Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa

Internet das Coisas
MEI/MI/MSI/MCD
2024/25

Project

A fitness monitoring IoT application

1. Introduction

The Internet of Things (IoT) is being incorporated into all enterprise and personal monitoring systems. Thus, we need efficient and dependable algorithms and approaches to analyze the IoT sensor data to find valuable patterns and insights.
The frequency of the upcoming data may be high-speed. Sensors may read hundreds to millions of data per second. There are many application areas in the real world, like leisure, health, weather monitoring and forecasting, agriculture, enterprise maintenance, data centers, and others. Therefore, wrangling, analyzing, and grasping insights from data is essential for multiple application sectors.
In this project, we want to create an IoT application for monitoring and analyzing personal performance in leisure activities. An IoT device (an iPhone) was used on the wrist to collect accelerometer and gyroscope data (measurements from the three axes are collected). Additionally, the dataset contains "date" and "time" columns, which provide information about the exact date and time associated with measurements.
In this work, we want to understand how the user's performance changes daily, how much they run and walk (distances), the duration of activity and the number of calories burned during walking and running activities per day, week and month. Additionally, we want to determine statistical information per week and month. All this information must be displayed in an intuitive dashboard, which allows users to view their performance/data easily.

The project can be split in two stages. In the first an offline data set is used (tranning.data), whereas in the second stage an on-line data set (online.data) will be employed.

In the first stage the groups must:

1. Implement the IoT architecture described in the project development section.
2. Understand the offline data set and create a dashboard to display statistical results about how the user's performance changes daily, how much they run and walk (distances), the duration of activity and the number of calories burned during walking and running activities per day, week and month.
3. Based on the average values (per day, week and month), is the user an active or a sedentary user? For example, suppose you were analysing walking and running data. In that case, you might set a threshold of 10 minutes of running per day or 30 minutes of walking as the cut-off between an "active" and a "sedentary" user.

If the user's average daily activity is above the threshold, they may be categorized as "active," while those with averages below that threshold might be considered "sedentary."
In analysing average values per week or month to determine a user's activity level, the statement provides guidance on adjusting the threshold value for comparison. For example, if the threshold for an "active" user is 10 minutes of running or 30 minutes of walking per day, and you want to determine if the user is "active" on average over a week, you would calculate their weekly average activity and compare it to 10 minutes x 7 = 70 minutes of running or 30 minutes x 7 = 3 hours and 30 minutes of walking. If the user's weekly average exceeds 70 minutes of running or 3 hours and 30 minutes of walking, they would be considered "active" for that week.
Note that these thresholds should be configured through the Dashboard.

In the second stage the groups must:

1. Use the offline data set to train machine learning models able to classify the type of activity that is being done.
2. Understand the online data set and create a dashboard to continuously display the evolution of the activity (how much the user ran and walked (distances), the duration of the activity, and the number of calories burned during walking and running parts). This dashboard must also classify if the user is walking or running at each new record.

2. Project development

In order to answer the questions raised before, students must create an IoT solution according to the architecture shown in Figure 1.

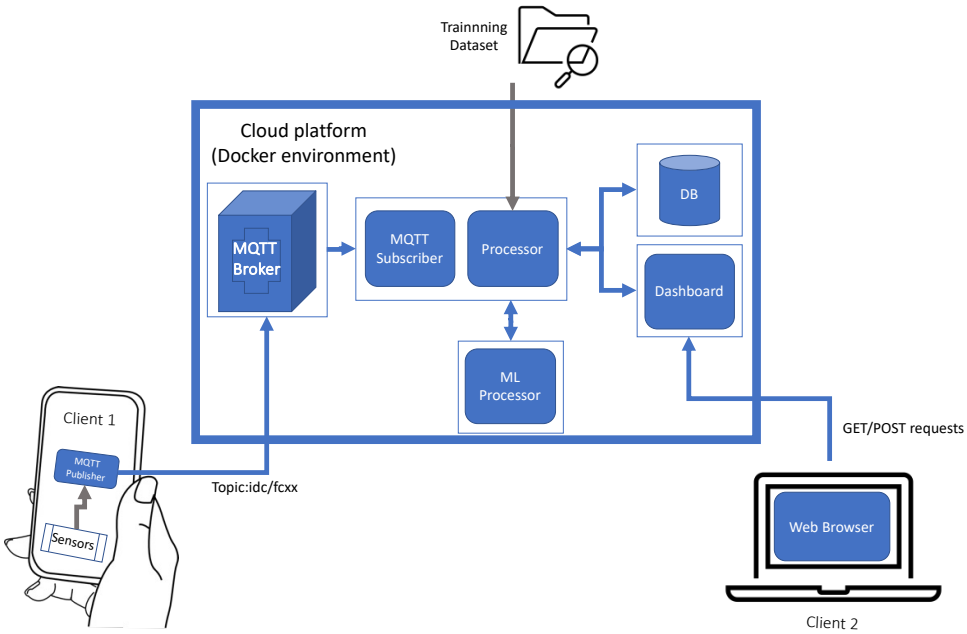

Figure 1: System architecture

This architecture is divided into three main parts: a cloud platform, client 1 and client 2.

The cloud platform will be established by utilizing the Docker virtualization platform. Docker is a versatile containerization tool that empowers developers by providing a comprehensive environment to create, distribute, and execute containerized applications, aligning seamlessly with the micro-services architectural paradigm. This approach encourages decomposing complex applications into smaller, independently deployable components, fostering modularity, scalability, and maintainability.
Docker containers offer a lightweight and consistent runtime environment, encapsulating the application code and all required dependencies, libraries, and configurations. This isolation ensures consistent behaviour across different environments, from development to production, streamlining the development lifecycle and facilitating collaborative development efforts. Furthermore, Docker plays a crucial role in modern cloud-native development, enabling agility, flexibility, and reliability.

In this project, we will develop and deploy five distinct services within our cloud environment. ***The first*** of these services involves the implementation of an MQTT Broker, a key component for enabling communication using the MQTT protocol. For this service, students should install, configure, and execute the Mosquitto broker [1]. It should be configured to listen at port 1883 (default MQTT port) and be accessible remotely (mainly from the host machine).
Alternatively, several docker containers are ready at Docker Hub (e.g., https://hub.docker.com/_/eclipse-mosquitto). It is allowed to use these containers, but students must be able to configure them according to the project needs.

The second service comprises an MQTT subscriber responsible for receiving data from the MQTT Broker, and a Processor service will be deployed, tasked with handling and processing the received data. This Processor plays an essential role in the system by executing several functions:

1. **Data Loading**: Firstly, it will load the training dataset, ensuring the system can access the necessary data for machine learning processes.
2. **Database Interaction**: The Processor will interact with the database service, transmitting the incoming data for storage. This interaction ensures data persistence and availability for future analysis and retrieval.
3. **Machine Learning Processing**: The Processor will call the ML Processor service to initiate the training of machine learning models. This step is essential for enhancing the system's predictive capabilities and insights.
4. **Dashboard Updates**: Lastly, the Processor will update the information on the Dashboard service, ensuring that the latest data and insights are shown to users in a user-friendly interface.

The third service in our project is the Database (DB) service, a crucial component responsible for storing and retrieving data processed by the Processor. Students undertaking this part of the project will be tasked with the following key responsibilities:

1. **Database Engine Selection**: Students will need to carefully select a suitable database engine based on the project's requirements.
2. **Installation**: Once the database engine is selected, students will install it within the docker environment. This involves deploying the chosen database software and ensuring it is correctly set up and accessible.
3. **Storage and Retrieval Setup**: This involves designing and implementing the necessary data tables, defining data structures, and establishing efficient data access mechanisms (e.g., primary and foreign keys).

4. **Integration with Processor Service**: Integration between the Database and Processor services must be established, ensuring that data can be efficiently transmitted from the Processor for storage and later retrieval.

Concerning the database service, there are also several containers with databases ready at Docker Hub (e.g., https://hub.docker.com/_/postgres, https://hub.docker.com/_/mysql, https://hub.docker.com/_/mariadb). Like the MQTT Broker, students can choose one of these containers or create their container with a database service.

The fourth service corresponds to the ML Processor, a service dedicated to the training, testing, and validation of Machine Learning (ML) models. This service enables the development of predictive models based on incoming online data to discern whether a user is walking or running.

Finally, the Dashboard service represents the user interface component of our project and must be meticulously designed, created, and deployed. We offer students the flexibility to choose the technology stack that aligns with their expertise and project goals. The suggested technology for implementing the dashboard functionality is the Node-RED server [2] [4], known for its versatility and ease of use. However, we also welcome alternative technology choices based on students' preferences and insights. If a different choice is made, students must provide a clear and comprehensive explanation in their project report. This explanation should justify the technology choice, highlighting its advantages and relevance to the project's objectives. Additionally, students should outline how the selected technology facilitates the creation of an intuitive and informative dashboard.
Concerning Node-RED, there are also several containers at Docker Hub with the Node-RED ready to be used (e.g., https://hub.docker.com/r/nodered/node-red).

In terms data, we made available, at Moodle, an initial dataset (training dataset - tranning.data) that must be used to supply the system. The dataset is organized as follows:

date	time	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
01/07/20	13:51:15:847724020	0	0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.9296
01/07/20	13:51:16:246945023	0	0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.1269
01/07/20	13:51:16:446233987	0	0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.4367
01/07/20	13:51:16:646117985	1	0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.9336
01/07/20	13:51:16:846738994	1	0.4814	-0.9312	0.0359	0.0527	0.4379	2.4922

Where the walking activity is represented by “0” and the running activity is represented by “1”.
As mentioned before, the application must be able to determine how much the user runs and walks (distances), and the duration of each activity per day, week, and month. Statistical information per week and month should also be computed. In order to make the application more interesting, the quantity of calories should also be estimated. In a simple way, it can be done as:

$$\mathrm{C} = 0{,}035 * body_{weight} + \left(\frac{v^2}{h}\right) * 0{,}029 * body_{weight}$$

Where:

- C represents the number of Calories burned per minute;

- $body_{weight}$= represents the body weight in kg;
- v is the velocity in m/s;
- h is the height of the person in m;

The average velocity of a person varies according to the age and gender. It is represented in Table 1:

Age	Gender	Meters/second
20 to 29	Male	1.36
	Female	1.34
30 to 39	Male	1.43
	Female	1.34
40 to 49	Male	1.43
	Female	1.39
50 to 59	Male	1.43
	Female	1.31
60 to 69	Male	1.34
	Female	1.24
70 to 79	Male	1.26
	Female	1.13
80 to 89	Male	0.97
	Female	0.94

Table 1: Average velocity per age and gender

For the running velocity, students can consider the average velocity times 2.
All information must be shown in an intuitive dashboard that will be accessed via client 2, which allows users to easily visualize their performance/data. Figure 2 shows an example on how this data can be presented to the user.

Figure 2: User interface example

Once all mechanisms and the dashboard are developed, client 1 must be launched to send new values to the system. Client 1 simulates a user iPhone that collects sensor data and sends it in real-time. It will be simulated by a local process that will read raw data from

the online.data file and sends an entry (a line) every 1 seconds to the MQTT Broker that will publish it to the subscribers.
The online.data file includes the accelerometer and gyroscope data without any time reference. It is organized as follows:

acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.9296
0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.1269
0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.4367
0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.9336
0.4814	-0.9312	0.0359	0.0527	0.4379	2.4922

The information about time must be added by Client 1 before sending data to the MQTT Broker. A topic with your identification must be used, e.g., idc/FCxx, where xx represents your student number.

These new values must be stored in a database and all computations must be updated with the new information from data that is arriving the system.
The project should be deployed in a Docker environment, simulating a real Cloud platform.

Additionally, all communications must be secure to ensure the confidentiality, integrity, and authenticity of the data being transmitted. Without secure communication, sensitive information such as personal information could be intercepted, modified, or tampered with by malicious entities.
Transport Layer Security (TLS) will be employed to secure our communications. TLS is a widely adopted to secure communication over computer networks. It encrypts the data transmitted between two parties (such as a client and server) to prevent eavesdropping, tampering, or forgery by third parties.

3. Report

Students are requested to prepare a report named IdC-2425-Project-Groupxx.pdf (where xx is your student number), in pdf format, to be submitted in the course Moodle page.
The report must:

- be elaborated according to the presentation rules of scientific articles, using the template that will be made available in Moodle;
- respect the minimum and maximum number of pages, 4 and 6, respectively;
- provide a description of the solutions implemented, describing the setup, communication and logic details adopted in the implementation. Additionally, describe the main challenges and difficulties encountered.

Furthermore, a compressed file named IdC-2324-Project-Groupxx.zip, containing the source code of the project as well as other documents that might have been used as references to carry out the project, must also be submitted through Moodle.

4. Discussion

In addition to the report, students will be required to demonstrate the solutions developed. As already suggested, the dashboards must be intuitive to allow for a fast and accurate assessment of the work developed.

5. Delivery date

December 4th, 2024, until 23h59.
Instructions will be given on Moodle.

6. References

[1] https://mosquitto.org/
[2] https://nodered.org/
[3] J. Cecílio, “Node-RED Tutorial”, FCUL, 2023, available at Moodle.
[4] J. Cecílio, “Docker Command Sheet”, FCUL, 2023, available at Moodle.
[5] Stack Overflow contributors, “Learning Docker”. Free eBook, 2017.
[6] Mouat, Adrian, “Using Docker: Developing and deploying software with containers”, O'Reilly Media, Inc., 2015.