MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE

Smoking Detection Using Smartwatch Sensors

LICENSE THESIS

Graduate: **Rares Tudor RUS**

Supervisor: **Șl.Dr.Ing. Marcel ANTAL**

2022

MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

DEAN,
Prof. dr. ing. Liviu MICLEA

HEAD OF DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Graduate: **Rares Tudor RUS**

Smoking Detection Using Smartwatch Sensors

1. **Project proposal:** The proposal of the project was to create an application that uses neural networks and smartwatch sensors to recognize human activity with a particular focus on distinguishing smoking from other forms of activity.
2. **Project contents:** Introduction, Project Objectives, Bibliographic Research, Analysis and Theoretical Foundation, Detailed Design and Implementation, Testing and Validation, User's manual, Conclusions
3. **Place of documentation:** Technical University of Cluj-Napoca, Computer Science Department
4. **Consultants:**
5. **Data of issue of the proposal:** November 1, 2021
6. **Data od delivery :** July 8, 2022

Graduate: _____

Supervisor: _____

MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

Declarație pe proprie răspundere privind autenticitatea lucrării de licență

Subsemnatul(a) **Rus Rares Tudor** legitimat(ă) cu act de identitate seria CJ nr. 267261 CNP 1990402125810, autorul lucrării **Smoking Detection Using Smartwatch Sensors** elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare (engleză) din cadrul Universității Tehnice din Cluj-Napoca, sesiunea iulie a anului universitar 2021-2022, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență.*

Data	Nume, Prenume
08.07.2021	Rus Rares Tudor
	Semnătura

Contents

Chapter 1. Introduction - Project Context

1.1. Project context

The primary goal of this project is to raise public awareness about the dangers of smoking to both smokers and others around them. The main objective is to create an application that accomplishes precisely what is indicated above. This aim was established to combat the rapidly increasing number of people dying as a result of the health problems that smoking causes all over the world.

As mentioned in [jha2009] smoking alone cause approximately five to six million people every year. Furthermore, tobacco smoking is one of the most dangerous killers in the entire world, so everyone aware of this problem should get involved to stop the premature deaths of the people around us, providing any kind of support. Accordingly, I decided to design a project that provides the above-mentioned support. My desire, is to create an application that will help the smokers, by informing them how fast and obviously smoking affects the health. In today's world, there are a lot of researches that witness how much does smoking affect health in the long run. At the same time, out there are a lot of studies that show how much damages are provoked immediately after the act began. Nevertheless, during my research, I did not find any application that has this kind of use, to inform the users what effects smoking have immediately after the cigarette is done.

Another big problem, is the fact that more and more teenagers star smoking in the early stages of their life due the temptation of E-cigarettes. Since, this E-cigarettes are allowed to be smoked inside and are almost always overlooked by parents, under the pretext" is only steam", no is not, because they are full of chemicals and they usually have nicotine just like normal cigarettes and the biggest problem of all is that it causes addiction. Usually, people who have smoked electronic cigarettes in their youth or in the past will almost certainly start smoking normal cigarettes.

In conclusion, my thesis project is an application composed of three parts, the user interface that is the one collecting movement data from the user and also information about user's body weight and height, the server side which records the data from the user's device and the part that collects the data and classified it based on a trained model, to express the movement the user was doing during recording time.

1.2. Problem statement

Nowadays, the top three causes of death in the world are: firstly, cardiovascular diseases, cancers and on the third place we have, respiratory diseases. All of these three reasons of death are stimulated by smoking. A considerable study [NBK179276], certify that smokers are more likely to develop heart and blood vessel problems (cardiovascular disease), problems like stokes or coronary heart disease, which are very dangerous and

have a high death count. Also, in [NBK179276] it is mentioned about how badly the lungs of a smoker can suffers, so bad that it can cause very dangerous diseases. Besides that, cigarette smoking can cause cancer practically in anywhere in the human body, it also makes it harder for the body to fight cancer. Smoking also damage the reproduction system, in the case of men, the sperm's fertility is reduced and for woman it makes it harder to become pregnant, even though the pregnancy happens, in the most cases, the baby's health is affected.

Another import issue is that the numbers of teenagers who smokes is rapidly increasing due the ease with which they can get E-cigarettes, which are the tobacco product that is most usually used. E-cigarettes are electronic cigarettes that convert a liquid into a vapor for inhalation. Nicotine, flavoring, and other chemicals are commonly included in this liquid. Nicotine, which is included in both e-cigarettes and traditional cigarettes, is highly addictive. Because most E-cigarettes include nicotine, which derives from tobacco. The most addictive thing about this E-cigarettes is the fact that are allowed in almost all the enclosed spaces, like malls, cinemas, etc. Consequently, almost all the teenagers who start smoking e-cigarettes will continue smoking them or will change them with normal ones and this is a big problem.

All of the information presented above suggests that smoking should be avoided by anybody, but still there are hundreds of thousands of people around the world who start smoking every day, so somebody should take some measures to combat this. In almost all the countries in Europe, the cigarettes pack have negative advertisements printed on them, to raise the awareness about the effects that smoking will have in the future. However, any teenager will think that they will only buy one pack of cigarettes and then will quit, but almost all the time, he or she will never quit smoking. All the advertising for the negative effects of smoking is made for future effects, although cigarettes are designed to have long-term consequences, not all smokers consider the impacts that occur shortly after smoking and these are presented almost nowhere.

Consequently, to reduce the number of smokers would be to make smokers aware of the detrimental health consequences that occur soon after smoking would be a useful endeavor to minimize the number of smokers.

1.3. Proposed solution

The product presented in this paper is a mobile application that after completion, can be published in the Android market store so it can help any smoker who need help to track the effects of smoking, in the usual basic human activities. This application was created with the intention of assisting the user to have a better understanding of the negative impacts of smoking, by taking a record of the body's vital indicators after a smoking session and comparing them to conventional levels.

The most successful strategy is the one known to everyone, visiting a medic in a hospital that specializes in lung and heart research. A medic who focuses on lung diseases is called a pulmonologist, a pulmonologist is a medic who specializes in the diagnosis and treatment of respiratory illnesses, and a cardiologist an expert on heart problems. However, your heart and lungs are inextricably connected, and they perform better together when both are in good shape. A frequent visit to each of them will guarantee that the health condition is kept up to date in a professional and timely manner. However, there is a problem, with our health-care system, you'll have to wait a long time to see a specialist who can keep you under control. If your financial circumstances

prevent you from seeing a private doctor, the horror begins, as you may have to wait up to a month. At the moment most of the world does not have time to go to the doctor, especially if there is no discomfort in the body. Therefore, the application presented offers a solution, not as accurate as a doctor, but it is not so expansive or so time consuming, so it is suitable for everyone's needs.

On the other hand, there are a lot of cheap or free solutions to keep an eye on the consequences of smoking but this involves a lot of effort from people. One of these alternatives would be to look for relevant information online provided by doctors or people specialized in the field. In order to compare the information found on the internet with personal data, a smartwatch or a smart band is needed. Regarding this approach, there are some problems, because the condition that must be met are quite costly both in terms of time and resources, due to the long time needed to read an article and find the needed information and money need to be spent on a smart wrist wear in case is not owned. This approach is fantastic, but it involves a lot of work on the part of individuals who are interested, and human errors, such as missing crucial material from the studies read or engaging in inappropriate behavior while the smart device is recording, are common.

1.4. Target Audience

The primary goal of this application is to raise awareness of the short-term consequences of smoking, by providing a friendly interface on a smartwatch that will monitor heartbeats during simple human activities.

The user experience will be highly pleasant as a consequence of the application interface's simplicity and beauty, while the needed results will be delivered in the most exact manner possible. Any smoking person that needs help to quit or that only wants to be more informed about what are the harmful effects of smoking will be pleasantly surprised about what this project has to offer. The simplicity with which the program is utilized, as well as the minimal prerequisites for obtaining an accurate result, make it appealing to smokers of all ages. In the following we added Figure 1.1 below, for a graphical presentation of alarming data of how many smokers are there.

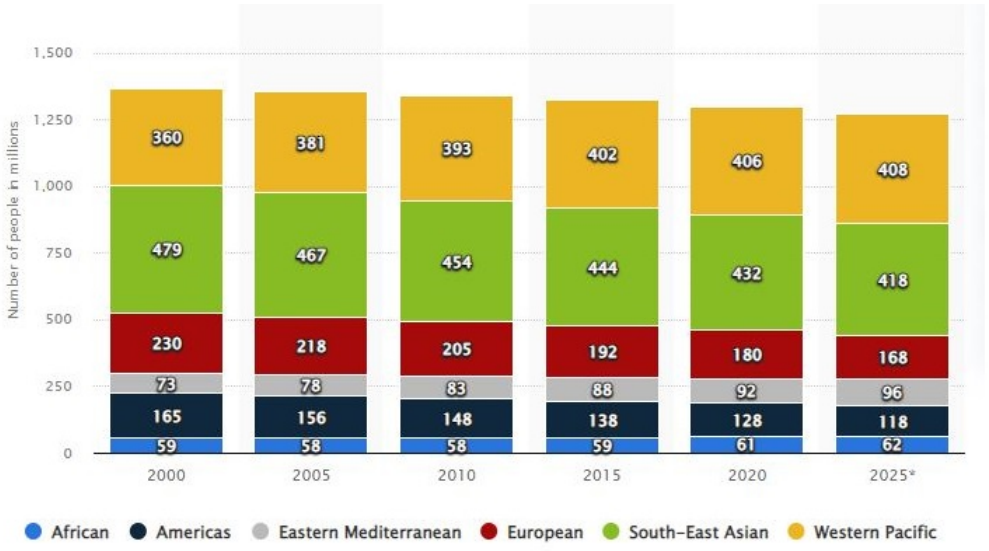

Figure 1.1: Number (in millions) of tobacco smokers

Chapter 2. Project Objectives

2.1. Main Objective

The major goal of this document is to support those who are in need of help or those who do not realize that they need help. This goal is tried to be achieved by developing an application that study the human activity through a smartwatch and based on the feedback returned by the designed system, the user can easily see the immediate effects of smoking on the body and on health.

In order to obtain the results mentioned above, the application should pass some serious tests. Firstly, the User Interface(UI) should work smoothly, has no interruptions and looks great and also to be easy to use, because users of all ages should easily understand how to work with our application. Secondly, all components should constantly communicate with each other and ensure the proper functioning of the entire system. A result of these requirements should be feedback that is very accurate. Therefore, the purpose of my project is to define and construct a system capable of providing clear data to a user based on the data provided by the smartwatch that communicates with the server.

2.2. Secondary Objective

The secondary objectives are listed below in the chronological order in which they should be accomplished.

- Understanding already existing solutions – If there are similar solutions, they will be summarized and presented in Chapter 3.
- Acclimatization with Android Studio – Looking for the functional elements needed to complete the client side of the project.
- Study the human activity recognition (HAR) – Finding several studies that deal with detecting human movements and summarizing them in Chapter 3.
- Implementation of gesture pattern recognition – Using a neural network base algorithm.
- Study about RabbitMQ – An online framework for publish and consuming messages.
- Study about smoking effects – Read works by doctors about the short-term effects of smoking on the body.
- Provide feedback to user – A graphical view of the pulse variation and also a labeled set of movements.
- Test different data – Get a bunch of people to perform some basic physical activity to test the correctness of the classifier model.

2.3. Requirements

2.3.1. Functional Requirements

Recording sensor data

A user who utilizes the project needs the smartwatch that has an accelerometer sensor and a gyroscope sensor that are working in the normal standard and also the hart beat recording sensor. When the user wants to perform an activity, he or she can press a button on the screen of the smartwatch and start the saving and sending over the internet the data recorded by the sensor during movement. When the action is over the user will have to press on the stop button and all the data recorded since he/she pressed the start button until now, is published.

All of this sensor mentioned above are, except the hart beat measuring one, are inertial sensors. The acceleration and angular velocity of an item, in this case the body of the user, are measured by an inertial sensor along three mutually perpendicular axes, the precision of the measurement depends on the quality of the sensors which are connected to the user's smart device.

Publishing data

After the user press the start button mentioned above, a connection between the client application and a RabbitMQ server is established and secured. This connection is made using a Uniform Resource Identifier (URI), and in this connection a channel for communication between the publisher and the consumer is established. Every time there is a change in the values of each sensor, a message is published in the channel, from the client-side application and in the server side that message is consumed from the same channel in the same connection. When the user decides that he wants to stop the communication and interrupt any kind of communication between his side and server side, he or she have to press the stop button, which close the connection and set the channel to null, so even if the sensor values are changing the new values have no place where to be published.

Consuming data

When the messages, the sensor data, ware published by the producer, the client side of the application, will stop the communication by closing the channel, the server which is the subscriber of the communication will consume the message. The server is always listening, if there is an internet connection, and every 10000 milliseconds, the server is interrogating the connection with the same URI as the one mentioned above, and if in the connection queue is a message it will be consumed. The values from the sensors that ware sent in the channel are received by the server as a string format, so in order to use the data from the message it has to be parsed and save into a data base.

Working servers

The perfect environment is when booth servers that will be described in the following chapters are up and running, so a user can retrieve feedback at any time he or she needs. This is because the raw data from the user's smartwatch needs to be firstly stored in a data base and in order to do that the Java server which provides tidy and clean data

for the classifier. Secondly, the Python server which is the part that returns the feedback to the user, needs to be always online. This part of the project is the main one, which classify the data from the other server into human activities, latter on this part will be also called as Human Activity Recognition (HAR) system.

CloudAMQP connection

In order to have a stable communication between the client side and the server side, a message broker is needed, in the implementation phase RabbitMQ was used. RabbitMQ is one of the most popular open-source message brokers, and it is perfect for the project needs. Therefore, in order to use RabbitMQ we need to find a cloud that provides managed RabbitMQ servers and for this I chose CloudAMQP. In order to use this cloud, it only requires an account and a plan for hosting the servers, of course that the free version has some limitations, but for the needs of this project the free plan is perfect.

Gesture detection

In this project are selected five different types of activities, the most basic ones, and they are: walking, jogging, sitting, standing and going on stairs. These activities are classified based on the different values the accelerometer and gyroscope sensors are recording during the phases mentioned above. Based on these values in the Python server side the classifier put a label on the set of data and return it to the user.

This method is working as follows, the users wear a smartwatch which is recording the movement performed. A complete activity is a time series of the sensor measurements, so an activity has many measurements and each measurement has values for all the three axes.

2.3.2. Non-Functional Requirement

Performance

This non-functional requirement is referring to the power of the application to run good enough on almost any android device, this requirement differ from the one bellow, scalability, because it's about how fast and accurate the actions on any Android smart devices should be. The performance criteria specify how well the program executes various operations under specified situations. Examples include software responsiveness, throughput, runtime, and storage capacity. So, in conclusion, if the device on which the application is running on, has the minimum requirements for an appropriate behavior, the performance should be consistent. All of this is due to the fact that, even when a large amount of raw sensor data is transferred, it is small in size.

Scalability

Scalability is a non-functional attribute of a system that characterizes its capacity to manage growing (and decreasing) workloads effectively. Once this application will be published as a finished product it will always respect this non-functional requirement, because it is designed that way. While at the moment the application is currently in a demo state it supports a finite number of users because the system for publishing data

from the customer's interface is used free of charge and has certain limitations. This is, however, not relevant because the system would cope even if all the users of the application will send data at the same time. This is possible due the implementation of the system, because it is done in such a manner that it can support any number of users.

Availability

This feature specifies how long the system runs, how long it takes to rectify a defect prediction, and how long it takes until the application fulfill the user expectations. As soon as the application will pass the demo phase the server side will be hosted online so it will be available at any time. However, in the demo phase the server will be hosted on a local machine so the online status of the server will depend on the availability of the local machine. The time between the user's request and the response from the system depends on how fast the intelligent component returns feedback, so it depends on the technical specifications of the system. In the demo state of the application the server is hosted on a local computer with performance components.

Serviceability

This characteristic illustrates how simple it is to provide service when it is required. Therefore, this requirement is mandatory because the purpose of the application is to provide customers with an easy-to-use system to be as attractive as possible in terms of ease of use. But still to provide a precise and satisfactory answer.

Security

This is one of the most important requirements because no system should be sabotaged. Once there is a vulnerability that the malicious intentions of certain individuals can exploit, the application will no longer be trusted by the users. First of all, one of the elements that ensures security is the fact that on the user side, it does not have many options to enter data to be processed directly by the server, the data that reaches the server is taken from the device's sensors, and they cannot be modified by the user. Secondly, on the server side, security is provided by the frameworks used in implementation, which according to the manufacturers ensure proper operation.

Evaluation metrics

Evaluation metrics are used to measure the accuracy of the prediction of human activity performed by the user done by the system designed. This is done in two ways, one in which the correctness of the prediction is checked humanly and one in which the time taken by the system to predict the activity.

Firstly, the correctness of the machine learning component is done based on the feedback of the model, and was done during the developing stage. During the training stage, data was collected and labeled, and split into two parts training data set and test data set and based on the well-known algorithms which provide a probability output. The higher the probability, the more accurate the model will predict the custom data provided by a user will be. Considering that the model developed is based on time series some special measure methods were applied.

Secondly, the time during which the system predicts the label of activities performed by the user based on the data provided by the sensors from the user's devices is measured as well. This is done by saving the system's time when the data is consumed by the server, and also saving the system's time when the prediction is completed. Having the two values, an absolute difference between the two is computed and that is the wanted result, because it represents the time taken by the application to finish the user's request.

Chapter 3. Bibliographic Research

3.1. Real time monitoring systems

In order to solve the serious problem of a sedentary and unhealthy lifestyle by identifying bad habits of an individual, a specific hardware must be used so human activity can be recognized and categorized. All studies in the last few years related to this vast field have begun to grow, so there are a lot of studies that support Human Activity Recognition with the help of smartwatches and phones. Human movement detection is crucial in a variety of fields, including healthcare, fitness, and eldercare. This goal may now be accomplished with the help of mobile applications. These apps provide users and others involved in their care a better understanding of everyday physical activity. One of the current research areas is the classification of human movements using motion sensor data. These sensors are integrated into smart watches so it is very useful to track the actions of a person who merely wears a smart watch. The term "Bad Habits" refers to behaviors that are harmful to one's health and prevent one from making full use of one's talents throughout one's life. The most harmful behaviors are those that are acquired at a young age and are difficult to break. Such behaviors bring significant harm to human living, including a loss of potential and drive, early aging of the human body, and the development of numerous diseases. Tobacco, alcohol, narcotics, poisonous and psychotropic chemicals are among the deadly substances consumed in such practices.

3.2. Human activity recognition using smartphones

In [1] the discussion examines activity detection systems that make use of the phone's inbuilt sensors, with a special focus on systems that are used for personal health and well-being applications. The main aim of the authors is to give a complex survey on the issue and to bring non-field workers up to date on the state of the art, possibilities, problems, and future themes in activity identification using mobile phones. First argument discussed in the paper work is definition and classification of each sensor of our everyday mobile phones. The three-axis accelerometer is one or even the most useful sensors used to detect movement detection. Although it was built into the phone with the intention of improving the user experience by changing the display's orientation based on the user's phone's orientation, it can also be used for activity recognition by inferring the user's movements, such as walking, standing, running, sitting, and even falling. The next important aspect in the paper was the process of the activity recognition who can be encapsulate as defining a set of movements by translate the data received from the sensor and to divide it into a set of different activities. The main steps to classification of the human activity recognition using a smartphone, described in the article are the typical ones. Training Data collected from a subject after that, the raw data from the sensor is collected and then sorted and labeled. After the Data Collection step is over. With the

help of raw data gathered the Training step can occur, so the Model Parameters can not be calculated and sent to the next and the last step Activity Recognition. On the other hand, there are problems that prevent optimal operation of human activity recognition using only smartphones as mentioned by the authors. Considering the big problem of a phone's battery life, collecting data from sensors is a big challenge, a solution mentioned is to systematic data collection in short intervals of time. Another big problem is to run classifiers on mobile phones, but considering the continuous evolution of smartphones this will not be a problem anymore. In the article are mentioned a lot of other burdens regarding this approach in which it is used only the mobile phone in order to recognize human movement. The same topic is discussed in [2], where focus is on how an individual always has a smartphone in his hand or in his pocket and how these smartphones nowadays have so many sensors that can help fulfil the state of the art. However, a new kind of problem is highlighted, that of not violating privacy of the users.

3.3. Human activity recognition using wrist devices

A different approach of the problem of activity classification is found in [3], because in this paper an analysis of the sensor placement in the wrist area is made but having the same purpose as the previous paper works mentioned above. However, the main interest in this article is the way the data is collected and how the authors find a new way to receive the raw data from the sensors, using only a smartwatch that was placed on the subjects' wrists. The main aspirations of this project were to record the human activity using different features of accelerometer which is worn on the subjects' wrists. A lot of attention must be paid to the features selected for collecting the raw data from the sensors. Following the classification of different movements is done by two different classifiers, Decision Tree and Artificial Neutral Network. The data used was received by a Ez430-Chronos watch from seven participants who did the normal movements that a human does every day. The conclusion of this article is that there is potential to use one single wrist-worn sensor in order to identify the activity of a human body but the difficulty encountered is that only the basic and simple activities can be recognized. An article similar to the previous one is [4] where the creators also used a single tri-axel accelerometer. The method used here starts differently from the others because after the digital signals are collected from the accelerometer, these are noise filtered, meaning that the values that are not normal are removed. However, as before, a Decision Tree is used in order to get the classification system, which is divided in three classifiers. Adaptive Sliding Window algorithm is adopted to optimize the usage of sensor by using the adaptive sliding window technique, this involves detection of increasing or decreasing in the acceleration movement. The best window size is achieved by the authors by doing some tests and applying some mathematical formulas on the test results. In this study, the authors tried to beat the limitations that were met in other existing works who used the same technique, that of an adaptive sliding window, by dynamic resizing the window dimensions. The goal of the paper was achieved nicely and the results were great, but again as in the previous article, our focus must be on the way that the data samples were collected using a single accelerometer and how the corrupted raw data was not taken into account. Just like previous studies, the first part of [5] is about how bad smoking is and the combination of smartphone and smartwatch is used again. A new method of approaching this topic is present here, where a two-layer hierarchical method is used, a classifier is used to detect the smoking segments and another classifier is used to identify

the misclassified segments. Based on the assumption that smoking is a set of repetitive segments of hand movement if a segment is misclassified then the neighboring segments are checked as being labeled as smoking or not, this is called A Hierarchical Lazy Smoking Detection Algorithm (HLSDA).

3.4. Human activity recognition using smartphones wrist devices

According to [6] the best way to recognize a wide variety of activities of the human body with a high efficiency and accuracy it is desirable to use both the smartphone, because it is placed in the pocket and from there is easy to record repetitive pattern of motion for activities as walking, running or siting, while a wrist worn device is more reliable when it comes to actions done using just our hands for example smoking or drinking or texting. [6] started the study about motion recognition by splitting activities in two categories, simple and complex, the difference between them is that the complex activities are not very repetitive for a long period of time and almost all the time hand or wrist movement is included. As discussed in [4] the sliding window technique is used here as well. The data collecting stage was done likewise all the other mentioned studies, but the difference was that every subject had two devices recoding the activities not only one as before and in addition the actions from the complex category ware executed and recorded. After results are great, indicating that by combining the wrist recording position with waist position is a great improvement for recognition of basic activities and for an even greater result combining the gyroscope with the accelerometer and with the linear acceleration sensor. Nevertheless, the reference position for recording was the wrist one, because with the device placed in the pocket near the waist position, they couldn't record activities done only by the hand of the subjects. The limitations are those of a controlled environment project, so if someone eat while on a bike or while walking, or someone eats whit the fork in the right hand and the knife in the left hand. Finally, the purpose of mentioning this paper is to emphasize that the usage of both devices: smartwatches and smartphones is the option when this type of subject, also usage of all the sensors related is crucial because complex actions can be recognized that way. A comparable study is [7] that was done in 2014 where the recognition of smoking is studied using only sensors on a Smart Watch. The first part of the paper discusses how harmful to health is smoking, similar to how we did in a paragraph above. The main challenge and purpose are to recognizing smoking from the rest of the activities done by hand during the day, thus adding a contribution to this subject by recognizing smoking by hand position. The device used in the data collection step is a wrist band having only an inertial measurement unit on it. The segmentation approach was used to recognize smoking among all activities, this is done by recording the smoking segment gesture from a rest body position until a final position, after each segment is extracted, it's duration, displacement, angle and velocity is calculated and recorded and labeled afterwards. In the end, the evaluation had three parts, the first is to detect the singular movement of smoking, the next is to recognize the entire smoking session and the last is to compare the result with other methods from other studies.

3.5. Artificial intelligence for human activity recognition

An important part of the process of human activity recognition is the classification and the interpretation of the raw data from the sensors. As one of the most common ways to approach this kind of issues is to use what Artificial Intelligence has to offer, because is a fast and very accurate regarding the data interpretation. As a firs example, [8] is an essay that tried to obtain real-time observation of the human activity, in order to achieve such a magnificent goal every part of the project must be fast and clear. First and foremost, the data preparation stage has to be well executed. The authors came up with using all three types of inertial measurement units' sensors for data collecting: accelerometer, gyroscope and magnetometer, the data received by these sensors was sent to the cloud through a wireless connection. After row signals are received by the server, it is processed by applying several filters, first one is the Low-Pass Elliptic Filter which splits the raw acceleration in two parts: gravity and liner acceleration, next used filter is the High Accuracy Attitude Filter (AHRS). In [9] is explained how AHRS is correcting and predicting filter for activity reference system that use data from all the sensor presented in the previous paper, [1][6]. Its implementation is vastly detailed in two different methods, but the conclusion regarding this filter is that it has very low computational costs and a high accuracy. To accomplish the desired result, the authors came with an idea to compress data by deleting the similar portion of signals having the same name. Besides, in order to boost the process speed and classification accuracy the writers designed a deep convolutional neural network to convert the raw data received from the sensors into labels for the activities performed by the users. Likewise, the main topic of the [8] paper is how unexplored is the potential of the deep learning for recognize the activity of humans using wearables like phones or writ worn devices. [10] was published in 2016 and at that time, accuracy of motion detection outside of a controlled environment was a total uncertainty because a lot of difficulties occurred, like background noise, undetectable positions of the device and many others. The authors emphasize that the smartwatch technology is in a constantly expansion, so more complex algorithms and programs can be run over. The algorithm proposed and used was Restricted Boltzmann Machines (RBM), is widely used in the deep learning filed being a very helpful when it comes to classification and filtering, the authors compared the RBM with other algorithms, and it turned out that it had the best accuracy. The next important step in this chapter is to examine studies that discuss a similar topic to the one proposed in this paper. [11] is a study made in 2015 where the purpose is to identify human movements using both a smartwatch and a smartphone, the authors were trying to identify thirteen activities divided into two categories complex and simple activities. Complex activities are the one that need more attention and more sensors the once studied in this paper are: smoking, eating writing, typing, drinking and talking and the simple ones are walking, jogging, biking, walking upstairs or downstairs, sitting and standing. The main goal during the data collection step was to test the performance of different combinations of sensor or devices, the data was collected from five different subjects who did all thirteen activities for different periods of time. The outcome of the study was that there has not been a great improvement in the recognition of simple activities when it came to using only the smartphone or the smartwatch but it was huge for complex activities data set. Another recent study is [20], where is investigate the main causes of Office Workers Syndrome (OWS) and how to track it using smartwatches and Human Activity Recognition (HAR), because the paper was done in 2018 it has a wide support provided by others studies in HAR. This time, the data

was collected from ten participants all wearing Apple Watches Series 2 having WatchOS 4 version 4.2, after the sensors received information, a smartphone was gathering it via Bluetooth and sent to a server where features like mean value or standard deviation was extracted and processed by a Decision Tree, and based on the models it alerts the user if something like sitting for a long period of time was detected. A related study [12] was made in 2020 in which the same topic is approached but with a different approach on the deep learning side.

3.6. Relation between health problems and smoking

The main purpose of this paper is to combat a bed and unhealthy lifestyle by recognizing the occurrence of events that are harmful to human health, especially targeting the desk workers, so in order to do that, we have to study which are the most common bad habits encountered in their lifestyle. The first serious problem discussed in this paragraph is sedentary behaviors among desk workers, the first work discussed is [13]. The main topic in [13] is that sedentary behavior can be associated with cardiovascular disease, at the beginning of the paper is presented how studies attested how among people who don't have the required weekly physical activity of 150 minutes have a bigger chance for cardiovascular disease (CVD) incidents. Supporting the subject, in the paper is also discussed how sedentary behavior is corelated to the cardiovascular risk factors, a lot of causes like cholesterol and diabetes are related to sedentary as many studies attest. The main source for this article is [14] where experimental results and studies done on the population are presented. A big problem as the authors say is that someone can have an active lifestyle and still be predisposed to sedentary risks, this is possible because if during work hours he or she works at the office and don't take systematic pauses to have at least ten minutes of physical activity and in the evening goes for a run, the amount of uninterrupted sedentary time is too much and still can cause the same problems as not having an active lifestyle, as studies confirms. An intervention proposed by the writers is to reorganizing the traditional ways of designing offices, by changing the buildings design and the furniture design and too also have an activity permission workstation. However, the main problem is how much people working in the office sit down, in [15] prolonged sedentary behavior is presented as a reason for disappointing health results. This study was done based only on women working in the offices of a Scottish University, and the outcome was that the longest periods of time spent sitting are in the morning. The same kind of study was done in [16], but this time the subjects were not just women but it was done on equal number of women and men. The results were frightening, the participants reported that they spend almost all the work hours sitting, to be more precise the average was seven and a half hours out of eight of their working time. Moreover, there is a steady and considerable increase in the study of sedentary lifestyle, one of which is [13], where the purpose of the paper is to find a connection between sedentary and health problems among young people between two and eighty years old. This study is done on this topic because nowadays more and more kids are spending their free time in front of a computer or a TV. In the closure, the authors sum up the results of several studies in which the idea of reducing the amount of time in a sitting position should be reduced in order to protect children's health because if someone starts to have problems in the early stage of life, when they reach adulthood, they will not know how to fight it or they will not see anything wrong with it and will develop health problems very early in adulthood. The last but not least significant problem that affects health, discussed

in this chapter is smoking during work hours. Everyone knows how dangerous smoking is, it is so dangerous that there is also anti-smoking advertising on cigarette packs, it's not just a known thing, it's a truth supported by many studies. [17] is research of the effects that smoking, active or passive, have on hart rate variability (HRV), it starts by mentioning some of the articles that brought biological evidence on how smoking cause health problems of cardiovascular nature. HRV is calculated based on the time difference between repeated heart beats. The effects of the 2 types of smoking, active and passive, on HRV are analyzed, active smoking is highly researched and it is biologically proven that it slows down HRV, also passive smoking slows down the HRV and sometimes it has acute effects and is associated with arrhythmia. As a conclusion of this study, both passive and active smoking slow down the heart rate during the act and even after. This work seeks to improve the health of people, especially those who work in IT and in offices in general. [18] is a study done back in 2013, that captures the link between smoking and working hours, it starts by presenting reasons why smoking is the source of many heart and lung diseases. The main theme of this paper is to find and demonstrate that exists a link between smoking and how many hours do someone work per day. The data was collected from surveys from both Australia and the United Kingdom, where working people have to fill in how many hours do, they work, how many cigarettes do they smoke during and off work. The result is to be expected, because based on the forums, the more someone works, the more willing they are to smoke and the lower the chance of quitting. [19] is another study that addresses the same topic, but this time it was made in South Korea. The same method of collecting data was used on about 5,000 Koreans, whose personal data were recorded and who were asked how different questions regarding their lifestyle. The outcomes found that the average cigarette smoked is influenced by age, by monthly income and by alcohol consumption, but one surprising factor that has been revealed following the surveys is that people who work in physically demanding jobs are more likely to smoke. However, the main factor in smoking remains the number of working hours and is analyzed based different methods like, time domain, where the seconds between heart beats are counted, statical indices and frequently domain where the distribution of power is monetarized.

Chapter 4. Analysis and Theoretical Foundation

This chapter is one of the busiest chapters of this license because in order to have a clear and successful research, the theoretical foundation of the project must be well constructed and clearly understood.

The main aim of this chapter is to introduce the read into the libraries, frameworks and concepts used in order to develop the final application presented in this thesis. Furthermore, in this chapter will be presented the main use case of the application, where it will be explained how the purpose of the project is achieved by any user. Even more, a conceptual diagram will be explained and illustrated in the end of the chapter.

Before each technology and component is studied and analyzed and then described, the entire system will be described. The first part descried is what the user firstly interacts with, the client side which is an application designed in Android Studio, from where the user can interact with the system. The client side communicates with the server through a message broker called RabbitMQ, being one of the most used message brokers in the industry. On the server side, the data from the user is processed and after that, a response to the user request is send back to the client side.

Therefore, this application can be called a distributed system because it has independent components that communicate one with another by shearing resources in the form of messages so it can achieve the goal they were designed for.

4.1. Technologies used

4.1.1. Android Studio

Android Studio[1] is one of the most popular Integrated Development Environment (IDE) for developing applications for almost all devices that run an Android operating system. Android Studio is actually based on IntelliJ IDEA[2], but specializes in developing for the Android platform, so as a result it inherited IntelliJ's powerful code editor and developer tools and even more features for developing the best applications.

Android Studio is a very friendly IDE where any type of programmer can easily code and use all its features. Furthermore, the nicest part about Android Studio is that it has a lot of documentation done based on it. Because it is so widely used around the world, there is online support for almost any problem that anyone may encounter, and in the special case where there is no answer to the problem sought, there are a lot of specialists in this IDE who are willing to help for free.

But apart from that, here are some of the most useful features that this environment has to offer:

[1] `https://developer.android.com/studio`

[2] `https://www.jetbrains.com/idea/`

- Visual layout editor, allowing the user customize the application's interface by dragging and dropping User Interface (UI) elements into an editor that generate XML files by hand. The XML files are the files where the code for UI pages is written.
- Application Package Android (APK) Analyzer, is a smart feature that reduce the size of the application and also the time spent on debugging.
- Android Emulator, letting the user to test an application on multiple virtual devices so the application can work well across many different devices. So as a result, it becomes very helpful when somebody who don't have access to multiple devices.
- Flexible build system, is a feature that lets the user to configure different builds of the same project for different devices, helping with the management of the dependencies and configurations.

4.1.2. Smartwatch Sensors

The smartwatch presented above has a lot of interesting and helpful features, but the one this thesis if focused on it's the sensors of it. In particular the accelerometer sensor and gyroscope sensor, because they are the one that provides the raw data from the user that is processed and used for predicting the human activity. Both of these two sensors mentioned above are inertial sensors, which are based on inertia and relevant measuring principles.

There are two types of sensor devices Piezoelectric sensors and Microelectromechanical systems (MEMS) based. The piezoelectric devices have a crystal layer made out of positive and negative ions, called Piezoelectric material placed between two electrodes one with positive charge and one with negative charge. When a force is applied this structure distorts and the electrons of these ions shift around which generates an electric charge (output voltage), that charge is proportional to the force that is being applied so by this it can measure the force mentioned. This mechanism is visually exemplified in Figure 4.1 below.

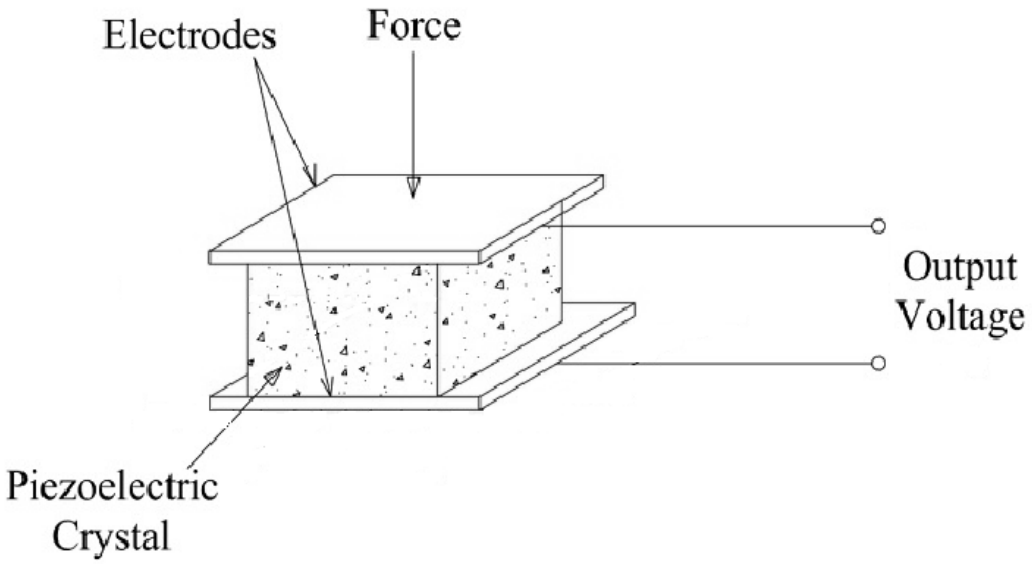

Figure 4.1: Arrangement of piezoelectric material between electrodes.

Microelectromechanical systems (MEMS) are microscopic mechanism measure force by measuring changes in capacitance. It is composed by two sets of plates, one set which is fixed called "Anchor" and one set which is spring loaded, called "Mass" and can move along one direction at a time and respond to acceleration. The measure is computed when an acceleration is applied in a particular direction the Mass will move and the capacitance between the plates of the Anchors' plates and Mass' plates will change. This change between the plates is measured and processed and will determine the acceleration applied in the beginning. The system is demonstrated in Figure 4.2 by presenting a MEMS accelerometer when no acceleration is executed and then when the acceleration is executed.

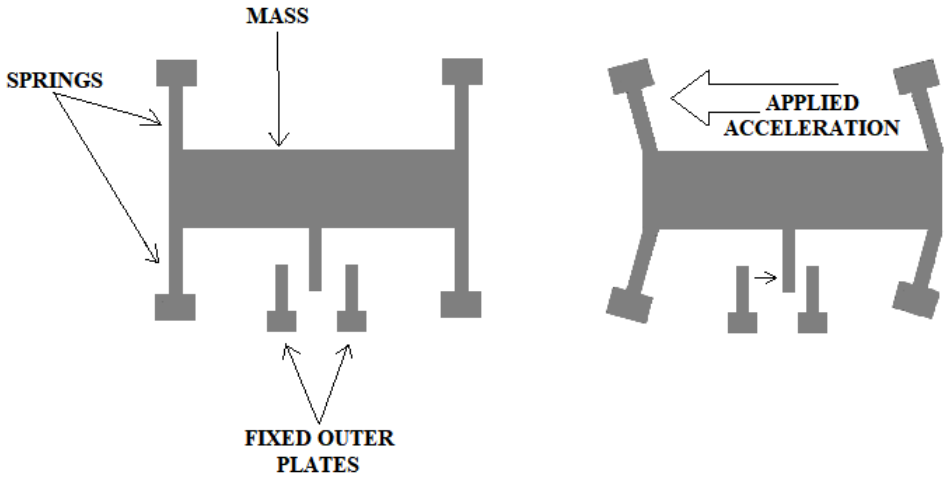

Figure 4.2: MEMS accelerometer structure.

Accelerometer

The accelerometer is measuring the physical acceleration, also known as "g-force", of an object. This device works by analyzing the acceleration of gravity, so it converts mechanical force into electrical signals. An easy-to-understand example, if a device that is not actioned by any acceleration it still measures an acceleration equal to 9.81 m/s2 straight upwards (on X axis of the accelerometer) being the gravity of Earth, denoted by g. The main purpose of an accelerometer is measuring the dynamic or/and the static forcers of acceleration. The difference between a static force and a dynamic one is that the static force will not change the size position or the direction of an object that the force is applied to.

The accelerometer is measuring the force mentioned above on three different planes, each plane denoted by X, Y and Z axis, any accelerometer being capable of measuring the force on one, two or all three axes at once. However, an accelerometer that measures the acceleration exerted on three axes is a system composed of three accelerometers that deals with each axis separately and in Figure 5.3 is depicted such a system.

The most common accelerometer technology is MEMS, which was described in an above paragraph, this type is also used on the mobile phone or smart device used, because it is super small, with dimensions between 1 and 100 micrometers.

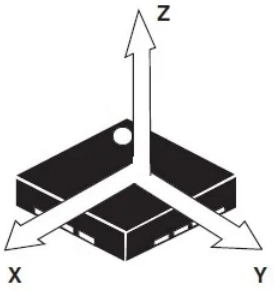

Figure 4.3: Direction of detectable accelerations

Gyroscopes

Gyroscopes also known as Angular Rate Sensors, are devices that can measure the angular rate using the Coriolis Effect, which is a force that is applied to an object in motion within a frame that rotates. In details we have an object that is moving in a direction with a particular velocity, plus an external angular rate is applied to the same object then as a result a new force will occur and the body will suffer a perpendicular displacement. This new force will cause changes in capacitance which will be processed and measured and will be the angular rate of the moving body. The output of a gyroscope, the angular velocity, is a measurement of degrees per second.

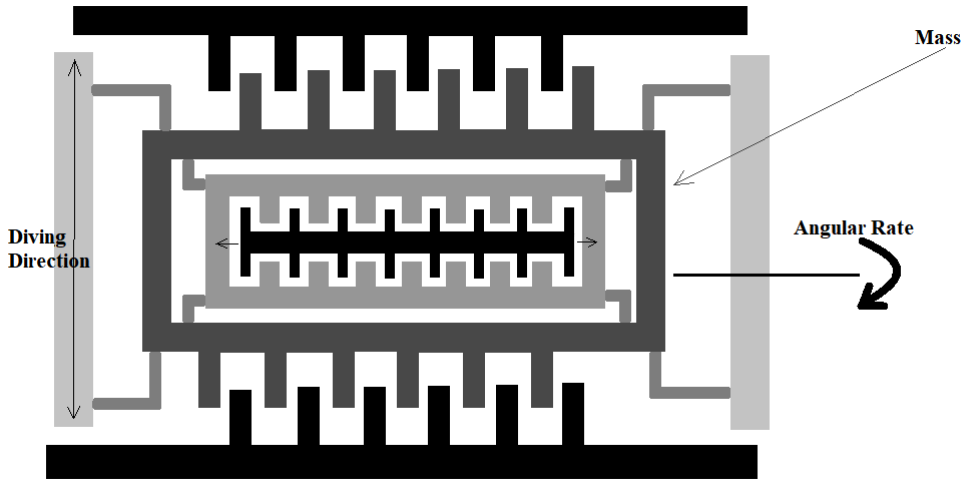

Figure 4.4: Microstructure of a gyroscope

Nowadays within almost all smart devices there is a MEMS gyroscope, we can have a closer look to its microstructure in Figure 5.4 In the figure we can observe the Mass, which in constantly moving or oscillating, when an external angular rate is applied to this system the mass will change its normal movement pattern and it will generate a perpendicular displacement.

4.1.3. CloudAMQP

CloudAMQP[3] is a platform to host RabbitMQ servers, RabbitMQ is one of the most known message brokers know in the industry. A message broker is a module that transform a message from the sender format to the receiver format protocol. The main purpose of a message broker is to have two different components of a system communicate one with another by exchanging messages. There are several reasons for choosing this platform, which are presented in the following:

- Management of RabbitMQ instance – CloduAMQP is providing a fully monitoring system which is ensures proper operation of each instance deployed on the platform.
- Easy to use – In only three easy to understand steps an instance of a message broker can be deployed and up and running. The three steps mentioned are: creating an account on the platform, creating an RabbitMQ server and completing a setup for the instance.
- Free to use – For a developing a small project, with few messages' communication, this platform offers a free to use plan with up to 100 queues at a time and maximum 10 000 queued messages.

In order to explain why RabbitMQ is the chosen communication technology, a detailed explanation on how Client-Server Communication works, will be given. When a client is waiting for a message, it is blocked until it receives a reply. The request made by the client is transmitted and received by the server, the server prepare a response and when it is ready it is transmitted back to the client which is unblocked. This type of communication is called synchronous because the exchange of messages is happening in real-time. This being said this technology is fitting very well in the thesis application. A visual exemplification of the system is present in the Figure 5.5 where we can find the main actors and the flow of message transfer.

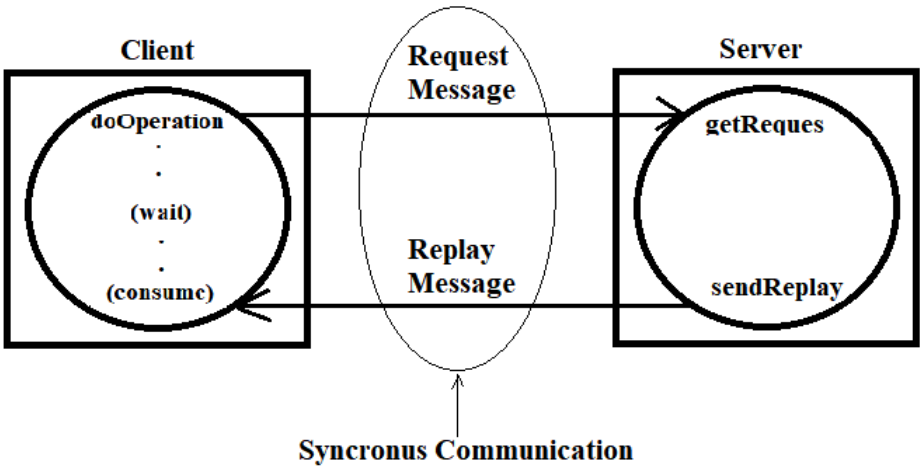

Figure 4.5: Client-Server Communication

There are two methods that stand out: Point-to-Point Messaging and Publish-Subscribe Messaging which both handle communication between two or more instances. The first one has a better use when there is only one application waiting or is sending a message and the other is better when more than one application is implied. In the

[3] https://www.cloudamqp.com/

current stage of the application, Point-to-Point Messaging is the better option to choose because the project is designed for presentation purpose only, so only one Consumer will be present. Message queue are the foundation of this method.

4.1.4. Sockets

Sockets are used to transfer messages or communicate over a network, meaning that we can use it to send messages to a device in our local network. It also provides an inter-process communication (IPC) which allow different process to manage data shared or produced by other processes. Basically, applications can use IPC, which is divided between clients and servers, where the client asks data and the server answers. It is a very powerful tool for exchanging data, when a client needs data, it starts listening to a connection established in advance by the server with that customer. Typically, in a localhost environment, the client listens constantly on a particular port on the same machine that the server is writing the data.

In this project we decided that a socket connection will be useful to achieve our goal, because it does not require much knowledge to understand the technology and to implement it, and another great reason for choosing this method of communication, is that this type of communication works continuously and sequentially, meaning it is perfect for our needs. Because the server in java must always be turned on and the classifier developed in python must send the results permanently to the client, an uninterrupted connection between the classifier and the server is needed.

In the following, it will be presented in writing how the data flow reaches from the customer to him. When the client opens a communication with the server, and it starts sending data, it is processed to meet the format criteria of the classifier. Once they are processed, they are sent via sockets, where as a host is the localhost and on the port 2004 which is the standard port used for this kind of communications. Once, the classifier receives enough data to iterate the label assignment process will open a separate connection from the previous one. Through this connection the client will receive on the application interface the answer to his requests.

4.1.5. IntelliJ IDEA

IntelliJ IDEA is a Java IDE (Integrated Development Environment) that brings a lot of useful tools that a developer uses on every project, because it makes his work easier. These features will be presented below.

- The first useful tool is the text editor which highlight the syntax and help moving between files in file system. Also, the code can be run in different ways, some of them are: run, debug and run with coverage, we have to keep in mind that Java is a compiled language so every time the code is run, the IDE is going to build the code for every user.
- Another code related feature is that when working with compiled languages is necessary to compile the code in order to see the compile errors, but IntelliJ provides a feature to signal that compile errors are present in the written code, by underlining the errors with red. Also, it suggests some common solutions to fix the error.
- Refracture, is another great feature which help the user to change an item throughout the project by a few simple consecutive tasks. When a user decides to use refactoring the IDE will provide a preview of all the changes of that item and in

chase of error the IDE will provide an easy-to-understand dialog with the list of the conflicts and a description for each.

Spring Framework

Spring[4] is the most used and known framework for developing applications in Java, being an open-source platform. It provides the patterns and the structure for any Java project and it deals with the things that the developer should normally think of. The main problems Spring is taking care of are:

- Application context dependency injection - by managing object instances and services using an application context and managing dependencies.
- Data access – by providing different ways to connect to a database, to query and many other actions performed on a data base. It is known as Java Database Connectivity (JDBC).
- Spring MVC – facilitates the process of creating a web page and REST APIs.

4.1.6. PostgresSQL

PostgresSQL[5], also known as Postgres, is relational database management system (RDBMS) being the most advanced open-source among all of the RDMS. It well known by all the developers in the domain being one of the long-lasting data bases out there, it has over 40 years of development. Because it is so popular, this database has a large user base, making it simple to handle typical problems. Another fantastic feature of Postgres is that it does not require a licensing fee for both personal and commercial use, which means that the user base is always growing. A very useful feature of Postgres is that it allows the user to save the data basses locally or on a server online.

An explanation of how data is saved in a database will be presented in the following sections. Data is stored in tables composed by columns and rows, where the columns are the attribute of each row, then the actual rows is the actual date.

In this project, a client is sending data to the server and waits for a response, the data sent by the client is supposed to be saved in, and then the data should be processed and after the system returns to the user a result, that data should be deleted. For all actions related to Postgres data management it is an easy to implement and very powerful solution.

4.1.7. PyCharm

PyCharm[6], also known as Postgres, is relational database management system (RDBMS) being the most advanced open-source among all of the RDMS. It well known by all the developers in the domain being one of the long-lasting data bases out there, it has over 40 years of development. Because it is so popular, this database has a large user base, making it simple to handle typical problems. Another fantastic feature of Postgres is that it does not require a licensing fee for both personal and commercial use, which means that the user base is always growing. A very useful feature of Postgres is that it allows the user to save the data basses locally or on a server online.

[4] `https://spring.io/`

[5] `https://www.postgresql.org/`

[6] `https://www.jetbrains.com/pycharm/`

An explanation of how data is saved in a database will be presented in the following sections. Data is stored in tables composed by columns and rows, where the columns are the attribute of each row, then the actual rows is the actual date.

In this project, a client is sending data to the server and waits for a response, the data sent by the client is supposed to be saved in, and then the data should be processed and after the system returns to the user a result, that data should be deleted. For all actions related to Postgres data management it is an easy to implement and very powerful solution.

NumPy

NumPy[7], also known as Numerical Python, is regarded the fundamental package to utilize when working with scientific calculations. It also serves as the foundation for the majority of scientific calculation libraries, making it a critical library. It is a library which deals with multi-dimensional array, letting the user to store all sorts of data, even n-dimensional arrays. The main competitor to this library is Lists, but it is much, much slower that NumPy because it stores all sorts of metadata that takes a long time to read and store, when NumPy stores all the metadata in the same place as the data itself, and this makes it use less space in computer's memory. Another example of performance is the fact that this library does not check the type every time iterates through objects. Contiguous memory, also gives this library speed because it allocates consecutive blocks in the memory, so when a program needs the variable stored by NumPy is easy for it to find all of the data, because almost every time a program needs all the data at once.

Pandas

Pandas[8] stands for Python Data Analysis Library and it is another useful library when it comes to data science in Python, Pandas let the user to load, prepare manipulate models and analyze data. According more flexibility when working with big data sets in Python, it is based on the functionalities of Excel, but is better because Excel struggles when it comes to large amount of data. For these reasons almost all projects that work with data are using Pandas, the main files that require working with data are: data analysis, data science and machine learning. The most powerful functionality it has to offer is data frames, which gives the user many advantages to process the data. It also have the feature that allows any developer to transform normal data into multi-dimensional data, combined with NumPy is a very powerful toll for developing any kind of data related project.

Keras

Keras[9] is an interface that allow the user to easily access and customize the machine learning frameworks like: TensorFlow, Microsoft Cognitive Toolkit or Theano, these frameworks are also the backends that do the hard work that Keras is using. All the mentioned backends are complex and hard to understand, that is why Keras is so popular, because it's an API that let the user develop any kind of program that needs machine learning or deep neural networks to easily use these backends. This interface is one of

[7] `https://numpy.org/`

[8] `https://pandas.pydata.org/`

[9] `https://keras.io/`

the most choose option to work in the domain by beginners but also by professional developers.

4.1.8. KOPSET Optimus

KOPSET[10] is a company developing different smart wear devices like smartwatches, headphones, and much more. However, the brand focuses on developing smart watches, the first model they launched in 2018 being a KOPSET Hope running an Android software. The Optimus model was a big hit for the brand, in this paper this model was used to take test and study data. This model has to offer a lot of useful features being a good hardware support for this thesis practical project.

KOPSET Optimus, is a smartwatch which runs Android 7.1.1, commonly known as Android Nougat, and has converted its Java Runtime Environment (JRE) TO OpenJDK, the official open-source implementation of the Java platform.

In Table 4.1 are specifications collected from the manufacturer company site.

Category	Details
Model	KOSPET OPTIMUS
Size	o48mm, Thickness 15.6mm, Length 274mm
Color	Black
Watch Case Material	Ceramic bezel + Plastic shell
Watch Strap Material	Silicone
Operating system	Android 7.1.1
CPU	MTK6739 1.2 GHZ
RAM	2GB
ROM	16GB
Camera	No
Battery	800mAh Polymer battery
Speaker	Yes
Microphone	Yes
USB Type	Pogo
Standby Time	Android mode: 1-2 Days, Lite Mode: 2-4 Days
Sensor	G-sensor, Hart rate sensor, pedometer, special action, trigger sensor
SIM Card	Nano SIM card
Network	GSM WCDMA FDD-LTE TDD-LTE
WIFI Support	2.4GHz / 5GHz
Bluetooth	Ver 4.0
GPS	GPS/GLONASS

Table 4.1: KOPSET Optimus specifications

4.1.9. Data Set

The data set was taken from a university in the Netherlands called the University of Twente[11] . It provides several sets of data, source code, and applications, which, with

[10] `https://www.kospet.com/`

[11] `https://www.utwente.nl/en/eemcs/ps/research/dataset/`

appropriate confirmation, can be used for research purposes. The data set consists of values recorded by both the smartwatch and the smartphone. These numbers reflect data acquired by several sensors on both devices, however only the data from the smartwatch's gyroscope and accelerometer were used in this thesis research. This data set was acquired from eleven different people of varying ages, who were doing a variety of tasks in order to provide a firm foundation for training the prediction model. In Figure 4.6 is presented a table with each set of actions performed by each participant, alongside with description data about each of them.

Every activity involving a hand movement was performed with the hand from which the wristwatch receives data. An explanation of how each task was carried out is provided below:

- Smoking (S) – It consists of a series of puffs (carrying a lighted cigarette to the mouth and sucking the end of it) which is repeated until the cigarette is extinguished. This practice was carried out in four distinct scenarios: standing (SSD), sitting (SST), walking (SW), and participating in a group chat (SG).
- Drinking (D) – Each participant was holding a glass or a cup filed with water and was requested to finish drinking each one at his own pace.
- Eating (E) - Each participant was given a cup of soup and told to finish it at his or her own leisure.
- Walking (W)
- Sitting (ST)
- Standing (SD)

	Activities Performed[Legend]	**Duration of Each activity (minutes)**	**total duration for all activities (minutes)**	**Gender**	**Height (cm)**	**Age (years)**	**Cigarette Usage**
Participant1	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G, S_W, W	43	430	male	180	25	8-10 per day
Participant2	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G, S_W, W	47	470	male	172	30	0-10 per week
Participant3	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G, S_W, W	48	480	male	175	25	2-6 per day
Participant4	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G	37	296	male	156	28	0-10 per week
Participant5	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G	18	144	male	174	23	18-20 per day
Participant6	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G	20	160	Female	164	20	3-7 per week
Participant7	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G	16.8	134.4	male	181	20	9-11 per day
Participant8	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD, S_G	20	160	Female	172	29	4-6 per day
Participant9	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD	24	168	male	167	35	0-10 per week
Participant10	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD	19	133	male	181	27	7-12 per day
Participant11	S_{ST}, S_{SD}, D_{ST}, D_{SD}, E, ST, SD	18.6	130.2	male	170	45	15-20 per day

[Legend] Smoking while standing (S_{SD}), while sitting (S_{ST}), while in a group conversation (S_G), while walking (S_W), SD:standing, ST:sitting, W:walking, E:eating, D:drinking.

Figure 4.6: Details about dataset

4.1.10. Data Gathering

In the data set mentioned above, the data was collected using a smartwatch and a smartphone, all the participants wore the smartwatch on the dominant hand used for eating, drinking and smoking. The smart phone was placed in a fixed position on the wrist of each participant, slightly above the hip. According to the following explanation, the watch must be put on the hand as accurately as possible for the data to be accurate. Wearing the watch correctly guarantees a good heart rate reading, thus it must be worn properly. At a normal use, the wristwatch should be worn one finger above the wrist bone and during exercises it should be wear two fingers above the same bone. The main features that are leading to misreading of the sensors are perspiration, body fat, water or any tattoos. Basically, any kind of layer that is present between the smart wear and the skin is bad for the correctness of the data.

All the sensors that ware available on each device ware sent to the system and saved in the data base. Each activity performed has a fixed duration, different for all

the participant. A total of 45 hours was reached after recording the tutor activities. The first four participants were repeating the activities ten times and the rest only five times. All participants have ages between 25 and 40 years and none of them have any health issues. For testing the model and also for training it, personal data was collected using a KOPSET Optimus.

4.2. Proposed Algorithm

Before explaining the method used to implement the algorithm, we need to better understand the reason why we chose this solution. Following the research in Chapter 3, we found that smoking is a continuous action that often has a repeated number of similar movements, these movements are to carry the hand in which the cigarette is, from the state of rest that mouth, then when the hand reaches the mouth, it stops in this position for a few seconds then the hand goes down to the initial state. In order to be able to classify this whole path correctly, you will need an algorithm that understands that it is necessary to analyze several data inputs from the sensors that record the movement. Which made us think of a classic time series analysis algorithm.

In the case of our time series, it is a data set that has as inputs values recorded by the same sensors with an equal time distance between each measurement. Most of the time the time series has a trend according to which we can realize a pattern, as mentioned earlier and smoking is an action composed of several short and repeated actions. This is why we cannot treat this type of classification like any other ordinary classifier that we have a label for each registered value. On the left side of figure 4.7 we can see that if we choose to classify a single entry (marked in red on all three axes) from the accelerometer sensor we cannot figure out what kind of action is taking place, but if we call to capture data as in the right part of the picture where several entries (marked with different shades of red) are sent as many classifications, we will have an impeccable accuracy of the classification of the activity done.

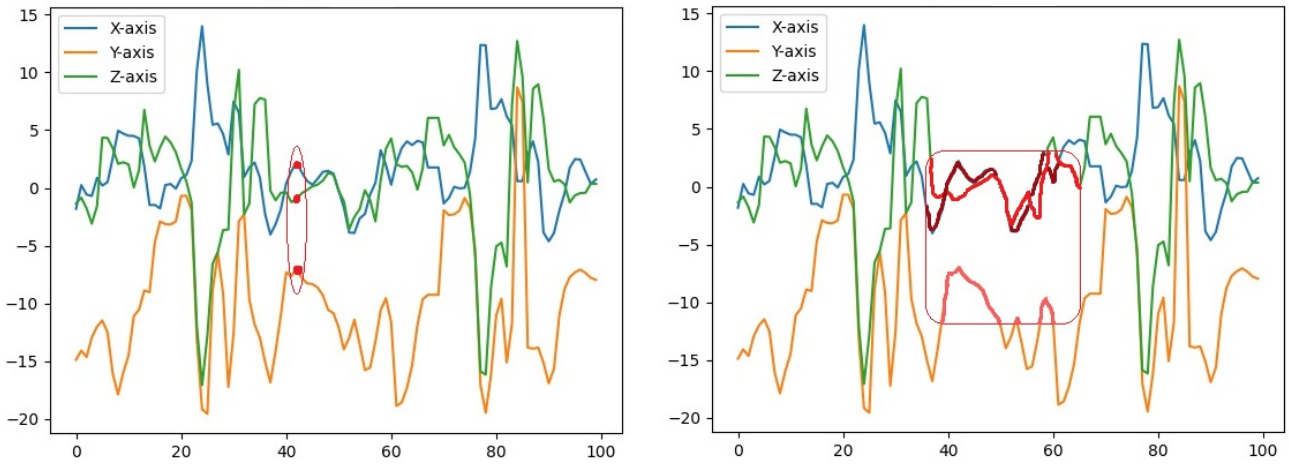

Figure 4.7: Accelerometer sensor variation during walking.

In order to be able to transmit to the classifier a suitable input containing several values from the sensor, we would need a window, these function as an interval that walks on the whole data set and each movement of the interval is used as input for model, this

window has a fixed size of 100 values. In the case of our application, sensors produce sequential values and count the time in which they were produced, which is why we will use this window to create batches for training, testing and validation.

4.3. Main application use cases

4.3.1. Activity Recognition

Brief Description

The user wearing a smartwatch that is connected to the internet, which is sending raw data from the accelerometer sensor and gyroscope sensor to a queue that is hosted by a server called RabbitMQ. From the queue the data will be consumed by a Java server that is consuming data at every 500 MS and storing it in a data base. Then the data is taken from the data base into a Python designed classifier, which based on the data returns smoking activity recognition. The found activity is then returned to the user interface through a similar queue as the sending one.

Primary actor

The primary actor in this use case is the direct User which is wearing the smartwatch. He or she wants to monitor the pulse during the smoking action.

Stakeholders and Interests

Two of the main stakeholders are:

- Health System: wants to have a more aware community regarding the main problems of smoking, by informing the smokers with the minimum knowledge about the damage caused by smoking.
- Diploma Project Supervisor: wants to have the main core of my diploma project working well.

Flow events

Basic Flow:
Use-Case Start: User wants to test the interaction with the software and hardware parts.
Step 1: User enters in the main menu of the application.
Step 2: User press the recording button.
Step 3: User starts the recording of the activity.
Step 4: User stops the activity.
Step 5: User press the stop recording button.
Step 6: Server side receive the data from the user.
Step 7: Server side process the data received and send to user.
Step 8: User receive the feedback from the server.
Use-Case End: New interactions are still welcome on the Main Application Menu.

Alternative Flows

1a. Different application problems:
1. User contacts the maintenance team to fix the application.
2. System stops functioning in this time period.
3. Go to Step 1.

2a. Recording don't start:
2a. Recording don't start:
1. Application signals error.
2. Go to 1a.

6a. Invalid Connection:
1. Client-side application signals error.
2. Go to 1a.

6b. No data received:
1. Signal Client-side application.
2. Go to step2. 8a. No feedback received:
1. Signal Client-side application.
2. Go to step1.

Precondition

- Actor has smart functional devices: should work normally and has no problems in order to use the application.
- Actor has access to internet: devices should be connected so the client can communicate with the server.

Postcondition

- System will return the result of the classification to the user.
- User will receive the result.

4.3.2. Visualization of Historical Charts

Brief Description

The user wearing a smartwatch that is connected to the internet, which is sending raw data from the accelerometer sensor and gyroscope sensor to a queue that is hosted by a server called RabbitMQ. From the queue the data will be consumed by a Java server that is consuming data at every 500 MS and storing it in a data base. Then the data is taken from the data base into a Python designed classifier, which based on the data returns smoking activity recognition. The found activity is then returned to the user interface through a similar queue as the sending one.

Primary actor

The primary actor in this use case is the direct User which is wearing the smartwatch. He or she wants to monitor the pulse during the smoking action.

Stakeholders and Interests

Two of the main stakeholders are:

- Health System: wants to have a more aware community regarding the main problems of smoking, by informing the smokers with the minimum knowledge about the damage caused by smoking.
- Diploma Project Supervisor: wants to have the main core of my diploma project working well.

Flow events

Basic Flow:
Use-Case Start: User wants to test the interaction with the software and hardware parts.
Step 1: User enters in the main menu of the application.
Step 2: User press the recording button.
Step 3: User starts the recording of the activity.
Step 4: User stops the activity.
Step 5: User press the stop recording button.
Step 6: Server side receive the data from the user.
Step 7: Server side process the data received and send to user.
Step 8: User receive the feedback from the server.
Use-Case End: New interactions are still welcome on the Main Application Menu.

Alternative Flows

1a. Different application problems:
1. User contacts the maintenance team to fix the application.
2. System stops functioning in this time period.
3. Go to Step 1.

2a. Recording don't start:
2a. Recording don't start:
1. Application signals error.
2. Go to 1a.

6a. Invalid Connection:
1. Client-side application signals error.
2. Go to 1a.

6b. No data received:
1. Signal Client-side application.
2. Go to step2. 8a. No feedback received:
1. Signal Client-side application.
2. Go to step1.

Precondition

- Actor has smart functional devices: should work normally and has no problems in order to use the application.
- Actor has access to internet: devices should be connected so the client can communicate with the server.

Postcondition

- System will return the result of the classification to the user.
- User will receive the result.

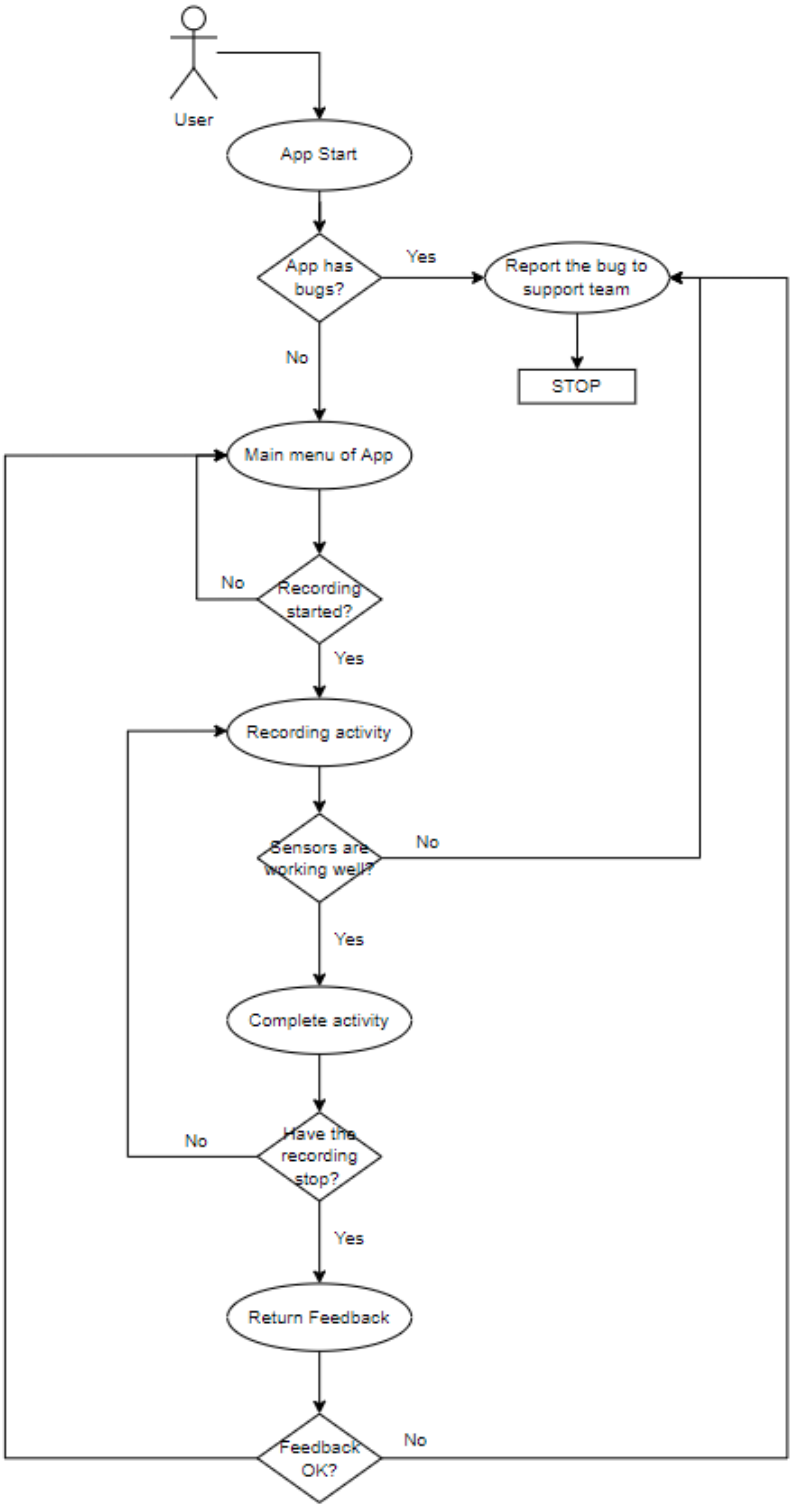

Figure 4.8: Activity Recognition use case diagram

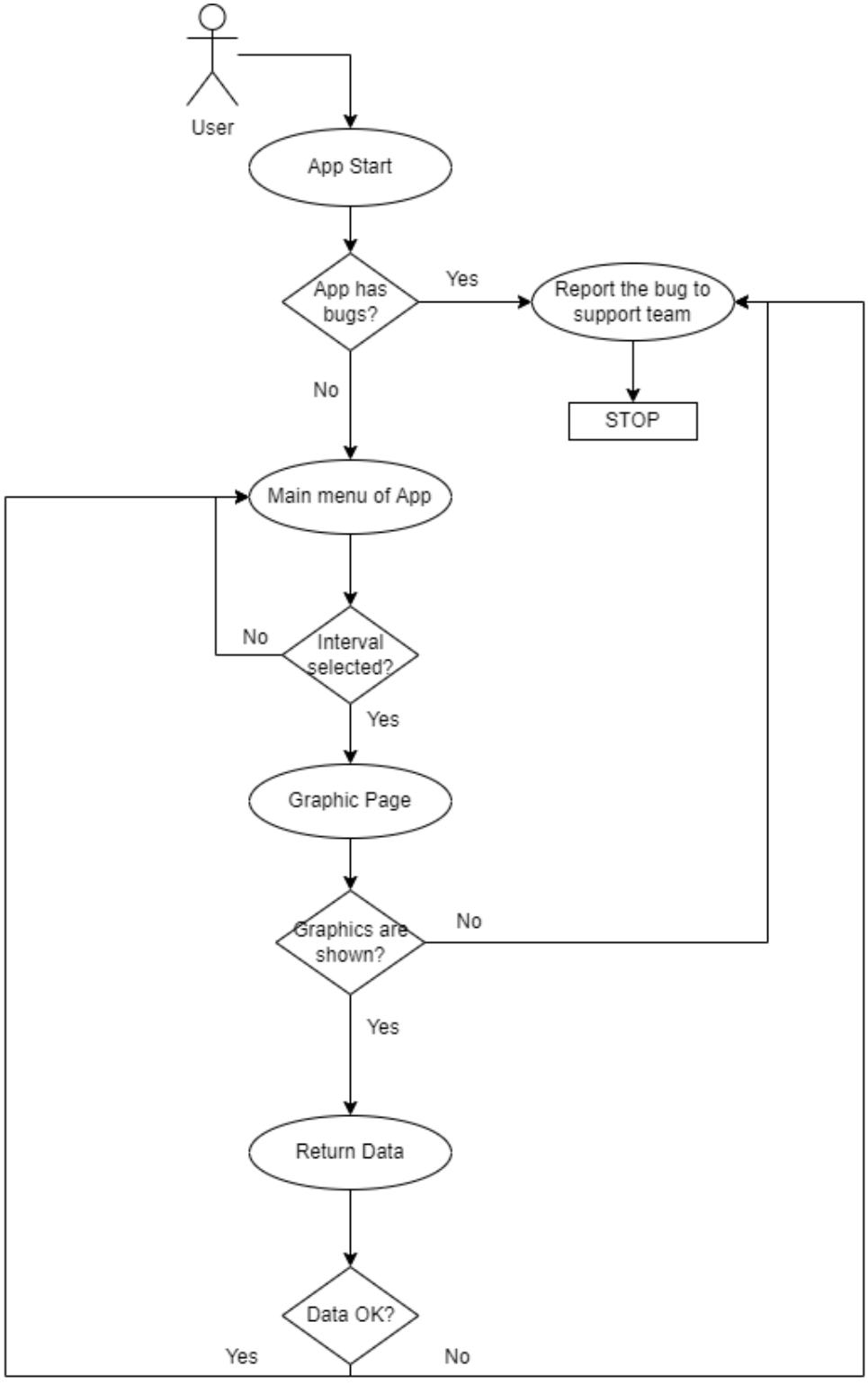

Figure 4.9: Visualization of Historical Charts use case diagram

Chapter 5. Detailed Design and Implementation

This chapter provides the reader with information related to the developing process of the application related to the study presented in this paper will be presented. The main subject found in this chapter will be the system architecture, the system implementation, the design pattern and lastly will be presented some code snippets that are important. All decisions made in the implementation process were made on the basis of the study made in Chapter 4: "Analysis and Theoretical Foundation" and on the basis of the knowledge acquired throughout the university.

5.1. Detailed Design

5.1.1. System Architecture

In the following paragraphs, all the major components will be presented and a detailed explanation will be provided for each of them. Also, ideas, methods, resources, and deliverables have all been mentioned and planned. The explanation will be done in such a way that anyone with a basic understanding of the area may replicate and comprehend how this method works. In the following illustration is presented the desired design, that is implemented.

The main components of the system are enumerated bellow:

- Main client Application
- Sensor data capturing
- Message posting consuming
- Data preparation
- Database Application
- Machine learning component

The main purpose of the frontend part in the system is to trigger a series of events in order to provide the user with the wanted results. The actor who starts the entire flow of the application has on the main screen of the application some very explicit buttons for an easy and delightful experience. The main role of the user is to send a request to the backend of the application. This request entails pressing a button to begin recording sensor data, making particular actions, and then pressing a button to notify the recording's completion. The frontend of the application is developed in Android Studio, which provides a very powerful framework for building user interfaces, called Layout Editor, it is easy to use and very intuitive.

Message broker is used for establishing communication between user interface and the sever side of the application. In both cases there is a need to publish and consume messages, so in the following the case, the user is publishing and the server is receiving(consuming), will be explained. Firstly, when the user presses the above-mentioned button, a flow of raw data from many sensors is sent via a queue directly to the server. In

order to create a working pipeline, we need to implement some support for the queue, also known as a Connection Factory, which contains a collection of connection configuration settings. The messages, in this case the sensor's data is push in the queue and taken out one by one by the subscriber(server).

The backend is implemented using the Spring library in Java language, the main purpose of this component is to maintain a good handling of the user's requests. This component is in good connection with the database system where the data from the user is stored and also with the module that detects the human activities done by the user.

The detection module will pick up requests from the backend and it will yield a label which will represent the name of the action done by the user. As soon as the label is conceived, the backend will publish on a different queue than the one described previously.

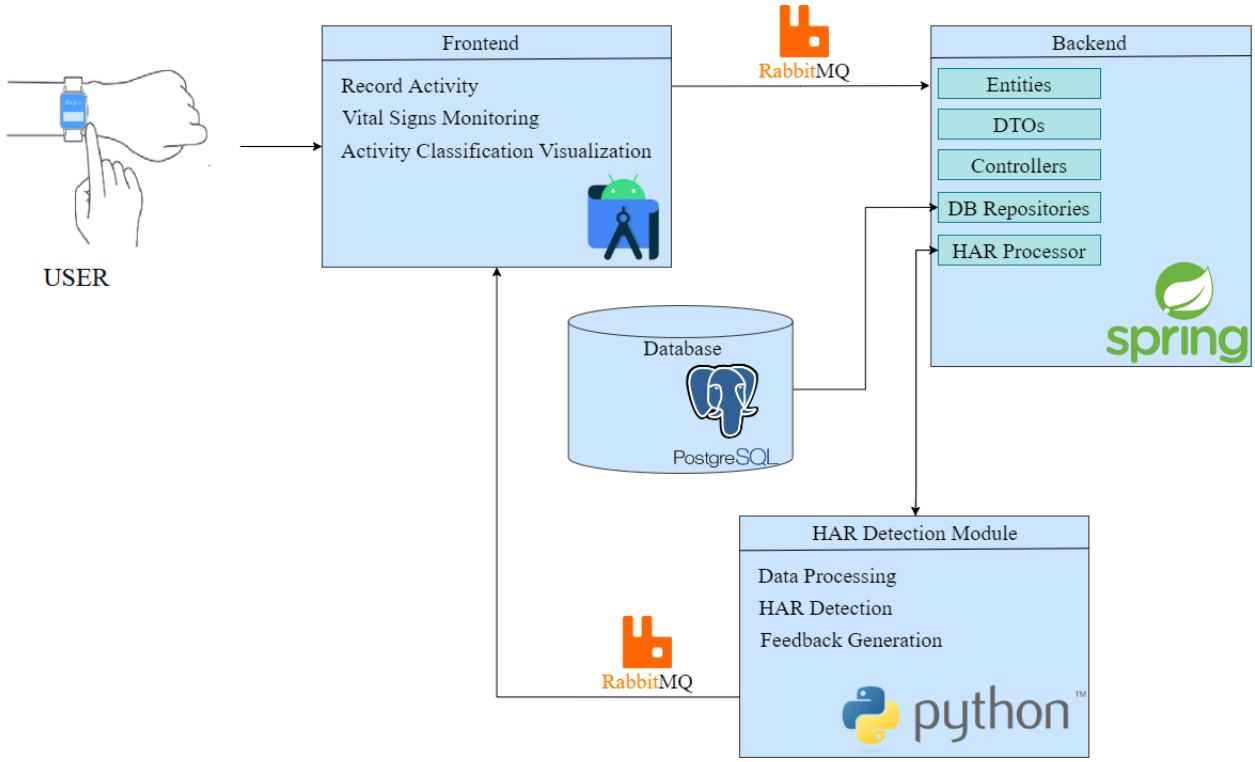

Figure 5.1: Conceptual architecture of the system

5.1.2. Deployment

In order to have a project that is continuously integrated, we have to set up a repository on Gitlab[1], because it provides pipeline features that will aid in the creation of a working application without the necessity of a physical server. This feature is known as Continuous Integration (CI), and it means that when new code, as well as a file for configuring the pipeline in which all the pipeline steps are specified, along with the commands and configurations required to successfully run that step, is pushed to a GitLab repository, a series of steps is initiated.

- Ensure that the project builds properly.
- Ensure that the tests run and are successful.

[1] `https://gitlab.com/`

- Ensure that there are no severe code style concerns.

After these steps are completed successfully, a new stage is initialized called Continuous Deployment (CD), is a method of software engineering in which software functions are regularly offered through automated deployments. We chose to use Heroku[2] servers for CD because several reasons presented in Chapter 4.

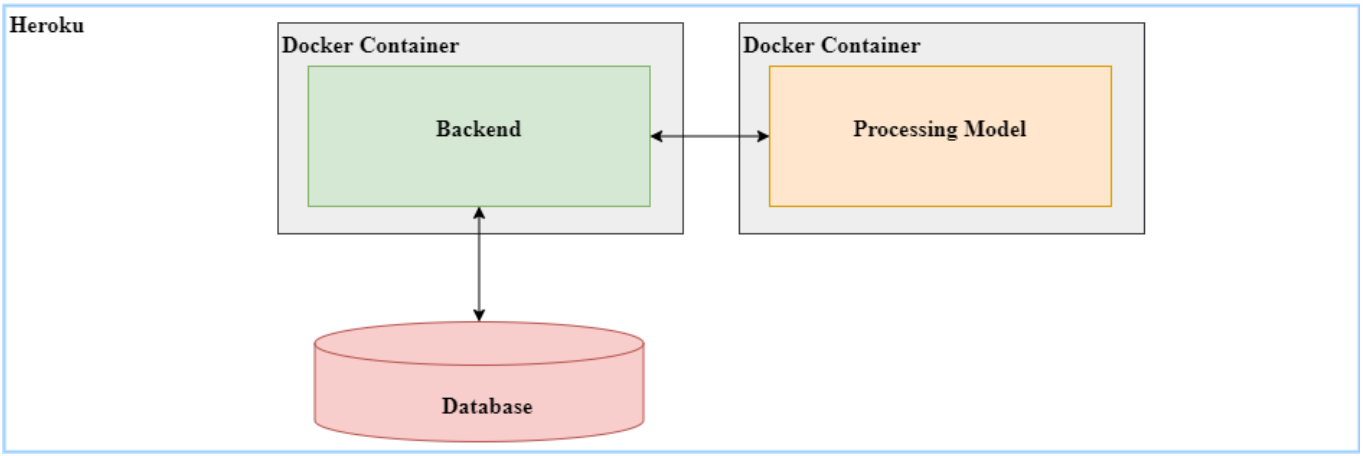

Figure 5.2: Deployment diagram

5.1.3. Design of Frontend

To have a good design of the frontend part of this project we have to specify a full description of the many aspects that comprise a well-developed strategy is required. In the following paragraphs these goals are explained.

Firstly, we have to mention what is the purpose of the frontend part and for who we implement this part and also a short description should be provided. The major reason an interface is required for our project is to give users with a means to utilize our system fast and readily that does not require knowledge in the area or time spent understanding what is on the backend. Furthermore, the frontend component is highly significant in this project since, in addition to receiving the data required by the backend and the rest of the system, it is the support on which half of the message broker part is constructed. It indicates that the functionality requirements for the front end should be accomplished before the development phase of the other components is done.

Secondly, a short presentation of the outcomes needs to be presented where the task that need to be completed are explained. All conclusions are Graphical User Interface (GUI) related, and here are some. Adding a feature that allows the user to choose its avatar should be one of the results; it is critical that the user feels special and involved in this application in order to have a pleasant experience. Another fantastic outcome is to present the user with both a textual and graphic result depending on the character they have picked.

Thirdly, we have to take in consideration the risk that can happen when the above outcomes will be implemented. Because the author of the paper is the lone developer of this project, the risk analysis is simplified. The most crucial requirement is that the paper be delivered in its whole by a date given by the university where the thesis is created.

[2] `https://www.heroku.com/`

Finlay, Figure 5.3 depicts a visualization of the process roadmap in order to finish the design of this component and have a thorough understanding of it. This flowchart is intended to show each step used to produce the final outcome.

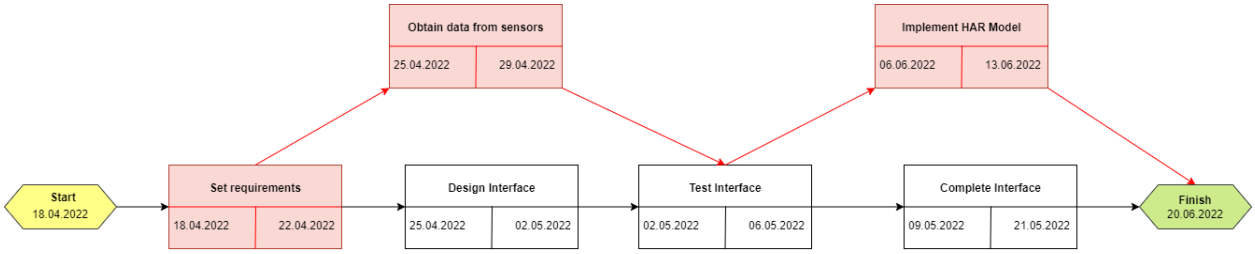

Figure 5.3: Frontend roadmap

5.1.4. Design of Backend

A detailed explanation of the numerous factors that compose a well-developed strategy is essential to have a decent design of the frontend section of this project. These objectives are detailed in the following paragraphs.

At first hand, the reason of the backend component is to use the data received from the frontend, when user requests a result, this component processes and analyzes the data, but only after it saves the values in a database system. Nothing can be returned to the user interface without this component. We may think of this component as the engine of a vehicle, where the car is the UI; a car without an engine would not move, therefore it does not fulfill its main purpose. This component is highly complex, offering the other implementation half of the broker of message where the flow of data is high and fast received. Even more, the communication with the data base is implemented in this part and the calls for the HAR model which produces the result that is transmitted to the fronted.

In addition, we will narrow down the project's outcomes. These are frequently more thorough than the original goal planning phase, and they cover the precise tasks this component has to cover. The Data Access Object (DAO) pattern needs to be implemented in order to have a fast and more abstract communication between the application and the database. A layered architecture needs to be followed alongside the implementation process to have an easy-to-understand system and to ease future application developments. Furthermore, certain parts must be made at the same time as those on the front end to ensure clean development.

At last, a timetable representation is shown in Figure 5.4 in order to complete the design of this component and have a thorough understanding of it This flowchart is meant to demonstrate each step that was taken to get the ultimate result.

5.1.5. Design of Message Broker

This component enables asynchronous communication between apps by routing messages through a queue. When the destination program is busy or not connected, the message queue offers interim storage between the sender and the receiver, allowing the sender to continue working uninterrupted. The following two paragraphs will describe

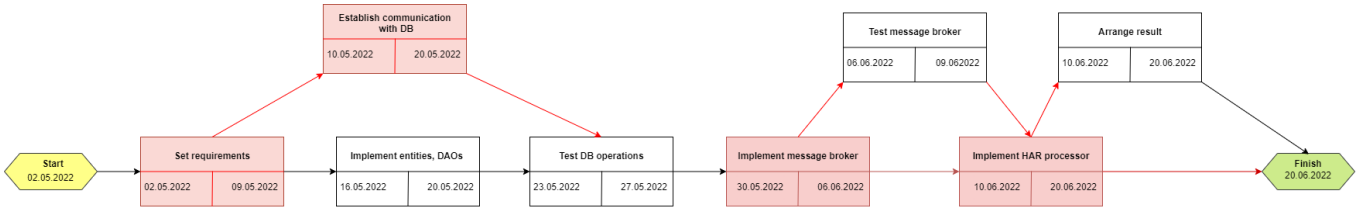

Figure 5.4: Backend roadmap

how each part of this component is developed. The first will describe how the client sends messages to the queue to which it is assigned, and how the server receives these messages through the same queue. The roles will be swapped in the second part, and a new queue will be involved.

In order to implement a posting instrument on the client side of the application we had to create a new thread that runs in parallel with the main one. The new thread is created when the user presses the lunch button on the screen, this new thread handles the data being published in the right queue. The data from the sensors is organized into JSON objects and then posted to the queue.

5.1.6. Design of HAR Detection Module

Before we start presenting how the model was design a briefly explain will be provided on how the data was preprocessed. First of all, the data was collected from different files and merged in one. Then the unnecessary data was drop from the table, using only the values from the wanted sensors and the time stamp of each entry plus the label. Then, the null values from the file ware replaced with the average of all the values of the column where the null value was identified. After that, we split the data int-o two sets, one contains the values from the sensors and the other containing the labels for each row. The following step was taken in order to have a more powerful algorithm, we encoded the label with numerical values. Next, the two sets mentioned above ware each split into two sets, train set and test set, very common step in machine learning algorithms. For this, we implemented a function, in which the split size is 80

An important observation in this model is that the output values from smartwatch sensor are continues and dependent on their time stamps, so we cannot predict anything by observing only a single point in the database because each point is dependent on the previous values so the prediction must be done observing in a fixed window. This fixed window is used to get values from the dataset, by selecting only the size of window number of data, this input will be feed to the model and it will predict. So, in the following we will define the size of the window, which is one hundred dataset rows and also the number of features, but for this to work we also need to convert the data set into a time series set.

The model's architecture is as follows:

- Layer 1 is an LSTM (Long Short-Term Memory) layer that learns from a series of 100 points at each timestamp and returns the sequence mapping. LSTM is a variation of a recurrent neural network which can store sequence of values over time. The input of this neural network will be two-dimensional input one is a group of 100 values from the set with values from the sensors and the other is 100 values from the label set, related with the first group. We must use here a function to avoid overfitting.

- Layer 2: The flatten layer is used to turn the 2-dimensional (number of timestamps, number of features) output from the previous LSTM layer into a 1-dimensional vector. This layer is only used to prepare the input for the next layer.
- Layer 3: The classifier component begins with this layer, which is a Dense layer that accepts flattened output from the previous layer and passes it to next layer. It is a feed-forward layer, simply defined, feed forward involves presenting future-oriented alternatives or solutions rather than delivering positive or negative feedback. This layer uses the same regularization function as the first layer but with a different name assigned to it. This layer will take as input values form the flatten layer and will behave as a classifier for the model.
- Layer 4: This is the SoftMax layer, which takes input from Layer 3 and predicts the likelihood of each activity. It will get the probability for each label using the input from the previous layer.

5.1.7. Design of Database

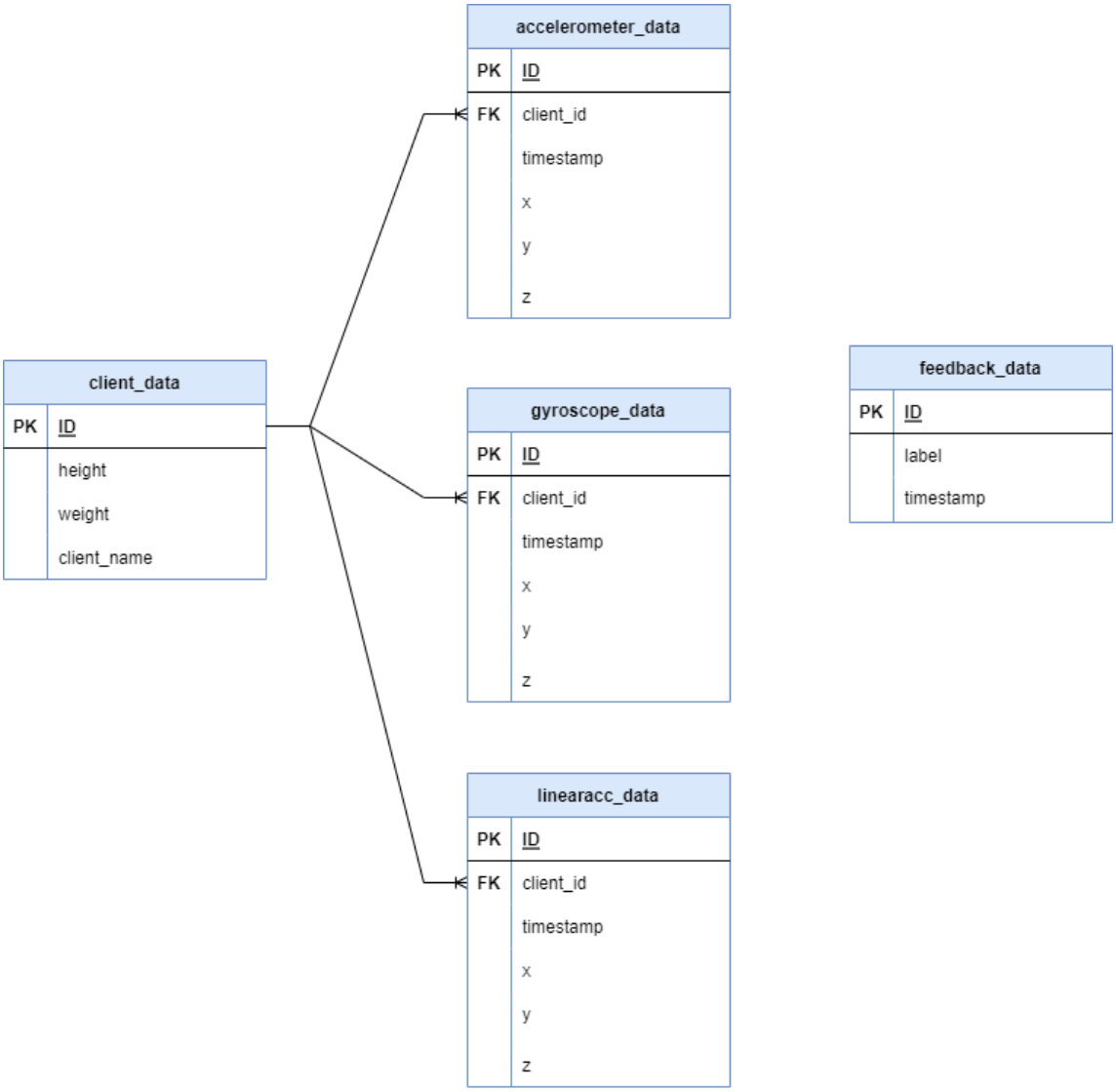

Figure 5.5: Database architecture

The storage component of the system is described thoroughly below, along with the relationships and tables that constitute it. We can see the database architecture drawing, in Figure 5.5 here are four tables in total, three of them are for storing information generated by the smartwatch sensor and the other table is used for storing the result given by the HAR Module. A brief summary of each table will be provided below:

- **client_data**
  This table, is storing data about each user of the project, each row represents a unique user, for the sake of fulfilling this thesis purpose only the main details that are influencing the predication will be stored. The data from this table is gathered at each request made by the user, in case the user already did previous request the data will not be changed only if the user changes it on its profile. When values are entered or changed in the database, they are encrypted to ensure that the user's personal information is not made public in the case of a malicious attack. As we can see in Figure 5.5, this table is only composed of three columns: height, which represents the height of the user wearing the smartwatch, weight, being the same as previous and client_name which is the name of the user, this column will be unique, meaning that each value is present only once in the table. Also, we can observe that, there are three relationships with three other different tables, each relationship has one-to-many type.
- **accelerometer_data**
  This table, is storing each change detected by the accelerometer, when any ax changes its value a new raw is stored in the table. The table is composed out of five columns, the last three columns represent the values for each axis used by the accelerometer. The first column is used to record the id of the user requesting the system; this column represents the previously specified one-to-many connection. We picked this method since remembering the user's static data with each change in axis values is pointless. In addition, this column allows us to extract the values sent to the classification module by a single command. The second column is used to send the same set change in all three sensors, for that this table's column alongside with the following two tables that are presented below, must have the same values in order to be sent to the model of prediction.
- **gyroscope_data**
  This table is similar with accelerometer_data, the difference is that it stores values recorded by the gyroscope. Similarly, the second column is utilized to synchronize the three tables.
- **linearacc_data**
  The usefulness is the same as in the previous two tables; the only difference is that it captures changes in data recorded by the sensor that deals with linear acceleration. In addition to the fact that the second column must be similar to all three tables, they must have the same customer ID. Only then can they be sent for processing.
- **feedback_data**
  This table is storing the results as a string, provided by the HAR Detection Module in the column named label. This table also have a column called timestamp where is stored the finishing time of each label. The prediction mechanism requires one hundred inputs, each input being made up of a set of 3 values from each sensor which also has a table mentioned above. The user sends over a thousand data sets

to a single request, a data set is made up of nine values that represent all the axes of each sensor, and for every one hundred entries, sed will produce a row in this table. In order to be able to differentiate and analyze these feedbacks correctly, we need to check which is the first and last row id, so for this feature we will use the timestamp column, and compare it with the first and last row recorded in accelerometer_data. All the rows in feedback_data that are between this interval will be selected and processed to give the user the best result.

5.1.8. Sequence Diagrams

5.2. Implementation

In this part of this chapter, we will present the implementation of each part of each part of this project. Each element that makes up the project structure will be presented in detail, as well as each method of implementing all the frameworks specified in the previous chapter. This part is important because it will contribute to a cursive understanding of what is being implemented in this project. At the same time, this part will contribute to a future development of the project with additional functions or more advanced technologies. In the following the explanation will be structured in two parts, in the first part it will be explained how the client part was developed, presenting each part of it in detail. The second part is intended as a server part explanation, this part includes the main server as well as the data processor that helps in classification. With these explained, a well-made blueprint will be defined that will allow anyone who has knowledge in the field of programming to remake this project.

5.2.1. Client side

The implementation of the client side was done in Android Studio, because this IDE allows us to develop applications for mobile devices such as phones and smart watches. This IDE not only allows us to implement the backend for the application to work but also allows the development of interfaces with which the user can interact and use them to use the application. Once the application is in working order, it is very easy for it to run on any smart device that has a sufficiently advanced version of Android to allow it to run. There are two ways in which the application that was created on a computer can be run on the phone, the first is to connect the phone to the computer with a USB cable and when you run the application starts to install on that phone, and the second option to send the folder with the extension .apk which contains all the files necessary for the correct installation of the application.

Therefore, in the following there will be a division into two parts, one for the backend and another for the application frontend developed in Android Studio.

Backend

Because in the latest versions of Android Studio we can develop applications in the Java language and not in Kotlin, the implementation of this part was designed in an OOP style written in Java. Therefore, the Backend part is presented by detailing each class present below:

- **MainActivity**
  This is the class that represents the first page of the application, when the user opens the application, this will be the code that represents the first page that he will view. When the activity is created for the first time, it is called the *OnCreate()* function where all the static variables of this activity are loaded, such as buttons, or lsite, mainly this function manages all the visual elements of the application as well as the maneuvering elements. This activity has as parameters two buttons that are used to navigate to different pages, one that takes us to the graphics selection page and the other to the main attraction of the application, the component that classifies the user's movements. The two mentioned buttons are initialized by assigning them with the graphic part of a button through the id defined on the Frontend side, in the function presented above. The first button is called *goToHistory*, it has the id on the graphic side *goToHARActivity*, after the connection between these two elements has been made, it will be assigned a function to listen when the button is pressed and then execute a list of orders. This function is called *setOnClickListener* and it is intended to move to another page that has been defined in *AndroidManifest.xml*. The other button is called *goToHAR* which has the id on the graphics side of the *goToHARActivity* application which uses the same type of action as the button mentioned above only it will take us to the *HARFragment* page, which is very important for this project. When the client decides that this activity no longer meets his needs, he will navigate to another activity, and the system, depending on the seventh in the memory of the device running the application, will decide whether or not the *OnDestroy()* function is called. This function frees all memory that stores data related to this page

- **RabbitManager**
  This class is created to facilitate the work of the main activities of this project, this class has two threads that will be used to not block the main thread of the application running on devices, and on the chain these two parameters have two more that are used to handle connections to the server hosting the queues. The main purpose of this class is to establish a connection with the server on which the queues are hosted and to be able to both consume data from the server and publish data to it. This class defines two important functions to create a communication bridge between the device running the client application and the main server. In addition, there is a parameter that is used to be used as a variable to communicate to activities that use this class. In the following we will explain the two functions that have been specified above. The first function is used to send data to the server through the queue, in other words this function allows the client to publish the data in a queue that the server listens to and will consume that data. The first event executed by this function is to start a thread, through the *thread.run()* function, and then as long as this thread lives, the following events will be executed. A new Factory instance will be created, which will be assigned the queue URL created on CloudAMQP, for an uninterrupted life-cycle. Once the URL has been set, a connection to this queue is empty, and in order to send a data stream, we need to create a channel through which we can do this. Once these things have been successfully completed, the next step can begin, and this is to enter an infinite block in which the client will send data constantly. It is not necessary to treat this case because this infinite loop will not block the execution of the main thread of the application, so therefore the client will not feel any interruption, another part that

we do not have to worry about is to close this thread, because when the activity that needs this function will be closed, the afferent thread, with all its children, will be finished. After the necessary instances will be created to establish the connections between the client and the server in a queue, a new infinite loop will start from which this thread will not come out until the thread ends. In this loop a function is called that returns the data to be published in the queue, this function will be modified according to the page errors on which it is used.The second important function in this class is subscribe, similar to the previous one it uses a separate thread than the main one of the application. A big difference is that this function has as a parameter a handle through which the message consumed from the queue is transmitted outside this thread. compared to the first is what happens in the second loop. Inside this loop is defined a variable that receives as value a lambda function, in this lambda function a message is received from the queue, this message was published in the queue on the server side. In this function, the case must be treated if the thread is blocked, because this is the only way to stop listening if the queue contains data or not. This page does not have a button or a graphical implementation that can exemplify what is happening in the background. But this chalice is used in the following activities which will be explained and presented below.

- **HARFragment**
  This activity is a very complex one, this activity combines both the specific part of Android and the parts explained above in class for configuring the queue. The parameters of these activities are two variables that will define the connections for the queues that RabbitMQ needs in order to create a communication channel. Moreover, there are 3 lists of floats for each axis of each sensor, *ax, ay, az* represent the lists in which the values recorded by the accelerometer on the 3 specific axes will be stored, then *gx, gy, gz* are the lists in which will retain every change of state of the gyroscope in the same way, on the 3 axes, and finally, *lx, ly, lz* are the lists where the changes recorded by the sensor that deals with the detection of differences in the linear acceleration of the device are recorded. Another list is declared as a parameter that stores each prediction made by the model classifier, in this list is added continuously until the user decides to press the button that interrupts communication with the server, and then based on this list will be done an analysis with which it will be possible to determine which activity is predominantly performed by the user. Moreover, there is another list, which stores the data that will be sent to the server to be classified, this list will always have a length of 100, more precisely 100 records that contain a value for each axis of the three sensors. This list is carefully handled because many times some sensors may be faster than others, and we want the values to be at the same time on all sensors. There is another parameter that is the representative of the class presented above to be able to use the queue configuration functions. Three parameters specific to applications on mobile devices are the accelerometer sensor, the gyroscope sensor and the linear acceleration sensor, these 3 sensors are the main source of data flow, these three sensors produce data which are then classified and which make up a result that the customer she wants to see him. These three parameters need another one to take care of them, this is the sensor manager, which acts as a super class for each sensor explained above. Finally, we have 2 more parameters which are the threads created by the functions for RabbitMQ, we need to retain these

two threads globally because we need to be able to stop them. In the following we will explain how the implementation was done in this activity, we start with the explanation of the events that happen in the *onCreate()* function in which the static elements of this are initialized. First, we set the View to be able to connect to the graphical part of this activity, then we initialize the 9 lists in which we will save the changes of the 3 sensors used for prediction, here all these lists will be initialized as empty instances of *ArrayList( )*. Then, we initialize with the same instance the lists in which the 100 records that will be sent to the server will be received, plus the list in which the predictions are added. A very important step is to join the *textViews* in which the results will be displayed, because we have 2 such entities, one to display in real time the title of activities done by the user and one for the general activity done by it. Then, the *SensorManager* is initialized together with sensors that can be obtained from this manager. In the following, we combine the graphical parts of the two buttons, one to start publishing data to the server and the other to stop the whole communication process and to stop the threads, in addition, we do the same for the button that returns us to main menu. The last event that takes place in this function is to initialize the *handle* for handling the results received from the server through the queue. In order for everything to work correctly, this function also uses the *RabbitManager* function to consume data. As a direct result, the related thread will be initialized and pronounced.The next important function to be presented is that of *onSensorChange()*, this function has as a parameter a *SensorEvent*, based on which we can determine which sensor on our device has created a value change event. As we use 3 sensors, we will have 3 separate conditions that check if the event was created by one of our sensors. If so, this change will be added to the above list. The next part is essential because it will explain the *onClick()* function which defines what each button does when the user presses them. We will start by presenting the button that takes us back to the previous page, it has been mentioned before, and its method of operation does not differ in this activity. The button that starts transmitting data to the server is called ”START ACTIVITY” and it has the role of starting to post the data in the queue at the user's decision. In order to give the client an idea of what is happening in the background, the first message displayed in the textbox for real-time results is ”Activity recognition started”, then at the first message from the server this text will change with the previous result. Technically, when this button is pressed the publishing function starts to run, there is still a change from the one in the class intended for it, a function that prepares the data received from the sensors so that it can be sent correctly. This function is called *sendDataToQueue()* and aims to ensure the integrity of the data to be published. The first line of code ensures that the list to be returned to you is empty, if it is not empty then it becomes empty, then a new auxiliary list is created. The next step is to go through the lists that record all sensor changes through a series of checks, in other words all 9 lists must have 100 values or more at the same time, otherwise this function will return a empty list. If the condition is met successfully, then the values in the following form are added to the auxiliary list, the first 100 values in the list for the X-axis of the accelerometer, then 100 values for the Y-axis, and finally for the Z-axis, with a total of 300 values for the accelerometer only, and then another 300 values for the gyroscope and another 300 for linear acceleration. A vital step in this function is that after the auxiliary list is loaded with values, the sensor lists must be emptied.

Each time a message is consumed on the client side it represents an action done by the user, therefore it must be saved in the corresponding list, in order to later calculate the final result which is the most frequent activity performed by pressing the start button and until the stop button is pressed. The second important button appears to the customer with the text "STOP" a very clear text for its purpose. Once this button is pressed, the data consumption and data publishing threads all die, and the final result is calculated using an auxiliary function called *mostFreq()*. This function works as a string of occurrences for the values in the list of results read from the queue. The value with the most appearances in this list will be considered the final result that will be displayed on the page viewed by the client.

- **HistoryManager**
  This activation is used to introduce the user to the presentation of the graphs that represent the number of smoking activities such as: smoking standing, smoking sitting down, smoking while walking and smoking in a group of people, these graphs will be presented later in this work. The classic functions of an android application page are also present here, but different from the main page, here are several buttons that have the same purpose, to navigate to different pages. The first button presented is *goToLast7days* which takes us to the first page with graphics, this button is linked to its graphic part by the id with the name *goToLast7Days*, it will lead us to a chart that shows the last 7 days from the current moment plus the number of cigarettes smoked. The second button is *goToLastMonth* which has the same name as the graphic link, for an easy follow-up of the program, it takes us to the second graph with predefined interval, thus displaying all smoking activities in the last month, more precisely in the last 30 days. when you press the button. The third button is defined for the same purpose as the previous ones, and the implementation is done similarly. The difference is that it will display 12 entries in a table representing each month of the year together with the number of activities registered as smoking. The last button does not look like the three but the principle for which it was implemented is the same, that of navigating to a page, more precisely to the page explained above, *MainActivity*, therefore this button is used for the purpose of Back known by everyone who had contact with any kind of digital device.Therefore, this page aims to be a side menu to give the user a variant of predefined graphics freeing him from the burden of selecting the right range, especially since the clock display is very small.

- **Last7Days**
  This activity is a complex one because it performs many functionalities at the same time. Parameters are small because it does not need too much data to be saved, because in this activity which is intended graphical representations of the user's history. In the *OnCreate()* function it is sent from the start, fixed when the user sends a predefined message to the server through the queue, all the configuration maneuvers of the queue are done as before, with the help of the *RabbitManager* class. This message contains the current date, plus the number of entries in the desired chart, which is 7. Because this chart is a bar chart, it will have 7 bars that represent the last 7 days from the current moment, and there will be values on the y axis. of whole numbers representing the number of cigarettes smoked. After the message has been published, a response is expected from the server, so following the function of consuming data from the queue is started immediately after the one

to be published has been started. Once these functions have registered the sending and receiving of the message can be stopped because no more requests will be made through them. Just as there is a button that returns us to the previous page, it is implemented here as well. Once received the values from the queue, 7 entries are created in the tables, on the X axis it is loaded statically with the previous 7 days and on the Y axis the values received from the queue are loaded.

- **LastMonth**
  The implementation of this activity is similar to the previous one, all parameters and all functions are similar. The differences are that in this class we use instead of 7 graph entries only 4, which represents each week of the previous 4. From the day the user makes the request up to 28 days in the past. I chose this implementation because it is enough for the user to see how many times he has performed the smoking movement per week, for a better understanding of the differences and an easier calculation of the total.

- **LastYear**
  This activity's implementation is similar to the previous one; all parameters and functions are the same. The difference is because in this class, instead of seven graph elements, we will utilize twelve, one for each month of the last year before the current date. From the time the user submits the request until 365 days ago. This solution was chosen because it is sufficient for the user to observe how many times, he has performed the smoking movement every month in order to better grasp the differences and calculate the total.

Frontend

This part is focused on the visual elements within the client application, here we can see the implementation of several elements that together make up the graphical interface of the user. Through this part we managed to create an application that is pleasing to the eye and easy to follow and use. You must keep in mind that all these files that will be explained in this part will be present on all the devices that the application will use, so we must be careful with the elements we use because we do not want to overload the device memory. We combined several knowledge acquired during the course in college, in order to obtain a product that is loved by all users. In the following we will focus our explanations on the file with resources from this project, because here are all the elements related to the graphic part of the application and we will present each package along with their content.

The first folder is called *Drawable* and it contains in the main image which were used in the pages that make up the application. There are two types of backgrounds, one to suggest to the user that this is in a menu page where nothing more happens than the presentation of buttons, and the second is in order not to load the frame too much because the user's attention it must be focused on the actions that are taking place, such as graphics or texts. There are also images that have been used to replace the buttons, such as the back button that uses an arrow pointing to the left as an image to suggest why.

The following folder is intended for XML files that are used to create the appearance of each page that the user can view, each file of this type corresponds to a few pages and that's it. A file of this type contains a *viewholder* for each graphic element, therefore each graphic element on these pages must have a unique name in the project, by graphic

elements we refer to buttons, texts and so on. At the same time, it defines the type of placement on the page of these graphic objects. In the following we present each XML file on the screen for a clearer understanding of the application.

- **activity_main**
  Here are four important elements, the first is a *ViewGroup* of type *ConstraintLayout* that allows the graphic elements of these pages to be placed on pages very easily and flexibly because they do not have to depend on anything, and at the same time can be resized. just as simple, last but not least, this element defines and imagines the source of the background. The next element is an *ImageView*, which allows you to place an image within the page, in our case, this image is the logo of the faculty in which this bachelor's thesis is developed. The next two items are two buttons that are intended to redirect to other pages, this has a specific name.
- **activity_har**
  In this page we have only 2 buttons that are used to transfer to other XML files plus a *ViewGroup* of type *ConstraintLayout* which is used to be able to place these buttons in different places with ease and mobility. We also have a pleasing background and an *ImageView* to place the logo of the institution.
- **activity_history**
  This XML file is very similar to the previous one, because it also contains only buttons that are used to redirect the user to a new page where he can view graphics. There are 3 buttons that have the text displayed on them, the time interval that the user can choose. The fourth button sends the user to the main page. This page has the same background as the above page.
- **activity_graph**
  In this file there are some new elements that I have not explained until now, these are taken from an unofficial Android source, more precisely it is a project on the throat. The special element is the *BarChart* where the data is displayed. In addition, there is a button that takes us back to the home page.

5.2.2. Server side

This part of the project was implemented in Java using the Spring framework to be able to handle all the data coming from the client, plus the processor that deals with data classification. The server handles the connection to the project database and all communications with the other components. A layered architecture was used to be able to organize the components for the same purpose in layers, and each layer has a specific role in the project. All layers are explained below, along with implementation details for each component within them.

Entities

In this layer are the classes specific to the concept of Object-Oriented Programming (OOP), these classes were also used to define the tables in the database with the help of the *Lombok* library, which allows annotations such as *@Data, @Entity*, etc. The first class is the one dedicated to clients and their personal data, where the following attributes have been defined: name which is of type String, weight and height which are of type Integer, and in order to have a conformity in the database, an id of Long type. The next three

classes will be applied together because they have the same purpose and attributes, and these are *AccelerometerData, GyroscopeData, and Linear AccelerationData*, all of which use the same library as the previous one for database reconciliation. These classes were used to store the values of specific sensors on the device running the client application. The attributes of these classes are the 3 axes of orientation and the time at which they were recorded. The last and probably the most important class in this package is the Consumer class, which aims to compose with the rest of the components that make up this project. This class has as parameters, services for all classes related to the database. Within these classes is defined a function that is repeated once every 1000 milliseconds, so that it can be executed as long as the server is running. A new Factory instance will be built and assigned the queue URL generated on CloudAMQP. Once the URL is specified, the connection to this queue is empty, and we need to construct a channel through which to transfer a data stream. Once these steps have been done successfully, the next step is to ingest the data. Once the data is consumed, the connection to the data classification processor made in Python is verified, this connection is established by local Sockets on the port 2004, if the connection is stable the data is sent to it.

Repositories

This layer creates the logic that makes the connection between the database and the application, so we can handle the data in the database with the help of predefined functions. We have created a Repository for each class that is related to sensors and their data plus another for the class that deals with customer data. We did not implement new repositories, but used JpaRepository which is an API that handles all CRUD operations which are the 4 functions used in an application that has persistent storage behaviors, these are: create, read, update and delete.

Services

This layer deals with creating a link between the previous layer and any individual who needs to work with the database. Here is a Service for each Repository created in the previous layer. In order to implement a level of abstraction, because we do not want the access to the database to be so simple, we decided to create an interface for each implementation of each Service. We did not develop these classes because we used the Spring framework that offers *@Service*, where the most important functions for their need are.

5.2.3. Model Training

This part of the project is no longer used because after a run it produces a file of type h5 which can be used later to classify different types of entries. The first step is to concatenate the data from the 10 subjects from whom the data was collected from the sensors and these data were labeled in order to have training and test data. Since data was recorded from all the existing sensors on the phone, we decided to get rid of the data we don't need, in order to reduce the training time and ventilate the computer's memory. After I managed to complete the previous steps, I noticed that some rows have values that are null, to get rid of them I used a function in the pandas library that replaces all those values with a certain value. Once we got rid of the null values, we divided the data set into 2, a set containing all the data from the sensors and a set containing their label.

These two sets were both split into two sets, one for training and one for testing, and then these sets were transformed into Timeseries, with a function in the keras library. Then, using Sequential from keras, which is a linear stack of layers, we created a model. This model has multiple layers, the first is a long short-term memory layer, which takes as input 100 rows from the training data set and uses a regularization function to prevent the overfitting. The next layer is used to convert the two-dimensional output from previous layer to a 1-dimensional layer which is used as input for next layer. This next layer is a regular feed-forward layer, which will act as the classifier for our model. The last layer is used to throw out probabilities for each class dependent on the input it gets from the previous layer. Then we will start the training, but not before we compile and prepare callbacks. The last step here is to deploy our model so we can use it. In the following we an image that visually supports the concepts explained above, also we included the epochs to show our final accuracy of our model.

```
[ ] from keras.preprocessing.sequence import TimeseriesGenerator

    n_time_steps = 100
    n_features = 9

    train_gen = TimeseriesGenerator(X_train.to_numpy(), y_train, length=n_time_steps, batch_size=1024)
    test_gen = TimeseriesGenerator(X_test.to_numpy(), y_test, length=n_time_steps, batch_size=1024)
```

```
[ ] from keras.models import Sequential
    from keras.layers import Dense,Flatten, LSTM
    from keras.regularizers import l2
    from tensorflow.keras.optimizers import Adam
```

```
[ ] model=Sequential()
    model.add(LSTM(32, return_sequences=True, input_shape = (n_time_steps, n_features),
                   kernel_regularizer = l2(0.000001), bias_regularizer = l2(0.000001), name='lstm_1'))
    model.add(Flatten(name='flatten'))
    model.add(Dense(64, activation='relu',kernel_regularizer = l2(0.000001), bias_regularizer = l2(0.000001), name='dense_1' ))
    model.add(Dense(10, activation='softmax',
                    kernel_regularizer = l2(0.000001), bias_regularizer = l2(0.000001), name='output'))
    model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 100, 32)           5376

 flatten (Flatten)           (None, 3200)              0

 dense_1 (Dense)             (None, 64)                204864

 output (Dense)              (None, 10)                650

=================================================================
Total params: 210,890
Trainable params: 210,890
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.6: Model Training

5.2.4. HAR Processor

This component of the application receives the clock data through the server and sends to the clock the label resulting from the analysis of the clock data, through the same type of technology as before, queues hosted online by CloudAMQP. The first step in this part is to configure the Socket corresponding to the server, ie the one on the 2004 port of the local host. Then the next step is to configure the connection and channel

through which data will be sent to the client device. In an infinite loop we collect data received from the server through the socket, when the data has reached a number of 100, because this constrains our model, they are sent as input to the mold, which returns a string of floats. This string represents the probability that the input we entered belongs to one of its classes. In order to send feedback that the user can read and understand very easily, we need to define a series of Strings that contain all the labels of the model, which we order alphabetically. Then, the position in the string with separates that has the highest value is selected from the string with strings and sent through the client's queue.

Chapter 6. Testing and Validation

This chapter will present the testing done to ensure the proper functioning of the application, at the same time, will explain the presentation methods. The manual testing part of the application was done for the whole application except the smart component. The accuracy of this component was tested using 20% of the total data collected from the 10 participants.

6.1. Manual Testing

This manual testing of the system was done by the same person who developed the application, therefore there is a deep knowledge of the system. For a successful test, several sets of steps have been made which are presented below. These sets were made to be able to compare the results with someone who has no knowledge in this field or in this project. These sets are presented as tables with 2 columns, one that defines the description of the step to be followed on the graphical interface on the device and the second is for the effect that must take place immediately after the step.

Step	Result
Open application	On the device screen, the home page of the application is displayed, with 2 buttons on it.
Press ”Activity Recognition” Button	The application will redirect to the page for human activity recognition.
Press ”START” button	Next to the ”Status” label should appear ”start recording...” and in a short time an activity name will change that text.
Press ”STOP” button	Next to the ”Status” label the text will change to ”no recording” and next to ”Result” label the main activity is displayed.

Table 6.1: User want to use activity prediction feature

The table above represents the set of steps that must be followed in order to have a correct execution of the most important feature of the application. We have specified all the changes that must appear on the graphical interface of the electronic clock.

Below we will present the set of steps and their effects to display the graphics page of the application, which displays the history of smoking activities from a certain period in the past.

Step	Result
Open application	The application's home page is presented on the device screen, along with two buttons.
Press "HISTORY" Button	The application will redirect to the page where the user select the interval, by pressing on one of the three buttons "Last 7 days", "Last Month" and "Last Year".
Press one button	The user is sent to a page where a bar chart is shown, the number of the bars depends on the button pressed.
Press "back" button	Redirect back to the page where the user can select the interval

Table 6.2: User want to use activity prediction feature

6.2. Experimental results

During the training of the model, 5 epochs were used, which led to an accuracy of 0.81, which is quite high in this field of work. Below are two graphs that represent the accuracy of the model during training and during validation over 5 epochs and data lost in the same parameters. Two graphs that visually express these works and that exemplify on a graph how the two mentioned values vary are found in image 6.1. These graphs were produced from training sessions which is also shown in image 6.2.

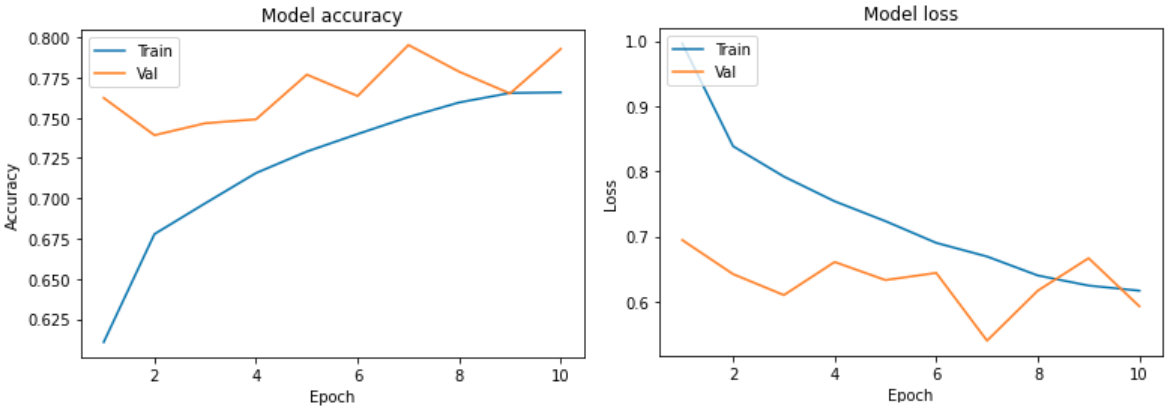

Figure 6.1: Data Lose Accuracy graphic

```
Epoch 1/10
1684/1684 [==============================] - 18s 4ms/step - loss: 0.9963 - accuracy: 0.6106 - val_loss: 0.6943 - val_accuracy: 0.7624
Epoch 2/10
1684/1684 [==============================] - 6s 4ms/step - loss: 0.8385 - accuracy: 0.6778 - val_loss: 0.6420 - val_accuracy: 0.7392
Epoch 3/10
1684/1684 [==============================] - 6s 4ms/step - loss: 0.7919 - accuracy: 0.6969 - val_loss: 0.6098 - val_accuracy: 0.7467
Epoch 4/10
1684/1684 [==============================] - 7s 4ms/step - loss: 0.7538 - accuracy: 0.7157 - val_loss: 0.6606 - val_accuracy: 0.7491
Epoch 5/10
1684/1684 [==============================] - 6s 4ms/step - loss: 0.7233 - accuracy: 0.7291 - val_loss: 0.6328 - val_accuracy: 0.7769
Epoch 6/10
1684/1684 [==============================] - 6s 4ms/step - loss: 0.6900 - accuracy: 0.7400 - val_loss: 0.6438 - val_accuracy: 0.7636
Epoch 7/10
1684/1684 [==============================] - 7s 4ms/step - loss: 0.6690 - accuracy: 0.7505 - val_loss: 0.5397 - val_accuracy: 0.7953
Epoch 8/10
1684/1684 [==============================] - 7s 4ms/step - loss: 0.6399 - accuracy: 0.7596 - val_loss: 0.6166 - val_accuracy: 0.7787
Epoch 9/10
1684/1684 [==============================] - 7s 4ms/step - loss: 0.6244 - accuracy: 0.7654 - val_loss: 0.6664 - val_accuracy: 0.7651
Epoch 10/10
1684/1684 [==============================] - 6s 4ms/step - loss: 0.6165 - accuracy: 0.7659 - val_loss: 0.5924 - val_accuracy: 0.7928
```

Figure 6.2: Training session

6.3. User validation

In order to verify the correct operation of the system, it is necessary to test the application outside the parameters with which it was developed. Therefore, the application was installed on a different device than the one on which it was created. Then, it was tested by five subjects who did not influence the implementation at all and who have no knowledge about the functionality of the application. The results were surprising, because most of the time the application correctly predicted, in real time, the physical activity done by each of the subjects.

Chapter 7. User's manual

This chapter aims to guide the customer to achieve the best possible navigation and use of the application. This guide will be done by explaining each page and each button with a visual aid to make it easy to follow.

7.1. Prerequisites

Because this application is dedicated to customers who use a mobile device, such as a smartphone or smartwatch, the application must be easy and secure to install. Therefore, the required file, where the installation kit will be found, can be obtained from on Google Drive[1] at the link: access link to google drive, the client do not need an account to download it. Once this file is downloaded, it requires only one touch and then a confirmation and the installation will begin.

Because the server side is in a different location than the client application, no extra installation is required. The only requirement that the client must meet is to have a poor internet connection.The only requirement that the client must meet is to have a poor internet connection, and also a device that runs at least Android version 8.0.

If you want to install the server for it to work on your own local machine, it is necessary to install a series of applications and libraries for it to work at normal rates. Which are:

- Database: PostgreSQL
- Development Kit: Python 3.6
- Frameworks: Spring
- IDEs: PyCharm, IntelliJ IDEA

7.2. User Interaction

In this part of this chapter, we will present each page individually with each of the features it has, plus an image to help guide the user and make it easier to understand.

7.2.1. Menu Page

This page is used to introduce the user to the application, practically this being the first contact with the application. I chose a pleasing background, along with 2 buttons that have a warm color and a large and intuitive text on them. This page will lead the user via the button to the two major features of the application, which will be presented in the following subchapters.

[1] `https://www.google.com/intl/ro/drive/`

Below we can see an image captured from this page, for the quality of the images all the screenshots were made from an emulator provided by Android Studio.

Figure 7.1: Menu Page

7.2.2. Human Activity Recognition Page

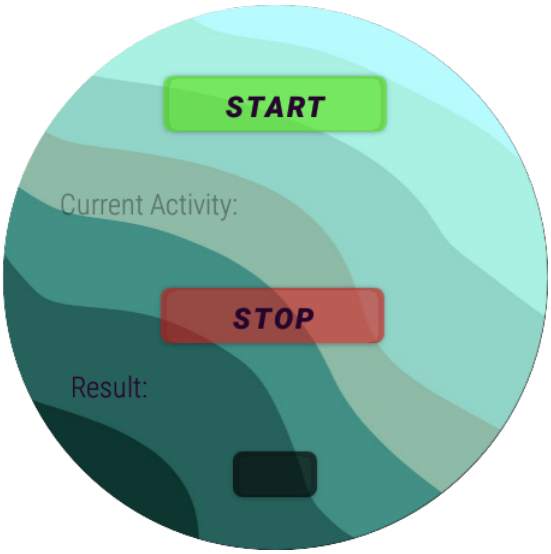

Figure 7.2: Human Activity Recognition Page

In this page we can see according to figure 7.2 that there are 3 buttons and 2 labels. In this part we will focus on how to use what this page offers, the first step to see changes is to click on the green button with the text "START". Once this button is pressed, the text next to the "Current Activity" label will change, which will represent the activity done by the user. As well as the text on the red "STOP" button, the system will stop working and a text will appear next to the "Result" labels.

7.2.3. History Page

This page works similarly to the first one presented, it is used to provide the user with visual support for navigating to graphic pages. The 3 buttons visible in the 7.3 image are very intuitive in terms of its actions.

Figure 7.3: History Page

7.2.4. Graphs Pages

This page is very simple when it comes to communicating with the user, because he can only visually follow the graphic chosen on the previous page.

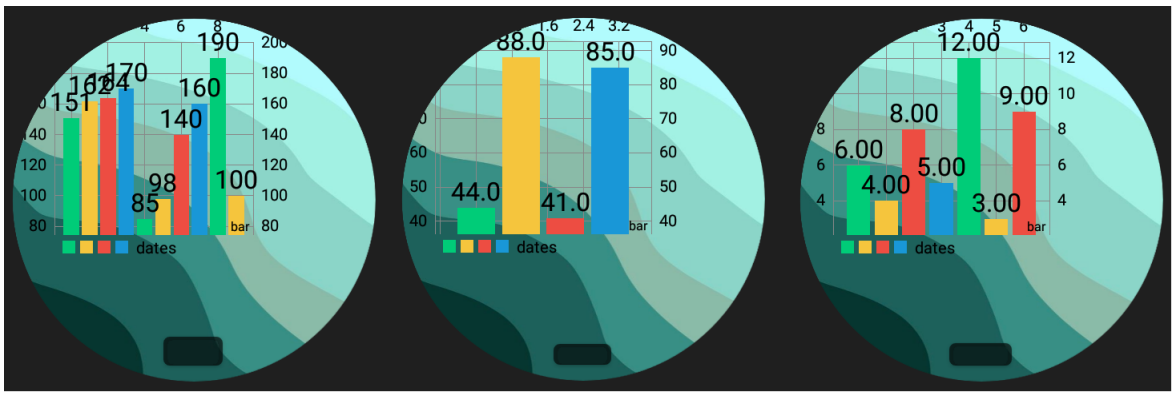

Figure 7.4: Graphs Pages

Chapter 8. Conclusions

In this chapter we will explain the conclusions we were left with after finishing this project, explaining the knowledge gained in the various fields that were researched and used in this project.

Considering that this project was a complete application that contains both the client side and the server side, we can say that we went through the process of creating a product that can be presented on the IT market in the world. During the development of this application I encountered all sorts of misunderstandings, bugs, incompatibilities and so on, but together with the coordinating teacher, I found ways to avoid this problem or even solve them. It was a difficult and difficult process, but one that led to a final product that we are proud of and through which we acquired very valuable knowledge.

For a clearer definition of the results obtained from this project, we will bring both technical and real details in the section below of this chapter. Moreover, in the final section, we will explain what are possible improvements that can be made and also the features that we want to implement.

8.1. Obtained results

The goal we started from was achieved, because at the beginning we wanted to develop a system that distinguishes smoking from other human activities, and especially from those that resemble smoking, such as drinking or eating. We managed to implement this system, which is largely accurate. In addition to this success, we have achieved the performance of differentiating types of smoking, by these types we mean smoking while walking, while sitting and even smoking in a group of people. At the same time, we managed to implement a part that represents the data that are significant for the topic approached in the paper.

Technically regarding the situation, we managed to combine several types of technologies, which worked very well together. We managed to establish a connection between many components that worked independently and make them a whole system. I managed to deepen the Android Studio, which I have never worked with before and which seemed to me an extraordinary tool for creating phone applications.

8.2. Future improvements

We must consider that this work was done in order to demonstrate how several different technologies work together and also to find the best method of classifying human activities.

Even if the application is in a good operating stage in which there are not many bugs and the graphical interface is pleasing to the eye, it is not an application that is

frequently used by the user or to attract new users. There are several reasons for this, such as the fact that it does not appear on any platform from which it is downloaded is one of the biggest problems.

Another improvement of the application would be that the plan selected for maintaining online queues should be one for money, which would allow more queues to allow more users to use applications at the same time.

The last feature that we thought about, and that we want to implement is one that leans more towards the part of warning or raising attention to smoking. We all know that smoking is harmful, and any reprimand from a smoker can help. So as a support for them we want to integrate in the application, a notification to be displayed, when the user is registered by the clock that he smokes.

Bibliography

[1] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *BioNanoScience*, vol. 3, no. 2, pp. 145–171, 2013.

[2] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[3] S. Chernbumroong, A. S. Atkins, and H. Yu, "Activity classification using a single wrist-worn accelerometer," in *2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA) Proceedings*. IEEE, 2011, pp. 1–6.

[4] M. H. M. Noor, Z. Salcic, I. Kevin, and K. Wang, "Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer," *Pervasive and Mobile Computing*, vol. 38, pp. 41–59, 2017.

[5] M. Shoaib, H. Scholten, P. J. Havinga, and O. D. Incel, "A hierarchical lazy smoking detection algorithm using smartwatch sensors," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, 2016, pp. 1–6.

[6] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors*, vol. 16, no. 4, p. 426, 2016.

[7] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, "Risq: Recognizing smoking gestures with inertial sensors on a wristband," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 149–161.

[8] W. Qi, H. Su, C. Yang, G. Ferrigno, E. De Momi, and A. Aliverti, "A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone," *Sensors*, vol. 19, no. 17, p. 3731, 2019.

[9] J. Justa, V. Šmídl, and A. Hamáček, "Fast ahrs filter for accelerometer, magnetometer, and gyroscope combination with separated sensor corrections," *Sensors*, vol. 20, no. 14, p. 3824, 2020.

[10] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *2016 IEEE International conference on pervasive computing and communication workshops (PerCom Workshops)*. IEEE, 2016, pp. 1–6.

[11] M. Shoaib, S. Bosch, H. Scholten, P. J. Havinga, and O. D. Incel, "Towards detection of bad habits by fusing smartphone and smartwatch sensors," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom Workshops)*. IEEE, 2015, pp. 591–596.

[12] S. Mekruksavanich and A. Jitpattanakul, "Smartwatch-based human activity recognition using hybrid lstm network," in *2020 IEEE SENSORS*. IEEE, 2020, pp. 1–4.

[13] R. V. Same, D. I. Feldman, N. Shah, S. S. Martin, M. Al Rifai, M. J. Blaha, G. Graham, and H. M. Ahmed, "Relationship between sedentary behavior and cardiovascular risk," *Current cardiology reports*, vol. 18, no. 1, pp. 1–7, 2016.

[14] A. Biswas, P. I. Oh, G. E. Faulkner, R. R. Bajaj, M. A. Silver, M. S. Mitchell, and D. A. Alter, "Sedentary time and its association with risk for disease incidence, mortality, and hospitalization in adults: a systematic review and meta-analysis," *Annals of internal medicine*, vol. 162, no. 2, pp. 123–132, 2015.

[15] A. Kirk, A.-M. Gibson, K. Laverty, D. Muggeridge, L. Kelly, and A. Hughes, "Patterns of sedentary behaviour in female office workers," *AIMS Public Health*, vol. 3, no. 3, p. 423, 2016.

[16] N. T. Hadgraft, C. L. Brakenridge, A. D. LaMontagne, B. S. Fjeldsoe, B. M. Lynch, D. W. Dunstan, N. Owen, G. N. Healy, and S. P. Lawler, "Feasibility and acceptability of reducing workplace sitting time: a qualitative study with australian office workers," *BMC Public Health*, vol. 16, no. 1, pp. 1–14, 2016.

[17] P. C. Dinas, Y. Koutedakis, and A. D. Flouris, "Effects of active and passive tobacco cigarette smoking on heart rate variability," *International journal of cardiology*, vol. 163, no. 2, pp. 109–115, 2013.

[18] D. Angrave, A. Charlwood, and M. Wooden, "Working time and cigarette smoking: evidence from australia and the united kingdom," *Social science & medicine*, vol. 112, pp. 72–79, 2014.

[19] Y.-S. Cho, H.-R. Kim, J.-P. Myong, and H. W. Kim, "Association between work conditions and smoking in south korea," *Safety and health at work*, vol. 4, no. 4, pp. 197–200, 2013.