A FaaS approach for long-term monitoring in rehabilitation

Pablo Serrano-Gutierrez[1][0000−0002−5397−690X], Inmaculada Ayala[2,3][0000−0002−5119−3469], and Lidia Fuentes[2,3][0000−0002−5677−7156]

[1] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, España
pserrano@lcc.uma.es
[2] ITIS Software, Universidad de Málaga, España
{ayala,lff}@lcc.uma.es

Abstract. The function as a service (FaaS) model, in which the costs are linked to the function time of use, is especially suitable in IoT or similar systems, where the information is collected periodically to consume less energy. A FaaS approach also allows us to fine-tune the application's behaviour at a function level, controlling which functions should be executed at any given time. This makes it possible to apply optimisation processes in an automated way. In this work, a FaaS-based platform is proposed to develop e-Health applications that adapt their behaviour to the different stages of the patient rehabilitation process. Its operation is illustrated by a case study on patients with heart disease. The system's operation will vary throughout the different stages of the rehabilitation, adjusting to the available sensors, the evolution of the patient's physical condition and other parameters such as the required precision.

Keywords: FaaS · Software product lines · Adaptive systems · e-Health.

1 Introduction

Telerehabilitation is a branch of telemedicine that complements traditional rehabilitation therapies by providing valuable information to help in the evaluation, monitoring, and treatment of patients through wearable sensors [18]. Telerehabilitation enables the long-term monitoring of patients, and it is essential in chronic diseases requiring extended monitoring periods, personalised attention and periodic visits to healthcare centres. Therapists could monitor patient data remotely to determine their progress and develop personalised interventions. This type of application, in which data collection and information processing are carried out discontinuously, either from time to time or in response to an event, fits perfectly with the function as a service (FaaS) model, in which applications are developed based on the use of serverless functions. The joint work of FaaS and sensor-based applications has mainly two benefits. First, they can be deployed in an infrastructure managed by a third party. In addition, the FaaS billing method only charges for the time of use of the functions, not for the rental of the infrastructure.

In long-term monitoring systems, wearable sensors are usually battery-operated and have poor computational resources. Therefore, the wise use of such resources is fundamental to elongating the system's operation. In this context, saving computational resources is challenging because the list of customisation options for wearable devices is usually short and contains actions related to increasing/decreasing the monitoring frequencies of services, modifying the estimation methods or performing task offloading. In this scenario, exploiting the patient's condition for adapting e-health applications is fundamental to saving computational resources. Depending on this, the system should take measurements at more regular intervals or use sensors with greater precision. In addition, it could be interesting to use different functions for processing or other operations needed by the application, to adjust it to desired quality of service (QoS) objectives. However, managing the variability of these systems and generating their optimal configurations on the fly is a complex task that is not properly managed by current FaaS platforms. They lack adequate tools to manage function reuse, abstractions, and programming models for building non-trivial FaaS applications [12]. The leading platforms provide specific tools [2, 7, 14] but they focus on the orchestration of functions and do not consider the QoS. Several works tackle optimisation and QoS in the context of FaaS [4, 6, 21], but focusing only in allocating the resources for functions dynamically at runtime. So, they do not consider alternative quality-valued functions nor the overall quality of function configurations.

Software Product Lines (SPLs) [17], are used to define families of software products using reusable elements. We will use an SPL approach to model the variability of the long-term monitoring application. For this, we will make use of Feature Models [11], that are hierarchical specifications of systems based on their features. Additional constraints added to those defined by the tree of features itself can also be specified on these models and are called cross-tree constraints. A selection of features that respect tree and cross-tree constraints is called a configuration, and if it also fits an objective function, it will be considered valid.

In this contribution, we propose the application of function selection techniques [19] for developing long-term monitoring systems that follow a FaaS architecture. We present a framework for developing e-health applications with self-adaptive capabilities using a FaaS approach. It can adjust the QoS of a system at runtime by meeting specific functional and non-functional requirements. Using our approach, a FaaS application can be developed focusing on the functionality and not on the characteristics of the different devices that make up the system in terms of suitability, precision or other restrictions. So, it can adapt its operation to changing needs during the monitoring process of patients, without human intervention.

The rest of the paper is structured as follows: Section 2 discusses some related work; Section 3 presents the case study; Section 4 presents our approach; Section 5 shows the experimental results of the platform in an execution environment, and Section 6 presents our conclusions.

2 Related work

The work presented in this paper combines the FaaS model with the self-adaptation of IoT devices to the user context. There are no other works in this line, but there are contributions to the combination of e-health with FaaS and the adaptation of user context in the IoT.

Several works propose using FaaS for healthcare applications due to its cost-effective model and simplified deployment and management. The paper [8] summarises four case studies of successful adoption of the FaaS architecture for healthcare applications. The authors stress the existing challenges in the field related to integrating the IoT with the serverless healthcare system. The work [9] proposes a Cost-Efficient Service Selection and Execution and Blockchain-Enabled Serverless Network for Internet of Medical Things systems. The proposed system focuses on enhancing security and reducing costs of operation. In this line, the paper [10] introduces the mobility-aware security dynamic service composition, an algorithmic framework for serverless that uses restricted Boltzmann to enhance security and reduce cost. The results of the author's experiments show an increase of 25% in terms of security and 35% in application cost. The work presented in [16] proposes dependable Serverless architecture for WBAN applications. The given architecture improves the dependability of WBAN applications exploiting properties of the Serverless model, scalability and availability. In addition, the solution is cost-effective and secure and private by design. These papers focus on optimising costs and enhancing security, but they do not focus on the context of the patient.

Several works have approached the self-adaptation of IoT devices to the user context in e-health. The RA architecture combines data-driven penalisation with self- adaptation [5] for e-health apps. CARA [23] is a pervasive health care system that supports daily activity analysis and alert generation. CARA uses fuzzy rule-based reasoning to adapt the application to the user's current condition. H. Mshali et al. propose a context-aware health monitoring system targeted at elderly and isolated people living alone [15]. The system can adjust the sampling frequency of the monitoring system, taking into account the user's current profile. The self-organised learning model presented in [22] can adapt its work to a particular patient by using feedback from medical staff and a real-time linear optimisation process. The work presented in [20] proposes a context-aware smart home system that allows older people to live independently. The system provides adapted services in terms of comfort, health and safety considering the user's profile.

3 Case Study

As a case study, we have selected a system to monitor patients with heart problems undergoing rehabilitation. In this context, it is essential to monitor the general vital state of the patients, as well as their progress in the physical exercises performed at the different stages of the rehabilitation process. Usually, four

different phases are considered [13] in which the patient's monitoring requirements differ, ranging from a more detailed follow-up of their health status to one focused on achieving longer-term goals. So, in each rehabilitation stage, requirements are different, and the system should be adapted without the intervention of the developer or the user. Likewise, the therapist can adapt the requirements at any time to the evolution of the patient.

The telerehabilitation system performs tasks depicted in Fig. 1. Initially, the application needs to obtain data from different sensors that may be present in the system. Subsequently, the system processes patient data and generates a report for patient follow-up. It is also necessary to carry out a setup process that allows configuring, activating or deactivating sensors. At every moment, the system can readjust its operation, returning to the setup process again.

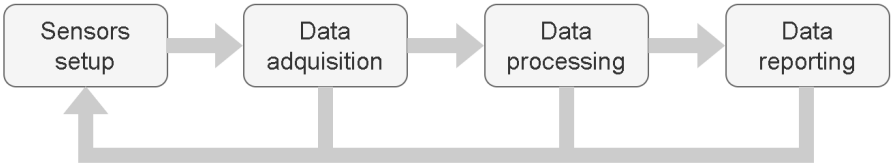

Fig. 1. Tasks performed by long-term monitoring applications

4 Our Approach

This section describes the different processes performed by our platform (Fig. 2) and its application to the presented case study. When an application using this platform needs to operate and calls a function, the system will execute the most appropriate one from among a set of functions that can perform the required task. Likewise, it selects function input parameters extracting information from the analysis of function behaviour. The input parameters are chosen considering the objectives pursued and the restrictions that affect the modelled system as a whole. The system makes its decisions by performing an optimisation process based on feature models.

Fig. 3 represents the feature model related to the different operations carried out by the application. We can observe both the services and the sensors involved. A certain sensor or device could be useful to measure the same parameter. For example, we can obtain the heart rate from a specific heart rate sensor, or we can use a wrist device that has a heart rate sensor integrated. Each one of them can be different in aspects like precision or energy consumption. Likewise, the processing of the measured data and the generation of reports can be done in different ways, considering different encryption algorithms, taking into account the platforms used, etc. All this variability is reflected in the model and will be

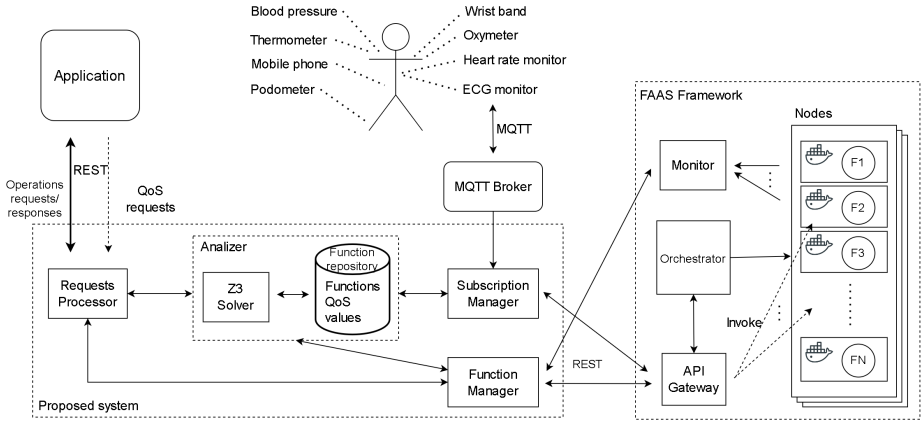

Fig. 2. Proposed platform architecture

used by our system to choose the best possible configuration according to the needs of the application at all times.

One of the advantages of using our platform is the ease of development. The developer program the application focusing on the operations that need to be carried out without considering the available devices. For example, if the application needs to get the patient's heart rate, it calls a serverless function like *Get_Heart_Rate* instead of calling a specific function like *Get_Heart_Rate_From_ECG_Monitor*. Then our system will be in charge of obtaining said information from one sensor or another depending on its characteristics and other requirements. Also, it will consider what sensors are present in the system to adapt its operation to it.

4.1 Model Generation and analysis

Our system uses the feature model of a telerehabilitation system shown in Fig. 3 and another series of models that are automatically generated based on the information entered by the developer when implementing any of the serverless functions that constitute the application. When the developer adds a function to the system, it must be registered in the function repository (see Fig. 2). All relevant information about the execution of that function is stored there and may be updated after every execution. As part of this information, the operation that is carried out by each function needs to be added. Fig. 4 shows part of the content of this repository for the case study. We can see information about functions related to reading data from medical devices, as well as properties such as the time consumed by the operation, the data's precision obtained, and the energy consumed by the device. It is essential to bear in mind that some of these values may vary according to the execution function parameters or, in the case of sensors or devices, due to their mode of operation. For example, while the wrist band considered in the case study offers a low level of accuracy for measuring

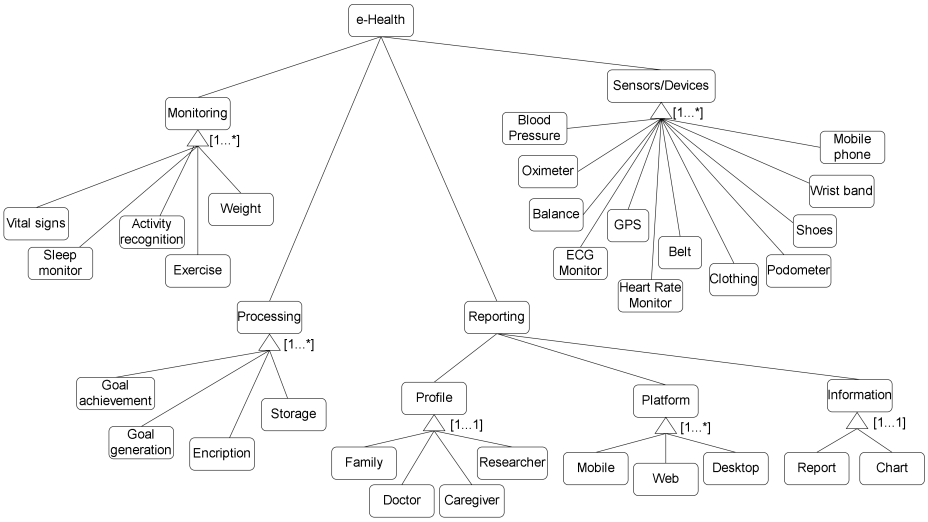

Fig. 3. Feature model of the e-health application

oxygen, the accuracy can be set to medium for the heart rate sensor. Therefore, the accuracy of the heart rate reading operation through the wrist band sensor will be the accuracy to which said device is configured, which is expressed by the equation: *Precision (op)=Precision(Wrist)*. All these equations will be considered in conjunction with the logical expressions and equations extracted from the feature models analysis. In other cases, more complex relationships may occur. For example, if the power consumption depends on the level of precision used. Thus, if we consider that each level of precision has an associated number that increases with precision (e.g. *1=Low, 2=Medium*), in the case of the heart rate monitor, if its energy consumption were 2.5 times said value, it would be expressed as *Precision (op)=PrecisionVal(HRM)*2.5*.

Our system automatically generates a function feature model using the information from the function repository. To do this, the system groups the functions that can perform the same operation (for example, two functions that serve to encrypt the data but use different algorithms). The functions that belong to the same group will lead to a group of features with an exclusive relationship and with as many elements as functions that perform the same operation have been found. Our tool generates another feature model using the information (also extracted from the function repository) about function parameters. In this case, parameters are associated with features, and an exclusive group with all the parameters' supported alternatives is generated. For our case study, the system automatically generates the functions feature model we can observe in Fig. 5. Due to the extension, only the part related to vital signs monitoring process is shown. In this process, the application works directly with the available sensors to monitor the patient's condition. The feature model's constraints depend on the patient's rehabilitation stage. For example, during the first phase, the pa-

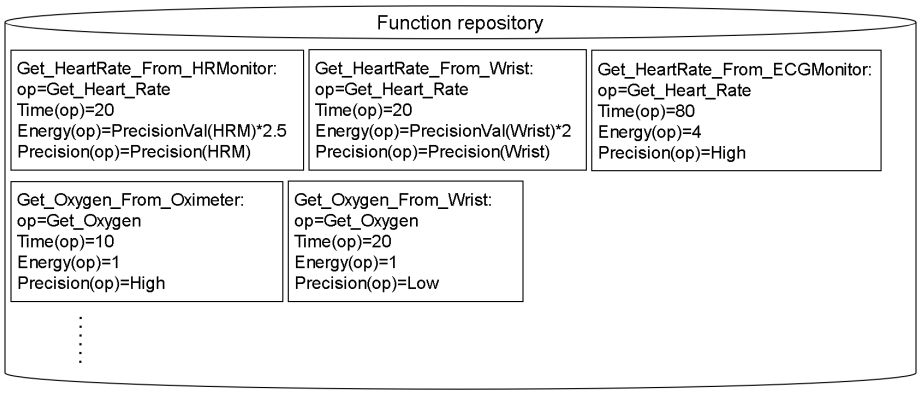

Fig. 4. Repository of functions

tient's vital signs must be monitored more thoroughly, requiring measurements with higher precision and greater frequency. As we can observe in the feature model related to vital signs monitoring (top left of Fig. 5), all the alternative sensors that can be used to obtain the different data needed are represented. At the same time, these involved sensors have their characteristics reflected in the feature models seen in the same figure. The information provided by the feature models is completed with the restrictions that may affect any of the elements of the tree. In this example, the constraint shown indicates that if the wrist band device is working in energy-saving mode, it will not be able to work as a blood pressure sensor, and the precision of the heart rate monitor of the wrist band will have to be set to low.

With all these feature models, we apply Software Product Line refactoring techniques [1] to generate a feature model with information about the application as a whole. From it, and using the Z3 solver, we will determine the valid configurations that adjust to the model's restrictions and the application's requirements. To do this, we first create a mathematical model from this feature model. This conversion is carried out automatically through a recursive process in which the tree representing the relationships between features is traversed. This way, all the logical relationships between the different features are obtained. At the end of this process, we get a logical expression representing the entire tree. After the Z3 optimisation, we get a set of functions and parameters that will allow us to comply with the application requirements in this stage. For example, initially it could be to operate with the higher precision level while minimising the energy consumption. These requirements can be modified at any time, which would allow adjusting the behaviour also according to the evaluation of the patient.

4.2 Request processing

An application that uses our system, instead of making serverless function requests to the Faas framework, introduces them into our system. The task of

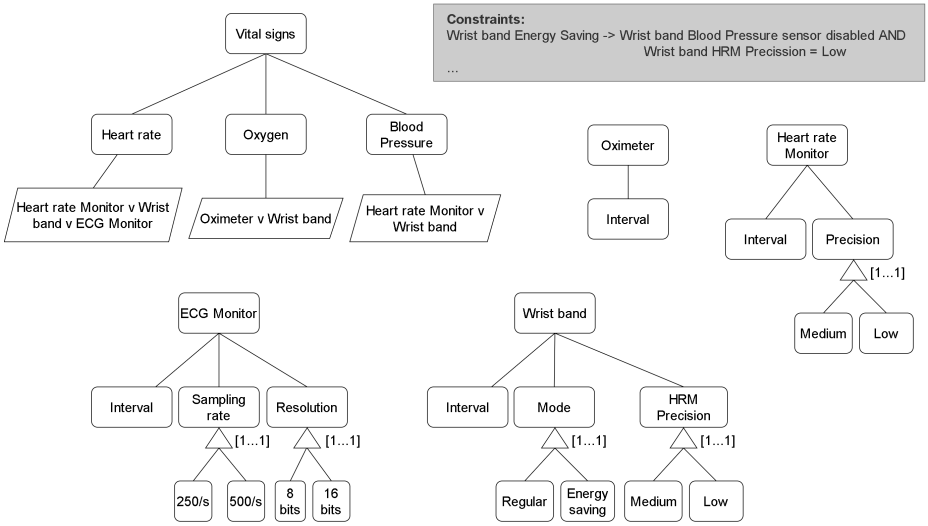

Fig. 5. Feature models of vital signs monitoring and the related sensors

processing is handled by the *Request Processor* module (see Fig. 2), which performs QoS adjustments as well. When the application requests the execution of operations, the calls, using the configuration obtained in the analysis process, are transformed into calls to serverless functions. On the other hand, QoS requests will trigger the analysis process and the generation of a new configuration that adapts to the requested parameters.

The work of the *Request Processor* makes the use of our solution transparent to the user. So, it is possible to work in the same way as it would be done directly with the FaaS framework. Our system listens to REST-type requests, widely used by these frameworks, and the response generated after the execution of the serverless functions is returned to the application using the same format.

4.3 Function selection and execution

An essential functionality of our system is redirecting a generic function call to the appropriate function implementation. This task is performed by the *Function Manager* module (see Fig. 2), which uses the information obtained in the analysis process. When a REST request for an operation arrives, the system proceeds to execute a specific serverless function through the FaaS framework. This is performed through the API Gateway, which in the Faas frameworks is the element in charge of processing the calls to functions that are deployed in containers located in different nodes and are managed by an orchestrator. After execution, the returned values are transferred directly to the application, which will be equivalent to calling the FaaS function directly.

This module also manages function monitoring, requesting this information from the framework after each execution. In our system, this information is used

to keep track of active sensors. If a change is detected, the model generation and analysis process will be relaunched, leading to the generation of a new optimal configuration using the current set of sensors.

4.4 Subscription management

Like our case study, applications based on sensors usually work using the publish-subscribe pattern. In this pattern, we have two roles: publishers and subscribers. Publishers periodically send information about a topic, e.g. the heart rate of the patient. The subscribers are entities interested in publishers' information and regularly receive the data generated by them. This pattern is used by MQTT [3], the protocol considered by our system to communicate with measurement devices. We have selected MQTT because it is one of the most extended protocols for IoT applications. Its simplicity makes it ideal for applications where the devices must consume as little as possible.

To adapt our system to the publish-subscribe pattern, we have implemented the *Subscritption Manager* module (see Fig. 2). This component is an intermediate between the *MQTT broker* and our platform. When an active sensor has new data, it generates an event. The *Subscritption Manager* manages the subscription to these events by calling the handler function of the serverless framework. For example, suppose the oximeter sensor has new data. In that case, it raises a *Oximeter* event, which is caught by the *Subscritption Manager* that generates a call to the handler associated with this event, i.e., the serverless function subscribed to this data, *Get_Oxygen_From_Oximeter*.

5 Results

We have conducted several experiments using different sets of sensors to evaluate the performance of our system. These experiments have been carried out using an Intel i5-7400 PC, 3.00 GHz, 24 MB of RAM using Python v.3.10.4 and Z3 v.4.8.15. We have used OpenFaaS as the FaaS framework and Prometheus for system monitoring. Concerning the assistive devices, we have used Mosquitto MQTT as the broker and Paho[3] as the client. We have simulated the different stages of the rehabilitation process to check that the system can adapt its operation over time.

As our system introduces additional steps over what a pure FaaS application would do, we evaluated the performance of the different processes. Considering a phase where all monitoring processes are necessary and that all possible sensors are present in the system, it took only 36 ms to obtain an optimal configuration to achieve the required levels of accuracy while minimising the energy consumption. In case it is detected that an active sensor is not present, the system reconfiguration process would take 20 ms. This is because rebuilding all previously generated feature models is unnecessary.

[3] https://www.eclipse.org/paho/

Once the optimisation process has been carried out, the system redirects the application's REST request to a serverless function through a new request so that, in reality, two requests are being carried out. The time consumed by this additional request is 5 ms measured on our local network. Likewise, when a subscription event is received, the call to the serverless function through a REST request also consumes an additional 5 ms. Therefore, there will be a delay of 5 ms each time a sensor sends new data, which is not a problem for our application.

We have also considered the scalability of the system. If, for example, we have multiple sensors that can be used for the same purpose and we want the application to select the most suitable ones, the number of functions would grow. The same happens if the devices used have multiple adjustable parameters since new levels of variability are introduced and, therefore, the system may take longer to find an optimal solution. As shown in Fig. 6, the times obtained are increased following an exponential model. However, the number of simulated alternative sensors and their corresponding configurable parameters are values much higher than those that will be found in a real system. For usual values, we can observe a more linear behaviour and with measurements that are still at very acceptable levels for this kind of applications. It is important to keep in mind that the system only performs this processing when the monitoring phase changes or if any condition in the system varies, such as a sensor deactivation or a change in requirements introduced in the application.

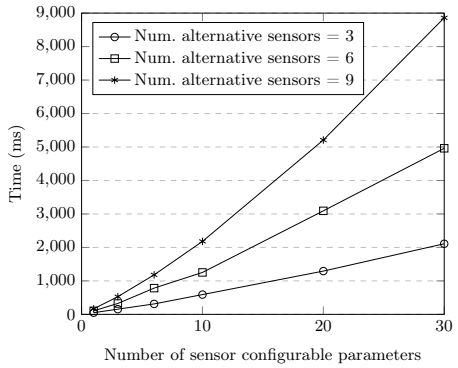

Fig. 6. Processing time varying the number of alternative sensors for every operation and the number of configurable parameters they have

6 Conclusions

This work presents a system that allows to apply the FaaS model for long-term monitoring applications like Telerehabilitation systems. Our platform uses feature modelling to enforce QoS parameters during the application's execution,

making it self-adaptive. In the case of long-term monitoring applications, such as the one presented in the case study, the development can be carried out without taking into account the changing requirements throughout each process stage. It is also not necessary to control which sensors are available at all times since the system will adapt its operation to the present sensors.

Our system is implemented in Python, using z3 as the solver used to carry out the optimisation process. Experimental results shows that the performance of the platform is good enough not to degrade the performance of the application, and remains good when considering a high variability.

The platform can be easily adapted to any other type of IoT application by simply defining a new SPL model of the application to use as the basis for optimisation. Likewise, it is easily adaptable to any protocol used for the IoT.

In future work, we plan to analyse the behaviour of our platform in Edge environments, which may be of interest to apply it in eHealth applications that require a rapid response from the system.

Acknowledgements This work is supported by the European Union's H2020 research and innovation programme under grant agreement DAEMON 101017109, by the projects co-financed by FEDER funds LEIA UMA18-FEDERJA-15, MED-EA RTI2018-099213-B-I00 and Rhea P18-FR-1081, the PRE2019-087496 grant from the Ministerio de Ciencia e Innovación and by the project DISCO B1-2012_12 funded by Universidad de Málaga.

References

1. Alves, V., Gheyi, R., Massoni, T., Kulesza, U., Borba, P., Lucena, C.: Refactoring product lines. In: Proc. of the 5th International Conference on Generative Programming and Component Engineering. pp. 201–210. Association for Computing Machinery, New York, NY, USA (2006)
2. Amazon Web Services: AWS Step Functions, available at https://aws.amazon.com/es/step-functions/
3. Barata, D., Louzada, G., Carreiro, A., Damasceno, A.: System of acquisition, transmission, storage and visualization of pulse oximeter and ecg data using android and mqtt. Procedia Technology **9**, 1265–1272 (12 2013)
4. Bocci, A., Forti, S., Ferrari, G.L., Brogi, A.: Placing faas in the fog, securely. In: 5th Italian Conference on Cybersecurity, ITASEC 2021. vol. 2940, pp. 166–179. CEUR-WS (2021)
5. Grua, E.M., De Sanctis, M., Malavolta, I., Hoogendoorn, M., Lago, P.: An evaluation of the effectiveness of personalization and self-adaptation for e-health apps. Information and Software Technology **146**, 106841 (2022)
6. Hoseiny Farahabady, M., Lee, Y.C., Zomaya, A.Y., Tari, Z.: A qos-aware resource allocation controller for function as a service (faas) platform. In: Service-Oriented Computing. pp. 241–255. Springer International Publishing, Cham (2017)
7. IBM: IBM Cloud Functions, available at https://www.ibm.com/cloud/functions
8. Kumari, A., Behera, R.K., Sahoo, B., Misra, S.: Role of serverless computing in healthcare systems: Case studies. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) Computational Science and Its Applications

– ICCSA 2022 Workshops. pp. 123–134. Springer International Publishing, Cham (2022)
9. Lakhan, A., Ali Dootio, M., Sodhro, A.H., Pirbhulal, S., Groenli, T.M., Khokhar, M.S., Wang, L.: Cost-efficient service selection and execution and blockchain-enabled serverless network for internet of medical things. Mathematical Biosciences and Engineering **18**(6), 7344–7362 (2021)
10. Lakhan, A., Mohammed, M.A., Rashid, A.N., Kadry, S., Hameed Abdulkareem, K., Nedoma, J., Martinek, R., Razzak, I.: Restricted boltzmann machine assisted secure serverless edge system for internet of medical things. IEEE journal of biomedical and health informatics **PP** (2022)
11. Lee, K., Kang, K.C., Lee, J.: Concepts and guidelines of feature modeling for product line software engineering. In: Gacek, C. (ed.) Software Reuse: Methods, Techniques, and Tools. pp. 62–77. Springer, Berlin, Heidelberg (2002)
12. Leitner, P., Wittern, E., Spillner, J., Hummer, W.: A mixed-method empirical study of function-as-a-service software development in industrial practice. Journal of Systems and Software **149**, 340–359 (2019)
13. de Macedo, R.M., Faria-Neto, J.R., Costantini, C.O., Casali, D., Muller, A.P., Costantini, C.R., de Carvalho, K.A., Guarita-Souza, L.C.: Phase i of cardiac rehabilitation: A new challenge for evidence-based physiotherapy. World journal of cardiology **3(7)**, 248–255 (2011)
14. Microsoft: Azure Functions, available at https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview
15. Mshali, H., Lemlouma, T., Magoni, D.: Adaptive monitoring system for e-health smart homes. Pervasive and Mobile Computing **43**, 1–19 (2018)
16. Paul, P.C., Loane, J., McCaffery, F., Regan, G.: A serverless architecture for wireless body area network applications. In: Papadopoulos, Y., Aslansefat, K., Katsaros, P., Bozzano, M. (eds.) Model-Based Safety and Assessment. pp. 239–254. Springer International Publishing, Cham (2019)
17. Pohl, K., Böckle, G., Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Berlin, Heidelberg (01 2005). https://doi.org/10.1007/3-540-28901-1
18. Porciuncula, F., Roto, A.V., Kumar, D., Davis, I., Roy, S., Walsh, C.J., Awad, L.N.: Wearable movement sensors for rehabilitation: A focused review of technological and clinical advances. PM&R **10**(9, Supplement 2), S220–S232 (2018), innovations Influencing Physical Medicine and Rehabilitation
19. Serrano-Gutierrez, P., Ayala, I., Fuentes, L.: Applying qos in faas applications: a software product line approach. In: International Conference on Microservices. pp. 1–6 (2022)
20. Staifi, N., Brahimi, S., Maamri, R., Belguidoum, M.: Towards a smart home for elder healthcare. In: 7th International Conference on Future Internet of Things and Cloud (FiCloud). pp. 230–237 (2019)
21. Tzenetopoulos, A., Marantos, C., Gavrielides, G., Xydis, S., Soudris, D.: FADE: FaaS-Inspired Application Decomposition and Energy-Aware Function Placement on the Edge, pp. 7–10. ACM (2021)
22. Verstaevel, N., Georgé, J.P., Bernon, C., Gleizes, M.P.: A self-organized learning model for anomalies detection: Application to elderly people. In: IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO). pp. 70–79 (2018)
23. Yuan, B., Herbert, J.: Context-aware hybrid reasoning framework for pervasive healthcare. Personal and Ubiquitous Computing **18**(4), 865–881 (Apr 2014)