applied sciences

MDPI

Article

Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment

Huriviades Calderón-Gómez [1,2], Luis Mendoza-Pittí [1,2], Miguel Vargas-Lombardo [2,*], José Manuel Gómez-Pulido [1,3], Diego Rodríguez-Puyol [3,4,5], Gloria Sención [6] and María-Luz Polo-Luque [3,7]

1 Department of Computer Science, University of Alcalá, 28805 Alcalá de Henares, Spain; huriviades.calderon@edu.uah.es (H.C.-G.); l.mendoza@edu.uah.es (L.M.-P.); jose.gomez@uah.es (J.M.G.-P.)
2 e-Health and Supercomputing Research Group (GISES), Technological University of Panama, 0819-07289 Panama City, Panama
3 Department of Medicine and Medical Specialties, Ramón y Cajal Institute for Health Research (IRYCIS), 28034 Madrid, Spain; diego.rodriguez@uah.es (D.R.-P.); mariluz.polo@uah.es (M.-L.P.-L.)
4 Foundation for Biomedical Research, Hospital Universitario Príncipe de Asturias, 28805 Alcalá de Henares, Spain
5 Department of Medicine and Medical Specialties, University of Alcalá, 28805 Alcalá de Henares, Spain
6 School of Medicine Autonomous, University of Santo Domingo, Santo Domingo 10105, Dominican Republic; dragloriasencion@gmail.com
7 Department of Nursing and Physiotherapy, University of Alcalá, 28801 Alcalá de Henares, Spain
* Correspondence: miguel.vargas@utp.ac.pa

Abstract: This article proposes a new framework for a Cloud-based eHealth platform concept focused on Cloud computing environments, since current and emerging approaches using digital clinical history increasingly demonstrate their potential in maintaining the quality of the benefits in medical care services, especially in computer-assisted clinical diagnosis within the field of infectious diseases and due to the worsening of chronic pathologies. Our objective is to evaluate and contrast the performance of the architectural patterns most commonly used for developing eHealth applications (i.e., service-oriented architecture (SOA) and microservices architecture (MSA)), using as reference the quantitative values obtained from the various performance tests and their ability to adapt to the required software attribute (i.e., versatile high-performance). Therefore, it was necessary to modify our platform to fit two architectural variants. As a follow-up to this activity, corresponding tests were performed that showed that the MSA variant functions better in terms of performance and response time compared to the SOA variant; however, it consumed significantly more bandwidth than SOA, and scalability in SOA is generally not possible or requires significant effort to be achieved. We conclude that the implementation of SOA and MSA depends on the nature and needs of organizations (e.g., performance or interoperability).

Keywords: cloud computing; eHealth; elderly people; infectious diseases; MSA; SOA; telemonitoring; versatile high-performance

Citation: Calderón-Gómez, H.; Mendoza-Pittí, L.; Vargas-Lombardo, M.; Gómez-Pulido, J.M.; Rodríguez-Puyol, D.; Sención, G.; Polo-Luque, M.-L. Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment. *Appl. Sci.* **2021**, *11*, 4350. https://doi.org/10.3390/app11104350

Academic Editor: Alessandro Di Nuovo

Received: 13 April 2021
Accepted: 6 May 2021
Published: 11 May 2021

1. Introduction

Recent developments in the field of Cloud computing have stimulated the need to create new software solutions for the monitoring, management and optimization of an organization's services, in which developers build an application and are not concerned about the underlying infrastructure. For this reason, several authors have proposed different software solutions, such as infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS) and function as a service (FaaS), focused on the intensive processing of data from different sources (e.g., Internet of Things (IoT) devices, data repositories or other sources) that will be consumed by the services to offer users different business-specific functionalities [1–5].

Within this context, the eHealth field is an area that is becoming increasingly important in public health policies because health care delivery systems (e.g., public health management or electronic health records (EHR)) are constantly evolving towards more patient-centered customized services and digital transformation (e.g., eGovernment) [6,7], which, based on data collected by the eHealth systems in conjunction with the recommender systems, can accelerate and drive decision making [8].

Considering the above, we have identified a growing concern regarding the increase in the aging population in our society. In Europe, the expected growth of the population over 65 will be 74% by 2060 [9–11], while in North America, it will be 20% by 2050 [12–14]. This trend places a serious burden on the public health system, since this group of people have high age-related comorbidity (e.g., degeneration in physiology and the abilities of memory, vision and decision), consuming extensive resources of both primary and specialized care [9,15], and consequently, it is more difficult for the elderly to adapt within society.

Telemonitoring systems are very popular today, since these systems are primarily based on the remote analysis of biometric data or daily activities, stemming from either a low availability of medical centers or mobility complications of people. These systems require the use of the Internet of Healthcare Things (IoHT) such as body sensors, bed sensors and smart watches for the collection and monitoring of people through their data, which allows the detection of anomalies in the vital signs of the elderly, and thus alerting medical personnel to these problems [16–19]. In addition, telemonitoring is an effective low-cost means for periodic health exams or preventive treatments in nursing homes [13,20].

Based on the ideas presented and the use of telemonitoring, we have focused on the design, development and implementation of a platform aimed at detection and clinical diagnosis assisted by a recommender system based on artificial intelligence (AI) algorithms within the field of infectious diseases for the elderly population living in nursing homes. It should be noted that this article is part of an extensive study in the field of infectious diseases focused on the three target groups of acute respiratory infections, urinary tract infections and skin and soft tissue infections. However, the article presented here is based in part on several previous studies by the partners of the Design and implementation of a low-cost intelligent system for the prediagnosis and telecare of infectious diseases in elderly people (SPIDEP) [21–25].

Another priority task in the concept of a platform targeted for Cloud computing environments is to define the architectural pattern to be implemented in the platform (e.g., service-oriented architectures (SOA) or microservices architecture (MSA)), which should be based on the needs (e.g., agility, maintainability or high-performance) and the nature of the organization (e.g., eHealth, transport, energy or supply chain management) [7,26–29].

For these reasons, this article describes the work done to evaluate and contrast which of the architectural patterns (SOA or MSA) is more consistent with the attributes of software focused on versatile high-performance within the eHealth context [21], while considering the strengths and weaknesses of each architecture based on quality of service (QoS) and its corresponding metrics. Therefore, it has been necessary to modify the SPIDEP platform into two variants following the guidelines of each architectural pattern, which allows us to analyze which of the variants optimally meet the proposed specifications.

However, the contributions of this article are: (i) to provide an overview of the differences between SOA and MSA; (ii) to identify the implications of designing, implementing and deploying SOA and MSA oriented platforms; (iii) to measure the performance of SOA and MSA variants, taking as reference the quantitative values obtained from experiments, and to contrast these obtained results with respect to existing research results. Nevertheless, the SOA and MSA architectural patterns can be considered complementary allies for an interenterprise architecture that confers a suite of different services rather than being competitors; therefore, it is expected that this future architecture will allow the integration and interconnection of different eHealth applications along with microservices adapted to machine learning for various medical scenarios (e.g., a patient monitoring system for hemodialysis or early forecasting of COVID-19).

This article is organized as follows: Section 2 presents a brief description of the related works, including the motivation for the SPIDEP project and its importance in the field of eHealth. Section 3 details the case study applied in SPIDEP, which is a project aimed at telemonitoring in nursing homes. Section 4 describes in detail the architectural models applied in the designs of the SPIDEP platforms and the technologies implemented. Section 5 shows the results of the spike testing based on the three categories of response time, efficient use of infrastructure and network consumption. The benefits and disadvantages of the results are evaluated in Section 6. Finally, Section 7 provides the conclusions of this study, analyzes its potential and suggests future research activities.

2. Background and Related Work

Currently, there is a growing interest in the use of archetypes for the development of various eHealth applications to represent the structure of clinical information and its specifications within a platform [30–32]; examples include traditional information systems, clinical decision support systems, platforms oriented to the HL7-FHIR protocol or telemonitoring systems, whose purpose is to significantly improve the accuracy of medical diagnosis by establishing expert systems-based prediction approaches or other means of artificial intelligence.

This interest is why we have performed a brief review of the various existing projects that show different analyses, applications and research conducted in this field; however, classifying these projects according to the architectural pattern implemented (i.e., SOA or MSA) was needed since it is necessary to analyze how SOA and MSA influence the development, integration and deployments of eHealth applications and how these patterns adopt principles or practices to address the software requirements.

2.1. Service-Oriented Architecture (SOA)

According to the norms and standards of service-oriented quality [33,34], SOA was one of the most used software architectural approaches in the early 2000s; it was used for the design and development of applications as services. This architecture aims to provide uniformity in the design, implementation and invoking of services required to meet the desired needs of the application [35]; that is, any process, subprocess or logic of an organization can be encapsulated in services [36].

As can be inferred, SOA is a business-focused information technology (IT) architecture that is supported by the integration of services. Consequently, all the services designed in SOA must comply with the following three fundamental software attributes [37,38]: First, each developed component must have the ability to be reused within the application (reusability). Second, the components must be able to communicate with each other or with other external applications (interoperability). Third, the application must allow the ability to adapt to future needs or objectives of the organization (extensibility).

As a follow-up to this activity, several research projects that propose the combined use of this architectural pattern with several software attributes will be outlined below, focused on various eHealth scenarios.

For example, Gavrilov et al. [38] propose a novel conceptual model of EHR focused on the aspect of cross-border interoperability [39], whose objective is to meet the needs of interconnectivity between the different eHealth systems of Macedonia, under the specifications of the European Patients Smart Open Services project [40,41]. Thus, the authors chose to combine the SOA architectural pattern with the use of the data warehouse (DW) and the following three technologies oriented to an extract, transform and load (ETL) process: the interface description (using Web services description language (WSDL) documents), the communication protocol simple object access protocol (SOAP) (using extensible markup language (XML)) and universal description, discovery and integration (UDDI) repositories (allowing users to find services) [38]. They have shown that the proposed model achieves a successful data exchange between the various platforms tested. Additionally, this model allows the classification and indexing of the data, which allows SOA to be able to commu-

nicate with different applications other than the predefined DW because each healthcare system is designed, implemented and validated by the HL7 fast healthcare interoperability resources (FHIR) protocol.

Likewise, Amin et al. [39] propose building a framework for the exchange of information among the eHealth systems of Indonesia, whose purpose is to integrate and synchronize data from different (heterogeneous) platforms through web interoperability. This framework was designed under the concept of service-oriented analysis and design (SOAD) [42,43], which is composed of the three main phases of the conceptual view (CV), which illustrates to those involved all the activities that encompass the workflow of the organization; the logical view (LV), which encapsulates all the logic identified by the CV in various software services; and the physical view (PV), which implements the different web layers, including the presentation, application service, domain model and data access layers. The three previous phases focus on a logic based on SOA, whose objective is to process all user information and exchange data among platforms using the RESTful protocol [44]. They validated their framework by successfully relating eHealth systems and integrated entities and the exchange of data and information in the form of an interoperability matrix (IM), so that organizations can use this matrix as a point of reference for adaptation.

Hameedet et al. [45] present a framework oriented to the real-time monitoring of the vital signs of patients (e.g., electrocardiogram, body temperature, pulse rate and oxygen in blood); their framework consists of the combined use of the SOA architecture, rich Internet application and the IoT (Arduino and eHealth sensor shield) [46]. This combined use is necessary because the framework requires the ability to communicate between services and their sensors and to make associations between different medical administrations (e.g., clinics or medical centers); therefore, the implementation of the REST protocol was required to comply with this aspect [47]. They have shown that the proposed framework increases the efficiency of data collection because each service is designed, implemented and validated in an automated manner for the monitoring of a large number of patients.

Silva et al. [48] develop a pilot system called the mHealth application, based on existing practices of IoT and the SOA architecture, focused on telemonitoring of patients. The main objective of this system is to optimize the follow-up services and treatment of arrhythmias or other heart conditions of the patients remotely, which provides coverage to rural or difficult to access areas. This system consists of the four main modules of the monitoring process and patient electronics (main component of the application), body sensor (cardiac arrhythmia detection), caching mechanisms (temporarily stores data in case of network unavailability) and an alert messaging system (sends a warning to health care professionals if an anomaly is detected). They validated their system using a performance assessment and quality of experience (QoE); their results demonstrated a significant improvement in medical care and assistance to patients. In addition, an unexpected result was that the management of medications was more accurate, efficient and less costly for the medical institution.

2.2. Microservice Architecture (MSA)

Currently, there is a growing interest in MSA in both the industrial and research sectors due to its various advantages, including flexibility, agility, infrastructure automation and loose coupling [49–51]. MSA is considered a more refined and simplified version of SOA [33,52], since this architecture responds to the needs of scalability of applications by segmenting the logic of an organization into a series of separate services that are executed as independent processes; that is, it is not necessary to use the same languages, databases or development platforms [21,53,54]. In addition, MSA inherits from SOA the concept of interoperability through the implementation of lightweight mechanisms such as HTTP-based API and others [39,50,55].

Many organizations have recognized the benefits in the migration of their legacy software to microservices-based solutions to enable their applications to evolve, rather than acquiring new third-party software [52,54]. However, performing this process (known as

granularity) requires using a pathway or model that can successfully migrate the application; consequently, this factor depends on the use of design patterns (e.g., decompose by business, by verb or use case, by subdomain or by nouns or resources) and how the performance stability of the services is affected [50,51,53,56].

A description of the most significant studies of the new MSA solutions within the eHealth domain is necessary.

Carranza-García et al. [57] present two technological solutions based on MSA (i.e., cognitive training "VIRTRAEL" and frailty prevention "PREFRA"). These solutions aim to integrate the different age-related aspects from the cognitive, physical and social levels; therefore, when addressing these factors, more complete and reliable assessments of the elderly are obtained. However, to achieve this objective, the systems must have the four essential software attributes of reusability, extensibility, interoperability and composition, since these are required for the design, development and implementation of more complex functionalities that make use of different recent technologies (e.g., web or mobile). They have shown that the proposed technologies are a viable solution aimed at meeting the required quality attributes.

García-Moreno et al. [58] develop an MSA-based system that collects sensory data of the elderly in their daily activities (e.g., toilet hygiene or functional mobility); these data include not only a physical dimension but also cognitive and social dimensions. This system is supported by using IoT (i.e., wearable devices), which are used by microservices focused on the collection, analysis and interpretation of data. In addition, this system is supported by several machine learning algorithms (e.g., kNN, RF or NB) to build more accurate models for detecting anomalies in people [59].

Jarwar et al. [17] provide a model focused on semantic interoperable data-driven microservices. This model is responsible for capturing, representing and analyzing various types of data related to depressive disorders (DD). These data are vital to providing the model with a way to monitor the symptoms of DD through the use of indicators (e.g., user facial expressions, voice tone, user emotions from wearable sensors, social network services posts (SNS) and tweets). Generally, microservices do not make intelligent decisions based on data [60]; therefore, it is necessary to implement the following two dedicated servers in the model for this scenario: data mining/machine learning (extracts features and monitors the symptoms from the sensor data and SNS) and web of objects server (applies the semantic web technologies and inference of the user situation to provide the services to the users).

Da Silva et al. [30] propose a tool called Microservice4EHR, which is based on the OpenEHR standard used for the computational representation of EHR (archetype) [61] and the conceptualization of a microservices-oriented software architecture. This tool consists of the five functionality phases of a tool access key, the host server address of the MSA component, a graphical interface with health data based on archetypes, a graphical interface filename and the input/output data (in JSON format), which is processed and presented in a web application [30]. Moreover, all these approaches have the purpose of building reusable components that can be used as building blocks in health applications (e.g., a blood donation center in northeastern Brazil). Additionally, this attribute improves the capacity of the maintainability and interoperability of the applications between the information systems.

2.3. SOA vs. MSA

Considering the above, the previously cited studies reveal different eHealth applications, based on SOA or MSA, that focus on one or several of the software attributes of interoperability, maintainability, reusability and scalability.

Both architectures have their strengths and weaknesses, according to the needs of the software and implementation [62]. SOA focuses mainly on the sharing of services in relation to the processes or capabilities of an organization [50,63]. In this sense, SOA can refer to a broad accumulation of knowledge over the last two decades [62]. Regardless, this

approach presents many issues when correcting errors or adding new functionalities, since the design of the application is very complex and time consuming. In addition, scalability in SOA is generally not possible or requires significant effort to be achieved. Consequently, organizations are beginning to migrate to other architectural solutions such as MSA [64,65].

MSA allows efficient development, implementation and maintenance of services compared to SOA. In fact, since the services are autonomous, they can be developed, maintained and tested independently, which facilitates their automation and continuous deployment (e.g., DevOps practices) [21,50,66]; however, such practices entail complexities inherent to a distributed computing environment in relation to the isolation, availability, security and transaction of services [33,50], which in turn causes a fundamental problem known as granularity (i.e., if a microservice needs to decompose or merge) [21,51].

Based on these considerations, the implementation of SOA or MSA depends on the nature of the target system to be developed. Thus, for the development of our eHealth application, it is of utmost importance to contrast the performance between SOA and MSA based on the concept of versatile high-performance [21], since it is necessary to meet the needs of the environment of health organizations, focused on horizontal scalability. Quicker and more accurate real-time responses to demand peaks are thus enabled [22]. Additionally, it is necessary to allow the hierarchy and consolidation of a patient's clinical information, including what is necessary not only for generating standard reports but also for allowing the integration of recommender systems, to support decision making with consistent and truthful parameterized indicators of the data collected [21].

3. A Case Study from eHealth

The work presented here is based on a previous study performed by the SPIDEP project part of the Joint Call ERANET LAC 2015—FP7 [67], whose objective was to build an intelligent system based on information and communication technologies (ICT) to support the early diagnosis of infectious respiratory and urinary diseases in the elderly [21,24]. For this, an international consortium was formed between the clinical research teams and the Infectious Diseases Unit of the Hospital Universitario Principe de Asturias (Alcalá de Henares, Spain), the Chronic Diseases and Cancer Area of the Ramón y Cajal Institute for Health Research (Madrid, Spain) and the School of Medicine Autonomous University of Santo Domingo (Dominican Republic) and the research teams of the Computer Sciences group of the University of Alcalá (Universidad de Alcalá) and the Technological University of Panama (Universidad Tecnológica de Panamá).

This study incorporates patients living in the Francisco de Vitoria and Cardenal Cisneros nursing home in Spain and residents of the Carls George and Nuestra Señora del Carmen Institution for the Care of the Elderly in the Dominican Republic. The main purpose of the study is to provide health services and care to people at risk or who have difficulties in accessing health services through the use of new technologies to achieve greater efficiency in the organization and care of special population groups, with the added value of cost savings [21].

Actually, health monitoring has become one of the biggest areas in the technology industry, developing many smart devices like watches or another more specific body sensors. This has allowed researchers to contribute to the development of eHealth platforms focused on regularly monitoring the vital signs of at-risk patients, with the aim of carrying out treatments or preventive interventions to minimize health problems and reduce emergency care [10,16–18,46,48,58].

4. Model and Design of SPIDEP Platforms

To perform this study, a merging of the computational paradigms of the Edge and the Cloud was used [68,69], based on a hybrid Cloud architecture to support medical telemonitoring applications [21] under the two architectural patterns (i.e., SOA and MSA).

In this sense, we focus on the evaluation of the design, development and implementation of the services created for our use case (SPIDEP) and how they affect performance related to the response time to queries, the efficient use of infrastructure and network consumption.

4.1. Analysis of Software Attributes

This preliminary step is fundamental for the design and development of applications, since it is necessary to define an adequate set of software attributes according to the needs identified [7].

In response to these considerations, it is of utmost importance to mention our needs; they are as follows: (i) neutral technology—those responsible for the development, implementation and monitoring of the assigned services may decide which is the best technology to use according to their objectives; (ii) automation and continuous deployment—each component should be autonomous, independent and focused on a specific task according to the established decomposition (e.g., by verb or use case). In addition, it must have the individual ability to replicate to balance the load, as necessary; (iii) scalable to the demand—the services require a horizontal scalability that allows a faster and more precise reaction to the demand peaks; (iv) interoperable—each service must run a lightweight communication mechanism (e.g., RESTful) to communicate between the different applications or services, either their own or those of third parties. In addition, this mechanism should not have access to visual elements, i.e., it must be separate from the front-end; and (v) high performance—the application must support a large volume of queries of the REST methods (e.g., GET, POST, PATCH and DELETE) from the integration, transfer and storage of the biometric sensors from the different nursing homes (Edge) to the various services offered by the SPIDEP hybrid (Cloud).

As explained in Section 2.3 (SOA vs. MSA), it was decided to adopt the concept of versatile high-performance, since it allows us to meet the needs of the project [21].

4.2. Variants of Software Architectures

Considering the above, it was necessary to build two variants of SPIDEP, according to the specific attributes of each architectural pattern; however, SOA and MSA have different approaches in terms of how their services are designed, implemented and deployed [33,70]. In addition, it must be remembered that MSA remains an architecture in training, while SOA has been studied for more than a decade [35,62].

Therefore, it has been decided to establish two branches of the SPIDEP application since it allows us to contrast and evaluate which of the variants fully meets our criteria [71–73].

4.2.1. SPIDEP SOA Variant: Platform Design

The first proposed platform (SPIDEP-SOA) is intended to manage the interactions between users (consumers in the upper layer) and health services (software functionalities in the lower layer), according to the changing needs of the platform (technology, demand and security), as shown in Figure 1.

Next, we will briefly outline our first variant of the platform, which consists of six layers focused on the management of services under the computational paradigms (i.e., Edge and Cloud) [23], as follows:

- Consumers: This layer allows interaction and exchange between the health professionals of the different homes (nurses and assistants), the hospitals in charge of each region (general physicians) and the eHealth applications (SPIDEP or others).
- Edge layer: Establishes HTTP connections with the APP for the asynchronous sending of medical data in the Cloud (i.e., the Cloud layer) [33], whose data come from the different users and their sets of biometric sensors (i.e., blood pressure, pulse rate, body temperature, oxygen saturation and electrodermal activity) [23]. It should be noted that this communication is encrypted by the security layer.
- Cloud layer: This layer performs heavyweight computations; that is, it helps consumers discover, route and deploy health services and infrastructure. In addition, this

layer consists of the three sections of infrastructures and resource, support functions and control system. Additionally, different algorithms are used to achieve various objectives (e.g., error tolerance and load balancing) [74].

- Service layer: Contains services for the end users like medical personnel and nursing home administrative staff with their respective profiles.
- Security layer: Ensures access control and consumer authentication through established security policies, whose objective is to determine the user privileges for certain resources and/or specified levels of services.
- Management system: This layer mainly controls the flow of messages between the different layers. In addition, it is responsible for executing the real-time adaptation of the services (versatility) according to demand, e.g., automatically adding a new instance, monitoring the status of the different components that integrate the platform or other monitoring tasks.

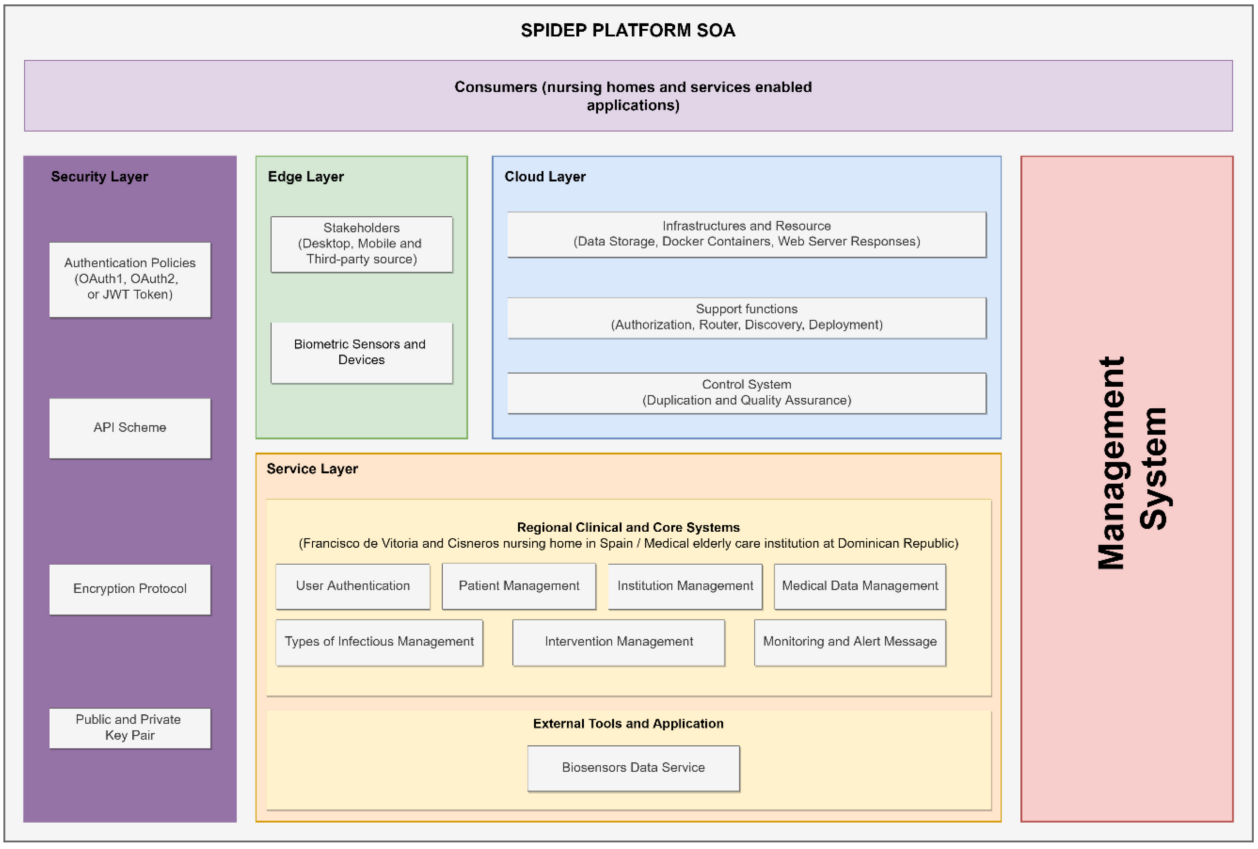

Figure 1. General scheme of the services applied in the SPIDEP platform under the SOA variant (SPIDEP-SOA).

It should be noted that this variant of SPIDEP was a refactoring of the legacy application [23] whose purpose was the modernization of existing services oriented to our current software attribute (i.e., versatile high-performance) [64]; however, the strengths and weaknesses of SOA must be considered.

Considering the above, the strengths of SOA are the following:

- Provides a higher level of compression related to the abstraction of the interfaces than MSA (decoupling) [62], since SOA traditionally has the integration of an enterprise service bus (ESB) [75], which facilitates the interaction between the services of the platform.
- Offers extensive knowledge that has been established over more than a decade, which has proven to be effective and reusable in building platforms. It is thus easier for developers to ensure the quality of service required by the organization [76].

- Suitable for large and heterogeneous systems consisting of many applications, non-independent services and shared components [33,77]. It provides a roadmap for the adoption of its principles, which allows developing or transforming the capabilities of an organization's software system into reusable services for greater flexibility and agility [78].

However, when applying SOA, the following weaknesses or difficulties must be considered:

- Message exchange is traditionally synchronous (wait-to-connect); i.e., it depends on the state of the ESB [33]. However, when applying a design oriented to service implementation patterns, the support of asynchronous messages is integrated, increasing complexity and maintainability and reducing flexibility [33,76,79,80].
- When developing platforms based on SOA, it is complex to add or modify functionalities to what has already been established; redesigning this type of platform consumes significant time and resources [64,81].

4.2.2. SPIDEP SOA Variant: Implementation of the Platform

For the implementation of the SPIDEP-SOA platform, we opted to modernize the first version developed (Beta "traditional implementation of seven SOA services, without API support") [23] to the release candidate (RC) ("Modernization of services and current technologies (language, database, libraries and integration of an API)"), whose purpose is to provide the necessary attributes for the implementation of versatile high-performance applications and contrast it with the MSA variant. Therefore, we proceeded to the refactoring and restructuring of the existing components along with their data [53,82], as shown in Table 1 for each version.

Table 1. General structure in the SPIDEP-SOA platform.

Description		Version	
Category	**Subcategory**	**Beta**	**RC**
System architecture	Architecture styles	SOA	SOA-based on Quality Requirements
	SOA patterns	Enterprise Service Bus (ESB)	Service Implementation Patterns
	Communication protocol	HTTP	HTTP
	HTTP methods	Disable	GET, POST, PATCH & DELETE
	Messaging type	Synchronous	Synchronous/Asynchronous
Used technologies	Programming language	PHP 5.6	PHP 7.4
	Framework	Laravel 5.1 LTS	Laravel 6 LTS
	Core libraries	acoustep/entrust-gui; zizaco/entrust; Laravel Passport; mockery/mockery; phpunit/phpunit; tinker; composer, and others	Guzzle; fideloper/proxy; kylekatarnls/laravel-carbon-2; zizaco/entrust; Queues; Laravel Sanctum; mockery/mockery; phpunit/phpunit; tinker; composer, Custom libraries, and others
	UI	Bootstrap 3	Bootstrap 4
	Web server	Apache	Apache as an endpoint and NGINX as a load balancer for the implementation
	DBMS	MariaDB 10.1	Postgresql 10
	Data access methods	Stored procedures	Stored procedures
	Object–relational mapping	Eloquent	Eloquent
	API Scheme	Disable	RESTful (JSON Request/Response HTTP)
	API Gateway	Disable	Central Entry Point
	Deployment	Bare Metal	Custom Docker Container
	OS	Ubuntu 16 LTS	Ubuntu 18 LTS
	Authentication Scheme	HTTP Basic Authentication	Custom Authentication (OAuth1, OAuth2 or JWT Web Token)
	Encryption Protocol	OpenSSL (AES-256 and AES-128)	OpenSSL (AES-256 and AES-128) and public/private rsa key pair

4.2.3. SPIDEP SOA Variant: Deployment of the Platform

When applying an SOA design pattern (compound, service implementation, service composition, inventory and others) [80], a positive impact is obtained on certain quality requirements (i.e., performance or interoperability), which negatively affects other quality requirements (i.e., redundancy or security). Therefore, it is important to select design patterns according to the software attributes and quality requirements [76].

Considering the above, service implementation patterns were chosen for the development of the SPIDEP-SOA platform, since they allow creating versatile and scalable services that run in the Cloud [83,84]; however, an automated workflow must be defined for the integration and continuous deployment of services [21,80].

Within this concept, we have adjusted our framework for this variant (mainly in the first three sections) [21], because SOA is geared to a modular design to reduce dependencies between platforms [37,85], i.e., that the services collaborate and share their resources (loose coupling), unlike MSA where the services are decoupled and isolated [33].

In addition to the situation, a general overview of the SOA-based proposal follows [21]:

- Teamwork: Responsible for the coordination and monitoring of all platform services (business-centric IT) [86].
- Services: Unlike our MSA framework, each team does not have the autonomy to decide which is the best technology for the development of services. These teams should adhere to the Teamwork decisions that follow the business functionalities. For example, all SPIDEP-SOA services were developed under the Laravel PHP framework and a single DBMS, PostgreSQL 10 (the core systems, except for the mobile app, were developed in Java); however, all services are supported by an asynchronous HTTPS server that is based on transport layer security (TLS) for client authentication and can replicate instances based on demand, unlike the traditional SOA approach.
- Code repository: A Git repository is used to maintain version control of the code from the development branch of the services that make up the platform [87,88].
- Software analysis and testing: The source code and the environment are reviewed to ensure correct functionality and that they meet the desired standard; then, unit tests of the services are performed (e.g., PHPUnit).
- Docker containers (test environment (TE)): To ensure the stability and scalability of our platform, we opted for the customization of a test container (Ubuntu Server 18.04 LTS, 2 Core, 4 GB RAM and 80 GB SSD); however, these containers must pass preliminary test criteria defined by Teamwork before being published in the Docker Hub in a private repository.
- Quality assurance (QA): Functional tests, interoperability tests, and load and performance tests are performed to ensure the quality of services [21,34].
- Docker containers (production environment (PE)): If the integration of all services in the platform is successful, the stable version of the container is deployed to Docker instances (under the same TE attributes); otherwise, the code should be debugged and the QA tests rerun.
- Performed postproduction: Additional tests (e.g., long-term spike testing) are run to ensure that the services work properly in all instances [21,53,89].
- Managed Kubernetes: Kubernetes is used for automating management of computerized services, since it reduces the manual and repetitive processes involved in container management [90]. In addition, Kubernetes scales according to demand, i.e., it increases the use of resources in high demand, and if demand decreases, idle resources are reused [91]. Unlike the traditional SOA approach (which does not have scalability support), it was necessary to make several adjustments to the RC version of the platform.

4.2.4. SPIDEP MSA Variant: Platform Design

The second proposed platform (SPIDEP-MSA) is geared to the use of microservices in the expected medical scenarios of infectious diseases (like its other variant) [21,22,24], as shown in Figure 2.

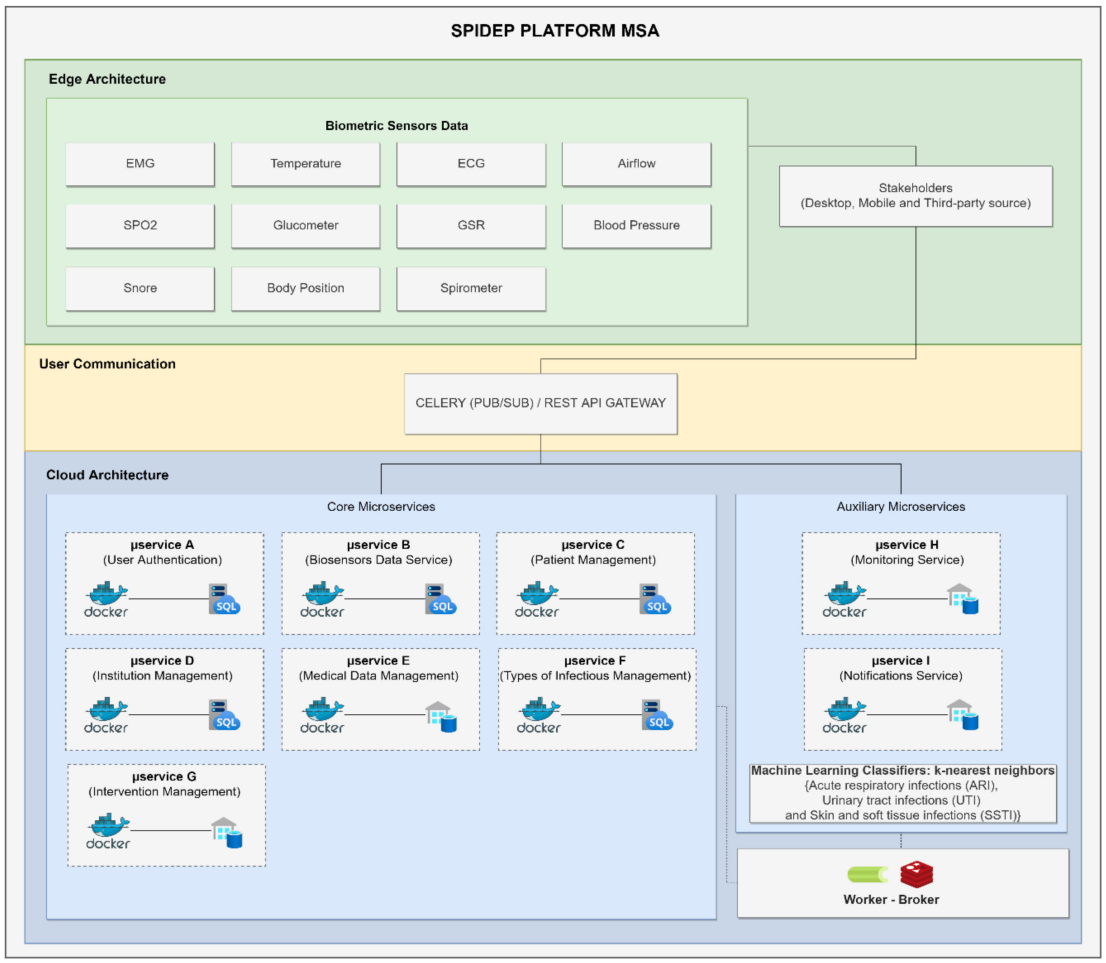

Figure 2. General scheme of the services applied in the SPIDEP platform under the MSA variant (SPIDEP-MSA).

Both platforms aim to support the interpretation of changes in vital signs of elderly people living in residential facilities, thus alerting the medical team in advance of possible infections; however, the MSA variant has a recommender system to improve decision support in the prediagnosis of infectious diseases [25,92]. Following, we will briefly explain the flow of the SPIDEP platform in relation to microservices [23]:

- Edge architecture: Manages the collection, preprocessing and sending of data from the different biometric sensors (e.g., ECG, blood pressure, SPO2 or others), whose data are backed up locally. If necessary, temporary actions can be performed (e.g., delete, filter or update) before the data are sent to the Cloud architecture, which enables analysis with local data to streamline decision making in case of emergency [16,93]. It should be noted that the connections with each APP are made asynchronously and independently for each stakeholder.
- User communication: Provides the necessary mechanisms for correct communication between the different layers and sublayers (e.g., data, validation of credentials or other). This communication is achieved through the representational state transfer (REST) protocol and its methods (i.e., GET, POST, PATCH and DELETE); however, this protocol requires the use of an API gateway. Therefore, taking advantage of some of the strengths of MSA (e.g., decoupling and isolation of microservices) [30], the

Backends for Frontends design was implemented in the API gateway [56,89]. Requests are handled by each stakeholder (i.e., desktop, mobile and third-party source) and not traditionally (i.e., single entry point); consequently, additional security mechanisms must be implemented for their validation [7,94,95].

- Cloud architecture: Provides services for end users (e.g., core or auxiliary microservices) according to their needs and the levels of user roles and permissions [21]. In addition, it manages the stored data of the different DBMS (database per service) [54,96] to provide benefits to target users (e.g., patients, medical personnel or health authorities) [16].

Similarly, we must remember the strengths and weaknesses of MSA. The strengths of microservices are the following:

- Each microservice is incapsulated; therefore, it has more flexibility to use new frameworks, libraries, data sources and other resources [64].
- It allows horizontal scaling of the services instead of vertical scaling as in the case of a traditional or monolithic SOA, which facilitates the ability to use more computing resources (e.g., CPU, GPU and RAM) that will act as a single unit; however, it can be distributed through multiple virtual networks [50,55].
- Each microservice is autonomous, has independence in the execution of its tasks, can be hosted in a specific server on the Internet, is available on the Internet (i.e., any software can interact with it) and does not share the same resources [30,97,98].
- The fault isolation can be obtained without interrupting the normal functioning of the whole system [16,52].
- However, when applying MSA, the following weaknesses or difficulties must also be considered:
- MSA provides many different advantages, but it also increases a solution's complexity. To address increased complexity, an organization should prepare the specific infrastructure for microservices (i.e., obtain a clear picture of the data structure) [64,99].
- If the level of granularity is determined before knowing the business process or the useful life of the platform, it can lead to problems in the reasoning about the services (e.g., if a microservice should be decomposed or merged with another) [33,51].

4.2.5. SPIDEP MSA Variant: Implementation of the Platform

Similarly, for this variant (SPIDEP-MSA), the respective modernization of the latest built version was performed (β v2 "implementation of the nine microservices, with hybrid instances in MariaDB Galera Cluster and NoSQL, and two EndPoint (Mobile/PC)" [21] to the RC ("Modernization and restructuring of microservices and current technologies (language, database, libraries and integration of a third EndPoint)").

The following is a general description of the technologies implemented in each version, as shown in Table 2.

Table 2. General structure in the SPIDEP-MSA platform.

Description		Version	
Category	**Subcategory**	**Beta v2**	**RC**
System architecture	Architecture styles	MSA	MSA
	MSA patterns	Service instance per-container; Database per service & API Gateway	Service instance per-container; Database per service & API Gateway-
	Communication protocol	HTTP	HTTP
	HTTP methods	GET, POST, PATCH & DELETE	GET, POST, PATCH & DELETE
	Messaging type	Synchronous/Asynchronous	Synchronous/Asynchronous

Table 2. *Cont.*

Description		Version	
Category	**Subcategory**	**Beta v2**	**RC**
Used technologies	Programming language	PHP 7.2 (µs-A) & Python 3.6 (µs-B~I)	Python 3.7 (µs-A~G) & Python 3.8 (µs-H~I)
	Framework	Laravel 5.4 LTS (µs-A) & Django 2.1 (µs-B~I)	Django 2.2 LTS (µs-A~G) & Flask 1.1 (µs-H~I)
	Core libraries	PHP/Laravel dependencies (Guzzle;Laravel Sanctum and others), Python/Django dependencies (setuptools;requests;psycopg2; requests_futures;lxml;six;django-cas-server;django-cas-ng;djangorestframework; celery, TensorFlow and others) & Custom libraries	Python/Django~Flask dependencies (setuptools; requests; psycopg2; requests_futures; lxml;six;werkzeug django-cas-server;django-cas-ng; python-cas; djangorestframework; Flask-RESTful; celery, TensorFlow and others) & Custom libraries
	UI	Bootstrap 3	Bootstrap 4
	Web server	Apache as an endpoint and NGINX as a load balancer for the implementation	Apache as an endpoint and NGINX as a load balancer for the implementation
	DBMS	MariaDB Galera Cluster (Galera 3), Cassandra & Redis	Postgresql 10, Cassandra & Redis
	Data access methods	Stored procedures & Column Family	Stored procedures & Column Family
	Object–relational mapping	Eloquent, QuerySet & Django-Cassandra-Engine	QuerySet, SQLALchemy & CQLAlchemy
	API Scheme	RESTful (JSON Request/Response HTTP)	RESTful (JSON Request/Response HTTP)
	API Gateway	Backends for Frontends (2 EndPoints)	Backends for Frontends (3 EndPoints)
	Deployment	Custom Docker Container	Custom Docker Container
	OS	Ubuntu 18 LTS	Ubuntu 18 LTS
	Authentication Scheme	Custom Authentication [OAuth1, OAuth2 or JWT Web Token] & CAS Protocol 3.0 Specification	Custom Authentication [OAuth1, OAuth2 or JWT Web Token] & CAS Protocol 3.0 Specification
	Encryption Protocol	HTTP Authentication (Tokens and SSL); PBKDF2 with a SHA256 hash; public/private rsa key pair and custom authentication settings	HTTP Authentication (Tokens and SSL); PBKDF2 with a SHA256 hash; public/private rsa key pair and custom authentication settings

4.2.6. SPIDEP MSA Variant: Deployment of the Platform

To meet the software attributes and quality requirements, it was necessary to establish an automated workflow for the integration and continuous deployment of the microservices in SPIDEP-MSA. Therefore, we assigned a cross-team (typically a small group of six to eight people [100,101]) for each case (e.g., decomposed by verb or use case [56]); each team is responsible for the development, debugging and deployment of the assigned microservice. This workflow reduces the time between updates or new system functionality (without affecting end users) and implementation of changes to the production environment (without affecting system performance) [21,102,103].

Considering the above, we implemented the same framework proposed for the MSA variant of SPIDEP (beta v2), whose framework is divided into the following nine phases: (i) cross-team project, (ii) service, (iii) code repository, (iv) software analysis and testing, (v) test environment, (vi) quality assurance, (vii) production environment, (viii) performed postproduction and (ix) managed Kubernetes [21].

Unlike the SPIDEP-SOA variant, this platform requires applying the implementation patterns based on MSA (i.e., service instance per container, database per service and Backends for Frontends), since it is necessary to fully support a horizontal scalability of the resources. However, this entails an increase in the effort and complexity to manage all interactions between microservices, according to the size of the organization to be implemented [60]. Additionally, in this variant, we verify the identity of the user for each message sent and/or received through the signal mechanisms; therefore, a secure communication channel is established for different computer attacks (e.g., man-in-the-middle) during active sessions [104].

5. Validation of Results

5.1. Experiments Settings

To evaluate the performance of the resources of each SPIDEP variant incorporating SOA and MSA (i.e., networks and infrastructure), spike testing was used to obtain quantitative values regarding the behavior of the services and their interaction with end users [53,89,105,106].

Before performing these controlled tests, it was necessary to create two Kubernetes environments (K8s). The first environment, named SPIDEP-SOA-K8s, contains all the services of the SPIDEP SOA variant platform, with the following specifications: a custom instance of an Ubuntu Server 18.04 LTS (2 Core, 4 GB RAM and 80 GB SSD), PostgreSQL cluster (database dedicated server only) and a PODs replica. The second environment, named SPIDEP-MSA-K8s, contains all the services of the SPIDEP MSA variant platform, with the following specifications: each microservice has a custom instance of an Ubuntu Server 18.04 LTS (2 Core, 4 GB RAM and 80 GB SSD), PostgreSQL cluster (database dedicated server only) and no PODs replica. Additionally, each environment uses Apache as an endpoint and Nginx as a load balancer. The various user requests are received initially by Nginx, which sends the request to the corresponding services to address the URI-request [21].

After deployment of the K8 environments, the use of scripts with random variables developed in Apache JMeter (5.2) is required to measure and contrast the performance among the 50 virtual users (simulating the terminals of the biometric sensors) and the SPIDEP platforms; however, to avoid altering the results, it is necessary to use dedicated servers to perform these tests. Therefore, we have selected the BlazeMeter servers (US East [Virginia, Google]) to generate a heavy workload towards the platforms, according to small or medium instances of medical telemonitoring environments and their specifications (approximately 200,000 queries every 20 min) [7,21,53,60,93]; however, these platforms have the capacity of automatic scaling that will determine the distribution of computer resources according to demand (for these tests they were disabled, since we had medium instances), as shown in Figure 3.

It should be noted that these experiments were evaluated and agreed upon by the experts within the project to be implemented in custom instance of an Ubuntu Server 18.04 LTS [21], as they were previously used to perform load and performance tests, all in order to ensure the quality of the services developed (i.e., performance, scalability and availability), and in conjunction with the use of recent technologies such as: (i) Docker, all platform services have been encapsulated as Docker images to enable their simple management, update and deployment processes within a Cloud environment. In addition, the Docker implementation can be based on individual containers or grouped into a combination of elements that form an overall service [28,29]; (ii) Kubernetes, this technology is used for the orchestration of Docker containers, as it allows the initialization and scaling of container-based jobs, the exposure of services, as well as the rescheduling of failed jobs and long running services [90,91].

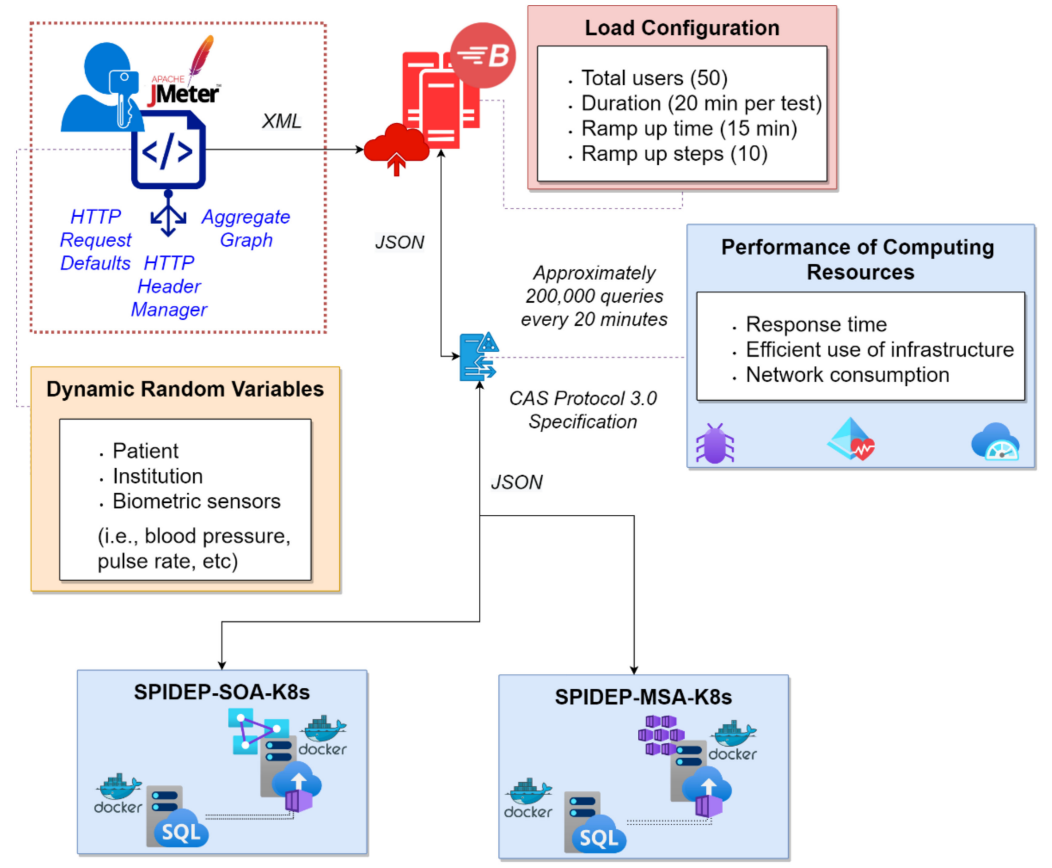

Figure 3. Process Flowchart for the simulation performed on the K8s SPIDEP environments.

However, all this set of technologies allows its adaptation and implementation in a simpler way by developers, which results in consistent, measurable, and replicable results. However, this cloud computing-based framework is intended to be implemented in different Linux distributions (e.g., Fedora, CentOS, Amazon Linux, Debian, and others) or Windows Server (e.g., 2016 or 2019). e.g., 2016 or 2019), since this framework is governed by the segmentation of an organization's logic into a series of separate services that run as independent and isolated processes; that is, it is not necessary to use the same languages, OS, database, or development platforms; however, to replicate these results all the specifications implemented in each variant must be taken into account, for more details please refer to Tables 1 and 2 in Section 4.

Other considerations to take into account would be: (i) type of input and output, for both platforms the RESTful protocol is used to communicate and HTTP methods (GET, POST, DELETE and PATCH). However, the data sent or received are delivered in JSON (Javascript Object Notation) format, since this format is easier to be interpreted by developers or electronic platforms. However, these platforms support other formats like HTML, YAML, XML, XLSX or TXT; (ii) general accessibility, the application was deployed in a cloud environment that can be accessed through a browser (i.e., Google Chrome, Fire-fox, Opera or others); (iii) number of tests performed, four joint tests were created divided into two concepts (Response time and Network consumption, and Efficient use of infrastructure) with an average of 200,000 samples per test for 20 min, grouped in two groups (SPIDEP-SOA-K8s-T1/T2 and SPIDEP-MSA-K8s-T1/T2).

It should be noted that both environments follow the same steps to summon their services. The first step is to authenticate all active sessions using the JWT token that will be verified by the API gateway (all user interface "UI" use is excluded). The second step is to generate approximately 150,000 initial data points in each database, using as seed the existing medical data (8500 records). In this sense, all the data generated is only intended

to generate a workload that will drastically increase (ten of ten users every five minutes) to emulate a specific number of concurrent user sessions (50 virtual users); it will not be used to train, validate or test any expert system. The third step is to randomly execute an HTTP command (e.g., GET, POST, PATCH and DELETE) in the specified service. We have chosen the two most demanding services (approximately 75,000 consultations per session) for both platforms (medical data management (service E) and intervention management (service G)). Services H and I are excluded because of their incompatibility in SPIDEP-SOA; they are in the experimental phase of being integrated with SPIDEP-MSA. The fourth step is to monitor and record all the queries made in each test.

5.2. Evaluations and Results

To evaluate these experiments in SPIDEP, a total of 800,000 queries were sent for a period of 80 min (approximately 200,000 queries every 20 min). It should be noted that these values are obtained from the experiments performed between the two variants of SPIDEP, for more details please refer to Table 3. Another factor to consider is that 50 users are simulated requesting a set of random data simultaneously. Now, the response times of the instances are acceptable and are sufficient for this number of users; however, it must be kept in mind that this quality degrades as the number of users increases, if not foreseen with an adequate scalability of the infrastructure vs the demand [21].

The results were collected through a tabular output report generated after each load test, corresponding to its environment, as shown in Table 3. For the tables, the following labels are used: SPIDEP-SOA-K8s-T1 is a K8 environment that hosts all the services of the SPIDEP-SOA platform, this test will run multiple HTTP queries to a single service (service E); SPIDEP-MSA-K8s-T1 is a K8s environment that hosts all the services of the SPIDEP-MSA platform, this test will run multiple HTTP queries to a single service (service E); SPIDEP-SOA-K8s-T2 is a K8 environment that hosts all the services of the SPIDEP-SOA platform, this test will run multiple HTTP queries to two services simultaneously (services E and G); and SPIDEP-MSA-K8s-T2 is a K8 environment that hosts all the services of the SPIDEP-MSA platform, this test will run multiple HTTP queries to two services simultaneously (services E and G).

The report contains several important values divided into four labels reflecting the type of environment executed, including the type of HTTP method executed, the number of samples, average response time, number of requests that are processed (i.e., hits per seconds), 90th percentile, 95th percentile, 99th percentile, number and percentage of failed requests, the average latency time and the data consumption transferred between the user and the service.

Additionally, several stress simulation tests in the infrastructure that host both variants of SPIDEP were performed, whose functions are to detect any problems that may arise in the platform. Reviewing CPU performance, memory, network I/O and connections (e.g., engine health to BlazeMeter) indicates whether the SPIDEP infrastructure itself is capable of supporting the demand-related bottlenecks or errors that appear [21,53], as shown in Figures 4–7, with one figure for each environment.

Table 3. General results of the simultaneous queries made to the RESTful methods of the SPIDEP application, under the SOA and MSA architectural styles, invoking single and dual services (i.e., medical data management (service E) and intervention management (service G).

Labels	HTTP Methods		Samples	Avg. Response Time *	Avg. Hits/s	90% Line *	95% Line *	99% Line *	Error Count and Percentage	Avg. LATENCY *	Avg. Bytes (Kbytes/s)
SPIDEP-SOA-K8s-T1		GET	25,245	395.2	21.04	607	631	711	0 (0%)	395.19	12.25
		POST	25,237	401.26	21.03	615	639	719	0 (0%)	401.25	4.87
		PATCH	25,226	388.84	21.02	603	627	707	0 (0%)	388.83	4.87
		DELETE	25,217	390.3	21.01	603	627	723	0 (0%)	390.29	4.86
SPIDEP-MSA-K8s-T1		GET	27,838	344.38	23.2	543	579	663	0 (0%)	344.34	22.99
		POST	27,830	363.9	23.21	567	599	691	0 (0%)	363.87	9.74
		PATCH	27,813	362.23	23.20	571	603	683	0 (0%)	362.19	9.98
		DELETE	27,805	358.68	23.19	563	595	683	0 (0%)	358.65	9.98
SPIDEP-SOA-K8s-T2	Intervention Management Medical Data Management	GET	50,511	390.89	42.09	603	631	715	0 (0%)	390.88	16.2
		PATCH	50,346	387.13	41.95	599	627	711	0 (0%)	387.12	9.65
		POST	50,487	390.39	42.01	615	639	719	0 (0%)	396.39	9.74
		DELETE	50,947	390.12	42.45	603	627	695	0 (0%)	390.11	9.79
SPIDEP-MSA-K8s-T2	Intervention Management Medical Data Management	GET	102,009	389.72	85.01	599	639	715	0 (0%)	335.65	57.68
		PATCH	108,978	364.84	90.82	579	623	699	0 (0%)	364.84	36.35
		POST	118,534	335.37	98.78	535	571	639	0 (0%)	384.02	42.51
		DELETE	112,221	354.22	93.52	555	595	683	0 (0%)	354.21	36.63

* Time measurements are averages in ms.

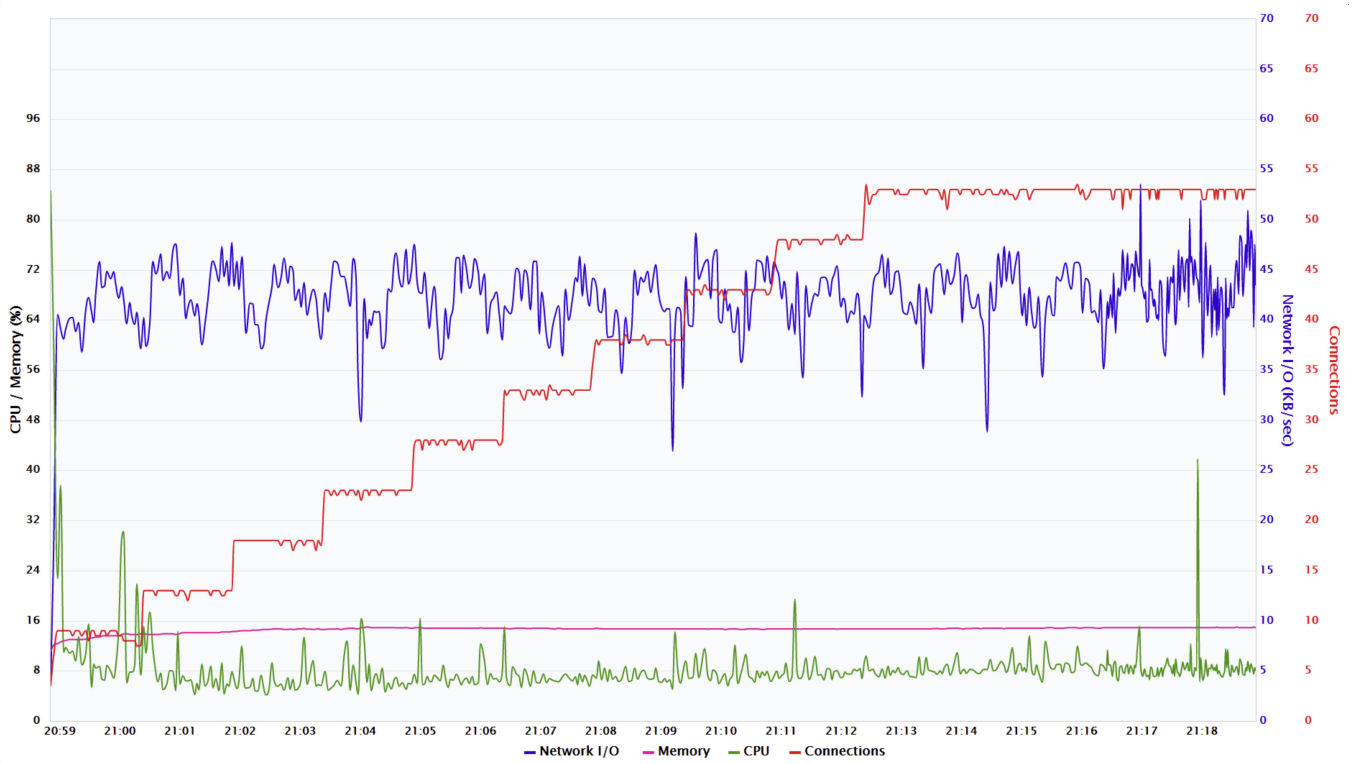

Figure 4. Engine Health Report SPIDEP-SOA-K8s-T1. The colored lines represent various parameters as follows: the blue line is the bandwidth traffic; the purple line is the memory load generated by the users; the green line is the CPU load generated by the users; and the red line is the active connections within the test.

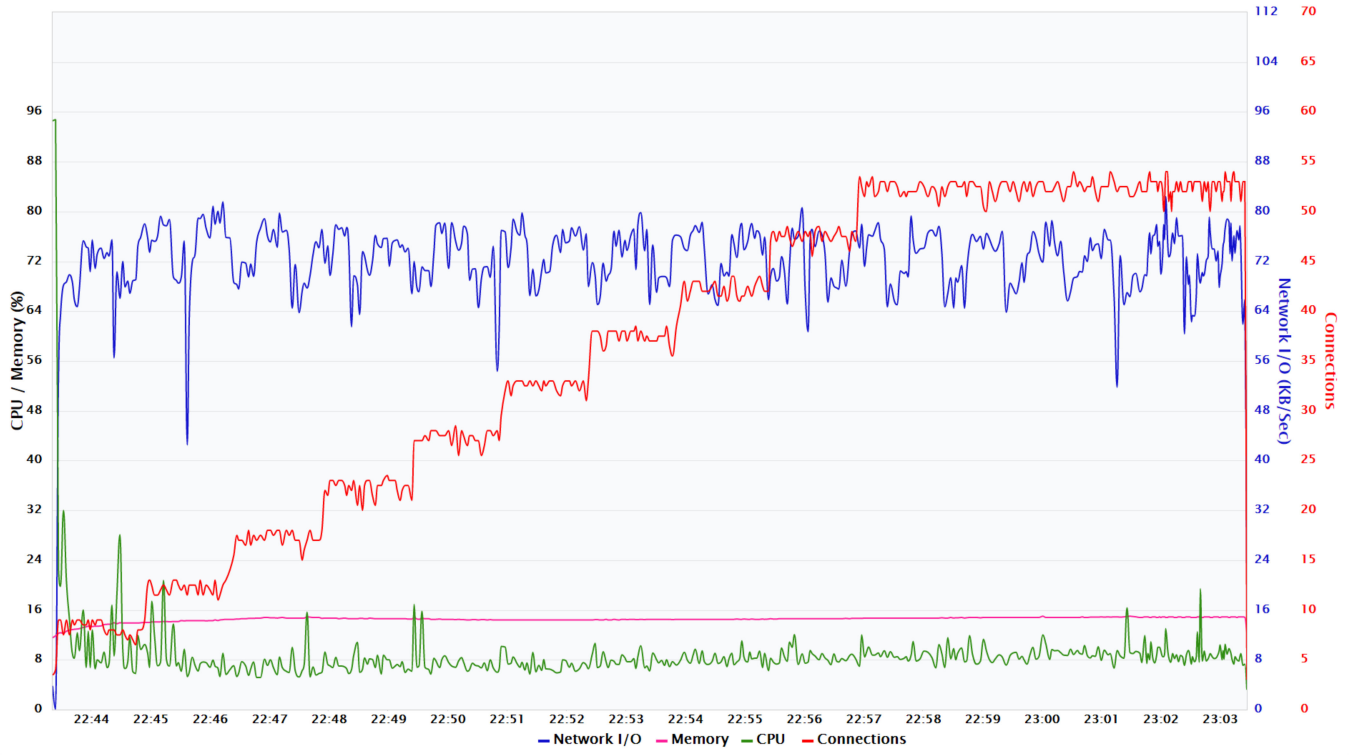

Figure 5. Engine Health Report SPIDEP-MSA-K8s-T1. The colored lines represent various parameters as follows: the blue line is the bandwidth traffic; the purple line is the memory load generated by the users; the green line is the CPU load generated by the users; and the red line is the active connections within the test.

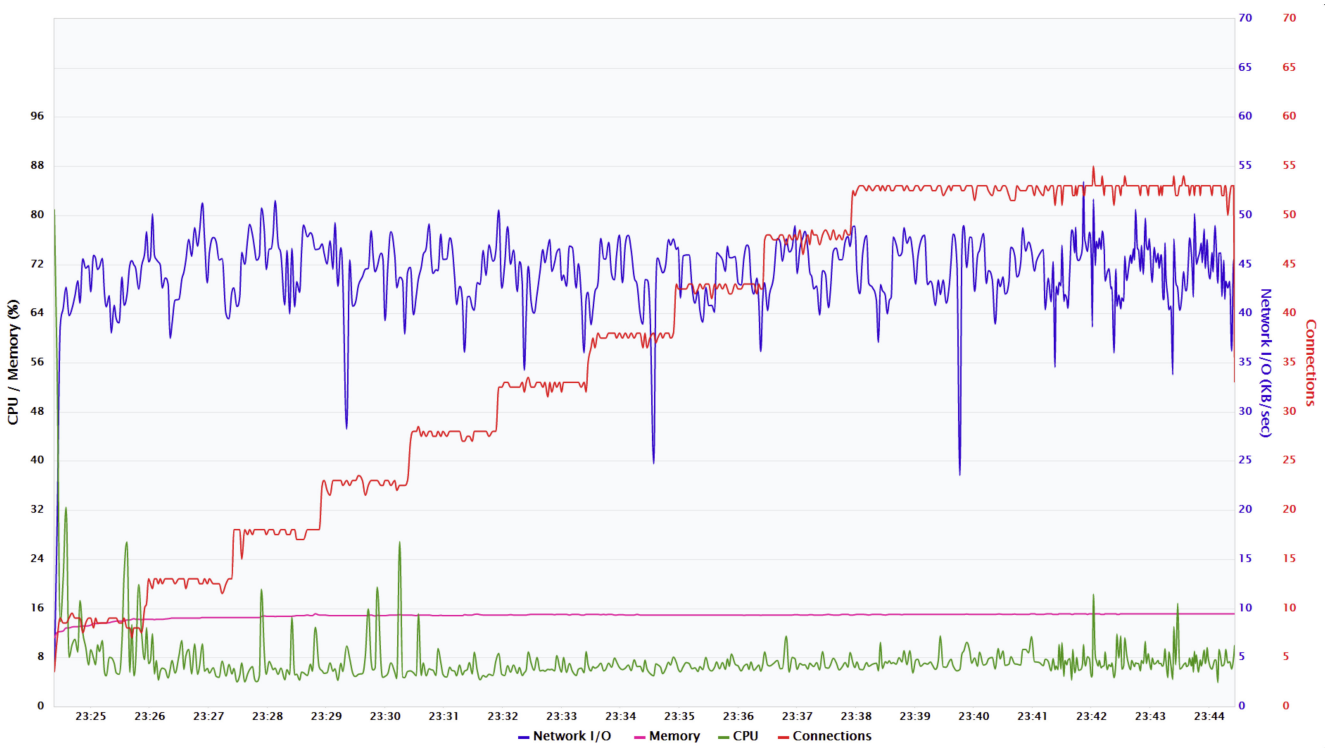

Figure 6. Engine Health Report SPIDEP-SOA-K8s-T2. The colored lines represent various parameters as follows: the blue line is the bandwidth traffic; the purple line is the memory load generated by the users; the green line is the CPU load generated by the users; and the red line is the active connections within the test.

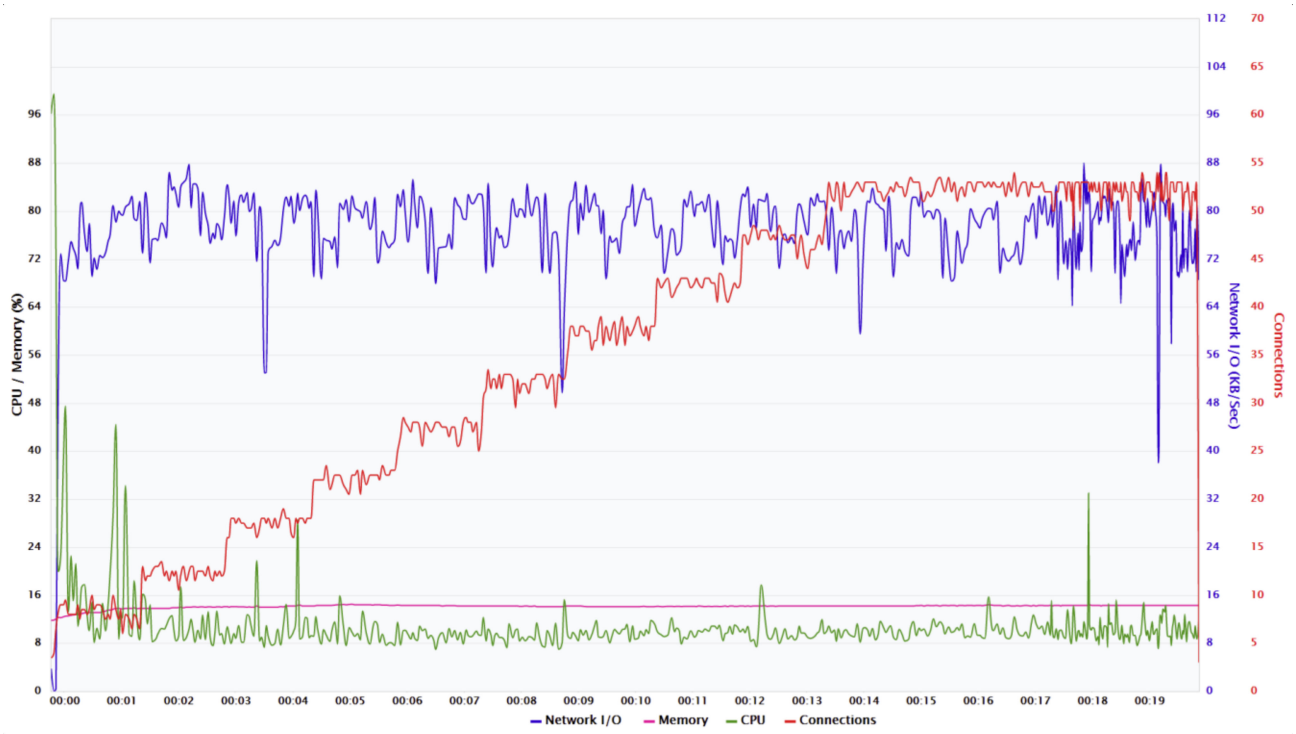

Figure 7. Engine Health Report SPIDEP-MSA-K8s-T2. The colored lines represent various parameters as follows: the blue line is the bandwidth traffic; the purple line is the memory load generated by the users; the green line is the CPU load generated by the users; and the red line is the active connections within the test.

6. Discussion

The purpose of this study is to evaluate and contrast the performance of computing resources (i.e., response time, efficient use of infrastructure and network consumption) used by the different services for each SPIDEP variant, based on the concept of versatile high-performance (acceptance criteria) and the quantitative values obtained from the various tests (performance testing). Another purpose is to offer a software solution focused on the strengths and weaknesses of each variant (SOA and MSA) to provide customized eHealth functionalities and host functionalities based on AI algorithms (e.g., recommender system based on deep learning), according to the storage of adequate data for remote medical care scenarios (telemonitoring) and the demand and the size of the instances of the medical organizations (e.g., clinics, nursing homes or hospitals).

There are several significant findings from this study. First, it is important to consider the acceptance criteria such as the average response time of the queries, the 90th percentile, and their relationship with the percentage of failed requests. Therefore, according to various investigations [107–110], it was established that the acceptable limit (criterion A) for the average response times of the queries and the 90th Percentile is 5000 ms (five seconds) and the acceptable limit (criterion B) for the percentage of failed requests is 5%. Any test with results beyond the established limits is considered to have unacceptable performance [21].

Considering the above, from Table 3 we can establish that the tests with 50 concurrent users are acceptable, since both variants meet the two criteria defined; however, we have an interesting case in the SPIDEP-SOA-K8s-T2 and SPIDEP-MSA-K8s-T2 environments, specifically in the parameters of total number of queries, average response time and network consumption. As shown, in the SPIDEP-SOA-K8s-T2 environment, there is an average of 50,572 queries made (42.14 avg. hits/s), while SPIDEP-MSA-K8s-T2 has an average number of 110,435 queries made (92.02 avg. hits/s). This remarkable difference is due to the following two factors:

- Architectural style: The attributes of MSA focus more on supporting the streamlining and reduction of microservice deliveries in the shortest possible time (all microservice must be lightweight, decoupled, isolated and independent of any programming language, libraries or databases). Conversely, SOA makes use of the ESB or central entry point, which are not considered agile enough [33] because the interactions between the services are interdependent; consequently, the overall performance of the system is affected for each service invoked simultaneously during a high demand under the SOA pattern [75]. This pattern cannot be changed since all the services developed remain loosely coupled and any change requires the rebuilding or reimplementation of the entire application (coarse-grained services) [62,79–81]; therefore, the MSA variant is approximately 54.21% more efficient than the SOA variant in terms of total query numbers and average response times.
- API gateway: It is important to consider that SOA has a centralized governance of services, which also affects the deliveries and responses of queries to users [33]. We have seen how services develop without considering the weak points of this technology, e.g., not considering the centralization of data (without instances) or not considering a Cloud infrastructure for the operation of the services. The resulting increased traffic of the API of HTTP resources degrades the overall performance of the application until its collapse [111,112]. However, for MSA, applying decentralized and independent governance requires applying additional security measures unlike SOA (e.g., authenticating the user identity for each message sent and/or received through the signal mechanisms, the use of custom encryption PBKDF2 or the use of the ticket-based protocol CAS 3.0). These additional mechanisms are intended to establish a secure communication channel between the different microservices (internal and external) of computer attacks (e.g., man-in-the-middle, DoS or other) [104]. Consequently, it brings with it an increase in the general network traffic between the microservices and their infrastructure, demonstrating a significant increase in the average transfer of

data to the users; therefore, the SOA variant is approximately 73.80% more efficient than the MSA variant in terms of network consumption.

The SPIDEP-SOA-K8s-T1 and SPIDEP-MSA-K8s-T1 environments do not have a noticeable difference in the results compared to the K8s-T2 environments, but their trend is similar to the previous environment (approximately 9.31% and approximately 49.05% for each case) because this T1 environment executes multiple HTTP queries to a single service (service E), while the T2 environments execute multiple HTTP queries to two services simultaneously (services E and G).

Regarding the second finding, the four engine health reports of the K8s-T1/T2 environments meet the infrastructure performance (CPU and RAM) under a demand of 50 concurrent users (connections), since the CPU values are lower than 80% (8% to approximately 30%) and the memory levels are lower than 70% (10% to approximately 15%) [113,114]; however, the network I/O value of the SPIDEP-MSA-K8s-T1/T2 environments shows a significant consumption in the bandwidth traffic between the services and the Dockers instances in comparison with the SPIDEP-SOA-K8s-T1/T2 environments. This high consumption is because the MSA variant requires managing more specialized coordination to redirect queries from external platforms to internal microservices (or vice versa) within the Cloud environment [115]. In addition, it must actively route and validate user requests through the API gateway to the respective instances of the microservices [74].

In the third finding, to determine if the SPIDEP variants meet the software attribute (i.e., versatile high-performance), we have analyzed and contrasted the results obtained from the various stress tests of each platform. Considering the above, the MSA variant is the most appropriate for our needs, since it optimally meets the following three important aspects: (i) Scalability—MSA can be scaled individually when running a heavy workload by replicating the microservices on several containers and not replicating those that are underutilized (i.e., maximize the performance with minimal cost); therefore, microservices can handle the increase in demand without latency being significantly degraded, but this requires consuming a significant bandwidth to mitigate the latency [110,114,116]; (ii) Versatility—MSA allows adding and integrating new functionalities to the platform without affecting the availability of the other microservices. Therefore, each team has the autonomy to decide which is the best technology for the development of the service, according to the needs identified [7]; and (iii) Performance—a platform geared towards high demand requires that all its computing resources be distributed in the Cloud, which allows a shorter response time for the services; however, the platform becomes more demanding with the computational load of the instances.

7. Conclusions and Future Work

In this article, we have presented all the steps involved in the design, implementation and deployment of the SPIDEP platform and its RC variants based on the SOA and MSA architectural patterns; the platform is focused on remote telemonitoring of the elderly. Therefore, our purpose is to offer a replicable framework to support the early diagnosis of infectious diseases and their derivatives by using recent ICT developments through artificial intelligence algorithms (e.g., deep learning and machine learning-based inference systems), with the added value of reducing logistical costs for medical institutions.

Therefore, it was necessary to evaluate which of the SPIDEP variants are best suited for our versatile high-performance concept, considering the strengths and weaknesses of each architecture and how they behave in a high demand scenario. As a result, we have analyzed and contrasted the performance of each variant vs. the metrics obtained from the various performance tests (i.e., response time, efficient use of infrastructure and network consumption), which found that MSA is a better performer in terms of the performance quality attribute (approximately 54.21%). In the same manner, when processing multiple requests for various services, the response time was lower compared to SOA (approximately 7.34%), but the bandwidth consumption in MSA was more significant than SOA (approximately 73.80%).

Based on the ideas presented, we feel that the implementation of SOA and MSA depends on the capabilities and needs of organizations (e.g., performance, flexibility, availability, interoperability or other characteristics in return for known and unknown consequences) [33,63]. Therefore, within the eHealth context and based on the results obtained, we observe that MSA is capable of meeting the majority of needs in support of medical decision making and is adaptable to different types of clinical systems (e.g., EHR, IoHT or telemonitoring systems) and various infrastructure solutions (e.g., nursing homes, hospitals and public health management) [16,62,117]; however, this brings with it many challenges [51,74,94,99,105].

The SOA and MSA architectural patterns can be considered complementary allies for an interenterprise or interbusiness architecture that confers a suite of different services, rather than being competitors [33,118,119], i.e., combining the SOA and MSA attributes within an environment. Therefore, we plan to work on a proposal for an intergenerational ecosystem for SOA-MSA, with the aim of developing a basis for the integration and interconnection of the different eHealth applications involved in medical organizations in conjunction with microservices adapted to machine learning-based inference systems to perform specialized tasks in decision support in the prevention, monitoring and treatment of various conditions. Additionally, we are exploring the possibility of extending our framework to other eHealth areas (e.g., a patient monitoring system for hemodialysis) [120,121], early prediction of COVID-19 [122–125] or prediction of heart and kidney risks in diabetic patients [126–128]).

Another possible area for future research would be the adaptation of this proposal to other sectors of industry 4.0 (e.g., smart buildings or tourism) [8,129–132]; however, additional research is needed to obtain sufficient results to demonstrate the robustness of the architecture in terms of the adaptations of the attributes for these sectors. Our findings will be published in the near future.

Author Contributions: Conceptualization, J.M.G.-P. and M.V.-L.; methodology, H.C.-G., L.M.-P., J.M.G.-P., and M.V.-L.; software, H.C.-G., L.M.-P. and M.V.-L.; validation, H.C.-G., D.R.-P., G.S., M.-L.P.-L., and J.M.G.-P.; formal analysis, H.C.-G., J.M.G.-P. and M.-L.P.-L.; investigation, H.C.-G., L.M.-P., M.V.-L., J.M.G.-P., D.R.-P., G.S., and M.-L.P.-L.; resources, J.M.G.-P., M.-L.P.-L., G.S. and D.R.-P.; data curation, M.-L.P.-L. and G.S.; writing—original draft preparation, H.C.-G., L.M.-P., J.M.G.-P., and M.V.-L.; writing—review and editing, H.C.-G., L.M.-P., M.V.-L. and J.M.G.-P.; visualization, H.C.-G., and L.M.-P.; supervision, M.V.-L. and J.M.G.-P.; project administration, J.M.G.-P.; funding acquisition, J.M.G.-P. and M.V.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This study forms part of the AC16/00061 project entitled "Design and implementation of a low-cost intelligent system for prediagnosis and telecare of infectious diseases in older people (SPIDEP)", which was financed by resources from the Instituto de Salud Carlos III of the Ministry of Science, Innovation of Spain, the Foundation for Biomedical Research of the Prince of Asturias University Hospital (FUHUPA), the National Secretariat of Science, Technology and Innovation of Panama (SENACYT-PANAMA) through the National Research System (SNI-PANAMA), and international scholarships (doctoral studies). In addition, this study forms part of the 2nd Joint Call for Research and Innovation (ERANet-LAC) of the Seventh Framework Program for Research and Technological Development (FP7) of the European Union under contract number ELAC2015/T09-0819 SPIDEP.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Exclude this statement if the study did not report any data.

Acknowledgments: We would like to extend special thanks to all the clinical and healthcare staff of the nursing homes for the elderly, the Francisco de Vitoria and Cisneros Centres of the Department of Social Policies and Family of the Community of Madrid in Spain and Nuestra Señora Carmen and Carls George in the Dominican Republic.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Van Eyk, E.; Toader, L.; Talluri, S.; Versluis, L.; Uta, A.; Iosup, A. Serverless is More: From PaaS to Present Cloud Computing. *IEEE Internet Comput.* **2018**, *22*, 8–17. [CrossRef]
2. Pfandzelter, T.; Bermbach, D. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. In Proceedings of the 2020 IEEE International Conference on Fog Computing (ICFC), Sydney, Australia, 21–24 April 2020; pp. 17–24.
3. Gadepalli, P.K.; Peach, G.; Cherkasova, L.; Aitken, R.; Parmer, G. Challenges and Opportunities for Efficient Serverless Computing at the Edge. In Proceedings of the 2019 38th Symposium on Reliable Distributed Systems (SRDS), Lyon, France, 1–4 October 2019; pp. 261–2615.
4. Sewak, M.; Singh, S. Winning in the Era of Serverless Computing and Function as a Service. In Proceedings of the 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 6–8 April 2018; pp. 1–5.
5. Kim, Y.; Cha, G. Design of the Cost Effective Execution Worker Scheduling Algorithm for FaaS Platform Using Two-Step Allocation and Dynamic Scaling. In Proceedings of the 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2), Paris, France, 18–21 November 2018; pp. 131–134. [CrossRef]
6. Khalique, F.; Khan, S.A.; Nosheen, I. A Framework for Public Health Monitoring, Analytics and Research. *IEEE Access* **2019**, *7*, 101309–101326. [CrossRef]
7. Suryotrisongko, H.; Jayanto, D.P.; Tjahyanto, A. Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot. *Procedia Comput. Sci.* **2017**, *124*, 736–743. [CrossRef]
8. Artemenko, O.; Pasichnyk, V.; Kunanec, N.; Tabachyshyn, D. Using context analysys for providing real time recommendations in e-tourism mobile location-based recommender systems. In Proceedings of the 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 17–20 September 2019; Volume 3, pp. 166–169.
9. Tsai, Y.-T.; Fan, C.-L.; Lo, C.-L.; Huang, S.-H. SmartLohas: A Smart Assistive System for Elder People. In Proceedings of the 2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC, Exeter, UK, 21–23 June 2017; pp. 369–374.
10. Mendes, D.; Jorge, D.; Pires, G.; Panda, R.; Antonio, R.; Dias, P.; Oliveira, L. VITASENIOR-MT: A distributed and scalable cloud-based telehealth solution. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 767–772. [CrossRef]
11. Kontis, V.; Bennett, J.E.; Mathers, C.D.; Li, G.; Foreman, K.; Ezzati, M. Future life expectancy in 35 industrialised countries: Projections with a Bayesian model ensemble. *Lancet* **2017**, *389*, 1323–1335. [CrossRef]
12. Wang, Z.; Saho, K.; Tomiyama, H.; Meng, L. Gender Classification of Elderly People using Doppler Radar Images based on Machine Learning. In Proceedings of the 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), Kusatsu, Japan, 26–28 August 2019; pp. 305–310.
13. Rizvi, S.; Sohail, I.; Saleem, M.M.; Irtaza, A.; Zafar, M.; Syed, M. A Smart Home Appliances Power Management System for Handicapped, Elder and Blind People. In Proceedings of the 2018 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 13–14 August 2018; pp. 1–4.
14. Kulik, C.T.; Ryan, S.; Harper, S.; George, G. Aging Populations and Management. *Acad. Manag. J.* **2014**, *57*, 929–935. [CrossRef]
15. Solé-Casals, M.; Chirveches-Pérez, E.; Puigoriol-Juvanteny, E.; Nubó-Puntí, N.; Chabrera-Sanz, C.; Subirana-Casacuberta, M. Perfil y resultados del paciente frágil valorado por la Enfermera de Práctica Avanzada en un servicio de urgencias. *Enfermería Clínica* **2018**, *28*, 365–374. [CrossRef] [PubMed]
16. Ianculescu, M.; Alexandru, A.; Neagu, G.; Pop, F. Microservice-Based Approach to Enforce an IoHT Oriented Architecture. In Proceedings of the 2019 E-Health and Bioengineering Conference (EHB), Iasi, Romania, 21–23 November 2019; pp. 1–4.
17. Jarwar, M.A.; Ali, S.; Chong, I. Exploring Web Objects enabled Data-Driven Microservices for E-Health Service Provision in IoT Environment. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 17–19 October 2018; pp. 112–117.
18. Kocak, S.; Artug, T.; Tulum, G. A Preliminary Study for Remote Healthcare System: Activity Classification for Elder People with on Body Sensors. In Proceedings of the 2018 6th International Conference on Control Engineering & Information Technology (CEIT), Istanbul, Turkey, 25–27 October 2018; pp. 1–3.
19. Gaddam, A.; Mukhopadhyay, S.C.; Gupta, G.S. Necessity of a bed-sensor in a smart digital home to care for elder-people. In Proceedings of the 2008 IEEE Sensors, Lecce, Italy, 26–29 October 2008; pp. 1340–1343.
20. Wishner, L.; Sollevad, J.; Rudowitz, P.; Paradise, R.; Antonisse, J. A Look at Rural Hospital Closures and Implications for Access to Care: Three Case Studies. 2016. Available online: https://www.kff.org/report-section/a-look-at-rural-hospital-closures-and-implications-for-access-to-care-three-case-studies-issue-brief/ (accessed on 12 January 2021).
21. Calderon-Gomez, H.; Mendoza-Pitti, L.; Vargas-Lombardo, M.; Gomez-Pulido, J.M.; Castillo-Sequera, J.L.; Sanz-Moreno, J.; Sencion, G. Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture. *IEEE Access* **2020**, *8*, 118340–118354. [CrossRef]
22. Sanz-Moreno, J.; Gómez-Pulido, J.; Garcés, A.; Calderón-Gómez, H.; Vargas-Lombardo, M.; Castillo-Sequera, J.L.; Luque, M.L.P.; Toro, R.; Sención-Martínez, G. mHealth System for the Early Detection of Infectious Diseases Using Biomedical Signals. In *The Importance of New Technologies and Entrepreneurship in Business Development: In The Context of Economic Diversity in Developing Countries*; Metzler, J.B., Ed.; Springer: Berlin/Heidelberg, Germany, 2020; pp. 203–213.

23. Calderon-Gomez, H.; Garces-Jimenez, A.; Vargas-Lombardo, M.; Gomez-Pulido, J.M.; Polo-Luque, M.-L.; Castillo, J.; Sencion, G.; Moreno, J.S. Proposal Using the Cloud Architecture in System for the Early Detection of Infectious Diseases in Elderly People Fed by Biosensors Records. In Proceedings of the 2019 7th International Engineering, Sciences and Technology Conference (IESTEC), Panama, Panama, 9–11 October 2019; pp. 631–634.
24. Calderón-Gómez, H.; Navarro-Marín, F.; Gómez-Pulido, J.M.; Castillo-Sequera, J.L.; Garcés-Jiménez, A.; Polo-Luque, M.-L.; Sanz-Moreno, J.; Sención, G.; Vargas-Lombardo, M. Development of ehealth applications-based on microservices in a cloud architecture. *Rev. Iber. Sist. Tecnol. Inf.* **2019**, *2019*, 81–93.
25. Baldominos, A.; Ogul, H.; Colomo-Palacios, R.; Sanz-Moreno, J.; Gómez-Pulido, J.M. Infection prediction using physiological and social data in social environments. *Inf. Process. Manag.* **2020**, *57*, 102213. [CrossRef]
26. Aazam, M.; Zeadally, S.; Harras, K.A. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4674–4682. [CrossRef]
27. Patel, A.R.; Azadi, S.; Babaee, M.H.; Mollaei, N.; Patel, K.L.; Mehta, D.R. Significance of Robotics in Manufacturing, Energy, Goods and Transport Sector in Internet of Things (IoT) Paradigm. In Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018; pp. 1–4.
28. Kousiouris, G.; Tsarsitalidis, S.; Psomakelis, E.; Koloniaris, S.; Bardaki, C.; Tserpes, K.; Nikolaidou, M.; Anagnostopoulos, D. A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management. *ICT Express* **2019**, *5*, 141–145. [CrossRef]
29. Baresi, L.; Mendonca, D.F. Towards a Serverless Platform for Edge Computing. In Proceedings of the 2019 IEEE International Conference on Fog Computing (ICFC), Prague, Czech Republic, 24–26 June 2019; pp. 1–10.
30. Da Silva, M.A.P.; Times, V.C.; De Araujo, A.M.C.; Da Silva, P.C. A Microservice-Based Approach for Increasing Software Reusability in Health Applications. In Proceedings of the 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 3–7 November 2019; pp. 1–8.
31. Castrillón-Betancur, J.C.; Flórez-Arango, J.F. Archetypes, terminologies and semantic interoperability in health. Arquetipos, terminologías e interoperabilidad semántica en salud. *Rev. Cuba. Investig. Biomed.* **2015**, *34*, 365–377. Available online: http://ref.scielo.org/kp9gdp (accessed on 14 January 2021).
32. Chaim, R.M.; Oliveira, E.C.; Araujo, A.P.F. Technical specifications of a service-oriented architecture for semantic interoperability of HER—Electronic health records. In Proceedings of the 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), Lisbon, Portugal, 21–24 June 2017; pp. 1–6.
33. Andriyanto, A.; Doss, R.; Yustianto, P. Adopting SOA and Microservices for Inter-enterprise Architecture in SME Communities. In Proceedings of the 2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE), Denpasar, Indonesia, 3–4 October 2019; Volume 6, pp. 282–287.
34. Hutapea, R.C.A.; Wahyudi, A.P. Design Quality Measurement for Service Oriented Software on Service Computing System: A Systematic Literature Review. In Proceedings of the 2018 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, 22–26 October 2018; pp. 375–380.
35. Bouguettaya, A.; Singh, M.; Huhns, M.; Sheng, Q.Z.; Dong, H.; Yu, Q.; Neiat, A.G.; Mistry, S.; Benatallah, B.; Medjahed, B.; et al. A service computing manifesto. *Commun. ACM* **2017**, *60*, 64–72. [CrossRef]
36. Mandal, A.K.; Sarkar, A. Service Oriented System design: Domain Specific Model based approach. In Proceedings of the 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 15–17 August 2016; pp. 489–494.
37. Gunawan, L.K.; Tama, N.S.; Nurjannah, R.; Fauziyah, U.; Fajar, A.N.; Qomariyah, N.N. WELTY System Design Based on Service Oriented Architecture for Smart HealthCare. In Proceedings of the 2019 International Conference on ICT for Smart Society (ICISS), Bandung, Indonesia, 19–20 November 2019; Volume 7, pp. 1–4.
38. Gavrilov, G.; Vlahu-Gjorgievska, E.; Trajkovik, V. Healthcare data warehouse system supporting cross-border interoperability. *Health Inform. J.* **2020**, *26*, 1321–1332. [CrossRef]
39. Amin, M.M.; Sutrisman, A.; Stiawan, D.; Ermatita, E.; Alzahrani, M.Y.; Budiarto, R. Interoperability framework for integrated e-health services. *Bull. Electr. Eng. Inform.* **2020**, *9*, 354–361. [CrossRef]
40. Chronaki, C.; Estelrich, A.; Cangioli, G.; Melgara, M.; Kalra, D.; Gonzaga, Z.; Garber, L.; Blechman, E.; Ferguson, J.; Kay, S. Interoperability standards enabling cross-border patient summary exchange. *Stud. Heal. Technol. Inform.* **2014**, *205*, 256–260.
41. European Commission. eHealth Action Plan 2012–2020—Innovative Healthcare for the 21st Century. 2012. Available online: https://ec.europa.eu/health//sites/health/files/ehealth/docs/com_2012_736_en.pdf (accessed on 16 January 2021).
42. Chang, S.H. A Systematic Analysis and Design Approach to Develop Adaptable Services in Service Oriented Computing. In Proceedings of the 2007 IEEE Congress on Services (Services 2007), Salt Lake City, UT, USA, 9–13 July 2007; pp. 375–378.
43. Park, J.; Moon, M.; Yeom, K. The BCD View Model: Business Analysis View, Service Composition View and Service Design View for Service Oriented Software Design and Development. In Proceedings of the 2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, Kunming, China, 21–23 October 2008; pp. 37–43. [CrossRef]
44. Kufner, J.; Marik, R. Restful State Machines and SQL Database. *IEEE Access* **2019**, *7*, 144603–144617. [CrossRef]
45. Hameed, R.T.; Mohamad, O.A.; Hamid, O.T.; Tapus, N. Patient monitoring system based on e-health sensors and web services. In Proceedings of the 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, Romania, 30 June–2 July 2016; pp. 1–6.

46. Srinivasan, C.; Charan, G.; Babu, P.C.S. An IoT based SMART patient health monitoring system. *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *18*, 1657–1664. [CrossRef]
47. Peng, C.; Goswami, P.; Bai, G. Linking Health Web Services as Resource Graph by Semantic REST Resource Tagging. *Procedia Comput. Sci.* **2018**, *141*, 319–326. [CrossRef]
48. Silva, B.M.C.; Rodrigues, J.J.P.C.; Ramos, A.; Saleem, K.; De La Torre, I.; Rabelo, R.L. A Mobile Health System to Empower Healthcare Services in Remote Regions. In Proceedings of the 2019 IEEE International Conference on E-health Networking, Application & Services (HealthCom), Bogota, Colombia, 14–16 October 2019; pp. 1–6.
49. Vayghan, L.A.; Saied, M.A.; Toeroe, M.; Khendek, F. Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes. In Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), Sofia, Bulgaria, 22–26 July 2019; pp. 176–185.
50. Cojocaru, M.-D.; Uta, A.; Oprescu, A.-M. Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications. In Proceedings of the 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), Amsterdam, The Netherlands, 3–7 June 2019; pp. 84–93.
51. Hassan, S.; Bahsoon, R.; Kazman, R. Microservice transition and its granularity problem: A systematic mapping study. *Softw. Pr. Exp.* **2020**, *50*, 1651–1681. [CrossRef]
52. De Lauretis, L. From Monolithic Architecture to Microservices Architecture. In Proceedings of the 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 27–30 October 2019; pp. 93–96. [CrossRef]
53. Guaman, D.; Yaguachi, L.; Samanta, C.C.; Danilo, J.H.; Soto, F. Performance evaluation in the migration process from a monolithic application to microservices. In Proceedings of the 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, Spain, 13–16 June 2018; pp. 1–8.
54. Akbulut, A.; Perros, H.G. Performance Analysis of Microservice Design Patterns. *IEEE Internet Comput.* **2019**, *23*, 19–27. [CrossRef]
55. Di Francesco, P.; Lago, P.; Malavolta, I. Architecting with microservices: A systematic mapping study. *J. Syst. Softw.* **2019**, *150*, 77–97. [CrossRef]
56. Li, S.; Zhang, H.; Jia, Z.; Li, Z.; Zhang, C.; Li, J.; Gao, Q.; Ge, J.; Shan, Z. A dataflow-driven approach to identifying microservices from monolithic applications. *J. Syst. Softw.* **2019**, *157*, 110380. [CrossRef]
57. Carranza-García, F.; García-Moreno, F.M.; Rodriguez-Dominguez, C.; Garrido, J.L.; Edo, M.B.; Rodriguez-Fortiz, M.J.; Pérez-Mármol, J.M. *Supporting Active Ageing Interventions with Web and Mobile/Wearable Technologies and Using Microservice Oriented Architectures*; Metzler, J.B., Ed.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 1016, pp. 114–123.
58. Garcia-Moreno, F.M.; Bermudez-Edo, M.; Garrido, J.L.; Rodríguez-García, E.; Pérez-Mármol, J.M.; Rodríguez-Fórtiz, M.J. A Microservices e-Health System for Ecological Frailty Assessment using Wearables. *Sensors* **2020**, *20*, 3427. [CrossRef]
59. Sianaki, O.A.; Yousefi, A.; Tabesh, A.R.; Mahdavi, M. Internet of Everything and Machine Learning Applications: Issues and Challenges. In Proceedings of the 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), Krakow, Poland, 16–18 May 2018; pp. 704–708.
60. Houmani, Z.; Balouek-Thomert, D.; Caron, E.; Parashar, M. Enhancing microservices architectures using data-driven service discovery and QoS guarantees. In Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, Australia, 11–14 May 2020; pp. 290–299.
61. Frade, S.; Freire, S.M.; Sundvall, E.; Patriarca-Almeida, J.H.; Cruz-Correia, R. Survey of openEHR storage implementations. In Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, Porto, Portugal, 20–22 June 2013; pp. 303–307.
62. Rademacher, F.; Sachweh, S.; Zundorf, A. Differences between Model-Driven Development of Service-Oriented and Microservice Architecture. In Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 5–7 April 2017; pp. 38–45.
63. Firmansyah, T.; Ganeswangga, A.; Fajar, A.N. Transforming the Monolith E-Procurement Application. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *9*, 4780–4784. [CrossRef]
64. Kazanavicius, J.; Mazeika, D. Migrating Legacy Software to Microservices Architecture. In Proceedings of the 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 25 April 2019; pp. 1–5.
65. Zhang, Y.; Liu, B.; Dai, L.; Chen, K.; Cao, X. Automated Microservice Identification in Legacy Systems with Functional and Non-Functional Metrics. In Proceedings of the 2020 IEEE International Conference on Software Architecture (ICSA), Salvador, Brazil, 16–20 March 2020; pp. 135–145.
66. Selmadji, A.; Seriai, A.-D.; Bouziane, H.L.; Mahamane, R.O.; Zaragoza, P.; Dony, C. From Monolithic Architecture Style to Microservice one Based on a Semi-Automatic Approach. In Proceedings of the 2020 IEEE International Conference on Software Architecture (ICSA), Salvador, Brazil, 16–20 March 2020; pp. 157–168.
67. Fundación para la Investigación Biomédica del Hospital Universitario Príncipe de Asturias. Design and Implementation of a Low Cost Smart System for Pre-Diagnosis and Telecare of Infectious Diseases in Elderly People | EU-CELAC. 2017. Available online: https://www.eucelac-platform.eu/project/design-and-implementation-low-cost-smart-system-pre-diagnosis-and-telecare-infectious (accessed on 11 February 2021).
68. Catarinucci, L.; De Donno, D.; Mainetti, L.; Palano, L.; Patrono, L.; Stefanizzi, M.L.; Tarricone, L. An IoT-Aware Architecture for Smart Healthcare Systems. *IEEE Internet Things J.* **2015**, *2*, 515–526. [CrossRef]

69. Javed, A.; Robert, J.; Heljanko, K.; Främling, K. IoTEF: A Federated Edge-Cloud Architecture for Fault-Tolerant IoT Applications. *J. Grid Comput.* **2020**, *18*, 57–80. [CrossRef]
70. Di Francesco, P. Architecting Microservices. In Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 5–7 April 2017; pp. 224–229.
71. Zhou, S. Improving Collaboration Efficiency in Fork-Based Development. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 1218–1221.
72. Duszynski, S.; Tenev, V.L.; Becker, M. N-way Diff: Set-based Comparison of Software Variants. In Proceedings of the 2020 Working Conference on Software Visualization (VISSOFT), Adelaide, Australia, 28 September–2 October 2020; pp. 72–83.
73. Chua, B. Detecting sustainable programming languages through forking on open source projects for survivability. In Proceedings of the 2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Gaithersburg, MD, USA, 2–5 November 2015; pp. 120–124. [CrossRef]
74. De Sanctis, M.; Muccini, H.; Vaidhyanathan, K. Data-driven Adaptation in Microservice-based IoT Architectures. In Proceedings of the 2020 IEEE International Conference on Software Architecture (ICSA), Salvador, Brazil, 16–20 March 2020; pp. 59–62.
75. Papazoglou, M.P.; Heuvel, W.-J.V.D. Service oriented architectures: Approaches, technologies and research issues. *VLDB J.* **2007**, *16*, 389–415. [CrossRef]
76. Zhao, H.; Chen, X.; Zhang, W.; Liang, P.; Wang, J.; Huang, W. SOA Patterns Selection and Application Based on Software Quality Requirements. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; pp. 361–364.
77. Rahim, R.; Suhardi; Kurniawan, N.B. Designing statistical metadata service computing system based on service oriented Architecture (SOA). In Proceedings of the 2017 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, 23–24 October 2017; pp. 120–127.
78. Pulparambil, S.; Baghdadi, Y. A Comparison Framework for SOA Maturity Models. In Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 19–21 December 2015; pp. 1102–1107.
79. Cheng, B.; Zhu, D.; Zhao, S.; Chen, J. Situation-Aware IoT Service Coordination Using the Event-Driven SOA Paradigm. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 349–361. [CrossRef]
80. Galster, M.; Avgeriou, P. Qualitative Analysis of the Impact of SOA Patterns on Quality Attributes. In Proceedings of the 2012 12th International Conference on Quality Software, Xi'an, China, 27–29 August 2012; pp. 167–170.
81. Pulparambil, S.; Baghdadi, Y. Towards a model for soa adoption based on methodical aspects. In Proceedings of the 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 18–19 July 2017; pp. 105–110.
82. Rodríguez, G.; Teyseyre, A.; Soria, Á.; Berdun, L. A visualization tool to detect refactoring opportunities in SOA applications. In Proceedings of the 2017 XLIII Latin American Computer Conference (CLEI), Cordoba, Argentina, 4–8 September 2017; pp. 1–10.
83. Rida, B.; Ahmed, E. Multiview SOA: Extending SOA using a private cloud computing as SaaS. In Proceedings of the 2015 International Conference on Cloud Technologies and Applications (CloudTech), Marrakech, Morocco, 2–4 June 2015; pp. 1–5.
84. Zhang, H.; Yang, X. Cloud Computing Architecture Based-On SOA. In Proceedings of the 2012 Fifth International Symposium on Computational Intelligence and Design, Hangzhou, China, 28–29 October 2012; pp. 369–373. [CrossRef]
85. Omar, W.; Taleb-Bendiab, A. E-health support services based on service-oriented architecture. *IT Prof.* **2006**, *8*, 35–41. [CrossRef]
86. Appandairaj, A.; Murugappan, S. Service Oriented Architecture Design for Web Based Home Banking Systems with Cloud Based Service. *Int. J. Emerg. Technol. Adv. Eng.* **2013**, *3*, 138–143.
87. Khoonsari, P.E.; Moreno, P.; Bergmann, S.; Burman, J.; Capuccini, M.; Carone, M.; Cascante, M.; De Atauri, P.; Foguet, C.; Gonzalez-Beltran, A.N.; et al. Interoperable and scalable data analysis with microservices: Applications in metabolomics. *Bioinformatics* **2019**, *35*, 3752–3760. [CrossRef]
88. Wan, X.; Guan, X.; Wang, T.; Bai, G.; Choi, B.-Y. Application deployment using Microservice and Docker containers: Framework and optimization. *J. Netw. Comput. Appl.* **2018**, *119*, 97–109. [CrossRef]
89. Gan, Y.; Delimitrou, C. The Architectural Implications of Cloud Microservices. *IEEE Comput. Arch. Lett.* **2018**, *17*, 155–158. [CrossRef]
90. Shamim, S.I.; Bhuiyan, F.A.; Rahman, A. XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices. In Proceedings of the 2020 IEEE Secure Development (SecDev), Atlanta, GA, USA, 28–30 September 2020; pp. 58–64.
91. He, Z. Novel Container Cloud Elastic Scaling Strategy based on Kubernetes. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1400–1404.
92. Baldominos, A.; Ogul, H.; Colomo-Palacios, R. Infection Diagnosis using Biomedical Signals in Small Data Scenarios. In Proceedings of the 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), Cordoba, Spain, 5–7 June 2019; pp. 38–43.
93. O'Brien, O.; O'Reilly, R.D. Beats-Per-Minute (BPM): A Microservice-based Platform for the Monitoring of Health Related Data via Activity Trackers. In Proceedings of the 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), Ostrava, Czech Republic, 17–20 September 2018; pp. 1–7.
94. Gong, Y.; Gu, F.; Chen, K.; Wang, F. The Architecture of Micro-services and the Separation of Frond-end and Back-end Applied in a Campus Information System. In Proceedings of the 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 25–27 August 2020; pp. 321–324.

95. Munonye, K.; Martinek, P. Evaluation of Data Storage Patterns in Microservices Archicture. In Proceedings of the 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2–4 June 2020; pp. 373–380.
96. Bao, L.; Wu, C.; Bu, X.; Ren, N.; Shen, M. Performance Modeling and Workflow Scheduling of Microservice-Based Applications in Clouds. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 2114–2129. [CrossRef]
97. Akbulut, A.; Perros, H.G. Software Versioning with Microservices through the API Gateway Design Pattern. In Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 5–7 June 2019; pp. 289–292.
98. Tapia, F.; Ángel Mora, M.; Fuertes, W.; Aules, H.; Flores, E.; Toulkeridis, T. From Monolithic Systems to Microservices: A Comparative Study of Performance. *Appl. Sci.* **2020**, *10*, 5797. [CrossRef]
99. Götz, B.; Schel, D.; Bauer, D.; Henkel, C.; Einberger, P.; Bauernhansl, T. Challenges of Production Microservices. *Procedia CIRP* **2018**, *67*, 167–172. [CrossRef]
100. Rademacher, F.; Sachweh, S.; Zundorf, A. Aspect-Oriented Modeling of Technology Heterogeneity in Microservice Architecture. In Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, 25–29 March 2019; pp. 21–30.
101. Álvaro, B.; Solé, M.; Huélamo, A.; Solans, D.; Pérez, M.S.; Muntés-Mulero, V. Graph-based root cause analysis for service-oriented and microservice architectures. *J. Syst. Softw.* **2020**, *159*, 110432. [CrossRef]
102. Chen, L. Microservices: Architecting for Continuous Delivery and DevOps. In Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA), Seattle, WA, USA, 30 April–4 May 2018; pp. 39–397.
103. Balalaie, A.; Heydarnoori, A.; Jamshidi, P. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Softw.* **2016**, *33*, 42–52. [CrossRef]
104. Roca, S.; Sancho, J.; García, J.; Álvaro, A. Microservice chatbot architecture for chronic patient support. *J. Biomed. Inform.* **2020**, *102*, 103305. [CrossRef]
105. Gribaudo, M.; Iacono, M.; Manini, D. Performance Evaluation of Replication Policies in Microservice Based Architectures. *Electron. Notes. Comput. Sci.* **2018**, *337*, 45–65. [CrossRef]
106. Lonetti, F.; Marchetti, E. *Chapter Three—Emerging Software Testing Technologies*; Memon, C., Ed.; Elsevier: Amsterdam, The Netherlands, 2018; pp. 91–143.
107. Shoumik, F.S.; Talukder, I.M.M.; Jami, A.I.; Protik, N.W.; Hoque, M. Scalable micro-service based approach to FHIR server with golang and No-SQL. In Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 22–24 December 2017; pp. 1–6.
108. Khan, R.; Amjad, M. Performance testing (load) of web applications based on test case management. *Perspect. Sci.* **2016**, *8*, 355–357. [CrossRef]
109. Ramakrishnan, R.; Kaur, A. Little's law based validation framework for load testing. *Inf. Softw. Technol.* **2019**, *108*, 88–98. [CrossRef]
110. Abdullah, M.; Iqbal, W.; Erradi, A. Unsupervised learning approach for web application auto-decomposition into microservices. *J. Syst. Softw.* **2019**, *151*, 243–257. [CrossRef]
111. Kumar, N.M.; Mallick, P.K. The Internet of Things: Insights into the building blocks, component interactions, and architecture layers. *Procedia Comput. Sci.* **2018**, *132*, 109–117. [CrossRef]
112. Alharbi, F.; Atkins, A.; Stanier, C. *Cloud Computing Adoption in Healthcare Organisations: A Qualitative Study in Saudi Arabia BT—Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXV*; Hameurlain, A., Küng, J., Wagner, R., Sakr, S., Razzak, I., Riyad, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 96–131.
113. Cortavitarte, J. Engine Health Report. 2021. Available online: https://guide.blazemeter.com/hc/en-us/articles/360000290058-Engine-Health-Report-Engine-Health-Report (accessed on 17 February 2021).
114. Khaleq, A.A.; Ra, I. Agnostic Approach for Microservices Autoscaling in Cloud Applications. In Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 5–7 December 2019; pp. 1411–1415.
115. Hassan, S.; Ali, N.; Bahsoon, R. Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity. In Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, Sweden, 3–7 April 2017; pp. 1–10.
116. Kiss, T.; DesLauriers, J.; Gesmier, G.; Terstyanszky, G.; Pierantoni, G.; Abu Oun, O.; Taylor, S.J.; Anagnostou, A.; Kovacs, J. A cloud-agnostic queuing system to support the implementation of deadline-based application execution policies. *Futur. Gener. Comput. Syst.* **2019**, *101*, 99–111. [CrossRef]
117. Enler, E.; Pentek, I.; Adamko, A. Healthcare Framework for Smarter Cities with bio-sensory data. In Proceedings of the 2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Mariehamn, Finland, 23–25 September 2020; pp. 337–342.
118. Xiao, Z.; Wijegunaratne, I.; Qiang, X. Reflections on SOA and Microservices. In Proceedings of the 2016 4th International Conference on Enterprise Systems (ES), Melbourne, Australia, 2–3 November 2016; pp. 60–67.
119. Sriraman, A.; Wenisch, T.F. µ Suite: A Benchmark Suite for Microservices. In Proceedings of the 2018 IEEE International Symposium on Workload Characterization (IISWC), Raleigh, NC, USA, 30 September–2 October 2018; pp. 1–12. [CrossRef]

120. Zainol, M.F.; Farook, R.S.M.; Hassan, R.; Halim, A.H.A.; Rejab, M.R.A.; Husin, Z. A New IoT Patient Monitoring System for Hemodialysis Treatment. In Proceedings of the 2019 IEEE Conference on Open Systems (ICOS), Pulau Pinang, Malaysia, 19–21 November 2019; pp. 46–50.
121. Takeda, Y.; Yokoyama, D.; Nakamichi, N.; Inaba, R.; Watanabe, K.; Yamada, T. Visualization of Remote Touch Panel for Dialysis Patient on Prototype Bed. In Proceedings of the 2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech), Kyoto, Japan, 10–12 March 2020; pp. 54–58.
122. Zhong, L.; Mu, L.; Li, J.; Wang, J.; Yin, Z.; Liu, D. Early Prediction of the 2019 Novel Coronavirus Outbreak in the Mainland China Based on Simple Mathematical Model. *IEEE Access* **2020**, *8*, 51761–51769. [CrossRef]
123. Rimsan, M.; Mahmood, A.K.; Umair, M.; Hassan, F. COVID-19: A Novel Framework to Globally Track Coronavirus Infected Patients using Blockchain. In Proceedings of the 2020 International Conference on Computational Intelligence (ICCI), Bandar Seri Iskandar, Malaysia, 8–9 October 2020; pp. 70–74.
124. Han, R.; Liu, Z.; Chen, C.L.P.; Xu, L.; Peng, G. Mortality prediction for COVID-19 patients via Broad Learning System. In Proceedings of the 2020 7th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS), Guangzhou, China, 13–15 November 2020; pp. 837–842.
125. Casiraghi, E.; Malchiodi, D.; Trucco, G.; Frasca, M.; Cappelletti, L.; Fontana, T.; Esposito, A.A.; Avola, E.; Jachetti, A.; Reese, J.; et al. Explainable Machine Learning for Early Assessment of COVID-19 Risk Prediction in Emergency Departments. *IEEE Access* **2020**, *8*, 196299–196325. [CrossRef]
126. Ananthi, S.; Bhuvaneswari, V. Prediction of heart and kidney risks in diabetic prone population using fuzzy classification. In Proceedings of the 2017 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 5–7 January 2017; pp. 1–6.
127. Patil, S.S.; Malpe, K. Implementation of Diabetic Retinopathy Prediction System using Data Mining. In Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 27–29 March 2019; pp. 1206–1210.
128. Qiao, L.; Zhu, Y.; Zhou, H. Diabetic Retinopathy Detection Using Prognosis of Microaneurysm and Early Diagnosis System for Non-Proliferative Diabetic Retinopathy Based on Deep Learning Algorithms. *IEEE Access* **2020**, *8*, 104292–104302. [CrossRef]
129. Saputra, R.Y.; Nugroho, L.E.; Kusumawardani, S.S. Collecting the Tourism Contextual Information data to support the tourism recommendation system. In Proceedings of the 2019 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 24–25 July 2019; pp. 79–84.
130. Bigheti, J.A.; Fernandes, M.M.; Godoy, E.D.P. Control as a Service: A Microservice Approach to Industry 4.0. In Proceedings of the 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), Naples, Italy, 4–6 June 2019; pp. 438–443. [CrossRef]
131. Prasetyo, Y.A.; Suhardi. Microservice Platform for Smart City: Concepts, Services and Technology. In Proceedings of the 2018 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, 22–26 October 2018; pp. 358–363.
132. Mendoza-Pitti, L.; Calderon-Gomez, H.; Vargas-Lombardo, M.; Gomez-Pulido, J.M.; Castillo-Sequera, J.L. Towards a service-oriented architecture for the energy efficiency of buildings: A systematic review. *IEEE Access* **2021**, *9*, 1. [CrossRef]