e-Health IoT Reactive Services for Elderly Care at Home in Smart City built on an Emerging Fast Data Architecture

Luis Jurado Pérez and Joaquín Salvachúa Rodríguez
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid, UPM
Madrid, Spain

Abstract - *Existing e-health monitoring systems, but they lack reactive characteristics that allow their scalability and use in Smart Cities. In this article, the design and implementation of e-Health reactive services for the elderly at home is presented. The system design follows a methodology based on Rozanski and Woods' iterative architectural design process, the principles of the Reactive Manifesto, the Domain-Driven Design (DDD) and the characteristics of the Emerging Fast Data Architectures. A set of services was implemented: (i) basic services for the management of the system (ii) monitoring services and (iii) services for emergency management. Finally, the authors conclude that this type of systems is essential for the elderly and those users who work collaboratively in elderly care due to the need to manage the flow of data in real-time.*

Keywords: Internet of Things (IoT); Emerging Fast Data Architecture; Microservices; Reactive System; e-Health;

1 Introduction

The World Health Organization (WHO) anticipates a growth of the elderly population in the coming years [1] and the field of e-Health has become an essential field to manage this growth. On the other hand, elders' relatives are concerned about the life that the elderly could have in their homes. Research lines such as Internet of Things (IoT) [2] are of great interest to provide pervasive systems with breakthrough innovative technologies, many of these concerns could be reduced ensuring a certain well-being of the elderly while they are in the house, relegating the stay in a retirement home to the background. Although in the case of IoT some designs and implementations of systems have been provided in various areas of human development including the e-Health field, there are some other challenges and open issues that need to be addressed, such as standardization, IoT healthcare platforms, the app development process, scalability, continuous monitoring, data protection, etc.[3]. Nowadays, also, crowd-sensing has emerged as one of most important paradigms to develop large-scale sensing services [4], this type of solutions based on crowd-sensing are low cost compared to the Wireless Sensor and Actuator Networks (WSANs), and besides they demand incremental data storage to save the sensed data in Big Data repositories [5].

Moreover, the design of distributed applications in Smart City requires high computing power provided by highly scalable hardware and software resources. To face these challenges, the Reactive Manifesto presents a coherent approach to system architectures with the following characteristics [6]: Responsive, Resilient, Elastic and Message Oriented. Besides, distributed applications must provide interactive features for a wide variety of users and devices concurrently, provide batch and near real-time processing as well as Big Data analytics and must be rapidly designed and developed. It is important to note that streaming applications in e-Health Systems must handle critical scenarios such as in emergency situations and near real-time applications are necessary in obtaining medical reports especially in chronic diseases. Therefore, this article promotes the use of Emerging Fast Data Architectures, which allow ingesting and processing continuous [7].

This article proposes the design and implementation of e-Health reactive services of IoT for the elderly at home built on Emerging Fast Data Architecture. The system uses crowd-sensing data and it is a system with Big Data sub-systems that support near real-time data analytics in the context of Smart City. This article makes the following contribution:

- Propose a methodology for the design and implementation of reactive services built on an Emerging Fast Data Architecture.

The rest of the article is organized as follows: In Section 2 provides a brief overview about the related work. The methodology used for the design of the system is presented and discussed in Section 3. In section 4 the authors introduce the system architecture. The implementation of the prototype system is presented in Section 5. In Section 6 an evaluation of the services is carried out and finally, Section 7 concludes the article and proposes the future work.

Table 1: Related work

Reference Number	Monitoring (Sensors)	Emergency Situations	WSAN/Wireless/ 3G/4G/Bluetooth or Others	Near Real-time Communication	Scalable	Characteristic Relate to Big Data	Near Real-time Analytics	Services Implementation
[8]	Y	Y	IEEE 802.11 Wireless Wi-Fi	N	N	N	N	N
[9]	Y	Y	3G	N	N	N	Y	Web Services (JSP, Servlet, EJB, MDB, and JDBC)
[10]	Y	Y	BASWN/3G/3.5G/4G/H2	Y	Y	N	N	Web Services (Boto,Python)
[11]	Y	Y	Bluetooth, BSNs, 3G	N	N	N	N	Web service – PHP scripts.
[12]	Y	N	Bluetooth-6LoWPAN, IEEE 802.15.4, BAN, HL7	Y	Y	N	Y	RESTful web services
[13]	Y	Y	6LoWPAN, WSNs	Y	Y	N	N	Web Services (XML)
[14]	Y	N	ZigBee/6LoWPAN-based WSN, RFID, Bluetooth, WLAN,IEEE 802.15.4, IR	N	N	N	N	Open Services Gateway Initiative (OSGi)
[15]	Y	Y	Bluetooth, BLE, IEEE 802.15.4, 6LowPan, 3G, Wi-Fi	N	N	N	N	Restful Web Service
[16]	Y	N	Ethernet, wifi, IEEE 802.15	N	N	N	N	Web Server (PHP + Apache)
[17]	Y	Y	Bluetooth, 3G	Y	Y	Y	Y	N/A
[18]	Y	Y	N/A	Y	Y	Y	Y	N

2 Related work

Table 1 shows a comparison of some research works related to the monitoring of the health status of the elderly at home. First, there are several proposals for frameworks and architectures that show their functional components for modeling healthcare systems [8][9]. In [8], a framework is proposed for supporting elders at home, which permits detecting situations of danger and to send emergency signals to the caregivers for a timely attention. It is structured in three layers: Monitoring and Assistance Layer, Response Management Layer and Group Collaboration Layer. Other initiatives provide a set of devices based on some combination of sensor technologies and the ZigBee, Bluetooth, and Wi-Fi protocols that use the Smartphone as a primary element for data extraction (Data Sink) [10][11].

On the other hand, some investigations have studied the use of WSAN based on ZigBee or 6LoWPAN in combination with Wireless Body Area Network (WBAN) based on Bluetooth or Wi-Fi to monitor patients and then send the data to external systems (data storage) through a Local Gateway [12][13][14]. In [14], a generic platform has been built in the SPHERE project, which fuses sensor data in order to generate datasets that support the detection and management of various health conditions. The SPHERE project integrates various sensing modalities and the collected data is then sent to SPHERE's data hub through the Internet. This hub has a dynamic library of data analytics services available for registered end users.

There are also healthcare systems that rely on Cloud Computing for their operations. In general, Cloud infrastructure is used to deploy healthcare system applications and services and to provide data storage by leveraging the flexibility and scalability of Cloud Computing [9][10][15]; in some cases, for example, implementations are made with technologies like JSP, Servlet, EJB, MDB and JDBC [9], and in others, modular implementations with REST web services are usually done [15]. However, systems that do not use the Cloud Computing rely on independent servers with a server application developed with typical programming languages (e.g. PHP) [11][16], or use a RESTful web-application [12].

Additionally, there are researches that have extended their studies to the data analytics supported by Big Data technologies [17][18]. It is necessary to emphasize that the investigations on the systems of Big Data in these days are oriented to the data analytics in near real-time. In [17], a scalable architecture is proposed that permits monitoring of patient health data through the Smartphone. This data is sent to the Cloud where Machine Learning processes are used.

The best of the author knowledge, there are few e-Health Systems that exploit both the principles of the Reactive Manifesto and the use of open source technologies to implement services in distributed pervasive systems with reactive characteristics. The objective of this article is the design and implementation of e-Health reactive services of IoT for the elderly at home built on an Emerging Fast Data Architecture [7]. The system architecture considers the management of data flow in near real-time due to its implications not only in the criticality of the data that this type of system handles but also in the use of a Big Data sub-system, which performs near real-time data analytics. Moreover, the authors use crowd-sensing which is a very effective distributed solution for gathering data where the cost for dedicated infrastructure like WSANs is too expensive [4][19]. Thus, the system can be used in a distributed way among various users working collaboratively for elderly care.

3 Methodology

The design of the system follows a methodology based on the use of iterative architectural design by Rozanski and Woods [20], and it is taken in consideration of the identification and use of a set of fundamental architectural patterns. In addition, the design is based on the principles of the Reactive Manifesto to provide highly scalable and reliable applications [6]. Moreover, the

characteristics of the Fast Data Emerging Architectures complement the design of the system [7]. Finally, the Domain-Driven Design is used to divide the various domains of the system into microservices [21].

3.1 IEEE 1471

The fundamental elements of the system software architecture were elaborated based on the iterative process of architectural design of Rozanski and Woods that is based on the standard IEEE 1471. In this process, a series of architecture views are used to define the Architectural Description.

3.2 Architecture patterns

Popular architectural patterns were used since they help to specify the fundamental structure of an application:(1) *Layered Architecture Pattern* was employed to decompose the system tasks into interrelated subtasks[22][23];(2) *Message-Oriented Broker Pattern* was used as an intermediary of the distribution of the data mainly coming from the sensors in such a way that there is an uncoupling in the readings and writings of the data[22][23];(3) *Microservice Architecture Pattern* was implemented to provide the necessary components for the construction of System Applications since the microservices can be deployed, scaled and tested independently and each of one performs a specific task, which is easily understandable [24]; (4) the *Model-View-Controller Architecture Pattern* was applicable due to the existence of electronic devices that support graphical interfaces since they provide a fundamental structural organization for the interactions of software systems [22]; and (5) the *Cloud Computing paradigm* was presented as the solution to meet computational economics, web-scale data collection, system reliability, and scalable performance [25].

3.3 Reactive principles

Responsive, Resilient, Elastic and Message Driven.

3.4 Fast data systems

Nowadays, in order to provide the near real-time data analytics services system (including batch processing), more stream oriented features are added through so-called Fast Data Systems that ingest and process continuously.

3.5 Domain-Driven Design (DDD)

DDD is an interesting way to structure a microservice.

4 System architecture

4.1 Scope and context

A basic scenario has been established where a series of concerns have been considered. An elderly person lives in a house of predetermined dimensions. The spaces are known and can be expressed in a two-dimensional plan. The elder can move around the house with complete freedom, vital signs data and room temperature are collected through a series of wearable devices and sensors in the Smartphone. Remote monitoring of vital signs is carried out, in addition certain controls can be set to trigger alerts and activate some emergency protocol. Authorized users can access the system: consult patient data, consult the characteristics of the house, check the temperature status in the house, and monitor the vital signs of the elder in real-time. In summary, the services designed were: (1) *basic services* for the management of the system, (2) *monitoring services*, and (3) *services for emergency management* based on real time data analytics.

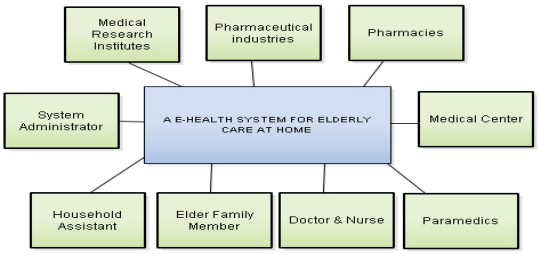

Fig. 1. Context view

4.2 Architecture description

4.2.1 Context view

The system context view describes the relationships, dependencies and interactions between the system and its environment (people, systems and external entities). Figure 1 shows the context view.

4.2.2 Functional view

Figure 2 shows functional components in a general way and Figure 3 shows functional components and their relationships. The functionality of the system has been divided into a set of components, which are detailed below:

4.2.2.1 The core system

This consists mainly of: (1) *Elder Manager* (it allows creating and maintaining the elder's personal data); (2) *User and User Groups Manager* (it permits creating profiles and security levels, create user-type

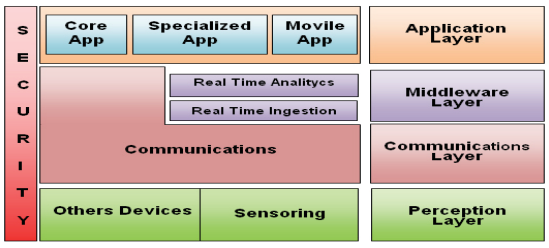

Fig. 2. Layered architecture pattern

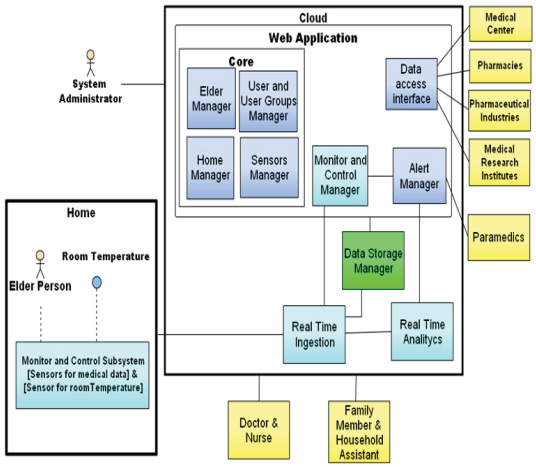

Fig. 3. Functional view

profiles and create users or groups of users of the system); (3) *Home Manager* (it allows creating data related to the elder's home like: location, number of rooms, types of rooms, free zones, dimensions, etc.); and (4) *Sensors Manager* (it permits recording the data about of features of the sensors).

4.2.2.2 Monitor and control manager

This consists mainly of: (1) the monitor and control subsystem of the sensors for the health data of the elder and the state of temperature in the house; (2) the Input/Output Transporter which is a transport element of the sensing data in near real-time (*Real-time Ingestion*); (3) the functional elements of the system that allow the parameterization and configuration of the Input/Output Transporter, for example: the creation of topics in distributed publish/subscribe systems; and (4) the elements for near real-time data analytics (*Real-time Analytics*).

4.2.2.3 Data storage manager

A system that is used to store the various data that the system needs to keep in a secure repository; this component is a common database management system.

4.2.2.4 Alert manager

Helps to create and maintain all types of system alerts related to the health status of the elder. In addition, it consists of a component (Alert Engine) in continuous operation that triggers alerts within the system based on the analysis of the various monitored patient data.

4.2.3 Information view

This view show the Data structure in the system. A first approach of the first level entities within the system and its relations is shown in Figure 4.

5 Implementation details

The components of the architecture are formed by a series of elements which were chosen due to their required functional characteristics described in the previous sections of this article. The main components are closely related to the Emerging Fast Data Architectures and are described below (Figure 5):

5.1 Mobile technologies

Inside the home, a Smartphone collects data from the sensors, which are carried by the elderly person. Such data is transferred via Bluetooth to the Smartphone, which transfers the data to an MQTT Broker Cluster via an MQTT-Client. In addition, this Smartphone can send data related to its built-in sensors for example, the room temperature. Because Smartphone is a device with limited resources, that is why the MQTT was used (MQTT is a lightweight protocol). In addition, the Smartphone is also employed to support applications related to emergency management that can be used by those users who work collaboratively in elderly care.

5.2 eMQTT cluster

MQTT is a lightweight publish/subscribe message protocol and its implementation is useful for the collection of data coming from devices with limited resources. In this article, a central broker to manage the distribution of data from the home of the elderly was implemented; for this, we used eMQTT, which is a distributed, massively scalable, highly extensible MQTT message broker [26].

5.3 Confluent platform

The incorporation of systems to large cities and large sectors of the population require highly scalable technologies and because eMQTT provides a medium scale (millions of messages), that is why Confluent was selected as an additional messaging broker [27]. In this way it is also possible to reduce the back pressure that can be exerted on the eMQTT cluster. Confluent improves Apache Kafka and support trillions of messages which is good for high scalable application in Smart City. From this platform, the authors used the Kafka Cluster with three Kafka servers and the Zookeeper server.

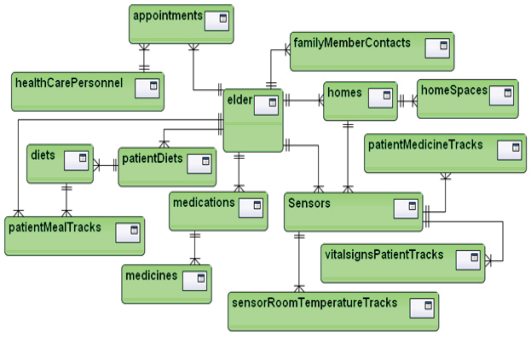

Fig. 4. Data structure.

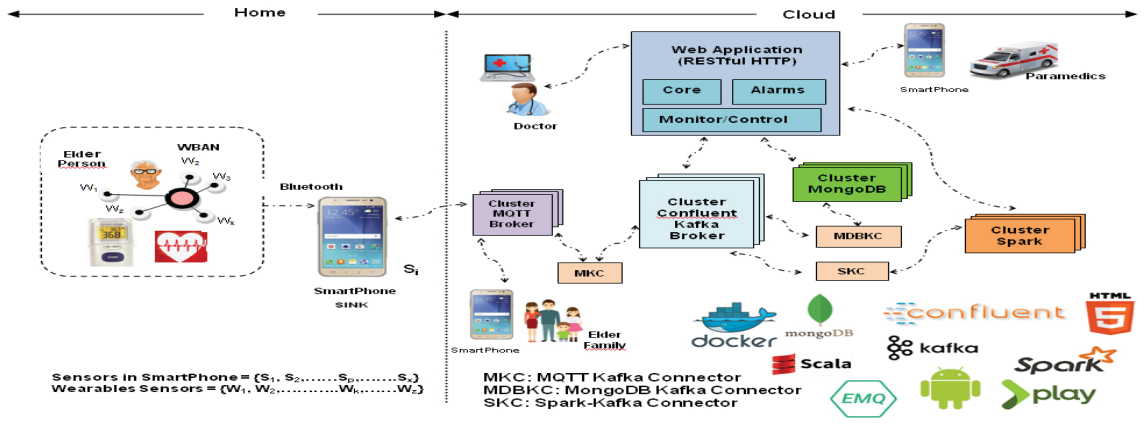

Fig. 5. System architecture

5.4 Apache Spark

The characteristics of Spark such as streaming processing, machine learning libraries, fast, SQL language and using in-memory distributed computing together satisfy the requirements of the system. Another design alternative with similar characteristics but of almost equal capacity is Apache Flink. On the other hand, unlike Hadoop, Spark can be used in batch mode and in near real-time.

5.5 Connectors

A set of additional elements were used to connect some components of the systems: (1) *Mqtt-Kafka connector* is a reactive connector, which has been specially built by the authors of this paper to connect the data flow between MQTT-Broker and Kafka-Broker, for its implementation the authors used Akka Streams-Kafka library [28] and Paho-Akka library [29]; (2) *Kafka-MongoDB Connector* is a Kafka-Connect-Mongo-Sink, which permits that the data coming from the Smartphone are stored quickly and safely without intermediate manipulation [30]; (3) *Spark-Kafka connector* is a Spark streaming package, which connect Kafka and Spark [31], through this connector, data analytics services were built; and (4) *MongoDB connector for Spark* helps to implement services to analyze the data extracted from MongoDB, moreover, it is especially useful in mining or machine learning techniques [32].

5.6 Data storage infrastructure

MongoDB is a free and open-source distributed and document-oriented. It ingests and stores data in near real-time and in an operational capacity, it uses JSON-like documents, it is easy to work with, and geographic distributions are built in and it is not difficult to use. MongoDB is fully scalable compared to traditional relational databases like MySql. In this paper, a MongoDB Cluster was deployed [33].

5.7 System Core

It is a Web Application (REST Web Services) that was developed with Play Framework [34] and the Scala programming language. Play Framework is a MVC framework and is based on a lightweight, stateless, web-friendly architecture for highly-scalable applications. The User interfaces (MVC views) were developed using HTML5, CSS, Javascript. Bootstrap and JSON. Figure 6 shows the microservices of the e-Health System.

5.8 Security

The security of the data in the processes of data transfer and storage is of great importance in the distributed processes, which is related to the selected protocols and technologies. First, the eMQTT broker supports to authenticate MQTT clients with ClientID, Username/Password, IpAddress and even HTTP Cookies. Moreover, confluent provides security through TLS or Kerberos authentication, encrypt network traffic via TLS and authorization via access control lists (ACLs). Besides, core security was implemented with Silhouette, which supports several authentication methods, including OAuth, OpenID, CAS, Credentials and Basic Authentication [35]. Additionally, the authors consider the use of anonymization to prevent inferences in the data. Furthermore, MongoDB provides various of security mechanisms, such as authentication, access

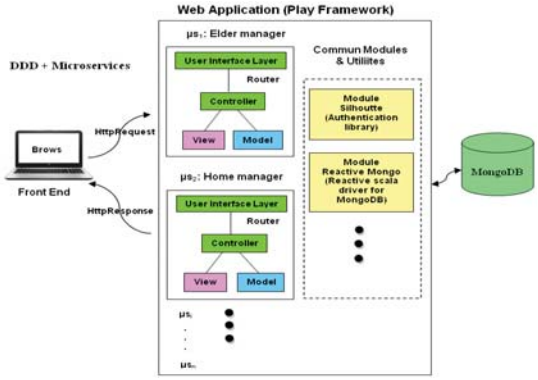

Fig. 6. The microservices of e-Health system.

control, encryption, to secure your MongoDB deployments: Role-Based Access Control, TLS/SSL, etc.

Finally, for the development of the system software, the authors used mainly: macOS Sierra version 10.12.3 (2 GHz Intel Core i5, RAM 8G), IntelliJ IDE 2016.3.6, Ubuntu/Toshiba (1.00 GHz, AMD 64 bits, RAM 3.3 GB), Android Studio Version 2.3.1, JRE: 1.8.0_112-release-408-b6 x86_64, JVM: Open JDK 64-Bit Server VM, Play Framework 2.4.8, the Scala programming language version 2.11.8, Apache JMeter 3.3, and Docker Machine 0.12.2.Emulator.

6 Experiment with the services

In the laboratory the services were implemented and deployed using a macOS Sierra machine (2 GHz, Intel Core i5, RAM 8G) that served as the Cloud environment and an Ubuntu/Toshiba machine (1.00 GHz, AMD 64 bits, RAM 3.3 GB) that served as client machine (the number of threads that can be created on both computers is limited). In addition, since our experimental analysis is based in obtaining the throughput of the services, certain critical points of them are analyzed. Besides, Apache JMeter was used as a tool to obtain these values.

6.1 Core services

To study the scalability of the core services (Web Services), some microservices related to the management of the data of the elderly were selected. Table 2 shows the throughput of the services according to the number of users requesting the service. It is clear that the services that do not have operations in the database respond more quickly.

6.2 Monitoring services

The most important elements of the monitoring services are the Message Brokers, eMQTT and Kafka, which allow the construction of clusters to provide more scalability. In this experiment, the tests were carried out with a cluster with three MQTT nodes and another cluster with three Kafka nodes (including a Zookeeper instance). In Table 3 you can see how Kafka is more powerful than the eMQTT. The use of Kafka helps to release the back pressure that may be exerted on the eMQTT and at the same time allows a large scale to the applications that use telemetry based on the MQTT protocol. Thus, the architecture can grow flexibly to add scalability with more nodes in the clusters.

Table 2: Core service scalability

List		Save		Update		Delete	
N[a]	T[b]	N[a]	T[b]	N[a]	T[b]	N[a]	T[b]
100	52.7	10	10	10	9.90	10	24.80
110	67.2	20	19	20	19.50	20	30.00
120	81.7	30	28.6	30	25.40	30	35.30
130	69.9	40	38.4	40	28.10	40	35.3
140	64.9	50	44.1	50	25.40	50	33.6
150	64.8	60	36.5	60	22.50	60	30.2
160	116.5	70	45.1	70	20.80	70	31.6
170	82	80	33	80	27.80	80	26.1

a. N: Numbers of threads (users)/second

b. T: Throughput (request/second)

Table 3: Monitoring service scalability

eMQTT		KAFKA	
N^c	T^d	N^c	T^d
2200	155.60	3500	131.6
2500	121.20	4000	125.9
2700	139.50	4500	134.6
2900	161.90	5000	146.7
3000	158.70	6000	152
3500	198.80	7000	168.2
4000	157.20	8000	173
4500	151.10	9000	186.3
5000	183.40	10000	185.2

c. N: Numbers of threads (users)/second

d. T: Throughput (request/second)

6.3 Emergency services

The emergency services are based on a rule engine that keeps a constant control over the vital signs of the elderly. It is a Streaming service that continuously and dynamically monitors the health status in near real-time. If at any time an elder person suffers a serious situation, alert messages are sent to the people and teams that take care of them through the monitoring interface or through mobile messaging. In Figure 7 shows each of the Elder vital signs. The interface collects the data at the same time points out the levels of each of these variables, for example the instant shown indicates that the Elder has a *Prehypertension* level, he has a *Normal* temperature level and his heart rate is categorized as *Warning*.

7 Conclusions and future work

This article presents a methodology for the design and implementation of e-Health IoT Reactive Services for Elderly Care at Home. This system is supported by an architecture with Fast Data features and presents reactive features in order to be applicable to Smart city

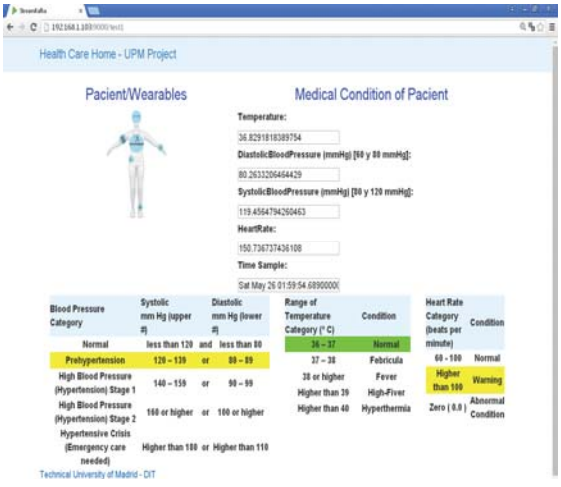

Fig. 7. Health care monitor

environments to be used by those users who work collaboratively in elderly care. Unlike the systems supported by infrastructure that are difficult to scale (for example: monolithic systems), the system presented in this article provides near real-time data flow management, high-capacity messaging systems and near real-time data analytics which play a key role in the e-Health Systems. The services implemented have a high degree of scalability and near real-time responses. The authors conclude that this type of system in the field of health is necessary due to its reactive capacity and for real-time data flow management. In the future, it is desirable to test scalability especially with the deployed system in the Cloud and to implement innovative services for the care of the elderly.

Acknowledgment

Special thanks the financial support of the "Secretaria de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) of the Republic of Ecuador" in this research and its progress.

References

[1] World Health Organisation. Health topics: eHealth. [Online]. Available: http://www.who.int/features/factfiles/ageing/en/ [Accessed: Abril 20, 2018].

[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions", Future Generation Computer Systems-the International Journal of Grid Computing and Escience, vol. 29, pp. 1645-1660, Sep. 2013, 2013.

[3] Islam, SM Riazul et al., "The internet of things for health care: a comprehensive survey", IEEE Access, vol. 3, pp. 678-708, 2015.

[4] R. K. Ganti, F. Ye and H. Lei, "Mobile crowdsensing: current state and future challenges," in IEEE Communications Magazine, vol. 49, no. 11, pp. 32-39, November 2011.

[5] Oussous, A., et al. Big Data technologies: A survey. Journal of King Saud University – Computer and Information Sciences (2017). [Online]. Available: http://dx.doi.org/10.1016/j.jksuci.2017.06.001 [Accessed: Abril 20, 2018].

[6] J. Bonér, D. Farley, R. Kuhn, and M. Thompson. Reactive Manifesto. [Online]. Available: http://www.reactivemanifesto.org [Accessed: Abril 20, 2018].

[7] D. Wampler. "Fast Data Architectures for Streaming Applications", O'Reilly, 2016.

[8] T. Taleb, D. Bottazzi, M. Guizani and H. Nait-Charif, "Angelah: a framework for assisting elders at home," in IEEE Journal on Selected Areas in Communications, vol. 27, no. 4, pp. 480-494, May 2009.

[9] M. A. Hemairy, M. A. Serhani, S. Amin and M. A. Ahmed, "Integrated and Scalable Architecture for Providing Cost-Effective Remote Health Monitoring," 2016 9th International Conference on Developments in eSystems Engineering (DeSE), Liverpool, 2016, pp. 74-80.

[10] V. Balasubramanian and A. Stranieri, "A scalable cloud Platform for Active healthcare monitoring applications," 2014 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), Hawthorn, VIC, 2014, pp. 93-98.

[11] S. Biswas and S. Misra, "Designing of a prototype of e-health monitoring system," 2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, 2015, pp. 267-272.

[12] H. Dagale et al., "CyPhyS+: A Reliable and Managed Cyber-Physical System for Old-Age Home Healthcare over a 6LoWPAN Using Wearable Motes," 2015 IEEE International Conference on Services Computing, New York, NY, 2015, pp. 309-316.

[13] T. N. Gia, A. M. Rahmani, T. Westerlund, P. Liljeberg and H. Tenhunen, "Fault tolerant and scalable IoT-based architecture for health monitoring," 2015 IEEE Sensors Applications Symposium (SAS), Zadar, 2015, pp. 1-6.

[14] N. Zhu et al., "Bridging e-Health and the Internet of Things: The SPHERE Project," in IEEE Intelligent Systems, vol. 30, no. 4, pp. 39-46, July-Aug. 2015.

[15] L. Pescosolido, R. Berta, L. Scalise, G. M. Revel, A. De Gloria and G. Orlandi, "An IoT-inspired cloud-based web service architecture for e-Health applications," 2016 IEEE International Smart Cities Conference (ISC2), Trento, 2016, pp. 1-4.

[16] A. Raji, P. Kanchana Devi, P. Golda Jeyaseeli and N. Balaganesh, "Respiratory monitoring system for asthma patients based on IoT," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-6.

[17] C. Vuppalapati, A. Ilapakurti and S. Kedari, "The Role of Big Data in Creating Sense EHR, an Integrated Approach to Create Next Generation Mobile Sensor and Wearable Data Driven Electronic Health Record (EHR)," 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, 2016, pp. 293-296.

[18] Y. Zhang, M. Qiu, C. W. Tsai, M. M. Hassan and A. Alamri, "Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data," in IEEE Systems Journal, vol. 11, no. 1, pp. 88-95, March 2017.

[19] J. Dutta, F. Gazi, S. Roy and C. Chowdhury, "AirSense: Opportunistic crowd-sensing based air quality monitoring system for smart city," 2016 IEEE SENSORS, Orlando, FL, 2016, pp. 1-3.

[20] Nick Rozanski and Eoin Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2nd Edition. Addison Wesley, 2011.

[21] E. Wolff, Microservices: Flexible Software Architecture. Addison-Wesley Professional, 2016.

[22] Qian, K. et al., 2010. Software Architecture and Design Illuminated. Massachusetts: Jones and Barlett Publishers.

[23] M. Richards, "Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them", O'Reilly Media, 2015

[24] M. Amundsen, M. McLarty, R. Mitra, I. Nadareishvili. "Microservice Architecture", O'Reilly, 2016.

[25] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, "Composite cloud application patterns," in Cloud Computing Patterns. Vienna, Austria, Springer, 2014.

[26] EMQ - The Massively Scalable Open Source MQTT Broker. [Online]. Available: http://emqtt.io/ [Accessed: Abril 20, 2018].

[27] Confluent Platform, a More Complete Distribution of Apache Kafka. [Online]. Available: https://www.confluent.io/product/confluent-platform/ [Accessed: Abril 20, 2018].

[28] Akka Streams kafka. [Online]. Available: http://doc.akka.io/docs/akka-stream-kafka/current/home.html [Accessed: Abril 20, 2018].

[29] Maven Repository: com.sandinh. [Online]. Available: https://mvnrepository.com/artifact/com.sandinh/paho-akka_2.11/1.3.0 [Accessed: Abril 20, 2018].

[30] Kafka Connect Mongo Sink. [Online]. Available: http://docs.datamountaineer.com/en/latest/mongo-sink.html [Accessed: Abril 20, 2018].

[31] Spark Streaming + Kafka Integration Guide - Spark 2.2.0 Documentation. [Online]. Available: https://spark.apache.org/docs/2.2.0/streaming-kafka-integration.html [Accessed: Abril 20, 2018].

[32] MongoDB Connector for Spark - MongoDB Spark Connector v2.2. [Online]. Available: https://docs.mongodb.com/spark-connector/master/ [Accessed: Abril 20, 2018].

[33] MongoDB for GIANT ideas – MongoDB. [Online]. Available: https://www.mongodb.com/ [Accessed: Abril 20, 2018].

[34] Play Framework. [Online]. Available: https://www.playframework.com/ [Accessed: Abril 20, 2018].

[35] Silhouette. [Online]. Available: https://www.silhouette.rocks/ [Accessed: Abril 20, 2018].