DOI: 10.2478/ijasitels-2020-0007

FitPi: Wearable IoT solution for a daily smart life

Sabin PRUNA[1], *Anca VASILESCU*[2]

[1]*Transilvania University of Brasov, Romania, sabinpruna@gmail.com*

[2]*Transilvania University of Brasov, Romania, vasilex@unitbv.ro*

Abstract

The extensive implementation of Internet of Things (IoT) solutions and its wide popularity to the public over the past decade enabled emergent applications that provide sophisticated, proactive health care solutions that can improve the quality of life of individuals. This work proposes an IoT architecture and implements a prototype solution that allows its users to improve their physical activity by collecting vital signs using wearables and other environmental and habitual information in order to monitor their activity and propose behaviors in a smart way that will allow them to achieve their preset goals. The focus is on the increased usability of the system employing refined solutions like voice recognition and smart visualization to enable its seamless use while offering an interoperable architecture that will enhance its flexibility. The prototype implementation offers a proof of concept evaluation of the proposed system, applying state of the art technologies and using existing hardware and popular gadgets.

.

Keywords: matching pennies, Hagelbarger, sequence extrapolating robot, Shannon, mind reading machine, contextual predictor

1 Introduction

The Internet of Things or IoT represents any system of interconnected devices which communicate with one another through internet-based solutions [4]. As the technology matured, numerous branches stemmed from it. These comprise from supply chain management, retail, surveillance to health care, data mining, research or numerous other directions which filled in the technological gaps of societal needs. A very useful and exhaustive survey on architecture, enabling technologies, applications and challenges of the Internet of Things is published in [6], and it mentions our targeted domain here, healthcare, as the first one ready to benefit from IoT potentiality. The diversity and variety of the IoT-based applications increases over time, and the interest of the software developers community intensifies accordingly to the point of writing platforms for ”rapid development of IoT applications” like the TinyLink system presented in [7].

A relevant area to this paper is the wearable market which consists of small electronic devices that can be worn on the body, like regular accessories, embedded in clothes or even implanted in the user's body. As more and more devices become connected to the internet via smartphones or directly through embedded chips, the wearable market has gained traction for allowing the user to be connected to the Internet spectrum with just something they wear [1]. Ranging from smart glasses to peacemakers, wearables are trying to enhance and improve how we carry on our daily lives. Due to progress in this area, this paper targets the use smartwatches, as mobile devices with a touchscreen display designed to be worn on the wrist, to provide a top view of one's daily activities wherever as long as the user has an Internet connection. For the implementation, a Fitbit [5] fitness tracker was used from the Versa lineup. This device already provides a plethora of fitness data, as this is its main purpose, however it lacks certain features related to lifestyle.

This paper aims to present an IoT-based system for supporting people who pay attention to their own healthcare and are moving from personal care towards a smart life. The purpose and, in addition, the novelty of our project are expressed in terms of providing a complete and unique project that covers the most important needs of a modern user by valuing the features of many hardware components and software technologies and tools. Having a portable hardware prototype and two interfaces, one website-based and one wearable, which are ready to communicate with the user about their current personal life status, the present FitPi is an original smartstyle solution. Moreover, it could be welcomed as a vector of IT&C for economics and life sciences since its implementation targets to improve the quality and standards of life from many perspectives, such as health, food, budget, working time log.

Our project, FitPi, has been developed mainly from a student perspective, as a wearable application that allows the user to quickly collect habitual and environmental information, and digest data using lists related to their logged records of personal health indices, money spent, food intake, as well as having measurements of the temperature in their choose enclosure and others. An interesting "framework for designing material representations of physical activity data" is presented in [9] as Shelfie, and it is connected with our project by the on-screen visualizations of physical activity data.

There are, evidently, numerous powered devices, at a higher price range, that integrate into different operating systems ecosystems and there are also options with a more sensible data-driven functionality such as the eButton proposed in [16]. All these solutions can measure many health data, as our FitPi project does as well, but, FitPi is distinguished by offering one application for processing the collected data and two synchronized solutions for interfacing these data. Therefore, the FitPi solution has many important characteristics, for example, it is cheaper, it is focused on daily and accurately tracking physical data, and it is smart-oriented in terms of involving its user in stepping to smart life.

Considering other similar Fitbit-based applications, we have remarked the Fitbit Garden app [3] designed to enable children to engage in more physical activity. Beyond this similarity of having the daily physical activity as a common goal, the FitPi app will provide the Fitbit collected data to a broader range of lifestyle features, from the wearable and weather station to the website and API. The IoT-based architecture adopted for developing the FitPi project with AWS cloud support [2, 13] has passed the test of analyzing other solutions, for example, the IBM- based weather station described in [10]. From the hardware point of view, our FitPi prototype accomplishes the necessary IoT infrastructure using boards, sensors, actuators, displays, as well as other interconnecting components described in the next sections. Combining the prefix from Fitbit Inc. name and the suffix from the Raspberry Pi board name, we have the name FitPi for our original project.

2 Platforms and tools as FitPi background

An important part of this paper context is around the notion of microservices. Such services are an approach to application development in which a large monolith product is built as a suite of modular services. ”Each module supports a specific business goal and uses a simple, well- defined interface to communicate with other sets of services” [14]. In SOA (Services Oriented Architecture), services use protocols describing how to transmit and parse messages using description meta-data. This meta-data describes both the functional features of the service and the characteristics of its service quality. Services-oriented architecture aims to allow users to combine large pieces of functionalities to form applications that are built exclusively from existing services and combine them in an ad hoc manner. A service presents a simple interface for the applicant that abstracts the essential complexity. Additional users can also access these independent services without knowing their internal implementation. A service consists of two parts: the Web Part where the API, models, and everything needed for communicating with a client, and the database section for persisting user data that contains only a Docker file so that the service can be containerized.

From the practical point of view, our results described here are based on the Fitbit submission. Fitbit Inc. is an American company headquartered in San Francisco, California. Its products ”are activity trackers, wireless-enabled wearable technology devices that measure data such as the number of steps walked, heart rate, quality of sleep, steps climbed, and other personal metrics involved in fitness.” [5] FitbitOS represents the patented operating system of the Fitbit company running on devices of different types, for which one can develop apps using the specific SDK.

Amazon Web Service has provided another significant support for our approach as it affords the cloud infrastructure needed to host applications like ours. The specific services can be accessed locally, but to be used by the client applications, they must be published in an online environment, facilitating access to them. For the FitPi project, we chose the Amazon Web Services ecosystem as it is one of the most popular platforms in the area, using the EC2 (Elastic Compute Cloud) and AWS (Amazon Web Services) Lambda to accomplish our goal [2, 13].

In order to have these software components physically interconnected by a hardware system that is capable of capturing ambient data from a room, a Raspberry Pi Arduino Uno board, a DHT-11 sensor, a microcontroller ESP8266, and an LCD were used. The cloud provider was chosen to be AWS. However, the Wi-fi chip has not been strong enough to pass the AWS authentication test with higher security, all these consequently required also to use the Raspberry Pi as an intermediary node between the station and the cloud.
As a branch of Amazon Web Services, AWS IoT allows the integration of IoT devices with the ecosystem and Cloud made available by Amazon. AWS IoT works using the concept of MQTT, each device having topics for specific operations like reading, updating, accepting or rejecting queue data. All these topics are incorporated in the Shadow concept that represents the device state at the current moment.

The architecture entails two UI applications, one for the wearable which will be the main focus of this paper, and a website which aggregates the data from both the tracker and the mobile application while also making it editable. As both applications require a common backend to fetch and post data, a decision was made to use microservices developed in Python. Therefore, the business logic has been treated as a service black-boxed from the users and exposed via a public API to the related applications. The services were hosted on AWS to provide reliability and scalability, making communication viable via HTTP protocols. Accordingly, the development time was halved and, most notably, the architecture had a common backend which resulted in uniformity. Firstly, the services were developed using a combination of the Python and C# programming languages in a domain-driven design workflow. The data was persisted using a document-oriented database (MongoDB) which was encompassed with the business logic layer and the API in a containerized image through Docker so that the service may run independently of the machine it resides on. Basically, AWS allows the services to be hosted with minimal costs and virtually no downtime, providing availability worldwide and scalability on the go, based on demands, in case of the user base would ever increase.
Moreover, a second IoT system was designed for the interest of this paper, namely a weather station, powered by Arduino and based on temperature sensors, a Raspberry Pi device and the Cloud, as we will describe in the next sections.

3 FitPi system components and features

This section is to present our FitPi system at the developing level: system architecture, components, internal applications, functionalities and features, workflows and specifically involved microservices.

FitPi is a system based on SOA design principles, requiring multiple frontends to run on. As a result, a backend capable of acting as a black box has been developed to assure the communication with many clients, regardless of their origin. It uses two client applications, as follows: (a) a website frontend that runs both in the browser, behind the scenes, and on the IoT device (Arduino and Raspberry) and (b) a wearable frontend available on Fitbit devices for the Versa and Ionic lineups, these versions being among the first running over the FitbitOS. These two components and their appropriate interconnecting modules are illustrated in Figure 1 and presented in detail in the next paragraphs.

The website application follows the ASP.NET MVC5 framework and the Model-View-Controller paradigm, the controllers being responsible for receiving incoming requests. After applying the required business logic, if any, the controller produces the desired model with the data received from the backend. This model is then applied to the desired view, which is finally returned to the user. To support reusability, extensibility, and also the separation of concerns, a 3-layer architecture was correspondingly developed for the website application, as follows:

1. Presentation layer as the MVC layer, i.e. controllers, views, models were created at this level;

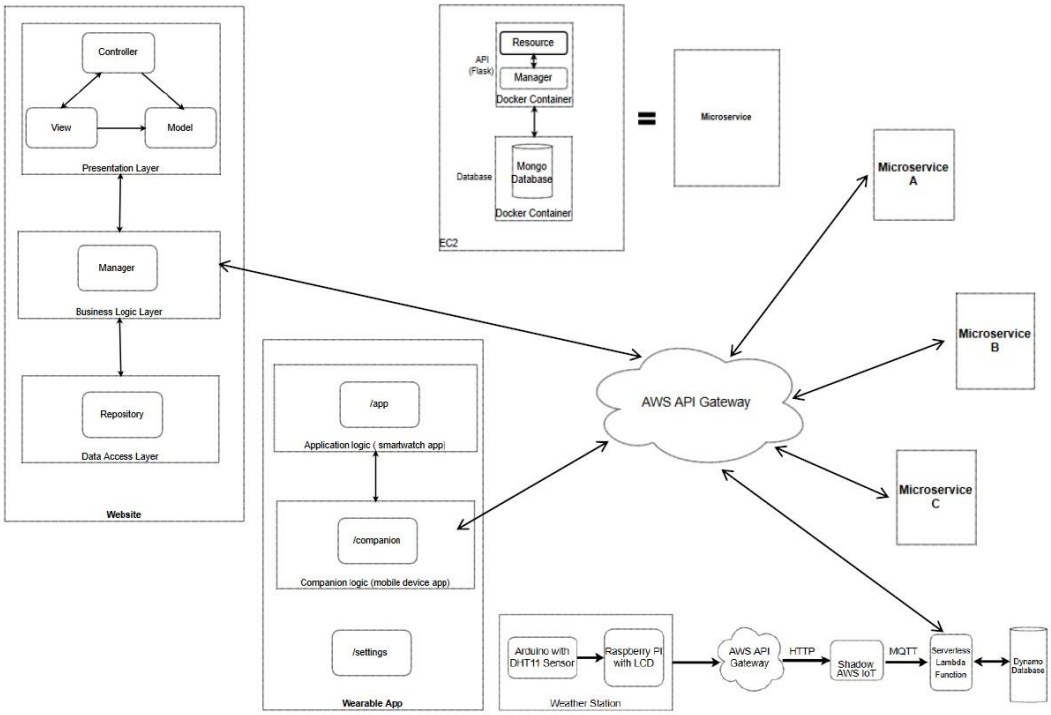

Figure 1: Application architecture

2. Business logic layer for residing the specific Services for Managers in order to keep the controllers light; for applying and housing the business logic, specific constructs are called by the controllers, acting as a black box to the Presentation MVC layer;

3. Data Access layer consisting of data manipulation processes and acting as a black box or an interface for working with the data source; this is where the repositories live and call on our data storage solution.

For the website, the Data Access layer was lightweight, in the sense that, owing to the shared backend, we had external APIs to call in the business layer. This solution exists mostly to persist information about the user's current state.
The second application, namely the wearable frontend, was created using the architecture required by FitbitOS. This application consists of a component running on the smartwatch and a mobile component running on the smartphone as a companion. Both communicate seamlessly with each other, acting like one. The wearable application structure is presented as follows:

• */resources* as a folder keeping all external resources like pictures, styles, or frameworks required by the app as well as the landing page (index.gui) which acts as a master container for the entire wearable app user interface;

• /app as a folder housing the application logic that runs on the smartwatch device; the files in here are responsible for interacting with the master view to display user-requested views as well as communicating with the companion app to fetch the data;

• /companion containing the companion logic side of the app which runs on the mobile device; a specific socket communication solution has been assumed here to ensure the interaction with the watch device and make the JavaScript fetch API to communicate with the backend, i.e. sending and fetching data through HTTP endpoints;

• /settings as a separate page of the companion app that establishes application-level configuration, for example, the application colour map.

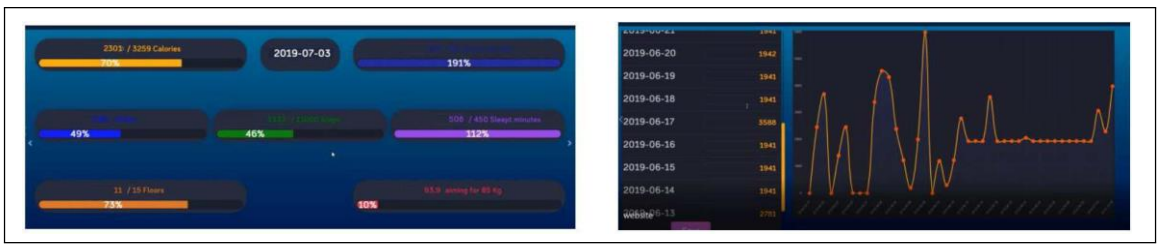

Figure 2: Daily activity module - dashboard and details on steps

Besides these two client applications, in Figure 1 we have an extra third client in the sense of a weather station that accomplishes our FitPi ecosystem. This

station is a hardware device based on two components: (1) an Arduino board that tracks the temperature data and (2) a Raspberry Pi that is connected to a small LCD where the default browser is shown in full screen with the website version of the app. The website has integrated the JavaScript speech- to-text engine functionality to allow the users to communicate with the system via their voice, similar, in this regard, to Amazon Alexa or Google Home.

The step forward to having a real software architecture based on these frontend layers was in the direction of using SOA to add a suitable shared backend. This backend application is user-oriented and responsible for doing the heavy lifting, applying complex business logic and calculations and acting as an interface for consuming the client applications. Therefore, our FitPi product performs as an original and complete solution for managing user's daily activities and allowing easy access to information related to the personal progress achieved. Current backend implemented functionalities are described as follows.

Daily activity is for manipulating data tracked by the wearable device, considering: the number of steps performed in a day, calories consumed, floors climbed, duration of intense activity, body weight, hours of sleep, distance travelled and heart rate. These are available in a daily format, with an appropriate overview, but also a detailed management tool for each element throughout the use of the application is available. We add in Figure 2 two captures from this module, one from the activity dashboard and the other with the steps target presented in details.

Nutrition is for keeping a micro- and macro-nutrient-oriented view upon the daily food in- take. Data on consumed foods such as calories and macro-nutrients like carbohydrates, fats, proteins, vitamins, sugars are managed. Again, data are presented in a daily format, with additional information on the user's monthly progress.

Budgeting is an original part for tracking the spent money, being focused on categorizing transactions made in budgets, compared to the transaction orientation of popular applications on the market, and setting thresholds for each of them. It is possible to add new budgets, in the monthly context and related transactions. At the same time, FitPi informs the user about exceeding the thresholds and about the percentage remaining in each budget.

Timetable is an educational topic for keeping track of the users' timetable, for example, classes at university. Although anyone can use the application, there is this custom section dedicated to students working parttime. Thus, FitPi presents, in an easily understood format, the courses, laboratories and seminars to which the student must attend, mentioning the current time at which they must be active.

Work-log is a work-time management tool for monitoring the hours done as usual work. As we already mentioned before, should one be working under a flexible schedule, that user may want a way to record the number of hours they spend at their day job, whilst also keeping track of having the hours required monthly.

Weather is an extra functionality of our project for persisting data about temperature for both the room and the city where the weather station device

is positioned. FitPi provides information about the temperature and humidity of the room where the physical system is located, as well as providing weather forecast data over several days.
Applying an SOA-based solution comes with powerful extensibility as each functionality or domain is isolated in its own microservice. Such architectures may enable multiagent solutions leading to emergent applications [12]. This way, in case of a business need for adding another requirement, a separate microservice can be created without affecting the existing functionality. The main microservices benefit is that each one can be created with a different tech stack, the best way to solve the services use case. Most services were created in Python programming language so that the Flask framework was used for fast bootstrapping of the exposed API. A service follows the three-layer architecture as specified in the frontend description with the attached details:

1. Presentation layer, where the domain models are present and the API side of the service is created; for this particular use case, we have resources (a resource construct consists basically of the endpoints of particular functionality, similar to a controller); for simple services, this is mostly the only layer used, owing to the simple CRUD business logic;
2. Business logic layer, where the managers are present with the role of encompassing business logic if any;
3. Data access layer, where we house the data storage solution, in this case, a NoSQL database: MongoDB; from a developer point of view, this could be considered as the other visible structural part of the microservice since there is a section for the API and another one for the mongo database.

Having these separate sections, namely API part and Database part, which now need to communicate with the frontends, we have accordingly used the built-in solutions provided by Amazon Web Services. Each microservice section is placed inside of a docker container, allowing the microservice to run on any machine that has that docker installed. We have used EC2 machines to host our microservices in the cloud, and these machines were then secured by hiding them behind Amazons API Gateway, which is responsible for load balancing and validating the requests. Only the frontend applications can access the backend endpoints using token validation.
We conclude this system description section by adding both a principal workflow and one
for the weather station part. The principal workflow example could be:

1. the user requests a specific page
2. a frontend controller or a function from the wearable app section receives this request
3. the controller calls the manager from the business logic to get the data
4. the controller either calls the data access layer or makes a call to a microservice for the data

5. the API gateway intercepts this request and validates it; in case of approval, it is then forwarded to one EC2 machine which is specific to the microservice called; load balancing part decides to which machine the request is forwarded (as there are several machines for each microservice), also considering the performance reasons
6. the EC2 machine forwards the request to the docker container
7. the docker than forwards the request to the resource of the microservice (API section)
8. the resource calls its manager if needed or uses the mongoDriver to communicate with its database or fetches the data from an external data source
9. once the service has finished manipulating the date, a roundtrip is done all the way back to the frontend application
10. the controller receives the data, puts it in a model and renders the required view for the user.

The weather station part has a similar but slightly different workflow as it uses AWS IoT, a SaaS functionality created by Amazon. For example, such a workflow could consist of next steps:

1. the Raspberry Pi sends the data collected by the Arduino board to AWS via the AWS Gateway
2. this is then sent to AWS IoT and stored in the device's shadow (this is a concept in AWS
1. which basically means the state of the device)
2. AWS IoT uses an MQTT protocol as the shadow acts as a message bus; we have a serverless function (in AWS this is called a Lambda function) that subscribes to this message bus; on each shadow change given by a new temperature data, the function stores this in a NoSQL database, DynamoDB
3. the function also exposes an endpoint for retrieving this data when a frontend application demands it.

4 User perspective on FitPi as an IoT system

Apart from being a modern and up-to-date IoT system, our FitPi project is significant and valuable support for any user interested in monitoring their everyday life and publishing the daily activities results on a social network, here Twitter. This feature was achieved by implementing a TwitterBot [15], and Python programming language has been used because it provides modules ready to use the Twitter API quickly, acting as a black box over low-level implementation. Since it has been observed that text messages sent to the user are not effective in keeping them motivated beyond a week, perhaps having the information published on a social media platform would motivate the user to keep trying to improve their lives and continue using the FitPi

product to pursue their goals. For example, in Figure 3 we have a page of statistics prepared for keeping the user's motivation at a high level.

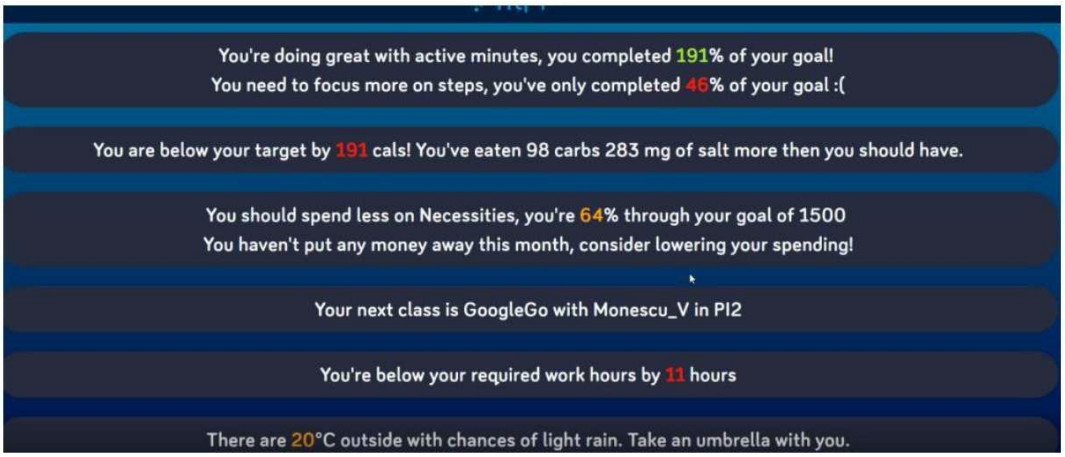

Figure 3: Statistics - user view

Figure 4: Nutrition module - website and wearable views

As a Fitbit-based project, our system has access to all the physical activity data which are stored in the Fitbit data service. The user simply must carry on with their daily life, and the tracker will monitor all that is related to steps taken, calories burnt, floors climbed, and some other. Data that are related to weather is either retrieved from OpenWeatherMap when the user wants to know more about temperature from other areas or by using the provided weather station which records information about temperature and humidity every one minute. For monitoring food consumption, we used the database provided by MyFitnessPal (MFP). Our users can either log their meals into this MFP as it is synchronized with FitPi or directly into the FitPi website. The timetable has data directly fetched and calculated from the Transilvania University of Brasov timetable. Other sections have their own in-house database so that when a user logs something into the website, data is stored and can be displayed both in the website and the wearable application. This mirroring solution for the specific case of the Nutrition data is represented in Figure 4.
Each of our system modules is implemented with respect to the IoT approach, and it succeeded by featuring the specific aspects of security, privacy, scalability and other given self- properties in different manners, as we highlight in the following.

Our system faces the security problem from many perspectives, for example. The services were hosted on AWS to provide a higher level of reliability and scalability (making communication viable via HTTP protocols), and the configuration files contain both private keys and other data requiring higher security. Moreover, we accessed the AWS console for EC2 services and chose a Linux AMI (Amazon Machine Image), being optimally a hosting environment for services. In the management console, we created a thing to which we assigned a certificate, a public key and a private key (used in the Raspberry python script), so data transmission can be done securely and encrypted via HTTPS.

As we have already mentioned before, the service that deals with physical activity has numerous managed data elements such as the number of steps taken in a day, calories consumed, floors climbed, duration of intense activity, body weight, hours of sleep, distance travelled and heart rate. The uniqueness of these data is given by their retrieval from a compatible physical activity monitoring device, for example, Fitbit Versa. The fitness tracker could be synchronized with any mobile application that sends data to the Fitbit Cloud. Using the access provided by the Web API, as a system developed by Fitbit to access data from tracking devices, we were able to fit it out in our architecture and synchronously obtain the data needed for the FitPi domain of interest. Since these data need to be retrieved at regular intervals to synchronizing the information between FitPi and fitness tracker, we used Flask and APScheduler (Advanced Python Scheduler), a Python tool that can configure jobs to run at regular time intervals on a separate thread, thus maintaining the functionality of the services in Flask. The APScheduler tool is also used to synchronize data from FitPi with data from MFP for the current day. Data is stored in an aggregate document, with information about all the tables in four options defined by the MFP (breakfast, lunch, dinner and sweets).

The nutrition service manages data on consumed foods such as calories and macro-nutrients, like carbohydrates, fats, proteins, vitamins, sugars. This data comes from the MyFitnessPal database, the food products being registered in the application provided by MFP arriving in the cloud. Because the MFP access via API is private and it is not possible to access it by an individual developer, we used a web scrapper that directly parses the desired data (if it is in the public domain) from the MyFitnessPal website based on the user id and successfully passed the privacy problem.

5 Discussion and future work

The more these new branches of technology continue to emerge, the more humans become involved in personal wellbeing, and the abundance of data their smart devices provide them allow a deeper insight into their lifestyle. Even

if are there countless health applications available on a multitude of wearable platforms, this FitPi project aims to complement these apps in the direction of daily activity monitoring based on smart solutions.

We may conclude now that the lifestyle management system, FitPi, targets to cover most of the daily activities adopting an intelligent approach. The FitPi website frontend assembles all the information related to the physical activity using real data and presents it in an user-friendly and easy-to-understand format. By means of the voice assistant, it becomes a reactive system that notifies the user about the progress made in reaching the personal set goals, as well as about a possible decline in the effort made, thus determining the user to better their approach. This improvement is also supported by the other functionalities of the application, such as monitoring the food intake and finance management. At the same time, FitPi facilitates certain elements of the user's life and, by suggesting possible actions to be taken accordingly to the recorded vital signs values, it performs like a self-adapting IoT system. Also, by developing an application available on a wearable device, the user could access the FitPi data transparently, which increases their motivation to maintain a healthy lifestyle.

For now, the current FitPi version is a single-user version, the developer. The biggest step that FitPi has to do in the near future is to create and integrate a service that allows multiple users to register, for example, using ASP.NET Identity support. This also involves versioning existing services, implementing new specific security features and personal data protection by saving unique data to each user account.

FitPi allows users to access internet services through a browser or Fitbit devices. Still, to grow their demographic and fully satisfy existing users, there is a need for native applications on mobile devices. Thanks to the adopted service orientation architecture, this can be achieved much faster - already having the backend functionality, it requires only the creation of native client applications to use these services.
Last but not least, integrating the voice system in all areas of the application is a valuable step forward in the direction of having an intelligent notification system. Now, FitPi offers limited support for data aggregation, and this allows the user to be aware of his or her progress or regress. But, by implementing an independent active notification system based on o set of recommended actions that the user should take, we add artificial intelligence to the system and greatly improve the user's success rate.
Once these extensions are implemented, we can say that FitPi is not only an alternative to current solutions, bringing various improvements, but a complete IoT system, able to positively influence its users every day, inspiring them in a smart way to reach their lifestyle goals.

References

[1] Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., Savaglio, C. Enabling IoT interoperability through opportunistic smartphone-based mobile gateway, Journal of Network and Computer Applications, March, 74-84, ISSN: 1084-8045, 2017
[2] Amazon Web Services homepage, https://aws.amazon.com/. Last accessed 9 November 2020
[3] Amresh, A., Lyles, A., Small, L., Aloysius Gary K., FitBit Garden: A Mobile Game Designed to Increase Physical Activity in Children, Proceedings of the 2017 International Conference on Digital Health (DH'17), 200-201, Elsevier, London, United Kingdom, ISBN: 978-1-4503-5249-9, 2017
[4] George Eleftherakis, Dimitrios Pappas, Thomas Lagkas, Konstantinos Rousis, Ognen Paunovski, Architecting the IoT Paradigm: A Middleware for Autonomous Distributed Sensor Networks, International Journal of Distributed Sensor Networks (IJDSN), 139735:1-139735:17, ISSN: 1550-1477, 2015
[5] Fitbit Inc. homepage, https://dev.fitbit.com/build/reference/. Last accessed 20 September 2020
[6] Arindam Giri, Subrata Dutta, Sarmistha Neogy, Keshav Dahal, Zeeshan Pervez, Internet of things (IoT): a survey on architecture, enabling technologies, applications and challenges, In Proceedings of the 1st International Conference on Internet of Things and Machine Learning (IML '17), Association for Computing Machinery, New York, NY, USA, Article 7, 112, ISBN: 978-1-4503-5243-7, 2017
[7] Gaoyang Guan, Borui Li, Yi Gao, Yuxuan Zhang, Jiajun Bu, Wei Dong, TinyLink 2.0: integrating device, cloud, and client development for IoT applications, In Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), Association for Computing Machinery, New York, NY, USA, Article 13, 113, ISBN: 978-1-4503-7085-1, 2020
[8] Kasun Indrasiri, Prabath Siriwardena, Microservices for the Enterprise: Designing, Developing, and Deploying, APress Media LLC, Springer Science + Business Media New York, ISBN: 978-1-4842-3857-8, 2018
[9] Rohit Ashok Khot, Larissa Hjorth, Florian Mueller, Shelfie: A Framework for Designing Material Representations of Physical Activity Data, ACM Transactions on Computer-Human Interaction, Vol. 27, No. 3, Article 14, Publication date: May 2020, ISSN: 1073-0516, 2020
[10] Kishore Kodali, R., Mandal, S., IoT Based Weather Station, International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 680-683, IEEE Kumaracoil, ISBN: 978-1-5090-5241-7, 2016
[11] Michael Papazoglou, Web Services and SOA: Principles and Technology, Pearson Education Canada, ISBN: 978-0-2737-3216-7, 2012
[12] . Paunovski, O, G. Eleftherakis, and A.J. Cowling, Disciplined Exploration of Emergence using Multi-Agent Simulation Framework, Computing and Informatics, 28(3):369-391, ISSN:1335-9150, 2009
[13] Sarkar, A., Shah, A., Learning AWS - Second Edition, PACKT Publishing House, ISBN:978-1-7872-8106-6, 2018
[14] SHIELDX homepage, https://www.shieldx.com/. Last accessed 20 September 2020
[15] Twitter development support homepage, https://developer.twitter.com/en. last accessed 11th November 2020
[16] Yicheng Bai, Chengliu Li, Yaofeng Yue, Wenyan Jia, Jie Li, Zhi-Hong Mao, Mingui Sun, Designing a wearable computer for lifestyle evaluation, 38t h Annual Northeast Bioengineering Conference (NEBEC), 93-94, IEEE, Philadelphia, PA, ISBN: 978-1-4673-1140-3, 2012