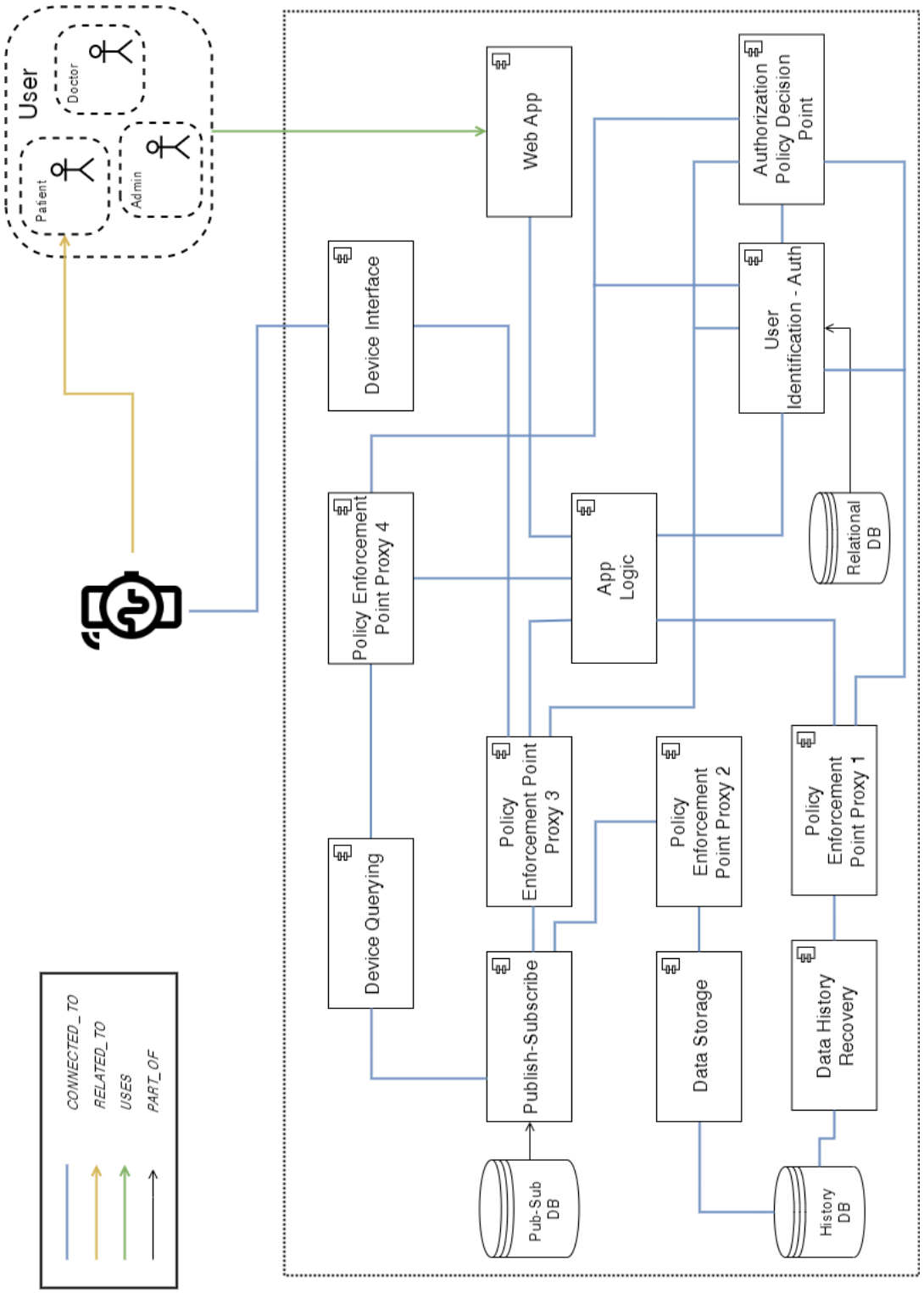

Figure 13: Remote Patient Monitoring System architecture diagram

3.2.3 Incorporating GDPR Compliance requirements

In this section we provide the methodology for incrementally constructing the unified class diagram of our Remote Patient Monitoring System by taking into account the requirements of each and every one of the specific 10 preliminary GDPR Compliance evaluation questions[1], in addition to the users, functional requirements and architecture defined in the previous subsection.

Our platform, as already mentioned, includes user patients wearing sensor devices measuring heart rate data, users that can subscribe to one another and user administrators having full-access privileges over all System information. Furthermore, all users use a web interface through which they can access the services' APIs and can perform a range of operations, depending on their roles.

3.2.3.1 Constructing the System class diagram

In this subsection we take each question and create its corresponding class diagram, leading up to the resulting class diagram of the whole System (Fig. 21).

3.2.3.1.1 0. What are the fundamental data entities?

Analysis: Since our domain of interest revolves around patient monitoring, these data entities are:

- *Person*: the data subjects the personal data belongs to
- *User*: patient, doctor and admin users of the Remote Patient Monitoring System, related to *Person*s
- *Device*: devices worn by user patients for remote health monitoring purposes

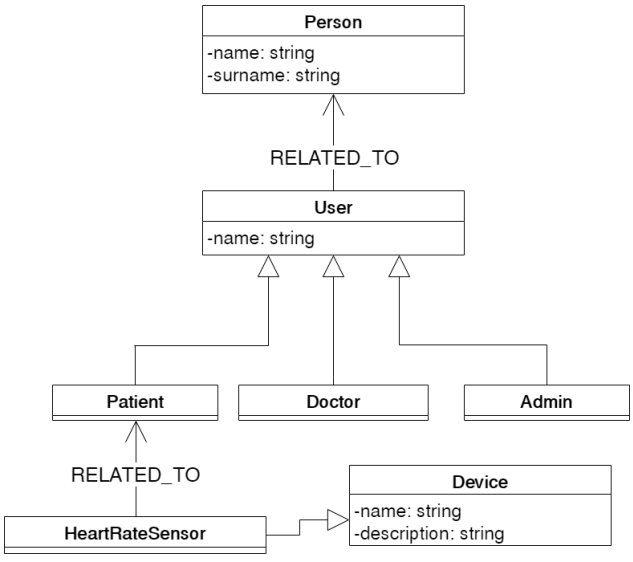

Figure 14: *Q0* Remote Patient Monitoring System class diagram

[1]Descriptions to the questions are provided in 2.1.3

3.2.3.1.2 1. What data do you have?

Analysis: The categories of personal data we have are:

- *UserData*: user data such as username, email and phone
- *SubscriptionData*: user subscription data such as subscriber and subscribees
- *SensorData*: sensor data such as heart rate

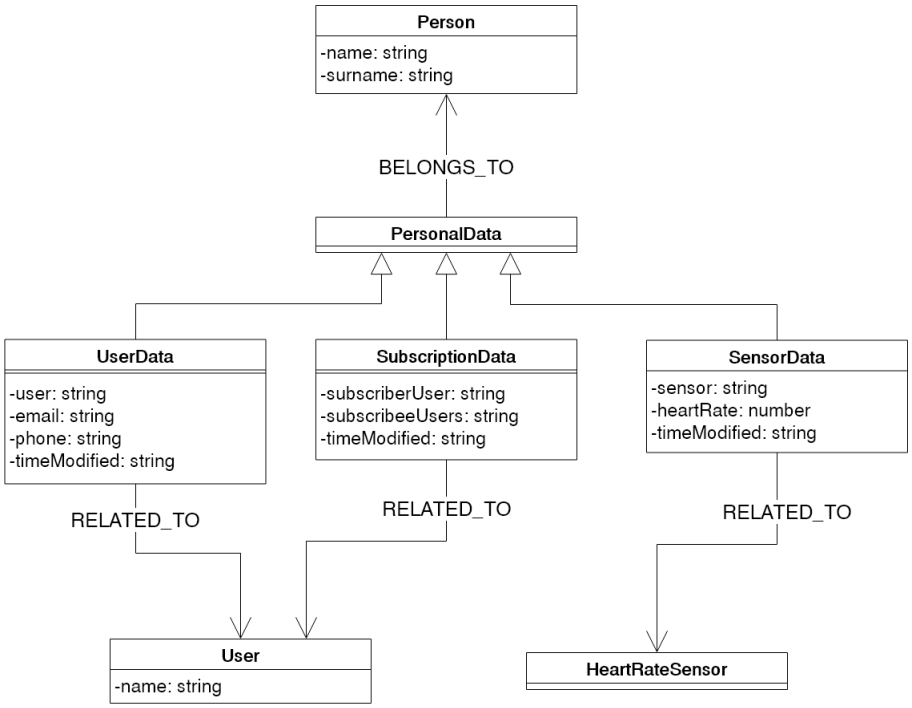

Figure 15: *Q1* Remote Patient Monitoring System class diagram

3.2.3.1.3 2. Where is the data stored?

Analysis: Where the data is stored refers to the categories of database systems that act as storage for the personal data which, according to our functional requirements, are:

- *RelationalDB*: storage database of *UserData*
- *PubSubDB*: storage database of *SubscriptionData*
- *HistoryDB*: storage database of *SensorData*

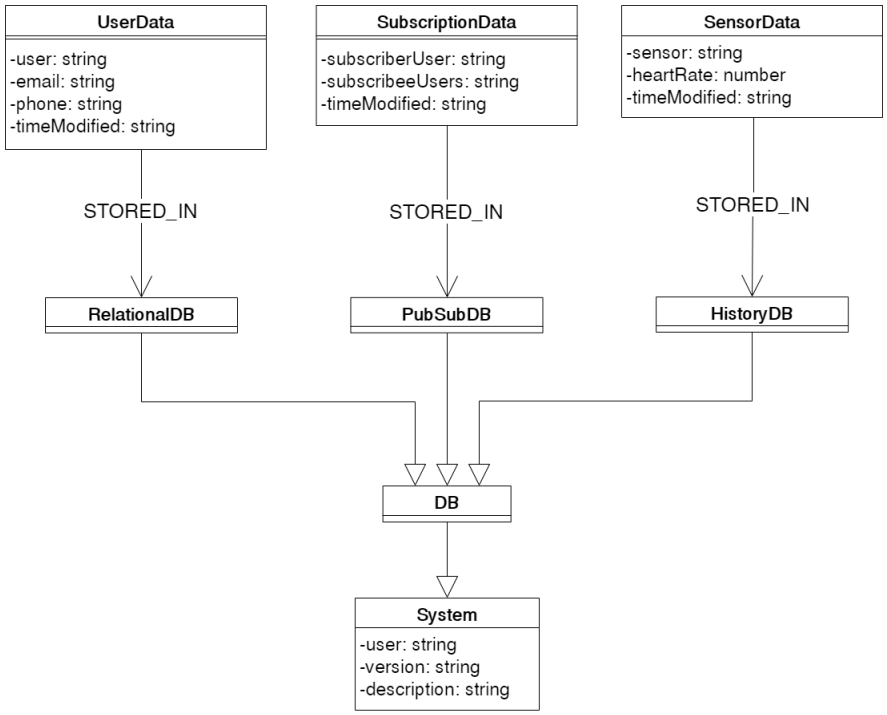

Figure 16: *Q2* Remote Patient Monitoring System class diagram

3.2.3.1.4 3. How and when did you obtain the data?

Analysis: The new entities that are of interest to us within the context of this question's requirements, are:

- *Consent*: a consent event, provided or revoked by a person
- *ConsentType*: the type of a *Consent* (e.g. 'Terms and Conditions')

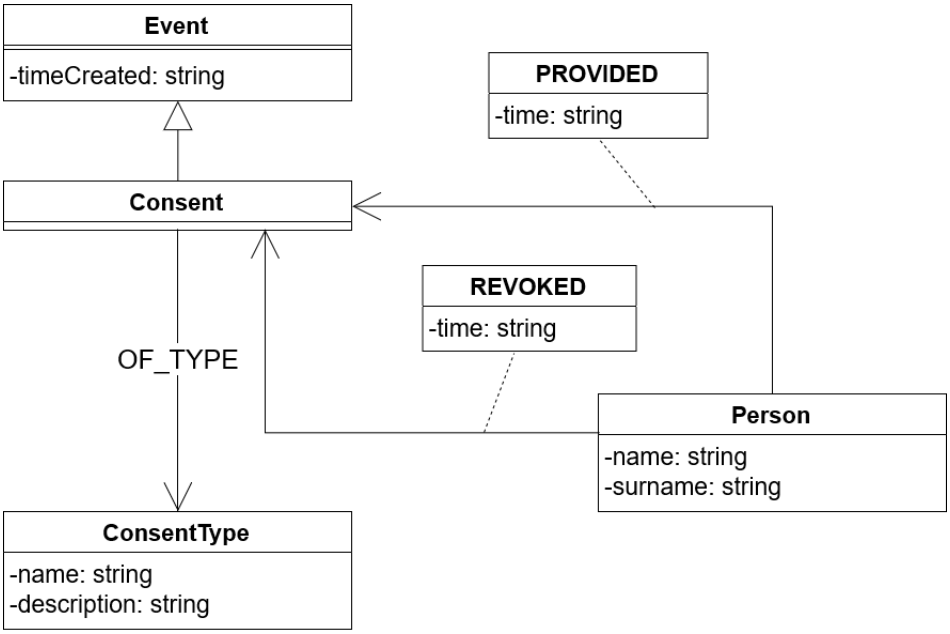

Figure 17: *Q3* Remote Patient Monitoring System class diagram

3.2.3.1.5 4. Why do you have the data?

Analysis: Why we have the personal data introduces two new entities (a) representing the data usage that a consent type enables and (b) the categories of data process events that process personal data. The corresponding entities are:

- *DataUsage*: the purpose of data processing, enabled by a specific *ConsentType*
- *DataProcess*: the event of data processing, involving personal data belonging to a person:
  - *DataAccess*: the event of data access to a *User* from a *WebApp*
  - *DataUpdate*: the event of data updating done from a *WebApp* to a *DB*
  - *DataExport*: the event of exporting data from a *DB* to an *AmbulanceService* (third party first-aid emergency ambulance service)
  - *InternalDataMovement*: the event of an internal data movement involving data exchanged from one *System* to another *System* (where *System* may be either a cloud Service or a DB) [2]

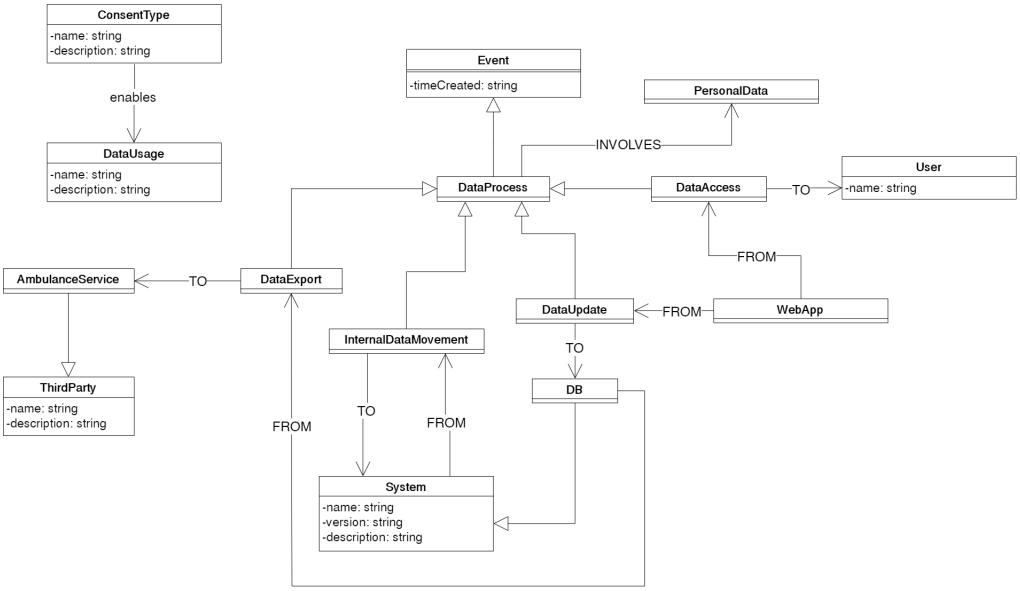

Figure 18: *Q4* Remote Patient Monitoring System class diagram [3]

[2] *System* is the superclass of *DB* and *Service* classes (see the resulting class diagram of Fig. 21) and should not be confused with the Remote Patient Monitoring System that contains it.

[3] *TO* and *FROM* associations indicate the flow of data to and from entities.

3.2.3.1.6 5. Who has access to the data?

Analysis: Data access rights are denoted as 'CAN_ACCESS_DATA_OF' associations between users.

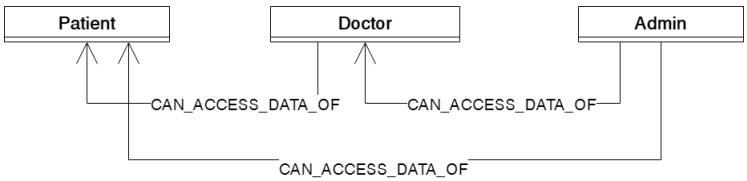

Figure 19: *Q5* Remote Patient Monitoring System class diagram [4]

3.2.3.1.7 6. Do you have permission to use the data? For what purposes?

Analysis: No new GDPR entities are introduced here (the entities that are involved in this question are: *PersonalData*, *Person*, *Consent*, *ConsentType*, *DataUsage*). For more information, see question description (2.1.3).

[4] We assume that a *Doctor* can access the data of one or more *Patient*s, whereas all *Admin*s can access the data of all *Doctor*s and all *Patient*s.

3.2.3.1.8 7. Is the data secure?

Analysis: According to its functional requirements, our Remote Patient Monitoring System supports the following security schemes:

- *RESTServiceSecurity*: The security scheme employed by *User*, *Device* and *Service* entities:
  - *OAuth2*: OAuth2.0 tokens are issued to users upon login for identity, role and permission confirmation
  - *MasterKey*: A secret key for secure inter-service communication
  - *Permission*: A permission for accessing a particular resource
  - *Role*: A collection of permissions, described in its corresponding XACML rule
  - *XACMLRule*: A rule formally describing the services and resources that users can access based on the roles and permissions they hold

[5]We assume that all *Device*s, *Service*s and *User*s of our platform employ REST service security.

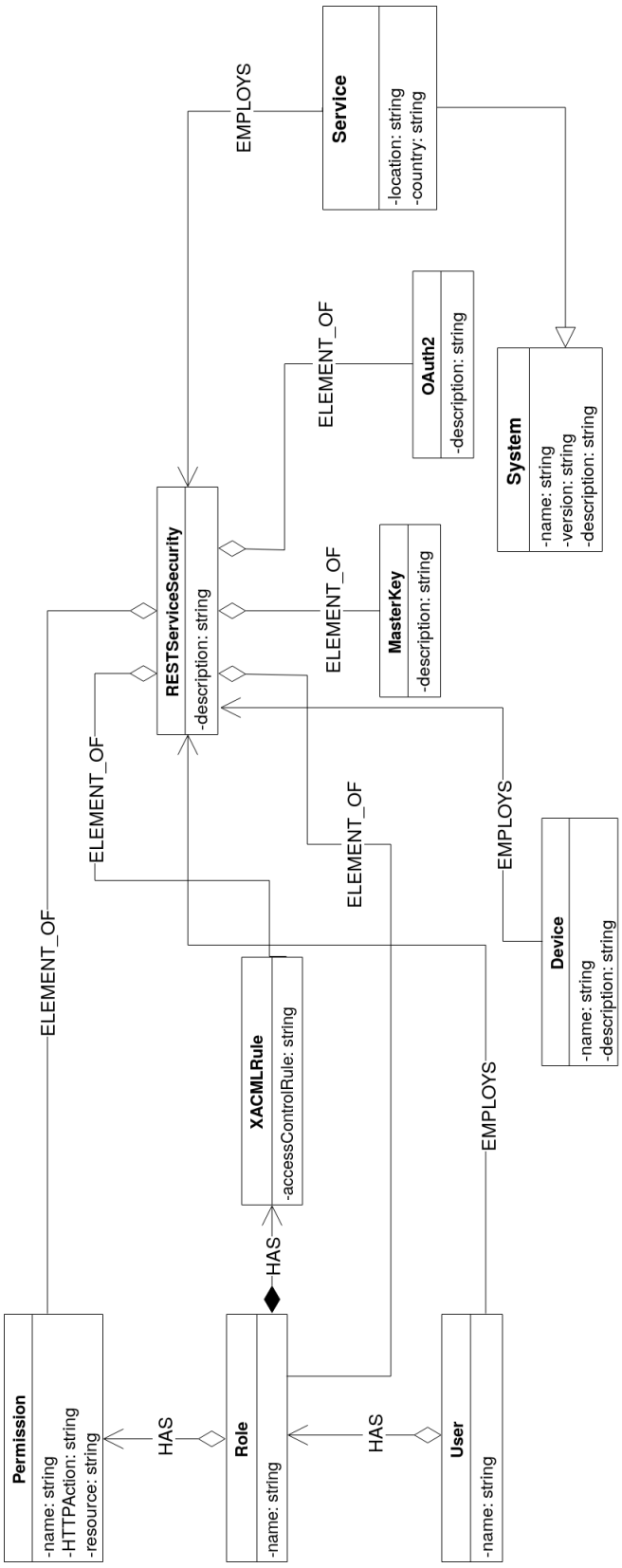

Figure 20: *Q7* Remote Patient Monitoring System class diagram [5]

3.2.3.1.9 8. How does the data travel through your systems?

Analysis: No new GDPR entities are introduced here (the entities that are involved in this question are: *PersonalData*, *Person*, *Event*, *System*, *App*, *User*, *ConsentType*, *DataUsage*). For more information, see question description (2.1.3).

3.2.3.1.10 9. Does the data ever cross international borders?

Analysis: No new GDPR entities are introduced here (the entities that are involved in this question are: *PersonalData*, *Person*, *DataProcess*, *System*, *App*, *User*). For more information, see question description (2.1.3).

3.2.3.1.11 The resulting System class diagram

Combining each separate question's class diagram and the architecture diagram (Fig. 13), we construct the final class diagram of our Remote Patient Monitoring System, as illustrated in Fig. 21. The role of the resulting diagram is twofold: it can be used to (a) answer the GDPR compliance-related questions and (b) implement the system. Therefore, both problems are handled in one step and there is no need to separate the system design process from evaluation of GDPR compliance, since the compliance requirements have been incorporated by design.

[6]The *USES*, PART_OF and CONNECTED_TO associations are taken directly from the architecture diagram.

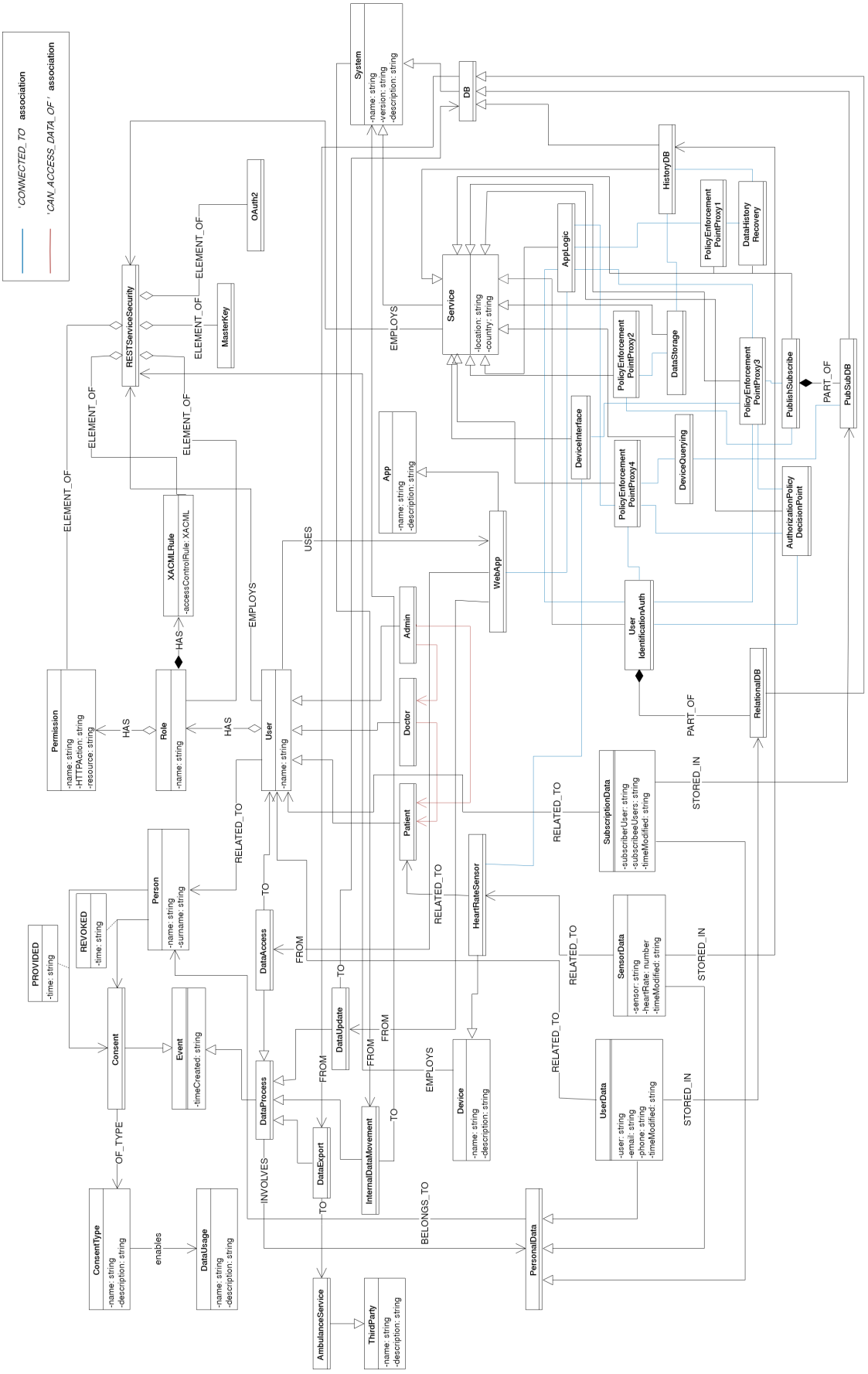

Figure 21: Remote Patient Monitoring System class diagram [6]

3.3 System GDPR Compliance evaluation

Although UML is appropriate for design purposes, it does not inherently support information processing and there are currently no tools for querying the information it represents. The ability for querying is provided by a tool such as a graph database, which we employ in our method. In the following subsection, we provide the algorithms and queries in the *PseudoQL* pseudo-language (that we defined in 2.2.2.1) for importing the class diagram and answering the questions in a property graph database.

3.3.1 Importing the System into a graph database

Prior to answering the questions for evaluating GDPR compliance, we must import the resulting Remote Patient Monitoring System class diagram (Fig. 21) into a graph database. This means that we must create a property graph equivalent to the diagram. To do so, we abide by the *UML-to-property-graph* mapping convention we defined in 9, where classes, associations and attributes are translated into property graph nodes, relationships and properties.

Creating the equivalent diagram property graph involves running the *PseudoQL* query 23 that is presented in the Appendix (A). Fig. 22 illustrates the resulting property graph.

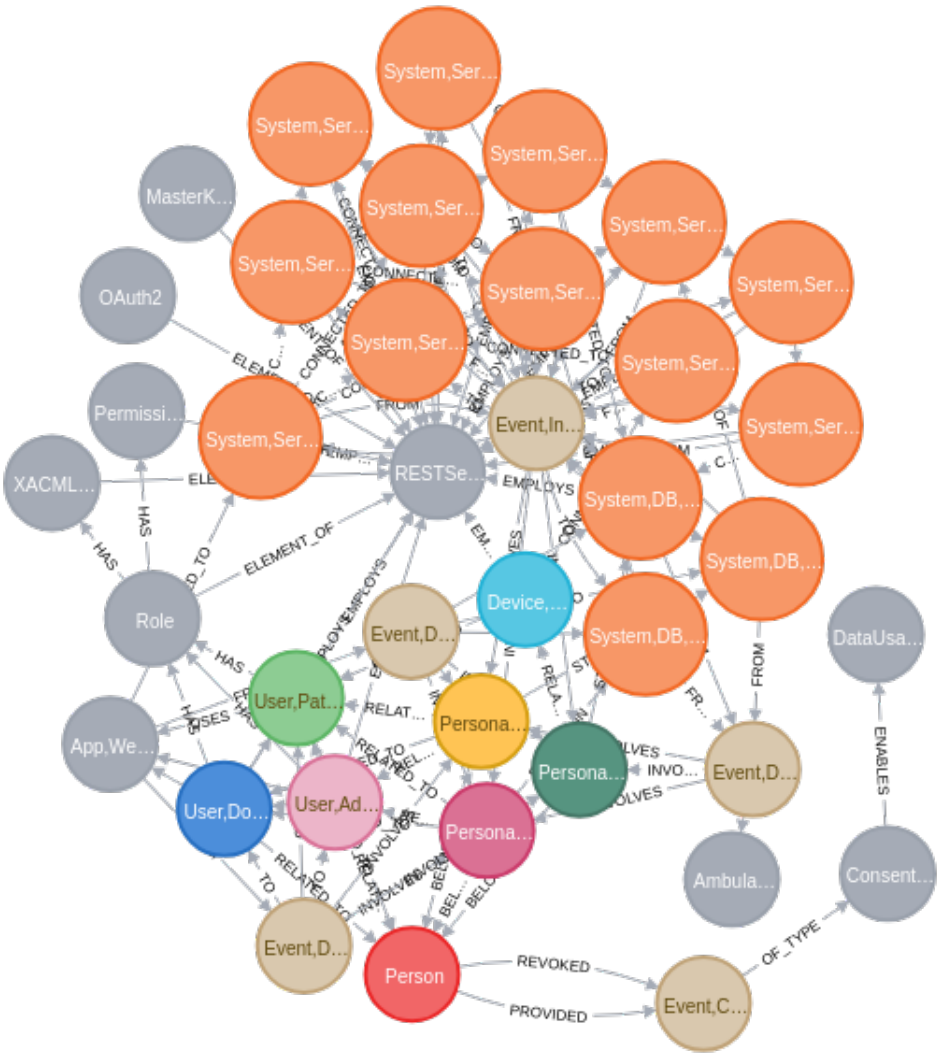

Figure 22: Remote Patient Monitoring System graph

3.3.2 Answering the compliance questions

For each of the following questions we provide a description, the algorithm of the answer query, the actual answer query in the *PseudoQL* query language, the tabular data report results it yields and brief comments on them. These reports can be more formally analyzed by a human expert such as a Data Protection Officer or equivalent compliance authority, for evaluating GDPR compliance. Although, instantiated data is required for determining compliance, the following result tables provide a template of how real-data reports of an instantiated platform will look like.

3.3.2.1 Answer 0. What are the fundamental data entities?

Description: Query all fundamental data entity nodes.

Query explanation:

1. Find nodes whose labels include *Device*, *Person* or *User*
2. Return them

```
SELECT (e)
WHERE e:Device or e:Person or e:User
RETURN LABELS_OF(e) as type,
        e as fdDataEntity
```

Listing 3: *PseudoQL* Q0 answer query

type	fdDataEntity
[Device,HeartRateSensor]	{name:string,description:string}
[Person]	{name:string,surname:string}
[User,Patient]	{name:string}
[User,Doctor]	{name:string}
[User,Admin]	{name:string}

Figure 23: *PseudoQL* Q0 answer query result

Query result comments: This answer yields rows containing each registered fundamental data entity (e.g. the patient user).

3.3.2.2 Answer 1. What data do you have?

Description: Query all personal data nodes and the person nodes they belong to.

Query explanation:

1. Find personal data nodes and person nodes they belong to
2. Return them

```
SELECT (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person
```

Listing 4: *PseudoQL* Q1 answer query

pdType	personalData	person
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}

Figure 24: *PseudoQL* Q1 answer query result

Query result comments: This answer yields rows containing the personal data held within the Remote Patient Monitoring System (e.g. *SensorData*), its contents and the information of the data subject it belongs to.

3.3.2.3 Answer 2. Where is the data stored?

Description: Query all DB nodes where personal data is stored in.

Query explanation:

1. Find personal data nodes and person nodes they belong to
2. Find the DB nodes the personal data is stored in
3. Find the service nodes the DB nodes are part of
4. Return the personal data nodes, their corresponding person nodes, the DB nodes and location properties

```
SELECT (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
OPTIONAL SELECT (pd)-[:STORED_IN]->(db:DB)
OPTIONAL SELECT (db)-[:PART_OF]->(dbS:Service)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS_OF(db) as DBType,
        db as DB,
        CASE
                WHEN EXISTS((db)-[:PART_OF]->(dbS)) THEN
                        dbS.location
                ELSE
                        db.location
        END as location,
        CASE
                WHEN EXISTS((db)-[:PART_OF]->(dbS)) THEN
                        dbS.country
                ELSE
                        db.country
        END as country
```

Listing 5: *PseudoQL* Q2 answer query

pdType	personalData	person	DBType	DB	location	country
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[System,DB,PubSubDB]	{name:string,description:string,version:string}	string	string
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[System,DB,Service,HistoryDB]	{name:string,country:string,description:string,location:string,version:string}	string	string
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[System,DB,RelationalDB]	{name:string,description:string,version:string}	string	string

Figure 25: *PseudoQL* Q2 answer query result

Query result comments: This answer yields rows containing the database system in which each personal data entity is stored (e.g. *RelationalDB*), its information (e.g. version) and the physical location of the database system is located at.

3.3.2.4 Answer 3. How and when did you obtain the data?

Description: Query all personal data nodes, the person nodes they belong to, consent nodes they provided and related consent type nodes and the time at which they were provided.

Query explanation:

1. Find personal data nodes, data entity nodes they are related to and person nodes they belong to
2. Find the consent nodes provided by the person nodes and their consent type nodes
3. Return the personal data nodes, the person nodes, the provided consent nodes, the consent type nodes and the time the consent was provided

```
SELECT (de)<-[:RELATED_TO]-(pd:PersonalData)
        -[:BELONGS_TO]->(p:Person)
OPTIONAL SELECT (p)-[prov:PROVIDED]->(c:Consent)
        -[:OF_TYPE]->(cT:ConsentType)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        de as fdDataEntity,
        c as consent,
        cT as consentType,
        prov.time as timeConsentProvided
```

Listing 6: *PseudoQL* Q3 answer query

pdType	personalData	person	fdDataEntity	consent	consentType	timeConsentProvided
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	{name:string}	{timeCreated:string}	{name:string,description:string}	string
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	{name:string}	{timeCreated:string}	{name:string,description:string}	string
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	{name:string}	{timeCreated:string}	{name:string,description:string}	string
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	{name:string,description:string}	{timeCreated:string}	{name:string,description:string}	string
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	{name:string}	{timeCreated:string}	{name:string,description:string}	string
...	...	...	...	...	...	...

Figure 26: *PseudoQL* Q3 answer query result

Query result comments: This answer yields rows containing the information of the user or device fundamental data entity each personal data entity is related to and information of the consent that the data owner has provided (e.g. time of consent creation, time of consent provision, consent type).

3.3.2.5 Answer 4. Why do you have the data?

Description: Query all personal data nodes, the person nodes they belong to, consent nodes, related consent type nodes and the data usage nodes they enable. Also, query the data process nodes that involve the personal data nodes and the nodes to and from which they exchange data.

Query explanation:

1. Find personal data nodes and person nodes they belong to
2. Find the consent nodes, provided by the person nodes and consent type nodes they are related to
3. Find the data usage nodes enabled by the consent type nodes
4. Find the data process nodes involving the personal data nodes
5. Find the nodes to and from which the data process nodes send and receive data
6. Return the personal data nodes, the person nodes, the data usage nodes, the data process nodes and the nodes to and from which the latter send and receive data

```
SELECT (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
SELECT (p)-[:PROVIDED]->(:Consent)-[:OF_TYPE]->(cT:ConsentType)
SELECT (cT)-[:ENABLES]->(du:DataUsage)
OPTIONAL SELECT (pd)<-[:INVOLVES]-(dp:DataProcess)
OPTIONAL SELECT (f)-[:FROM]->(dp)-[:TO]->(t)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        COLLECT(du) as dataUsages,
        LABELS_OF(dp) as dpType,
        dp as dataProcess,
        [LABELS_OF(f), f] as from,
        [LABELS_OF(t), t] as to
```

Listing 7: *PseudoQL* Q4 answer query

Query result comments: This answer yields rows containing information about the data usages of each personal data entity (enabled by the consent provided by its owner), information of events that involve it by means of data processing and the entities (e.g. apps, *System*s, *User*s, third parties) to and from which it travels (in the context of each data processing event).

personalData	person	dataUsages	dpType	dataProcess	from	to
{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[{name:string,description:string}]	[Event,DataProcess,DataAccess]	{timeCreated:string}	[[App,WebApp],{name:string,description:string}]	[[User,Doctor],{name:string}]
{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[{name:string,description:string}]	[Event,InternalDataMovement,DataProcess]	{timeCreated:string}	[[System,Service,PEPProxy3],{name:string,country:string,description:string,location:string,version:string}]	[[System,Service,PublishSubscribe],{name:string,country:string,description:string,location:string,version:string}]
{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[{name:string,description:string}]	[Event,DataProcess,DataAccess]	{timeCreated:string}	[[App,WebApp],{name:string,description:string}]	[[User,Doctor],{name:string}]
{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[{name:string,description:string}]	[Event,DataProcess,DataExport]	{timeCreated:string}	[[System,DB,RelationalDB],{name:string,description:string,version:string}]	[[AmbulanceService,ThirdParty],{name:string,description:string}]
{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[{name:string,description:string}]	[Event,DataProcess,DataUpdate]	{timeCreated:string}	[[App,WebApp],{name:string,description:string}]	[[System,DB,PubSubDB],{name:string,description:string,version:string}]
...	...	...	...	...	...	...

Figure 27: *PseudoQL* Q4 answer query result

3.3.2.6 Answer 5. Who has access to the data?

Description: Query all personal data nodes, the person nodes they belong to, the user nodes (data owners) related to the person nodes, the DB nodes they are linked to (denoting storage) and the non-owner user nodes that can access the owner's data.

Query explanation:

1. Find personal data nodes, the person nodes they belong to, the user nodes related to the person nodes (personal data owners) and the DB nodes the personal data is stored in
2. Find the user nodes that can access the data of the owner user nodes
3. Return the personal data nodes, the person nodes, data usage nodes, the DB nodes, the user owner nodes and the other user nodes that can access the personal data

```
SELECT (pdOwnerUser:User)-[:RELATED_TO]->(p:Person)
        <-[:BELONGS_TO]-(pd:PersonalData)-[:STORED_IN]->(db:DB)
OPTIONAL SELECT (otherUser:User)-[CAN_ACCESS_DATA_OF]->(pdOwnerUser)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS_OF(db) as DBType,
        db as DB,
        pdOwnerUser,
        COLLECT(otherUser) as otherUsersWAccess
```

Listing 8: *PseudoQL* Q5 answer query

Query result comments: This answer yields rows containing the access information for each personal data entity, meaning the owner and all other users that can access it.

pdType	personalData	person	DBType	DB	pdOwnerUser	otherUsersWAccess
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[System,DB,PubSubDB]	{name:string, description:string,version:string}	{name:string}	[]
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[System,DB,PubSubDB]	{name:string, description:string,version:string}	{name:string}	[{name:string}]
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[System,DB,Service,HistoryDB]	{name:string, country:string ,description:string,location:string,version:string}	{name:string}	[{name:string}]
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[System,DB,RelationalDB]	{name:string, description:string,version:string}	{name:string}	[{name:string}, {name:string}]
...	...	...	...	...	...	...

Figure 28: *PseudoQL* Q5 answer query result

3.3.2.7 Answer 6. Do you have permission to use the data? For what purposes?

Description: Query all personal data nodes, the person nodes they belong to, the DB nodes they are linked to (denoting storage) and whether the person nodes have provided and not revoked a consent node for its data usage.

Query explanation:

1. Find personal data nodes and the person nodes they belong to
2. Find the consent nodes provided or revoked by the person nodes, the related consent type nodes and the data usage nodes they enable
3. Return the personal data nodes, the person nodes, the data usage nodes, and a boolean value of whether there exists a provided and not a revoked relationship between the person nodes and the consent nodes (as denoting permission to use the data)

```
SELECT (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
OPTIONAL SELECT (p)-->(c:Consent)
        -[:OF_TYPE]->(cT:ConsentType)
        -[:ENABLES]->(du:DataUsage)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        COLLECT(du) as dataUsagesEnabledByPerson,
        EXISTS((p)-[:PROVIDED]->(c)-[:OF_TYPE]->(cT)
                        -[:ENABLES]->(du)) AND
                NOT EXISTS((p)-[:REVOKED]->(c)-[:OF_TYPE]->(cT)
                        -[:ENABLES]->(du))
                                as permissionToUseData
```

Listing 9: *PseudoQL* Q6 answer query

personalData	person	dataUsagesEnabledByPerson	permissionToUseData
{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[{name:string,description:string},{name:string,description:string}]	false
{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[{name:string,description:string},{name:string,description:string}]	false
...	...	...	...

Figure 29: *PseudoQL* Q6 answer query result

Query result comments: This answer yields rows containing information about the data usages enabled by the owners of each personal data entity and a boolean value denoting whether the data processors holding the data have permission to use it (which depends on whether consent for related data usages has been provided and not revoked). This answer may be a determinant for non-compliance if the permissionToUseData evaluates to false too many times.

3.3.2.8 Answer 7. Is the data secure?

Description: Query all personal data nodes, the DB nodes they are linked to (denoting storage), the user or device fundamental data entity nodes and whether or not they employ REST Service Security.

Query explanation:

1. Find personal data nodes, the person nodes they belong to and the DB nodes the personal data is stored in
2. Find the fundamental data entity nodes related to the person nodes
3. Find the service nodes the DB nodes are part of
4. Find the REST Service Security node and its security element nodes
5. Return the personal data nodes, the person nodes, the DB nodes, a boolean value of whether the fundamental data entity nodes and the db nodes (or the service the db is part of) employ REST Service Security and the REST Service Security node and its security element nodes

```
SELECT (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
        -[:STORED_IN]->(db:DB)
OPTIONAL SELECT (fdEntity)-[:RELATED_TO]->(p:Person)
OPTIONAL SELECT (db)-[:PART_OF]->(serviceOfdb:Service)
OPTIONAL SELECT (restServiceSec:RESTServiceSecurity)
        <-[:ELEMENT_OF]-(secElem)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS_OF(db) as DBType,
        db as DB,
        CASE
                WHEN "Service" IN LABELS_OF(db) THEN
                        EXISTS((fdEntity)
                                -[:EMPLOYS]->(restServiceSec)) AND
                        EXISTS((db)-[:EMPLOYS]->(restServiceSec))
                ELSE
                        EXISTS((fdEntity)
                                -[:EMPLOYS]->(restServiceSec)) AND
                        EXISTS((serviceOfdb)
                                -[:EMPLOYS]->(restServiceSec))
        END as isDataSecure,
        [LABELS_OF(restServiceSec), restServiceSec]
                as securityType,
        COLLECT([LABELS_OF(secElem), secElem])
                as securityElements
```

Listing 10: *PseudoQL* Q7 answer query

Query result comments: This answer yields rows containing information about the security type available in our Platform (REST Service Security), its elements (e.g. OAuth2.0)

pdType	personalData	person	DBType	DB	isDataSecure	securityType	securityElements
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[System,DB,PubSubDB]	{name:string,description:string,version:string}	true	[[RESTServiceSecurity],{description:string}]	[[[OAuth2],{description:string}],[[XACMLRule],{accessControlRule:string}],[[MasterKeySecurity],{description:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Role],{name:string}],[[OAuth2],{description:string}],[[XACMLRule],{accessControlRule:string}],[[MasterKeySecurity],{description:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Role],{name:string}]]
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[System,DB,Service,HistoryDB]	{name:string,country:string,description:string,location:string,version:string}	true	[[RESTServiceSecurity],{description:string}]	[[[OAuth2],{description:string}],[[XACMLRule],{accessControlRule:string}],[[MasterKeySecurity],{description:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Role],{name:string}],[[OAuth2],{description:string}],[[XACMLRule],{accessControlRule:string}],[[MasterKeySecurity],{description:string}],[[Permission],{name:string,resource:string,HTTPAction:string}],[[Role],{name:string}]]
...	...	...	...	...	...	...	...

Figure 30: *PseudoQL* Q7 answer query result

and a boolean value denoting whether each personal data entity is secure (which depends on whether the fundamental entity it is related to, such as Users and Devices, and the System it is stored in both employ REST Service Security). Since all Users, Devices and Systems of our platform employ REST Service Security by design, the boolean value is true for every personal data entity. This answer may be a determinant for non-compliance if a human expert, such as a Data Protection Officer deems the security elements involved with each personal data entity as inadequate for fullfiling GDPR security demands.

3.3.2.9 Answer 8. How does the data travel through your systems?

Description: Query all event nodes, the personal data they involve and the nodes (systems, users, apps, etc.) they exchange it with.

Query explanation:

1. Find personal data nodes, the person nodes they belong to and the events nodes the person nodes are connected with, where the events nodes are either consent nodes or data process nodes involving the personal data nodes
2. If the event nodes are data process nodes: find the nodes to and from which the event nodes send and receive data, whereas if the event nodes are consent nodes: find their corresponding consent type nodes and the data usage nodes they enable
3. Return the personal data nodes, the person nodes, a custom description of the event and times of creation, provision or revocation (depending on the type of the event nodes)

```
SELECT (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
SELECT (e:Event)
WHERE e:Consent OR (e:DataProcess AND
        (pd)<-[:INVOLVES]-(e))
OPTIONAL SELECT (e)-[consent_rel]-(p)
OPTIONAL SELECT (t)-[:TO]-(e)-[:FROM]-(f)
OPTIONAL SELECT (e)-[:OF_TYPE]->(cT:ConsentType)
        -[:ENABLES]->(du:DataUsage)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS_OF(e) as eventType,
        CASE
                WHEN e:Consent THEN
                        "OF_TYPE: "+cT.name+" ("+LABELS_OF(cT)[0]+
                                ") | FOR: "+du.name+" ("+
                                        LABELS_OF(du)[0]+")"
                ELSE
                        "FROM: "+f.name+" ("+LABELS_OF(f)[0]+
                                ") | TO: "+t.name+" ("+
                                        LABELS_OF(t)[0]+")"
        END as eventDescription,
        CASE
                WHEN consent_rel IS NOT null THEN
                        consent_rel.time+" ("+TYPE(consent_rel)+")"
                ELSE
                        e.timeCreated
        END as time
ORDER BY time DESC
```

Listing 11: *PseudoQL* Q8 answer query

pdType	personalData	person	eventType	eventDescription	time
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[Event,Consent]	OF_TYPE: string (ConsentType) | FOR: string (DataUsage)	string (REVOKED)
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[Event,Consent]	OF_TYPE: string (ConsentType) | FOR: string (DataUsage)	string (PROVIDED)
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[Event,Consent]	OF_TYPE: string (ConsentType) | FOR: string (DataUsage)	string (PROVIDED)
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[Event,Consent]	OF_TYPE: string (ConsentType) | FOR: string (DataUsage)	string (PROVIDED)
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[Event,InternalDataMovement,DataProcess]	FROM: string (System) | TO: string (System)	string
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[Event,InternalDataMovement,DataProcess]	FROM: string (System) | TO: string (System)	string
[PersonalData,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[Event,InternalDataMovement,DataProcess]	FROM: string (System) | TO: string (System)	string
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[Event,DataProcess,DataAccess]	FROM: string (App) | TO: string (User)	string
[PersonalData,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[Event,DataProcess,DataUpdate]	FROM: string (App) | TO: string (System)	string
[PersonalData,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[Event,DataProcess,DataExport]	FROM: string (System) | TO: string (AmbulanceService)	string
...	...	...	...	...	...

Figure 31: *PseudoQL* Q8 answer query result

Query result comments: This answer yields rows containing descriptions of each *Consent* or *DataProcess* event related to each personal data entity (such as consent types and data usages (for *Consent* events) and apps, users, systems and third parties to and from which the data was exchanged (for *DataProcess* events) and time related to each event.

3.3.2.10 Answer 9. Does the data ever cross international borders?

Description: Query the data process event nodes, the personal data nodes they involve and the nodes (systems, users, apps, etc.) they exchange it with.

Query explanation:

1. Find personal data nodes, the person nodes they belong to and data process events involving the personal data nodes
2. Find the nodes to and from which the data process nodes send and receive data
3. Find the service nodes the 'to' entity nodes are part of
4. Find the service nodes the 'from' entity nodes are part of
5. Return the personal data nodes, the person nodes, a description of the event, the 'from' node countries, the 'to' node countries and the times of creation of the data process nodes

```
SELECT (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
        <-[:INVOLVES]-(dp:DataProcess)
OPTIONAL SELECT (t)-[:TO]-(dp)-[:FROM]-(f)
OPTIONAL SELECT (f)-[:PART_OF]->(serviceOfFromDB:Service)
OPTIONAL SELECT (t)-[:PART_OF]->(serviceOfToDB:Service)
RETURN LABELS_OF(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS_OF(dp) as dataProcessType,
        "FROM: "+f.name+" ("+LABELS_OF(f)[0]+")| TO: "+
                t.name+" ("+LABELS_OF(t)[0]+")"
                        as eventDescription,
        CASE
                WHEN NOT EXISTS((f)
                        -[:PART_OF]->(serviceOfFromDB)) THEN
                                f.country
                ELSE
                        serviceOfFromDB.country
        END as fromEntity_country,
        CASE
                WHEN NOT EXISTS((t)
                        -[:PART_OF]->(serviceOfToDB:Service)) THEN
                                t.country
                ELSE
                        serviceOfToDB.country
        END as toEntity_country,
        dp.timeCreated as time
ORDER BY time DESC
```

Listing 12: *PseudoQL* Q9 answer query

Query result comments: This answer yields rows containing concatenated descriptions of each *DataProcess* event related to each personal data entity and the country of the

pdType	personalData	person	dataProcessType	eventDescription	fromEntity_country	toEntity_country	time
[Personal-Data,SubscriptionData]	{subscriberUser:string,subscribeeUsers:list_of_strings,timeModified:string}	{name:string,surname:string}	[Event,DataProcess,DataUpdate]	FROM: string (App)| TO: string (System)	null	string	string
[Personal-Data,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[Event,DataProcess,DataAccess]	FROM: string (App)| TO: string (User)	null	null	string
[Personal-Data,SensorData]	{heartRate:number,timeModified:string,sensor:string}	{name:string,surname:string}	[Event,InternalDataMovement,DataProcess]	FROM: string (System)| TO: string (System)	string	string	string
[Personal-Data,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[Event,DataProcess,DataUpdate]	FROM: string (App)| TO: string (System)	null	string	string
[Personal-Data,UserData]	{phone:string,timeModified:string,user:string,email:string}	{name:string,surname:string}	[Event,InternalDataMovement,DataProcess]	FROM: string (System)| TO: string (System)	string	string	string
...	...	...	...	...	...	...	...

Figure 32: *PseudoQL* Q9 answer query result

entity from which the data derived and of the entity to which it has traveled (if such information does not exist for the given entities, *null* is returned). This answer may be of special interest to a human expert, such as a regulator, for inspecting the countries from and to which personal data entities may travel (e.g. whether EU borders were crossed).

3.4 Demo

In this section, we demonstrate our method on an example instance of our Remote Patient Monitoring System using the Neo4j graph database and the *Cypher* querying language. This System is based off of the instantiated architecture illustrated in Fig. 33, consisting of FIWARE services (running on the OpenStack infrastructure), including dummy data for representing the properties of users, personal data, consents, data usages, and all the rest of the instance's components. [51][52]

Creating the equivalent property graph, as illustrated in Fig. 34, is accomplished by first running the *Cypher* parameter-related query 24 which creates the dummy data (by binding key-value pairs of data to parameters). After that query has ran, we run the *Cypher* query 25 that creates all the necessary nodes and relationships. Both queries are presented in the Appendix (A).[7] Finally, we run the *Cypher* answer queries presented in the following subsections and inspect their results.

Assumptions: For the purposes of limiting the complexity of our graph and providing a clear view of our method without getting lost in data specifics, we have made the following assumptions about the information held in our instantiated system:

- There are two consent types:
  1. Terms & Conditions
  2. Patient Emergency Condition
- There are four types of data processing events:
  1. Internal data movement
  2. Data Access
  3. Data Update
  4. Data Export
- Our instance only holds information about all the users, devices and services which employ REST Service Security. For the sake of simplicity, role, permission and XACML information data was excluded from this demo.[8]

[7] Queries 24 and 25 and the following answer queries can be run and inspected in any running sandbox, cloud, desktop or containerized deployment of the Neo4j database.

[8] For matters of convenience, we bound the instance's data (properties) to *Cypher* parameters prior to executing our main import query. To explore the specific data used to populate our nodes and relationships, you may inspect the :*param* query JSON objects (query 24)

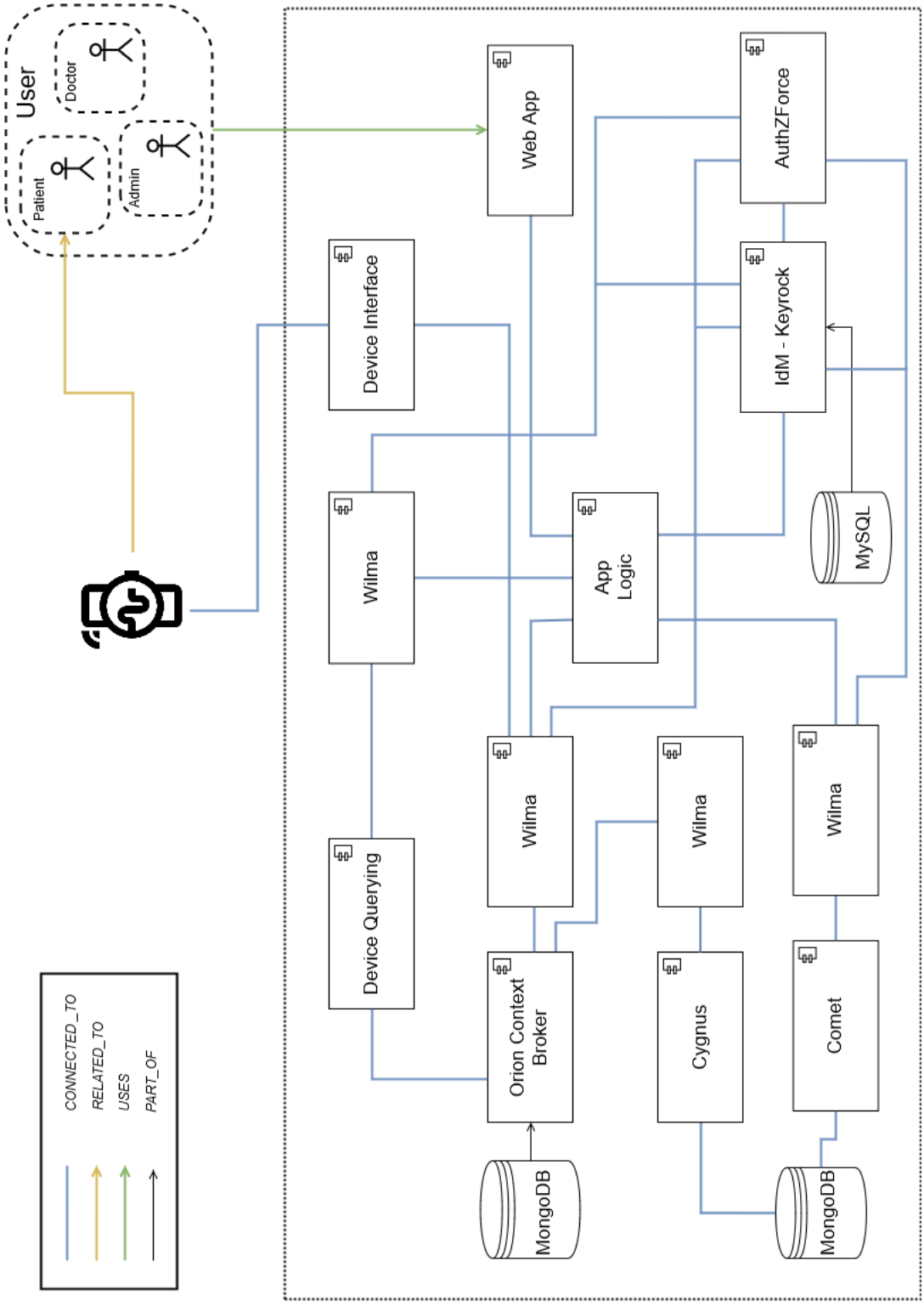

Figure 33: Instantiated Remote Patient Monitoring System architecture diagram

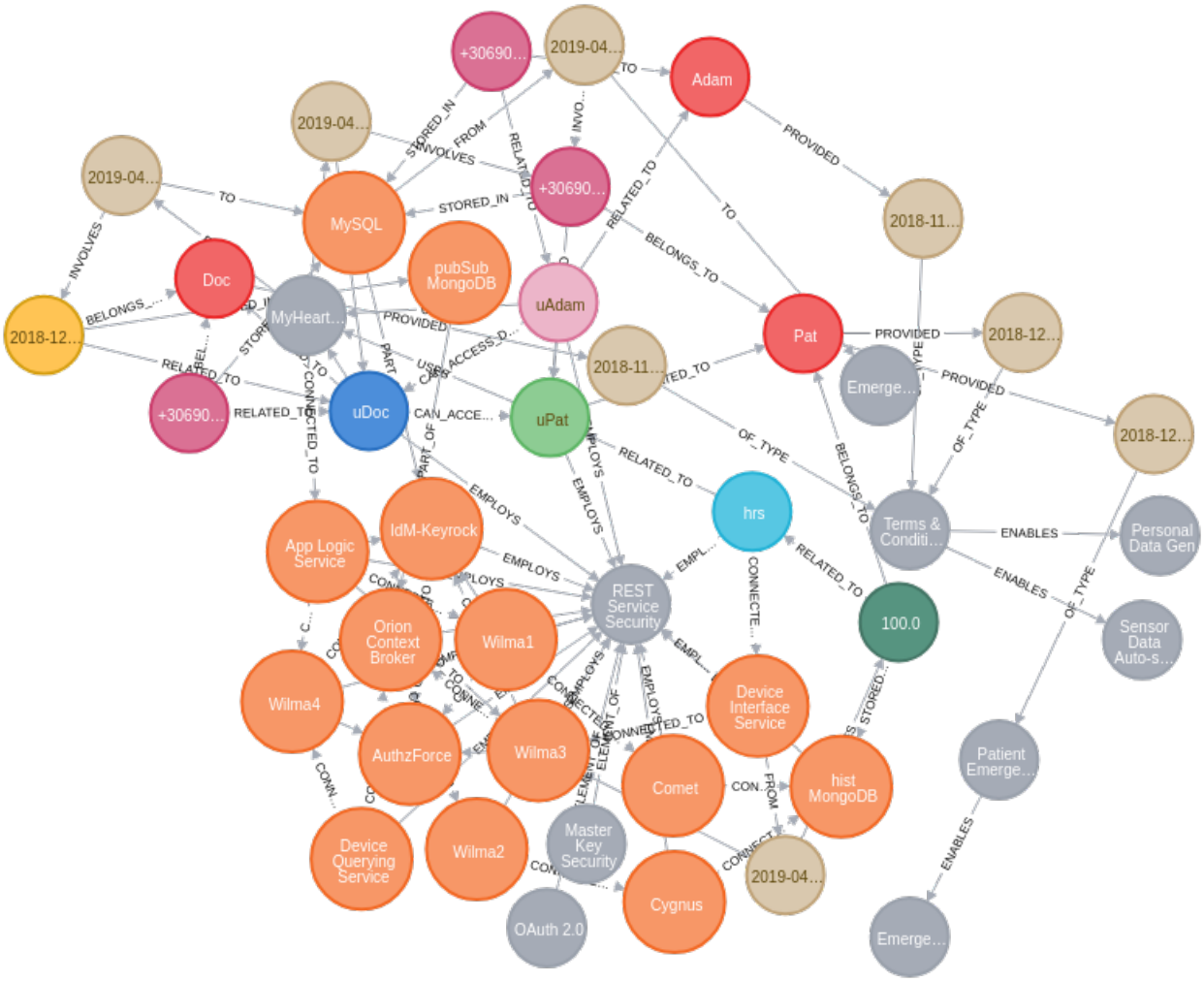

Figure 34: Instantiated Remote Patient Monitoring System graph

3.4.1 Answering the compliance questions

These queries were based off of the *PseudoQL* answer queries presented in 3.3.2. *Cypher*'s MATCH clause is equivalent to *PseudoQL*'s *SELECT* command, whereas self-explanatory functions and expressions such as as *EXISTS*, *CASE*, etc. are mostly the same. [53]

3.4.1.1 Answer 0. What are the fundamental data entities?

```
MATCH (e)
WHERE e:Device or e:Person or e:User
RETURN LABELS(e) as type,
        e as fdDataEntity
```

Listing 13: *Cypher* Q0 answer query

type	fdDataEntity
[Device,HeartRateSensor]	{name:hrs,description:A sensor device that tracks your heart rate.}
[Person]	{name:Pat,surname:Williams}
[Person]	{name:Doc,surname:Brown}
[Person]	{name:Adam,surname:Smith}
[User,Patient]	{name:uPat}
[User,Doctor]	{name:uDoc}
[User,Admin]	{name:uAdam}

Figure 35: *Cypher* Q0 answer query result

3.4.1.2 Answer 1. What data do you have?

```
MATCH (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
RETURN LABELS(pd) as pdType,
       pd as personalData,
       p as person
```

Listing 14: *Cypher* Q1 answer query

pdType	personalData	person
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11 -26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}

Figure 36: *Cypher* Q1 answer query result

3.4.1.3 Answer 2. Where is the data stored?

```
MATCH (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
OPTIONAL MATCH (pd)-[:STORED_IN]->(db:DB)
OPTIONAL MATCH (db)-[:PART_OF]->(dbS:Service)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS(db) as DBType,
        db as DB,
        CASE
                WHEN EXISTS((db)-[:PART_OF]->(dbS)) THEN
                        dbS.location
                ELSE
                        db.location
        END as location,
        CASE
                WHEN EXISTS((db)-[:PART_OF]->(dbS)) THEN
                        dbS.country
                ELSE
                        db.country
        END as country
```

Listing 15: *Cypher* Q2 answer query

pdType	personalData	person	DBType	DB	location	country
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[System,DB,ServiceHistoryDB]	{name:hist MongoDB,country:GR,description:Heart Rate Sensor History DB,location:Heraklion,version:4.0.10}	Heraklion	GR
[PersonalData,UserData]	{phone: +306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com }	{name:Pat,surname:Williams}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	Heraklion	GR
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers: [uPat]}	{name:Doc,surname:Brown}	[System,DB,PubSubDB]	{name:pubSub MongoDB,description:User Publish-Subscribe DB,version:4.0.4}	Heraklion	GR
[PersonalData,UserData]	{phone: +306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00, email:docbrown@ domain.com }	{name:Doc,surname:Brown}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	Heraklion	GR
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	Heraklion	GR

Figure 37: *Cypher* Q2 answer query result

3.4.1.4 Answer 3. How and when did you obtain the data?

```
MATCH (de)<-[:RELATED_TO]-(pd:PersonalData)
        -[:BELONGS_TO]->(p:Person)
OPTIONAL MATCH (p)-[prov:PROVIDED]->(c:Consent)
        -[:OF_TYPE]->(cT:ConsentType)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        de as fdDataEntity,
        c as consent,
        cT as consentType,
        prov.time as timeConsentProvided
```

Listing 16: *Cypher* Q3 answer query

pdType	personalData	person	fdDataEntity	consent	consentType	timeConsentProvided
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	{name:hrs,description:A sensor device that tracks your heart rate.}	{timeCreated:2018-12-11T12:32:17.645876+03:00}	{name:Patient Emergency Condition,description:Patient health emergency situation condition.}	2018-12-11T12:32:17.645876+03:00
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	{name:hrs,description:A sensor device that tracks your heart rate.}	{timeCreated:2018-12-11T12:32:13.645876+03:00}	{name:Terms & Conditions,description:General user usage terms and conditions.}	2018-12-11T12:32:13.645876+03:00
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	{name:uPat}	{timeCreated:2018-12-11T12:32:17.645876+03:00}	{name:Patient Emergency Condition,description:Patient health emergency situation condition.}	2018-12-11T12:32:17.645876+03:00
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	{name:uPat}	{timeCreated:2018-12-11T12:32:13.645876+03:00}	{name:Terms & Conditions,description:General user usage terms and conditions.}	2018-12-11T12:32:13.645876+03:00
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	{name:uDoc}	{timeCreated:2018-11-26T15:02:10.648376+03:00}	{name:Terms & Conditions,description:General user usage terms and conditions.}	2018-11-26T15:02:10.648376+03:00
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	{name:uDoc}	{timeCreated:2018-11-26T15:02:10.648376+03:00}	{name:Terms & Conditions,description:General user usage terms and conditions.}	2018-11-26T15:02:10.648376+03:00
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}	{name:uAdam}	{timeCreated:2018-11-25T10:13:03.448076+03:00}	{name:Terms & Conditions,description:General user usage terms and conditions.}	2018-11-25T10:13:03.448076+03:00

Figure 38: *Cypher* Q3 answer query result

3.4.1.5 Answer 4. Why do you have the data?

```
MATCH (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
MATCH (p)-[:PROVIDED]->(:Consent)-[:OF_TYPE]->(cT:ConsentType)
MATCH (cT)-[:ENABLES]->(du:DataUsage)
OPTIONAL MATCH (pd)<-[:INVOLVES]-(dp:DataProcess)
OPTIONAL MATCH (f)-[:FROM]->(dp)-[:TO]->(t)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        COLLECT(du) as dataUsages,
        LABELS(dp) as dpType,
        dp as dataProcess,
        [LABELS(f), f] as from,
        [LABELS(t), t] as to
```

Listing 17: *Cypher* Q4 answer query

pdType	personalData	person	dataUsages	dpType	dataProcess	from	to
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export},{name:Emergency User Data Export,description:Export of personal patient data to an external, third party ambulance service (in case of an emergency health situation)}]	[Event,DataProcess,InternalDataMovement]	{timeCreated:2019-04-08T09:01:03.508032+03:00}	[[System,Service,DeviceInterface],{name:Device Interface Service,country:GR,description:Device Interface Service,location:Heraklion,version:}]	[[System,Service,PEPProxy3],{name:Wilma3,country:GR,description:https://fiware-pep-proxy.readthedocs.io,location:Heraklion,version:}]
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export},{name:Emergency User Data Export,description:Export of personal patient data to an external, third party ambulance service (in case of an emergency health situation)}]	[Event,DataProcess,DataExport]	{timeCreated:2019-04-15T17:06:00.27773+03:00}	[[System,DB,RelationalDB],{name:MySQL,description:User Id & Authentication DB,version:8.0.16}]	[[ThirdParty,AmbulanceService],{name:Emergency Ambulance Service,description:First aid emergency ambulance service.}]
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export},{name:Emergency User Data Export,description:Export of personal patient data to an external, third party ambulance service (in case of an emergency health situation)}]	[Event,DataProcess,DataAccess]	{timeCreated:2019-04-10T11:15:00.448076+03:00}	[[App,WebApp],{name:MyHeartMonitor,description:A patient heart rate monitoring web app}]	[[User,Doctor],{name:uDoc}]
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	null	null	[null,null]	[null,null]
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	[Event,DataProcess,DataUpdate]	{timeCreated:2019-04-15T17:06:00.27773+03:00}	[[App,WebApp],{name:MyHeartMonitor,description:A patient heart rate monitoring web app}]	[[System,DB,RelationalDB],{name:MySQL,description:User Id & Authentication DB,version:8.0.16}]
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure}, {name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	null	null	[null,null]	[null,null]

Figure 39: *Cypher* Q4 answer query result

3.4.1.6 Answer 5. Who has access to the data?

```
MATCH (pdOwnerUser:User)-[:RELATED_TO]->(p:Person)
        <-[:BELONGS_TO]-(pd:PersonalData)-[:STORED_IN]->(db:DB)
OPTIONAL MATCH (otherUser:User)-[CAN_ACCESS_DATA_OF]->(pdOwnerUser)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS(db) as DBType,
        db as DB,
        pdOwnerUser,
        COLLECT(otherUser) as otherUsersWAccess
```

Listing 18: *Cypher* Q5 answer query

pdType	personalData	person	DBType	DB	pdOwnerUser	otherUsersWAccess
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[System,DB,Service,HistoryDB]	{name:hist MongoDB,country:GR,description:Heart Rate Sensor History DB,location:Heraklion,version:4.0.10}	{name:uPat}	[{name:uDoc},{name:uAdam}]
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	{name:uPat}	[{name:uDoc},{name:uAdam}]
[PersonalData,Subscription]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[System,DB,PubSubDB]	{name:pubSub MongoDB,description:User Publish-Subscribe DB,version:4.0.4}	{name:uDoc}	[{name:uAdam}]
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	{name:uDoc}	[{name:uAdam}]
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	{name:uAdam}	[]

Figure 40: *Cypher* Q5 answer query result

3.4.1.7 Answer 6. Do you have permission to use the data? For what purposes?

```
MATCH (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
OPTIONAL MATCH (p)-->(c:Consent)
        -[:OF_TYPE]->(cT:ConsentType)
        -[:ENABLES]->(du:DataUsage)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        COLLECT(du) as dataUsagesEnabledByPerson,
        EXISTS((p)-[:PROVIDED]->(c)-[:OF_TYPE]->(cT)
                        -[:ENABLES]->(du)) AND
                NOT EXISTS((p)-[:REVOKED]->(c)
                        -[:OF_TYPE]->(cT)-[:ENABLES]->(du))
                                as permissionToUseData
```

Listing 19: *Cypher* Q6 answer query

pdType	personalData	person	dataUsagesEnabledByPerson	permissionToUseData
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure},{name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	true
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[{name:Emergency User Data Export,description:Export of personal patient data to an external, third party ambulance service (in case of an emergency health situation)},{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure},{name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	true
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure},{name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	true
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	[{name:Sensor Data Auto-sending,description:Automatic transmission of health-related data (generated by the heart rate sensor device) to services within the Remote Patient Monitoring infrastructure},{name:Personal Data Gen & Processing,description:Generation of health-related sensor data, user data and subscription data and its processing by means of internal data movement, data access, update and export}]	true

Figure 41: *Cypher* Q6 answer query result

3.4.1.8 Answer 7. Is the data secure?

```
MATCH (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
        -[:STORED_IN]->(db:DB)
OPTIONAL MATCH (fdEntity)-[:RELATED_TO]->(p:Person)
OPTIONAL MATCH (db)-[:PART_OF]->(serviceOfdb:Service)
OPTIONAL MATCH (restServiceSec:RESTServiceSecurity)
        <-[:ELEMENT_OF]-(secElem)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS(db) as DBType,
        db as DB,
        CASE
                WHEN "Service" IN LABELS(db) THEN
                        EXISTS((fdEntity)
                                -[:EMPLOYS]->(restServiceSec)) AND
                        EXISTS((db)-[:EMPLOYS]->(restServiceSec))
                ELSE
                        EXISTS((fdEntity)
                                -[:EMPLOYS]->(restServiceSec)) AND
                        EXISTS((serviceOfdb)
                                -[:EMPLOYS]->(restServiceSec))
        END as isDataSecure,
        [LABELS(restServiceSec), restServiceSec]
                as securityType,
        COLLECT([LABELS(secElem), secElem])
                as securityElements
```

Listing 20: *Cypher* Q7 answer query

pdt:type	personalData	person	DBT:type	DB	isDataSecure	security:type	securityElements
[PersonalData,UserData]	{phone:+306900000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam, email:adamsmith@domain.com}	{name:Adam,surname:Smith}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	true	[[RESTServiceSecurity],{name:REST Service Security,description:}]	[[OAuth2],{name:OAuth 2.0,description:https://oauth.net/},[[-MasterKeySecurity],{name:Master Key Security,description:}]]
[PersonalData,UserData]	{phone:+306900000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	true	[[RESTServiceSecurity],{name:REST Service Security,description:}]	[[OAuth2],{name:OAuth 2.0,description:https://oauth.net/},[[-MasterKeySecurity],{name:Master Key Security,description:}]]
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[System,DB,RelationalDB]	{name:MySQL,description:User Id & Authentication DB,version:8.0.16}	true	[[RESTServiceSecurity],{name:REST Service Security,description:}]	[[OAuth2],{name:OAuth 2.0,description:https://oauth.net/},[[-MasterKeySecurity],{name:Master Key Security,description:}]]
[PersonalData,SubscriptionData]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribedUsers:[uPat]}	{name:Doc,surname:Brown}	[System,DB,PubSubDB]	{name:PubSub MongoDB,description:User Id & Authentication DB,version:4.0.4}	true	[[RESTServiceSecurity],{name:REST Service Security,description:}]	[[OAuth2],{name:OAuth 2.0,description:https://oauth.net/},[[-MasterKeySecurity],{name:Master Key Security,description:}]]
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[System,DB,Service,HistoryDB]	{name:MongoDB,country:GR,description:Heart Rate Sensor History DB,location:Heraklion,version:4.0.10}	true	[[RESTServiceSecurity],{name:REST Service Security,description:}]	[[OAuth2],{name:OAuth 2.0,description:https://oauth.net/},[[-MasterKeySecurity],{name:Master Key Security,description:}]]

Figure 42: *Cypher* Q7 answer query result

3.4.1.9 Answer 8. How does the data travel through your systems?

```
MATCH (pd:PersonalData)-[:BELONGS_TO]->(p:Person)
MATCH (e:Event)
WHERE e:Consent OR (e:DataProcess AND
        (pd)<-[:INVOLVES]-(e))
OPTIONAL MATCH (e)-[consent_rel]-(p)
OPTIONAL MATCH (t)-[:TO]-(e)-[:FROM]-(f)
OPTIONAL MATCH (e)-[:OF_TYPE]->(cT:ConsentType)
        -[:ENABLES]->(du:DataUsage)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS(e) as eventType,
        CASE
                WHEN e:Consent THEN
                        "OF_TYPE: "+cT.name+" ("+LABELS(cT)[0]+
                                ") | FOR: "+du.name+" ("+
                                        LABELS(du)[0]+")"
                ELSE
                        "FROM: "+f.name+" ("+LABELS(f)[0]+
                                ") | TO: "+t.name+" ("+
                                        LABELS(t)[0]+")"
        END as eventDescription,
        CASE
                WHEN consent_rel IS NOT null THEN
                        consent_rel.time+" ("+TYPE(consent_rel)+")"
                ELSE
                        e.timeCreated
        END as time
ORDER BY time DESC
```

Listing 21: *Cypher* Q8 answer query

personalData	person	eventType	eventDescription	time
{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[Event,DataProcess,DataUpdate]	FROM: MyHeartMonitor (App) | TO: MySQL (System)	2019-04-15T17:06:00.27773+03:00
{phone:+30690000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com }	{name:Pat,surname:Williams}	[Event,DataProcess,DataExport]	FROM: MySQL (System) | TO: Emergency Ambulance Service (Third-Party)	2019-04-15T17:06:00.27773+03:00
{phone:+30690000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,DataProcess,DataAccess]	FROM: MyHeartMonitor (App) | TO: uDoc (User)	2019-04-10T11:15:00.448076+03:00
{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[Event,DataProcess,InternalDataMovement]	FROM: Device Interface Service (System) | TO: Wilma3 (System)	2019-04-08T09:01:03.508032+03:00
{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Patient Emergency Condition (ConsentType) | FOR: Emergency User Data Export (DataUsage)	2018-12-11T12:32:17.645876+03:00 (PROVIDED)
{phone:+30690000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Patient Emergency Condition (ConsentType) | FOR: Emergency User Data Export (DataUsage)	2018-12-11T12:32:17.645876+03:00 (PROVIDED)
{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[Event,Consent]	OF_TYPE: Patient Emergency Condition (ConsentType) | FOR: Emergency User Data Export (DataUsage)	2018-12-11T12:32:17.645876+03:00
{phone:+30690000002,user:uDoc,timeModified:2018-11-26T15:01:10.648376+03:00,email:docbrown@domain.com}	{name:Doc,surname:Brown}	[Event,Consent]	OF_TYPE: Patient Emergency Condition (ConsentType) | FOR: Emergency User Data Export (DataUsage)	2018-12-11T12:32:17.645876+03:00
{phone:+30690000003,timeModified:2018-11-25T10:12:03.448076+03:00,user:uAdam,email:adamsmith@domain.com}	{name:Adam,surname:Smith}	[Event,Consent]	OF_TYPE: Patient Emergency Condition (ConsentType) | FOR: Emergency User Data Export (DataUsage)	2018-12-11T12:32:17.645876+03:00
{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Terms & Conditions (ConsentType) | FOR: Sensor Data Autosending (DataUsage)	2018-12-11T12:32:13.645876+03:00 (PROVIDED)
{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Terms & Conditions (ConsentType) | FOR: Personal Data Gen & Processing (DataUsage)	2018-12-11T12:32:13.645876+03:00 (PROVIDED)
{phone:+30690000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Terms & Conditions (ConsentType) | FOR: Sensor Data Autosending (DataUsage)	2018-12-11T12:32:13.645876+03:00 (PROVIDED)
{phone:+30690000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,Consent]	OF_TYPE: Terms & Conditions (ConsentType) | FOR: Personal Data Gen & Processing (DataUsage)	2018-12-11T12:32:13.645876+03:00 (PROVIDED)

Figure 43: *Cypher* Q8 answer query result

3.4.1.10 Answer 9. Does the data ever cross international borders?

```
MATCH (p:Person)<-[:BELONGS_TO]-(pd:PersonalData)
        <-[:INVOLVES]-(dp:DataProcess)
OPTIONAL MATCH (t)-[:TO]-(dp)-[:FROM]-(f)
OPTIONAL MATCH (f)-[:PART_OF]->(serviceOfFromDB:Service)
OPTIONAL MATCH (t)-[:PART_OF]->(serviceOfToDB:Service)
RETURN LABELS(pd) as pdType,
        pd as personalData,
        p as person,
        LABELS(dp) as dataProcessType,
        "FROM: "+f.name+" ("+LABELS(f)[0]+")| TO: "+
                t.name+" ("+LABELS(t)[0]+")"
                        as eventDescription,
        CASE
                WHEN NOT EXISTS((f)
                        -[:PART_OF]->(serviceOfFromDB)) THEN
                                f.country
                ELSE
                        serviceOfFromDB.country
        END as fromEntity_country,
        CASE
                WHEN NOT EXISTS((t)
                        -[:PART_OF]->(serviceOfToDB:Service)) THEN
                                t.country
                ELSE
                        serviceOfToDB.country
        END as toEntity_country,
        dp.timeCreated as time
ORDER BY time DESC
```

Listing 22: *Cypher* Q9 answer query

pdType	personalData	person	dataProcessType	erentDescription	fromEntity_country	toEntity_country	time
[PersonalData,Subscription Data]	{subscriberUser:uDoc,timeModified:2018-12-11T16:01:16.645876+03:00,subscribeeUsers:[uPat]}	{name:Doc,surname:Brown}	[Event,DataProcess,DataUpdate]	FROM: MyHeartMonitor (App)] TO: MySQL (System)	null	GR	2019-04-15T17:06:00.27773+03:00
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,DataProcess,DataExport]	FROM: MySQL (System)] TO: Emergency Ambulance Service (ThirdParty)	GR	null	2019-04-15T17:06:00.27773+03:00
[PersonalData,UserData]	{phone:+306900000001,user:uPat,timeModified:2018-12-11T12:31:14.645876+03:00,email:patwilliams@domain.com}	{name:Pat,surname:Williams}	[Event,DataProcess,DataAccess]	FROM: MyHeartMonitor (App)] TO: uDoc (User)	null	null	2019-04-10T11:15:00.448076+03:00
[PersonalData,SensorData]	{heartRate:100.0,timeModified:2018-12-11T20:08:13.539991+03:00,sensor:hrs}	{name:Pat,surname:Williams}	[Event,DataProcess,InternalDataMovement]	FROM: Device Interface Service (System)] TO: Wilma3 (System)	GR	GR	2019-04-08T09:01:03.508032+03:00

Figure 44: *Cypher* Q9 answer query result

3.5 Discussion

What our work accomplishes is a means for creating compliance data reports (related to fundamental personal data aspects of the GDPR) for a Remote Patient Monitoring System. More specifically, the steps for designing it involve identifying all components that use or could potentially use GDPR-regulated personal information entities (and their relationships to each other) and building a logical model of all components and their connections represented as a UML data model. The UML model must then be converted into a property graph model that can be instantiated and loaded into a graph database. Once, the data has been loaded, answer queries such as the ones we provided can be created and run upon the graph for addressing the personal data requirements of the GDPR. The query results provide a visual representation of how the entities of the whole system interact with the personal data they both hold and use.

3.5.1 Determining GDPR Compliance

It is important to state that definitive decisions about compliance and non-compliance must be made by GDPR experts. These decisions are determined primarily by the instances of the system's data model containing real data and not exclusively by the data model itself. Nevertheless, the data model must be carefully constructed a-priori, based on the provided GDPR questions so as to include appropriate GDPR-related entities (consent, data processes, etc.) related to personal data compliance requirements, which are crucial for determining the outcome of the compliance evaluation process.

3.5.2 Applicability

All privacy-minded applications involving personal data require and operate based on the following mechanisms and activities:

(a) data acquisition

(b) data exchange between users and services

(c) database storage and retrieval of data

(d) data communication with third-parties

(e) security for data and services and devices that it relates to

Thus, we argue that our solution is general and that it can be applied to a wide range of application domains wherein personal data security and privacy considerations pertaining to GDPR compliance requirements must be enforced and maintained by design and in operation.

To show proof of concept, the effectiveness of the solution is demonstrated in a remote health monitoring application, but it is not strictly limited, specific, nor tightly-bound to IoT, Cloud or patient monitoring contexts.

Chapter 4

Conclusions & Future Work

4.1 Conclusions

In this thesis, we addressed the problem of designing a Remote Patient Monitoring System that can be queried with the help of a graph database for evaluating basic GDPR compliance. Our work's contributions include providing a step-by-step methodology for:

- constructing the a Remote Patient Monitoring System based on the requirements of 10 GDPR-compliance-related questions
- creating reports for the evaluation of basic GDPR compliance by querying the System's property graph in a graph database

4.2 Future Work

Regarding improvements upon our work, the following ideas may prove fruitful for future research:

- Refine the GDPR security requirements of the design process by including 'pseudonymization' schemes, such as data encryption
- Utilize semantic web technologies such as OWL, RDF and Linked Data in order to better define and possibly automate or replace the graph database operations and/or implement SPARQL queries as an alternative to a graph query language (such as *Cypher*) queries for evaluating the platform's GDPR compliance
- Investigate the design of a "*Compliance Cloud Service*" as part of a Compliance as a Service (CaaS) infrastructure, wherein a "compliance micro-service" receives constant system changes in order to update internal graph structures and subtly manage all issues related to GDPR compliance. For example, in a distributed system containing an Authentication service, among other services, every incoming request and outcoming response (such as user login or registration) may be signaled to a Compliance Service that updates a set of nodes and relationships storing personal data and related entities (such as users, their credentials and time of login) offering a means for real-time compliance monitoring.

Appendix A

Graph creation queries

Listing 23: PseudoQL query for creating the Remote Patient Monitoring System class diagram schema graph

```
// Q0
// Create nodes
CREATE (p:Person {name: 'string', surname: 'string'})
CREATE (upat:User:Patient {name: 'string'})
CREATE (udoc:User:Doctor {name: 'string'})
CREATE (uad:User:Admin {name: 'string'})
CREATE (hrs:Device:HeartRateSensor {name: 'string',
    description: 'string'})
// Find existing elements
SELECT (u:User)
// Create relationships
CREATE (u)-[:RELATED_TO]->(p)
CREATE (upat)<-[:RELATED_TO]-(hrs)
// Q1
// Create nodes
CREATE (ud:PersonalData:UserData {
        user: 'string',
        email: 'string',
        phone: 'string',
        timeModified: 'string'
})
CREATE (sd:PersonalData:SensorData {
        sensor: 'string',
        heartRate: 'number',
        timeModified: 'string'
})
CREATE (subd:PersonalData:SubscriptionData {
        subscriberUser: 'string',
        subscribeeUsers: 'list_of_strings',
        timeModified: 'string'
})
// Find existing elements
SELECT (pd:PersonalData)
// Create relationships
```

```
CREATE (pd)-[:BELONGS_TO]->(p)
CREATE (ud)-[:RELATED_TO]->(u)<-[:RELATED_TO]-(subd)
CREATE (sd)-[:RELATED_TO]->(hrs)
// Q2
// Create nodes
CREATE (relDB:System:DB:RelationalDB {
        name: 'string',
        version: 'string',
        description: 'string'
})
CREATE (pubSubDB:System:DB:PubSubDB {
        name: 'string',
        version: 'string',
        description: 'string'
})
CREATE (histDB:System:DB:Service:HistoryDB {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (app:App:WebApp {
        name: 'string',
        description: 'string'
})
CREATE (appLogic:System:Service:AppLogic {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (uIdAuth:System:Service:UserIdAuth {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (authPDP:System:Service:AuthorizationPDP {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (pubSub:System:Service:PublishSubscribe {
        name: 'string',
        version: 'string',
```

```
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (dataStorage:System:Service:DataStorage {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (dataHistRecovery:System:Service:DataHistoryRecovery {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (devInterface:System:Service:DeviceInterface {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (devQuerying:System:Service:DeviceQuerying {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (pepProxy1:System:Service:PEPProxy1 {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (pepProxy2:System:Service:PEPProxy2 {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (pepProxy3:System:Service:PEPProxy3 {
        name: 'string',
        version: 'string',
```

```
        description: 'string',
        location: 'string',
        country: 'string'
})
CREATE (pepProxy4:System:Service:PEPProxy4 {
        name: 'string',
        version: 'string',
        description: 'string',
        location: 'string',
        country: 'string'
})
// Create relationships
CREATE (relDB)<-[:STORED_IN]-(ud)
CREATE (pubSubDB)<-[:STORED_IN]-(subd)
CREATE (histDB)<-[:STORED_IN]-(sd)
CREATE (u)-[:USES]->(app)
CREATE (app)-[:CONNECTED_TO]->(appLogic)
CREATE (authPDP)<-[:CONNECTED_TO]-(uIdAuth)<-[:PART_OF]-(relDB)
CREATE (pubSub)<-[:PART_OF]-(pubSubDB)
CREATE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy1)-[:CONNECTED_TO]->(authPDP)
CREATE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy3)-[:CONNECTED_TO]->(authPDP)
CREATE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy4)-[:CONNECTED_TO]-> (authPDP)
CREATE (devQuerying)-[:CONNECTED_TO]->(pubSub)
CREATE (devQuerying)-[:CONNECTED_TO]->(pepProxy4)
CREATE (pepProxy3)-[:CONNECTED_TO]->(pubSub)
CREATE (pubSub)-[:CONNECTED_TO]->(pepProxy2)
        -[:CONNECTED_TO]->(dataStorage)-[:CONNECTED_TO]->(histDB)
CREATE (pepProxy1)-[:CONNECTED_TO]->(dataHistRecovery)
        -[:CONNECTED_TO]->(histDB)
CREATE (hrs)-[:CONNECTED_TO]->(devInterface)-[:CONNECTED_TO]->(pepProxy3)
CREATE (appLogic)-[:CONNECTED_TO]->(uIdAuth)
CREATE (appLogic)-[:CONNECTED_TO]->(pepProxy1)
CREATE (appLogic)-[:CONNECTED_TO]->(pepProxy3)
CREATE (appLogic)-[:CONNECTED_TO]->(pepProxy4)
// Q3
// Create nodes
CREATE (c:Event:Consent {timeCreated: 'string'})
CREATE (cT:ConsentType {
        name: 'string',
        description: 'string'
})
// Create relationships
CREATE (p)-[:PROVIDED {time: 'string'}]->(c)
CREATE (p)-[:REVOKED {time: 'string'}]->(c)
CREATE (c)-[:OF_TYPE]->(cT)
// Q4
// Create nodes
CREATE (du:DataUsage {
        name: 'string',
        description: 'string'
})
```

```
CREATE (internalDataMovement:InternalDataMovement:DataProcess:Event {
        timeCreated: 'string'
})
CREATE (dataAccess:DataAccess:DataProcess:Event {timeCreated: 'string'})
CREATE (dataUpdate:DataUpdate:DataProcess:Event {timeCreated: 'string'})
CREATE (dataExport:DataExport:DataProcess:Event {timeCreated: 'string'})
CREATE (ambulance:AmbulanceService:ThirdParty {
        name: 'string',
        description: 'string'
})
// Find existing elements
SELECT (db:DB)
SELECT (s:System)
SELECT (dp:DataProcess)
// Create relationships
CREATE (cT)-[:ENABLES]->(du)
CREATE (dp)-[:INVOLVES]->(pd)
CREATE (s)-[:FROM]->(internalDataMovement)
CREATE (internalDataMovement)-[:TO]->(s)
CREATE (app)-[:FROM]->(dataAccess)
CREATE (dataAccess)-[:TO]->(u)
CREATE (app)-[:FROM]->(dataUpdate)
CREATE (dataUpdate)-[:TO]->(db)
CREATE (db)-[:FROM]->(dataExport)
CREATE (dataExport)-[:TO]->(ambulance)
// Q5
// Find existing elements
SELECT (pat:Patient)
SELECT (doc:Doctor)
SELECT (admin:Admin)
// Create relationships
CREATE (doc)-[:CAN_ACCESS_DATA_OF]->(pat)
CREATE (admin)-[:CAN_ACCESS_DATA_OF]->(doc)
CREATE (admin)-[:CAN_ACCESS_DATA_OF]->(pat)
// Q7
// Create nodes
CREATE (restServiceSecurity:RESTServiceSecurity {description: 'string'})
CREATE (oauth2:OAuth2 {description: 'string'})
CREATE (masterKeySecurity:MasterKeySecurity {description: 'string'})
CREATE (role:Role {name: 'string'})
CREATE (xacmlRule:XACMLRule {accessControlRule: 'string'})
CREATE (perm :Permission {name: 'string', HTTPAction: 'string',
    resource: 'string'})
// Find existing elements
SELECT (service:Service)
// Create relationships
CREATE (oauth2)-[:ELEMENT_OF]->(restServiceSecurity)
CREATE (masterKeySecurity)-[:ELEMENT_OF]->(restServiceSecurity)
CREATE (role)-[:ELEMENT_OF]->(restServiceSecurity)
CREATE (xacmlRule)-[:ELEMENT_OF]->(restServiceSecurity)
CREATE (perm)-[:ELEMENT_OF]->(restServiceSecurity)
```

```
CREATE (role)-[:HAS]->(xacmlRule)
CREATE (role)-[:HAS]->(perm)
CREATE (u)-[:HAS]->(role)
CREATE (u)-[:EMPLOYS]->(restServiceSecurity)
CREATE (hrs)-[:EMPLOYS]->(restServiceSecurity)
CREATE (service)-[:EMPLOYS]->(restServiceSecurity)
```

Listing 24: Cypher query for creating the data parameters for the Demo's (3.4) instantiated Remote Patient Monitoring System graph

```
// Q0 params
:params patProps: {name: "Pat", surname: "Williams"},
        docProps: {name: "Doc", surname: "Brown"},
        adamProps: {name: "Adam", surname: "Smith"},
        uPatProps: {name: "uPat"},
        uDocProps: {name: "uDoc"},
        uAdamProps: {name: "uAdam"},
        hrsProps: {
        name: "hrs",
        description: "A sensor device that tracks your heart rate."
    },
// Q1 params
        PatudProps: {
                user: "uPat",
                email: "patwilliams@domain.com",
                phone: "+306900000001",
                timeModified: "2018-12-11T12:31:14.645876+03:00"
        },
        DocudProps: {
                user: "uDoc",
                email: "docbrown@domain.com",
                phone: "+306900000002",
                timeModified: "2018-11-26T15:01:10.648376+03:00"
        },
        AdamudProps: {
                user: "uAdam",
                email: "adamsmith@domain.com",
                phone: "+306900000003",
                timeModified: "2018-11-25T10:12:03.448076+03:00"
        },
        DocsubdProps: {
                subscriberUser: "uDoc",
                subscribeeUsers: ["uPat"],
                timeModified: "2018-12-11T16:01:16.645876+03:00"
        },
        PatsdProps: {
                sensor: "hrs",
                heartRate: 100,
                timeModified: "2018-12-11T20:08:13.539991+03:00"
        },
// Q2 params
```

```
appProps: {
        name: "MyHeartMonitor",
        description: "A patient heart rate monitoring web app"
},
relDBProps: {
        name: "MySQL",
        version: "8.0.16",
        description: "User Id & Authentication DB"
},
pubSubDBProps: {
        name: "pubSub MongoDB",
        version: "4.0.4",
        description: "User Publish-Subscribe DB"
},
histDBProps: {
        name: "hist MongoDB",
        version: "4.0.10",
        description: "Heart Rate Sensor History DB",
        location: "Heraklion",
        country: "GR"
},
appLogicProps: {
        name: "App Logic Service",
        version: "",
        description: "App Logic Service",
        location: "Heraklion",
        country: "GR"
},
uIdAuthProps: {
        name: "IdM-Keyrock",
        version: "",
        description: "https://fiware-idm.readthedocs.io",
        location: "Heraklion",
        country: "GR"
},
authPDPProps: {
        name: "AuthzForce",
        version: "",
        description:
    "https://authzforce-ce-fiware.readthedocs.io",
        location: "Heraklion",
        country: "GR"
},
pubSubProps: {
        name: "Orion Context Broker",
        version: "",
        description: "https://fiware-orion.readthedocs.io",
        location: "Heraklion",
        country: "GR"
},
dataStorageProps: {
```

```
        name: "Cygnus",
        version: "",
        description: "https://fiware-cygnus.readthedocs.io",
        location: "Heraklion",
        country: "GR"
    },
    dataHistRecoveryProps: {
        name: "Comet",
        version: "",
        description: "https://fiware-sth-comet.readthedocs.io",
        location: "Heraklion",
        country: "GR"
    },
    devInterfaceProps: {
        name: "Device Interface Service",
        version: "",
        description: "Device Interface Service",
        location: "Heraklion",
        country: "GR"
    },
    devQueryingProps: {
        name: "Device Querying Service",
        version: "",
        description: "Device Querying Service",
        location: "Heraklion",
        country: "GR"
    },
    pepProxy1Props: {
        name: "Wilma1",
        version: "",
        description: "https://fiware-pep-proxy.readthedocs.io",
        location: "Heraklion",
        country: "GR"
    },
    pepProxy2Props: {
        name: "Wilma2",
        version: "",
        description: "https://fiware-pep-proxy.readthedocs.io",
        location: "Heraklion",
        country: "GR"
    },
    pepProxy3Props: {
        name: "Wilma3",
        version: "",
        description: "https://fiware-pep-proxy.readthedocs.io",
        location: "Heraklion",
        country: "GR"
    },
    pepProxy4Props: {
        name: "Wilma4",
        version: "",
```

```
                description: "https://fiware-pep-proxy.readthedocs.io",
                location: "Heraklion",
                country: "GR"
        },
// Q3 params
        patTermsNCondProps: {
                timeCreated: "2018-12-11T12:32:13.645876+03:00"
        },
        patEmergCondProps: {
                timeCreated: "2018-12-11T12:32:17.645876+03:00"
        },
        docTermsNCondProps: {
                timeCreated: "2018-11-26T15:02:10.648376+03:00"
        },
        adamTermsNCondProps: {
                timeCreated: "2018-11-25T10:13:03.448076+03:00"
        },
        patTermsNCondProvidedProps: {
                time: "2018-12-11T12:32:13.645876+03:00"
        },
        patEmergCondProvidedProps: {
                time: "2018-12-11T12:32:17.645876+03:00"
        },
        docTermsNCondProvidedProps: {
                time: "2018-11-26T15:02:10.648376+03:00"
        },
        adamTermsNCondProvidedProps: {
                time: "2018-11-25T10:13:03.448076+03:00"
        },
        termsNCondProps: {
                name: "Terms & Conditions",
                description: "General user usage terms and conditions."
        },
        emergCondProps: {
                name: "Patient Emergency Condition",
                description:
            "Patient health emergency situation condition."
        },
// Q4 params
        pdGenNProcessingProps:{
                name: "Personal Data Gen & Processing",
                description: "Generation of health-related
                      sensor data, user data and subscription
                      data and its processing by means of
                      internal data movement,
                      data access, update and export"
        },
        sdAutoSendProps: {
                name: "Sensor Data Auto-sending",
                description: "Automatic transmission of
                      health-related data (generated by the
```

```
                    heart rate sensor device) to services
                    within the Remote Patient Monitoring
                    infrastructure"
        },
        udEmergExportProps: {
                name: "Emergency User Data Export",
                description: "Export of personal patient data to an
                    external, third party ambulance service
                    (in case of an emergency health
                    situation)"
        },
        internalDataMovementProps: {
                timeCreated: "2019-04-08T09:01:03.508032+03:00"
        },
        dataAccessProps: {
                timeCreated: "2019-04-10T11:15:00.448076+03:00"
        },
        dataUpdateProps: {
                timeCreated: "2019-04-15T17:06:00.27773+03:00"
        },
        dataExportProps: {
                timeCreated: "2019-04-15T17:06:00.27773+03:00"
        },
        ambulanceProps: {
                name: "Emergency Ambulance Service",
                description: "First aid emergency ambulance service."
        },
// Q7 params
        restServiceSecurityProps: {
                name: "REST Service Security",
                description: ""
        },
        oauth2Props: {
                name: "OAuth 2.0",
                description: "https://oauth.net/"
        },
        masterKeySecProps: {
                name: "Master Key Security",
                description: ""
        }
```

Listing 25: Cypher query for creating the instantiated graph of the Remote Patient Monitoring System for the Demo (3.4)

```
// Q0
MERGE (Pat:Person {name: $patProps.name, surname: $patProps.surname})
MERGE (uPat:User:Patient {name: $uPatProps.name})
MERGE (hrs:Device:HeartRateSensor {name: $hrsProps.name,
    description: $hrsProps.description})
MERGE (uDoc:User:Doctor {name: $uDocProps.name})
MERGE (Doc:Person {name: $docProps.name, surname: $docProps.surname})
```

```
MERGE (uAdam:User:Admin {name: $uAdamProps.name})
MERGE (Adam:Person {name: $adamProps.name,
    surname: $adamProps.surname})
// Create relationships
MERGE (Pat)<-[:RELATED_TO]-(uPat)<-[:RELATED_TO]-(hrs)
MERGE (uDoc)-[:RELATED_TO]->(Doc)
MERGE (uAdam)-[:RELATED_TO]->(Adam)
// Q1
MERGE (Patud:PersonalData:UserData {
                user: $PatudProps.user,
                email: $PatudProps.email,
                phone: $PatudProps.phone,
                timeModified: $PatudProps.timeModified
        })
MERGE (Docud:PersonalData:UserData {
                        user: $DocudProps.user,
                        email: $DocudProps.email,
                        phone: $DocudProps.phone,
                        timeModified: $DocudProps.timeModified
                })
MERGE (Adamud:PersonalData:UserData {
                user: $AdamudProps.user,
                email: $AdamudProps.email,
                phone: $AdamudProps.phone,
                timeModified: $AdamudProps.timeModified
        })
MERGE (Docsubd:PersonalData:SubscriptionData {
                subscriberUser: $DocsubdProps.subscriberUser,
                subscribeeUsers: $DocsubdProps.subscribeeUsers,
                timeModified: $DocsubdProps.timeModified
        })
MERGE (Patsd:PersonalData:SensorData {
                sensor: $PatsdProps.sensor,
                heartRate: $PatsdProps.heartRate,
                timeModified: $PatsdProps.timeModified
        })
// Create relationships
MERGE (uPat)<-[:RELATED_TO]-(Patud)-[:BELONGS_TO]->(Pat)
MERGE (uDoc)<-[:RELATED_TO]-(Docud)-[:BELONGS_TO]->(Doc)
MERGE (uAdam)<-[:RELATED_TO]-(Adamud)-[:BELONGS_TO]->(Adam)
MERGE (uDoc)<-[:RELATED_TO]-(Docsubd)-[:BELONGS_TO]->(Doc)
MERGE (hrs)<-[:RELATED_TO]-(Patsd)-[:BELONGS_TO]->(Pat)
// Q2
// Create nodes
MERGE (app:App:WebApp {
                name: $appProps.name,
                description: $appProps.description
        })
MERGE (relDB:System:DB:RelationalDB {
        name: $relDBProps.name,
```

```
        version: $relDBProps.version,
        description: $relDBProps.description
})
MERGE (pubSubDB:System:DB:PubSubDB {
        name: $pubSubDBProps.name,
        version: $pubSubDBProps.version,
        description: $pubSubDBProps.description
})
MERGE (histDB:System:DB:Service:HistoryDB {
        name: $histDBProps.name,
        version: $histDBProps.version,
        description: $histDBProps.description,
        location: $histDBProps.location,
        country: $histDBProps.country
})
MERGE (appLogic:System:Service:AppLogic {
        name: $appLogicProps.name,
        version: $appLogicProps.version,
        description: $appLogicProps.description,
        location: $appLogicProps.location,
        country: $appLogicProps.location
})
MERGE (uIdAuth:System:Service:UserIdAuth {
        name: $uIdAuthProps.name,
        version: $uIdAuthProps.version,
        description: $uIdAuthProps.description,
        location: $uIdAuthProps.location,
        country: $uIdAuthProps.country
})
MERGE (authPDP:System:Service:AuthorizationPDP {
        name: $authPDPProps.name,
        version: $authPDPProps.version,
        description: $authPDPProps.description,
        location: $authPDPProps.location,
        country: $authPDPProps.country
})
MERGE (pubSub:System:Service:PublishSubscribe {
        name: $pubSubProps.name,
        version: $pubSubProps.version,
        description: $pubSubProps.description,
        location: $pubSubProps.location,
        country: $pubSubProps.country
})
MERGE (dataStorage:System:Service:DataStorage {
        name: $dataStorageProps.name,
        version: $dataStorageProps.version,
        description: $dataStorageProps.description,
        location: $dataStorageProps.location,
        country: $dataStorageProps.country
```

```
})
MERGE (dataHistRecovery:System:Service:DataHistoryRecovery {
        name: $dataHistRecoveryProps.name,
        version: $dataHistRecoveryProps.version,
        description: $dataHistRecoveryProps.description,
        location: $dataHistRecoveryProps.location,
        country: $dataHistRecoveryProps.country
})
MERGE (devInterface:System:Service:DeviceInterface {
        name: $devInterfaceProps.name,
        version: $devInterfaceProps.version,
        description: $devInterfaceProps.description,
        location: $devInterfaceProps.location,
        country: $devInterfaceProps.country
})
MERGE (devQuerying:System:Service:DeviceQuerying {
        name: $devQueryingProps.name,
        version: $devQueryingProps.version,
        description: $devQueryingProps.description,
        location: $devQueryingProps.location,
        country: $devQueryingProps.country
})
MERGE (pepProxy1:System:Service:PEPProxy1 {
        name: $pepProxy1Props.name,
        version: $pepProxy1Props.version,
        description: $pepProxy1Props.description,
        location: $pepProxy1Props.location,
        country: $pepProxy1Props.country
})
MERGE (pepProxy2:System:Service:PEPProxy2 {
        name: $pepProxy2Props.name,
        version: $pepProxy2Props.version,
        description: $pepProxy2Props.description,
        location: $pepProxy2Props.location,
        country: $pepProxy2Props.country
})
MERGE (pepProxy3:System:Service:PEPProxy3 {
        name: $pepProxy3Props.name,
        version: $pepProxy3Props.version,
        description: $pepProxy3Props.description,
        location: $pepProxy3Props.location,
        country: $pepProxy3Props.country
})
MERGE (pepProxy4:System:Service:PEPProxy4 {
        name: $pepProxy4Props.name,
        version: $pepProxy4Props.version,
        description: $pepProxy4Props.description,
        location: $pepProxy4Props.location,
        country: $pepProxy4Props.country
```

```
})
// Create relationships
MERGE (u:User)
MERGE (u)-[:USES]->(app)
MERGE (app)-[:CONNECTED_TO]->(appLogic)
MERGE (ud:UserData)
MERGE (relDB)<-[:STORED_IN]-(ud)
MERGE (authPDP)<-[:CONNECTED_TO]-(uIdAuth)<-[:PART_OF]-(relDB)
MERGE (subd:SubscriptionData)
MERGE (pubSubDB)<-[:STORED_IN]-(subd)
MERGE (pubSub)<-[:PART_OF]-(pubSubDB)
MERGE (sd:SensorData)
MERGE (histDB)<-[:STORED_IN]-(sd)
MERGE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy1)-[:CONNECTED_TO]->(authPDP)
MERGE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy3)-[:CONNECTED_TO]->(authPDP)
MERGE (uIdAuth)<-[:CONNECTED_TO]-(pepProxy4)-[:CONNECTED_TO]->(authPDP)
MERGE (appLogic)-[:CONNECTED_TO]->(uIdAuth)
MERGE (devQuerying)-[:CONNECTED_TO]->(pubSub)
MERGE (devQuerying)-[:CONNECTED_TO]->(pepProxy4)
MERGE (pepProxy3)-[:CONNECTED_TO]->(pubSub)
MERGE (pubSub)-[:CONNECTED_TO]->(pepProxy2)-[:CONNECTED_TO]->
(dataStorage)-[:CONNECTED_TO]->(histDB)
MERGE (pepProxy1)-[:CONNECTED_TO]->(dataHistRecovery)
    -[:CONNECTED_TO]->(histDB)
MERGE (hrs)-[:CONNECTED_TO]->(devInterface)-[:CONNECTED_TO]->(pepProxy3)
MERGE (appLogic)-[:CONNECTED_TO]->(uIdAuth)
MERGE (appLogic)-[:CONNECTED_TO]->(pepProxy1)
MERGE (appLogic)-[:CONNECTED_TO]->(pepProxy3)
MERGE (appLogic)-[:CONNECTED_TO]->(pepProxy4)
MERGE (appLogic)-[:CONNECTED_TO]->(uIdAuth)
// Q3
// Create nodes
MERGE (patTermsNCond:Event:Consent {
        timeCreated: $patTermsNCondProps.timeCreated
})
MERGE (patEmergCond:Event:Consent {
        timeCreated: $patEmergCondProps.timeCreated
})
MERGE (docTermsNCond:Event:Consent {
        timeCreated: $docTermsNCondProps.timeCreated
})
MERGE (adamTermsNCond:Event:Consent {
        timeCreated: $adamTermsNCondProps.timeCreated
})
MERGE (termsNCond:ConsentType {
        name: $termsNCondProps.name,
        description: $termsNCondProps.description
})
MERGE (emergCond:ConsentType {
        name: $emergCondProps.name,
        description: $emergCondProps.description
```

```
})
// Create relationships
MERGE (Pat)
    -[:PROVIDED {time: $patTermsNCondProvidedProps.time}]->
        (patTermsNCond)
MERGE (Pat)
    -[:PROVIDED {time: $patEmergCondProvidedProps.time}]->
        (patEmergCond)
MERGE (Doc)
    -[:PROVIDED {time: $docTermsNCondProvidedProps.time}]->
        (docTermsNCond)
MERGE (Adam)
    -[:PROVIDED {time: $adamTermsNCondProvidedProps.time}]->
        (adamTermsNCond)
MERGE (patTermsNCond)-[:OF_TYPE]->(termsNCond)
MERGE (patEmergCond)-[:OF_TYPE]->(emergCond)
MERGE (docTermsNCond)-[:OF_TYPE]->(termsNCond)
MERGE (adamTermsNCond)-[:OF_TYPE]->(termsNCond)
// Q4
// Create nodes
MERGE (pdGenNProcessing:DataUsage {
        name: $pdGenNProcessingProps.name,
        description: $pdGenNProcessingProps.description
})
MERGE (sdAutoSend:DataUsage {
        name: $sdAutoSendProps.name,
        description: $sdAutoSendProps.description
})
MERGE (udEmergExport:DataUsage {
        name: $udEmergExportProps.name,
        description: $udEmergExportProps.description
})
MERGE (internalDataMovement:Event:DataProcess:InternalDataMovement {
        timeCreated: $internalDataMovementProps.timeCreated
})
MERGE (dataAccess:Event:DataProcess:DataAccess {
        timeCreated: $dataAccessProps.timeCreated
})
MERGE (dataUpdate:Event:DataProcess:DataUpdate {
        timeCreated: $dataUpdateProps.timeCreated
})
MERGE (dataExport:Event:DataProcess:DataExport {
        timeCreated: $dataExportProps.timeCreated
})
MERGE (ambulance:ThirdParty:AmbulanceService {
        name: $ambulanceProps.name,
        description: $ambulanceProps.description
})
// Create relationships
MERGE (termsNCond)-[:ENABLES]->(pdGenNProcessing)
```

```
MERGE (termsNCond)-[:ENABLES]->(sdAutoSend)
MERGE (emergCond)-[:ENABLES]->(udEmergExport)
MERGE (Patsd)<-[:INVOLVES]-(internalDataMovement)
MERGE (Patud)<-[:INVOLVES]-(dataAccess)
MERGE (Docsubd)<-[:INVOLVES]-(dataUpdate)
MERGE (Patud)<-[:INVOLVES]-(dataExport)
MERGE (devInterface)-[:FROM]->(internalDataMovement)-[:TO]->(pepProxy3)
MERGE (app)-[:FROM]->(dataAccess)-[:TO]->(uDoc)
MERGE (app)-[:FROM]->(dataUpdate)-[:TO]->(relDB)
MERGE (relDB)-[:FROM]->(dataExport)-[:TO]->(ambulance)
// Q5
// Find existing elements
MERGE (patient:Patient)
MERGE (doctor:Doctor)
MERGE (admin:Admin)
// Create relationships
MERGE (uPat)<-[:CAN_ACCESS_DATA_OF]-(uDoc)
MERGE (doctor)<-[:CAN_ACCESS_DATA_OF]-(admin)
MERGE (patient)<-[:CAN_ACCESS_DATA_OF]-(admin)
// Q7
// Find existing elements
MERGE (d:Device)
MERGE (s:Service)
// Create nodes
MERGE (restServiceSecurity:RESTServiceSecurity {
        name: $restServiceSecurityProps.name,
        description: $restServiceSecurityProps.description
})
MERGE (oauth2:OAuth2 {
        name: $oauth2Props.name,
        description: $oauth2Props.description
})
MERGE (masterKeySec:MasterKeySecurity {
        name: $masterKeySecProps.name,
        description: $masterKeySecProps.description
})
// Create relationships
MERGE (oauth2)-[:ELEMENT_OF]->(restServiceSecurity)
MERGE (masterKeySec)-[:ELEMENT_OF]->(restServiceSecurity)
MERGE (u)-[:EMPLOYS]->(restServiceSecurity)
MERGE (d)-[:EMPLOYS]->(restServiceSecurity)
MERGE (s)-[:EMPLOYS]->(restServiceSecurity)
```

For write and match operations we chose Cypher's *MERGE* clause instead of *CREATE* and *MATCH*. *MERGE* first checks if existing nodes and/or relationships already exist. If they already exist, it binds them to a given variable. If they do not exist, it creates them. This guarantees uniqueness in node and relationship creation (no duplicate elements), in case the queries are re-run. [54]

References

[1] European Parliament and Council of European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016.

[2] Ben Wolford. What are the GDPR Fines? https://gdpr.eu/fines/, 2018.

[3] Facebook–Cambridge Analytica data scandal — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Facebook%E2%80%93Cambridge_Analytica_data_scandal&oldid=927996299.

[4] AOL search data leak — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=AOL_search_data_leak&oldid=924339868.

[5] Data retention — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Data_retention&oldid=925527721.

[6] Recital 58, The Principle of Transparency. https://gdpr-info.eu/recitals/no-58/.

[7] Neo4j. https://neo4j.com/.

[8] Neo4j Privacy Shield: The Graph Solution for GDPR. https://neo4j.com/use-cases/gdpr-compliance/.

[9] Data lineage — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Data_lineage&oldid=927367434.

[10] Unified Modeling Language (UML) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&oldid=927130803.

[11] Graph database — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Graph_database&oldid=927409485.

[12] Renzo Angles. The Property Graph Database Model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*, 2018.

[13] Ontology (information science) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=927115114.

[14] Web Ontology Language (OWL). https://www.w3.org/OWL/.

[15] Service-oriented architecture — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Service-oriented_architecture&oldid=927325893.

[16] Florian Kammüller, Oladapo O Ogunyanwo, and Christian W Probst. Designing Data Protection for GDPR Compliance into IoT Healthcare Systems. *arXiv preprint arXiv:1901.02426*, 2019.

[17] A Consent and Data Management Model. http://openscience.adaptcentre.ie/projects/CDMM/.

[18] trust-hub. https://www.trust-hub.com/.

[19] Cambridge Intelligence. https://cambridge-intelligence.com/.

[20] Art. 1 GDPR, Subject-matter and objectives. https://gdpr-info.eu/art-1-gdpr/.

[21] Art. 4 GDPR, Definitions. https://gdpr-info.eu/art-4-gdpr/.

[22] Art. 25 GDPR, Data protection by design and by default. https://gdpr-info.eu/art-25-gdpr/.

[23] trust-hub: using graph technologies to power personal data compliance. https://cambridge-intelligence.com/trust-hub-using-graph-technologies-to-power-personal-data-compliance/.

[24] Neo4 Privacy Shield Data Sheet. https://neo4j.com/resources/neo4j-privacy-shield-data-sheet/.

[25] Directed graph — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Directed_graph&oldid=921240292.

[26] Information Commissioner's Office — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Information_Commissioner%27s_Office&oldid=917179011.

[27] Art. 38 GDPR, Position of the data protection officer. https://gdpr-info.eu/art-38-gdpr/.

[28] draw.io. https://about.draw.io/.

[29] SQL. https://en.wikipedia.org/w/index.php?title=SQL&oldid=926073015.

[30] Graph DB vs RDBMS. https://neo4j.com/developer/graph-db-vs-rdbms/.

[31] Neo4j Graph Platform. https://neo4j.com/product/.

[32] Cypher Graph Query Language. https://neo4j.com/cypher-graph-query-language/.

[33] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 1433–1445, New York, NY, USA, 2018. ACM.

[34] Cypher Manual. https://neo4j.com/docs/cypher-manual.

[35] Chapter 3 GDPR, Rights of the data subject.
https://gdpr-info.eu/chapter-3/.

[36] GDPRtEXT. http://openscience.adaptcentre.ie/projects/GDPRtEXT/.

[37] GConsent - A consent ontology based on the GDPR.
https://w3id.org/GConsent.

[38] GDPRov - GDPR Provenance Ontology.
http://openscience.adaptcentre.ie/projects/CDMM/GDPRov/.

[39] Data Protection Rules Language.
https://openscience.adaptcentre.ie/projects/CDMM/DPRL/.

[40] Queryable Provenance Metadata For GDPR Compliance - GDPR Readiness-Checklist SPARQL demo.
http://openscience.adaptcentre.ie/GDPR-checklist-demo/demo/.

[41] SPARQL Query Language for RDF.
https://www.w3.org/TR/rdf-sparql-query/.

[42] Exploring GDPR Compliance Over Provenance Graphs Using SHACL.
http://openscience.adaptcentre.ie/projects/CDMM/compliance/model.html.

[43] Shapes Constraint Language (SHACL). https://www.w3.org/TR/shacl/.

[44] Test-driven approach for GDPR Compliance.
http://openscience.adaptcentre.ie/projects/CDMM/compliance/index.html.

[45] Xenophon Koundourakis and Euripides G.M. Petrakis. iXen: Secure Context-Driven Service Oriented Architecture for the Internet of Things in the Cloud. 2019. unpublished.

[46] Microservices — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Microservices&oldid=928830113.

[47] Security by Design Principles.
https://www.owasp.org/index.php/Security_by_Design_Principles.

[48] Representational state transfer (REST) — Wikipedia, The Free Encyclopedia.
https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=928353400.

[49] OAuth 2.0. https://oauth.net/2/.

[50] XACML — Wikipedia, The Free Encyclopedia.
https://en.wikipedia.org/w/index.php?title=XACML&oldid=920117198.

[51] FIWARE. https://www.fiware.org/.

[52] OpenStack — Wikipedia, The Free Encyclopedia.
https://en.wikipedia.org/w/index.php?title=OpenStack&oldid=922665020.

[53] Cypher MATCH clause.
https://neo4j.com/docs/cypher-manual/current/clauses/match/.

[54] Cypher MERGE clause.
https://neo4j.com/docs/cypher-manual/current/clauses/merge/.

[55] Ascii art — Wikipedia, the free encyclopedia.
https://en.wikipedia.org/w/index.php?title=ASCII_art&oldid=925954266.

[56] Semantic Web. https://www.w3.org/standards/semanticweb/.

[57] Konstantinos Douzis, Stelios Sotiriadis, Euripides G.M. Petrakis, and Cristiana Amza. Modular and Generic IoT Management on the Cloud. *Future Gener. Comput. Syst.*, 78(P1):369–378, January 2018.

[58] Cypher Parameters.
https://neo4j.com/docs/cypher-manual/current/syntax/parameters/.

[59] Art. 39 GDPR, Tasks of the data protection officer.
https://gdpr-info.eu/art-39-gdpr/.