



Grant Agreement No.: 723174
Call: H2020-ICT-2016-2017
Topic: ICT-38-2016 - MEXICO: Collaboration on ICT
Type of action: RIA



D2.5: Reference architectures for data-intensive and IoT-based Smart City, Smart Healthcare and Smart Security applications

Revision: v.1.0

Work package	WP 2
Task	Task 2.5
Due date	30/06/2018
Submission date	26/07/2018
Deliverable lead	ITESM
Version	1.0
Authors	Nestor Velasco-Bermeo (ITESM), Joanna Alvarado (ITESM), Miguel González (ITESM), Hugo Estrada, Samuel Jiménez, Yolanda Baca (INFOTEC), Netzahualcóyotl Hernández (CICESE), Luis Valentin (INAOE), Alma Rios (INAOE), Miguel Palacios (INAOE), Blanca Onofre (CENIDET), Alicia Martínez (CENIDET), Daniel Torres (CENIDET)
Reviewers	Federico M. Facca (MARTEL); Tomas Aliaga (MARTEL)

Abstract	The following report presents not only the current status of the SmartSDK applications in their corresponding scenarios but also presents the Road Map of all the different developed modules. Furthermore, a collection of generalized reference
----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	architectures that are self-contained functional blocks are provided to facilitate the development of new Smart Applications.
Keywords	Reference Architectures, FIWARE, Recipes.

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1.0	23/07/2018	Overall Review	Tomas Aliaga (MARTEL)
V0.2	21/07/2018	Updated all sections for final version	Nestor Velasco Bermeo (ITESM), Miguel Palacios (INAOE), Blanca Haidee Onofre Ramirez (CENIDET)
V0.1	20/06/2018	Draft version	Nestor Velasco Bermeo (ITESM)

Disclaimer

The information, documentation and figures available in this deliverable, is written by the SmartSDK (A FIWARE-based Software Development Kit for Smart Applications for the needs of Europe and Mexico) – project consortium under EC grant agreement 723174 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

Copyright notice

© 2016 - 2018 SmartSDK Consortium

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		R
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CI	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to SmartSDK project and Commission Services	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

EXECUTIVE SUMMARY

SmartSDK aims to provide a set of ready-to-use “recipes” to develop smart applications in the Smart City, Smart Healthcare, and Smart Security domains. Such recipes are based on: components (i.e. Generic Enablers and Specific Enablers), data models (i.e. NGSI formalisation of the data exchanged among components) and reference architectures (i.e. the combination of components and data models to support production grade requirements).

SmartSDK already provided a set of Generic Reference Architectures based on the combination of typical FIWARE architecture patterns and cloud architecture patterns (c.f. D3.1 - SmartSDK Reference Models and Recipes). This deliverable takes on from that general work and the architectures developed in the context of the SmartSDK applications to abstract a set of generalized reference architectures (i.e. FIWARE-based architecture solutions) to cope with typical functionalities provided by smart applications in different domains. The deliverable besides covering the current status of the SmartSDK applications’ architectures, summarizes the Road Map of the different modules developed to solve specific Smart Application needs. Finally, for each application domain covered by SmartSDK, the deliverable presents a set of reference architectures. Such architectures are self-contained functional blocks derived from the actual architecture of the Applications developed in SmartSDK and provide reusable building blocks to facilitate the development of new Smart Applications.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	2
TABLE OF CONTENTS	3
LIST OF FIGURES	4
ABBREVIATIONS	5
1 INTRODUCTION	6
1.1 Structure of the deliverable	6
1.2 Audience	6
2 SMARTSDK ARCHITECTURE PATTERNS	7
3 SMART CITY REFERENCE ARCHITECTURE	11
3.1 Smart City Scenario Description	11
3.2 Smart City Scenario Architecture in SmartSDK	11
3.3 Smart City Generalized Architecture Patterns	12
Environment monitoring Architecture pattern	12
Transport Routing Architecture pattern	13
Mobile Alerts Architecture pattern	14
User Management Architecture pattern	15
3.4 Roadmap	16
4 SMART HEALTH REFERENCE ARCHITECTURE	18
4.1 Smart Health Scenario Description	18
4.2 Smart Health Scenario Architecture in SmartSDK	19
4.3 Smart Health Generalized Architecture Patterns	19
4.4 Roadmap	21
5 SMART SECURITY REFERENCE ARCHITECTURE	23
5.1 Smart Security Scenario Description	23
5.2 Smart Security Scenario Architecture in SmartSDK	23
5.3 Smart Security Generalized Architecture Patterns	24
Video Event Detection Architecture pattern	24
Event Detection Through Smartphone Architecture pattern	25
Alerts Architecture pattern	25
5.4 Roadmap	26
6 CONCLUSIONS	27

LIST OF FIGURES

Figure 1: FIWARE Reference Architecture for IoT and Data intensive applications	7
Figure 2: SmartSDK extended FIWARE Reference Architecture for IoT and Data intensive applications.	9
Figure 3: SmartSDK extended FIWARE Reference Architecture for IoT and Data intensive applications.	10
Figure 4: Complete architecture of Smart City Scenario	12
Figure 5: Environment monitoring pattern	13
Figure 6: Transport Routing Architecture pattern	14
Figure 7: Mobile Alerts Architecture pattern	15
Figure 8: User Management Architecture pattern	16
Figure 9: SmartHealth Scenario Architecture	19
Figure 10: Sensible Data Collection Architecture pattern	20
Figure 11: Data Analysis Architecture pattern	21
Figure 12: Smart Security Scenario Architecture	24
Figure 13: Video Event Detection Architecture pattern	24
Figure 14: Event Detection Through Smartphone Architecture pattern	25
Figure 15: Alerts Architecture Pattern	26

ABBREVIATIONS

API	Application Programming Interface
DNS	Domain Name Server
GE	Generic Enabler
GEri	Generic Enabler reference implementation
HA	High Availability
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
REST	REpresentational State Transfer
TCP	Transmission Control Protocol
Poi	Point of Interest
LWM2M	Lightweight Machine to Machine
MQTT	Message Queue Telemetry Transport
GTFS	General Transit Feed Specification
TUG	Timed Up and Go
HDFS	Hadoop Distributed File System

1 INTRODUCTION

SmartSDK has a strong foundation due to its standardized data models, thus providing not just a structured way of collecting, storing and serving data but also key capabilities such as scalability and reusability. The benefits of such capabilities provide potential developers to adopt not only the data models but also refer to the recipes that will be presented in the following document. The recipes described in the upcoming sections are based on a top-bottom approach. They start from the particular solution that's been developed to attend the specific needs of each domain: Smart City, Smart Healthcare and Smart Security. Each domain's Architectures are described accordingly, providing an insightful view of all the components and their interaction. Furthermore, each Scenario is described to provide the reader a clear picture of its main goals and considerations. While each scenario depicts a particular problem, the reader will be presented with generalized Architecture Patterns that will attend different problems within the scope of each scenario. The goal of such generalized Architecture Patterns is to prove the benefits that smartSDK provides by being flexible enough to generate a new solution by reusing components that weren't originally conceived to solve such new problem. Finally, each scenario provides a Roadmap of the components that will be created to further improve the provided solution and a description of how the new elements will interact.

1.1 Structure of the deliverable

The Deliverable is structured as follows.

Section 2 presents the different components (Data/Context Management and IoT Services Enablement) in a southbound-northbound macro architecture. The section also introduces SmartSDK's novel enablers in a reference architecture for both the Data/Context Management and IoT Services chapters. It also provides an insight of how SmartSDK extended FIWARE reference Architectures in derived subsets to be adopted in specific smart scenarios.

Sections 3, 4 and 6 describes the general considerations and aspect of the Smart City, Smart Health and Smart Security scenarios respectively. Each section presents the general Architecture in which it is being built in SmartSDK. The generalized Architecture patterns are presented to implement new solutions formulated upon the scenario's general architecture. Finally, an overall roadmap is outlined to provide a better understanding of the currently created and upcoming components.

1.2 Audience

This deliverable is mainly intended for:

- ➔ Developers and Operators interested into deploy FIWARE Smart applications in a production context.
- ➔ Developers and Knowledge modellers interested into adopting FIWARE data models or contributing to the initiative;

2 SMARTSDK ARCHITECTURE PATTERNS

Figure 1 presents the different components of the Data/Context Management chapter and of the IoT Service Enablement chapter as a southbound-northbound architecture: this is a macro architecture pattern representing how data can be “lifted” in a interoperable format through a standardized protocol (NGSI) and a set of associated standardized data models (FIWARE Data Models¹). Developers, depending on their needs, can select to adopt a given subset of components in their application and the relevant data models.

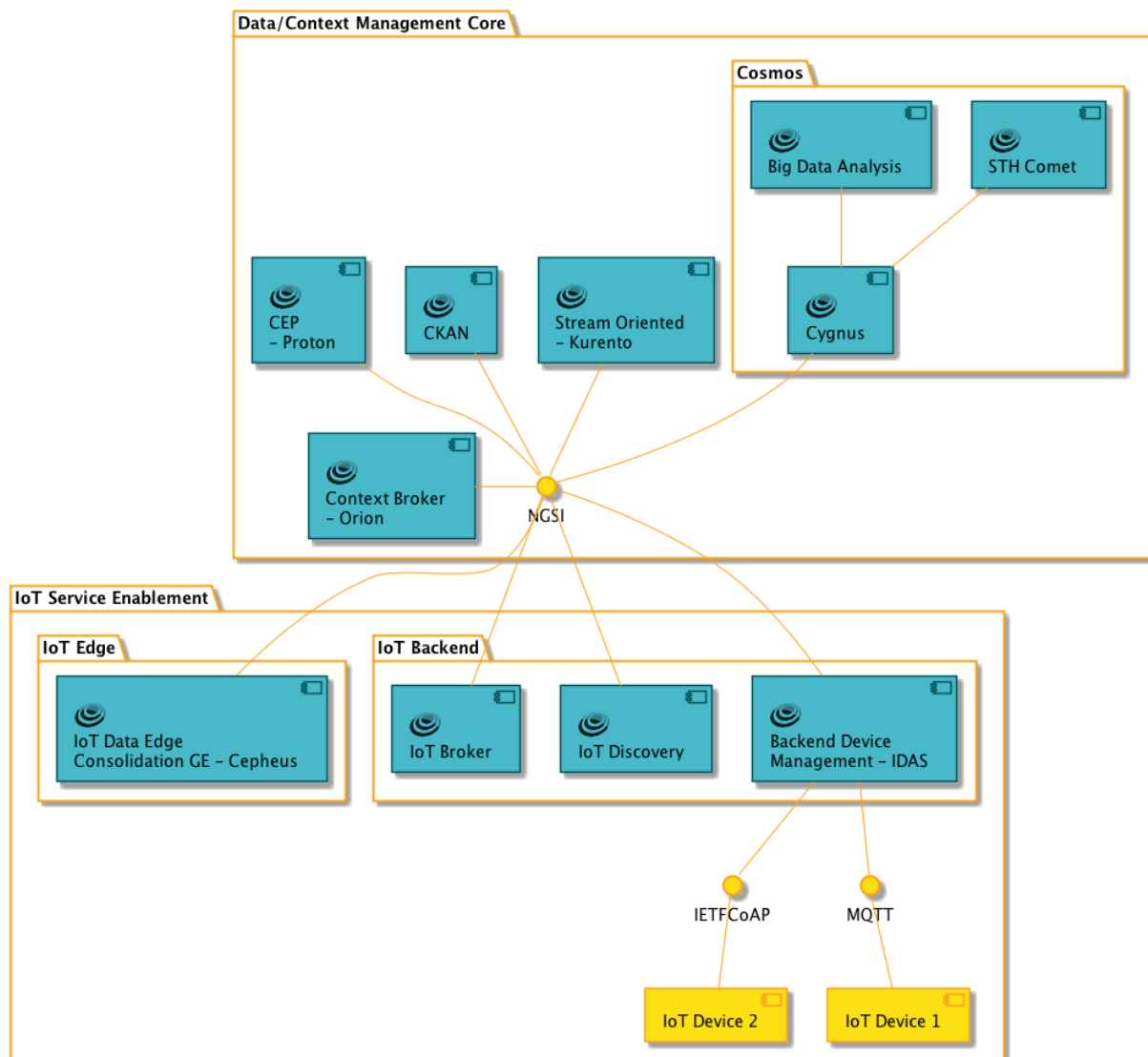


Figure 1: FIWARE Reference Architecture for IoT and Data intensive applications

¹ <http://schema.fiware.org>

Core components of the Data/Context Management² chapter include:

- ➔ *Context Broker* - Orion, the central element of the Data Context Management architecture, provides a publish/subscribe service for context data.
- ➔ *Big Data Analysis* - Cosmos, the solution for the analysis of NGSI data sets, is made of different tools including the Short Time Historical data storage (STH Comet) along with the capabilities of integrating different datastores (relational databases, big data filesystems, etc.) thanks to the adaptation capabilities provided by Cygnus.
- ➔ *Stream Oriented* - Kurento, an NGSI integrated multimedia processing server.
- ➔ CKAN - a repository for Open Data sets.
- ➔ *Complex Event Processing (CEP)* - Proton, an event processing tool that identifies patterns over NGSI data and generates response based on identified patterns.

The IoT Services Enablement³ chapter includes the following core components:

- ➔ *Backend Device Management* - IDAS, a set of IoT Agents allow IoT devices to be registered and transforms all collected data into NGSI compliant format.
- ➔ *IoT Data Edge Consolidation GE* - Cepheus, a solution for edge processing and aggregation of sensor data NGSI compliant.
- ➔ *IoT Broker*, an NGSI middleware that support the aggregation of data from multiple sensors.
- ➔ *IoT Discovery*, an NGSI middleware that provides support towards the discovery of context producers by context consumers.

It is worth mentioning that some of these services are meant to run only as Global Instance of FIWARE Lab. This is the case for example of *Big Data Analysis Cosmos*.

SmartSDK, through the development of novel components (c.f. D3.2) extended the above reference architecture to cover additional generic requirements coming from the applications developed in SmartSDK, introducing also Open Hardware-based solutions in the sensing layer.

Figure 2 presents the Northbound-Southbound reference architecture extended to include SmartSDK novel enablers. Also, depending on their needs, developers can choose to adopt a given subset of components in their application along with the relevant data models.

² <https://catalogue.fiware.org/chapter/datacontext-management>

³ <https://catalogue.fiware.org/chapter/internet-things-services-enablement>

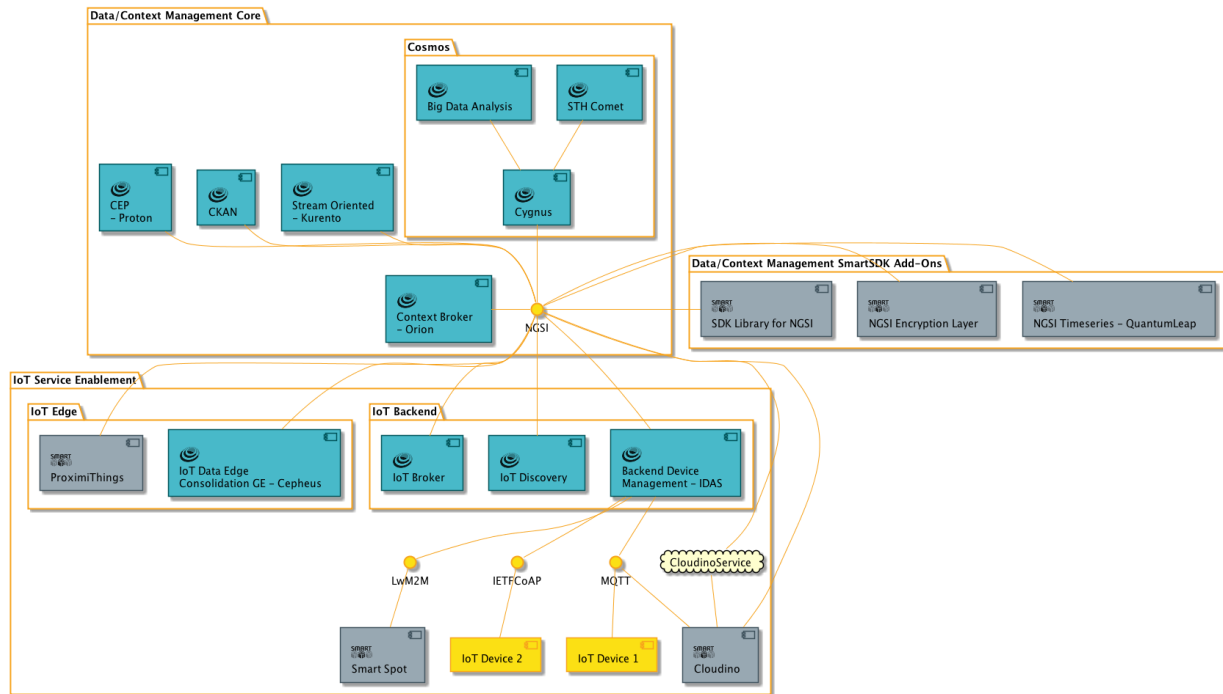


Figure 2: SmartSDK extended FIWARE Reference Architecture for IoT and Data intensive applications.

Novel components introduced in the Data/Context Management chapter by SmartSDK include:

- ➔ *SDK Library for NGSI* - an SDK for mobile devices that allows the developer to update and query context data using NGSI v2.
- ➔ *NGSI encryption layer* - a solution that encrypts NGSI data on the client's side to ensure privacy and protection of data stored in any context broker implementation based on the NGSIv2 protocol.
- ➔ *NGSI timeseries* - a solution to support natively timeseries processing supporting NGSIv2 protocol (meant to replace STH Comet).

Novel components introduced in the IoT Services Enablement chapter by SmartSDK include:

- ➔ *Cloudino* - an Open Hardware IoT solution that natively integrates with FIWARE and allows to natively push sensor data into FIWARE.
- ➔ *Smart Spot* - a solution that provides a Physical Web point of interactions natively integrated to FIWARE standards.
- ➔ *ProximiThings* - a FIWARE-enabled framework for the incorporation of proxemic interaction capabilities in IoT systems.

To take the most of cloud and microservice technologies, the above reference architectures need to be combined with cloud architecture patterns supporting the deployment in production-like environments of services. Typical cloud architecture patterns include (cf. D3.1):

- ➔ Scalability
- ➔ High Availability
- ➔ Multi-site pattern

- ➔ Co-locate pattern
- ➔ Queue-Centric workflow pattern

In D3.1 “*SmartSDK Reference Models and Recipes*” we discussed in detail how such patterns can be combined with FIWARE Reference Architecture. For instance, the Scalability Pattern aims to allow services to easily adapt to a sudden change in the demand of its resources. To apply such pattern to the FIWARE Reference Architecture for IoT and Data intensive applications, we need to understand how the single components of the architecture can scale based on their internals. Considering the Context Broker service, as an example of the process, this boils down to having multiple instances of the context broker service answering in “a coordinated way” to user's requests (cf. Figure 3). This can be attained in the case of web APIs (as the context broker is) using a load balancer (cf. D3.1 for more details). By all means, the picture is actually more complex, given that the Context Broker is not a stateless service, and so also data consistency among its backends need to be considered as discussed in D3.1.

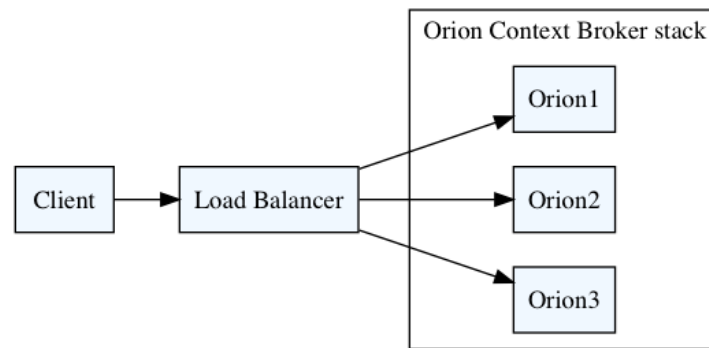


Figure 3: SmartSDK extended FIWARE Reference Architecture for IoT and Data intensive applications.

The deliverable will instead focus on deriving subsets (and related extension) of the SmartSDK extended FIWARE Reference Architectures for its adoption in specific smart scenarios, e.g. collecting environmental data, computing transport routing and so on.

3 SMART CITY REFERENCE ARCHITECTURE

3.1 Smart City Scenario Description

The application developed in the Smart City domain (named “*Green Route*”) aims at providing support to citizens’ mobility in high polluted cities like Mexico City. With that in mind, the goal is to both improve the citizen’s quality of life and foster environment-friendly behaviours. The end-user perspective is briefly described below.

Green Route’s intention is to help the final user determining the best route to follow in order to reach a specific destination considering the user’s: profile (such as health conditions), personal preferences (such as transportation type, etc.) Green Route is able to trace the ideal route for the user avoiding routes with: high levels of pollution, traffic jams or pollen saturated areas, etc. With the latter, the suggested route not only complies with the user’s preferences but prevents possible health crisis or asthma attacks due to the elevated pollen concentration levels at certain areas and, hence suggesting custom made routes for people with respiratory diseases.

3.2 Smart City Scenario Architecture in SmartSDK

The current Smart City architecture consists of a set of modules that allows the final user to be able to find the best route to follow in order to reach a specific destination, based on distinctive aspects that describe the user such as its profile (health conditions), preferences, transport type, etc. *Figure 4* presents the complete architecture for the Smart City Scenario. This architecture includes some of the components developed as part of the SmartSDK (cf. D3.2), such as: *Smart Spot*, *Cloudino*, and *SDK library for NGSI*.

In order to develop an architecture that allows the creation of an application for both pollution and traffic monitoring, we started by obtaining air quality data from IoT fixed and mobile devices. For instance, Smart Spot and Cloudino are able to record temperature, humidity, location data and are also able to detect various gas types (NO₂, SO₂, CO, O₃). First, captured data is sent to the *Backend Device Management* - IDAS through LWM2M and MQTT protocols. Then, the data is sent to the *Orion Context Broker* in NGSI format. Furthermore, the Smart City scenario uses the concept of human-as-a-sensor in which a novel mobile alert application is being developed. This mobile application will allow users generate alerts concerning: accidents, traffic jam, pollution, weather conditions, pollen and trigger an asthma attack alert. Additionally, data captured from users is send to the Context Broker. Furthermore, data will be stored using the Cosmos GE in a mongo database. A routing engine was developed to work on data collected from public transportation open data sites, with such information the engine is capable calculating and finding the best route between two locations using public transportation. In order to accomplish that, route engine allows the user to specify several parameters related to its route; departure and arrival times, maximum walking distance and others.

In order to present the collected data to users in a friendly and novel way, Front and Back ends were constructed. Interfaces and services work together to present the user relevant information such as alerts generated by other users and IoT devices. In the Front-end, the user registers data concerning its health

condition, interest groups subscription, its own alerts generated, etc. The Back-end is connected to the *Identity Manager* - Keyrock to authenticate user and services.

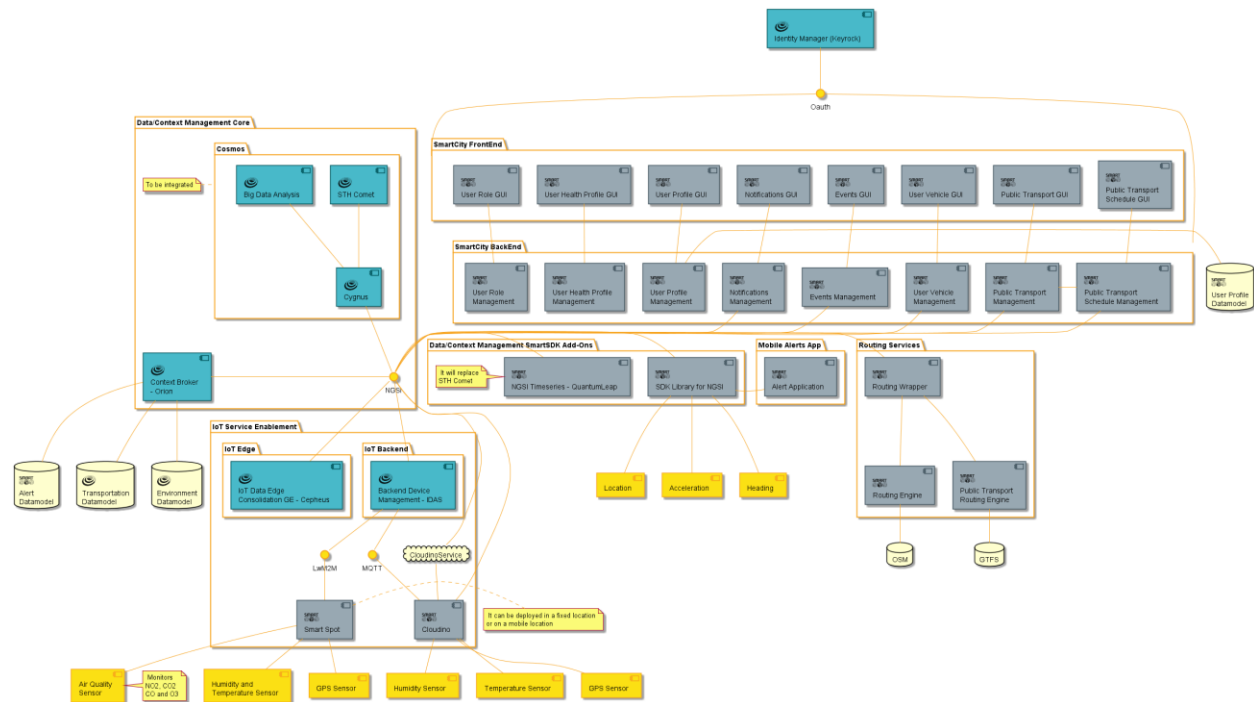


Figure 4: Complete architecture of Smart City Scenario

3.3 Smart City Generalized Architecture Patterns

Based on the architecture presented in the previous section and considering the domain requirements along with the work presented in D3.1 “*SmartSDK Reference Models and Recipes*”, the following reference architectures are formulated based on specific needs to exemplify additional viable scenarios.

Environment monitoring Architecture pattern

In order to be able to monitor the pollutant levels in a city, an environment monitoring pattern has been designed. The Environment Monitoring Architecture pattern distinguishes a collection of components that can be used to develop an application that presents users a selection of alerts based on collected data from both IoT devices and users. Such application is a mobile alerts app, that notifies users about different events such as: traffic jams, pollution levels, weather conditions, pollen and accidents.

The aforementioned IoT devices are a collection of sensors that measure: Temperature, Humidity, Gas Detection Sensor (NO₂, SO₂, CO, O₃), and a GPS sensor. Collected data is sent using MQTT and LWM2M protocols to an IoT Back-end device management GE. This IoT GE sends data to the Orion Context Broker on NGSI format. In addition, when a user reports an event it generates an alert that’s sent using a mobile alert application. Alongside with the report, data is sent to the Context Broker. There, an application that is subscribed to Orion Context Broker and when presented with a new alert it notifies a collection of users based on their preferences and a collection of predefined rules. Alerts

are shown in the application's Front-end (such application is called Green Route in the Smart City scenario). Figure 5 depicts the aforementioned Architecture Pattern.

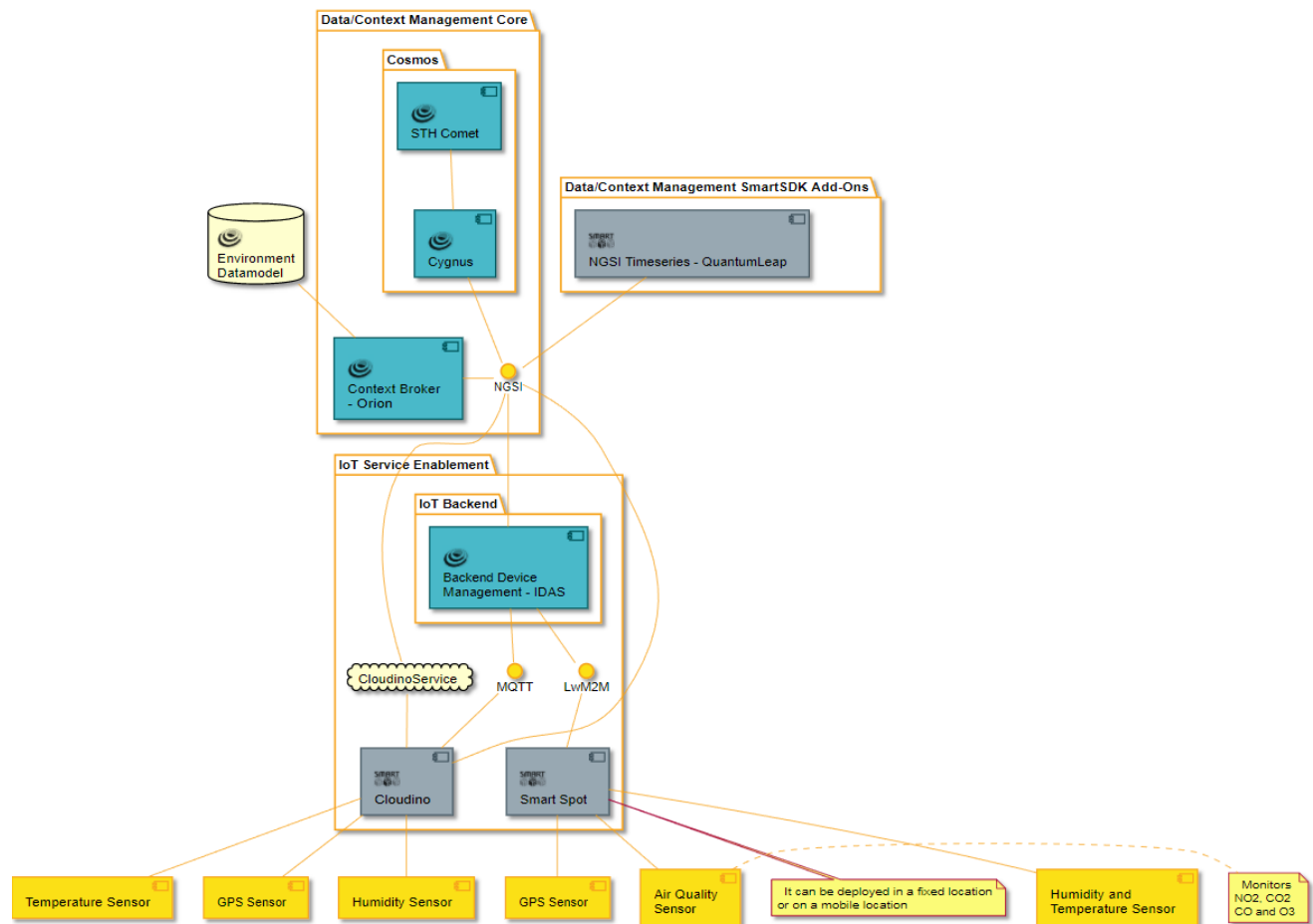


Figure 5: Environment monitoring pattern

Transport Routing Architecture pattern

With the following Architecture Pattern, users are able to trace the best route between two locations using the public transportation system of a city. The Transport Routing Architecture pattern outlines the usage of IoT devices to retrieve the position of any given public transport in real time, this information will be presented to users in a mobile application.

The aforementioned IoT devices integrate a series of sensors such as a: GPS, accelerometer, and gyroscope. Such equipment can be located on an assortment of public transport modes (buses, subway, rail, etc.). The goal is to monitor key aspects such as: its schedule, stops and location of each transport. The aggregated information is sent to a Back-end Device GE and to the Cosmos GE to be stored. Then, a route service enabler calculates the best route, using GTFS data and a routing engine. The best route is sent to the Back-end Smart City Application with to be presented to the users. Figure 6 depicts the Transport Routing Architecture pattern for Smart City scenarios.

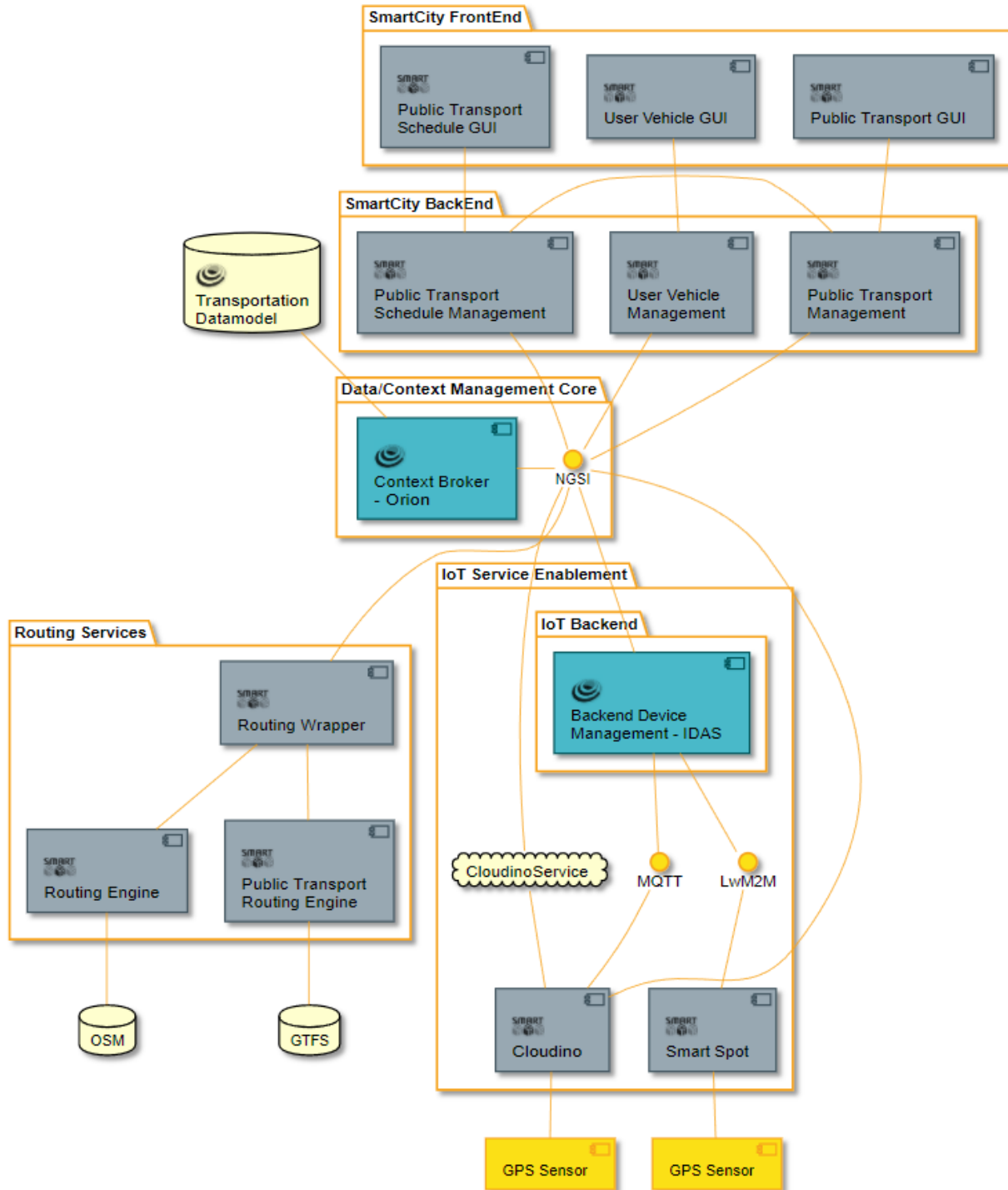


Figure 6: Transport Routing Architecture pattern

Mobile Alerts Architecture pattern

The revolutionary concept of human-as-a-sensor is a key element of the SmartSDK project. The main idea is that users send an assortment of alerts such as accidents, traffic jams, pollution, etc. In order to be

able to carry out such task the following discussion presents a Mobile Alerts Architecture pattern. The pattern comprises of a Library for NGSI that dedicates to collect location, acceleration and heading data. Such data are sent to Orion Context Broker using the NGSI format. From the gathered data a set of notifications are sent to user. Using the same application, a user is able to submit a new alert based on an observed event that needs to be reported. Figure 7 presents the main components implemented on the aforementioned Architecture pattern.

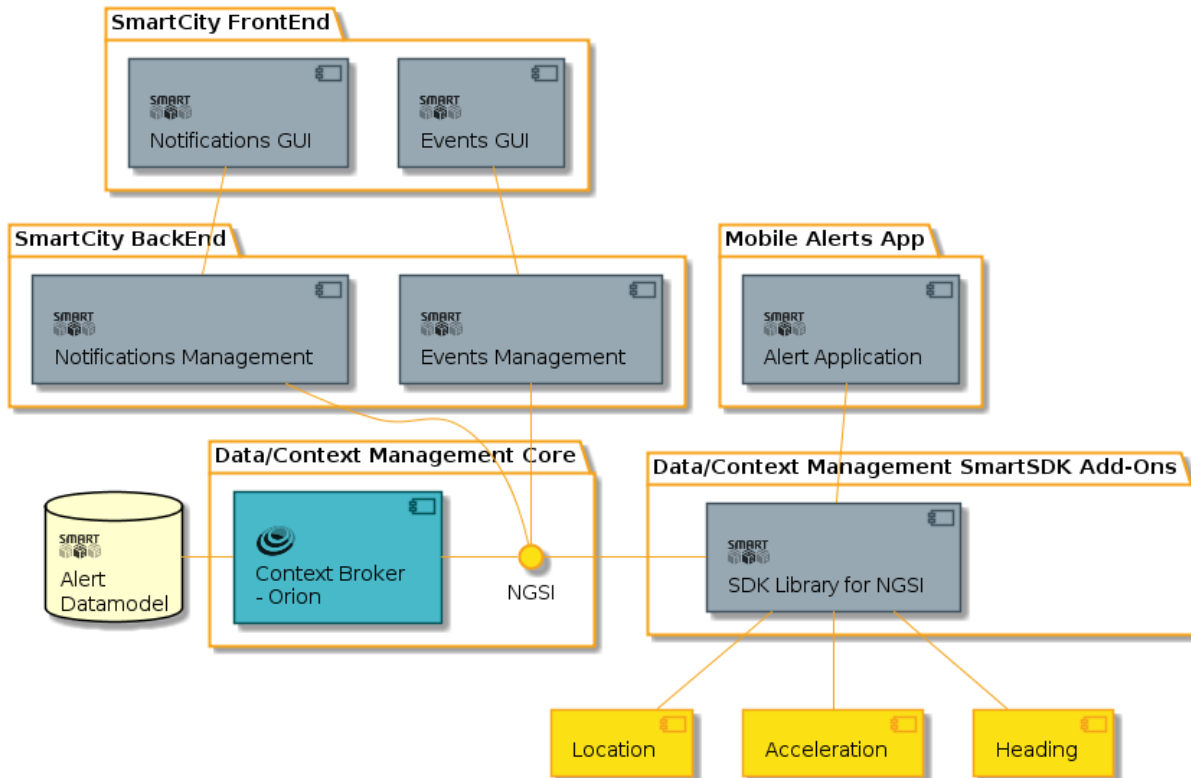


Figure 7: Mobile Alerts Architecture pattern

User Management Architecture pattern

To guarantee the privacy of the user's data and the safety of the application, the following User Management Architecture pattern is presented. The main goal of this Architecture pattern is securing the authentication process of users and applications. To be able to accomplish this, the *Identity Manager* - Keyrock is invoked. In it, domains, roles and permissions are assigned for each type of user. Regarding this, a user only can access from the Front-end applications that are stored in the Back-end and are declared within the Identity Manager, hence, an application can only be used by users that have permissions. The objective is to ensure the security of the applications by avoiding the modification of its content by users without the proper permissions.

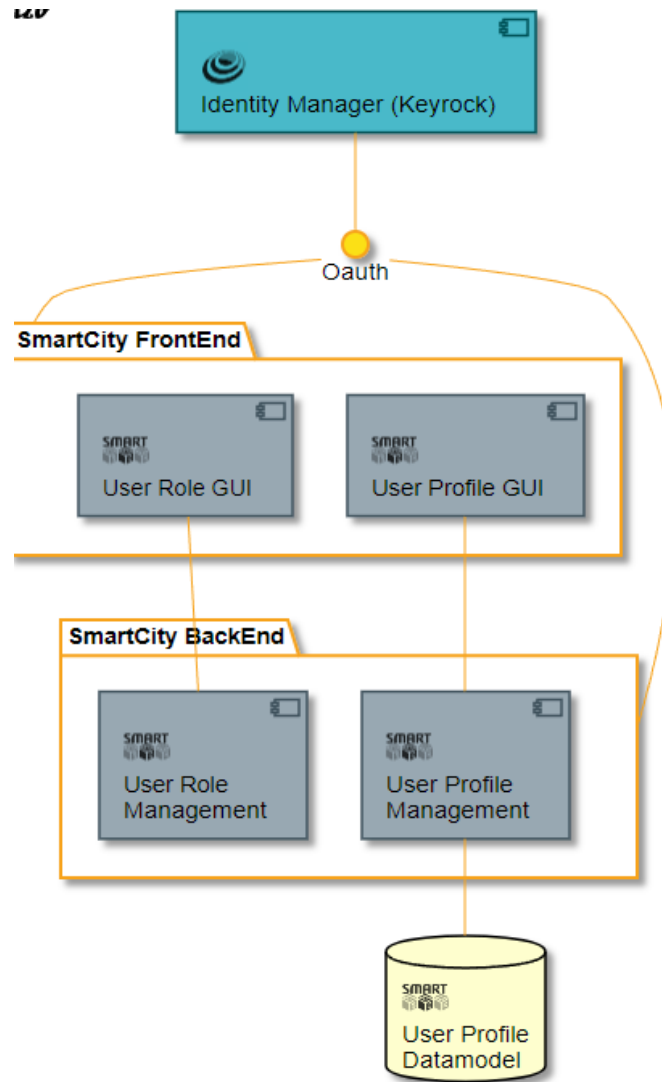


Figure 8: User Management Architecture pattern

3.4 Roadmap

The previously described architecture is presented as the current solution developed by the project partners that tries to unravel the scenario's specific needs. It is worth noting that anyone reading the following document might implement a different solution, but the key point is to serve as a hint to consider while solving issues similar to the ones described in this scenario. Likewise, the recipes that are being currently created might suffer slight changes due to the new deployment strategies being employed along with their implementations. Currently, after almost two years of work effort, the following components were developed attending the principles of the Smart City scenario:

➔ A new data model has been developed to contribute to the FIWARE Community. The *AeroAllergenObserved model* facilitates the user to share information about aero allergens at a given location.

- The *AeroAllergenObserved* model represents aero allergens observed at a

given location and related overall allergen risk.

- ➔ A crawler was developed to retrieve real-time traffic information by using the Here API, and send it to the OCB.
- ➔ A crawler was developed to retrieve weather information from the Weather Channel and send it to the OCB.
- ➔ A crawler was developed to retrieve pollen information from the Mexican Network of Aerobiology and send it to the OCB.
- ➔ An application to visualize time series information by using Grafana, CrateDB, OCB and QuantumLeap was implemented.
- ➔ A version of Green Route has been dockerized.
- ➔ The pollen, real-time traffic information and weather data are visualized in the Green Route map.
- ➔ Guides to connect IoT devices by using Cloudino and SmartSpot have been developed.
- ➔ The alert catalogue was updated.
- ➔ Endpoints to save, update, and retrieve user's routes, have been developed in the backend of the Green Route application.

The next steps of the Smart City scenario consist on the creation of the following elements and components:

- ➔ A collection of recipes based on the open platform Docker.
- ➔ Cloudino will be implemented as a Generic Enabler, so it could be downloaded in FIWARE Lab.
- ➔ Guidelines for using data models, such as Smart Spot, Smart POI and device will be developed.
- ➔ A trial will be deployed with the INFOTEC community.
- ➔ The terms and conditions specified by the ethics committee will be published in the Green Application.
- ➔ A component that is able to record information of the public transportation of Mexico City.
- ➔ Integrate the real time location of the public transport vehicles in the maps.
- ➔ Integrate traffic and pollution data while calculating an optimal route.
- ➔ Perform a single LWM2M connection to multiple entities of the NGSI data model.

4 SMART HEALTH REFERENCE ARCHITECTURE

4.1 Smart Health Scenario Description

The main purpose of the healthcare application is to expedite the harmonization and sharing of mobile sensing datasets for the healthcare domain. It focuses on mobile devices that collect data from sensors used on physical tests by following clinical protocols to assess the risk of falls.

The generated applications have been designed for research purposes, thus, parameters of interest (associated to the risk of falling) are analyzed *a-posteriori* and raw sensor data is kept allowing different stakeholders, such as patients and physicians to better understand how patients' activities and behaviours influence a healthier lifestyle.

As the project's proof of concept, we describe three different scenarios:

➔ Estimation of the risk of a fall

This application aims to collect sensor data from mobile devices (e.g., smartphones and smartwatches) while a participant performs a specific physical test under supervision and directions of a trained assistant. Controlled tests will focus on three physical performance measurements: Timed Up and Go (TUG), 4-Stage Balance Test, and 30-Second Chair Standard Test.

➔ Rehabilitation at home

This service has two different front-ends; one for the patient which collects data and one for the clinician, which summarizes the data from the patient. In particular, it measures mobility data to estimate differences in limb usage during the day for rehabilitation monitoring purposes. A pilot will be carried out to calibrate the system. First, with healthy phantoms and later with at least 1 patient with motor impairment undergoing rehabilitation therapy. Validation will be carried out using standard motor function assessment scales.

➔ Human-robot assistance

This application aims to collect sensor data from wearable devices (e.g., smart watches) whilst performing robot meditation assistantship; to decide how the robot address it. The development uses the wearable devices to monitor heart rate variability in order to trigger proper behaviour under the human-robot interaction to adapt the behavior of the robot.

4.2 Smart Health Scenario Architecture in SmartSDK

Aforementioned scenarios are being developed under a single architecture, which encloses some of the most common services required in terms of a data sensing application.

Both applications are to be used as tools to collect mobile devices sensor-data (e.g., accelerometer, orientation) while patients perform a set of physical activities. The generated applications are meant to be installed in a smartphone/smartwatch that will be worn by the participant.

In this context, data have been collected on each device and will eventually be sent to FIWARE Cloud by calling respective RESTful verbs. Data will connect to Orion Context Broker, which is wired to QuantumLeap in order to persist relevant data. Once data are available in QuantumLeap, it is analysed by a third-party component (build upon each app needs). Data are then retrieved to be presented on respective dashboard modules.

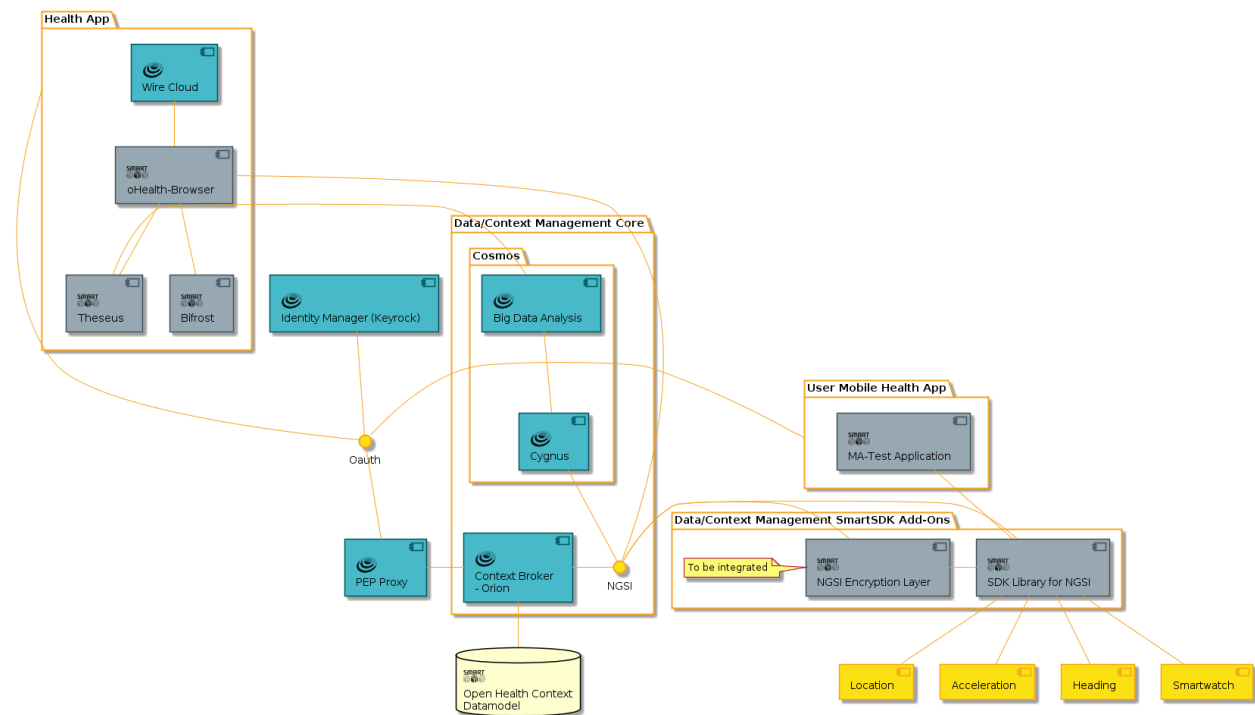


Figure 9: SmartHealth Scenario Architecture

The Scenario's main contributions focus on two components:

1. *Theseus*. A python-based implementation of open-source software to facilitate the analysis of data. It doesn't try to compete with the available Data Analytic GE, but to complement it for quick prototyping purposes by using public libraries, such as SciPy.
2. *oHealth-Dashboard*. A software developed to handle three different modules: Participants, Physical test, and Parameters of Interest. Altogether, functions are built into a single component which can be extended in order to cover a wider range of services.

4.3 Smart Health Generalized Architecture Patterns

Smart Health scenarios rely on a general architecture (Figure 9), that consists of three stages: (1) a software implementation for mobile devices (e.g., smartphones and smartwatches), (2) a set of schemas to wrap health data accordingly into NGSI specifications, and (3) a visualization mechanism to consult physical test results.

As illustrated in Figure 10, persistent sensor data is uploaded to the FIWARE cloud under NGSI standards through the Orion Context Broker GE and observing a set of open health-context schemas (oHealth Contexts). Then (Figure 11), by using the Cygnus connector and the Keyrock authentication mechanism sensor data are stored for further analysis under the Cosmos GE shelter. In this context, data are analyzed by using both Cosmos GE and Theseus algorithms to finally display processed reports in a graphical interface.

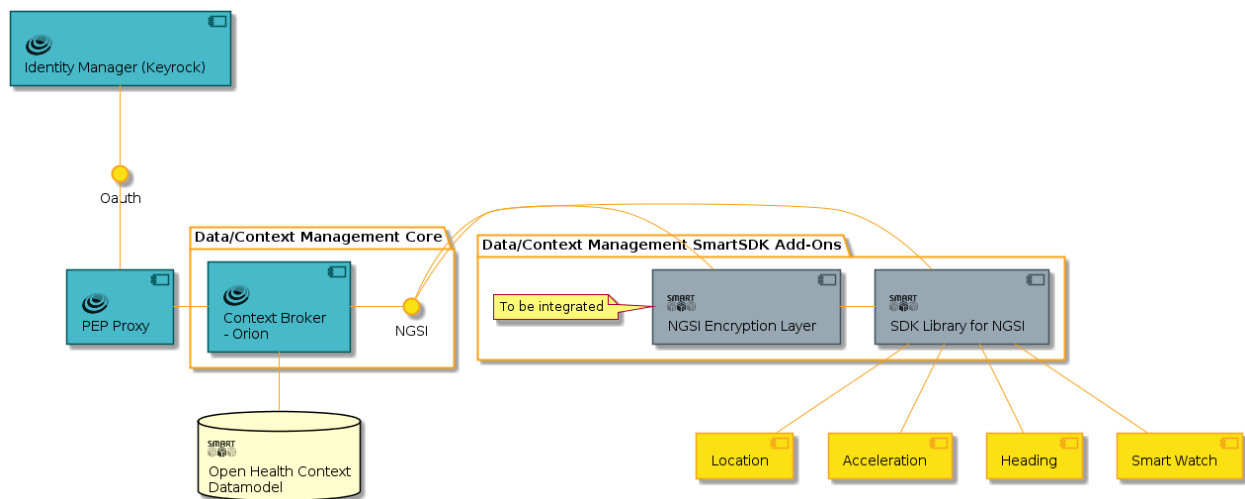


Figure 10: Sensible Data Collection Architecture pattern

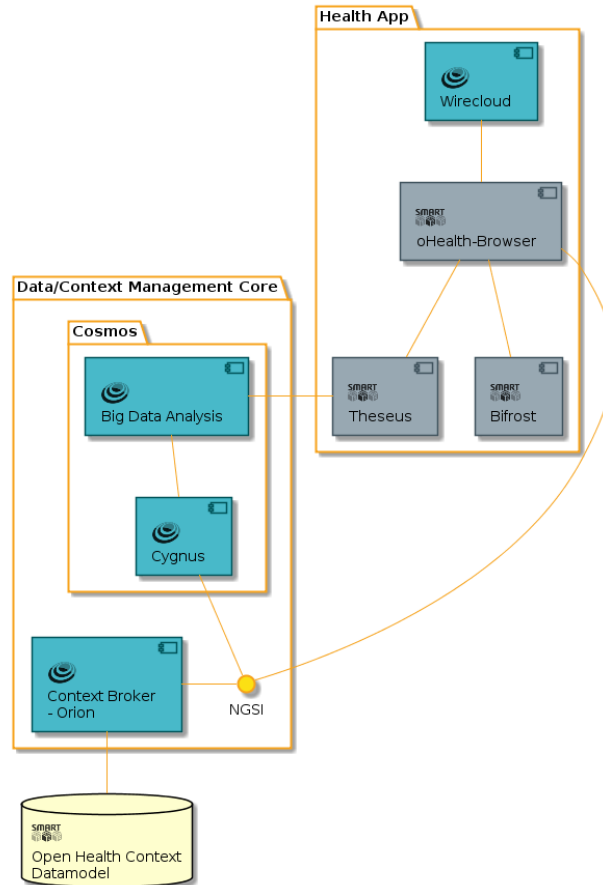


Figure 11: Data Analysis Architecture pattern

4.4 Roadmap

The components developed within the first two years of work are outlined below.

➔ Usage protocol

- For the rehabilitation scenario, define a protocol of device usage, for instance, specifying the smartwatch placement and orientation, identification of arm associated to the smartwatch, non-wear maximum times, etc.

➔ Upload Information

- A data model focused on representing sensor data gathered from smartphones, wearable devices, and health data inferred from the sensor data.
- A mobile phone application implemented on Android O.S. to retrieve sensor data. The collected data gathered during the physical tests is stored in the mobile device and then transferred to the cloud. It becomes persistent by storing it using the Orion Context Broker.
- Dashboard monitor. A front-end dashboard to display retrieved data.
- Connection to QuantumLeap so data can be persistent.

➔ Data analysis

- *Walking speed.* An initial approach for detecting events using accelerometer data recorded while performing a controlled physical test (e.g., TUG or Strength).
- *Overall Stability Index (SI).* A first version of a mechanism to calculate the SI which represents the variance of the platform displacement in degrees, from level, in all motions during a test.
- *Subject study.* A first subject study is conducted to test both: FIWARE's integrated component and application development.
- *Estimate of patient total activity.* Area under the accelerometer curve across all sensed devices (smartwatch x2, and smartphone).
- *Estimate of patient activity per arm.* Area under the accelerometer curve per device
- *Differences in bilateral arm usage for patient.* Differences between estimate of activity per arm.

➔ Access Control

- Enrich the dashboard monitor of the rehabilitation patient with sign in and sign up capabilities.
- Enrich the dashboard monitor of the rehabilitation clinician with sign in and sign up capabilities.
- The health data sensing management allows access to data only to authorized users. In addition, private data is stored independently in the public Cosmos GE.

➔ Data integration

- Data analysis mechanisms are integrated on the application architecture; thus, results can be retrieved from respective dashboard.
- Bring together data from different devices (2x smartwatches -one on each arm- and 1 smartphone) into a single packet of data to be uploaded to FIWARE's cloud.

➔ Wearable devices integration

- Creation of the project's proof of concept in order to practically test the flexibility of the data-model, over wearable sensing devices such as smartwatch.
 - Design and development of an application for rehabilitation purposes to monitor motor function on a coarse scale during a regular week amid daytime (outside the rehabilitation ward).
 - Design and development of an application as tool to recollect motor data and measure the risk of fall.
 - Design and development of an application to personalize human-robot interaction whilst performing assisted powermind activities.
- Test the proposed data-model to be used on smart watches in the rehabilitation scenario.
- Test the proposed data-model to be used on wearable devices to monitor heart rate variability.
- Test of retrieval data to address human-robot interaction to adapt the behavior of the robot.

Overall, future activities will cover:

- ➔ Refinement of the architecture pattern.
- ➔ Production of recipes based on Docker compose.

5 SMART SECURITY REFERENCE ARCHITECTURE

5.1 Smart Security Scenario Description

The Smart Security scenario focuses on the development of an application capable of providing support to a security guard to prevent risky situations. In order to prevent such situations, the risk analysis is conducted in areas with video surveillance cameras, thus improving the quality of life of the people.

Within the scope of this scenario, the application's core is the video stream analysis from a collection of cameras. The visual information contained in the video stream is used to detect different events such as: people and vehicle detection, crowd analysis, etc. Security risks such as driving in unauthorized speed, in wrong way or sudden stops of a vehicle, are identified by the combined use of video cameras and mobile sensors of a smartphone in outdoor scenarios like parking lots.

The security application has been designed for research purposes, thus, resulting data from the developed algorithms will be kept for further analysis protecting at all time the privacy of recorded people and events.

5.2 Smart Security Scenario Architecture in SmartSDK

The Smart Security architecture is being developed to detect security risk events based on the video stream acquisition from video cameras connected to a network, and sensors integrated in a smartphone. The data is then analyzed to detect, label, store and highlight security-relevant events automatically.

To capture the incoming video streaming from IP cameras we are using Kurento GE through the Kurento Media Server (KMS). The KMS is based on Media Elements (ME) and Media Pipelines. The former consists of modules that perform a specific action on a media stream receiving or sending media from other elements, the latter is a chain of media elements. For the security application we have developed a set of filters (ME) capable of subtracting objects (people and vehicles) in motion within a scene, classify such objects and track them. All labeled data generated from the stream processing are sent to the Orion Context Broker through its connection with Kurento. Furthermore, the data from the detected security events are stored into Quantumleap.

The contributions of the aforementioned architecture (Figure 12) consist of the design and implementation of new Kurento filters to extract relevant information within the scene, both custom-made graphical user interfaces (web and mobile) with various functionalities.

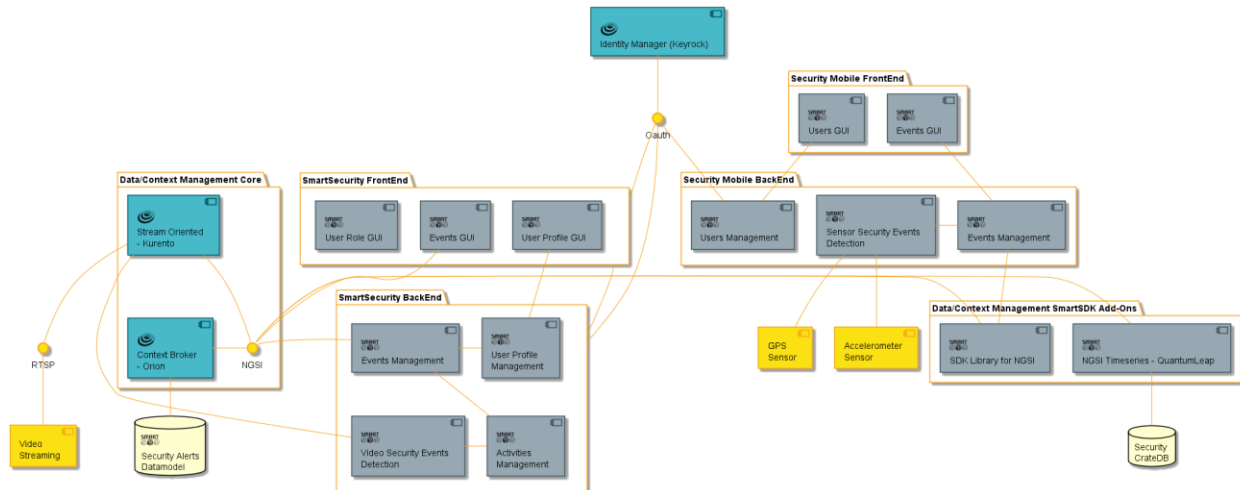


Figure 12: Smart Security Scenario Architecture

5.3 Smart Security Generalized Architecture Patterns

Based on the architecture presented in the previous section, we have formulated the following architectures according to the specific needs of the security scenario.

Video Event Detection Architecture pattern

The Smart Security application aims to detect security risks through video stream analysis, thus, a Video Event Detection Architecture pattern was detailed. This Architecture pattern (Figure 13) describes the main FIWARE components needed to build the application. In general, this pattern allows us to capture a video stream from a set of cameras, analyze the information through a set of novel Kurento filters that are implemented using OpenCV and to generate notifications/alerts to the user about events happening in the monitored area.

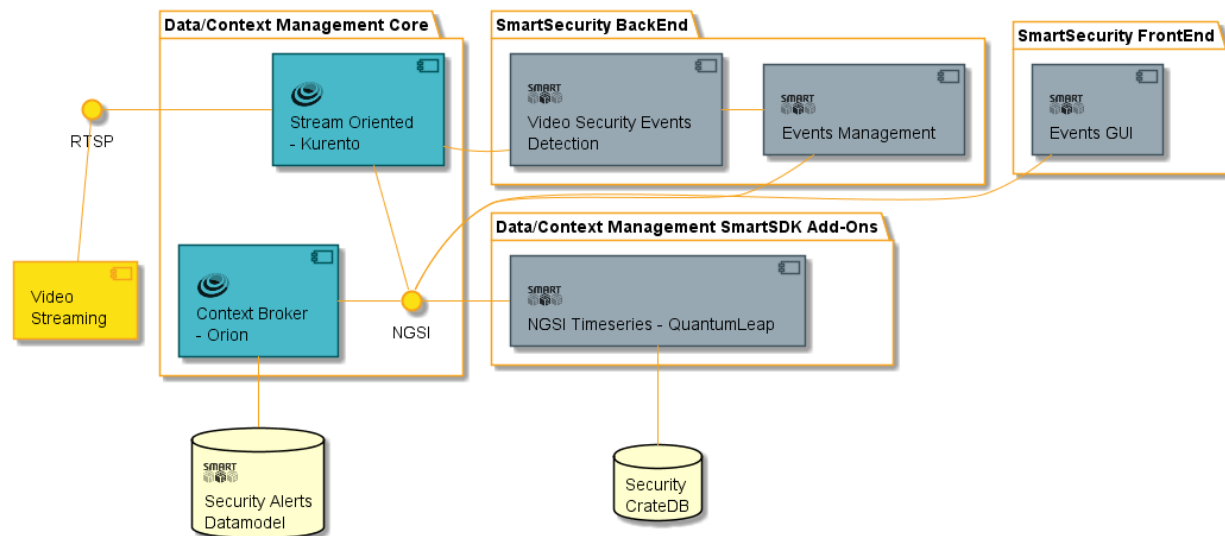


Figure 13: Video Event Detection Architecture pattern

Event Detection Through Smartphone Architecture pattern

In order to improve the detection of risk situations and also be able to detect them in areas that cannot be monitored by a camera, an Event Detection Through Smartphone Architecture pattern was generated. With this Architecture pattern () we are able to collect data from the smartphone. The collected data contains information regarding: position coordinates of the user and speed acceleration of the vehicle, this information is provided by the sensors (GPS and accelerometer) integrated on the smartphone and is analyzed inside of it to detect the occurrence of a possible event, and then the data is sent to the Orion Context Broker and NGSI Timeseries using the NGSI format.

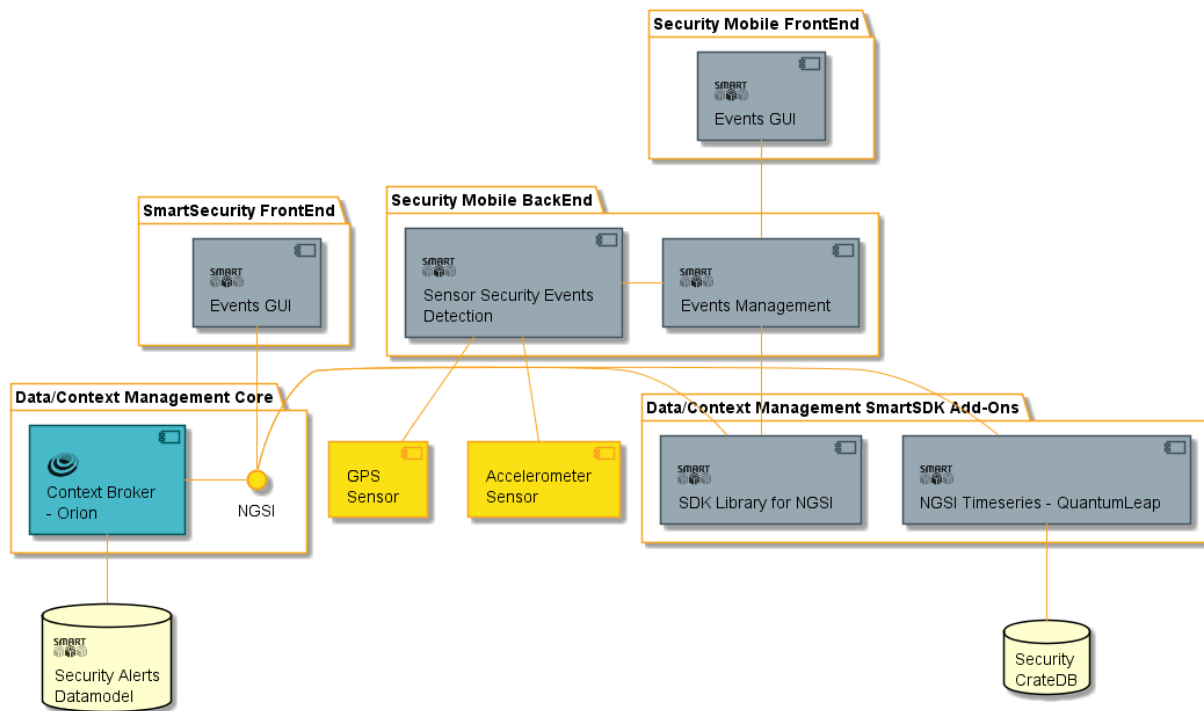


Figure 14: Event Detection Through Smartphone Architecture pattern

Alerts Architecture pattern

To help the user to quickly respond at any given risky situation, the Alerts Architecture patterns has been designed. This pattern () is being developed using the NGSI format to gather information from the video/smartphone event detection. The data are then sent to the Orion Context Broker and then to the front-end application where the user is notified (by the Web and mobile GUI).

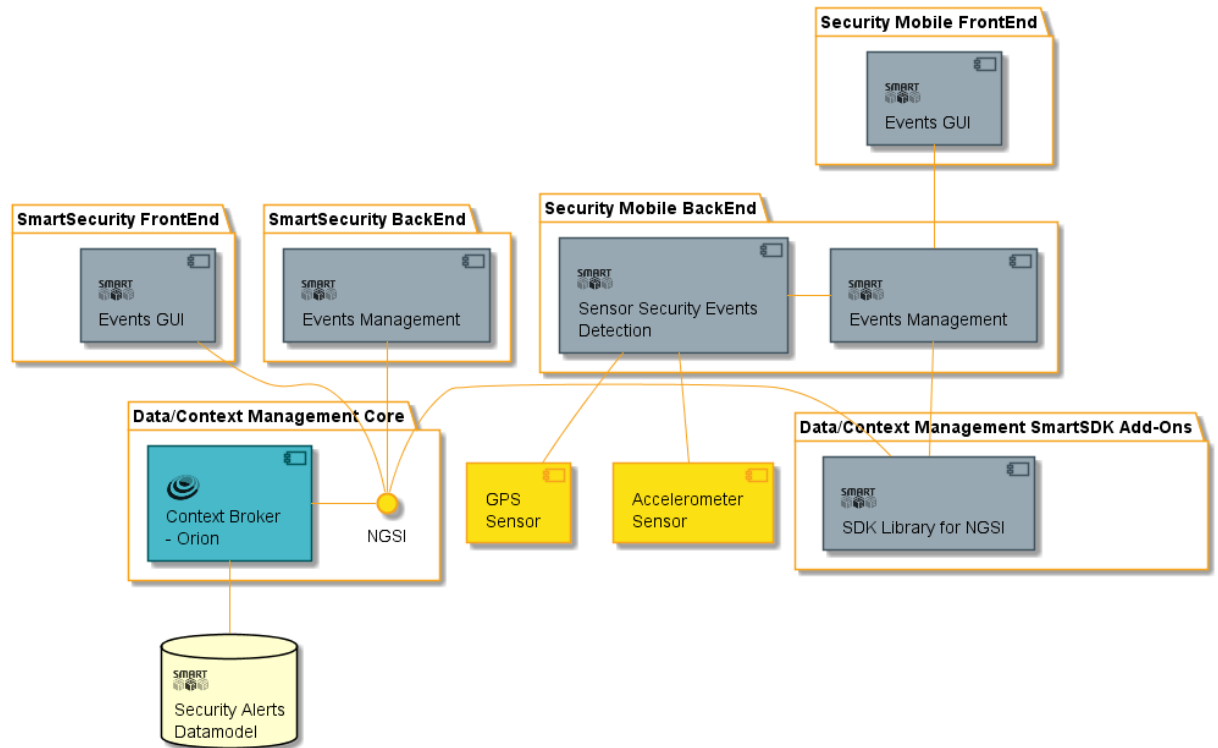


Figure 15: Alerts Architecture Pattern

5.4 Roadmap

The set of components to support the smart security scenario that have been developed are:

➔ Data management

- A database has been implemented to store both, video and smartphone alerts.
- Storing and management of user's information.
- An authentication module was developed using Keyrock (specifically Keystone).
- Design and implementation of a module to make online queries about whether a registered smartphone is in the campus or not.
- Design of a software module to make historical queries about whether a registered smartphone was on campus or not in a specific time period.
- Design and implementation of notifications to alert the user when a vehicle is parked incorrectly.
- Implementation of a software module, based on the data from a smartphone, capable of identifying people driving at an unauthorized speed on campus.
- Implementation of notifications to alert the user when a vehicle goes in the wrong direction.
- Implementation of a component to determine when a registered smartphone is inside a specific Campus.

- The web GUI was extended to support:
 - Login and Registration of Security and Administrator users.
 - Manage users.
 - Geo-visualization of the cameras.
 - Query and geovisualization of mobile users that are or were in the campus.
 - Queries of alerts generated by the smartphone
 - Add a new surveillance area.
- An android mobile app was designed and implemented. The app supports:
 - Login and Registration of: Mobile, Security and Administrator users.
 - Visualization on map of the area where the user is.
 - Generation of manual alerts by mobile or security users.
 - Visualization of the alerts list generated during the day.
 - Geo-visualization on map of the location of a specific alert of the list.
 - Visualization of the list of last ten alerts.
 - Geo-visualization on map of the last ten alerts.
 - Driving View for users that drives a vehicle.

The Smartphone and Web approaches integrated and tested were:

➔ Pattern recognition

- Basic algorithms have been integrated. Performance on cloud was evaluated.
- Development of a software module (Kurento filter) for vehicle detection on a video stream.
- A face recognition algorithm was implemented. The algorithm is capable of verify the identity of a person in a video in indoor environments.
- An algorithm (Kurento filter) capable of detecting people in a video stream in outdoor environments was implemented and integrated to the system.
- An algorithm (Kurento filter) capable of detecting if an object is abandoned in a video stream in indoor environments was implemented.
- An algorithm (Kurento filter) capable of detecting people walking, running or fighting in a video stream in indoor environments was implemented.
- An algorithm (mobile app) capable of detect unauthorized speed events when the user is driving a vehicle.
- An algorithm (mobile app) capable of detect sudden stops events when the user is driving a vehicle.
- Development of a software module based on the data from a smartphone capable of detecting if vehicles are going on the wrong direction within the parking lot.

6 CONCLUSIONS

This document presents the specific Architecture used to develop smart applications in the Smart City, Smart Healthcare, and Smart Security domains. Such architectures consist of components both Generic

D2.5: Reference architectures for data-intensive and IoT-based Smart City, Smart Healthcare and Smart Security applications

and Specific Enablers, data models along with generalized architecture patterns that provide an insight of additional applications derived from the main architectures of each domain. Providing the tools needed to expand and create new applications within each of the three domains.

Domain Specific and generalized architecture patterns are the main contributions of the document the represent FIWARE's main benefits: flexibility and reusability of the components. By providing the building blocks to develop and enhance current solutions within a domain or even integrating components that might belong to a different domain but that might be the key element to solve a specific and fine grain problem pertaining another domain.

REFERENCES

- [1] General Transit Feed Specification. Retrieved July 01, 2017, from <http://gtfs.org/>
- [2] Office of Atmospheric Monitoring. Retrieved July 01, 2017, from <http://www.aire.cdmx.gob.mx/default.php>