FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Interoperability In Software Applications For Smart Cities: Towards A Reference Architecture

José Pedro Vieira de Carvalho Pinto

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo J. F. Rossetti

Co-supervisor: João Pedro M. T. Dias

July 15, 2016

Interoperability In Software Applications For Smart Cities: Towards A Reference Architecture

José Pedro Vieira de Carvalho Pinto

Mestrado Integrado em Engenharia Informática e Computação

July 15, 2016

Abstract

With the growth of population in urban areas, there is a constant necessity for development and innovation on technologies that surround its citizens. Intelligent cities or *Smart Cities* are the answer to that necessity. A city is considered *smart* when investment is made in social elements, traditional transportation and communication infrastructures as means of maintaining a sustainable economy while increasing the quality of life taking into account a wise management of natural resources through a participative governmental entity. A fundamental part of the development process of a *Smart City* is the engaged participation of its citizens. This aspect is essential for the production of innovative and quality services useful to them.

Taking an important role, with the use of mobile applications, citizens provide fundamental information that is taken into consideration for several moments, namely, in decision making for different contexts during the development of new services and the evolution of *Smart Cities*. The use of mobile applications, as a participative and involving activity by a city's population, ends up not lasting very long and not being very productive, often caused by the poor quality of existing applications and the lack of means for data aggregation. Such aspects result on the dispersion of data between inconsistent databases without any information standards whatsoever.

With this we see a decrease in the amount of data retrieved as well its quality overall and validation possibilities, possibly negating any future usefulness it might have. However, with the specification of a software architecture and its development which leads us to an environment for open data, directed at, but not restricted to, mobile applications. Within, there is the opportunity for Smart Cities to collect data input on mobile data of citizens' engagement occurrences as well and supporting endless setups for either business models or development of support tools for the city's government officials. Such an architecture enables the handling of heterogeneous data specifications while promoting an active participative role in citizens and mobile application developers given that they may later model other applications of top this open data ecosystem.

Resumo

Com o crescimento da população em zonas urbanas cria-se uma necessidade constante de desenvolvimento e inovação das tecnologias que rodeiam os cidadãos. As cidades inteligentes ou *Smart Cities* são a resposta a essa necessidade. Uma cidade é considerada *smart* quando são feitos investimentos em elementos sociais, transportes tradicionais e infraestruturas de comunicação no sentido de manter uma economia sustentável, aumentando a qualidade de vida, tendo em conta uma gestão astuta dos recursos naturais através de uma entidade governativa participativa. Uma parte fundamental do processo de desenvolvimento de uma *Smart City* é a participação comprometida dos cidadãos. Este aspeto é essencial para que sejam produzidos serviços com qualidade e inovadores, úteis para os próprios cidadãos.

Tomando um papel importante, com a utilização de aplicações móveis, os cidadãos fornecem informações fundamentais tidas em conta em diferentes momentos, nomeadamente, na tomada de decisões dentro de diferentes contextos durante da geração de novos serviços e no desenvolvimento das cidades inteligentes. Esta atividade de participação envolvente com a utilização de aplicações móveis por parte da população verifica-se por vezes não ser duradoura e pouco produtiva, muitas vezes causada pela existência de aplicações de fraca qualidade e métodos para a recolha de dados por vezes inexistentes. Desta forma ficamos com dados dispersos em diferentes bases de dados inconsistentes, não havendo padrões para definir a informação a ser guardada.

Com isto vemos prejudicado a recolha de dados constante assim como a sua validação, negando qualquer utilidade que esta possa ter. No entanto, com a especificação de uma arquitetura de software e o seu desenvolvimento que nos leva a um ambiente de Open Data, direcionado a, mas não limitado por aplicações móveis. Dentro disto, existe a oportunidade para Smart Cities para colecionar input de ocorrências de atividade de cidadãos através de dados móveis assim como suportar inúmeros modelos de negócio ou o desenvolvimento de ferramentas de apoio para oficiais do governo da cidade. Esta arquitetura permite lidar com dados especificados de forma heterogénea enquanto promove um papel ativamente participativo nos cidadãos e nos desenvolvedores de aplicações móveis dado que estes últimos poderão mais tarde modelar outras aplicações tendo em conta o ecossistema de dados abertos que criaram com esta arquitetura.

Acknowledgements

Admittedly, I'd much rather not extend my acknowledgements. However, the fact that I'm writing these words means that there have been enough influential people to allow me to reach this stage. For them, I'd like to give my biggest, most meaningful thank you.

Both my mother and my father, alongside my brother, need not have a personal note as they know how important they are for me and their essential role in my life.

For my girlfriend who withstood my constant absence ever since I started this project, but who'd still miss me as if it were the first day.

For my friends, Nuno, João and JP, thank you. Of course, it goes without saying that, I would have never been able to reach this much had I not had the most amazing fun times accompanied by the one and only Jota.

Finally, but not at all the least important note for the $\rho 7$ Boyz and RR himself, with whom I was together in a scientific battlefield, attempting to reach the Master's goal.

José Pedro Vieira de Carvalho Pinto

"Research is to see what everybody has seen
and think what nobody has thought."

Albert Szent-Györgyi

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

SLR	Systematic Literature Review
API	Application Programming Interface
DBMS	Database Management System
HTTP	Hyper Text Transfer Protocol
WSDL	Web Services Description Language
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
ICT	Information and Communications Technology
IoT	Internet of Things
MCS	Mobile Crowdsensing
EU	European Union
NGO	Non-governmental Organization
UMI	User Management Interface
MVC	Model View Controller
JSON	Javascript Object Notation
JVM	Java Virtual Machine
LIACC	Laboratório de IA e Ciência de Computadores

Chapter 1

Introduction

1.1 Context

The evolution of technology, in very distinct areas, brought a huge finite number of improvements to the world. One of the most noticing experiences generated by such an evolution is the globalisation and increase in information flow throughout the world. While helping people seek for a better way of life, the better the technology the more it gets implemented. This is what normally occurs in the major cities in developed countries, where we see the greats feats of engineering get used and tested by the general population. This scenario generates an increase in population in urban areas, competitiveness between cities and the challenge of maintaining a sustainable and attractive technological development environment. The concept of *Smart Cities* slowly gained popularity being defined as a city that invests in human and social assets, traditional and modern communication infrastructures. This translates to a sustainable economic growth and a high quality of life through participatory governance [Dea12, SKP+11].

Quite a lot can be done by either the government authorities, public and private businesses and citizens to perpetrate the Smart City vision in terms of creating infrastructure, technological foundations and the provisioning of services and applications in a sustainable fashion. Amongst all the different ways of contributing to the community, this document focuses on solutions that tackle

issues such as Open Data, Interoperability between systems and Citizen-driven Innovation enablers. Successful cases are looked upon, analysed while underlining the necessary requirements and conditions that leads those initiatives to prosper.

1.2 Motivation

With all the buzz created around the idea of having a Smart City, we stray further away from a data model, still implemented in most cities today, where we have a unilateral flow of data. Data producers such as governments and private sector organisations generate greats amount of information that might never reach a public platform, thus never available to the majority. Not having a two-way flow, prevents any existence of an Open Data Ecosystem.

Lack of incentive from government authorities and major industry actors decreases the sustainability and long-term existence of currently available applications. The result is an unstable market of services and applications which are of unpredictable nature in terms of supply. Without an offer that is both with good quality and quantity, it produces an environment with low-quality data, decentralised accesses and no absolute standards or innovation guidelines.

Mobile applications, in the context of *Smart Cities*, are the number one method of retrieving citizen information and the most probable source of innovation given the current direction of today's technological evolution. It is important to mention its marketplace because it is an area of extreme competitiveness. There is little room for successors to the top applications, innovation and overall room for new and rising applications. This is a factor that shortens applications' lifetime, reduces visibility and sustainability. Especially for low budget initiatives that depend highly on app markets for their distribution and business model.

The development environment described above can be visualised in Figure 1.1 (p. 3). It produces this kind of scenario where there are applications/services, being a product of either government, third-party businesses or citizens, which maintain their information and generated data to themselves.

Without incentive to do so, developers won't have attractive opportunities that make it viable as to implement mechanisms of communication to other applications or systems.

1.3 Goals

The main goal of this dissertation is to develop an application/framework that enables the creation of an Open Data Ecosystems within a Smart City. The framework tries to solve issues of interoperability and communication between all data generating and/or consuming devices while promoting Open Data applications/services and Citizen-driven Innovation with public ease-of-access methods to all stored information.

Figure 1.2 (p. 4) illustrates an expected scenario given an implementation of the aforementioned framework. While maintaining their information however they see fit, the application provides some sort of copy of that same information to the framework which will then be made public

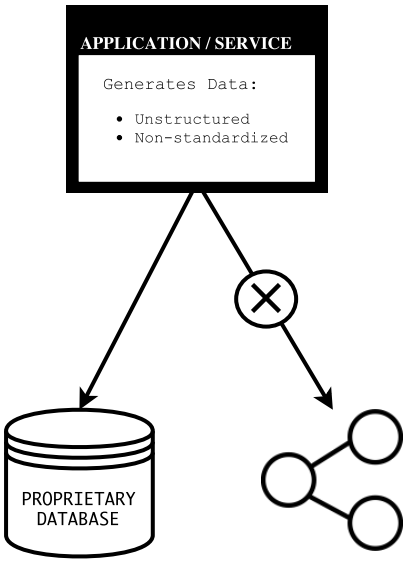

Figure 1.1: Data-flow of services and applications within a faulty development environment. In this scenario, data in not shared in any way only remaining in the possession of the owner of the application.

and accessible to other entities to make use of it. Applications providing data to the framework should be able to access their own content. Meanwhile, the framework should also support retrieving only, as a use case, an application that only retrieves a specific dataset without contributing directly information to the database.

Having various sources of content fed to the framework and turning it publicly available may also, contribute for situations outside the context of *Smart Cities* where there could be other possible uses. An Open Repository is sure to enable other causes and at the same time encourage Citizen Engagement.

1.4 Expected Contributions

After designed and implemented the proposed framework which, it is assumed, meets its methodological goals, it is expected that a completed framework to have contributions at several levels:

(i) Provided an Open Data enabling framework accessible in the format of an Application Program Interface (API) available to any data generating or consuming device;

(ii) Provided a tool to potentially solve interoperability issues with Smart City communicating devices;

(iii) Created a framework that allows for other tools to be implemented that make use of its collected information;

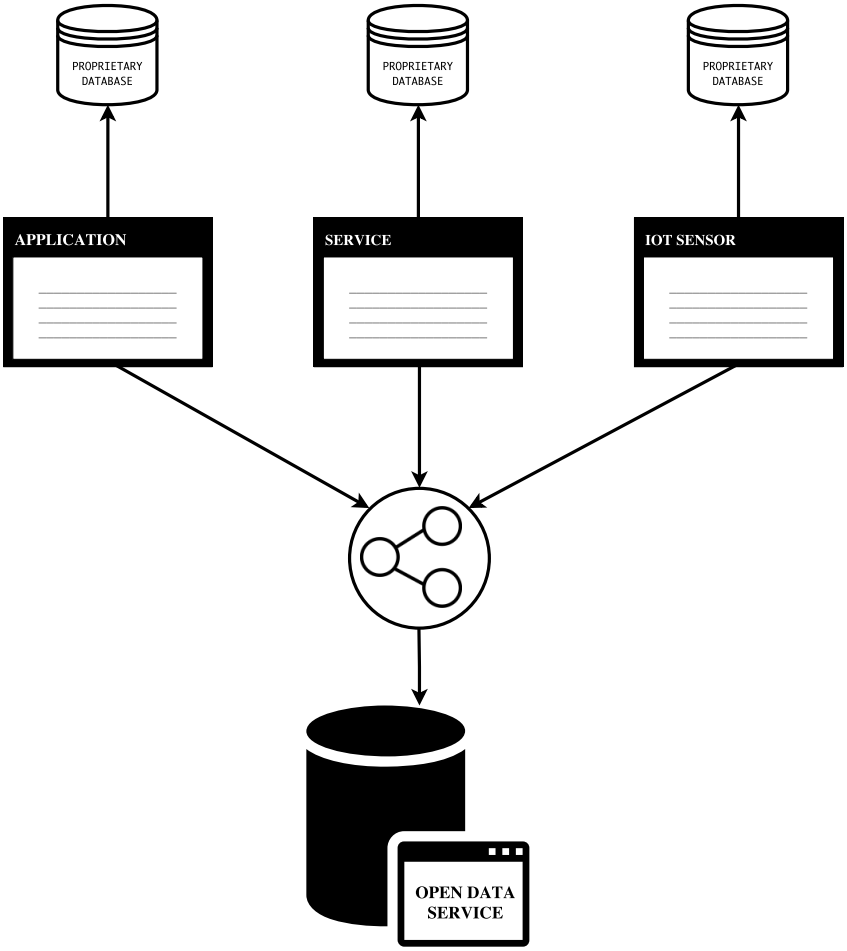

Figure 1.2: Expected scenario for applications using our framework. Data from each application is now shared with an Open Data service as well as remaining on the owner's database.

(iv) Created an incentive for individual and collective action towards a better Open Data Ecosystem.

1.5 Document Structure

In this section, there is an overview of this dissertation's main elements and what is expected from them. There are 5 chapters, starting with Chapter 2 (p. 5) which presents the literature review. It covers the relevant concepts of this dissertation, executing in-depth analysis on each topic. The chapter ends with several examples of real-world applications of related work alongside a more detailed approach to three specific cases. As a means of better explaining the proposed solution for the issue mentioned previously, Chapter 3 (p. 29) contains a brief contemplation of the main problem being tackled throughout this project and major adversities to be encountered. It also demonstrates an overview of the concept which is intrinsically explained in Chapter 4 (p. 41). This chapter contains, as detailed as possible, information about the architectural design and implementation. Completing this document, conclusions, future work and a few final remarks are exposed in chapter 6 (p. 61).

Chapter 2

Literature Review

By looking into the subjects of *Smart Cities* and Interoperability, it is intended that to understand the point of view of other authors in what takes for an initiative to succeed in a Smart City contemplating major strengths, points of focus and issues to avoid. A study on the different types of interoperability and assessment of the optimal conditions that ease interoperability problems is conducted as to better approach to same ones on this dissertation project. Lastly, a look at various instances of related work and of the real-world impact of applications, worldwide initiatives and citizen-driven innovation projects.

2.1 Interoperability

Competitive markets and technological growth have forced, in some situations, collaboration between large enterprises. This takes on an aggressive role towards a greater economic value for all the parties involved. This collaboration is done at several levels, such as logistical where organisations might make use of the others' infrastructures. Here the focus is on the exchanges at information level, namely data interoperability [Bla05].

First off, interoperability is the capability that two or more systems/applications have when they can exchange information and successfully make use of that same information within itself at data, application and business levels [Bla05]. However, the topic on level of interoperability straits a more complex path as there are several interoperability levels, from none to a complete level of interoperability. Muguira *et al.* [Mug03] defines five levels of interoperability where the definition of each level focuses on the data that is going to be interchanged and the interface documentation that is available. From level 0, which specifies that there is no interoperability

between two systems, to level 4, where there is a semantic connection between data. The desired interoperability level on this dissertation work is at level 2. This level specifies the existence of a reference model which documents the data. This is possible by using metadata standards or by using a standard reference model where the component, which is interchanged, is a black box with a standard interface.

Heterogeneous data creates issues in terms of different entities using different standards of data, semantic inconsistencies when having attributes on data with the same name but different meaning, and to solve issues such as these and generate interoperability between systems, there are a few approaches worth mentioning. The following approaches all solve, to a certain extent, some part of the communication problem, but they all tackle problems such as costs in maintenance and human resources that each has while trying to reduce its own complexity.

2.1.1 Data Standardisation

Applying standards to data and storage is, in theory, a way to completely cleanse problems with interoperability using the same structures, representations, names, etc. for the same systems. But what is observed, is that applying a standard model that covers every different data type, requirement and system is most likely impossible and also very challenging to maintain [RSR96].

As one might try and implement a single complete model as a data standard, visualised in Figure 2.1 (p. 7), its success is more likely to be seen on small organisations rather than larger ones, and as companies grow, so do its number of applications and their complexity introducing heavy adjustment costs. In this situation, application of several standards broadening fewer databases and systems is a more optimal choice since changing or adding a system does not imply modifying the model that deals with every component. However, interoperability problems will rise since communication still occurs between different standards [RSR96].

When mentioning this type of interoperability, consistency is key, but here systems/applications can, of course, not all apply the standard recommended which leads to complications. Other complications surge as is it time for an update on the standards, businesses change and markets evolve into an environment which forces these changes. Modifying a model means that the systems using it must also be adapted to it and this time-frame creates an inconsistency where we might come across communication between systems using different types of data (ones following the updated standard and ones which have yet to adapt to it) [GMS94].

The notion of standardising data is simply defining all the concepts on semantics and representation of every element possibly present on data. With this, all systems have knowledge of the properties of each element without the need for translations which facilitates data communication [LNS+07].

When specifying all the concepts of a standard, there might be cases where there is a need to define more than a single representation or semantic value for the same attribute. Case in point, systems that measure the same value with different precisions mechanisms. Solutions to some of these encounters are, defining an extra attribute for each system that has a different precision value

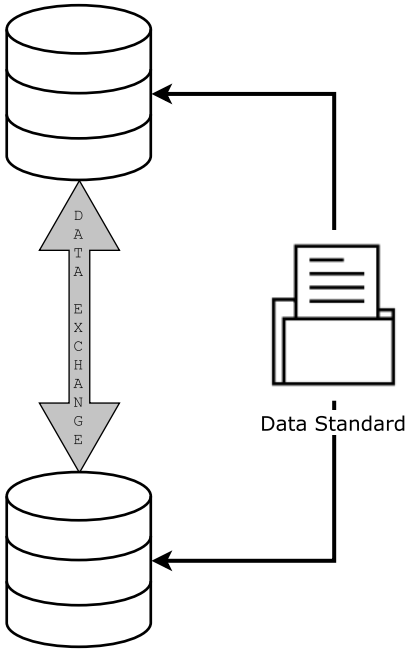

Figure 2.1: Data standard applied as a single model in an ecosystem of communicating components. Both database systems were built following the same data standard, *a priori*.

read, reduce the precision value to the lowest of both or fully define all precision levels within the context. As a means for standardization another focus is found with the use of ontologies [SR15].

These scenarios are far from optimal and all pose great disadvantages in terms of confidence of the read values and quantities of storage used by addition of new elements to resolve the issue, causing a turn to a more relaxed approach, away from the fully specified data, on section 2.1.2 (p. 7) [RSR96].

2.1.2 Data Mediation

Mediation is another approach that tries to solve the problems that come with interoperability. In this form of communication, we'll have a mediator, or a component, which stands between the receiver and sender of data, translating the queries being made by the receiver and adapting the responses generated by the sender [RSR96].

A receiver will query the source or sender system. This query will go through the mediator which will translate it against the source database and act a semantic adapter for all the terms contained in the query, which can then be interpreted by the source. The mediator contains a schema specific to this receiver and it is used to convert the query to an equivalent one but which is able to be interpreted by the source system. The source will then, having a format adapted query, execute it and, after grabbing the results from a specific database, send them back to the mediator, whose role is to translate the results back to the format specific to this receiver [RSR96].

The mediator acts as a gateway for a system to place a query on another system, for either to retrieve or send information, which is encapsulated. When a response is sent, the initial system,

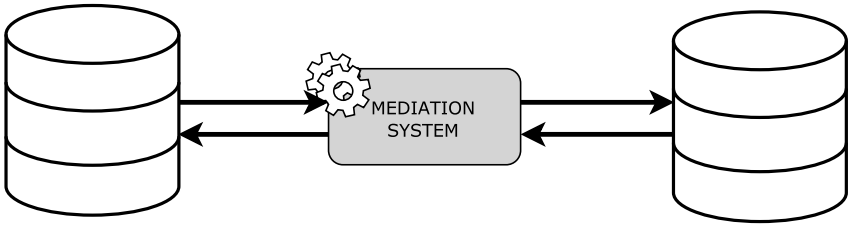

Figure 2.2: Interaction between two systems using a mediator for the data exchange. Queries from one database system to the other first goes through the mediator, as well as their responses.

from its perspective, receives it as a response from a database adapter and not another system, as shown in Figure 2.2 (p. 8).

Contrary to the standardisation approach which has a hard-coded vocabulary at which to look at when building a system storage or communication components, this approach does not enforce any systems to change their outer data layout, it will only have them produce descriptors for their attribute translation and schema which is the semantic adaptor for the mediator to use for the query formatting [CHS91, SAKR15].

As complex as systems tend to get, especially in large enterprises, many times being composed of several types of Database Management Systems (DBMS), the use of the mediator approach can bring a big advantage in terms of access to several of these DBMSs. By creating an abstract layer that connects all the mediators of the different data source systems and making it seem like the user is accessing a local database when instead he is really accessing the global spectrum of data sources which makes use of the semantic gateways to communicate with this abstract layer. An example representation can be seen in Figure 2.3 (p. 9).

The schema, aside from serving as an adapter also introduces the possibility for meta-attributes that alongside the actual attributes will describe them and specify in a natural language the meaning of each by exposing measurement values, units and semantic interpretation for the human developer [SSR94].

2.1.3 RESTful web services

Communication between devices has been one of the most volatile areas of the web with the evolution of web services since protocols such as Hyper Text Transfer Protocol (HTTP), which was first designed to serve as a communication protocol for human-to-machine but is generally used for machine-to-machine only [DB04a]. Defined as software systems that enable communication interoperable between machines over a network [DB04b], these systems use protocols such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and also, the focus of this chapter, Representational state transfer (REST).

A RESTful web service is a system that communicates with another system over a network requesting access to web resources and handle their textual representation, these opposing systems have to be REST-compliant permitting other machines to request web resources using the REST's protocol predefined, uniform and stateless operations, defined as HTTP verbs such as GET, POST,

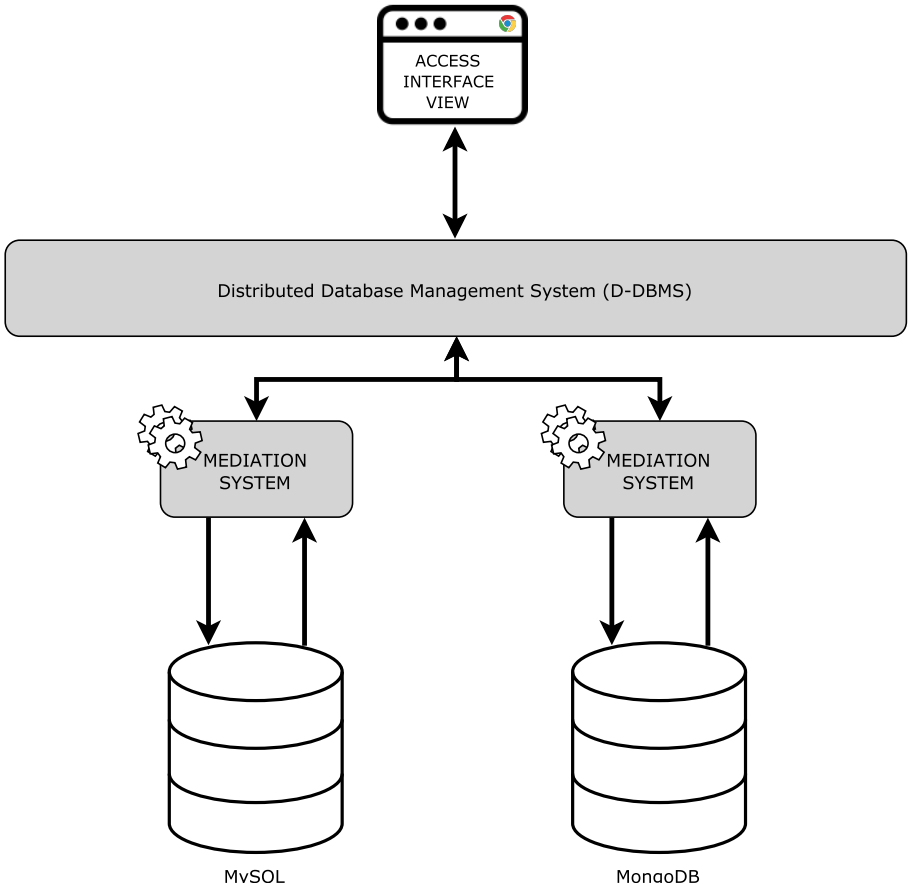

Figure 2.3: Global layer connecting all DBMS systems of an enterprise through the interaction with their respective mediator modules.

PUT, DELETE [Fie00]. The usual scenario for a request using a RESTful web service can be seen in Figure 2.4 (p. 10) showing the web components and infrastructure.

The REST protocol was itself a means to tackles many of the, at the time, limitations of environments on which APIs were deployed on [MMY09]. Compared with REST, approaches using open standards as SOAP, WSDL and others that completed the traditional WS-* protocol stack, lacked in the following aspects:

1. APIs built, exposed only a single point through POST method, restricting important information by containing them inside internal processes.

2. With the use of several services, several interfaces were needed.

3. Resources are not identified using URI making harder to grasp useful information from communication logs for support or debug purposes.

Issues like the above along with others made it so that REST was introduced, handling a panorama of topic including [MMY09]:

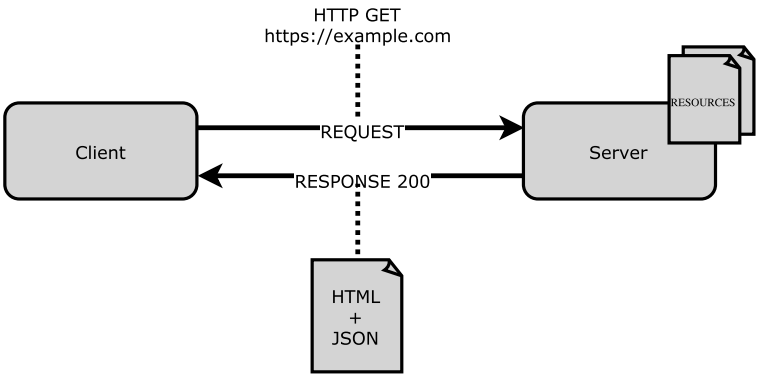

Figure 2.4: RESTful web service request environment. An HTTP GET request is made, being responses with code 200 and respective resources.

1. Addressability - Once a resource is exposed since it uses a Universal Resource Identifier (URI), it won't need an external service such as UDDI for its location services.

2. Connectedness and Links - Although stateless and because resources are connected together using hyperlinks, is it possible to represent a state through these. Following the hyperlinks also allows a state transfer.

3. Statelessness - There is no need for the server executing the request to keep any kind of state since each RESTful request comes packed with all the essential information to make itself totally independent

4. Uniform Interface - As previously mentioned, this protocol defines its resource methods' access through HTTP verbs that do not require extra code handlers

The ability to encapsulate legacy systems, deploy independent services and impose security measures while decreasing latency between interactions makes RESTful web services very much capable of solving the current scalability and interoperability problems of today's Internet distributed data applications. Contrasting with the more traditional WS-* protocol stack better suited for enterprise computing environments, REST's flexibility, high performance and simplicity cover emerging issues and is more likely to expand in use while complementing other approaches.

2.2 Smart Cities

As mentioned on 1.1 (p. 1) the definition for Smart or Smarter Cities is given, now reiterating that same idea by complementing it with the following. By settling on where the necessary efforts needed to be deployed on to a city which will translate to the benefit of the community as a whole, in areas such as health, environmental sustainability, energy consumption, communication and transportation infrastructures, innovation platforms we can better define a Smart City [Dea12, Hol08].

Since every community is different, its development should start with the people. As many cities nowadays rush to loose the pressure on becoming a Smart City by implementing the means for quality Information and Communications Technology (ICT), which is assumed to transform a city into smart one, this notion is largely wrong [Hol08]. As important as ICTs are, given that it is the underlining factor for the successful accomplishment of many, if not all, of the aspects considered fundamental for a Smart City [CDN11], it is the interaction of the cities actor components with the ICTs and provides core value and make it so that it actually becomes a Smart City. Components such as the people, which can make use of ICTs to become empowered and educated citizens with the ability to create disruptive motions on a city development and innovation environment [CDN11, Hol08, BG05].

Towards this flow on economic expansion, there is the challenge of equalising the capacity of the systems that run the cities necessities. The remainder of this section will follow on the major challenges in *Smart Cities* as well as focusing on topics relevant to this dissertation within the same context.

2.2.1 Current Challenges

Having evolved from the traditional definition of the 90's on urban development directed at globalisation, innovation and technology [GKS92], *Smart Cities* have a slightly dissimilar meaning according to The World Foundation for Smart Communities where they acknowledge that one must make satisfy the demands of the city within a global knowledge economy [Sma15] although recently it has shifted towards maintaining or simply providing sustainability in accordance to the rise of internet technologies [SKP+11].

As effort goes into utilising the latest technology *"in mobile and pervasive computing, wireless networks, middleware and agent technologies as they become embedded into the physical spaces of cities"* [SKP+11] with the goal of better managing and forecasting urban developments, the first encountered challenge is the prioritisation of development choices of the city within the innovation world [CRHPP09]. Following this, issues such as active labour market policies which decrease the risk poverty and helps to sustain employment, sustainable environmental development mainly targeting greenhouse gas emissions and maintaining competitiveness for the areas of research innovation and enterprise by providing means of increasing the population skills and knowledge, have been conspired as top priorities by the European Commission referred in their most recent public consultation [C+08, SKP+11].

In order to address these mentioned topics, government and general public authorities strategise means of developing business models that would integrate both physical and digital worlds with the ultimate goal of establishing long lasting sustainable partnerships creating an environment promoting e-services and creative applications [SKP+11].

2.2.2 Citizen-driven Innovation

Schaffers *et al.* [SKP+11] refers to 3 major tasks that a city can carry through in order to become a digitally sustainable Smart City. This makes a community with sound groundwork on which every economic activity is seen as an innovation ecosystem that can operate under the active participation of every citizen and every organisation whose role can be of either development, provision or consumption of goods and services.

These tasks are respectively:

1. Development of structural and technological basis;
2. Kick-start the participatory innovation process;
3. Creation of sustainable business models and open ecosystems.

On a first step, Schaffers *et al.* staple an infrastructural foundation by establishing that a Smart City's digital sustainability comes from the development of broadband infrastructure which connects citizens and organisations within the geographical area of the city supplying quality connectivity and bandwidth followed by a physical enrichment of sensors and embed systems on infrastructures. Lastly, making it possible to reach out for all the data created by the latter mentioned infrastructure and devices, by providing the appropriate interfaces which enable all types of data processing and joint user collaboration.

With the basis complete, the second task dwells over each sector of activity and evaluating themselves in order to create the appropriate application for it. Schaffers *et al.* also illustrate what they consider being the key sectors or areas of application, namely innovation economy, city infrastructure and utilities and finally governance. Alongside Internet of Things (IoT), there are several opportunities that arose due to research on future media and technology mainly for content management, together with a merge of context-aware information as well as sensory and user dependent data, such as location, the possibilities for innovation thrive, possibly adding value to each of these sectors and also leads to the creation of e-services. These are the direct consequence of when an application for an utility or activity of some sort eventually reach a stage of sustainability and can be seen as a success, making it possible for it to transition its business logic to an e-service, broadening availability and usage [FAU+10, NMF+10, SKP+11].

Applications and e-services should surge from the second task making it possible for this third one, in short, sustainability. The authority officials should intervene in order to take up the newly forthcoming services and create business models around them that sustain the long-term functioning of the city. Successful creation of such business models has, for the most part, been a rather complicated process given the lack of means of retrieving usable data, although these same means introduce the issue of management of great amounts of information from various sources and formats. Schaffers *et al.* don't detail on solutions to this problem but as thoroughly explained in 2.1 (p. 5) we see the many approaches towards solving the situation where there

lack of interoperability. The presence of Open Source communities can also turn out to be a very good asset to the city given that it often contributes not only to the existence of better quality applications but the interchange of good practices leading to open solutions that are of very great value to the environment created in the development of applications and services.

The current scenario on *Smart Cities*, within the European Union, is still a long way before it reaches a state where it can show its real impact. With interchange of mechanisms between different business models and trade of development techniques between major cities we can fundamentally obtain better solutions. *Smart Cities* still have to solve their biggest problem which is to maintain the high living standards throughout the innovation economy that is ever-growing, however, this citizen-driven approach constitutes a good chance of thriving.

2.2.2.1 Data Collection through Crowdsensing using Mobile Applications

When we have the mass production of technology, a direct consequence is its price being lowered as it becomes popularised in the economic markets which have led to the creation of innovative devices such as wearables and great advances in mobile devices whose native and sensing capabilities have skyrocketed in terms of both power and creativity. The process of collecting data can be done in various ways, here will focus disperse, distributed and non-fixed infrastructures which help and pioneer this same process, looking at the concepts of crowdsensing and mobile applications adapted to an environment of *Smart Cities* [CFB+13].

As mentioned in Section 2.2.2 (p. 12), *Smart Cities* should have the proper environment so that using the foundation it has created promoting innovation and citizen participation, it can take advantage of its technological efforts as a means of instaurating and aiding the creation of new applications and forms of data intelligence. There is the possibility of *Smart Cities* utilising crowdsensing, which briefly explains the concept of collecting data by splitting the responsibility and power of collection to a large number of people rather than one or more fixed structures, often referred as urban sensing or monitoring [PPSR17]. In principle, this should bring great advantages to future applications and the creation of new services. Unlike the more traditional approach of infrastructure-based sensing solutions, crowdsensing brings up a very positive point, which is the fact that it does not need a fixed, heavy cost infrastructure deployment platform on which to rely solely as the data collection source. Nonetheless, both approaches complement each other filling different existing gaps [CFB+13].

Sensor capacity of embedded devices and their ever-increasing computing power eventually lead to the era of Internet of Things (IoT). We can have two categories of sensing, personal or community (or participatory sensing) [BEH06], their classification is based on the type of data being sensed. On one hand, we have personal sensing where the data being observed/retrieved matters to a single subject, not necessarily the person that is recording the information, where movement patterns, health attributes and habits and transportation choices are some examples of data classified as personal sensed data. On the other hand, community sensing, broadens a much wider array of data, classifies every other type which cannot be measured of a single individual, where we have phenomenon of bigger proportions which demand the sensing capabilities of crowds of people.

Some examples of community sensing are traffic and air pollution monitoring, crowd sentiment analysis and predictive analysis on markets' purchase trends [GYL11].

Given the value obtained by a single device that contains a variety of different sensors alongside with the connectivity sharing capabilities of social networks now possible due to the high capacity of mobile broadband networks, mobile phones and devices are the main driver of for embedded sensor systems and have defined an evolution stage in the area of data collection. As coined by Ganti *et al.* Mobile Crowdsensing (MCS) is the term that better defines this for it encapsulates "a broad range of community sensing paradigms" [GYL11, GYZZ14, SJK+12].

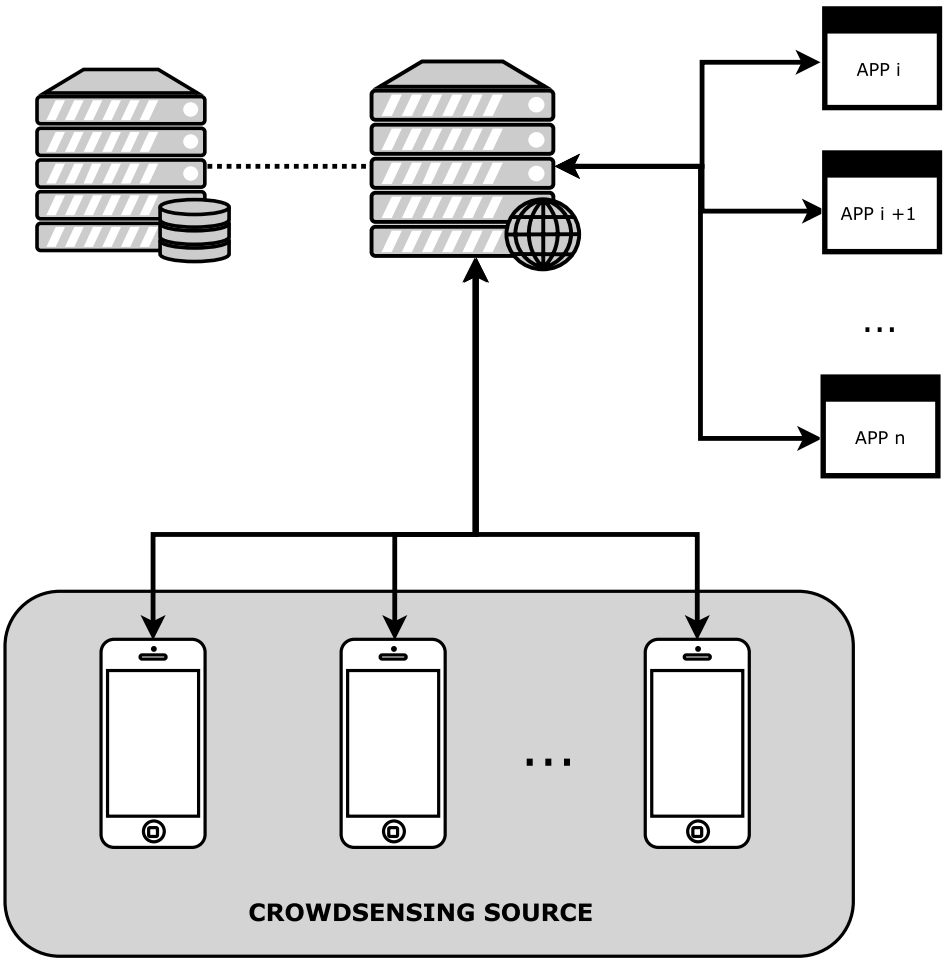

Figure 2.5: Crowdsensing using mobile applications. The mobile applications give use to the phone's sensor capabilities fuelling a database server.

2.2.3 Open Data Ecosystems

Building an active open data network that facilitates interaction and communication amongst citizens and organisations with the intent of reusing the information and data contained on that

same network is one major step in settling an Open Data ecosystem granted the sheer act of making data publicly available does not suffice.

"In the case of opening government data, advocates assume that simply supplying more and more data sets freely and in more formats will lead to more and more use" [Roy03].

By having data available in quantity and in format variety accompanied by services that utilise data in an appealing way makes the open environment attractive for a variety of entities such as Non-governmental Organisations (NGO's), private sector organisations and individual citizens [Roy03].

These are needed to foster the development of new innovative products and services while interacting with the government decision making [Roy03, Not]. Processes like these also go will into account towards the goals of the PSI Directive which is European legislation directive that provides a common legal framework for Europeans markets for government-held data (public sector information, PSI) and is built around two key pillars of the internal market: transparency and fair competition [Est, Eura].

Specified by the Project Open Data [Opeb], the accordant principals of open data are the following:

- Public - Government agencies should make the effort in reassuring data openness, embracing the concept taking in consideration any restrictive aspects such as privacy, security and confidentiality;
- Accessible - Open data should be made available in machine-readable format, using non-proprietary data formats, catering the widest crowd possible without restriction on its use;
- Described - Data is only as good as the use it is given. The use of descriptors in the form of metadata or any other kind of descriptive element helps the data consumer understand strengths, weaknesses, analytical limitations, security requirements and to process it;
- Reusable - All data should be made available under open license, imposing no limitations on its use;
- Complete - Open data should be published as pure as possible, directly from source, taking in consideration laws or other restrictions. If derived or in form of an aggregated collection, indicating the original source;
- Timely - Besides adding value to the data, making data available as close to the release as possible also facilitated real-time and downstream needs;
- Managed Post-Release - On-demand, a maintenance process that accompanies audiences' doubts on usage or inconsistencies should be made available.

"Knowledge is open if anyone is free to access, use, modify, and share it — subject, at most, to measures that preserve provenance and openness." [Opeb]

As an example of an Open Data ecosystem, urban IoT environments, fundamentally allow part of, or most of, their data to be publicly accessible to citizens and organisations, in order to promote participation, awareness about public matters and increase innovative services with the data collected [MO13]. We can observe another example on Mirri *et al.* work as they present a geospatial mapping service making use of OpenStreetMaps services which aims to cater the spatial needs of their users. Mirri *et al.* define their open data environment in order to provide real-time availability of all different public transportations options as well as routes, arrival times, facilities and possible barriers that might arise, getting this information through a combination of, previously privately held, information on public transportations by bus operating companies alongside crowd sensed data on urban accessibility related with the types of barriers and facilities by sensing activity on mobile phones of users [MPS+14].

This last example portraits a consequential result from having open data ecosystems where two means of data gathering and their combination which exploits an open data service from several private transportation companies and crowd sensing capabilities of city inhabitants' mobile phones, translating their merged data to another innovative service. Their work, now embedded on the city's open data ecosystem can now be used; adapted into different applications; used by other services and thus completing a feedback loop which fuels the whole ecosystem and allows it to properly function [Bui].

2.3 Related Work

On the matter of work related to this dissertation, this section analyses work that focuses solutions in the format of platforms, applications or whole software stacks which tackle problems in interoperability between systems and data, providing services over that information and/or promoting a participatory information open environment for Smart City's entities with long-term effects making available opportunities for technological innovation [40B].

There are more than enough interesting and relevant projects to mention but, for the sake of content length, all the projects will be briefly described followed by an in-depth analysis of only a few of them.

For better understanding, the projects researched were classified into the following categories, all of then fitting on the topic of open data with most fo them currently placed in action:

1. Navigation on Open Data Sources;
2. Transparency and Accountability;
3. Performance Management;
4. Transportation and Infrastructure;

5. Resilient City Planning;
6. The IoT of *Smart Cities*;
7. Civic Engagement.

2.3.1 Navigation on Open Data Sources

Open data portals and means of navigating open data tend to be quite difficult to maintain especially in terms of investments since these tools are expensive to build and constitute neither immediate nor large amount of return income. Here are some of web portals, mobile applications and platforms which counter these issues. They are the results of efforts done by city officials and third-party entities.

Dat [Data] Dat is an open sourced, decentralised data sharing tool trying to do for open data what Git did for source codes.

LocalData [Imp] LocalData is a mobile application allowing community groups to participate in open government data planning.

Open Data Impact Map [Opea] The Open Data Impact Map is a geospatial platform identifying communities around the world engaged in open data projects.

OpenGrid [Oped] OpenGrid, an interactive map platform. Residents can easily search curated datasets and visualise the results on an interactive map.

Resident Card [Tel] Tel Aviv's Resident's Card App allows residents to customise their interactions with local officials by opting to receive real-time updates from city services of interest.

SmartAppCity [Thea] Uses a Public-Private Partnership framework to encourage local businesses in Madrid, Spain to participate in city-wide open data initiatives by sharing their data in the city's open data portal.

2.3.2 Transparency and Accountability

As a means of reducing bureaucratic processes improving efficiency, here are some projects for that same purpose.

Clear My Record [Cle] A free web application that centralises criminal records in a repository shared among various government agencies in an effort to help citizens have low-level offences expunged from their records. The level of efficiency provided led the San Francisco Public Defender's Office to implement the application to optimise operational performance.

Commonwealth Connect [See] A mobile application that facilitates data sharing between government officials throughout Massachusetts. Residents can now report non-emergency issues that are housed in a data repository before being dispatched to the appropriate city or state agency.

DataHaven [Homa] DataHaven's mission is to collect, interpret, and share open data for better decision-making across Connecticut. Data visualisations, for example, are helping elected officials understand the interrelations among city and state services such as income-based segregation, which has caused significant urban structural strain.

InvestigativePost [int] Uses open data to expose higher than reported levels of lead in Buffalo's drinking water. Open data empowers citizens to hold local officials accountable.

OpenDurham [Dur] The City of Durham avoided creating a city-wide data silo by building a shared city and county open data portal. Now local and municipal agencies can work together to improve services for their constituents.

Openlaws [Homc] Staying informed about one country's initiatives, regulations, and laws is hard enough let alone for each member state of the European Union (EU). Openlaws allows EU residents working across national borders to sign up for mobile updates on latest information and changes to ensure all legal requirements are met.

OpenStates [Opec] Provides a resource for United States (US) citizens to research upcoming local legislature as well as compares legislative proposals to other states.

2.3.3 Performance Management

Keeping users informed of their rights is important is just one step towards better citizens management, open government data can help with that.

BOS:311 [Bos] This web and mobile application lets residents report non-emergency issues to the city's Constituent Service Center, which dispatches a request to appropriate agencies across Beantown.

ChattaData [Cha] ChattaData's High Performing Government Dashboard provides progress reports on initiatives to cultivate a more inclusive city. Setting a goal to increase Diverse Business Entities (DBE) participation by 14 percent, and fair and equal access to procurement opportunities.

Residential Typology Analysis Tool [Datb] A geospatial tool using cartographic and satellite images, to identify high and low residential density areas, which help city planners prioritise data smart renovations.

NOLAlytics [NOL] A scorecard tracking progress data-driven projects across New Orleans, not only ensures clear communication between city services, but also encourages interdepartmental collaborations on more open data project with NOLAlytics.

Urban Spatial [Pro] Urban Spatial built predictive models that pinpointed buildings in need of city inspection and created a data visualisation to communicate these findings.

Young Europeans [Eurb] Uses open data from Eurostat to provide a demographic sketch of EU youth ages 16 to 29. This web application is a tool for youths throughout EU to stay informed on economic opportunities in neighbouring countries.

2.3.4 Transportation and Infrastructure

Transportation and infrastructure make up a city's lifeline, explaining why there is such a heavy focus on these two areas and why the innovation generated by them often see the biggest impact.

CoAXs [Job] Job maps for Boston, Massachusetts and Los Angeles, California with the Census Bureau's Longitudinal Employer-Household Dynamics open data. Locating job growth across a city in order to prioritise Smart City projects encouraging further economic development.

Citi Bike NYC [Cit] The largest bike-sharing system in the United States, provides open data on riderships, routes, and individual bikes.

Digital Matatus [DIG] A cross-sector solution to the problem of limited government resources. Developers and private citizens in Nairobi, Kenya are helping local officials standardise a city-wide transit system by collecting location data with mobile devices to establish an open data network.

MARTA on the Go [MAR] A mobile app providing commuters real-time service conditions for bus and rail transit throughout Atlanta's Metropolitan Area. MARTA is assured to build "livable" communities in areas with limited access to public services by building transit stops to encourage urban renewal in these locations.

Red Eléctrica de Espanã (REE) [CAR] In compliance with the City of Madrid's open government data initiative, Red Eléctrica de España (REE) created an open data platform to discern power supply usage across Spain. Subsequently, residents can now monitor electricity usage and explore its economic impact on their local communities.

Smart Columbus [Smab] A holistic approach toward *Smart Cities* proposed by officials in Columbus, Ohio. Transit centres will be built in areas throughout Franklin county where open government data reveals alarmingly high rates of infant mortality that exceed the national average.

2.3.5 Resilient City Planning

Better information channels on issues as the environment have helped create strong partnerships with cross-sector on open data and prepare cities for complex environmental challenges.

BINGO [Aboa] Portugal's Laboratório Nacional de Engenharia Civil is leading an international coalition that is leveraging open data from EU member states to improve water management resources. SmartWater planning is a growing area with the field of open data and Smart City planning.

Level Alarm [App] A free platform featuring mobile emergency response app alerting residents about rising water levels in nearby bodies of water across Europe.

National Operational Assessment of Hazards (NOAH) [NOA] NOAH is the Philippine government's disaster mitigation system that measures water and rainfall levels to identify areas exposed to high risks of landslides, flooding, and other health concerns related to water contamination.

Partnership for Resilience and Preparedness (PREP) [Ena] The Partnership for Resilience and Preparedness (PREP) is an international partnership dedicated to making climate data accessible and meaningful for non-specialists to promote resilience planning.

Safecast [Saf] Following the devastating tsunami and subsequent nuclear power plant meltdown in Fukushima, Japan in 2011, a global network of volunteers created Safecast. This geospatial web and mobile application visualises open environmental data to provide accurate information to residents at risk of radiation exposure.

2.3.6 The IoT of Smart Cities

Recent efforts to improve Smart City grids and networks has led to the creation of sensor networks capable of monitoring a range of issues impacting residential living conditions. The Internet of Things (IoT) has enabled local officials to connect various devices through the city to this type of sensor network.

Autolib [Par] As an intelligent public car-sharing service, Autolib's fleet of electric vehicles harness a constant stream of open data from kiosks and sensors around Paris not only to optimise commutes, but also help in reducing air and noise pollution.

Array of Things (AoT) [Arr] The Array of Things (AoT) serves as a "fitness tracker" for Chicago. But instead of monitoring individual residents, sensors monitor the city at a macro-level to ensure privacy rights remain intact. The data collected from AoT sensors will be made open to encourage developers to create more applications aimed at improving living conditions for Chicago residents.

RideKC Streetcar [KCS] Kansas City is piloting an Internet of Things (IoT) corridor along the RideKC Streetcar downtown route as part of an energy conservation initiative. Local officials are installing interactive kiosks with real-time updates on service conditions, free WiFi hotspots and a responsive sensor system to monitor traffic flows so as not to waste energy during off-peak hours.

Smart Nation Singapore [Smac] Singapore is aiming to become the world's first "smart" nation thanks to its open government data portal, which also functions as a data repository for collected public sector data. Singapore's definition of "smart" has less to do with "technology" and more with how well society uses technology to address social inequality.

Smart Benches [Smaa] Smart benches located throughout London are built with sensors to monitor and collect open data on traffic volume and environmental conditions. At the same time, these smart benches provide more access to WiFi hotspots and solar-powered charging stations throughout London.

Urbanflow [Urb] Urbanflow has situated responsive, "living" screens throughout Helsinki projecting an interactive city map overlayed with a wealth of information to keep residents and visitors informed on local events.

2.3.7 Civic Engagement

Civic Engagement is a topic with immense potential but with still very few application using open data, relative to other topics. It is capable of giving power to the citizens for them to contribute and improve their communities and cities.

BA Accessible [BAA] A mobile application for residents of Buenos Aires. It is a crowdsourcing tool that provides accessibility information for spots around the city. Not only does this inclusive project provide valuable information for residents with impaired mobility, but it also alerts local officials to areas that need to be made handicap accessible.

The Displacement Alert Project (DAP) [Theb] The Displacement Alert Project (DAP) exemplifies how location intelligence can be used for public good. The Dap.Map is an interactive, web-based data visualisation built with open data provided by the city, state, and national agencies.

GetCalFresh [Get] A web and mobile application for California residents in need of food assistance. Residents can apply electronically for assistance, which helps ensure resident's needs are met in a dignified manner.

Mapping Police Violence [Map] Open data does not always present the full picture. The Mapping Police Violence project is a timely and important initiative demonstrating the need for even more transparency from open government data as it leverages data from various sources to address and redress racial injustice across the United States.

2.3.8 Highlights of Related Work

As it can be seen above and throughout the whole chapter of the Literature Review 2 (p. 5), there is an enormous amount of projects and effort being put on contribution to open data and

Smart City applications. Focusing the following 3 projects, it is intended to perceive the impact of this kind of work on different areas with different approaches.

1. FIWARE Platform and Community;
2. Smart Nation Singapore;
3. SmartAppCity.

2.3.8.1 FIWARE Platform and Community

Supported by the Future Internet Public-Private Partnership (FI-PPP) project of the European Union, FIWARE constitutes an open cloud-based infrastructure for the acceleration on SME and Start-ups their development of innovative services and applications using its own technologies. For FIWARE the role developers have is crucial, with the involvement of users, in order for the platform and its projects to become both standards and reusable solutions for the deployment of Future Internet applications and services in various areas such as *Smart Cities*, sustainable transport, logistics, renewable energy, and environmental sustainability [FIW].

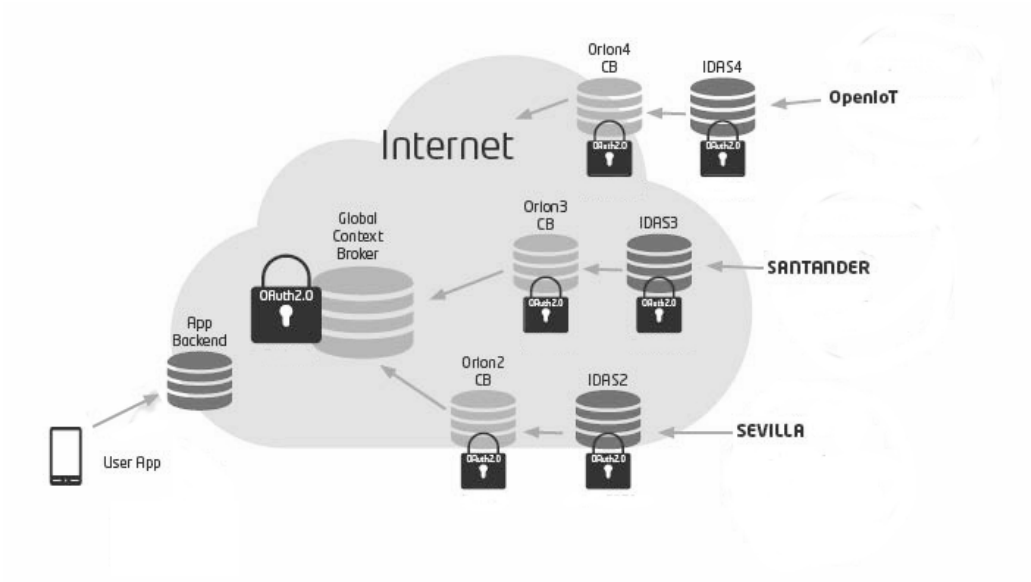

Figure 2.6: Typical interaction of FIWARE components in an IoT scenario.

The platform exposes a wide catalogue of components denominated Generic Enablers (GE) spanning a variety that covers Data/Context Management, IoT Service Enablement, Advanced Web-based User Interface, Security, Network and Devices and Cloud Hosting. These serve a general purpose and functioning through the well-defined API specifications of FIWARE with the addition of promoting good architectural practices given their internal implementation and usage. Everything about theses GE API's is public and royalty-free, both their reference and architecture and also alternative implementations are open source [Homb].

Together with the large FIWARE community of people, both developers and users, the whole initiative also gathers an academy, tutorials, very well-defined and open Data Models and most important the FIWARE Lab which aims at providing an environment that enables real testbeds for developers and companies to test their applications and services with access to the whole open data available to FIWARE throughout Europe and the rest of the world [Homb].

Figure 2.6 (p. 22)depicts a scenario in use on a FIWARE Lab having three actors OpenIoT, Santander, Sevilla and a User App. In this IoT testbed, the 3 first actors are sharing their data with the User App utilising FIWARE components, the GEs. By using the IDAS, Orion and Global ContextBroker components these 3 actors have a seamless integration with each other and the User App without having to develop any extra software modules, saving time on development while also implementing good architectural practices [dev].

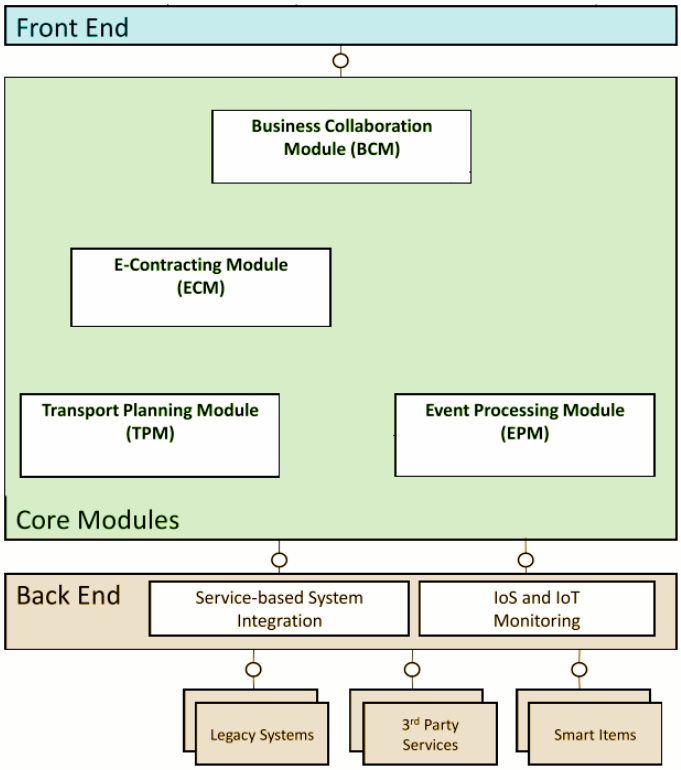

Figure 2.7: FInest platform high-level conceptual architecture [FMS+11].

In [FMS+11], Franklin *et al.* demonstrate a setting where they make use of FIWARE practices and components. They go through with building a platform, as conceptually shown in Figure 2.7 (p. 23), that serves the transportation and logistic industry named FInest. By taking advantage of the rapid development and cost-efficient perks of using Future Internet components, they tackle their generic implementation problems using the FIWARE GE and implement themselves the domain-specific services for transportation and logistic domain.

Figure 2.7 (p. 23) demonstrates the FInest platform Front End for end-users' interaction and monitoring of services, the Core Modules which contains most of the platform's domain-specific

knowledge and ending on the Back End that handles interaction with legacy systems and outside services and data.

Franklin *et al.* use the following Future Internet's resources:

- Cloud-based platform and application development tools;
- End-user service support tools;
- Service description language to cover both technical and business requirements;
- Integration and technical handling of sensory data tools, data registered during the execution phase of transport and logistics processes;
- Large workloads handling and event processing modules;
- Full use of a security and privacy management integrated framework.

With such a large base of GE in use, the authors were able to cover every non-domain-specific implementation, rapidly deploying a platform that would otherwise require development all those modules, integration testing and long-term maintenance.

2.3.8.2 Smart Nation Singapore

Island with no natural resources, boarding the technology evolution like no other country did. Defining itself as a Smart Nation, Singapore has a government that was able to shift completely, embracing Information Technologies, kick-starting a massive program to redefine its infrastructure keeping in mind the quality of life of inhabitants and the necessary economic growth that would make Singapore not just a smart city but also a quality standard [Mah15].

The Smart Nation program's goal is to ultimately improve the lives of the citizens of Singapore. The first step was identifying the key domains on where to engage, and that could be deeply impacted by digital technology. The main identified domains are Transport; Home and Environment; Business productivity; Health and enabled ageing; Public sector services. On these domains, looking at public and private businesses and city inhabitants, it is expected to create potential solutions for the domain-specific challenges. To serve as a background facilitator, the government provides infrastructure, resources, policies and necessary motivation for this cooperation to happen, enabling an environment of experimentation, dependent on a risk-taking initiative with the purpose of generation innovation [Abob].

"In our vision, some 15 years from now, Singapore, the Intelligent Island, will be among the first countries in the world with an advanced nationwide information infrastructure. It will interconnect computers in virtually every home, office, school, and factory (NCB, 1992)." - [Mah15]

Some examples of enablers for smart solutions are seen on initiatives such as the Smart Housing Development Board (HDB) Town Framework, which brings forward a reality still far in the

western households, residents of Yuhua estate will experience a wide variety of technologies applied to their homes on water and energy management systems. While helping these residents track and manage, in real-time, their costs and usage of resources, their data is used as feedback to determine the viability and sustainability of smart home entering day-to-day lives [Ho16, Abob].

The cultural aspect is heavily outlined throughout literature on this initiative as how much importance needs to be given to educating and empowering the people of Singapore in order to produce sustainable innovation [Mah15].

Government efforts were made in the following areas to help cultivate this culture:

Figure 2.8: Typographical overview of existing testbeds in Singapore.

- Open Data - As mentioned in the above examples on related work on IoT projects, in 2.3.6 (p. 20), Singapore's government open data initiative and the Smart Nation Sensor Platform which is an infrastructure being built to enable greater pervasive connectivity, better situational awareness through gathering and sharing of useful data between agencies;
- Investment in R&D - Singapore's government has continued efforts in maintaining a strong R&D development by supplying 1% of its GDP;
- Living laboratory - Smart City testbeds and collaboration projects are a focused subject. Institute for Infocomm Research (I2R) and the Singapore-MIT Alliance for Research and Technology (SMART) are examples of collaborations that spawn a number of places for experimental projects to be executed and challenged, see Figure 2.8 (p. 25) for a topographic overview of existing initiatives in Singapore;
- Cybersecurity and Data Privacy - Cybersecurity is a key enabler of the Smart Nation initiative. The government recognises the possible risks and prioritised safeguarding relevant systems and networks that relate to the security of citizens and privacy of data. The government, industry and public must all play their part and take measures to safeguard data, and ensure that critical control systems are protected even as we make them smart [Abob].

2.3.8.3 SmartAppCity

After looking into a worldwide initiative as FIWARE,2.3.8.1 (p. 22), is, Smart Nation Singapore had a spot on analysis as a countrywide initiative. This section has a local project, in Madrid, Spain specifically, SmartAppCity which is also mentioned in section 2.3.1 (p. 17).

This application has been created with the intention of bringing together all services offered by the city of Madrid, providing real-time information. All this is possible by making use of an open data PPP (Public–Private Partnership) framework, that the city councils use to show their information, as well as shops and businesses which offer their products and services [MMD15].

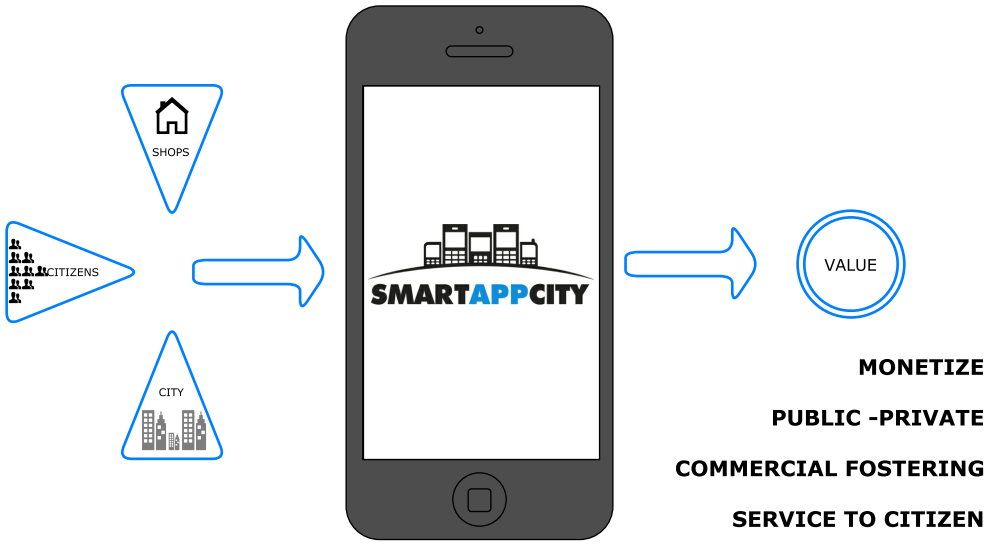

Figure 2.9: Diagram illustrating the application's data flow [Smad].

The idea of the proposed collaboration facilitates access to services, without having to resort to various applications to be aware of what the city offers, as seen in Figure 2.9 (p. 26).

After its local success on Madrid, a business model was created aimed at providing the same fashion of services in other *Smart Cities* by taking advantage of the modular construction, different packages can serve different cities with specific purposes. The application is then deployed and distributed through local partners.

2.4 Summary

The literature review exposed means of counteracting common issues of interoperability, as well as, going through the most important aspects of *Smart Cities* in order to understand the role of data, its impact and the requirements for its success. A look at the related work available demonstrates the impact that Open Data initiatives have on cities and their population, in some cases, affecting people at a worldwide level which is made possible in most cases due to an achievable communication between multiple applications.

Interoperability is one the major key issues that the purposed architecture is focused on and the level of that interoperability, which is enabled by this architecture is level 2 accounting for the

definition presented in Section 2.1 (p. 5). In this dissertation, the approach taken, which is thought to be the most viable one, follows standards rather than compiling a mediation system, by building a standard reference model which encapsulates data that is interchanged. The research done on the RESTful web services also justify the choice made on deciding to go for the REST communication protocol. Given its flexibility, extensibility and the being one the most used protocol in current times, an approach built upon a RESTful web service is more than capable of fulfilling all the necessary architectural gap in communication.

Chapter 3

Conceptualization Overview

The platform is not defined by only one element but by the collection of all 3 components and their interactions. As the conceptualization is explained in the next sections, it is important to keep in mind that this concept can be implemented in a wide panoply of scenarios, ranging from just municipalities to cities, to an entire country. Here the framework's internal processes and logic functioning is explained in a high-level overview, which does not go into any details, whatsoever, on technologies and their choices, often encapsulating component's behaviours. First off, the Problem Statement is addressed and contextualised.

3.1 Problem Statement

Smart Cities use innovation as fuel, maintaining sustainability that accompanies the global economic growth. This requires the consistent creation of long-term and self-sustainable business models. Both government entities and private sector already have such models, businesses that throughout the normal behaviour generate large amounts of information, which by itself cannot be taken advantage of. It is by providing the means of communication and solving issues on interoperability, which create barriers between organisations, can we generate other ideas and models that will help the city in its growth and evolution.

Another component that comes with great benefits are the citizens as there is great potential and opportunities hidden inside *Smart Cities*' communities. Lack of means of access to information does not allow the communities to explore their ideas and creativity.

For *Smart Cities*, its all about finding the right solutions for the right problems aided by the latest in technology and in the interest of harvesting and utilising data generated by governments and private sector organisations, which can be of great value if able to be explored. Open Data Ecosystems provide the tools to do so. Environments founded on the basis of Open Data constitute the instruments for citizens and all the other entities involved, that themselves also inject content for the platform, to seize and kick-start innovative projects utilising the data from every possible element of a Smart City, potentially solving domain-specific issues within their communities.

Government authorities invest heavily on ICTs such as building sensors networks throughout the city and private sector organisations themselves generate information. Allowing this content to be accessed by everyone constitutes an essential factor in fulfilling one of the fundamental goals of *Smart Cities*, which is sustained economic growth through constant innovation.

3.1.1 Emerging Issues

Right from the start, understanding the described problem statement raises questions which should be highlighted. These questions comprise some of the difficulties that are to be expected and that are tackled in terms of both architecture of the whole platform and its implementation:

- How to allow an exchange of information between different devices?
- What makes a platform an Open Data enabling one?
- How to design a self-sufficient application?
- What is required for data standardisation?
- How can value be added to publicly available data?

Throughout the following chapters 3 (p. 29) and 4 (p. 41), these questions are kept in mind as they define the critical points that go against the expected contributions for this dissertation.

3.1.2 Goal Statement

The main objective of this dissertation is to successfully contribute with a bridge between government, organisations and people. In order to fulfil this and the expected contributions mentioned in Chapter 1 (p. 1), the proposed framework is to be built by driving its development by the following guiding steps:

1. Establish a data standard for the information stored on the framework;
2. Develop a mediation method for information being written for it to become in compliance with the previously established standard;
3. Develop an Application Program Interface (API) for the framework, enabling users to *write* data;

4. Systematically analyse incoming information as means of maintaining, at all time, currently available information;
5. Add to the API a *read* functionality, allowing users to retrieve information from the platform;
6. Standardised read and write requests by making them follow a specific data schema;
7. Build an interface which caters the management needs of users using the framework.

Completing these guidelines ensures the foundation needed for a fully functioning platform for data interoperability, which also serves as an Open Data platform publicly available for all data producer or consumer actors within a Smart City technological environment.

Issues that might be encountered in the making of the aforementioned framework are detailed. By providing context on the target environment, emerging challenges are highlighted. Although technological issues and implementation details on architectural and design choices are not within this scope, the given guidelines provide a roadmap for the architectural conceptualisation of the platform initially defining its individual components.

3.2 Framework Overview

The aspect of modularity is essential for the framework to be able to reach its expected contribution. The definition of the nature of the interactions between each component needs to be clear and well thought out. Later, it is also discussed in more detail the aspect of modularity and how to translate it into a layer of abstraction which enables this framework to become extremely adaptable to various environments and use cases. The three main components that constitute the framework are the following:

1. Back-end Application, in the exposed format of an API;
2. User Management Interface (UMI);
3. Data Storage System.

3.2.1 Back-end Application Layout

Starting on the most vital part of the whole platform, the back-end application, which contains most of the logic processes and flows of information in the framework. This application exposes all its functionalities through its API's endpoints.

3.2.1.1 Subscriptions

Internally, every entity which constitutes either a data consumer or producer is considered an *App* represented through the data model *Apps*. It is only once a user registers its *App* can he begin

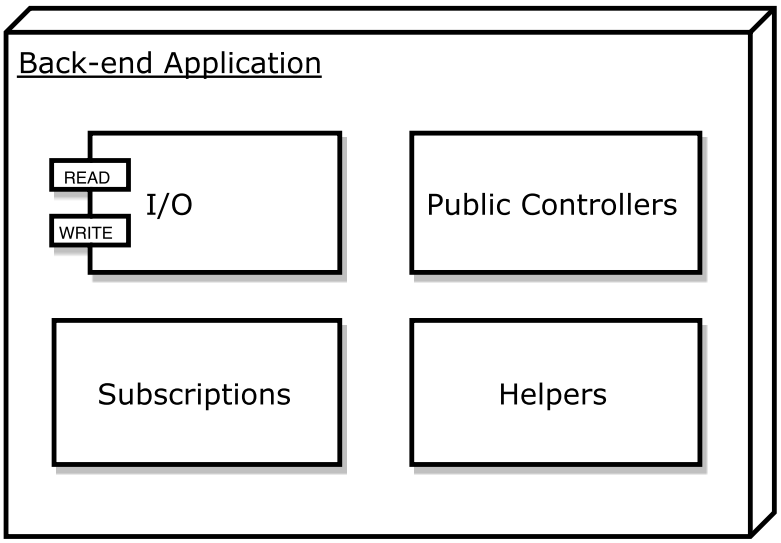

Figure 3.1: Back-end application layout including its internal modules. Comprises the I/O module, exposing read and write functionalities, public controlers, subscription modules and helpers.

to request any type of information exchange or consultation.

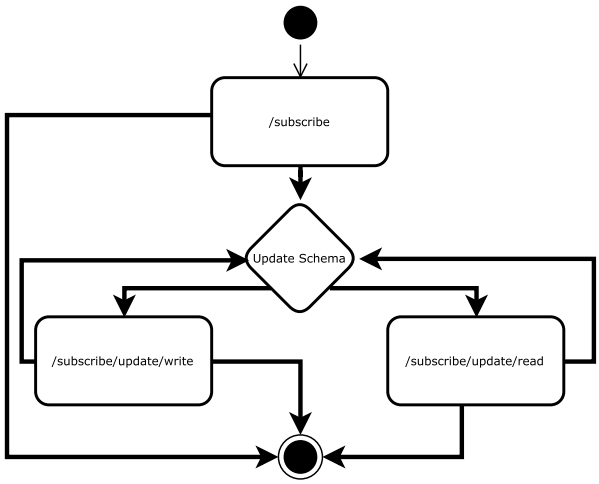

Figure 3.2: Flow diagram of the creation of an App, its subscription on the framework and following steps.

The Subscription module encapsulates the API methods that allow a user to:

1. Subscribe an Application as an App - `/api/subscribe`;
2. Define a read schema - `/api/subscribe/update/read`;
3. Define a write schema - `/api/subscribe/update/write`.

The process of subscribing an entity/application/system, see Figure 3.2 (p. 32), gives the user the information needed for him to makes other types of I/O requests, provided that he, at the time

of the request, sends a name for the entity he intends to register. It is this name that, combined with a randomly generated number, serves as the universal identifier for the entity.

For a user to either receive information or inject any, he must first inform the framework what that data is. This is done by the two methods (methods 2 and 3) on the API, each receives either a read or write schema which is a piece of information that clearly states/defines a model that can be used to be verified against another piece of information. That verification states true or false for whether the information given fits a certain model and its required attributes. This behaviour on both these two latter mentioned methods works slightly differently as is seen in the following Section 3.2.1.2 (p. 33).

3.2.1.2 Input/Output

The data processing methods are defined on the Input/Output modules since it is through them that the API exposes the *read* - `/read` - and *write* - `/write` - routes. Additionally, these are the only methods on this module.

In order for a user to be able to either *read* or *write* on to the framework, he must first have the appropriate permissions. This can be obtained through the module Subscriptions, as mentioned in the previous section, 3.2.1.1 (p. 31).

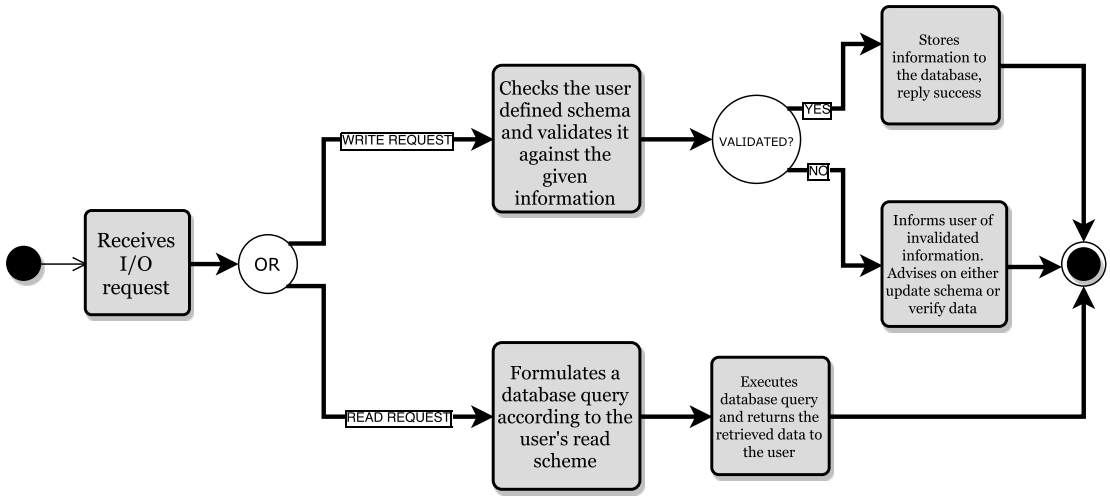

Figure 3.3: Typical Input/Output request from a user to the platform, referencing both cases of either *read* or *write* and their respective paths.

Having the necessary permissions a user can perform a *read/write* request by using their respective schemas. For a *write* request, having previously defined a write schema, the application validates the information about to be written against the defined model, if successfully validated, then that data proceeds to follow the rest of the process, if not, it is discarded and the user is informed of the invalidation of the information provided. On the other hand, for a *read* request, the read schema defines what the user wants to retrieve from the platform. After the request has been made, the application looks at the previously defined read schema and builds a database query that fulfils the models' requirements. In this case, the request made, outside of other unknown or

internal issues to the platform will never fail as the application always tries to give the user the information he himself defined through the read schema, visual representation in Figure 3.3 (p. 33).

3.2.1.3 Helpers

As the name states, the functionalities provided by this module are mostly helper methods, which are methods that perform small tasks and don't belong to any particular module. However, they are given some relevance because they offer the schema validation methods mentioned in the two previous sections. From validating data payloads from user's requests, loading schema files and managing the state of the kept information. This last functionality updates the information that a user checks when choosing which data attributes he wants to read through a read request. Every time data in injected to the platform, its payload is run through the helper's Operations methods, which extracts the metadata fields that define the attributes. This information is stored in the database but is also later on used in the user management interface, as it is adapted to a more user-friendly format to be displayed.

3.2.1.4 Public Controllers

Although most endpoints provided by the API are restricted and must require a key for access, the Public Controller module exposes all endpoints which do not necessitate said key. This module contains only one method and is often overlooked in favour of other functionalities, during implementation. Currently, it has an endpoint that replies with the attributes available and from which App they belong, also including all available metadata on those same attributes.

3.2.2 User Management Interface

The framework's functionalities are all encapsulated in a RESTful API, by default there is no visual interface on which to interact with and manage. Additionally, the back-end application does not support any kind user data model nor does it make any kind of associations between Apps and other entities. The reason for this is discussed later in Section 3.3.1 (p. 38). Thus came the necessity of a User Management Interface (UMI), accompanied by a User data model, associating Apps with users. It allows users to manage their Apps.

The way that this is interface is built, serves a purpose. It is proposed that this interface only be used an example, although its a fully functioning one. As this platform, as a whole can be used in various situations and environments, it is only fair to make it possible for whoever uses it to be able to build their own interface adapted to their specific needs. With the use of the Public Controller module's exposed functionalities and the ease of integration of the Apps with any type of user data model, the assembly of domain-specific UMIs should come as a relatively easy task for any developer.

Figure 3.4 (p. 35) displays the initial page of the interface, listing the current users available and the Apps that each has associated with them. There are then two additional options for creating a new User and the possibility of choosing to go to a User's personal Dashboard.

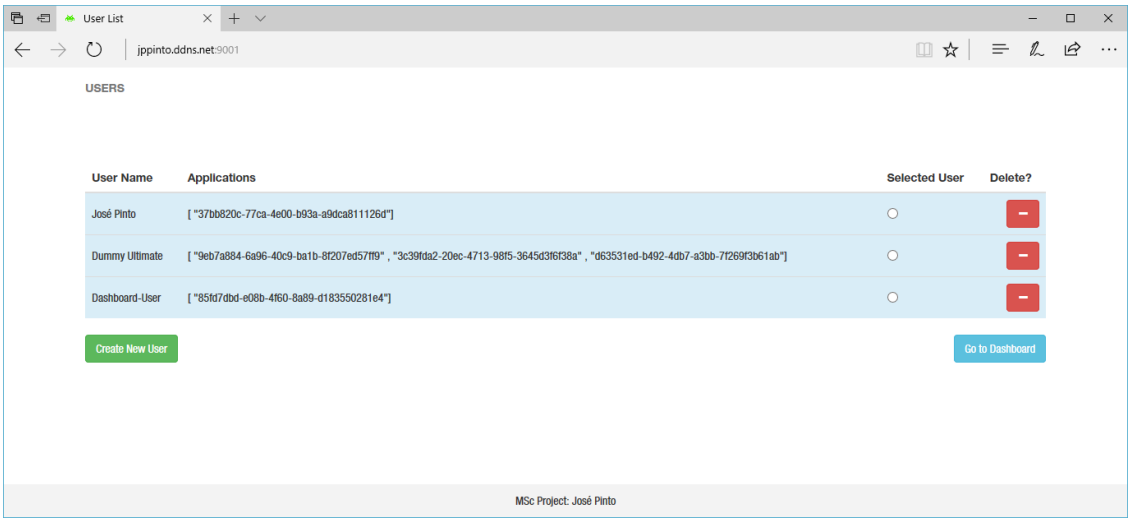

Figure 3.4: Web page that displays the User list and the Apps associated with each User.

Now, the reason why the front page of the interface shows all the available Users and their Apps, possibly enabling any person to access another User's information, is due to the scope of this dissertation. Given that the focus of this framework was not security, User's authentication and permission management for the UMI was not implemented, also it increased development time.

The rest of the UMI contains an interface for the User (User Dashboard), individual Apps (App Dashboard) and their respective creation and deletion functionalities.

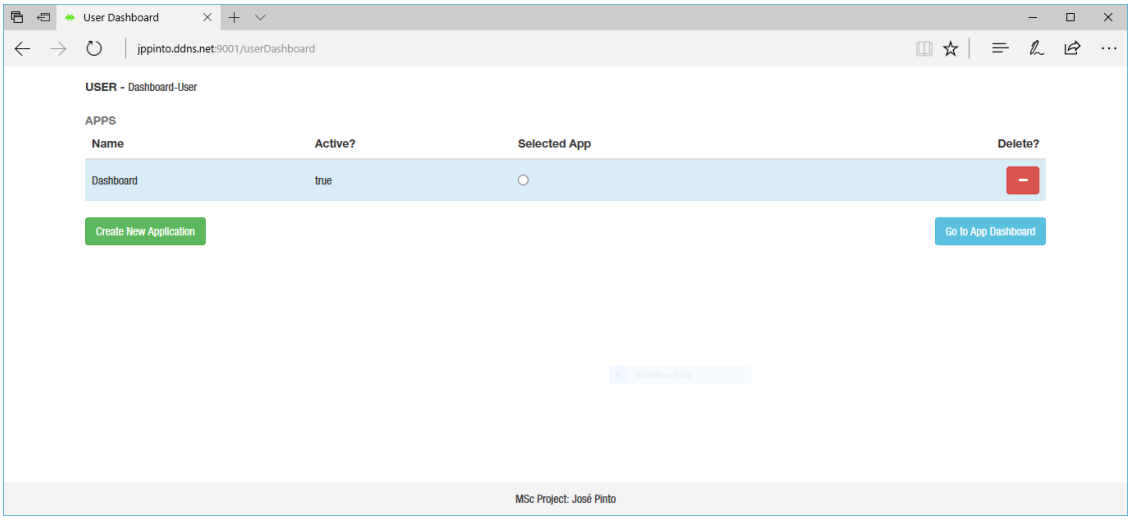

Figure 3.5: Web page that displays the User Dashboard and its associated Apps.

The User Dashboard depicted in Figure 3.5 (p. 35) follows a similar layout as the initial page, except instead of managing Users, it manages the Apps belonging to the selected User from the previous page. It contains an App creation button, which when successfully done, creates an App that becomes associated with the current User and a deletion button that deletes the selected App.

Contrary to the delete button on the Users List interface, which purges the User's data from the database, this one, and since the App's information belongs to the framework and not to the UMI, only disassociates the App from the current User, keeping in storage all the data injected by that App, its details and the read/write schema configurations. Selecting one of the Apps sends the user to the selected App's Dashboard.

The two following interface, in Figures 3.6 (p. 36) and 3.7 (p. 37), display the two functionalities available for Apps management. These are the fields for configuration of both the read/write schemas.

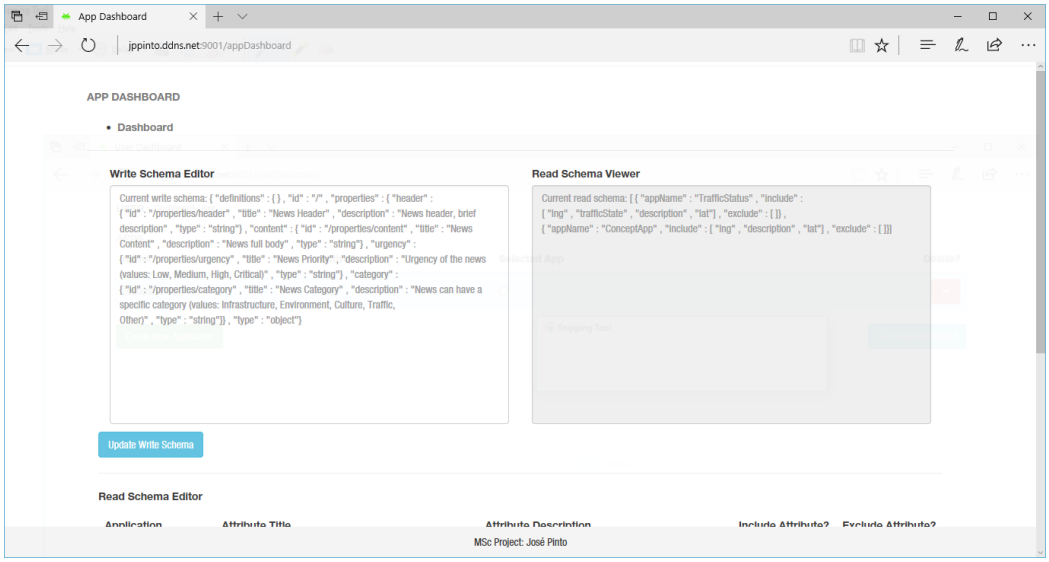

Figure 3.6: Web page that displays the Dashboard of a selected App from a User Dashboard, focusing the write schema edition panel.

On the write schema edition field, it is displayed as a placeholder for the field, the current write schema. The User simply inputs the desired write schema and clicks "Update Write Schema". For the read schema, the process is slightly different.

It is given a "Read Schema Viewer", which is a not editable field, and only displays the current read schema. Displayed in Figure 3.7 (p. 37), the User here can see all the available attributes and to which Apps they are associated, allowing him to pick which attributes he wants to include and exclude for this read requests. Not selecting any attribute on a particular App, excludes that App's data from the read schema completely. Meanwhile, if an attribute is selected, on either include or exclude, the other none selected attributes are considered unimportant, which means that their presence in the query results is not considered at all, signalling that the User does not care whether that attribute's information is present or not in the query's result dataset. After this selecting and clicking "Update Read Schema" the UMI will then generate a read schema and use it to update the one on the current App.

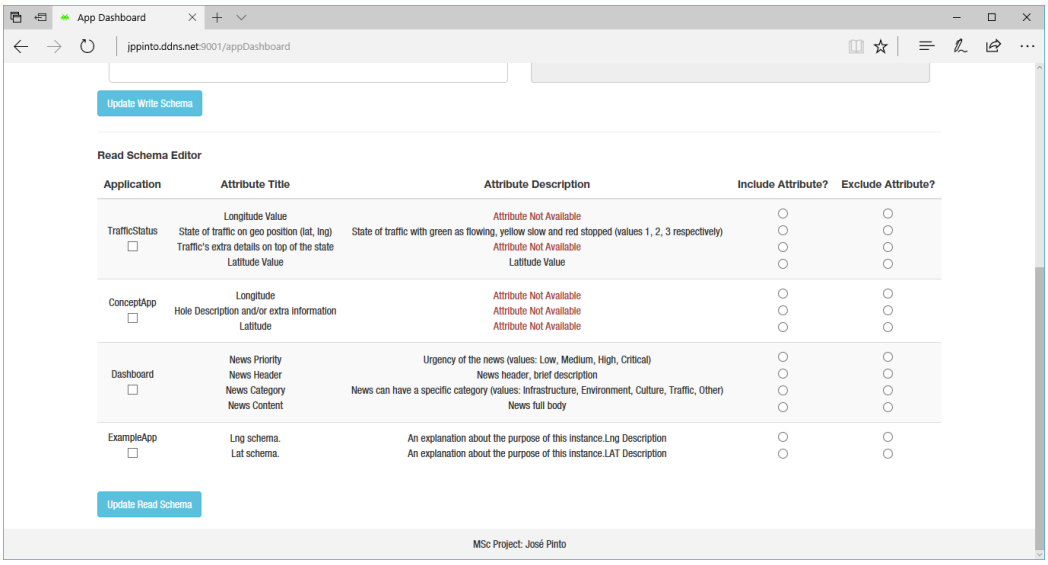

Figure 3.7: Web page that displays the Dashboard of a selected App from a User Dashboard, focusing the write schema edition panel.

In an attempt to better clarify what the typical UMI user might be and given that this web application is rather small in scale, Figure 3.8 (p. 38) is used to illustrate the UMI site road map. This section describes the functionalities implemented on this UMI in particular, made for the purpose of this dissertation. In future instances and implementations of the whole platform, it is encouraged that the developers either use this example, extending its capabilities or, given the tools provided by the platform, design their own domain-specific UMI.

3.2.3 Data Storage System

Storage in the project was approached having mind the decentralisation concept, which comes with several advantages in terms of scalability and data redundancy for fault recovery mechanisms. This also enables scenarios which have several instances of the framework running with different storage systems [MSTR12].

When having an App write information to the platform, it gets placed in a database table with a unique identifier, allowing that table to be recognised as the only instance allowed in its current working domain. This means, that an App's data can only be written to that, and only that table.

Looking at the other two components, the Back-end Application and the UMI, it is clear that both systems require storage. One, for storing information given by the Apps registered, the other to keep information on its registered users. There is, one particular thing about this separation, which is the independence of each other, because aside from the communication between the Application and the UMI, their storage is completely independent, since the elements stored on the side of the UMI, contain the necessary information to be identified within the Application's environment. This independence of data is an advantage when considering large-scale cases as the framework itself can serve as a data support in simulation of traffis in GIS [APR10].

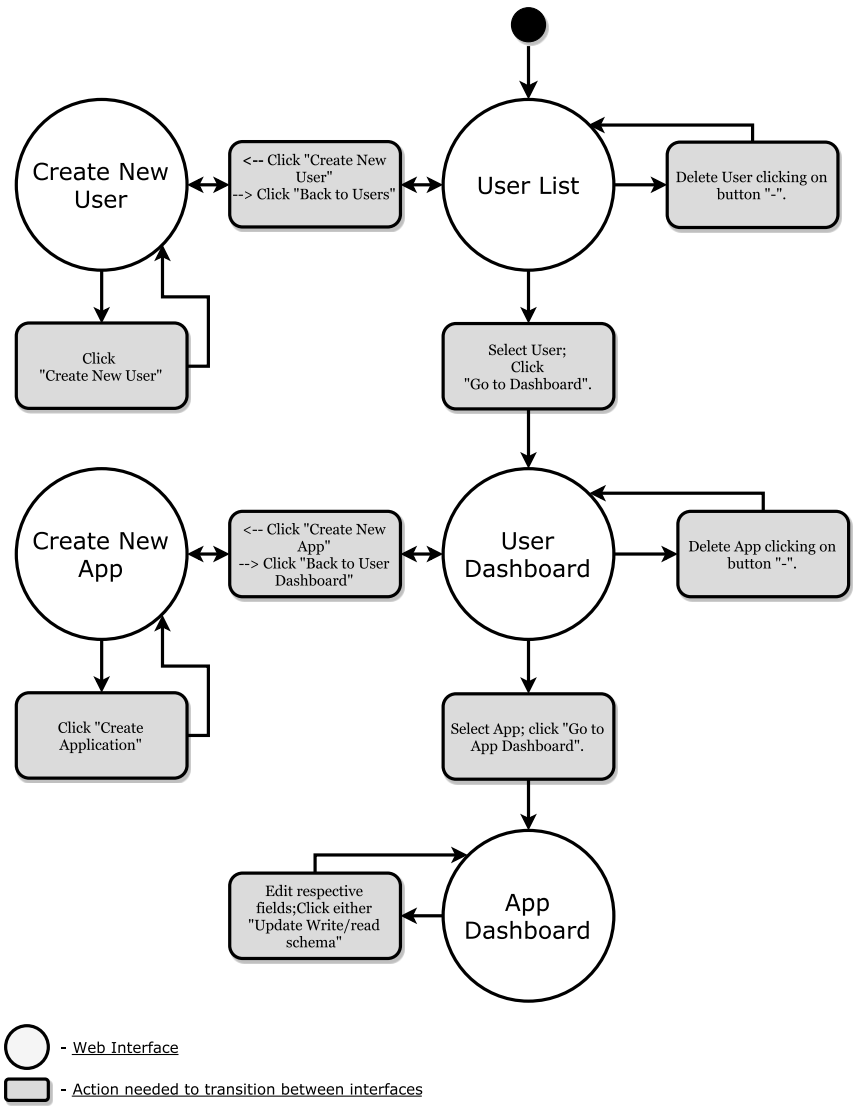

Figure 3.8: Portraits every possible user navigation through the different pages of the UMI web application, specifying the actions needed to do so.

3.3 Abstraction Level

One of the goals of this projects was extensibility. The whole process of building the platform always had in mind the fact that this could easily be adapted to a variety of situations. In response to that notion, the framework was built abstracting all the components and the logic processes on it.

3.3.1 Component Modularity

As mentioned when defining the purpose of the UMI, it was made out of necessity in order to manage the Application through a visual interface, but still, serves to prove the fact that it was

built using only the Public Controller module's functionalities not touching the other encapsulated modules.

As for the Application, down to the core, it was built abstracting methods and definitions. The only processes an App can trigger are the "read" and "write" ones, data is encapsulated and sent through a mediator which abstracts it and adds the necessary metadata for it to be placed inside the data storage system. Also, payload[1] validation is done through the use of schemas, defined by the same owner of the App requesting the information, allowing this validation to be as flexible as the users want them to be.

3.4 Summary

The conceptualisation of the project was given great importance due to the amount of implementation that was done to be able to build the whole framework from scratch with aiding interfaces. With a better understanding of the overall methods and processes of the major components of the platform, the comprehension of the architecture and some of the decisions made with the implementation is certain to be well received.

[1] Designation for a chuck of information stripped from metadata sent to the platform

Chapter 4

Architecture & Implementation Details

This chapter covers the technical gap of the previous, going into detail on the main processes and interactions between users, applications, and framework components. Covering the technological overhead by defining the technologies used and their justified choice, here is also provided with a descriptive exposure of the API's endpoints and a look it to some concepts which are taken into consideration, having such a meaningful impact that it reflects on the architecture of the platform.

4.1 Technological Environment

For this platform, all technologies used were Open Source software projects. Both the Backend Application and the UMI are built as Maven[1] (v3.3.9) projects for dependency management, using the Spring Framework[2] (v4.3.6). This is an application framework intended to provide the necessary tools onto which to build applications upon, meant to run anywhere, since they can be executed in any environment running a Java Virtual Machine (JVM) while being REST compliant. Every storage system uses a database of NoSQL type, namely MongoDB[3] (v3.4.1), as its flexibility justifies its use on this projects as almost every block of stored information is dynamic due to the platform allowing the user to define what he wants to store and the ability to change so, as he needs.

[1] Apache Maven's Official Web Page, https://maven.apache.org/
[2] Spring Framework Wikipedia Page, https://en.wikipedia.org/wiki/Spring_Framework
[3] MongoDB's Documentation Page, https://docs.mongodb.com/

The Spring ecosystem also has some open source projects that proven useful. Some utilities such as the Spring Data, that offers a Spring-based programming model for data access were used, specially, the Spring Data for MongoDB, simplifying the integration with the project and enabling functionalities for dynamic query derivation, custom object-mapping abstractions, extensively used when receiving information payloads by reflecting the data onto an object. This combined with the Spring Data Repositories, allowed for zero integration effort for the later persistence operations to the database. Another utility used that had a significant impact on the development time and complexity was the Spring Boot, as it automatically manages embedded services as the Apache Tomcat Servlet, used as a server platform, integration with Maven projects and numerous configurations while effortlessly providing endpoints for either metrics, health checks and log configurations.

The UMI, besides the web technologies used, HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and Javascript, the UMI takes advantage of Spring MVC, for the generation of web pages connected with the data models, controllers and the library Thymeleaf[4] (v3.0.3), as web page template engine, which was picked due to its practicality of method integration, available documentation and compatibility with the Spring Framework.

Since its publication in the RFC 4627 [Cro06] document, the Javascript Object Notation[5] (JSON) gained preference over the conventional XML language for its traditional use in web services as a data format for transmitting data back and forth. As a lightweight format with enormous support and adoption, in the technological community, JSON is used in this project as the format for all data interchanging processes. Also taking a major role, JSON, is used a the specification language for data validation and description, with the notation of JSON Schema [Wri16, WLA17].

Lastly, the framework's exposure is done through an API and to properly document and test it, the library Swagger[6] (v2.4.0) was used, automatically generating documentation for all API's endpoints and offering a web interface for prototype testing making development an easier process.

4.2 Data Interoperability

The most important aspect of this platform is the interoperability it offers between data exchanges. This is possible due to how data is processed once it reaches the platform, added, the interaction the data owners, the users, have with the framework by defining read/write schemas, make it so that it is possible for different systems to communicate.

4.2.1 Schema Definition and Validation

The use of data schemas are the basis of the data exchanges as they define what is written and what is read. These schemas are essentially JSON Schema[7], which is a JSON-like object

[4]Thymeleaf's Official Web Page, http://www.thymeleaf.org/
[5]JSON Notation Offical Web Page, http://www.json.org/
[6]Swagger IO Offical Web Page, http://swagger.io/
[7]JSON Specification media done by Google, http://json-schema.org

that describes other pieces of data, in the same format. They can be used to assert: (1) ways of interaction with information; (2) how to extract information from it; (3) to define how the data must look like, by describing it in a human and machine readable way; (4) validate client-submitted data, which is the functionality most used in the framework; (5) perform automated testing. The document being validated against the schema is usually named an `instance` [Wri16].

A blank JSON object is the most basic form of a schema which does not describe anything nor does it constrain anything either, allowing every JSON-like data to the successfully asserted against it:

```
1 { }
```

Listing 4.1: The most basic form of JSON Schema which is empty brackets.

Exemplified in Figure 4.2 (p. 43), the schema example allows a better understanding of how the JSON Schema works by first analysing its constraints and description and then running it against two other instances. It should also be highlighted that this next example serves as a demonstration of how a write request data is validated but only in terms of the payload validation.

The example of JSON Schema describes an occurrence of fire in an area, specifying: (1) that the instances validated against it must be an `object`; (2) can contain five distinct properties a *lat*, *lng*, *forrestFire*, *surroundingConditions* and *dangerScale*, which, if present on an instance, must be, respectively, of type *number*, *number*, *boolean*, *string* and *integer* (or *number*); (3) must contain the properties *lat*, *lng* and *dangerScale*; (4) the *dangerScale* property has its value constrained to a minimum of 0 and a maximum of 10.

For the purpose of the project, here it is only taken upon analysis the validation capabilities of the JSON Schema.

```
1  {
2    "type": "object",
3    "properties": {
4      "lat": {
5        "type": "number"
6      },
7      "lng": {
8        "type": "number"
9      },
10     "forrestFire":{
11       "type": "boolean"
12     },
13     "surroundingConditions" : {
14       "type": "string"
15     },
16     "dangerScale": {
17       "type": "integer",
18       "maximum": 10,
19       "minimum": 0
20     }
```

```
  },
  "required": ["lat", "lng", "dangerScale"]
}
```

Listing 4.2: JSON Schema example for a write schema

This schema, when ran against the following two instances, which describe the same fire, assert true for the first one and false for the second.

```
{
  "lat":8.1354,
  "lng":-42.30255,
  "surroundingConditions" : "Very wind and likely chance of spreading to other
      buildings",
  "dangerScale": 9,
  "fireHeight": 15
}
```

Listing 4.3: JSON data instance that is validated against the JSON Schema 4.2 is successful on being asserted

```
{
  "lat":8.1354,
  "lng":-42.30255,
  "surroundingConditions" : "Very wind and likely chance of spreading to other
      buildings",
  "forrestFire" : false
}
```

Listing 4.4: JSON data instance that is validated against the JSON Schema 4.2 failing on being asserted

On the first instance, it asserts as true, against the schema 4.2 (p. 43) by being in compliance with every constraint and because the schema does not take into consideration the non-existence of the *forrestFire* property nor does it account for the *fireHeight* property. On the second instance, it asserts as false due to the missing property *dangerScale* that, alongside the *lat* and *lng* properties, is an obligatory property, hence the instance is invalidated.

Note that this exemplification does not cover every possible validation setting and the proper documents should be check in order to know all the possible applications of this tool [WLA17, Wri16]

These schemas by themselves serve as an exceptional tool for validation and other specification tasks but so far, with this last example, there is still a gap in content, namely, in context. The schema, as it was given, did not provide any context on what its content means, how it is applied or specification on measurement types for numerical values. By using metadata keywords, that are part of the JSON Schema specification, the information given becomes context-aware. These keywords are **title** and **description** and they must be a string value when used. They can be placed on the root level of the schema and at the property level, providing context for the schema's

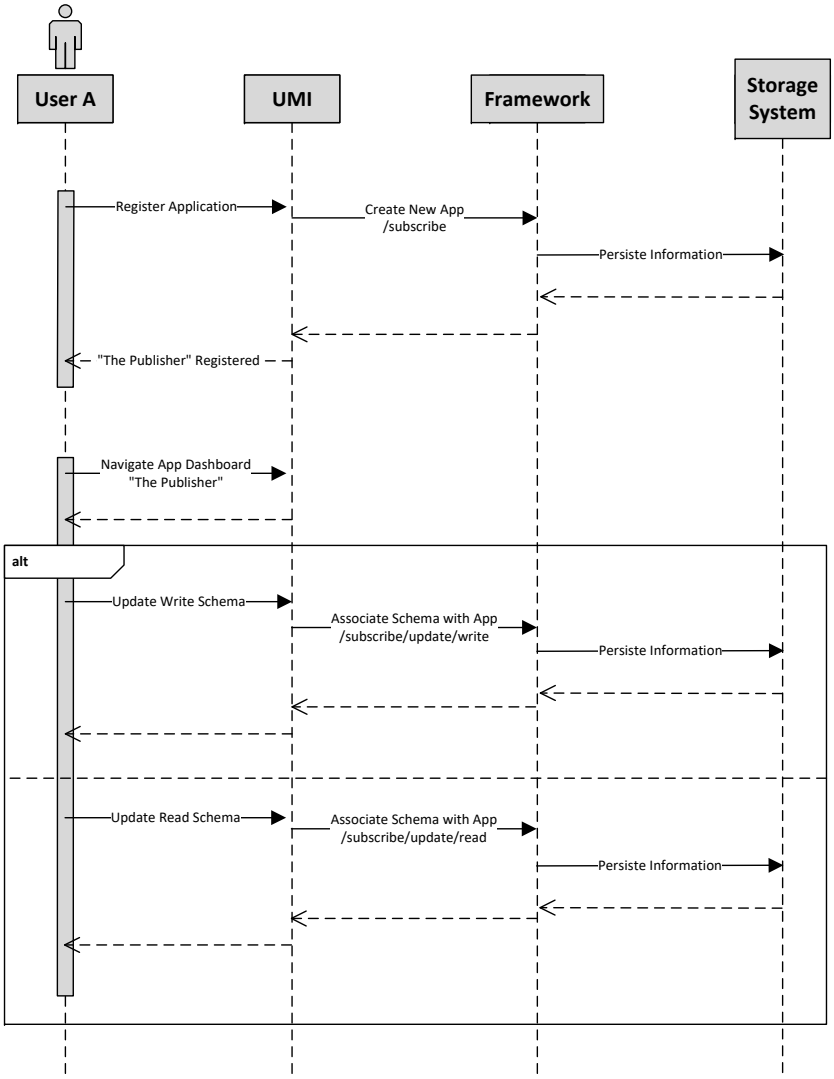

Figure 4.1: Sequence diagram for User A's schema configuration actions with the components of the platform.

purpose and its individual properties' meaning and usage, respectively. Both serve as decorators where the title is a short piece of information where the description is a more extended one.

```
{
  "title" : "Sighting of a fire",
  "description" : "This schema serve as a descriptor for data on sightings of fires
      , focusing its location and danger level.",
  "type": "object",
  "properties": {
    "lat": {
      "title" : "Latitude Value",
      "description" : "Latitude value of the exact coordinates on the sighted fire.
          ",
      "type": "number"
```

```
10      },
11      "lng": {
12        "title" : "Longitude Value",
13        "description" : "Longitude value of the exact coordinates on the sighted fire
            .",
14        "type": "number"
15      },
16      "forrestFire":{
17        "title" : "Forrest Fire Indicator",
18        "description" : "Indicates whether this is fire is located at a forrest site
            or not.",
19        "type": "boolean"
20      },
21      "surroundingConditions" : {
22        "title" : "Surrounding Conditions",
23        "description" : "Description of the fire's surrounding environment, useful to
             indicate roads or hazardous elements on the field.",
24        "type": "string"
25      },
26      "dangerScale": {
27        "title" : "Danger Scale",
28        "description" : "Indicator for the level of urgency (from 0 to 10) on which
            to attend to the fire. 0 being extinguished not needing assistance and 10
             being to be attended at the earliest time.",
29        "type": "integer",
30        "maximum": 10,
31        "minimum": 0
32      }
33    },
34    "required": ["lat", "lng", "dangerScale"]
35  }
```

Listing 4.5: Previously shown JSON Schema example now carrying metadata keywords providing context of itself and its properties

Applying the metadata keywords on all the schema's allowed properties offers a much more complete and informed solution for the identification and validation of JSON data.

4.2.2 Read/Write Requests

The read and write methods are the most important endpoints of Input/Output for the platform but, in terms of implementation, they are quite different. In the interest of better explaining how these requests function and as an illustrative scenario of these interactions is it used a fictitious and typical user, named User A, who has a journalism mobile application, named The Publisher, that, amongst other things, is able to publish news article to his personal web page.

Write Request

As illustrated in Figure 3.3 (p. 33), before a user executes a write request, it is assumed that he has already defined a write scheme, otherwise, a blank JSON Schema (`{}`) is not capable of either describing any data, resulting in an invalid write request.

User A has his App registered in our framework and assuming he has submitted the following schema 4.6 (p. 47), he now is able to send a write request with a payload carrying a news article containing a *header*, *content*, *urgency* of the news piece and its *category*.

```
{
  "type" : "object",
  "properties" : {
    "header" : {
      "title" : "News Header",
      "description" : "News header, brief description",
      "type" : "string"
    },
    "content" : {
      "title" : "News Content",
      "description" : "News full body",
      "type" : "string"
    },
    "urgency" : {
      "title" : "News Priority",
      "description" : "Urgency of the news (values: Low, Medium, High, Critical)",
      "type" : "string"
    },
    "category" : {
      "title" : "News Category",
      "description" : "News can have a specific category (values: Infrastructure,
          Environment, Culture, Traffic, Other)",
      "type" : "string"
    }
  }
}
```

Listing 4.6: User A's write schema allowing to publish news articles

After linking his application with the framework in order to feed the news article to both his web page and the framework, the body content of the HTTP request to the platform should look similar to the following:

```
{
    "uuid" : "********-446c-49b8-a422-a759a8828d34",
    "apiKey" : "********-ca28-47e7-b836-d4a98c7309b8",
    "payload": {
      "header" : "Grand Opening the Porto's Latest Art Galery",
      "content" : "Yesterday evening was the opening of the most recently built art
          gallery. Part of the city's cultural program, having a successful
          opening, the gallery's pieces promises to keep its guests entertained.",
      "urgency" : "Low",
```

```
8        "category" : "Culture"
9    }
10 }
```

Listing 4.7: User A's HTTP request's body content containing a payload to be written to the framework

Using the properties *Universal Unique Identifier (UUID)*, to identify the App, and *apiKey* allowing the user to perform the request, the framework validates the object present on the property *payload* against the aforementioned schema, successfully, proceeding to store the *payload*'s information. Along with other attributes such as a timestamp in milliseconds and a hash, generated from the timestamp with User A's IP address for identification, the writing process is completed.

Read Request

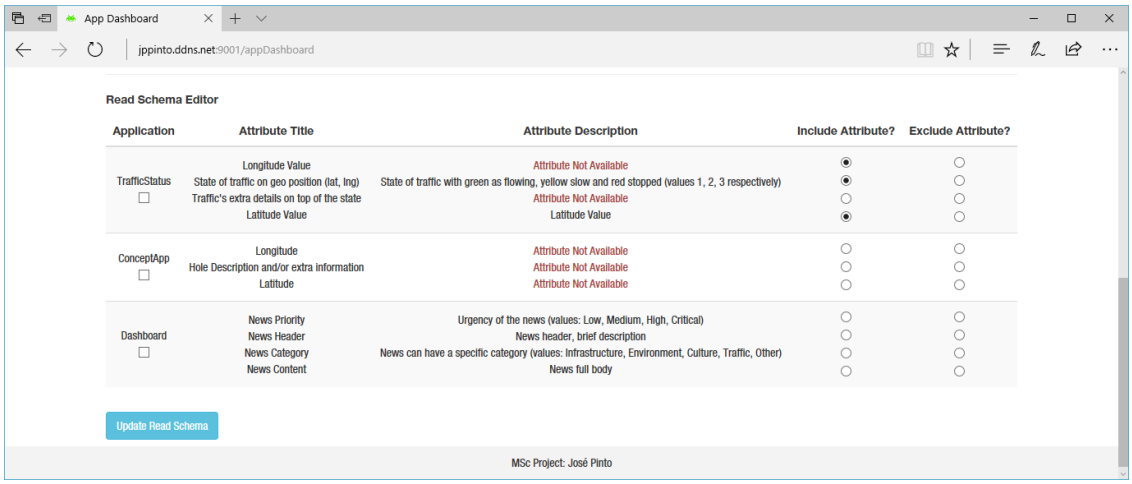

Figure 4.2: User A's information choices on available information, selected attributes can be seen on the radio buttons.

Still, on User A, he now wants to take advantage of some of the information allocated on the platform. As explained in section 3.2.2 (p. 34), in order to do so, User A uses the UMI to access his dashboard to check what information is available on the platform. He then chooses the scenario depicted on Figure 4.2 (p. 48).

```
1 {
2   "appName": "TrafficStatus",
3   "include": [ "lng", "trafficState", "lat" ],
4   "exclude": []
5 }
```

Listing 4.8: User A's write schema that allows him to publish news article

As he sees that there is traffic related information on the App *TrafficStatus*, which he can cross-check the geo-locations with some articles he has written, he then chooses the location and

the state of the traffic (attributes *lat*, *lng* and *state*), discarding any other details available. In this described case, the UMI generates the schema 4.8 (p. 48) and associates it to User A's App.

```
{
  "TrafficStatus": [
    {
      "lat": -8.656956666666668,
      "lng": 41.162926666666664,
      "description": "Traffic is kind of stuck today D: #slowAsHell #trafficStatus
          #redTraffic",
      "trafficState": 3,
      "timestamp": 1493392539396
    },
      (...)
    {
      "lat": -8.656956666666668,
      "lng": 41.162926666666664,
      "description": "Almost got to work #hopeNeverEnds #trafficStatus #
          yellowTraffic",
      "trafficState": 2,
      "timestamp": 1493392561911
    }
  ],
  "timestamp": 1498131419822,
  "code": 200
}
```

Listing 4.9: User A's results from *TrafficStatus* dataset given his read schema generated on the UMI

With this completed process, User A's read requests will then return all entries on the App `TrafficStatus` which must have the attributes on the array `included` present of the read schema. The results produced by the query are shown in Listings 4.9 (p. 49).

Summing up this section, the sequence diagram 4.3 (p. 50) illustrates how the interactions between User A and the platform occur. User A only interacts with the UMI, which then communicates directly with the framework. Meanwhile, after registered, the User A's App, interacts only with the framework and never with UMI. Here we see the use of the endpoints `/subscribe`, `/read` and `/write` assuming that the schema configuration has already been made before any I/O requests, as seen on the sequence diagram 4.1 (p. 45).

4.2.2.1 Query Formation

For every read request and before querying the storage system, the framework first check the read schema and generates the query it later executes. In terms of the query formation its generation follows a simple process.

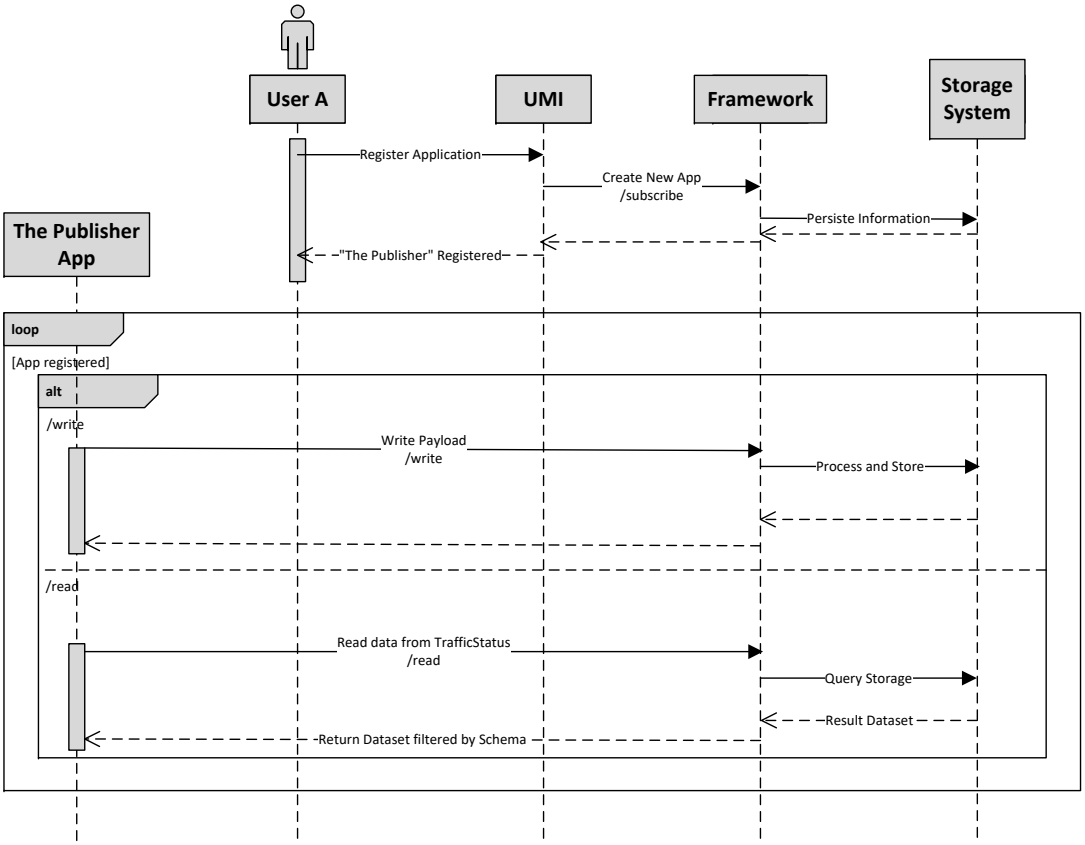

Figure 4.3: Sequence diagram for User A's interactions with the components of the platform while subscribing his application and following interactions of the App itself with the framework.

```
db.getCollection('TrafficStatus-********-b492-4db7-a3bb-7f269f3b61ab')
  .find(<read schema conditions here>)
```

Listing 4.10: Use of the *find* function of MongoDB on the collection *TrafficStatus* indicating the placeholder for the conditions that filter the returned dataset to match the read schema

Using the *find* function of MongoDB, present in every collection, it is the placed on this function, 4.10 (p. 50), the necessary conditions for the query results to become in compliance with the read schema. These conditions are generated by adding a separate condition for each attribute described on the schema, using User A's read schema, 4.8 (p. 48), as an example.

```
{
  "payload.lat" : {$exists : 1},
  "payload.lng" : {$exists : 1},
  "payload.trafficState" : {$exists : 1}
}
```

Listing 4.11: Conditions generated from analysis of User A's read schema

Besides the information stored by an App, that is contained within an object named payload in each block of sent information, there are also other attributes but, it is only intended that the read scheme describe the payload object. Bearing this in mind, the three attributes in the `include`

array are part of the payload object and by making use of the search keyword `$exists`, which forces the query to only return results where the specified attribute is present, it generates the conditions, 4.11 (p. 50). Every condition generated targets an attribute on the payload object, if that attribute is on the `included` array than the keyword `$exists` has the value **1**, otherwise if that attribute is on the `excluded` array than the keyword `$exists` has the value **0**.

4.3 API Endpoints

The Spring Framework by itself provides a series of diverse endpoints which complete the environment with essential information such as health checks, logs level configurations, service monitors, environment status and various hardware and software metrics, complemented with the addition of other dependencies such as the actuator.

Table 4.1: Description of the developed endpoints for the framework

Endpoint URI	Description	Parameters	Return Content
/read	Accesses user's read schema and executes generated query on the storage system.	**apiKey**: hash for the permission to all user endpoints; **UUID**: unique identifier that universally matches a user's App.	Resulting dataset from the generated query execution.
/write	Enables a user to send a payload of information associated with an App.	**apiKey**: hash for the permission to all user endpoints; **UUID**: unique identifier that universally matches a user's App; **payload**: JSON-like object previously validated against the user's write schema.	Whether or not the writing action was successful.
/apps	Provides the functionality of viewing the Apps of the specified users.	**apiKey**: hash for the permission for the public endpoints; **payload**: specification of the users from which to return their Apps.	The list of Apps and associated user.
/apps/available	View data available on the framework.	**apiKey**: hash for the permission for the public endpoints;	Complete set of available attributes and their respective metadata describing components, associated with each available App.
/apps/delete	Delete an App by providing its UUID.	**apiKey**: hash for the permission for the public endpoints; **payload**: specification of the UUID of the App to be deleted.	Disables an App which is the equivalent of deleting it. This maintains the App's data, details and configuration schemas.
/subscribe	Register an App to the framework.	**name**: string identifier for the to be created App	The associated apiKey and UUID of the App.
/subscribe/update/read	Associate a read schema to an App.	**apiKey**: hash for the permission to all user endpoints; **UUID**: unique identifier that universally matches a user's App; **schema**: JSON-like object representing the read schema.	Whether or not the process of association was successful.
/subscribe/update/write	Associate a write schema to an App.	**apiKey**: hash for the permission to all user endpoints; **UUID**: unique identifier that universally matches a user's App; **schema**: JSON-like object representing the write schema.	Whether or not the process of association was successful.

Table 4.1 (p. 52), focuses on the developed endpoints that ultimately defined the framework. For more information on all the available endpoints, consult annexe in A.2 (p. 74).

4.4 Storage Sharing

Even though this framework portraits a centralised model by gathering all the logic processes in a single module of components, when it comes to storage, the issue is not the same. Aside from a collection to store the details of Apps, named *apps*, including their configurations and schemas, there are the collections that store each of the Apps data blocks, injected by their users. These have a particular aspect, which is their name, by identifying each collection with a unique identifier, in the format of *<appName>-<UUID>*. This UUID identifies a collection, a single and only collection to which that App, whose UUID is the same as the one on the collection name, can write information to. That App, this way, is able to have its information not only sharded through several instances of the framework, it is also able to write to any of them. This same detail provides data redundancy, fault tolerance and allows the systems' storage to scale horizontally.

4.5 Module Interoperability

In the beginning of this project and platform specification, there was a group of ideas that were intended to be implemented. These increased the development time and complexity by a good margin, and would most likely never be finished in time.

The focus of the project became the construction and development of good foundations for an Open Data platform and enabling an environment that could surpass the issues of interoperability.

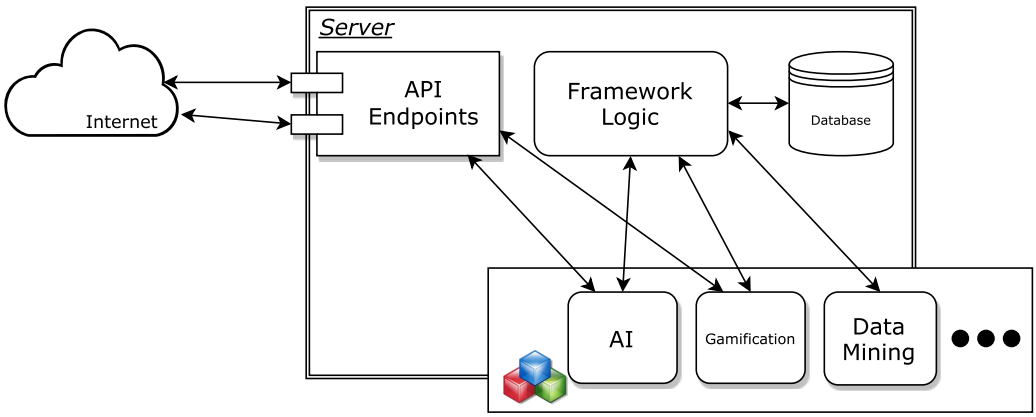

Figure 4.4: Server with deployed instance of the framework, portraying component modularity and interactions.

With that in mind, the platform was built not only encapsulated but with all its components highly modular. Additionally, the framework has available the structure to easily integrate plug-and-play components, 4.4 (p. 53), that could contribute quite substantially to its value as a tool. The extensibility option for this platform is quite big, having the possibility of integrating modules

of machine learning and data mining services through either endpoints of real-time access to data or just the normal archived data access. The field of artificial intelligence is easily explored given the amount of geo-located information that the platform is likely to store, given its context it is inserted in, namely the Smart City environments [USRS16]. As a last note, as mentioned in chapter 1 (p. 1), the most viable source of information about a Smart City's citizens is through mobile applications, which open a panoply of options that might combine the data provided by these application and the possibilities of gamification between them and the framework [KMR+14, RAKG13, RLCB02].

4.6 Summary

This chapter described the technological environment justifying those choices, mainly, by their flexibility and the open source aspect. The technologies played a very important role in terms of implementation, since the framework directly relies on them, especially on the data interoperability part, to create distinct value and to allow for a modular and yet abstract environment that has the goal of being easily extensible.

Chapter 5

Case Studies

As a means of properly testing the performance of the platform in a properly built setup scenario, the following test cases are presented. This chapter is dedicated to bringing forward the technological environments onto which the framework is tested in regard to its performance, communications and consistency.

For each of the following case studies, in the framework's perspective, all the interacting components act as Apps. These can read and write, subscribe to framework data and share information. The case studies indicate scenarios on which there is the development of some sort of tools, services or architectures that resort to the framework as its information hub.

5.1 City Dashboard

Our first case study reflects the intention of building a dashboard that would harness the information of several mobile applications within the context of a Smart City.

This dashboard is placed onto a small-scale technological environment able to execute all the developed functionalities. All the details on the hardware specification of the components onto which the platform is deployed are available in Section 5.2 (p. 58).

The major actors in this scenario are, the Dashboard itself and two mobile applications 5.1 (p. 56). These two applications are, respectively, a real-time traffic state reporter, *TrafficStatus*, which its interface can be seen in Figures 5.2 and 5.3 (p. 57), for users to tweet out their current traffic situation while expressing their feelings towards it. The other is an application that allows its users to report information on holes and abnormalities in streets or city transit infrastructures, named *Pavements*, which its interface can be seen on 5.4, 5.5 (p. 57).

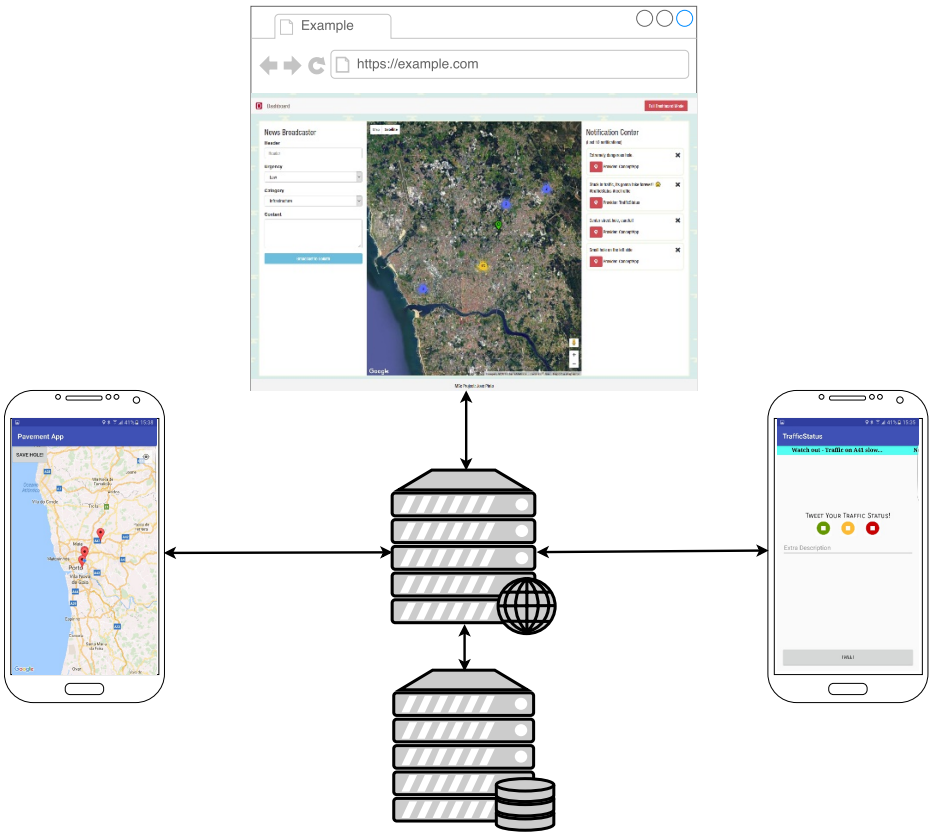

Figure 5.1: City Dashboard case study components' diagram, illustrating interactions and communication with the framework.

Both the applications have their main purpose but they are presented with the chance of taking advantage of the platform, which they do by replicating their normally generated information on their users' activity to the framework.

On the other hand, the dashboard is built around the information provided by the applications on the framework, comprising a visualisation interface for the different types of data such as traffic information through the use of tweets [CSR10, KFC+15, RSR15]. Since in this scenario this dashboard is supposed to be run by a city hall, or some sort of government authority, it has the added functionality of a broadcaster, utilising the framework as a means of sharing relevant information to every mobile application or systems that want to access it. This can be quite relevant for disaster scenarios, major city events or simply general news broadcasting.

Figure 5.1 (p. 56) illustrates the four components integrating the case study with their connections towards the framework and the other applications involved in the same environment. The communication between any component is done through the framework.

The Dashboard has 3 major components, namely the News Broadcaster (on the left panel), the Google Map instance (on the centre panel) and the Notification Center (on the right panel). The panel on the centre and on the right are focused on the display of information read by the Dashboard application from the framework. The centre panel displays every type of data that, after retrieved is analysed and if containing geo-located attributes, is displayed in a distinct colour on

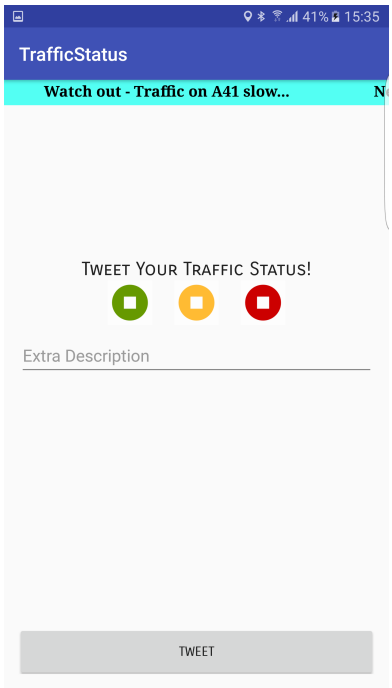

Figure 5.2: Screenshot of *TrafficStatus*, upon opening the application, also a blue news broadcasting bar can be seen on the top.

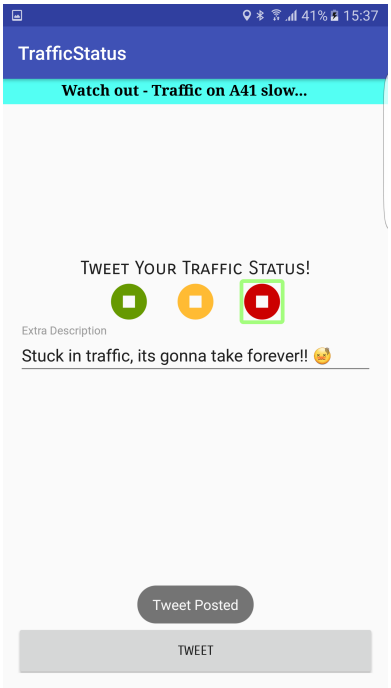

Figure 5.3: Screenshot *TrafficStatus*, after sending a Tweet, also a blue news broadcasting bar can be seen on the top.

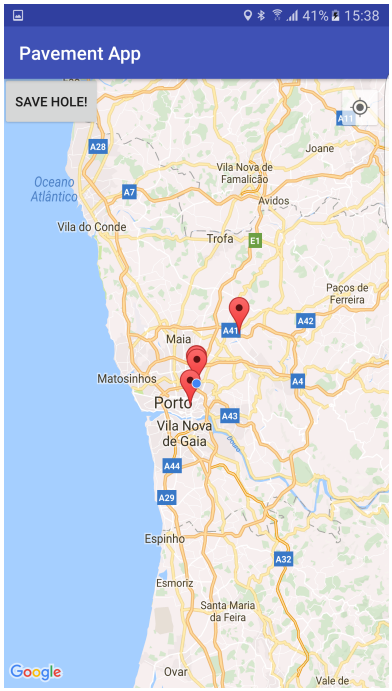

Figure 5.4: Screenshot of *Pavements* mobile application, upon opening the application.

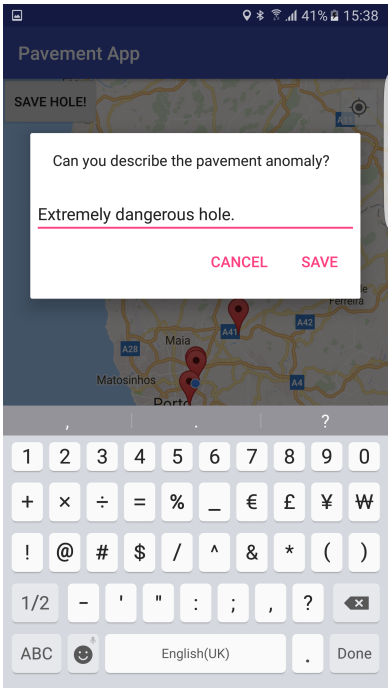

Figure 5.5: Screenshot of the application *Pavements* after saving an ground anomaly.

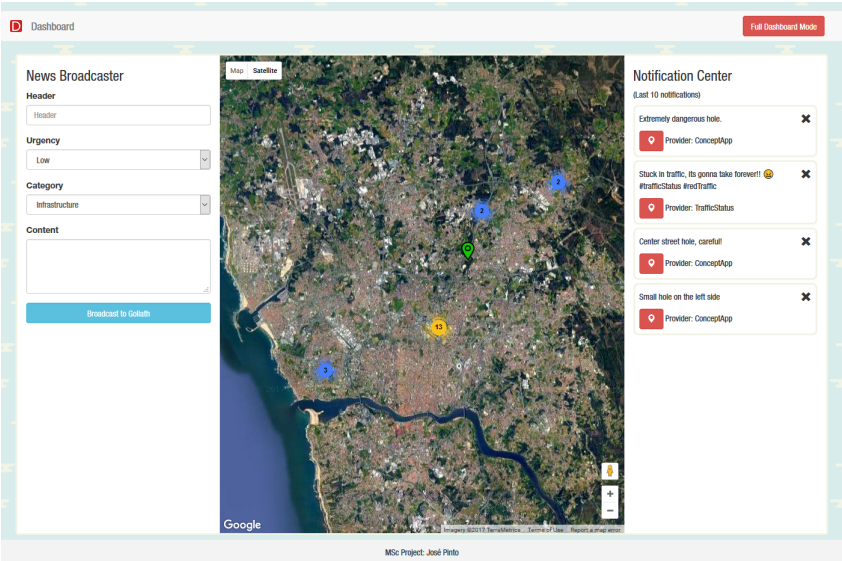

Figure 5.6: City Dashboard interface which integrates 3 parts, News Broadcaster, Google Map instance and Notification Centre (left, centre and right respectively).

the map. The Notification Center will display tiles for every new piece of information, indicating the App provider of that information, and since it is already known which data has geo-located attributes, each tile has a button that focuses the map on the location of that particular piece of information, see Figure 5.6 (p. 58). Lastly, the left panel, the News Broadcaster, is a simple form, that acts as a broadcaster for the operator of the Dashboard and allows submission of pieces of information with content, urgency and category (categories include *infrastructure, culture, traffic, environment and other*). Practically, what happens is that the Dashboard, in the perspective of the framework, is also an App, which means, it can read and write information just like any other App. It reads data from other mobile applications and writes its news pieces that are submitted through the left panel.

The *TrafficStatus* mobile application, in this case, besides writing its tweets to the framework also reads data. It acts as a broadcaster to its users, broadcasting the information sent by the Dashboard. Since every piece is categorised, this application chooses to filter the retrieved data and only display the ones that concern *traffic* related information.

5.2 Deploy Environment

At the moment, for the purpose of these case studies and to validate a real-world integration, the framework, UMI and database are all deployed in a Raspberry Pi 2 Model B[1].

The low specifications of the machine running the platform also serve a goal which envisions being able to run this framework in low-end, low-cost devices, going towards the idea of Citizen-driven innovation to obtain more flexibility. With this project becoming Open Sourced, it

[1]Official Raspberry Pi web page - https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

empowers the average salary developer to easily build-up an Open Data environment and apply his visions and principals.

As for the mobile applications on the first test case, they were both deployed in the same mobile phone, a Samsung S6 Edge Plus. The choice for this device was made because, at that time, it was the resource available.

Table 5.1: Specification for the devices present in the deployment environment.

Device	CPU	RAM	Other
Raspberry Pi 2 Model B	BroadcomBCM2837 Arm7 Quad Core Processor	1 GB	Capable of 1Gpixel/s compute power and features texture filtering and DMA infrastructure
Samsung S6 Edge Plus	Processor Exynos 7420 octa-core (2.1GHz quad + 1.5GHz quad) 64bit	4 GB	5.7-inch QHD (2560x1440, 518 ppi) Super AMOLED Dual edge screen

5.3 Summary

The case studies shown here serve mainly as placeholders that show the different possibilities and scenarios on to which this framework could be applied, since this is just the foundation needed for the development of a larger more complete scenario. The main issue tackled is interoperability, which by itself is an architectural problem nested on the communication layer and does not have a means of measurement. By creating the needed scenarios that outright demonstrate the possible use and complete, normal functioning and performance of the expected and implemented design features, it can validate its overall foundation and try to access its applicability.

Chapter 6

Conclusions and Future Work

6.1 Final Remarks

In the literature review, chapter 2 (p. 5), it is shown that Smart City's biggest challenges come from trying to maintain the quality of life of its habitants while accompanying the global economic growth. Given the approach taken and the city's priorities, these problems can be tackled by empowering the population, providing the basis for Citizen-driven innovation and enabling a governmental Open Data.

Considering the work done for this dissertation, the implemented framework offers a consistent and well-set foundation for an Open Data Environment that partially solves the communication deficits that interoperability between different and various systems creates. By encapsulating the information exchange and isolating the context-aware data provided by the subscribed applications, there are no longer any further problems with the exchange itself. Such encapsulated information is only appended with metadata, useful only for internal processes of the framework. This mediation approach assumes that whichever system consuming information, is bound to do a minimum of data processing in terms of adapting that data to their own systems. As shown in the related work, sections 2.3.1 and 2.3.6, there are several and diverse real-world applications for this framework to be built upon and others that give use to the subscribed data sources. The UMI is a good example since it is not in anyway attached to the framework, and if it was to be removed at any time, the framework could still maintain its normal functioning without any problems, because, every functionality could still be accessed one way or another through the conventional API endpoints, alternatively to the graphical interface.

With the implementation of the framework, there were no concrete validations applied and no measurable metrics were recorded, however, it is important to keep in mind that the case studies in Chapter, 5 (p. 55), served as a validation methodology in this dissertation, as efforts were made to create a case scenario which would tackle every functionality implemented and served the purpose established. Instead of providing a proof-of-concept approach, a real-world scenario complements a better view for validation, it composition includes two Android mobile applications and a web application, the Dashboard.

6.2 Contributions

At the end of this dissertation, the contributions given through this project met the expected ones. Having provided an Open Data enabling framework, which solves the interoperability issues up to a certain extent, given that applications using it do not have to concern themselves with the communication but with just the data that they will eventually have to deal with. It is considered that this framework, given a large enough usage with various and diverse applications, constitutes itself, as a powerful promoter of innovation, specifically citizen-driven-wise. Altogether, the framework's implementation was executed as expected. With the scheduled implementation of Artificial Intelligence modules and/or providing data mining tools on top of the platform, they were proven to be complex and required an already built consistent platform. This platform was then the focus of this work.

As initially stated in the this document, and giving a one-to-one answer to each challenge and their respective contribution to the community, we now state an extended version of the proposed contributions and how they were accomplished.

(i) *Provided an Open Data enabling framework accessible in the format of an Application Program Interface (API) available to any data generating or consuming device*, accessible through the modular set of endpoints provided for application/service subscription that allow the execution of I/O operations open and publicly;

(ii) *Provided a tool to potentially solve interoperability issues on Smart City communicating devices*, by maintaining the state of all present information in the framework and treating each App's injected information as an abstract block that is untouched, throughout the whole process of either read of write, and only merged with necessary metadata;

(iii) *Created an environment with enormous room for improvements and other tools to be implemented, that makes use of the collected information*, by exposing endpoints, modularly, that permit access to the necessary information and functionalities of the framework that allow the creation of tools on top of it;

(iv) *Created an incentive for individual and collective action towards a better Open Data Ecosystem*, by taking an otherwise complex approach to data and simplifying it to atomic read and write actions.

We can conclude that our developed approach met the existing and enumerated flaws as they were encountered during the literature review, giving valuable contributions.

6.3 Future Work

This project as a whole focuses on creating a solid foundation that, with its success leaves a large room for improvements. As such, there are still a few aspects, mostly in terms of implementation, that are to be highlighted.

Starting off with the more basic planned features, that due to time management did not make it to the final version of the framework. It was intended at first, that Apps could have several read profiles and, by having such configurations allows for queries, to the available information, to become much more dynamic and rich with the combination of profiles in the same query and also allowing the Apps to have different read schemas for the same dataset.

The work herein presented will serve to underlie the data integration layer of the MAS-Ter Lab platform [ROB07], an artificial transportation system [RL14] under development at Laboratório de IA e Ciência de Computadores (LIACC).

Other features that are classified as future work are:

(i) **Webhook endpoints:** This functionality would allow for reactive interfaces, decreasing the server's processing efforts and allow for better availability in real-time information for the users;

(ii) **Integrate the framework with a distributed streaming platform like Kafka**[1]: Integrating the platform with Kafka would offer immense advantages in terms of performance and would be a very likely choice given a large scale scenario with increased number of applications subscribed, a larger userbase and an intense and constant number of requests;

(iii) **Allow two different read mode, archive mode and streaming mode:** Different applications have different needs in terms of when displaying or retrieving data, archive mode would provide a preferred access for large quantities of data while streaming mode would only retrieve incoming real-time data;

(iv) **Automatic generation of heatmaps using datasets containing geo-located information:** Having large datasets of geo-located data would easily allow for the creation of heat-map that could be exported or stored, providing time-preserving snapshots of the dataset at that time, while still serving other purposes;

(v) **Data support for Simulation Systems:** The framework serving as a data model support system [KPRG11, TARO10] which can easily be integrated with simulation environment where is taken as a bridge between the data and the simulation system [ARB15, FERO08].

(vi) **Predictive analysis over traffic data:** By becoming a data repository, the framework's storage system and be of use in terms of execution of predictive analysis over traffic data [BAR15, SRM16, LAB+13, FRK+14].

Lastly, in order to better validate this work, the built platform should be tested in a larger environment, with a likely implementation on a local municipality or small scale city. With this, there could be, not just an analysis on the performance, scalability and stability parameters but also the analysis of cultural repercussions on the population and understanding if there was indeed the creation of an environment that allowed both Citizen-driven Innovation and Open Data environments. Examining the applications using the platform would provide a look at the quality of the solution in terms of the interoperability problem and whether or not it managed to help the communication between the systems.

References

[40B] Brilliant open data projects (and how they're redefining smart cities) — carto blog. https://carto.com/blog/forty-brilliant-open-data-project. (Accessed on 06/14/2017). Cited on page 16.

[Aboa] About bingo | bingo. http://www.projectbingo.eu/about-bingo. (Accessed on 06/15/2017). Cited on page 20.

[Abob] About smart nation | enablers. https://www.smartnation.sg/about-smart-nation/enablers. (Accessed on 06/16/2017). Cited on pages 24 and 25.

[App] App. http://www.pegelalarm.at/index-6_app.html?app_btn=LevelAlarm-App. (Accessed on 06/15/2017). Cited on page 20.

[APR10] José RG Alvarado, José LF Pereira, and Rosaldo JF Rossetti. A scalable data support model for traffic simulation in gis. In *[Online] 13th AGILE International Conference On Geographic Information, May*, volume 10, page 14, 2010. Cited on page 37.

[ARB15] Tiago Azevedo, Rosaldo J.F. Rossetti, and Jorge G. Barbosa. A state-of-the-art integrated transportation simulation platform. *2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2015*, (June):340–347, 2015. Cited on page 63.

[Arr] Array of things. http://arrayofthings.github.io/. (Accessed on 06/15/2017). Cited on page 20.

[BAA] Ba accesible | buenos aires ciudad - gobierno de la ciudad autónoma de buenos aires. http://www.buenosaires.gob.ar/aplicacionesmoviles/ba-accesible. (Accessed on 06/15/2017). Cited on page 21.

[BAR15] Joaquim Barros, Miguel Araujo, and Rosaldo J. F. Rossetti. Short-term real-time traffic prediction methods: A survey. *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, (June):132–139, 2015. Cited on page 63.

[BEH06] JA Burke, D Estrin, and M Hansen. Participatory sensing. *ACM Sensys World Sensor Web Workshop*, 2006. Cited on page 13.

[BG05] Christopher R Berry and Edward L Glaeser. The Divergency of Human Capital Levels Across Cities. (September), 2005. Cited on page 11.

[Bla05] Séverine Blanc. Interoperability problems: Management of evolution of collaborative enterprises. *Interoperability for Enterprise Software and*, 5, 2005. Cited on page 5.

[Bos] Boston 311 | city of boston. https://www.cityofboston.gov/311/. (Accessed on 06/14/2017). Cited on page 18.

[Bui] Building the (open) data ecosystem – open knowledge international blog. https://blog.okfn.org/2011/03/31/building-the-open-data-ecosystem/#fn-5084-1. (Accessed on 06/13/2017). Cited on page 16.

[C+08] EU Commission et al. Growing regions, growing europe: Fifth progress report on economic and social cohesion. 2008. Cited on page 11.

[CAR] Carto case study—visualize and analyze information for optimization of public utilities. http://go.carto.com/red-electrica-optimize-public-utilities-casestudy. (Accessed on 06/15/2017). Cited on page 19.

[CDN11] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart Cities in Europe. *Journal of Urban Technology*, 18(2):65–82, 2011. Cited on page 11.

[CFB+13] Giuseppe Cardone, Luca Foschini, Paolo Bellavista, Antonio Corradi, Cristian Borcea, Manoop Talasila, and Reza Curtmola. Fostering participaction in smart cities: A geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, 2013. Cited on page 13.

[Cha] Chattadata | chattadata | city of chattanooga, tn. https://performance.chattanooga.gov/. (Accessed on 06/14/2017). Cited on page 18.

[CHS91] C Collett, M Huhns, and W M Shen. Resource integrating using a large knowledge base in carnot. *Computer Magazine*, 24(12):55–62, 1991. Cited on page 8.

[Cit] Citi bike system data | citi bike nyc. https://www.citibikenyc.com/system-data. (Accessed on 06/15/2017). Cited on page 19.

[Cle] Clear my record | get help clearing your criminal record in counties throughout california. https://clearmyrecord.codeforamerica.org/#learn_more_section. (Accessed on 06/14/2017). Cited on page 17.

[CRHPP09] C-H Chen-Ritzo, C Harrison, J Paraszczak, and F Parr. Instrumenting the planet. *IBM Journal of Research and Development*, 53(3):1–1, 2009. Cited on page 11.

[Cro06] Douglas Crockford. The application/json Media Type for JavaScript Object Notation, 2006. Cited on page 42.

[CSR10] Sara Carvalho, Luís Sarmento, and Rosaldo J. F. Rossetti. Real-time sensing of traffic information in twitter messages. In *4th Workshop on Artificial Transportation Systems and Simulation (ATSS), 2010 13th International IEEE Conference on Intelligent Transportation Systems - (ITSC 2010), Funchal, Portugal, 19-22 Sept. 2010*, pages 1–4, 2010. Cited on page 56.

[Data] Dat. https://datproject.org/about. (Accessed on 06/14/2017). Cited on page 17.

[Datb] Data driven detroit » cdad residential typology analysis. http://datadrivendetroit.org/cdad-residential-typology-analysis/. (Accessed on 06/14/2017). Cited on page 18.

[DB04a] Francis McCabe David Booth, Hugo Haas. Web services architecture. https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest, February 2004. (Accessed on 06/04/2017). Cited on page 8.

[DB04b] Francis McCabe David Booth, Hugo Haas. Web services architecture. https://www.w3.org/TR/2003/WD-ws-arch-20030808/, February 2004. (Accessed on 06/04/2017). Cited on page 8.

[Dea12] Mark Deakin. *Creating Smart-Er Cities*, volume 2012. 2012. Cited on pages 1 and 10.

[dev] developers » fiware » page 2. https://www.fiware.org/tag/developers/page/2/. (Accessed on 06/15/2017). Cited on page 23.

[DIG] Digital matatus. http://www.digitalmatatus.com/about.html. (Accessed on 06/15/2017). Cited on page 19.

[Dur] Durham open data. https://opendurham.nc.gov/page/aboutus/. (Accessed on 06/14/2017). Cited on page 18.

[Ena] Enabling collective action to manage climate risks. http://www.prepdata.org/. (Accessed on 06/15/2017). Cited on page 20.

[Est] Establish an open data ecosystem. http://opendatahandbook.org/solutions/en/Open-Data-Ecosystem/. (Accessed on 06/12/2017). Cited on page 15.

[Eura] European legislation on reuse of public sector information | digital single market. https://ec.europa.eu/digital-single-market/en/european-legislation-reuse-public-sector-information. (Accessed on 06/12/2017). Cited on page 15.

[Eurb] Eurostat, young europeans. http://ec.europa.eu/eurostat/cache/infographs/youth/index_en.html. (Accessed on 06/14/2017). Cited on page 19.

[FAU+10] Future Media Internet-task Force, Media Coordination Action, Networked Media Unit, D G Information Society, and European Commission. *Research Challenges and the Road Ahead*. Number April. 2010. Cited on page 12.

[FERO08] Paulo A. F. Ferreira, Edgar F. Esteves, Rosaldo J. F. Rossetti, and Eugénio C. Oliveira. A Cooperative Simulation Framework for Traffic and Transportation Engineering. In *Cooperative Design, Visualization, and Engineering*, pages 89–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. Cited on page 63.

[Fie00] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. *Building*, 54:162, 2000. Cited on page 9.

[FIW] Fiware technologies. http://www.europeanpioneers.eu/en/fiware-technologies.html. (Accessed on 06/14/2017). Cited on page 22.

[FMS+11] Rod Franklin, Andreas Metzger, Michael Stollberg, Yagil Engel, Kay Fjørtoft, René Fleischhauer, Clarissa Marquezan, and Lone Sletbakk Ramstad. Future Internet Technology for the Future of Transport and Logistics. pages 290–301, 2011. Cited on page 23.

[FRK+14] João Filgueiras, Rosaldo J F Rossetti, Zafeiris Kokkinogenis, Michel Ferreira, Cristina Olaverri-Monreal, Marco Paiva, João Manuel R S Tavares, and Joaquim Gabriel. Sensing bluetooth mobility data: Potentials and applications. *Advances in Intelligent Systems and Computing*, 262:419–432, 2014. Cited on page 63.

[Get] Getcalfresh.org. https://getcalfresh.org/. (Accessed on 06/15/2017). Cited on page 21.

[GKS92] David V Gibson, George Kozmetsky, and Raymond W Smilor. The technopolis phenomenon: Smart cities, fast systems. *Global Networks*, pages 756–767, 1992. Cited on page 11.

[GMS94] Cheng Hian Gol, Stuart E. Madnick, and Michael Siegel. Context Interchange : Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment. *Proceedings of the third international conference on Information and knowledge management*, (February):337–346, 1994. Cited on page 6.

[GYL11] R.K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011. Cited on page 14.

[GYZZ14] Bin Guo, Zhiwen Yu, Xingshe Zhou, and Daqing Zhang. From participatory sensing to Mobile Crowd Sensing. *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PERCOM WORKSHOPS 2014*, pages 593–598, 2014. Cited on page 14.

[Ho16] E. Ho. Smart subjects for a Smart Nation? Governing (smart)mentalities in Singapore. *Urban Studies*, 2016. Cited on page 25.

[Hol08] Robert G. Hollands. Will the real smart city please stand up? *City*, 12(3):303–320, 2008. Cited on pages 10 and 11.

[Homa] Home | datahaven. http://www.ctdatahaven.org/. (Accessed on 06/14/2017). Cited on page 18.

[Homb] Home | fiware catalogue. https://catalogue.fiware.org/. (Accessed on 06/14/2017). Cited on pages 22 and 23.

[Homc] Home | openlaws. https://openlaws.com/home. (Accessed on 06/14/2017). Cited on page 18.

[Imp] Improving commercial real estate data in detroit | localdata - better data makes beter cities. http://localdata.com/case-detroit.html. (Accessed on 06/14/2017). Cited on page 17.

[int] interactive.investigativepost.org/lead-buffalo-2016/. http://interactive.investigativepost.org/lead-buffalo-2016/. (Accessed on 06/14/2017). Cited on page 18.

[Job] Job map. http://coaxs.mit.edu/job-map/. (Accessed on 06/15/2017). Cited on page 19.

[KCS] Kc streetcar - ridekc | free to ride - public transportation system. http://kcstreetcar.org/. (Accessed on 06/15/2017). Cited on page 20.

[KFC+15] Zafeiris Kokkinogenis, João Filguieras, Sara Carvalho, Luís Sarmento, and Rosaldo J.F. Rossetti. Chapter 12 - mobility network evaluation in the user perspective: Real-time sensing of traffic information in twitter messages. In Rosaldo J.F. Rossetti and Ronghui Liu, editors, *Advances in Artificial Transportation Systems and Simulation*, pages 219 – 234. Academic Press, Boston, 2015. Cited on page 56.

[KMR+14] Z. Kokkinogenis, N. Monteiro, R. J. F. Rossetti, A. L. C. Bazzan, and P. Campos. Policy and incentive designs evaluation: A social-oriented framework for artificial transportation systems. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 151–156, Oct 2014. Cited on page 54.

[KPRG11] Zafeiris Kokkinogenis, Lúcio Sanchez Passos, Rosaldo Rossetti, and Joaquim Gabriel. Towards the next-generation traffic simulation tools: a first evaluation. *6th Iberian Conference on Information Systems and Technologies*, pages 15–18, 2011. Cited on page 63.

[LAB+13] Siyuan Liu, Miguel Araujo, Emma Brunskill, Rosaldo Rossetti, Joao Barros, and Ramayya Krishnan. Understanding sequential decisions via inverse reinforcement learning. *Proceedings - IEEE International Conference on Mobile Data Management*, 1:177–186, 2013. Cited on page 63.

[LNS+07] David Lieberman, Marion Nadel, Robert A. Smith, Wendy Atkin, Subash B. Duggirala, Robert Fletcher, Seth N. Glick, C. Daniel Johnson, Theodore R. Levin, John B. Pope, Michael B. Potter, David Ransohoff, Douglas Rex, Robert Schoen, Paul Schroy, and Sidney Winawer. Standardized colonoscopy reporting and data system: report of the Quality Assurance Task Group of the National Colorectal Cancer Roundtable. *Gastrointestinal Endoscopy*, 65(6):757–766, 2007. Cited on page 6.

[Mah15] Arun Mahizhnan. Smart Cities. 16(1):13–18, 2015. Cited on pages 24 and 25.

[Map] Mapping police violence. https://mappingpoliceviolence.org/. (Accessed on 06/15/2017). Cited on page 21.

[MAR] Marta. http://www.itsmarta.com/marta-on-the-go.aspx. (Accessed on 06/15/2017). Cited on page 19.

[MMD15] Eva María, Olmedo Moreno, and Adrián López Delgado. De La Smart City a La Smart Human City. Inclusión Digital En Aplicaciones From Smart Cities To Smart Human Cities: Digital Inclusion in App'S. *Revista Fuentes*, 17:41–65, 2015. Cited on page 26.

[MMY09] Jian Meng, Shujun Mei, and Zhao Yan. RESTful Web Services: A Solution for Distributed Data Integration. *International Conference on Computational Intelligence and Software Engineering*, pages 1–4, 2009. Cited on page 9.

[MO13] Catherine EA Mulligan and Magnus Olsson. Architectural implications of smart city business models: an evolutionary perspective. *IEEE Communications Magazine*, 51(6):80–85, 2013. Cited on page 16.

[MPS+14] Silvia Mirri, Catia Prandi, Paola Salomoni, Franco Callegati, and Aldo Campi. On combining crowdsourcing, sensing and open data for an accessible smart city. *Proceedings - 2014 8th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST 2014*, pages 294–299, 2014. Cited on page 16.

[MSTR12] J. Macedo, M. Soares, I. Timóteo, and R. J. F. Rossetti. An approach to advisory-based traffic control. In *7th Iberian Conference on Information Systems and Technologies (CISTI 2012)*, pages 1–6, June 2012. Cited on page 37.

[Mug03] James A Muguira. The Levels of Conceptual Interoperability Model A Traditional View of Levels of Information Systems Interoperability The Levels of Conceptual Interoperability. (September), 2003. Cited on page 5.

[NMF+10] Future Media Networks, Networked Media, Systems Fp, D G Information Society, and European Commission. *Future Media Networks research challenges 2010*. Number October. 2010. Cited on page 12.

[NOA] Noah. http://noah.dost.gov.ph/#/. (Accessed on 06/15/2017). Cited on page 20.

[NOL] Nolalytics - datadriven - city of new orleans. https://datadriven.nola.gov/nolalytics/. (Accessed on 06/14/2017). Cited on page 18.

[Not] Notes on open data ecosystems | open data research network. http://www.opendataresearch.org/content/2014/719/notes-open-data-ecosystems. (Accessed on 06/13/2017). Cited on page 15.

[Opea] Open data impact map. http://opendataimpactmap.org/. (Accessed on 06/14/2017). Cited on page 17.

[Opeb] Open definition 2.1 - open definition - defining open in open data, open content and open knowledge. http://opendefinition.org/od/2.1/en/. (Accessed on 06/13/2017). Cited on pages 15 and 16.

[Opec] Open states: discover politics in your state - open states. https://openstates.org/. (Accessed on 06/14/2017). Cited on page 18.

[Oped] Opengrid - city of chicago. http://opengrid.io/. (Accessed on 06/14/2017). Cited on page 17.

[Par] Partners | autolib'. https://www.autolib.eu/en/our-commitment/partners/. (Accessed on 06/15/2017). Cited on page 20.

[PPSR17] João Pereira, Arian Pasquali, Pedro Saleiro, and Rosaldo JF Rossetti. Transportation in social media: an automatic classifier for travel-related tweets. In *Progress in Artificial Intelligence, EPIA 2017, Porto, Portugal, September 5-8, 2017. Proceedings*. Springer International Publishing, 2017. Cited on page 13.

[Pro] Proof of concept: Using predictive modeling to prioritize building inspections | urban spatial. http://urbanspatialanalysis.com/portfolio/proof-of-concept-using-predictive-modeling-to-prioritize-building-inspecti (Accessed on 06/14/2017). Cited on page 18.

[RAKG13] R. J. F. Rossetti, J. E. Almeida, Z. Kokkinogenis, and J. Gonçalves. Playing transportation seriously: Applications of serious games to artificial transportation systems. *IEEE Intelligent Systems*, 28(4):107–112, July 2013. Cited on page 54.

[RL14] Rosaldo JF Rossetti and Ronghui Liu. Advances in artificial transportation systems and simulation, 2014. Cited on page 63.

[RLCB02] Rosaldo JF Rossetti, Ronghui Liu, Helena BB Cybis, and Sergio Bampi. A multi-agent demand model. In *Proceedings of the 13th Mini-Euro Conference and The 9th Meeting of the Euro Working Group Transportation*, pages 193–198, 2002. Cited on page 54.

[ROB07] Rosaldo JF Rossetti, Eugénio C Oliveira, and Ana LC Bazzan. Towards a specification of a framework for sustainable transportation analysis. In *13th Portuguese Conference on Artificial Intelligence, EPIA, Guimarães, Portugal*, pages 179–190. APPIA, 2007. Cited on page 63.

[Roy03] Rahul Roy. The dynamics of opening government data. *Information Systems*, 86(Mitnick):921–927, 2003. Cited on page 15.

[RSR96] Scott A Renner, James G Scarano, and Arnon S Rosenthal. Data interoperability: Standardization or Mediation. *1st IEEE metadata conference*, pages 1–8, 1996. Cited on pages 6 and 7.

[RSR15] Francisco Rebelo, Carlos Soares, and Rosaldo J F Rossetti. TwitterJam: Identification of mobility patterns in urban centers based on tweets. *2015 IEEE 1st International Smart Cities Conference, ISC2 2015*, 2015. Cited on page 56.

[Saf] Safecast. https://blog.safecast.org/. (Accessed on 06/15/2017). Cited on page 20.

[SAKR15] Semantically-modeled Multidatabase Systems, Yigal Arens, Craig A Knoblock, and Marina Rey. Planning and Reformulating Queries for Planning and Reformulating Queries for Semantically-Modeled Multidatabase Systems 1 Introduction. (December), 2015. Cited on page 8.

[See] Seeclickfix. http://commonwealthconnect.io/. (Accessed on 06/14/2017). Cited on page 17.

[SJK+12] Wanita Sherchan, Prem P. Jayaraman, Shonali Krishnaswamy, Arkady Zaslavsky, Seng Loke, and Abhijat Sinha. Using on-the-move mining for mobile crowdsensing. *Proceedings - 2012 IEEE 13th International Conference on Mobile Data Management, MDM 2012*, pages 115–124, 2012. Cited on page 14.

[SKP+11] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. Smart cities and the future internet: Towards cooperation frameworks for open innovation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6656:431–446, 2011. Cited on pages 1, 11, and 12.

[Smaa] Smart city | strawberry energy. http://senergy.rs/benefits/pametan-grad/?lang=en. (Accessed on 06/15/2017). Cited on page 21.

[Smab] Smart columbus home. https://www.columbus.gov/smartcity/. (Accessed on 06/15/2017). Cited on page 19.

[Smac] Smart nation singapore. https://www.smartnation.sg/. (Accessed on 06/15/2017). Cited on page 21.

[Smad] Smartappcity at the fiware summit. https://www.slideshare.net/FI-WARE/smartappcity-at-the-fiware-summit. (Accessed on 06/16/2017). Cited on page 26.

[Sma15] Smart communities. https://web.archive.org/web/20150420032215/http://www.smartcommunities.org/about.htm, April 2015. (Accessed on 06/08/2017). Cited on page 11.

[SR15] A. Seliverstov and R. J. F. Rossetti. An ontological approach to spatio-temporal information modelling in transportation. In *2015 IEEE First International Smart Cities Conference (ISC2)*, pages 1–6, Oct 2015. Cited on page 7.

[SRM16] Miguel Sandim, Rosaldo J F Rossetti, and Daniel C Moura. A Systematic Review on Using AVL Data to Obtain and Predict Traffic Network Performance Metrics. *19th International Conference on Intelligent Transportation Systems*, pages 1692–1699, 2016. Cited on page 63.

[SSR94] Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, 1994. Cited on page 8.

[TARO10] Ivo J P M Timóteo, Miguel R. Araújo, Rosaldo J F Rossetti, and Eugenio C. Oliveira. TraSMAPI: An API oriented towards multi-agent systems real-time interaction with multiple traffic simulators. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1183–1188, 2010. Cited on page 63.

[Tel] Tel aviv-yafo municipalityresident's card. https://www.tel-aviv.gov.il/en/Live/ResidentsCard/Pages/default.aspx. (Accessed on 06/14/2017). Cited on page 17.

[Thea] The app of smart cities - smartappcity. http://www.smartappcity.com/en/. (Accessed on 06/14/2017). Cited on page 17.

[Theb] The displacement alert project map | anhd. http://map.dapmapnyc.org/app/#openModal. (Accessed on 06/15/2017). Cited on page 21.

[Urb] Urbanflow helsinki. http://helsinki.urbanflow.io/. (Accessed on 06/15/2017). Cited on page 21.

[USRS16] Daniela Ulloa, Pedro Saleiro, Rosaldo J. F. Rossetti, and Elis Regina Silva. Mining Social Media for Open Innovation in Transportation Systems. 2016. Cited on page 54.

[WLA17] A Wright, G Luff, and H Andrews. JSON Schema Validation: A Vocabulary for Structural Validation of JSON. pages 1–21, 2017. Cited on pages 42 and 44.

[Wri16] Austin Wright. JSON Schema: A Media Type for Describing JSON Documents. (draft-wright-json-schema-00):1–18, 2016. Cited on pages 42, 43, and 44.

Appendix A

Appendix

A.1 Marginal Research Effort

A.1.1 A Hands-on Approach on Botnets for Behavior Exploration

Abstract

A botnet consists of a network of computers that run a special software that allows a third-party to remotely control them. This characteristic presents a major issue regarding security in the Internet. Although common malicious software infect the network with almost immediate visible consequences, there are cases where that software acts stealthy without direct visible effects on the host machine. This is the normal case of botnets. However, not always the bot software is created and used for illicit purposes. There is a need for further exploring the concepts behind botnets and network security. For this purpose, this paper presents and discusses an educational tool that consists of an open-source botnet software kit with built-in functionalities. The tool enables anyone with some computer technical knowledge, to experiment and find out how botnets work and can be changed and adapted to a variety of useful applications, such as introducing and exemplifying security and distributed systems' concepts.

A.1.2 Three dimensional modelling of Porto's network for electric mobility simulation

Abstract

Elevation data is very important for precise electric vehicle simulation. However, traffic simulators are often strictly two-dimensional and do not offer the capability of modelling urban networks taking elevation into account. In particular, SUMO - Simulation of Urban Mobility, an often used microscopic traffic simulation platform, relies on urban networks previously modelled with elevation data in order to provide access to this information during simulations. This work intends on tackling the problem of how to add this elevation data to urban network models - in particular for the case of the Porto urban network, in Portugal. With this goal in mind, a comparison

between different altitude information retrieval approaches is made and a simple tool to annotate network models with altitude data is proposed. This paper starts by describing the methodological approach followed to develop the work, then describing and analysing its main findings. This description includes an in-depth explanation of the proposed tool. Lastly, this paper reviews some related work to this subject.

A.1.3 A systematic review protocol on shared transportation

Abstract

Recently, many studies in the literature have been intensified in the scope of mobility services, presenting new technological solutions that can minimize problems, such as traffic congestion and the emission of greenhouse gases resulting from the population growth in large cities. Some services, namely shared services, are mainly dedicated to trying to encourage people to use transport that can be shared. In this paper, we propose a systematic literature review protocol oriented to shared transportation, so as to support bibliographical reviews by researchers on this domain. From the proposed systematic review methodology, we present preliminary results of a survey conducted on shared services.

A.2 Framework's API YAML Documentation

The following documentation is generated automatically by Swagger. The listing A.1 provide a template example of how the documentation is structures, every entry is an available endpoint.

```
{"<endpoint URI>",methods="<method type>",consumes="<media-type>",produces="<media-type>"} :
{
  bean : "<Spring bean in charge of the logic behing the endpoint>",
  method : "<java method name, containing the endpoint logic>"
}
```

Listing A.1: Swagger Auto-generated API Documentation Template Endpoint Entry

```
---
"/webjars/**":
  bean: resourceHandlerMapping
"/**":
  bean: resourceHandlerMapping
"/**/favicon.ico":
  bean: faviconHandlerMapping
"{[/subscribe/update/write],methods=[POST],consumes=[application/json],produces=[application/json]}":
  bean: requestMappingHandlerMapping
  method: public org.springframework.http.ResponseEntity<java.lang.String> com.goliath.subscritions.SubscriptionsController.updateAppWriteSchema(java.lang.String)
```

```
11 "{[/subscribe/update/read],methods=[POST],consumes=[application/json],produces=[
     application/json]}":
12   bean: requestMappingHandlerMapping
13   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.subscritions.SubscriptionsController.updateAppReadSchema(java.lang.
       String)
14 "{[/subscribe],methods=[POST],consumes=[application/json],produces=[application/
     json]}":
15   bean: requestMappingHandlerMapping
16   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.subscritions.SubscriptionsController.subscribe(java.lang.String)
17 "{[/write],methods=[POST],consumes=[application/json],produces=[application/json]}"
     :
18   bean: requestMappingHandlerMapping
19   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.io.WriterController.write(java.lang.String,java.util.Map<java.lang.
       String,
20   java.lang.String>,javax.servlet.http.HttpServletRequest)
21 "{[/read],methods=[POST],consumes=[application/json],produces=[application/json]}":
22   bean: requestMappingHandlerMapping
23   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.io.ReaderController.write(java.lang.String)
24 "{[/apps/delete],methods=[DELETE],consumes=[application/json],produces=[application
     /json]}":
25   bean: requestMappingHandlerMapping
26   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.controllers.WebController.deleteApp(java.lang.String)
27 "{[/apps/available],methods=[GET],produces=[application/json]}":
28   bean: requestMappingHandlerMapping
29   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.controllers.WebController.getAvailableData(java.lang.String)
30 "{[/apps],methods=[POST],consumes=[application/json],produces=[application/json]}":
31   bean: requestMappingHandlerMapping
32   method: public org.springframework.http.ResponseEntity<java.lang.String> com.
       goliath.controllers.WebController.getApps(java.lang.String)
33 "{[/v2/api-docs],methods=[GET],produces=[application/json || application/hal+json]}
     ":
34   bean: requestMappingHandlerMapping
35   method: public org.springframework.http.ResponseEntity<springfox.documentation.
       spring.web.json.Json>
36   springfox.documentation.swagger2.web.Swagger2Controller.getDocumentation(java.
       lang.String,javax.servlet.http.HttpServletRequest)
37 "{[/configuration/ui]}":
38   bean: requestMappingHandlerMapping
39   method: org.springframework.http.ResponseEntity<springfox.documentation.swagger.
       web.UiConfiguration>
40   springfox.documentation.swagger.web.ApiResourceController.uiConfiguration()
41 "{[/swagger-resources]}":
42   bean: requestMappingHandlerMapping
```

```
43    method: org.springframework.http.ResponseEntity<java.util.List<springfox.
          documentation.swagger.web.SwaggerResource>>
44    springfox.documentation.swagger.web.ApiResourceController.swaggerResources()
45  "{[/configuration/security]}":
46    bean: requestMappingHandlerMapping
47    method: org.springframework.http.ResponseEntity<springfox.documentation.swagger.
          web.SecurityConfiguration>
48    springfox.documentation.swagger.web.ApiResourceController.securityConfiguration()
49  "{[/error],produces=[text/html]}":
50    bean: requestMappingHandlerMapping
51    method: public org.springframework.web.servlet.ModelAndView org.springframework.
          boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.
          HttpServletRequest,javax.servlet.http.HttpServletResponse)
52  "{[/error]}":
53    bean: requestMappingHandlerMapping
54    method: public org.springframework.http.ResponseEntity<java.util.Map<java.lang.
          String,
55    java.lang.Object>> org.springframework.boot.autoconfigure.web.
          BasicErrorController.error(javax.servlet.http.HttpServletRequest)
56  "{[/env/{name:.*}],methods=[GET],produces=[application/vnd.spring-boot.actuator.v1+
        json || application/json]}":
57    bean: endpointHandlerMapping
58    method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EnvironmentMvcEndpoint.value(java.lang.String)
59  "{[/env || /env.json],methods=[GET],produces=[application/vnd.spring-boot.actuator.
        v1+json || application/json]}":
60    bean: endpointHandlerMapping
61    method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
62  "{[/info || /info.json],methods=[GET],produces=[application/vnd.spring-boot.
        actuator.v1+json || application/json]}":
63    bean: endpointHandlerMapping
64    method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
65  "{[/mappings || /mappings.json],methods=[GET],produces=[application/vnd.spring-boot
        .actuator.v1+json || application/json]}":
66    bean: endpointHandlerMapping
67    method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
68  "{[/trace || /trace.json],methods=[GET],produces=[application/vnd.spring-boot.
        actuator.v1+json || application/json]}":
69    bean: endpointHandlerMapping
70    method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
71  "{[/auditevents || /auditevents.json],methods=[GET],produces=[application/vnd.
        spring-boot.actuator.v1+json || application/json]}":
72    bean: endpointHandlerMapping
73    method: public org.springframework.http.ResponseEntity<?> org.springframework.
          boot.actuate.endpoint.mvc.AuditEventsMvcEndpoint.
```

```
      findByPrincipalAndAfterAndType(java.lang.String,java.util.Date,java.lang.
      String)
"{[/health || /health.json],methods=[GET],produces=[application/vnd.spring-boot.
    actuator.v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      HealthMvcEndpoint.invoke(javax.servlet.http.HttpServletRequest)
"{[/configprops || /configprops.json],methods=[GET],produces=[application/vnd.
    spring-boot.actuator.v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      EndpointMvcAdapter.invoke()
"{[/heapdump || /heapdump.json],methods=[GET],produces=[application/octet-stream]}"
    :
  bean: endpointHandlerMapping
  method: public void org.springframework.boot.actuate.endpoint.mvc.
      HeapdumpMvcEndpoint.invoke(boolean,javax.servlet.http.HttpServletRequest,
      javax.servlet.http.HttpServletResponse)
  throws java.io.IOException,javax.servlet.ServletException
"{[/dump || /dump.json],methods=[GET],produces=[application/vnd.spring-boot.
    actuator.v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      EndpointMvcAdapter.invoke()
"{[/beans || /beans.json],methods=[GET],produces=[application/vnd.spring-boot.
    actuator.v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      EndpointMvcAdapter.invoke()
"{[/metrics/{name:.*}],methods=[GET],produces=[application/vnd.spring-boot.actuator
    .v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      MetricsMvcEndpoint.value(java.lang.String)
"{[/metrics || /metrics.json],methods=[GET],produces=[application/vnd.spring-boot.
    actuator.v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      EndpointMvcAdapter.invoke()
"{[/loggers/{name:.*}],methods=[GET],produces=[application/vnd.spring-boot.actuator
    .v1+json || application/json]}":
  bean: endpointHandlerMapping
  method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
      LoggersMvcEndpoint.get(java.lang.String)
? "{[/loggers/{name:.*}],methods=[POST],consumes=[application/vnd.spring-boot.
    actuator.v1+json || application/json],produces=[application/vnd.spring-boot.
    actuator.v1+json || application/json]}"
  : bean: endpointHandlerMapping
```

```
101   method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          LoggersMvcEndpoint.set(java.lang.String,java.util.Map<java.lang.String,
102   java.lang.String>)
103 "{[/loggers || /loggers.json],methods=[GET],produces=[application/vnd.spring-boot.
      actuator.v1+json || application/json]}":
104   bean: endpointHandlerMapping
105   method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
106 "{[/logfile || /logfile.json],methods=[GET || HEAD]}":
107   bean: endpointHandlerMapping
108   method: public void org.springframework.boot.actuate.endpoint.mvc.
          LogFileMvcEndpoint.invoke(javax.servlet.http.HttpServletRequest,javax.servlet
          .http.HttpServletResponse)
109   throws javax.servlet.ServletException,java.io.IOException
110 "{[/autoconfig || /autoconfig.json],methods=[GET],produces=[application/vnd.spring-
      boot.actuator.v1+json || application/json]}":
111   bean: endpointHandlerMapping
112   method: public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.
          EndpointMvcAdapter.invoke()
```

Listing A.2: Swagger Auto-generated API Documentation