Microservice chatbot architecture for chronic patient support

Surya Roca[a,1], Jorge Sancho[a], José García[a], Álvaro Alesanco[a]

[a] *Aragón Institute of Engineering Research (I3A), University of Zaragoza, Zaragoza, Spain*

Abstract

Chatbots are able to provide support to patients suffering from very different conditions. Patients with chronic diseases or comorbidities could benefit the most from chatbots which can keep track of their condition, provide specific information, encourage adherence to medication, etc. To perform these functions, chatbots need a suitable underlying software architecture. In this paper, we introduce a chatbot architecture for chronic patient support grounded on three pillars: scalability by means of microservices, standard data sharing models through HL7 FHIR and standard conversation modelling using AIML. We also propose an innovative automation mechanism to convert FHIR resources into AIML files, thus facilitating the interaction and data gathering of medical and personal information that ends up in patient health records. To align the way people interact with each other using messaging platforms with the chatbot architecture, we propose these very same channels for the chatbot-patient interaction, paying special attention to security and privacy

Email addresses: surya@unizar.es (Surya Roca), jslarraz@unizar.es (Jorge Sancho), jogarmo@unizar.es (José García), alesanco@unizar.es (Álvaro Alesanco)

[1] Corresponding author

issues. Finally, we present a monitored-data study performed in different chronic diseases, and we present a prototype implementation tailored for one specific chronic disease, psoriasis, showing how this new architecture allows the change, the addition or the improvement of different parts of the chatbot in a dynamic and flexible way, providing a substantial improvement in the development of chatbots used as virtual assistants for chronic patients.
Keywords: Artificial Intelligence Markup Language (AIML), Chronic patient support, Fast Healthcare Interoperability Resources (FHIR), Medical chatbot, Messaging platforms, Microservice architecture

1. Introduction

Chatbots are currently a hot topic in eHealth scenarios. A chatbot is a computer program that automatically provides services conversing with the final user through diverse communication channels (e.g. messaging platforms, mobile apps, etc.). In the last five years, fostered by the explosion of Artificial Intelligence and the enormous penetration in society of messaging platforms, chatbots have been gaining momentum in the eHealth world. Studies show that there exist positive results in support-chatbots for patients with breast cancer [1], with overall satisfaction of 93.95%. Chatbots can be used as virtual assistants helping their users by playing many different roles, for example as symptom checkers [2], medication reminders [3] or personal data gatherers [4]. Moreover, physicians agree with the idea that chatbots can help in most of the automatic simple tasks in healthcare scenarios [5]. However, their usability is limited by the algorithms behind them, their ability to share data, their scalability and the sense of security and privacy they are able to

implement and transmit to their users.

One of the eHealth scenarios that is growing fast and is expected to grow even faster in the coming years is caring for patients with chronic diseases or comorbidities [6]. While research on patient-centred care has largely spawned from technical and clinical research traditions, it inevitably (not intentionally) concentrates mainly on medical staff and supporting clinical work. Traditionally, healthcare and telemedicine are focused on acquiring data and information from patients and transferring them into healthcare contexts, not the other way around. Chatbots are the perfect tools to make possible this turnaround, being aligned with chronic patients' needs, interacting with them in the same way that patients communicate with their friends and relatives through their favourite messaging platforms. Nevertheless, current eHealth chatbots are not taking advantage of this great potential and are only being used in a very generic manner (nutritional disorders and neurological disorders are the areas most tracked [7]). Although these are interesting and necessary approaches, they do not realize the full potential of chatbots which remain quite limited for chronic patient support. To unleash this potential, chatbot software architecture should be grounded on three pillars: scalability that enables easy growth, standard data models that foster data sharing and reusability, and standard conversational modeling that facilitates the conversation between the user and the chatbot.

Scalability provides a growing ecosystem of services available to patients while they evolve (e.g. a chronic patient develops a new comorbidity). Thus, the chatbot would be able to provide them with solutions without the need of using other chatbots or even the need to go back to a specific mobile app.

In other words, due to the rise in patients' comorbidities, eHealth chatbots should be scalable to adapt to new patients' needs, providing them with new tasks and services over time. The lack of scalability in these systems can hamper the chatbot growth, i.e. the offering of new services to healthcare actors (patients, doctors, health professionals in general).

To solve the problem of scalability, the chatbot architecture must be modular and flexible. Microservices are an architectural paradigm which emphasizes modular, lightweight services with a high degree of cohesion. This is in contrast to monolithic applications where tightly integrated components implement the applications' functionality and changing requirements affects the system as a whole. In contrast, microservices can be developed, deployed and scaled independently of other services that make up the system. The basis of microservices is to split a single application into a set of small services, each running its own process. Microservices communicate through well-defined interfaces and standard lightweight protocols such as HTTP and do not need to use the same development languages or platforms. Thus, they are the perfect choice for a modular and continuously growing architecture to support the needs of chronic patients. New functionalities addressing new health conditions can be added as new microservices, fostering a care ecosystem and open to contributions from collaborative developers.

The choice of an open model for patient data is crucial to endow the chatbot architecture with an open nature for data sharing, avoiding the creation of isolated data silos, so dangerous for eHealth expansion [8]. In eHealth environments, the ability of standard data sharing enables all the gathered data to be integrated in other bigger structures related with the patient such

as his/her Electronic or Personal Health Record (EHR/PHR). With this in mind, all the data exchanged with the chatbot regarding the patient's conditions would end up, eventually, in his/her EHR, enriching it and enabling carers (doctors, nurses, etc.) to complete their health assessment of the patient with important data gathered on a regular basis. Fast Healthcare Interoperability Resources (FHIR) is a next generation standards framework created by HL7 [9]. The FHIR standard defines a list of data models which can represent a wide range of healthcare related features, both clinical and administrative. Instances of these data models, which are named resources, are used to exchange and/or store the data using different serialization formats (e.g. XML or JSON). These resources can easily be assembled into working systems making them suitable for use in a wide variety of contexts e.g. mobile phone apps, EHR-based data sharing, server communication in large institutional healthcare providers, etc [10]. All these features make it the perfect choice for data sharing and storing in the microservice chatbot architecture.

Regarding chatbot-user interaction, the chatbot should be able to process the user input and, through natural language processing, to obtain a response as consistent as possible with the user conversation. Nowadays, there are many advances in natural language processing that offer tools and standards to allow fluent interactions with users. Among them, one appears to have a widespread acceptance, Artificial Intelligence Markup Language (AIML) [11]. AIML is a language based on XML that serves for the development of software agents and adds the capability to communicate with users in natural language. Thanks to the abilities of natural language processing, it is

possible to obtain valuable health data from the user-bot conversation. Thus, AIML seems the perfect choice for modeling interaction in our proposed architecture.

In this paper, we introduce a chatbot architecture for chronic patient support grounded on three pillars: scalability by means of microservices, standard data sharing models through HL7 FHIR and standard conversation modeling using AIML.

The remainder of the paper is structured as follows. Section 2 provides a review of related works. In Section 3, a complete description of the proposed architecture is presented. Section 4 describes how the validation of the architecture has been performed and also presents the developed prototype in a specific healthcare scenario. Section 5 provides an analysis of the main challenges of the proposed architecture. Finally, we present the conclusions and future work in Section 6.

2. Related work

Some research works propose the use of software architectures for eHealth scenarios which are not primarily intended for them. Catarinucci et al. [12] propose an IoT-Aware Architecture that uses an IoT smart gateway (containing a two-way proxy, management application and secure access manager) to connect the hybrid sensing network with the user interfaces. In [13], a microservice architecture for an Internet of Things healthcare scenario is proposed to provide future requirements of scalability and resilience as the failure of a service should not adversely affect the overall system. The fundamental use cases of care provision have each action directly designed

as a microservice in a microservice architecture. The microservices are described in [13], but no technical details are given about the link between the proposed system architecture and the microservices. Furthermore, no privacy protocols are studied. Another approach [14] describes a microservice-based platform that uses activity trackers to provide a monitoring solution for health-related data. This is an interesting approach providing elasticity and scalability thanks to the microservices, but no security features are taken into consideration and there is no information about the data standardization for medical data storage. In [15], an IoT platform is presented in which a set of recommendations microservices is proposed for depressive disorders. A very attractive microservice model has been implemented, but the authors do not provide information about user interaction with the system and security issues.

The interest in using chatbots in eHealth environments is growing and attracting a lot of attention [2], [16], [17], [18], [19]. Recent advances in chatbots show that they can improve the efficiency of healthcare delivery by performing clinical tasks that can be automated. Fadhil et al. [20] propose an AI-chatbot for delivering support to nutrition education. From this work it can be concluded that chatbots have a lot of advantages in the eHealth domain both for healthcare providers and patients, but no information about the architecture used is provided. Other healthcare chatbots like HealthBot [21] and Your.MD [16] are symptom checkers, providing the user with medical advice and useful tips about different medical conditions. Again, no hint about their underlying architecture is provided. Tschanz et al. propose a chat-like smartphone app (eMMA) to manage patient medication through

a chatbot [22] focused on patient interactions and medication management. Also, in [3] a chatbot has been developed to provide patients with medication reminders and a health tracker using patient data gathering, but without providing end-to-end encryption. These chatbot applications follow the same pattern: they are tailored to a specific medical condition or activity and no implementation details are provided, making it hard to evaluate their potential in terms of modularity, standard data management and standard chatbot-user interaction. Among those chatbot proposal that provide implementation details, Augellio et al. [23] propose a web-based infrastructure for chatbots with a modular knowledge base. Even though the division into modules is of interest, the system needs an extra effort to activate and deactivate the modules and to reload the core to apply changes upon the modules. In [24] M. Yan et al. present a generic architecture for a chatbot framework built on top of a serverless computing platform. Although the approach is very interesting due to its decentralized nature, its serverless orientation together with the need to rely on IBM Watson services for chatbot-user interaction makes it hard to apply in healthcare scenarios.

Other research works have explored generic chatbot architectures. These architectures are composed of intent classification, entity recognition, candidate response generator and response selector [25]. An example of this type of chatbot architecture in healthcare scenarios can be observed in [26], where a chatbot is developed for weight control and health promotion. The work focuses on how the message is processed to obtain the user response, rather than giving more information on how the software distribution is addressed to give flexibility and modularity to the system.

To address the remaining challenges for providing modularity, standard data exchange and standard user interactions, we propose a complete chatbot architecture based on microservices, HL7 FHIR and AIML to give support in healthcare environments focused on chronic patient care.

3. Microservice architecture

This paper proposes the application of a microservices-based architecture in the design of a chatbot architecture for healthcare scenarios (see Fig. 1). The proposed architecture is intended to provide new services and functionalities for evolving user needs. The logic based on microservices in the proposed architecture serves to process the user information and perform automatic tasks to provide scalability in a healthcare chatbot ecosystem. Furthermore, modularity is covered by the proposed microservice structure, allowing a new microservice to be built independently and without needing to modify the developed chatbot.

The standardization of user data and the modeling of user conversations are added within the proposed microservice structure, to provide interoperability to the system. The structure of the architecture allows having different types of databases depending on the necessities of the chatbot storage. In order to be able to manage the entire amount of data that is generated and provide interoperability between different healthcare systems, the architecture supports EHR for medical data storage. Furthermore, each microservice has its own small database to store all the information needed during the run time.

The proposed architecture includes microservices intended for the correct

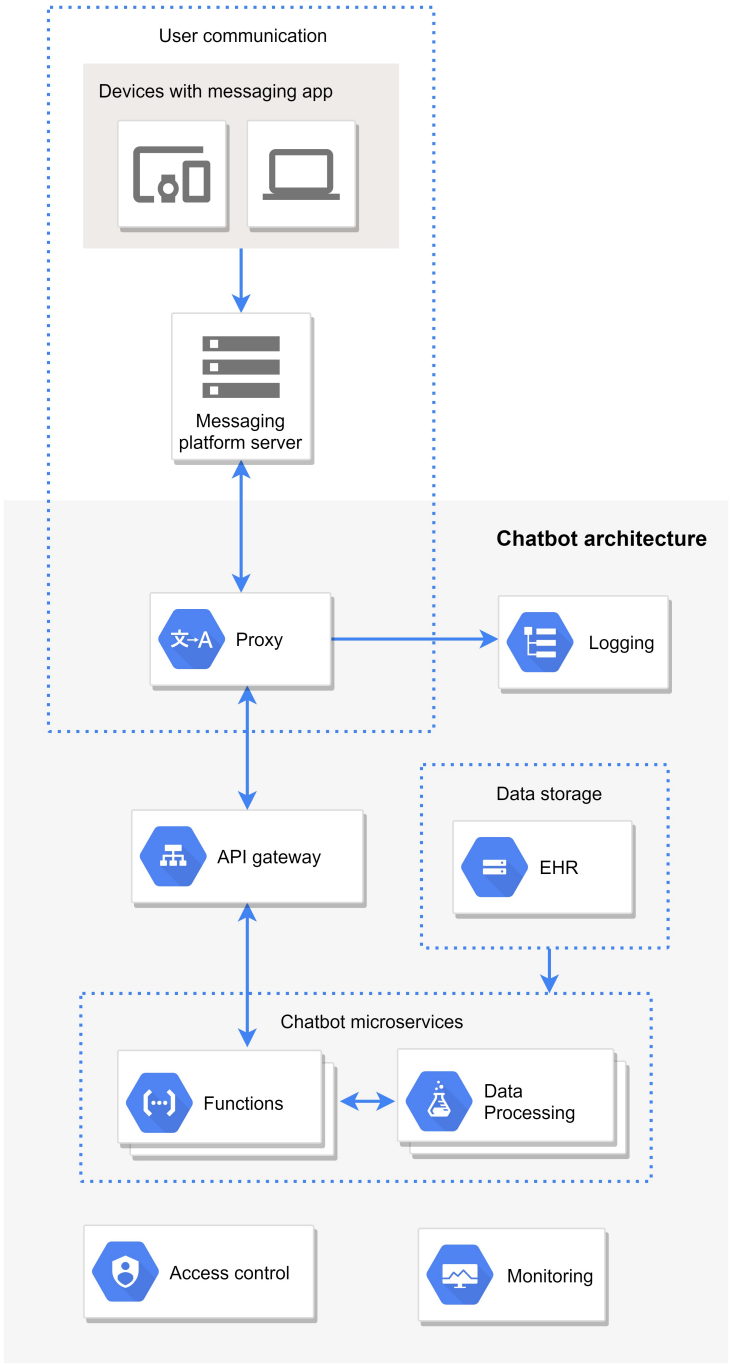


Figure 1: Overview of the chatbot internal structure.

functioning of the system (monitoring, authentication, and logging). One microservice is dedicated to checking if the system is running correctly. Another microservice is dedicated to providing authentication between all the microservices running in the platform. Finally, there is a microservice that provides logging facilities. The number of microservices can be increased according to the needs that arise within the system.

The coordination and distribution of the system tasks among the microservices of the architecture is essential to be able to perform a fluent user-bot interaction. The architecture has two core microservices: the proxy and the API gateway. The proxy translates the message received from the user through the messaging platform into the internal standard format of the architecture (e.g., a Telegram format message translated into the internal standard modeled in the JSON Schema that contains user information, message body, attachment content, and timestamp). The API gateway is the smart gateway that decides, with the information obtained from the message body and the current state information of the system, which chatbot microservice is best suited to receive the user message. Apart from the core microservices, there is a chatbot microservices pool where each microservice has independent and specific tasks to perform. The chatbot microservices pool allows the chatbot to offer different functionalities to the user. Each microservice is completely independent from others, giving the opportunity to personalize the scenario by developing specific tasks for each group of patients' needs. Moreover, an advantage of using microservices is that new functionalities can be tested separately of the rest of the functionalities, added into the system without causing any disturbance to the user (the prototype

does not need to be restarted in order to add new pieces of code into the microservice architecture, in contrast with monolithic architectures). One example of functionality is the microservice that allows users to check the opening hours and the location of their health clinic. Within the chatbot microservices, there are two different types: functions (microservices that have an interaction with the users) and data processing (microservices that perform more complex tasks, such as processing the relevant information of an image).

The microservices are based on an internal structure that consists of three main components designed for this architecture (that may appear in the microservice or not). The first component of the microservice is the part that communicates with other microservices. The communication uses JavaScript Object Notation (JSON) to exchange data between all microservices. Furthermore, the communication is established using the secure version of HTTP (Hypertext Transfer Protocol Secure, HTTPS). The second component is the interaction with the databases of the architecture. These interactions use HTTPS for the communication, in some cases using the standardization needed for some specific data types. More specifically, FHIR is used in the architecture for medical exchange data. The last component is the part of the microservice dedicated to the interaction with the user. This interaction is designed with workflows and modeled with AIML.

4. Architecture validation

In [27], we studied twelve different chronic diseases in order to obtain the data gathering requirements so as to implement a complete telemonitor-

ing scenario based on ontologies. This work has provided us with a highly valuable study in order to know the data gathering requirements as well as patient interactions for a very complete set of chronic conditions.

These requirements were obtained working with primary care physicians, creating a questionnaire that contains all the needs for the supervision of each chronic disease. Moreover, we have added additional requirements for psoriasis with the collaboration of dermatologists and their patients. A summary of these needs is shown in Table 1.

Studying the needs of different chronic diseases, some common functionalities can be obtained, as well as specific functions for concrete illnesses. The need of monitoring some vital constants such as weight or height are integrated into the same microservice, called *Questionnaire*, which can ask periodically the users' vital constans, giving the possibility to monitor their data. We have modeled this microservice to allow physicians to create customized questionnaires, generating automatically the alerts and the interaction with the patients. Other more specific microservices, such as one that offers deeper tools in monitoring the quantity and the type of food eaten, are not included in the generic prototype because not all diseases need it.

In order to validate the viability of the architecture, we have developed a generic virtual assistant with the most common functionalities. The rest of specific functionalities can be independently and easily added to the generic virtual assistant developed. To validate the flexibility of the scenario, after building a generic virtual assistant, we have modeled and developed three functionalities that are specific for psoriasis scenarios, described in the following sections.

4.1. Proof of concept: prototype development

This proof of concept aims to create a generic virtual assistant including the most common functionalities for chronic disease management. Afterwards the following section shows how the virtual assistant can be easily adapted to an specific disease, in this case psoriasis.

In a health environment scenario, the most important feature to consider while choosing the messaging platform is to guarantee the privacy of the data that is exchanged between the user and the chatbot. Signal [28], with its end-to-end encryption, is used as a messaging platform in this prototype. The Proxy microservice is the only microservice that interacts directly with the messaging platform server, as shown in Figure 1. In this proof of concept, the microservice Proxy sends and receives the client messages through the Signal server, using the Java libraries signal4j [29] version 1.0.4 and signal-service-java [30] version 2.3.1_unofficial_1 (both libraries have been modified because they are not updated with the latest changes of the Signal official repository [31]). The Proxy microservice has been developed in Android, based on a sample bot [32]. Despite having developed the proof of concept with the Signal messaging platform, the Proxy is designed so that new messaging platforms can easily be added to the architecture.

The pool of microservices for the chatbot architecture developed for this proof of concept is described in Table 2. As mentioned before, we have the *Questionnaire* microservice to ask users about their healthcare data. Also, generic microservices such as *Register*, *Unsubscribe* or *Verification* are included due to the necessities associated to the use of the virtual assistant. *Specialist* and *Patients* help to see the different actors that use the platform.

Another tool that is shared between all chronic diseases is *Appointment* to manage the appointments with the healthcare professionals.

The microservices in this project have been developed in two principal programming languages: Java (using Java 8 Update 121) and Python (using Python 3.7). The microservices developed in Java use the open source Jersey framework supporting JAX-RS APIs to implement the RESTful service and client development, and the Grizzly framework to implement the HTTPS server. The microservices developed in Python use the Flask framework for developing small server applications and Gunicorn is used to serve the Flask application at a production level. All the microservices developed in this work have an asynchronous HTTPS server that relies on TLS for clients authentication. In order to provide a way to add new microservices easily into the generic chatbot, we have created a template in Java and another in Python (the templates in both programming languages are available in `https://github.com/ehealthz-lab`). Thanks to these templates that contain the basic code to develop a microservice which works in the proposed architecture, new functionalities related to specific illnesses could be added into the generic chatbot. The developer needs to model the conversation flow and create the AIML file with the conversation of this new functionality of the generic chatbot. After, the AIML file and the programmed tasks of this specific microservice are added in the template, creating the new microservice. We have used these templates with three students, that have developed new functionalities inside the chatbot without a deep knowledge of the architecture itself, obtaining good results about the easiness of the usage of the templates.

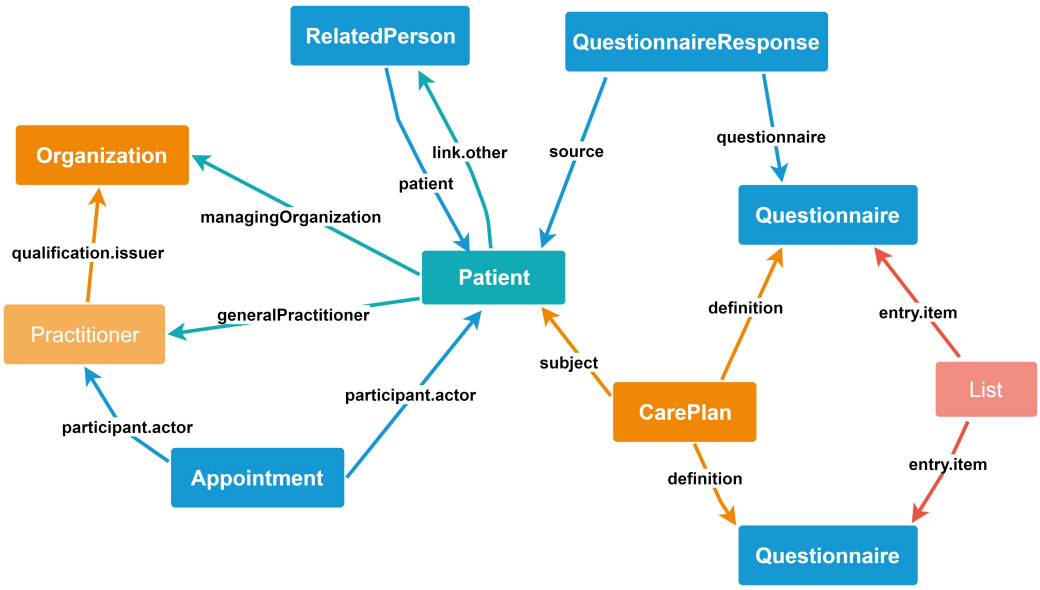


Figure 2: Relations of the FHIR resources.

We use the Docker platform to build lightweight containers, each one with a different microservice of the architecture, allowing dynamic deployment. This allows us to have a modular, independent and flexible microservice architecture with the possibility of adding new microservices without compatibility problems with other programming languages or versions.

In order to provide interoperability with other eHealth systems, the user healthcare information is stored in FHIR resources, that contain all the relevant data for each case. The FHIR resources used to cover the information storage needs in this scenario are shown in Fig. 2. The user information is stored in *Patient*, *Practitioner* and *RelatedPerson* resources, depending on the type of the user. The other resources shown in Fig. 2 are for storing the relevant information that each user provides during the conversation with the chatbot.

AIML version 2.0 is used in this proof of concept as a language to define the chatbot behavior. The AIML interaction is divided between the different microservices of the architecture. This separation of the conversation among different microservices generates the problem of knowing to which microservice any message is directed in order to obtain the user's response from his AIML file. To solve this problem, we propose an expansion of AIML with a new element called *Microservice*. This new element, *Microservice*, can be found within the AIML tag *Category*, providing the necessary information to know if this base unit of knowledge is in the middle of a conversation with the user or the conversation has ended to allow the user to initiate a new conversation with another microservice.

In order to provide personalization in the conversation with the user based on the information stored in the FHIR resources, the creation of AIML files must be dynamic and flexible to adapt to each specific case. The AIML files used generically in the chatbots are created in a static manner, not offering this degree of personalization based on the information stored in databases. This study tries to solve this problem by offering an AIML file generator that, based on the FHIR resources available in the database, generates a new file with the interaction to be produced with the user, resulting in a more personalized interaction.

The use of this AIML file generator can be useful in generating a user interaction for a questionnaire. This questionnaire is stored in the database with the statement of the questions and the type of data stored for each question as a *Questionnaire* resource (an example is shown in Fig. 3). This *Questionnaire* resource can be created by the doctor or by another user. The

AIML file generator is able to read this resource and create a new file with the user interaction where the user has to fill out the complete questionnaire. An example of an AIML file created by the AIML file generator is shown in Fig. 4. This file is generated from the resource *Questionnaire* shown in Fig. 3. The file contains the user-bot interaction generated from the questionnaire. First, when the user asks for the questionnaire, the chatbot asks the first question to the user (in this case, "How many hours did you sleep last night?"). Then, with the response of the user, the chatbot saves the response thanks to the oob tag, knowing that it should be a decimal answer ("<oob>SAVE_DECIMAL</oob>"). Then, the chatbot finishes the questionnaire because there are no more questions in the FHIR resource. All this conversation is automatically created thanks to the AIML file generator.

This AIML generator has been built based on the patterns used to build AIML files. These patterns have been programed in Java (using Java 8 Update 121) to be able to create the conversation flow in the case of surveys. Each *Questionnaire* resource is linked to its AIML file by the name of the file and also with the first pattern of the first category of the AIML file, where the *id* of the resource in FHIR is set to know which AIML file should use the virtual assistant (as shown in Fig. 3, line 3, with the FHIR id 45467). Also, the *Microservice* AIML tag extension is used to know when the user is in the middle of a conversation and when the questionnaire is finished.

In this example, we have shown how the translation from a FHIR resource to an AIML file is performed, showing one of the many cases in which this generator can be used.

All the user interaction and the usage and the state of the microservices

```
{
  "resourceType": "Questionnaire",
  "id": "45467",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2019-01-14T10:14:25.000+00:00"
  },
  "title": "Sleep quality",
  "subjectType": [
    "Patient"
  ],
  "item": [
    {
      "linkId": "1",
      "text": "How many hours did you sleep last night?",
      "type": "decimal",
      "required": true
    }
  ]
}
```


Figure 3: FHIR resource example: Questionnaire.


```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="2.0">

<category><pattern>QUESTIONNAIRE45467</pattern>
<template>How many hours did you sleep last night?</template>
<microservice>questionnaire</microservice>
</category>

<category><pattern>*</pattern>
<that>How many hours did you sleep last night?</that>
<template>Thanks for answering the
questions<oob>SAVE_DECIMAL</oob></template>
</category>

<category><pattern>*</pattern>
<template><random>
<li>Sorry, I have not understood you</li>
<li>Can you repeat it to me in another way?</li>
</random></template>
</category>
</aiml>
```


Figure 4: AIML example: Questionnaire auto-generated.

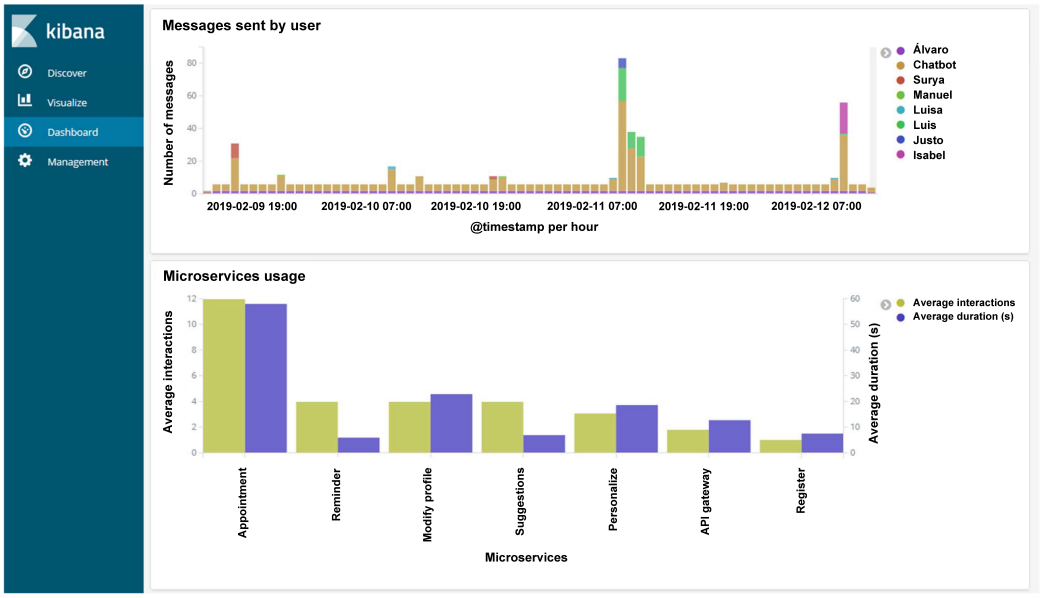


Figure 5: Logging snapshot: users' messages and microservices usage.

can be shown in the monitoring microservice thanks to the logging facility. In this prototype, Kibana [33] has been used as a tool to display the relevant information such as, for example, the messages sent over time or the average usability values of microservices (shown in Fig. 5).

The privacy of the data is preserved due to all the data exchanged in the scenario are encrypted. Because of Signal is used as a messaging platform in the scenario, the conversation between the end-user and the virtual assistant has end-to-end encryption. Inside the architecture proposed, all the data exchange between microservices use the HTTPS protocol, which encrypts the data, providing privacy. All the sensible data related to personal data and illnesses are stored in FHIR, which uses the HTTPS protocol to exchange the information and provide privacy. The FHIR data are stored in an encrypted hard disk. The periodic backups are configured to cipher the data and save

them in an external encrypted hard disk. Applying all these measures of privacy and user's rights, we follow both national data protection law LO 03/2018 [34] and European GDPR [35] due to the sensitivity of the data stored in the system.

4.1.1. Psoriasis profile

The psoriasis prototype aims to demonstrate the feasibility of just one of the several possible use-case scenarios for the proposed microservice chatbot architecture. The new chatbot seeks to support and help chronic psoriasis patients by providing specific tools for psoriasis monitoring. Patients have the possibility of saving images of the body surface area through the chatbot which is then able to show the saved images over time. Thanks to these functionalities, patients, caregivers and healthcare professionals are able to see the evolution of the affected skin area and how it reacts to the treatment.

The functionalities are designed with the collaboration of a group of dermatologists, which are given their needs and opinion about what they want to have in the virtual assistant to give support to psoriatic patients. In order to obtain the specific functionalities needed in a psoriasis scenario, we have followed two steps. First, we have asked what are the parameters that the dermatologists usually monitor in their medical consultations, and we have obtained the questionnaires that they use to ask periodically to their patients. Then, with this information, we have designed a group of functionalities and discussed them with the dermatologists, obtaining the final functionalities specially designed for the psoriasis scenario. The microservices that we have added to the generic virtual assistant are shown in Table 3. We have added new monitoring tools that helps with tasks which were very difficult to per-

form before, the image storage of the affected areas. The chatbot provides an improvement in the facility to store images and watch them in a timeline, in order to see the evolution of the affected area. Essentially, the psoriasis disease is addressed through the *Image*, *Record* and *Query* microservices. The FHIR resource "Media" has been added into the scenario, to store the images generated from the users.

Some possible functionalities that can be added into the scenario, to fulfill more needs from other diseases could be a food tracker with a specific tool that counts the calories in obesity scenarios or a sleep monitor. All these new functionalities could be added following the needs proposed by the physicians of each chronic disease.

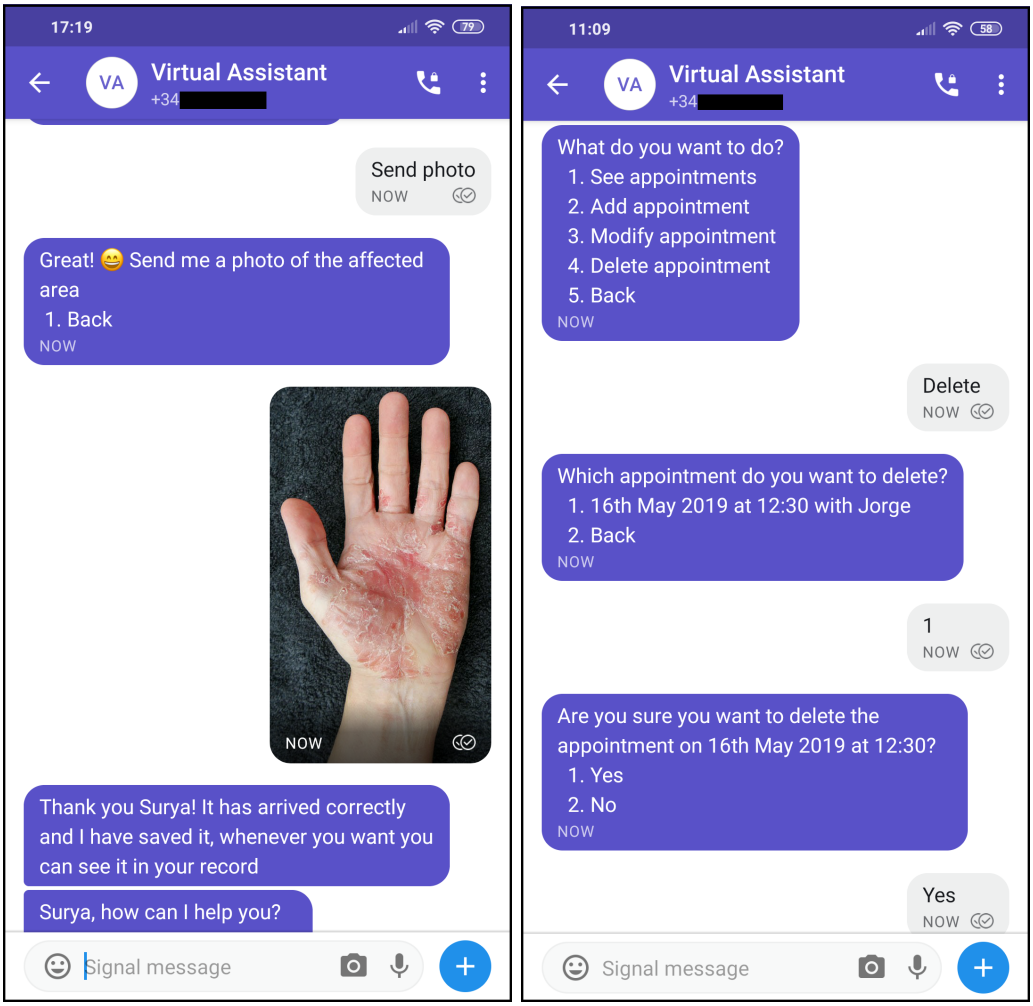
Finally, two usage examples are shown in Fig. 6, where a user interacts with the chatbot to store a photograph and to delete an appointment.

5. Analysis

Various aspects and challenges are addressed in the proposed architecture. We classify these in three categories: Modularity, Standardization and Security.

5.1. Modularity

The microservices architecture pattern accomplishes a level of modularity that in practice is extremely difficult to achieve with a monolithic architecture [36]. An important point in microservice architecture is the service size. In the proposed architecture, the service size offers a chatbot split into tasks called functionalities. These functionalities are offered to users in a main



(a) Image example. (b) Appointment example.

Figure 6: Real interaction with the chatbot implemented.

menu with numeric options. The menu can adapt in real time to the functionalities running at that moment, to add modularity in the expansion of the microservices. Thanks to this feature, new microservices related to new functionalities can be added dynamically and independently in the architecture.

Furthermore, when a microservice with interaction is incorporated to the running system, the interaction is added inside the new Docker created for that specific microservice. The new interaction can be added independently as a behavior on the chatbot even when the core system is running thanks to the new element, called *Microservice*, in AIML syntax discussed above. This provides the possibility of splitting the chatbot conversation in different components in the system architecture, knowing which AIML file the program should search for the user response.

5.2. Standardization

Standardization in scenarios where data is stored and used is fundamental to guarantee the interoperability between different systems. Furthermore, standardization is fundamental to guarantee robustness against the difficulties faced during the deployment of the architecture. The main drawback is the initial effort needed from the developers to learn and adapt the system to the different standards used.

In a patient support scenario, medical data integration is required to guarantee interoperability. The proposed chatbot architecture supports the storage and sharing of medical information strictly aligning with HL7 FHIR standards. Furthermore, the standard AIML has been used as the conversation modelling to give homogeneity in the development of the interaction of

```
{
  "type": "object",
  "properties": {
    "user": { "type": "string",
      "description": "Unique user identifier." },
    "platform": { "type": "string",
      "description": "Indicates the platform through which the
      communication is being made." },
    "message": {
      "type": "object",
      "properties": {
        "timestamp": { "type": "integer",
          "description": "Includes all the relevant information of the
        message." },
        "attachments": {
          "type": "object",
          "properties": {
            "contentType": { "type": "string",
              "description": "Specifies the type of attachment." },
            "data": { "type": "string",
              "description": "Contains the attachment (base64 encoded)." },
            "size": { "type": "integer",
              "description": "Number of bytes of content." }
          }
        },
        "body": { "type": "string",
          "description": "Contains the text of the message." }
      },
      "required": [ "timestamp" ]
    }
  },
  "required": [ "user", "platform", "message" ]
}
```


Figure 7: JSON Schema: message format.

the user with different microservices. Finally, a JSON Schema is proposed to standardize the communication between the microservices that require interaction with the user. A schema is shown in Fig. 7.

5.3. Security

Besides all the things that have to be taken into account in any system, such as general security considerations (e.g. database configuration, known service vulnerabilities, etc.) and the specific hardening tactics required when using Docker (e.g. unprivileged container execution, per-container firewalling, etc.), some specific security issues have been considered in the proposed architecture. These issues are related with access control (authentication and

authorization) for both users and microservices, so that the privacy of the user's data meets the expected requirements.

Inside the architecture, services authentication relies on TLS for both client and server. Since all communications are secured using HTTPS and all services in the architecture play at least the role of server (most of them are also clients), each service already have its own digital certificate. Relative to service authorization, each service in the architecture is provided with an Access Control List (ACL) at configuration time. This list includes the services that would need to perform calls to its API in a normal way of operation.

Furthermore, a user's identity is verified as soon as a new message reaches the system. User authentication relies on signal's user management which is based on public key architecture where a user's identity key pair is generated at the time the application is installed and the public keys are exchanged the first time a communication is established between two users. To ensure that there are no man-in-the-middle attacks when the session is established, the users can check the chatbot's public key fingerprint which is shared with them beforehand. Once the system has verified the user identity, a JWT is issued which is included in further requests between services. Moreover, authorization is performed in a finer-grained way using the XACML framework for policies definition and evaluation.

This security and privacy solution is based on standard security mechanisms that are typically used in microservices architectures, providing a complete vision of how existent technology could be wellsuited to address security and privacy issues in the proposed architecture (critical when clini-

cal data are involved). Notwithstanding, the proposed solution is not unique since other standard solutions already exist to that end (e.g. OAuth) and adhoc mechanism might be designed to be used in place.

6. Conclusions and future work

The purpose of this study is to offer a solution based on microservices to provide personalized eHealth functionalities and data storage using virtual assistants in healthcare scenarios. The internal chatbot architecture allows the addition of new services and tools over time, splitting of the conversation and development of a wide range of healthcare functionalities as microservices. This architecture is designed to suit any chronic patient, overcoming the shortcomings of other mobile applications that are only intended for a specific type of disease [37], [38]. Furthermore, our approach makes some additional contributions in terms of automation to translate a FHIR resource into an AIML interaction file, making the conversations more personal. Standardization, security and fluent interaction are considered in the proposed architecture. The final chatbot architecture design integrates different possible healthcare scenarios, with the aim of providing telemonitoring of any disease through the use of messaging platforms and user-bot interaction. It provides the advantages of flexibility, modularity and expansiveness in a virtual assistant scenario, enabling the chatbot to be modeled following the specifications of patients' illnesses. Specifically, our architecture is validated and adjusted following a prototype developed for psoriasis patient care. The chatbot architecture and the use of microservices provide a good flexible solution for personalized monitoring services. This architecture has

been proposed to address the challenges of personalization and data storage in mHealth scenarios. The proposed chatbot can also be re-used for telemonitoring of other diseases as well.

A future extension to the proposed architecture may be the addition of speech recognition functionalities for better user-bot interaction.

The viability of the virtual assistant architecture has been validated through the development of a generic chatbot along with the extension for one concrete scenario. This specific scenario is the psoriasis profile, which provides a photographic record as the main feature for skin area tracking. Future work can be focused on adding advanced and automated image processing-based advice services or adding new functionalities for new chronic diseases into the generic chatbot. The objective of automation in the field of mHealth is to provide comfort and convenience when using medical services, since the tool can be used anywhere. This, in turn, can reduce costs in the healthcare sector while improving the quality of life of patients and reducing the number of face-to-face medical consultations.

To conclude, it is worth highlighting that the chatbot architecture proposed in this work covers the necessities of mHealth scenarios. The security measurements are addressed, as well as data storage and FHIR-AIML automation. Moreover, the proposed architecture has been designed for interoperability with any communication channel, such as messaging platforms or web interfaces, thanks to the proxy translator that is able to convert from the specific formats of communication channels into the general data format used within the chatbot architecture.

Acknowledgements

Research funded by Ministerio de Economía, Industria y Competitividad from Gobierno de España and European Regional Development Fund (TIN2016-76770-R and BES-2017-082017) and Gobierno de Aragón (Reference Group T31_17R) and FEDER 2014-2020 "Construyendo Europa desde Aragón".

References

[1] B. Chaix, J.-E. Bibault, A. Pienkowski, G. Delamon, A. Guillemassé, P. Nectoux, B. Brouard, When chatbots meet patients: One-year prospective study of conversations between patients with breast cancer and a chatbot, JMIR Cancer 5 (1) (2019) e12856. doi:10.2196/12856.
URL http://cancer.jmir.org/2019/1/e12856/

[2] Symptomate, last accessed 2019/01/21.
URL https://symptomate.com/

[3] Florence, last accessed 2019/02/18.
URL https://www.florence.chat/

[4] Forksy, last accessed 2019/02/18.
URL https://getforksy.com/

[5] A. Palanica, P. Flaschner, A. Thommandram, M. Li, Y. Fossat, Physicians' perceptions of chatbots in health care: Cross-sectional web-based survey, J Med Internet Res 21 (4) (2019) e12887. doi:10.2196/12887.
URL https://www.jmir.org/2019/4/e12887/

[6] E. Wallace, C. Salisbury, B. Guthrie, C. Lewis, T. Fahey, S. M. Smith, Managing patients with multimorbidity in primary care, BMJ (Online) 350 (January) (2015) 6–11. doi:10.1136/bmj.h176.
URL http://dx.doi.org/doi:10.1136/bmj.h176

[7] J. Pereira, Ó. Díaz, Using health chatbots for behavior change: A mapping study, Journal of Medical Systems 43 (5) (2019) 135. doi:10.1007/s10916-019-1237-1.
URL https://doi.org/10.1007/s10916-019-1237-1

[8] C. Marcos, A. González-Ferrer, M. Peleg, C. Cavero, Solving the interoperability challenge of a distributed complex patient guidance system: A data integrator based on HL7's Virtual Medical Record standard, Journal of the American Medical Informatics Association 22 (3) (2015) 587–599. doi:10.1093/jamia/ocv003.

[9] Fhir, last accessed 2018/11/07.
URL http://www.hl7.org/FHIR/

[10] J. Ruminski, A. Bujnowski, T. Kocejko, A. Andrushevich, M. Biallas, R. Kistler, The data exchange between smart glasses and healthcare information systems using the hl7 fhir standard, in: 2016 9th International Conference on Human System Interactions (HSI), 2016, pp. 525–531. doi:10.1109/HSI.2016.7529684.

[11] R. S. Wallace, The elements of AIML style, Alice AI Foundation (2003). doi:10.1.1.693.3664.

[12] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, L. Tarricone, An iot-aware architecture for smart healthcare systems, IEEE Internet of Things Journal 2 (6) (2015) 515–526. doi:10.1109/JIOT.2015.2417684.

[13] R. Hill, D. Shadija, M. Rezai, Enabling community health care with microservices, CoRR abs/1709.07037 (2017). arXiv:1709.07037.
URL http://arxiv.org/abs/1709.07037

[14] O. OBrien, R. D. OReilly, Beats-per-minute (bpm): A microservice-based platform for the monitoring of health related data via activity trackers, in: 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), 2018, pp. 1–7. doi:10.1109/HealthCom.2018.8531169.

[15] S. Ali, M. G. Kibria, M. A. Jarwar, S. Kumar, I. Chong, Microservices model in woo based iot platform for depressive disorder assistance, in: 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 864–866. doi:10.1109/ICTC.2017.8190800.

[16] Your.md, last accessed 2018/09/25.
URL http://www.your.md

[17] ada, last accessed 2019/01/21.
URL https://ada.com/

[18] mediktor, last accessed 2019/01/21.
URL https://www.mediktor.com/en

[19] Healthtap, last accessed 2019/01/21.
URL https://www.healthtap.com/

[20] A. Fadhil, S. Gabrielli, Addressing challenges in promoting healthy lifestyles: The al-chatbot approach, in: Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth '17, ACM, New York, NY, USA, 2017, pp. 261–265. doi:10.1145/3154862.3154914.
URL http://doi.acm.org/10.1145/3154862.3154914

[21] Healthbot, last accessed 2018/09/25.
URL https://healthbot.in

[22] M. Tschanz, T. L. Dorner, J. Holm, K. Denecke, Using emma to manage medication, Computer 51 (8) (2018) 18–25. doi:10.1109/MC.2018.3191254.

[23] A. Augello, M. Scriminaci, S. Gaglio, G. Pilato, A modular framework for versatile conversational agent building, in: 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, 2011, pp. 577–582. doi:10.1109/CISIS.2011.95.

[24] M. Yan, P. Castro, P. Cheng, V. Ishakian, Building a Chatbot with Serverless Computing, Proceedings of the 1st International Workshop on Mashups of Things and APIs - MOTA '16 (2016) 1–4doi:10.1145/3007203.3007217.
URL http://dl.acm.org/citation.cfm?doid=3007203.3007217

[25] A. M. Rahman, A. A. Mamun, A. Islam, Programming challenges of chatbot: Current and future prospective, in: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2017, pp. 75–78. doi:10.1109/R10-HTC.2017.8288910.

[26] C. Huang, M. Yang, C. Huang, Y. Chen, M. Wu, K. Chen, A chatbot-supported smart wireless interactive healthcare system for weight control and health promotion, in: 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2018, pp. 1791–1795. doi:10.1109/IEEM.2018.8607399.

[27] N. Lasierra, A. Alesanco, S. Guillén, J. García, A three stage ontology-driven solution to provide personalized care to chronic patients at home, Journal of Biomedical Informatics 46 (3) (2013) 516–529. doi:10.1016/j.jbi.2013.03.006.
URL http://dx.doi.org/10.1016/j.jbi.2013.03.006

[28] Signal, last accessed 2018/08/21.
URL https://signal.org/

[29] A facade to make using the libsignal-service easy, last accessed 2019/01/11.
URL https://github.com/Turakar/signal4j

[30] A java library for communicating via signal, last accessed 2019/01/11.
URL https://github.com/Turasa/libsignal-service-java

[31] Signal oficial repository, last accessed 2019/01/11.
URL https://github.com/signalapp

[32] A bot for signal, last accessed 2019/01/11.
URL https://github.com/nerdclub-tfg/signal-bot

[33] Kibana, last accessed 2019/02/18.
URL https://www.elastic.co/products/kibana

[34] Constitutional law 3/2018 of the 5th december concerning protection of personal data and guarantee of digital rights, last accessed 2019/03/20.
URL https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673

[35] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016, last accessed 2019/03/20.
URL https://eur-lex.europa.eu/eli/reg/2016/679/oj

[36] C. Richardson, F. Smith, Microservices - From Design to Deployment, Nginx (2016) 80.

[37] Psoriasiscalc, last accessed 2019/01/30.
URL https://play.google.com/store/apps/details?id=org.dipler.psoriasiscalc.android

[38] Diabetes diagnostics, last accessed 2019/01/30.
URL https://play.google.com/store/apps/details?id=uk.ac.exeter.DiabetesDiagnostics

Profile	**Needs to ask**
Chronic Obstructive Pulmonary Disease (COPD)	Height, weight, pulse rate, blood pressure, SpO2, FEV1, glucose, medication, number of cigarettes, patient location, room temperature, room humidity, health test questionnaire (COPD specific)
Obesity	Height, weight, pulse rate, blood pressure, SpO2, glucose, medication, liquids quantity, food quantity, body fat, body water, health test questionnaire (Obesity specific)
Thyroid Disorders	Height, weight, pulse rate, blood pressure, SpO2, temperature, glucose, medication, liquids quantity, food quantity, health test questionnaire (Thyroid specific)
Ischemic heart disease (IHD)	Height, weight, pulse rate, blood pressure, SpO2, Glucose, medication, number of cigarettes, health test questionnaire (Ischemic specific)
Asthma	Height, weight, pulse rate, blood pressure, SpO2, FEV1, Temperature, Glucose, patient location, room temperature, medication, number of cigarettes, room humidity, health test questionnaire (asthma specific)
Hypertension (HTA) or high blood pressure	Height, weight, pulse rate, blood pressure, SpO2, Glucose, medication, number of cigarettes, health test questionnaire (HTA specific)
Osteoporosis	Height, weight, medication, health test questionnaire (Calcium), health test questionnaire (Falls)
Heart failure (HF)	Height, weight, pulse rate, blood pressure, SpO2, medication, liquids quantity, health test questionnaire (Liquids), health test questionnaire (HF)
Diabetes mellitus	Height, weight, pulse rate, blood pressure, Glucose, medication, liquids quantity, food quantity, number of cigarettes, HbA1c, health test questionnaire (Diabetes)
Dyslipidemia	Height, weight, pulse rate, blood pressure, medication, number of cigarettes, health test questionnaire (Dyslipidemia)
Osteoarthritis	Height, weight, blood pressure, temperature, medication, number of cigarettes, health test questionnaire (Osteoarthritis)
Psoriasis	Height, weight, pulse rate, medication, patient location, sleep hours, stress level, health test questionnaire (Psoriasis)

Table 1: Chronic patient profiles with their needs

Microservice	**Tasks**
Proxy	1. Pull messages from Signal server 2. Translate the messages into a formatted JSON 3. Send the messages to the API gateway
API gateway	1. Choose the best microservice to send the message to 2. Forward the message to the chosen microservice
Verification	Register the user in the chatbot
Personalize	Enable or disable the functionalities in the menu
Modify profile	Modify personal data
Reminder	1. Check the reminder queue 2. Send the reminders to the users
Unsubscribe	Delete all the data related to the user
Specialist	1. Show the specialists list 2. Change the data permissions for the specialists
Patients	1. Show the permissions granted to a specialist 2. Show a summary of the patient's activity
Register	Register, by another user, a new user in the chatbot
Appointment	1. Show the medical appointments 2.Add new medical appointments 3. Modify medical appointments 4. Delete medical appointments 5. Set reminders related to the appointments
Questionnaire	1. Create questionnaires 2. Modify questionnaires 3. Delete questionnaires 4. Fill in questionnaires 5. Generate AIML files based on questionnaires 6. Set reminders related to fill in the questionnaire
Suggestions	Receive the feedback from users

Table 2: Proof of concept: generic chatbot

Microservice	**Tasks**
Image	1. Store the user's images in the FHIR database
Record	1. Display the images that are stored in the database
Query	Redirect the questions that patients have about their illnesses to the dermatologists

Table 3: Proof of concept: psoriasis chatbot