License Project

Moldovan Horia-Andrei

March 2021

Contents

1 Introduction **2**
1.1 Project Context . . . 2
1.2 Thesis Structure . . . 3

2 Project Objectives and Requirements **3**
2.1 Main Objective . . . 3
2.2 Secondary Objectives . . . 4
2.3 Requirements . . . 4
2.3.1 Functional Requirements . . . 4
2.3.2 Non-Functional Requirements . . . 9
2.4 Metrics used for evaluation . . . 11

3 Bibliographic research **11**

4 Analysis and Theoretical Foundation **17**

5 Design and Implementation **27**
5.1 Design . . . 27
5.2 Formulas and mathematical models . . . 34
5.3 Data structures . . . 34
5.4 Algorithms . . . 34
5.5 Implementation . . . 34
5.5.1 Android Studio . . . 34
5.5.2 Visual Studio . . . 40
5.5.3 PyCharm . . . 41

6 Experiments and Validation **44**
6.1 Experimental Setup . . . 44
6.1.1 Data sources for data used in experiments (real or simulated) 44
6.1.2 Metrics used to evaluate the solution . . . 44
6.1.3 Experimental setup description . . . 44
6.1.4 Algorithms and parameter values used in experiments . . 44
6.1.5 Classical Algorithms/Methods used in comparisons 44
6.2 Experimental Results . . . 44

7 User Manual **44**
7.1 Installation Prerequisites 44
7.2 User Interaction 44

8 Conclusion **44**

1 Introduction

1.1 Project Context

Nowadays a lot of people struggle with confidence regarding their physical appearance. Either if we talk about obese people which are trying to lose weight or about skinny people which are trying to put on mass, they both have a common goal, namely to get closer to their ideal looks. Especially between teenagers, children are becoming more and more discriminated and getting laughed at, based on their physical appearance. Health also plays a big role in this domain, because the lack of physical exercise can cause severe health problems. It is no secret that in this technological era everything is at our footsteps and physical movement is no longer required in order to perform task, which in the past could not have been done by a few clicks. Moreover, it is not a coincidence that the gaming industry is doing better than never but the obesity rate is quickly increasing. So the real question is, if we rely that much on technology, and if it proves to be of such a big help is so many domains, why not use it into the fitness industry as well in order to help people make such an easy but also significant change in their life.
Regarding to the people which have already started their fitness journey, another very encountered problem is the apparition of plateaus, meaning that you get to a point where no matter how much you try, no more improvement can be seen. Because in this industry the main thing which drives you forward and provides motivation is the presence of real achievements and improvements, for such a thing to happen, it would mean the loss of all the interest. People which find themselves in this situation usually take one of the two following bad decisions. The first one is to stop training at all while the second one is to consider that the lack of supplements or of external help is the cause of your stagnation. The latter one may lead people to start using excessive supplements or even resort to illegal substances. Both of this paths may lead to serious health problems as well. In most of the cases, a plateau appears because the lack of knowledge or of implication, problems which can be easily fixed with a bunch of techniques.
For the people which desire to start or at least give it a try to fitness and bodybuilding, but also for the more experienced people which want to set goals and see real improvements in their strength and their physique, this application might be the answer they are searching for.

This application was designed with the scope of helping the user throughout his day to day workout routines. An easy way to do such a thing would be to capture most of the functionalities which can be computed by a machine and implement them into a compact software application in order to ease the job of the user as much as possible. By doing such a thing, the user can focus better on his exercises and worry less about other supplementary tasks. There are a lot of things which can hinder the focus of the user and this applications tries to take care of most of them. When working out, the user should have as main focus the exercises themselves if he want to see improvements. Moreover, there are a lot of days when motivation cannot be found and the mindset in this situations mostly is to skip the workout until a more satisfying time comes. Applications of this type should help the users in this kind of situations and make them realise that a small part of the work is performed by a computer so they can find a helping hand which tries to make them follow their goals no matter what. Another good approach would be to implement notifications in form of rewards and penalties when a workout is performed or not. Following, there are a lot of tasks which require the user to manually write or memorise certain things regarding to the daily workouts. An application which takes this burden away of the user and does this by itself, with only a few information provided by the user, could be also of big help. In the next subsection are described all the objectives which were implemented in order to ease the users tasks, having the scope of helping the user throughout his journey of becoming a better self.

1.2 Thesis Structure

2 Project Objectives and Requirements

2.1 Main Objective

The main objective of this project is to study, design and implement a system to track physical activities and provide various type of feedback and information regarding different exercises and performances by using a smartwatch. Studying the existing fitness applications and creating new functionalities or updating the already existing ones in order to help the user to have an experience as good as possible with the application falls into this chapter of main objectives as well. Designing an easy to use and understand graphical interface is of main priority as well, because a well implemented system without an accurate graphical view is of no much use. Thus meaning that a flawless design with easy to understand functionalities and accurate feedback is to be desired. On the other side, a good graphical interface has to be matched with an errorless and well thought implementation. Combining different frameworks and implementation techniques should be done in such a way that the output will bring no harm to the good functioning of the application. Following, the required connections to the database or to the internet must be designed in such a way that the speed and reliability of the application don't have to suffer. As mentioned above, in order to achieve such functionality, the existing approaches must be studied and among all the found resources, the ones with the highest performance rate and the ones which fit best into the design, should be chosen. Above that, with the help of different pieces of information, new techniques should be created in order to put this application above most of the existing ones.

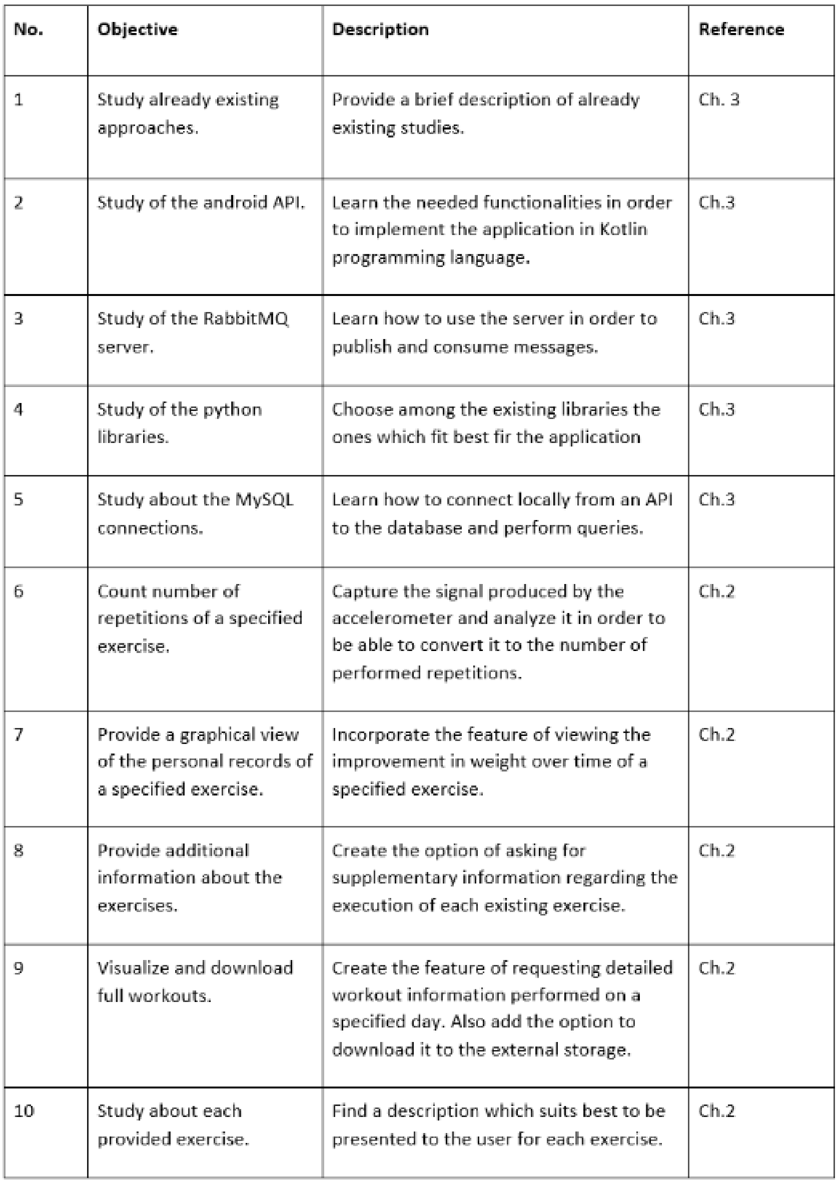
2.2 Secondary Objectives

2.3 Requirements

2.3.1 Functional Requirements

1. Counting Repetitions
As mentioned above, the main objective of the application is to help the user throughout his workout as much as possible. One of the best way of doing such a thing would be to successfully count the number of repetitions from the set of a certain exercise. Most of the time, especially when the exercise is more complex and requires intensified concentration, the focus put into the execution of the exercise can make you miscalculate the performed number of repetitions. For some people, counting throughout the execution of an exercise can help them to remain focused and to push another one or two repetitions at the end of the exercise. On the other side, the remaining category of people can encounter difficulties while trying to execute an exercise and to count the repetitions at the same time. Usually the power lifters fall into the first category, and the people more interested into a cross fit type of workout, or a workout which targets weight loss and muscle definition having as main concept high repetitions and low resting time, fall into the second category. The latest group of people can make use of this repetition counting feature which does all the counting work by itself, letting the user fully focus on the execution of the exercise. In order for the system to be able to count the number of repetitions, the user has to specify the start and the end of his performance. When pressing the button indicating the end of the execution, after a small period of time, the user will be provided with the number of performed repetitions detected by the system.

2.Retrieving Data
A lot of athletes, bodybuilders or just ordinary people tend to track their daily workouts in order to have a full written description about was has been done. Mainly, this task requires the user to write by hand, after performing each exercise, into a notebook, either a physical one or using a digital format, each detail about each exercise, like the name of the exercise, the weight used and the number of performed repetitions. This process would not just steal a lot of time but, in time, it could become really annoying to have to track the mentioned details after each set of each exercise. Moreover, if the physical format is used, the user has to always carry a notebook and a pen by himself in order to be able to write those details. While using a digital format, like the notebook of the phone, things could be eased a little while thinking about writing the data, but while it comes to reading and analysing the introduced information the problem remains the same. Concluding, even if we talk about introducing the data by physical



No.	Objective	Description	Reference
1	Study already existing approaches.	Provide a brief description of already existing studies.	Ch. 3
2	Study of the android API.	Learn the needed functionalities in order to implement the application in Kotlin programming language.	Ch.3
3	Study of the RabbitMQ server.	Learn how to use the server in order to publish and consume messages.	Ch.3
4	Study of the python libraries.	Choose among the existing libraries the ones which fit best fir the application	Ch.3
5	Study about the MySQL connections.	Learn how to connect locally from an API to the database and perform queries.	Ch.3
6	Count number of repetitions of a specified exercise.	Capture the signal produced by the accelerometer and analyze it in order to be able to convert it to the number of performed repetitions.	Ch.2
7	Provide a graphical view of the personal records of a specified exercise.	Incorporate the feature of viewing the improvement in weight over time of a specified exercise.	Ch.2
8	Provide additional information about the exercises.	Create the option of asking for supplementary information regarding the execution of each existing exercise.	Ch.2
9	Visualize and download full workouts.	Create the feature of requesting detailed workout information performed on a specified day. Also add the option to download it to the external storage.	Ch.2
10	Study about each provided exercise.	Find a description which suits best to be presented to the user for each exercise.	Ch.2

Figure 1: Secondary Objectives

means or using a digital format, the user will have to secure a significant part of his workout time to this task which will hinder him from performing his routine at maximum capability, which is exactly what this application tries to achieve. In order to help the user skip this unpleasant experience, or at least speed it to a certain degree which is almost unrecognisable, and also bring improvements to the part concerning reading and analysing the introduced data, the application was designed in such a way to cover all this tasks. Each and every one exercise performed by the user, along with the introduced weight and the number of repetitions, will be stored into a database. The user will then be able to sort the saved workouts by the desired date. He will be provided with a list of each performed exercise, and number of sets performed for each exercise along with the number of repetitions done and the weight used at each set. Moreover the user will be provided with information about the duration of each exercise, this being very useful for exercises like planks which do not depend on repetitions but more like on time. This way, the user will have a full written specification of the performed workout.

3.Saving Data

Among regular or sport interested people, there are a lot of personal trainers which choose to design such text based workouts in order to send them to clients. Most of the personal trainers do want to help their clients even if they are on holidays or not being able to go to the gym. That's why they tend to send this kind of workout formats to their clients in order to guide them to what should be done. Even if they are not away, such written workouts could help them with their clients by not having to memorise each exercise. That's why a possible solution would be to use the application while training for himself, or even recommend it to a client, in order to have all the information nicely saved into a friendly format. In order to make the sending task possible, the user will have the option of saving the workout to his device after visualising it. By using this feature, the trainer could even make minor changes to the workout, like updating the repetition number or the used weight, in order to match with the specific client.

4.Personal Records

A big percent of bodybuilders think of personal records as the perfect way to see improvements in their strength. Moreover, a new personal record could "fuel up" the user by motivating him to give his best. Striving for new records at different exercises could even help people avoid plateaus(not being able to advance/stuck at a certain level). That's why the application provides an interface which accurately shows PR improvement through time at each specific exercise. This way, the user has the capability of checking whenever he wants the status of his progress through a bar graph which indicates at each day the recorded set using the heaviest weight among with the number of repetitions regarding the set of the specified exercise. Moreover, the user has the option of filtering the results in order to check the performance captured in a certain month or a certain year. As mentioned, the system will remember only the set with the highest used weight in order to keep track only on the possible personal records. Concluding, if the users objective is to constantly outdo his performance and keep getting stronger, this feature fits best for him to check if his goals have been reached or not. The visual representation of the graph could be of a big help when the user wants to check his performance over a longer period of time, like a whole year for example.

5.Exercise Info

People going to the gym often find themselves in situations where they don't know what exercise they should do or, if they are roughly new to gym training, they don't even know what the correct execution of a certain exercise should look like. That's why I thought of implementing some additional functionalities to the application in order to solve this problem. First of all, the user is provided a scrollable list with different exercises grouped on muscle groups. This way the user has a variety of exercises from which he can choose from, among them being exercises he already knew but maybe forgot about them or even new exercises which the user didn't heard of before. After selecting the desired exercise, the user also has the option of requesting additional information about it. A short description about the exercise will be provided in which the exercise is briefly explained, a photo indicating what specific muscle is this exercise actually targeting and also a short video indicating the correct execution of the exercise. With this feature, the user won't have to worry about not knowing what exercise to do anymore. Moreover, studies show that muscle growth is strongly dependent on the variety of exercises performed. When doing the same exercises over and over again, the muscle tends to learn the movement thus making it less effective when repeated on a weekly basis. As the famous bodybuilder Arnold Schwarzenegger said, "shocking the muscle" is essential to muscle growth. By that he means to always try new things with which the muscle is not used to. One of the best modality to do such a thing would be to have a variety of exercises from which you can choose, which is exactly the main objective of this task. Lastly, an archive with already performed exercises would be also useful in order to be able to look at the exercises performed one week before, in order to know what to change. This feature is also existent within the application and has already been explained above.

6.Selecting the desired exercise type

When first opening the application, the user is presented with the home screen which contains a scroll bar from where the user has the possibility to select the name of the desired exercise. After doing such a thing, a lot of other options become possible, like getting information about the exercise, requesting PR information about the exercise or starting to train by performing the selected exercise. The user can always select another exercise from the provided list. By doing such a thing, the before presented options will have the same scope but change their implementation regarding to the new selected exercise. As a fitness application, it is very important to give the user the option to easy change from one exercise to another, because through the workout he should use the application as fast as possible, the main focus still remaining on the performed workout, not on the application.

7.Capturing and saving sensor data

When performing a specified exercise, the user has the option of starting or stopping the recording of the accelerometer signal generated from the execution the exercise. When the start button is pressed, the system knows that he has to record all the incoming data signal and save it somewhere, namely into a database, in order to be used later for calculating the number of repetitions. On the other hand, when the stop button is pressed, the system know that the data coming from the accelerometer is of no use thus ignoring it completely. By default, the system does not care about the signal data so in order to trigger it, the start button should be pressed.

8.Navigating through the application
One of the most important aspects regarding the graphical interface of the application is strongly connected to how easy the user can navigate from one page onto another. Buttons have been strategically placed on each view in such a way that the user can trigger the load of another page by simply pressing them. Image buttons have also been used in order to help with this matter, each of them having specific images which help the user know where the activation of that button will lead. For example by pressing the question mark button button, the user will be redirected to the page regarding supplementary information about a specific exercise. Moreover, back buttons are available on each existing page in order to help the user reach the previous visited page.

9.Filtering the provided data
Some of the existing functionalities provide a way for the user to request from the system data regarding previous performed workouts and exercises. After a long time and many recorded data, the user will start to have difficulties understanding the returned results. That's why a filtering method was implemented in all cases where this approach could be possible. The first place is the functionality where the user requests for the personal records for a certain exercises. Here, the user can select the desired month along with the desired year of the period he wants to analyse the results of. Following, a similar approach was used at the functionality where the user can request the whole performed workout of a specified day. The date is chosen by navigating through a calendar and choosing the wanted day, month and year. With the help of this feature, the user can reach the searched information in a more favorable time interval and he can as well understand it better.

10.Specifying desired file name
As already explained above, the user has the possibility of saving the requested daily workout. When doing such a thing, the file containing the workout data will be saved to the external storage of the phone in a predefined folder. In order to skip the user of the extra work of saving the data with a random name and having then to rename it, the possibility of selecting the name of the folder, where the data will be saved, was added to the application. By doing such a thing, custom names for each saved workout could be assigned in order to help the user recognise each text file by their description.

11.Internet connection
In order for the application to be able to perform all the designed functionalities, a working internet connection is required. Because of the fact that the application uses the CloudAMQP RabbitMQ server in order to send messages between the backend applications, the device hosting the application will not be able to send this messages to the server without being connected to an internet source. Moreover, the user should make sure, if he uses a public wiFi for example, that the internet connection does allow such a functionality, because some of them are designed is such a way to block this type of message transfers.

12.Active running servers
The backend application code was divided in three main parts. Each of their functionalities will be explained in the following sections. The main idea is that two of them should run continuously in order to be ready whenever the third one wants to send or to request data. If they are not running, then the application won't be able to get the requested information and present it to the end user. Alongside the two servers, the database server should be working as well in order for the system to be able to query the requested information. Each and every one of the presented servers should be running at the same in order for the application to work as desired.

13.Text editor
As presented above, the application has the functionality of downloading specific workouts to the external storage of the device as text files. In order for the user to be able to open this saved text documents he should already have an application, like notepad for example, which permits him to read and even edit files with the .txt extension. Otherwise, the user won't be able to view the downloaded workouts hence neither be able to edit them.

2.3.2 Non-Functional Requirements

1.Scalability
The application was designed in such a way that not only smartwatch users, but also other android using devices, can run it as well. But having in mind that the application should gather data from a wrist applied sensor, anything beside a smartwatch or a smartphone won't be recommended to be used for this task. However, if a further implementation will provide authentication functionality, it could be of great use to be able to log from a table, a laptop or a personal computer in order to visualise and analyse the recorded results.

2.Reliability and Consistency
As long as the functional requirements are satisfied, the system should perform without failure all the available functionalities. Moreover, the results will remain consistent, meaning that for a given input, different executions of the same method will give the same output. For example if the user requests the

performed workout of a specified day, the workout information will remain the same no matter how many time the user requests it.

3.Efficiency
Although some of the functionalities require communication with the outside world in order to present to the user the requested information, either if some queries must be made in order to retrieve some database information or if some application must compute received data and send the result back to the user, the implementation, which will be in further sections discussed, was designed in such a way that the client won't have to wait only for a short period of time. In both cases, the client should receive the result in less than five seconds.

4.Fault Tolerance
Some of the provided functionalities could be misused, reason why a backup option will always be provided if something like that happens. For example, the user could start recording the performed exercise by mistake or when not ready, that's why if no actual repetition of the current exercises is performed, the recorded data won't be saved in the database and the user will be able to try again when ready.

5.Learnability
An new user which is not used to working with the application should perform almost as good as good as an experienced user and be able to learn the full working flow after using it for a few times. This is because of the predictability of the system which means that the provided lables and button names should help the user know the output of each action even before performing it. Buttons with specific images were used in order to enhance this system attribute. Moreover, some operations provide feedback information in order to help the user better understand what happened. For example, a not so experienced user won't know where to find the downloaded workouts, that's why after each save, a message specifying the file path will be presented. Another two features which help the user better learn and use the application are generalisation and consistency. This means that similar functionalities will have similar looks and similar implementation. For example, all buttons which trigger an action have the same form and color and all input forms will work the same way.

6.Maintability
Because the fitness industry is continuously increasing and new exercises with new executions always appear, the system should have the possibility to be updated at a regular time. The system was designed in such a way that it permits not only updates of existing information but also addition of new exercises in a very easy manner. By doing such a thing, the user will have the confidence that he always stays updated with the newest things regarding this concern.

2.4 Metrics used for evaluation

3 Bibliographic research

Regarding the problem of activity recognition, a lot of studies have already been made in this domain because the need of physical exercise has been constantly increasing in the last years. This domain has not just only the goal of helping the ones which have as goal muscle increase, but also helping people with certain disabilities or health problems. Moreover, a lot of articles describe the way in which normal daily activities can be tracked with the help of various technologies.

In [14] for example, the discussion is mainly centered on the location and the type of the sensors used in order to gather data. The first discussed approach being the one in which only a sensor is used, namely the one incorporated in the smartphone, which is usually carried at the trouser pocket position. For recognising repetitive activities like cycling, running, walking upstairs, walking downstairs this approach would show pleasing results because the repetitive pattern of motion takes place mostly around the trouser pocket location. On the other side, there are a lot of movements which involve a lot of hand motion, like smoking, writing, drinking coffee, eating, etc, which makes them very hard to be decipherable only with the help of the previously discussed smartwatch sensors. That's why the addition of minimum one wrist worn sensor becomes necessarily. Furthermore [14] divides the activities in two main categories: simple and complex. The difference between the two is marked by the type of sensors used in order to recognise them. For example, simple activities are the ones which keep a repetitive motion, like walking, jogging, biking, writing, typing, sitting and standing. This group of activities could be easily recognised only with the use of an accelerometer at the pocket and wrist position. Following come the more complex activities which, in comparison with the simple ones, are not as repetitive and they also involve more hand gestures. In order to successfully classify such gestures, some sensors are needed in addition to the accelerometer, like gyroscope and barometer. With a combination of this three sensors, even differentiation between stairs up walking or down walking becomes possible. Another important matter discussed in [14] refers to the size of the segmentation window used in order to label the activities, simple activities needing a smaller window size due to their repetitive state and complex activities requiring a wider segmentation window because of their constantly changing pattern. More information about sliding window segmentation can be found in [7].

As previously said, [7] focuses mainly on recognising physical activities with the help of variable sliding window segments and a single tri-axial accelerometer. This sliding window is nothing more than a time interval in which the gathered data is analysed. Because the sensors provide a continuous data output, over a longer time period, it would be very hard to analyse at all the data at once, that's why experts say that it is way better to look at it divided in smaller segments. Until recently, this segments had a fixed size, independent of the activity type. What [7] tries to achieve is an implementation of a sliding window of variable segment sizes, in order to increase the recognition success rate. The publisher states that the increase rate is quite significant, being the fact that the previous implementations could achieve only a maximum of 89.9 percent accuracy, when the variable size segments approach rises this rate to a maximum of 96.5 percent. The proposed solution consists in observing the initial signal, the first step being to classify it into a dynamic(more physical activity is captured) or a static(less physical activity is captured) signal. Based on this decision, the segment size of the sliding window will be increased, decreased or kept constant. Using this dynamic changing approach, the accuracy has seen a significant increase in comparison to the fixed size approach.

Another approach, vastly used in the domain of human activity recognition(HAR), is the concept of neural networks. This field of artificial intelligence was of big help in terms of improving the results of HAR. Article [12] makes use of this concept in order to build a fast and robust deep convolutional neural network which is used in order to speed up the recognition process up to 0.0029 seconds per activity and with an accuracy of 95.27 percents. The data is gathered from a tri-axial acclerometer embedded in a smartphone and send to the software application via a wireless connection. After the raw data is collected, it must be processed in a way that it could be feed as input to the neural network. The three steps in order to obtain such a thing presented in [12] are: signal processing, data compression and signal selection. The first step, signal filtering, has the goal of reducing the interfering noise as much as possible. In order to do such a thing the raw data must be passed through an Low-Pass Elliptic Filter. This filter has the scope of removing the abrupt changes and the high-frequency components of the original signal. Furthermore, the pre processed data could be improved by using data compression in order to remove similar or even identical signal fragments. This way, a smaller sized signal will spare not only memory but also computing time. After all the data is ready, the neural network will use its inner layers in order to transform the input signal into classified activity names.

In comparison to the previously discussed articles, [3] does not require the need of a smartphone but rather of a smartwatch in order to gather and send data to the software. The main focus of this article falls onto the best deep learning techniques existing in machine learning, which can be used in order to classify physical activities. As [3] explains, the smartwatch technology has drastically increased in the past years, making it possible to use their knowledge and algorithms in order to achieve the main goal of this project. Among all those algorithms, the publisher of this article has decided to use the one of the most commonly used deep learning algorithms today, namely Restricted Boltzmann Machines. Restricted Boltzmann machine(RBM) is an algorithm useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling. This algorithm is constructed on the foundation of a neural network which works similarly to the one described in [12]. Throughout the experimental work of this article, the publisher states that each result was compared to already existing outcomes which use algorithms different form the Restricted Boltzmann machine one, and in most of the cases, the RBM pipeline was able to outpeform the existing results.

Until now we have seen that a sliding window approach would fit to reach the goal of classifying signals into activities and as [7] presented, using a variable segment length can significantly improve the algorithm performance. However, [19] intends to prove that activities with complex motion states and non-periodicity can be better monitored if the monitoring algorithm is able to accurately detect the duration of meaningful motion states, thing which cannot be acquired in the sliding window implementation. The proposed solution in order to resolve this problem is to divide the activities in two classes: non-periodic activity with complex motion states(tennis) and the weakly periodic activity with complex motion states(swimming). [19] presents the approach of first detecting a meaningful motion within the input signal and identify the activity only after a such segment is found. Based on the activity class mentioned above, different detection methods are used in order to find a meaningful segment.

The first and one of the most important step in this project has to be type of sensors used in order to capture the analog data which will be later interpreted by the neural network in order to classify it into an activity type. This step is deeply analysed in [9] where various tests are done in order to differentiate the success recognition rate of data recorded with the help of an accelerometer, with the help of a gyroscope or with data gathered from both an accelerometer and a gyroscope combined. As the results show, the accelerometer performs best in counting the repetition of a certain exercise, whereas the gyroscope does a great job in identifying the exercise type. The article concludes the fact, that in most exercise types, the accelerometer performs better than the gyroscope sensor, with small exceptions. Even then, there is no doubt that when both sensors are combined the accuracy rate reaches the highest values.

Rehabilitation exercises have a very important role in the recovery of a patient which suffer of muscle injuries. With their help, muscle pain could be driven away and in many cases muscles can be rebuild in time if a proper routine is followed. Moreover, most doctors recommend certain exercises to patients which have just had a surgery in order to speed their recovery. With the help of activity recognition, [18] tries to create a system which is able to monitor and help patients in need, throughout their whole rehabilitation period. The author classifies activity recognition in two main categories, namely, vision-based and sensor-based methods. For vision-based methods, human actions can be viewed as a set of spatio-temporal changes of appearances or motions. This paper focuses on the other category type, which uses wearable sensors for data gathering. With the help of multipath convolutional neural network [18] successfully creates a system which is able to rate execution of exercises by score.

This way, patients can improve themselves by analysing the results and trying to achieve an execution score as high as possible.

Among the already discussed relevant topics like muscle increase and rehabilitation assistance [1] comes with a new use for the human activity recognition domain, namely the remote monitoring of elderly people with the use of three accelerometers, one placed on the chest, one on the left ankle and one on the right thigh. As the author states, elderly people need continuously to be monitored because health problems could appear at any time after a certain age. That's why the main goal would be to gather relevant data, preprocess and send it to healthcare suppliers which will be able to monitor the subject's motions during daily activities and also to detect unpredictable events that may occur, like a fall or even a heart attack. With this type of technology preventing serious and permanent health injuries could not be a problem anymore. Four supervised classification techniques namely, k-Nearest Neighbor, Support Vector Machines, Supervised Learning Gaussian Mixture Models and Random Forest as well as three unsupervised classification techniques namely, k-Means, Gaussian Mixture Models and Hidden Markov Model, are compared in order to find out which technique performs best in activity recognition and classification. It seems that the supervised approaches perform better when compared to the unsupervised ones in terms of raw data.

Next to exercise recognition and repetition counting stands another important factor, not only for muscle increase but also for medical recovery. The intensity of the exercise plays a very important role and could be the key which makes the difference. Hence it would be a necessity for an activity recognition system to be able to calculate the intensity as well. [10] focuses on the aspect in more detail. The author states that in general it would be easy to keep track of an intensity with the help of a GPS by measuring the position and the speed, but when talking about upper body weighted exercises, this approach would most likely fail. The proposed solution is a hierarchical algorithm which is divided in two parts, first identifying the exercise, then calculating the intensity. The author chooses to use three accelerometer sensors placed in the chest, right wrist and right elbow. In order to best recognise the intensity of an upper body exercise, it seems that the sensor placed on the chest is of the biggest help because the higher the intensity, the more the person swings throughout the whole motion of the exercise.

An important problem in recognising physical activities is how to make the difference between free weight(i.e., bench press, deadlifts or squats) and non free weight (i.e., walking, running or sitting) exercises. The solution proposed by [11] are One Class Support Vector Machines(OC SVM). Depending on the result of the OC SVM, different methods will be used in order to classify the exercises, using Neural Network fine-grained classification for non free weight and LVQ-HMM finegrained classification for free weight activities. So the support vector machines are not used in order to find the final output representing the activity name, but rather to divide each activity into one of the two classes, free weight or not free weight activity. Support vector machines are suitable for this problem because they are able to analyse an initial data set and to create a boundary between the two classes we have to differentiate based on some activity specific parameters. In our case this parameter could be the values gathered from the accelerometer. SVM could work on multiple parameters as well, like accelerometer and gyroscope values, in order to gain a better accuracy.

The size and quality of the data set on which the initial training is done represents an important factor when targeting for high accuracy results. [5] tries to implement activity recognition algorithms on data collected from a single person performing 30 upper body gym exercises. The data was gathered using an accelerometer. Even though the results were not bad, the author said that a more detailed data set, collected from more individuals and perhaps on different exercises, can significantly improve the current results. That's why the author lets the project open for further improvements.

Another important step in physical activity recognition is the separation of, from the sensor gathered data, into actual exercises and background activity. [8] describes the fact that the separation of physical activity from background activity depends a lot on the working routine of each individual. If for example an exercise is performed, then some resting is done, then it would be quite easy to differentiate the two states based on the dynamic and static movement of the sensor data. On the other side, if the user chooses for example to stretch or to do some non-resting movement between sets, then the separation task becomes quite problematic. The suggested approach is use a five seconds sliding window and transform each segment window into 224 features, representing an actual number of 28 features computed over 8 one dimensional signals, which are used in order to characterise exercises.

The next step after identifying a specific exercise would be to provide real time but also post workout feedback. In order to do such a thing, [17] proposes machine learning combined with pattern recognition techniques in order to detect mistakes or to use a model based approach and to compare motion traces recorded using sensors with a predefined specification of what constitutes a correct execution. The author states that within the training data some of the individuals have been told to execute exercises in a wrong manner, in order to better recognise mistakes when analysing data. However, this approach would not be scalable because there are too many wrong executions to keep track of all.

As previously discussed, repetition counting is a big help when designing a physical activity recognition system for supplying real time and post workout feedback. Another important concept in this domain is brought up in [6], namely weight monitoring. As the author state, the weight used when performing an exercise does have a significant role in muscle increase. For this time being, [6] has developed an application used in order to monitor weight only for weight assisted machines, letting place for further implementations like monitoring the weight of dumbbells and barbells loaded with disks. In order to achieve a successful monitoring of the weight, three sensors have been used on the weight rack of each weight assisted machines. By calculating the movement of the weight plates along with the weight placed on the bottom plate, not only the used weight can be monitored, but also the repetition count of each set. By achieving such a functionality, the user could be able to analyse his workout and make a summary of the performed work after a certain number of workouts. In time, this statistics will prove useful if the athlete wants to test if any improvement in term of weightlifting could be seen.

Depending on the targeted result, data gathering could be easy or more complex. For example if the main goal is to detect exercises designated to heavy weightlifting, the resting period between sets would be made of a considerable period of time, hence data could be split more easily into exercise performing and resting. [15] has designed a system in order to track a workout routine build only of cross fit exercises. As the author states,in this case, data gathering is no easy task because cross fit exercises have a more dynamically structure, with a very short period of resting time, making them harder to be recognised and analysed. That's why, the author comes with the solution of supervised data gathering. This means that a certain number of individuals were asked to perform cross fit exercises wearing a smartwatch, the main requirement being to start executing the exercise only after they feel a vibration from the smartwatch. By doing such a thing, the splitting of the data into resting time and exercise execution could be made more efficiently. However, the author chooses to perform a series of unsupervised data gathering as well in order to use it for testing.

An important problem in physical activity recognition, namely the classification of exercises consisting of low repetitions, is described in [16]. The author explains that it is a lot easier to recognise an exercise which consists of a continuous movement, like for example a repetition range of 8-15 reps, rather than exercises which consists of only one repetition. The article proposes the method of converting, from the sensor gathered data, into a 2D image which will be feed as input to a convolutional network. The lower the repetition count, the more increases the need of supplementary convolutional layers inside the network. So a low rate of repetitions represent a more complex neural network in terms of computation knowledge.

In article [13], the author comes with an interesting approach, in order increase the physical activity of the users with the help of his exercise monitoring application. He suggest to implement a feature in which the daily monitored data could be shared among friends or other people. By doing such a thing, the users could be motivated to execute more physical activity than they usually do in order to compete with the others. The author state that the most important thing is to make an user friendly application which makes them enjoy doing the physical work. Furthermore, [13] explains how the activities should be well defined in order to make rid of confusions like which is the difference of fast walking or slow running. There should be a speed barrier defined for such exercises in order to make their classification as easy as possible.

An important problem regarding data gathering from sensors is described in [2]. In real life situations things often do not go as smoothly as planned, that's why when gathering data from individuals performing physical exercises we would have to deal with sensor displacement. It is highly possible that the sensor will move from his initial position during the performing of the exercise or even the case when the sensor is badly placed at the beginning and it remains that way through the entire process. This unwanted behaviour could affect a lot the interpretation of the data, hence it could lead to wrong results. The author states that within the sensor, the acclerator component due to rotation is highly sensitive. A recommended solution would be to combine other sensors with the accelerometer, like a gyroscope for example, in order to try to compensate for the missplaced sensor with some additional information. In [4] an interesting method is described, in which a genetic programming approach is used in order to extract discriminative features from acceleration data. The main idea is to find a mapping from the sensor signals to a feature, with the help of a fitness function.

4 Analysis and Theoretical Foundation

In order to create the final application numerous frameworks, programming languages and concepts have been combined. In this section they will be analysed and described in more detail. We will try to focus as much as possible on the information which is related to the design of the application. The features will be firstly explained separately from another, following to be described their relation with the other concepts and how they work together in order to achieve a certain goal or implementation.
In order to better understand the design, let's think of the whole system as a pipeline having the user interface, which sends and requests data, at one end and the computing functions, which receive, process and send the data back, at the other end. One type of data send are the values recorded by the smartwatch sensor with the goal of receiving in return a number representing the number of performed exercises. In order to better understand how this process works we will firstly split it in two parts.

Sensor data
The device used, being it a smartphone or a smartwatch, contains a lot of useful sensors which can be used not only for activity recognition but also for repetition counting. Some of the most important ones are the accelerometer, the gyroscope and the magnetometer. As [3] explains, the accelerometer alone can achieve a performance of 96-97 percent. It is of no question that the accelerom-

eter performs way better than the gyroscope alone, which reaches only around 80-85 percent accuracy. Of course, because of the vast type of movements, these measurements can vary a little depending on the type of the performed exercise. Even then, there is no doubt that the best results are achieved when both of the sensors are combined. In this work I have chosen to use only a three dimensional acceleremoter in order to calculate the number of repetitions. The way an accelerometer works is the following. The main definition is that the accelerometer is an electronic sensor which measures the acceleration forces acting on an object, in order to determine the objects position in space and monitor the objects position. For example, in mobile phones it is used in order to determine the orientation of the device. As mentioned, the accelerometer tries to measure the acceleration forces which act on an object, the smartphone in our case. In order to better understand let us explain what acceleration really means. Acceleration is defined as change in velocity over change in time. So as long as a constant velocity is kept, there will be no acceleration present. Either the speed or the direction of the movement has to change in order to say that an object has accelerated. There are many formulas which help us to calculate the acceleration, but the one used in smart devices is the one which states that the acceleration equals the force applied on the object, which actually causes the change of velocity, divided by the mass of the object($a=F/m$). By having this in mind, let's try to understand how the accelerometer inside a smartwatch is able to calculate its acceleration. This is possible with the help of MEMS, which stands for Micro Electro Mechanical System. As its name suggests, they are very small cips, at the order of 20 micrometers, which contain not only electrical but also mechanical parts, namely capacitors respectively springs. Let us analyse the following two figures in order to understand how MEMS are able of calculating the acceleration.

The concept is the following. As we can see, the first figure represents the initial state of the object. On the other side, when a force is applied, the system starts to move, making the distance between the dark blue capacitors, which always remain in a fixed position, and the light blue capacitors, which change as a force is applied, to vary in time. As this happen, by analysing the created distance, the capacitors are able to calculate the applied force. After adding the value of the force in the above defined formula, the calculation of the acceleration becomes really simple.
An important mention would be that the acceleration is a vector, meaning that it has a magnitude but also a direction. That is the reason for which in order to achieve the best accuracy we need to record the data of three different accelerometers, in order to cover all the three dimensional axis: X, Y and Z. There are a lot of movements which encounter major changes only on a single axis. For example, take the bench press exercise, where the sensor, if placed on the wrist, moves only up and down. That's why in this case we would need only at the axis on the vertical axis. Some other exercises, like standing rows for example, encounter changes only on the horizontal axis. These details were closely analysed at the computation of the repetition number computation. An-

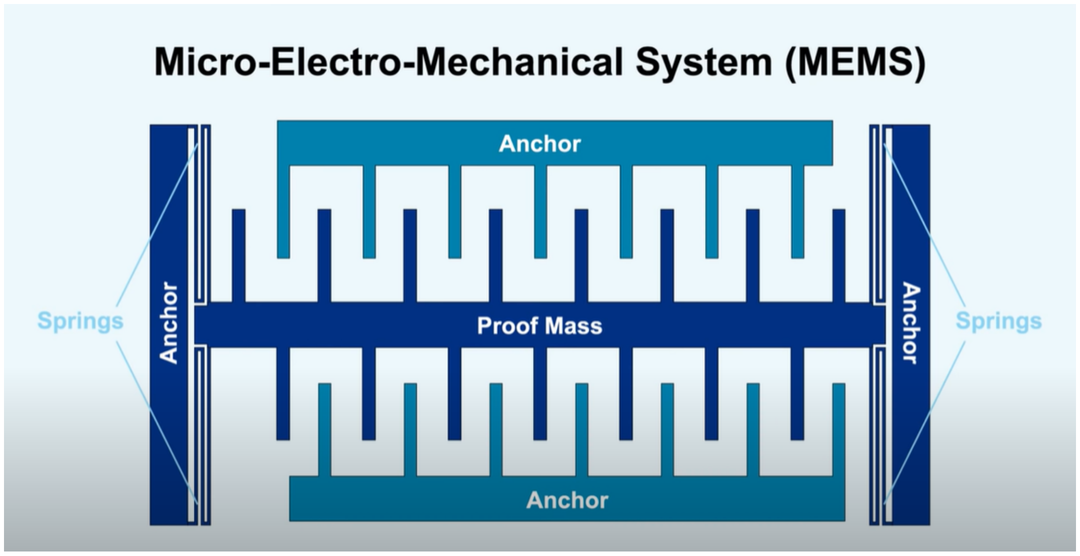


Figure 2: Initial position

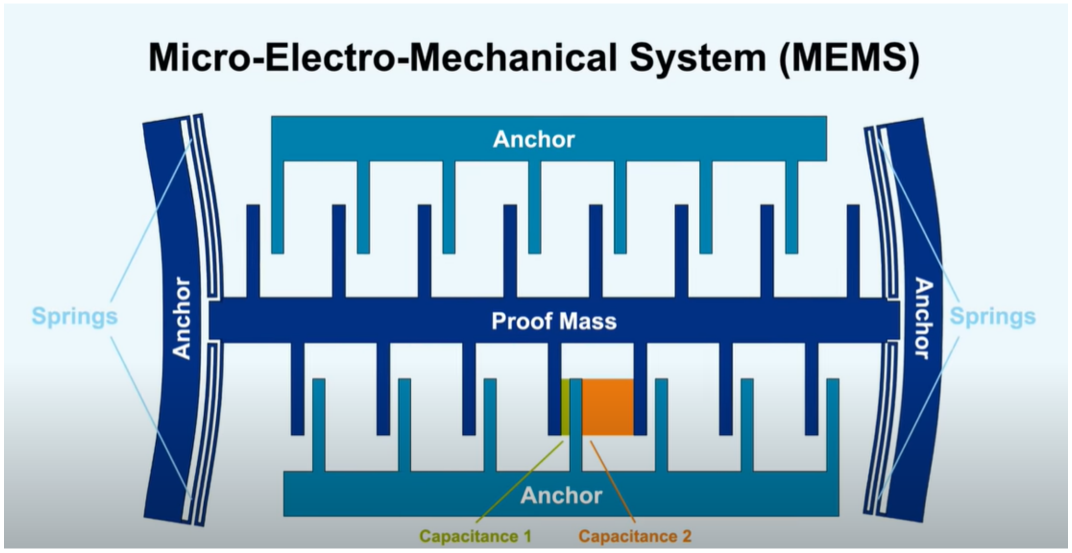


Figure 3: After velocity change

other important matter is the positioning of the axis, because different devices could have different positions. Most of them, including the Kospet Optimus Pro, which is used for building the application, has the axis as presented in the following picture.

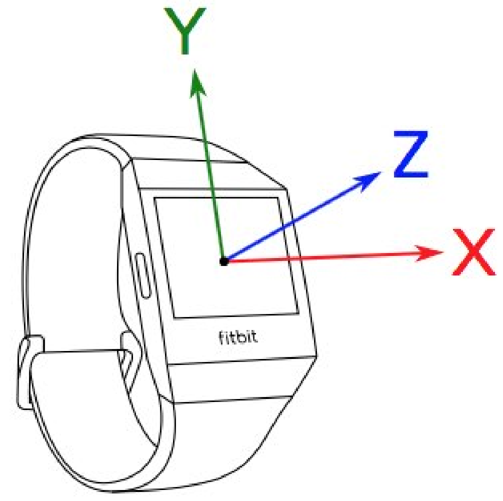


Figure 4: Direction of accelerometer sensors

Being the fact that the direction of the sensors are always fixed relative to the position of the smartwatch means that when performing an exercise, the user must try to keep the wrist as much as possible in the same position as the initial one, because if a specific exercise was computed by capturing changes only on a certain axis, if that axis changes the system might not return a correct result. That's why an almost correct execution is desired, reason why the information feature, described in chapter two, was implemented for each exercise. This fact brings another possible problem and question regarding to the placement of the sensor, which will be described in the following section.

Sensor Placement
A lot of studies regarding the ideal number of sensors along with the best way in which to place this sensors have already been made. In [14] are presented some results calculated on different activities with different number of sensors. It is of no doubt that in most cases a large number of cases can prove to significantly increase the accuracy, but as I found myself, a lot of physical exercises performed at the gym do not interfere changes only on a single axis, reason for why a single tri-axial accelerometer would be enough. Moreover, an increased number of sensors would increase the computational operations as well, hence increase the time in which the user waits for a response. Because of that, a single device containing a tri-axial accelerometer was chosen for the design of this application.
Following let us talk about the placement of the sensor. There cannot be declared a certain spot for the sensor, because different exercises will require the sensor to be placed in different places. Most of the exercises, including all of

them which require hand movements, will work just fine with the sensor placed at the wrist. Some of this exercises are: bench press, all type of rows, biceps curls, triceps extensions, deadlifts, etc. Following come the exercises which involve leg movement. Most of this exercises, with some small exceptions, like lunges or squats for example, will require the sensor to be placed around the ankle. This way, the movement of the legs can be traced and send to be processed. Some examples of this exercises are: leg extensions, leg curls, leg press, bulgarian squats, etc. Lastly we have the exercises which require the sensor to be placed somewhere at the waist. There are not a lot of exercises which cannot be processed with a leg or hand placed sensor, but some of them won't be classified as long as the sensor is placed at the waist. Exercises like hip thrust, pull ups or even push ups will keep the ankle as well as the wrist in a fixed position throughout the whole execution of the exercise, but if we look at the waist, it changes its position at each and every repetition. In order to better place the sensor at the waist location it is recommended to attach it to a weightlifting belt.
As [2] specifies, sensor misplacement is a considerable problem, because the system expects some kind of data and if the sensor is placed at a wrong position or in a wrong manner, it won't be able to successfully count the number of repetitions. In order to make sure that the user won't encounter this kind of problem, at the additional information feature, each exercises comes with a description and an execution video, where the position of the sensor is clearly specified.
Either if we talk about a smartphone or a smartwatch the above mentioned requirements remain the same, meaning that both of them could use the fitness app for repetition counting if the conditions are meet.

If the physical part of the application is clear, let us move on to the theory behind the application code. The following concepts servers and environments along with the used programming language were carefully combined in order to achieve the final product. Each one of the has a significant part in the good functioning of the end application.

Android Studio
The main part of the application was build using the Android Studio environment with the help of kotlin programming language. The android version of the Kospet smartwatch being 7.1.1, the environment was build in order to support this version of android. The application user interface was build in order to satisfy the dimensions of a round smartwatch, but it works just fine on the mobile smartwatch as well. In order to be able to communicate with the RabbitMq server, which will be discussed next, the aplication was configured to permit internet communication. All the data going outwards or inwards from the application has to pass through the RabbitMq server. Neither data processing nor database operations are done in this part of the application but rather in the following ones.
As for the environment, Android Studio is the official integrated development

platform used by Google developers to design the Google android operating system. The platform is build on top of JetBrains IntelliJ IDEA software. It was especially designed for android development and was build in with the scope of replacing the Eclipse Android Development Tools. The platform was announced in may 2013 and released in jun 2014. Until the early start of the year 2019, the main programming language used by the developers was Java. After may 2019, the Google's preferred language was changed with Kotlin, which is the same programming language which was used throughout the android part of this project as well. Even so, Java and C++ are still supported. The version used for this project is the latest one, which was released in May 2021, namely version 4.2. Some of the most important features provided by this latest version are:

- Gradle based built in support which controls the development process regarding to compilation, testing, deployment and publishing.
- Andoid based refactoring as well as quick fixed
- Lint tolls used at run time in order to detect potential bugs and check for correctness and optimisation.
- ProGuard tool for code optimisation(inlining and deletion of unused code plus renaming of classes methods and fields).
- Templates with already created functionalities on top on which applications can be build.
- A well build graphical layout editor which contains numerous type of components which can be drag and dropped in order to build an interface, which can be previewed on multiple configuration screens.
- Support for building Android Wear applications.
- The presence of and android virtual device called emulator which permits to run and debug programs inside of the environment.

Android Studio is available on operating systems like Windows, Linux or macOS.

CloudAMQP
CloudAMQP is a hosted RabbitMQ server which lets you exchange messages between processes and other systems. Following, RabbitMQ is a messaging broker which means that it is an intermediary for messaging. It provides the applications a platform to send and receive messages, and the messages a safe place to be kept until received.
Basically, in order to enable communication between the different working API applications, running on the Android Studio, Visual Studio and PyCharm environments, I decided to use a message broker for sending and consuming messages. The message broker is working on top of the RabbitMq server. As

previously said, internet connection is needed in order to send or consume messages because RabbitMq is an online server.
Following will be explained in more detail how publishing consuming messages really work. If we take a look at the following figure we can identify four main components which all work together in order to achieve the functionality of publishing and consuming messages.

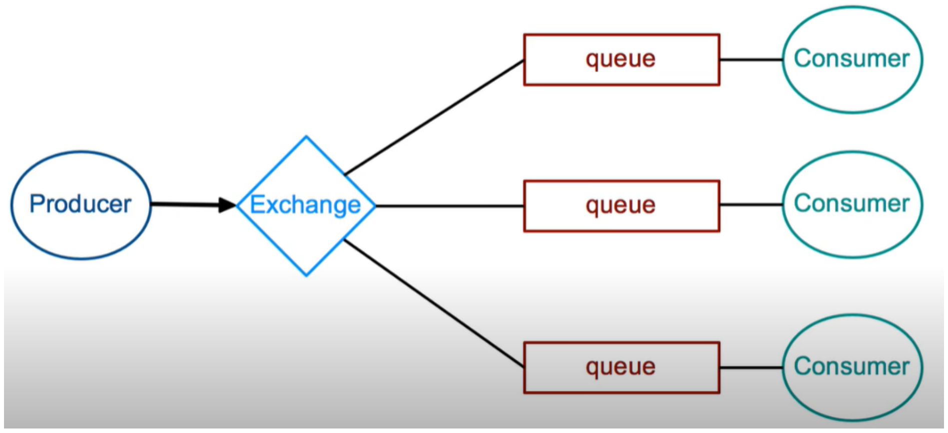


Figure 5: RabbitMq components

At the ends of the diagram we can recognise the Producer and the Consumer entities. Briefly speaking, the producer is the one which sends the message to the server and the consumer is the one which gets the message from the server. The main question is where are this messages stored until the consumer gets them. Here comes the next main component of this architecture, namely the queue. The messages send by the producer will be kept in a queue until being take away by the consumer. The definition of the queue works on a FIFO(first in first out) principle, meaning that if n messages are sent to the queue, if the consumer chooses to consume one message, only the message which was first send by the producer will be taken out of the queue and send to the consumer. The last question is what happens if we have more than one queue. In order to successfully route each message to its specific queue, the producer never sends the message directly to the queue but rather uses an intermediate component, namely the exchange. The exchange uses a routing key send by the producer in order to identify the correct queue. There are many implementations which help to bind each routing key to the appropriate queue. The fanout implementation, for example, ignores the routing key and sends the message to all the queues. Each exchange is connected to a queue via a binding line, and each binding is associated with a binding key. The direct implementation sends the message to the queue where the routing key equals the binding key. However we will use the nameless exchange, which is also the default exchange of the server. This method implies all queues to have a specific name, because the way it works is that at the binding level, the broker roots each message which has the routing key the same as the queue name. The routing key in this case must be a string. Let us take an example. If the producer will want to send a message to the first queue, which is named "Queue A", he will have to send along with the message the a routing key with the value "Queue A" as well. This way, the consumer connected to "Queue A" will be able to receive the message sent by the producer. By using this type of communication, messages can be send between different applications at any time.

Visual Studio

Microsoft Visual Studio is an IDE(integrated development environment) created by Microsoft with the scope of creating computer programs, websites, web applications, mobile applications and web services. The main feature of this environment is that it accepts a lot of plug-ins which help a lot with the design of different applications. For example, the application for this project created in Visual Studio was only possible with the help of the MySQL and RabbitMQ plug-ins which permitted the use of their specific functionalities at the users will. Visual Studio support as much as 36 different programming languages. Throughout the implementation of this project, the .Net Core programming language was used, which was created in order to create server applications, this being the scope of our application as well. Two of the most important features of this environment are:

1. Code Editor: It uses InteliSense and LINQ queries which provide syntax highlighting along with code completion. Furthermore, while code is being written, visual studio has the feature of compiling in the background.

2. Debugger: Visual Studio debugger allows setting breakpoints at specific code lines. The debugger activates when that specific line code is met, when the debugging window is opened. This one contains not only the possibility of stepping into or over functions but also setting watches which help the user to better visualise the value of different variables.

The current available Microsoft Visual Studio editions are Community, Professional and Enterprise. The only editions which are compatible with the scope of this project were the Professional and the Enterprise one. This is because the designed server application has to be connected with the MySQL database server, in order to fulfill the needs for which it was created, and the Community edition does not provide this kind of capability. On the other side, the Professional edition, which is also the one used for the application creation, do provide, along other additional features, integration with Microsoft SQL Server. Because of this fact, the written application code won't work if run on the Community edition, because in that edition, the connection with the database will not be possible. On the other side the Enterprise edition, because is an enhanced version of the Professional one, can be used as well for running the application

server.
The Microsoft Visual Studio version used is also the latest one, namely 16.9.4.

MySQL Database Server
In order to store different information related to exercises, diet and workouts a database server has been used. A database server is a server used in order to centralise and manage databases stored on the server. They are build on top of the client-server architecture and provide access to authorised users. In order to achieve such a functionality a local database has been used, meaning that its contents can be accessed only locally. The difference between this type of approach and a remote database server is that the latter one can be accessed by any client on the web whereas the first one resides on the local system and can be run and accessed only on the machine it was build on. Regarding to the client-server architecture this mean that a locally run server has only a client, namely the one on the machine the server was build.
Among different types of database servers the SQL server has been chosen, which is a relational database management system implemented by Microsoft. Its main purpose, as a database server, is to store and permit retrieval of data. In order to retrieve data from the database, queries are used. Different database servers use different queries syntax, the one used by MySQL is a variant of SQL called Transact-SQL. The query specifies what is to be retrieved. There are a lot of different Microsoft SQL Server editions, each one aimed at different audiences and providing similar concepts with different features. The one chosen for this project is MySQL Workbench. MySQL Workbench is an integrated development environment and a visual database design tool for database architects which provide data modeling, SQL development, backup and other features as well. It provides the required tools needed in order to build databases, tables and execute queries on it. Moreover, relations between tables can be created by specifying between specific columns. Some of the most encountered relations are one-to-many, many-to-many and one-to-one. After creating such logic with the help of this relations, MySQL Workbench provides the option of Reverse Engineering which creates a visual diagram based on the provided tables in order to better visualise the database. Being created on a local server, the database will be running as long as the machine on which it was created is running as well, if not, the database server will be stopped. When is running, the server permits connectivity to numerous applications running on the same local machine. This way, application could send different queries to the server, which will perform according to the message contained in the query, either by creating, updating, deleting or reading data contained in the database. In order to be able to connect to the database, each application has to create an instance of a MySQL connection having as parameter a connection string containing the credentials which help identify the server and the database, because on the same machine there could be different servers and each of them could contain different databases. That is why the connection string has to contain the server

name along with the port number, which in case of local servers is ”localhost” respectively ”3306”, the user id which is set as ”root” by default, the password set when creating the server and the name of the database you want to perform queries onto.
The version used is the latest released one, namely MySQL Workbench 8.0 CE. If needed for future implementation, this version also permits data migration onto Microsoft SQL Server, PostgreSQL, SQL Anywhere, SQLite, and Sybase ASE.

PyCharm
PyCharm is an IDE(Integrated Development Environment) used for programming a wide range of applications. It was mainly designed for Python, which is an object oriented, high level, easy to learn and use programming language. The program was designed by the JetBrains(IntelliJ) company and some of the most important provided features are code analysis, support for web development, a debugger with a well deigned graphical interface, an integrated unit tester and support for data science because of the fact that the IDE can be combined with Anaconda distribution platform. One of the most important thing at the python environment is that there are a vast number of different libraries which contain specific functionalities which can be very helpful in a lot of cases. This means that a lot of the implementation is already build, giving the user the possibility to use it in order to ease his job. One of the most important library used for the implementation of this project is the scipy library, which stands for scientific python. From this library, the one relevant to our project is the signal module which provides signal processing functions. Among these functions, the findPeaks function was used in the case of counting the number of repetitions of a certain performed exercise. Another library used for the development of the application is the MySql library which facilitates the connection of the application with the database server. Another one is the pika library which helps the application connect to the desired RabbitMq server. This functions will be described in more detail in a following section. Besides libraries, there are more than 1000 plugins which are compatible with the IDE. The PyCharm IDE provides the user the ability to divide the application code into modules, in order to keep it organised and simple to be understood. Because of the fact that python provides this vast range of facilities, all the complex computations will be performed in this IDE and the result will be sent back to the android application in order to be presented to the user.
The application was build by using the currently latest version of the IDE, namely 2020.3.3 and the edition used is the PyCharm community edition.

Personal involvement
Some of the information presented to the user comes from my personal bodybuilding knowledge which has been assimilated throughout many years of training and learning from online means and from meetings with people, involved

in this industry, as well. Because of the fact that the internet is not always trustworthy and a lot of information might be misleading I have chosen to supply the application regarding the physical training and the diet with my own knowledge. Even though I know that everyone is different and some people might need different approaches than others, by trying most of the approaches on myself I have come to understand what works and what not, speaking from a general matter. This the reason why at the informative section which supplies each exercise with a description and a video with the correct execution, I have chosen to write the description without requesting aid from online documentations. Furthermore, the provided video is a recording of me while performing the selected exercise. Moreover, in the section where the user selects his fitness goal, I mostly used my knowledge in order to calculate the required number of calories to be consumed and burned.

5 Design and Implementation

5.1 Design

A lot of individual components have to be put together in order to create the full working application. Each of this components has a specific role into building the final product and each of them must be able to communicate with at least one of the other components meaning that every single component must be capable of receiving data from or sending data to the component or components he is connected with. This way, data can be send from one end of the fragmented application to the other end, meaning that component A can send data to component C without being logically connected if there exist a component B which is able to receive data from A and send it to C. We will first analyze the architectural diagram while describing the role of each component and how it is connected to its neighbouring components. All this components along with their connections are presented in the following image.

First of all let's start with the user, because he is the one which triggers the start of all the functionalities of the system. The main goal of the application is to successfully transfer data coming from the smartwatch through the "pipeline" with the end goal of being saved into the database server and also to send requests to the system which will be answered in the shortest amount of time. By pipeline is meant a channel which connects more applications and which can sustain overlapping of information(requests). From our point of view, briefly speaking, the pipeline has the role of facilitating the communication between all the components. When an application makes a request, we assume that the pipeline knows to which component to send it and from where to take the appropriate response in order to send it back to the application if needed.
In a first instance, the smart device sens information, being it data gathered by the accelerometer sensor or requests made by the user, to the android ap-

plication. In the above presented diagram, this two parts, the smartwatch and the android application, are presented as two separated entities, when actually they are strongly related one with the other, because the android application is incorporated in the smartwatch as one of his applications. The only reason why we present it as two separate components is because it is easier to follow the flow of data. So we can think of the line binding this two components as an imaginary one but for now, let's think of it like presented above. Having this in mind, we agree that all the data coming and going to the user has to pass through the android application. Let us take a closer look at how it works.

Android

This is the place where decisions are made regarding to where to send data coming from the graphical interface of the smartwatch and where to send the requests in order to retrieve the data which has to be displayed to the user. We already mentioned the fact that some of that data must be stored into a database, but as we can see on the architectural diagram, there is no direct connection between the android application and the MySql server. On the other side, we can see that there exists a connection between the android application and the RabbitMq message broker. This means that the only way for the application to send to or receive data from the pipeline is by publishing or consuming messages with respect to the appropriate queues. Beside sending or receiving messages the android application has numerous other functionalities. One of the most important one is to create the graphical interface with which the user will have to interact. Beside doing the user interface, there should be assigned implementations to each possible actions. As a summary, this is the place where each interaction of the user with the system, namely the smartwatch, has to be created and implemented. Furthermore, before sending data to the pipeline, some of it must be firstly preprocessed. Moreover, the application must be ready to receive some of the data from the pipeline into some specific objects. In order to do such a thing, data transfer objects(DAO) are being used. Because the kotlin programming language is object oriented, we can create objects which hold specific data. More about their implementation will be discussed in the implementation subsection. For now, the important thing is that in order to transmit or receive large blocks of data with the same properties, this kind of DAOs are being used. They simply encapsulate more properties together in order to create a so called object. At the receiving end, this object has to be decapsulated in order to use its contents. Multiple objects can be send through the pipeline at the same time. Another feature is the functionality on being able to save information to the external storage of the smart device. Regarding to the described non functional requirements, this party of the system has to make sure that the application is scalable, reliable, consistent, learnable, fault tolerant and maintainable. Most of these concepts have to do with the implementation of the user interface. They were already described in chapter 2.3.2 of this paper.

RabbitMq

Following we should discuss about the central point of the architecture and

also about the single communication method between components, namely the RabbitMq server. As mentioned in the previous chapter, the RabbitMq server is defined as a message broker which stores incoming messages, send by a producer, into a queue, until a consumer makes use of them. There can be multiple queues, each one connected to a different consumer. Clients all among the internet can access the service because the the used RabbitMq server is hosted on the CloudAMQP platform. The RabbitMq manager, provided by the CloudAMQP service, offers an user friendly interface which permits the client to visualise at any time the existing connections, channels, exchanges and queues. Moreover, exchange and queues can be deleted, updated or created from the interface. Another useful feature is the possibility of visualising the current messages waiting in each queue, this being very useful when debugging the applications. In order for the client to be able to open a connection to the RabbitMq server, an Uniform Resource Identifier(URI) is requested. This URI can be found at the details section provided by the advanced message queuing protocol service. If the connection to the server is successfully created, the client application has to create or open a new Channel one which a queue, containing its name and other properties, has to be declared. As explained in chapter 4, the exchange type used in the implementation of the system is the nameless exchange. This means that when the client wants to publish a message to a specific queue, a routing key has to be used, which must be the same as the name of the queue. Moreover, the message has to be converted to an array of bytes before publishing it to the queue.
As presented in the design of the architecture, the RabbitMq component is connected to all of the three applications, meaning that at some point each of them wants to publish or consume a message from the server. In total, there are three queues, each one having a different purposes. The first one is called andreiQueue and is created in the android application. This queue receives messages whenever the user is exercising and the accelerometer data is changing. The same data has then to be consumed by the server build using the visual studio IDE. So this queue is used only for sending sensor data and represents the connection between the android application and the specified server. Not to forget that before sending a message each connection has to be opened and respectively closed after publishing the data.
The communication between the android and python applications are done via the finishSending and repCountResult queues, the first one being declared on the android side and the latter one on the python side. Briefly speking, the finishSending queue is used in order to send a message containing the user request to the python application, which will have then to compute the answer and send it back to the android application, in order to display it to the user, by using the repCountResult queue. This way, messages can pass both ways from one application to another.
All queues were declared as durable, meaning that the containing messages are stored on the disk, not in memory, and in case of a server restart they won't be lost.
As already specified, the service hosting this servers lays on the internet, which

means that it provides remote access only to those clients which are connected to a working internet connection. The components alone cannot fulfill the user needs, meaning that if even one of the three queues is not working, the application will lose most of its functionalities.

Visual Studio app
This component has the simplest implementation among all the applications. It was designed with the scope of saving the data produced by the sensor into the database. The sensor gathered data is produced by the smartwatch component, which is transmitted by the smart device to the android application and following to be send to the RabbitMq component, which is directly connected with the Visual Studio server application. Taking a look on the other part, the MySQL database component is directly accessed by the server application which explains the direct connection between the two. So briefly speaking, the server application receives objects containing data regarding the performed exercises, adds the current timestamp to it, and stores them into the specified table into the database. This application is called a server because it has to be running non stop in order to be able to verify continuously if new messages were published into the queue designed for storing the sensor data and to be ready at any time to take those messages out of the queue and store them into the database server. The andreiQueue queue is used in order to communicate between this application and the android one. If the server is not working, the messages will be stacked one upon the other in the queue and will be waiting until the application will resume its execution. In this case, the communication bus is a simplex one, meaning that information flows only in one direction, namely from the android application to the visual studio server, and not vice versa.

Python app
If we take a look at the above presented figure of the architectural diagram, we can see that the python application and the previously described server are both connected to the same components, namely the RabbitMq and the MySQL server. This means that both application have similar purposes, regarding message transportation. Both were designed in order to create a path between the android application and the MySQL database. Even so, the python application has a much bigger role in the design and implementation of the whole application. First of all, regarding the design, we classified the previously described server connection as a simplex one, but the connection established between the android application and the database is half duplex. This means, that data can not only flow in both directions, but both ends can transmit information at the same time. Even though this possibility exists, because of the connection working on the request-response principle, a half duplex connection(in which only one device can send data at a time) is enough. All the database queries, besides adding the sensor data, are performed here. Each request received from the android application requires, in one way or another, at least one type of CRUD operation on the database information. In order to recognise the different types of request send by the android application, the finishSending queue is

being used, where the message published is the type of the request to which the user expects a response. The response send by the python application might be a number, when the user requests the number of performed repetitions, or a list of DAOs(Data Transfer Objects), when the user requests specific information about the already performed exercises, like personal records or single day workout feedback. The current described component has the role of analysing the database data and after the required computations are made, send to the android application the desired result, result which will presented to the user. How the returned result is calculated and what formulas, libraries and methods were used in order to do such a thing, will be explained in more detail in the implementation section of this chapter. Another relevant thing related to how the connectivity of the design, is the mode in which the python application publishes the data to be returned on the RabbitMq server. In order to do such a thing, a queue with the name of repCountResult was created. As a conclusion, when the android application sends a request on the finishSending queue, the python application, which continuously listens on this queue, will be able to receive the request and act accordingly by computing the requested information. Meanwhile, the android application will be waiting until the result is processed, by listening on the repCountResult queue, on which the python application will publish the result. When sending messages, simple strings converted to array of bytes are used, but when objects are send, either DAOs are used or a list of array strings. The receiving end should know what type of data to expect and act accordingly in order to convert it into what he needs to display.

MySQL database
The last described component is the main place where all the application data is stored. The two components with which it is connected to are the C and the Python applications. The both perform queries which interfere with the information from the database. Each of the two applications make use of the credentials provided by the database server in order to connect to it and to perform CRUD operations. The are several tables created in the database, each one of them serving a different purpose. Even though MySQL Workbench is a relational database which allows different type of relations between tables, for the scope of this project there was no need for such a thing, just for storing the information. So all the four tables are independent one from another, even if some of them hold similar type of information. In the following image we can see the whole database schema containing each table with its columns and their data types.

Each one of the tables has as primary key the ID, an integer which automatically increments itself. Following let's describe each table.

- **traindata**
  This table stores the data gathered by the three axis accelerometer recorded with the help of the smartwatch. The C application is the one responsible for executing the queries which fill this table. The table contains the values of the accelerometer recorded for each of the x, y and z axis as floats;

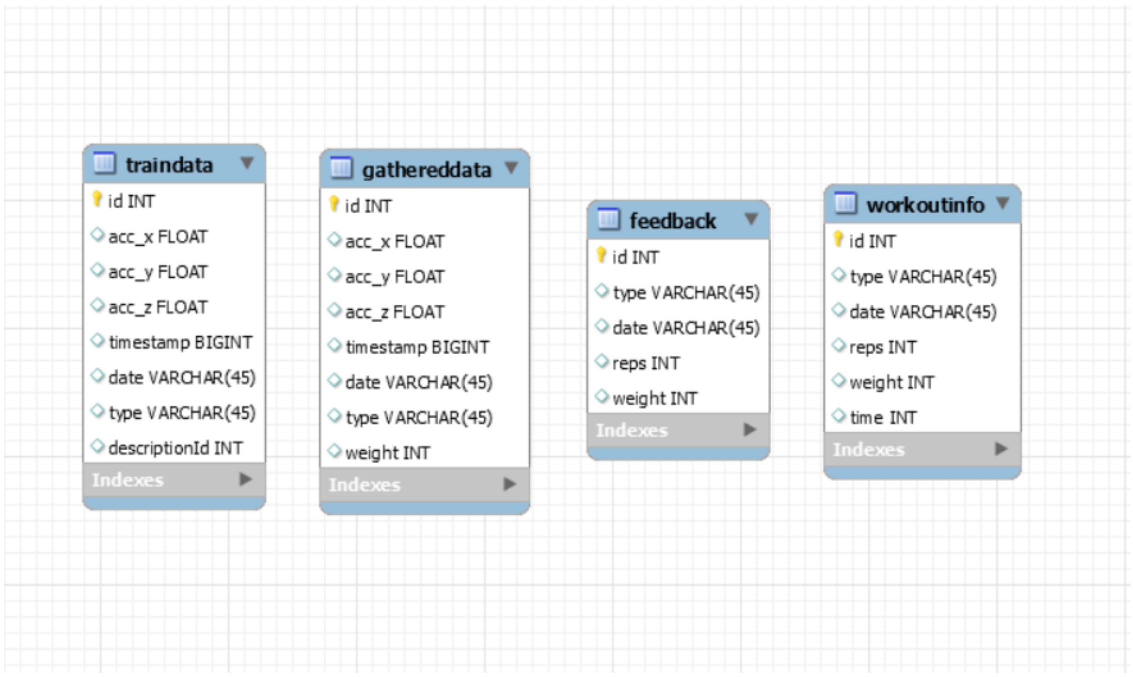


Figure 6: Database architecture

the value of the timestamp, at which those values were recorded, the timestamp representing the number of seconds which have passes since January 1st 1970; the date at which the values were recorded in the day/month/year hour:minute:second format; the type(name) of the performed exercise and lastly, an integer which represents the person which has performed the exercise. Exercises performed by different people will have different description Ids. As the name of the table suggests, this table was created with the scope of gathering as much data as possible in order to be able to visualise data of different exercises with different executions with the goal of creating and refining the function which calculates the number of repetitions. The more data, the more precise the output of the function. Concluding, the data from this table was used only for informational purpose when implementing the application and is of no use to the final product.

- **gathereddata**
  This table is very similar with the above described one. It contains the same accelerometer data, timestamp, date and type. The only difference is that instead of the description id, the used weight is stored. Just like before, this data comes from the user when performing an exercise. The main difference between this table and the previous one is that this is the

one used by the final application. When the user requests the number of repetitions of a certain exercises, data has to pass through this table. The concept is simple: data is recorded from the sensor and at the same time saved into this table. When the user finishes exercising, this data is requested by the python application in order to be able to compute the result and send it back to the user. At the end of the computation all the data saved in this table is deleted, in order to make room for the data from the next exercise and also in order to ease and fasten the query process. This process is repeated every time the user executes a certain exercise in order to help determine the repetition number. The timestamp, the date and weight do not have any importance when calculating the repetition number, they were added just for testing or for possible further implementations. The C application is the one which creates the rows of the table whereas the python one performs the queries for reading and deleting the data.

- **feedback**
  This is the table which occupies itself with storing information related to the personal best feature. It has 4 columns: the type of the exercise, the date on which the exercise was performed, the number of repetitions and the used weight. The implementation was build in such a way that only the maximum of each exercise of each day is saved. In other words, each entry of the table represents a personal record of the specified day. The reps do not count when calculating the personal best, only the used weight. All the operations performed on this table are done via the python app. It firstly creates and stores it when the user finished performing an exercise. When the user wants to use the personal best feature, the application simply retrieves all the data and sends it to the user, which will be then able to filter it by date.

- **workoutinfo**
  Like in the first situation, we have again two similar tables, having almost identical columns. The only difference between this table and the previously discussed one is that it also includes the time interval in seconds in which the exercise was executed. The feature related with this table is the one which provides the user a full specification of the exercises performed throughout a day. Like in the previous case, the python app performs all the queries. The retrieval query is performed in such a way that only the exercises corresponding with the requested date will be send to the android application to display to the user. By doing the filtering before at the python side, before sending the data, efficacy is improved by reducing congestion and time.

5.2 Formulas and mathematical models

5.3 Data structures

5.4 Algorithms

5.5 Implementation

Until now, the scope of the project was clearly explained. The design chapter presented in detail the role and goal of each component used for constructing this project. Now it's the time to decompose each component one by one and explain how everything was build. Each module and each designed class has a specific function which will be explained here. The implementation part will focus mostly on the three environments along with their programming languages: Android Studio with Kotlin, Visual Studio with C and PyCharm along with python. The main goal of this section is to explain how each functionality was achieved and how the interactions between the mentioned applications, along with the RabbitMq Server and the MySQL database were made. Snippets with class diagrams and how the classes were divided into packages will be provided in order to offer a better explanation. We will firstly start with the Android Studio application, which was used not for backend functionalities, but also for creating the frontend part of the application with which the user will be able to interact.

5.5.1 Android Studio

As previously mentioned, the android application provides not only implementation of backend methods, but also a graphical user interface. We will refer to the frontend part as the tools used in order to build the user interface and explain the role of each used component.

1.Frontend
When it comes to building the graphical user interface(GUI), Android Studio provides the programmers two alternatives. The first one is to create the design by using the design mode of the layout editor, in which components can be organised with simple drag and drop. The design mode provides a separate panel in which the attributes of each used component can be specified. Moreover, the editor provides the option to choose among different devices on which the created layout can be previewed. Among those options, the Round Smartwatch could be used, but I decided to create a custom one, with the exact specifications of the smartwatch used for the testing of the application. The dimension used were 454x454 pixel resolution with 372 PPI along with a 1.39 inch screen. The other option would be the code mode of the layout editor, in which the programmer can create the user interface by simply writing XML(extensible markup language) code. In this mode, the programmer has to specify the attribute of each component by hand. Every functionality can be achieved in both of the described modes, the first one being used mostly by new users which are

not so experienced with the IDE and the latter one being used by the more experienced user in order to build designs a little faster. The layout editor provides a third mode as well, which was mostly used for this project as well, namely the split mode, in which the design and the code mode are being combined in order to easily switch from one mode to another and to quickly preview in the design mode the changes made by the code mode.
Following, the resources package provides a place from where the user interface takes all its external added media and information. The figure below presents the four main packages and their contents. We have:

- Drawable: Here lie all the images used for the design. Most of them are the images which were assigned to the image button components. They are icons found on the internet of png format and most of them were resized to a specific 50x50 pixel resolution before being used.

- Layout: Here are the XML files which contain the code for each design layout. Each file corresponds to a certain page of the application. Furthermore, each view is strongly connected with a viewholder in the backend part in which the functionalities and the connections for the design components are build, for example the action which should start when pressing a button. Each of the layout components will be explained in more details in the following subsection.

- Raw: All the raw data used throughout the application design lies here. The only used files are the mp4 videos which contain the correct execution of each exercise and which are provided at the informational functionality of the app.

- Values: This package contains two xml files which help with the design of the components. The first one contains all the colors, in hexadecimal format, used for coloring the components with specific colors. The second one contains all the text used for naming certain buttons, labels or even for providing placeholder information. In order to use it, an id was assigned to each created string, which is set at the text attribute of the component.

Backend
The backend part mostly contains the functionality needed in order to make the design flow to perform as desired. It contains three main packages: Models, RabbitMq and Viewholders.
1.Models: This package contain two classes, namely SensorData and SensorDataTrain. Each of them are data class objects with the purpose of being used as data transfer objects(DAOs) when communicating with the C server application. They contain the sensor gathered information related to the execution of certain exercises. The main goal is to create this kind of objects with the captured data and save them into the MySQL database. Two data classes were created because one is used in order to save data for training and the other one is used for saving data to test the application. More of the use of this DAOs

when encountering them in the viewholder which deals with them.
2.RabbitMq: This package contains the RabbitServer class which is used in order to establish a communication with the CloudAMQP server. The required methods are provided by the com.rabbitmq.client package. In order to do such a thing, a ConnectionFactory is firstly created with the unique AMQP URI provided by the Advanced Queue Messaging Protocol Cloud server. Following, a connection is build with the newConnection method and on top of it, a channel is created with the createChannel method. Lastly, the queueDeclare method is used in order to create and bind queues to the created channel. Each queue receives an unique name at creation, which will have the role of the routing key when publishing to or consuming from a specific queue. This created channel will be used throughout the application for sending and receiving messages.
3.Viewholders: This package contains all the activities which are related with each created layout. This means that the number of classes will be equal with the number of different pages the application will have. In the AndroidManifest.xml all this viewholders are firstly declared as activities and among all of them one is being chosen as the main activity, which should be the one appearing when the application starts. In our case, the HomeView activity is chosen, meaning that the homeView layout will be the one firstly presented to the user at the start of the application. In the following each viewolder will be in detail explained, starting with the main activity. Each view will be presented back to back with the corresponding layout. An important mention would be that all the communication between the layouts and the activities is done through IDs. Each component, being it a button, a label, a text field, a layout, etc, has an unique ID which can be accessed by the viewholders with the help of findViewById method and giving the desired id as parameter. Instead of manually hardcoding the ID values when passing them as parameter, one can make use of the R class which contains all the IDs from the XML resource files. For example the code findViewById¡Button¿(R.id.testButton) will return an instance of the button component which has as id the string testButton.
HomeView: As previously mentioned, this view is the first thing the user sees when opening the application. This is done by binding this class with the homeView layout by using the setContentView view method and passing the id of the desired layout. Following we retrieve the result TextView which work as a label with the text "Please choose an exercise then press select". The content of the label is set by assigning a string to the text attribute of the TextView. Attributes can be accessed or changed by using a dot. Example: Component.Attribute. In order to create a drop down list containing all the supported exercises, we use the spinner component. An array of strings containing all the exercise names is firstly declared and then set to the adapter attribute of the spinner. A variable named choosenExercise was created in order to always represent the value of the selected exercise. This was achieved by updating the choosenExercise variable whenever a spinner value was selected, with the help of the onItemSelected function.
Following we have 4 different buttons, each of them making the transition to another activity layout. In order to make this transition, another activity has

to be started. First of all an intent variable has to be created specifying the name of the desired activity. Then this intent is given as parameter to the startActivity function. In order to pass messages, like variables for example, between activities, the putExtra function is used in order to add to an intent the desired variable. The method also takes as parameter a name which will be used in order to retrieve the desired variable in the new opened activity. This viewholder is the only one from which transitions to other layouts are made. The other activities do not need to interact one with another. Each activity starts when the user presses one of the four buttons displayed on the homeView layout. Let us take all the other activities one by one.

MainActivity: This activity is displayed to the user when he presses the "Select" button from the HomeView layout and a new intent is created. The message passed between the two activities represents the name of the chosen exercise. This value is firstly retrieved with the getExtra method and saved into the chosenExercise variable. This is the most complex class, as this is the place where the sensor data is computed and send to the message broker and the result is retrieved and displayed. In order to do such a thing, a SensorManager variable was created which records all the values of the 3-axis accelerometer incorporated into the smart device. By implementing the SensorEventListener interface, we make use of the onSensorChanged method which actives whenever one of the accelerometers changes their value. So every time a value changes, we create one of the two model DAOs described above, depending if we need training data or if we want to test or just run the application. Besides the acclerometer values, a timestamp variable is created, representing the time in milliseconds which have passed since 1 January 1970 until this execution point and a type variable which represents the type of the current exercise, received from the previous activity(HomeView). Depending on the type of data object created, among the above described variables it will also contain the used weight, for testing and running, or the description id of the exercise, for analysing data. Two different description id's represent two different users which performed the exercise. After the data transfer object is created, if the rabbitMq channel is not closed, the object will be published to the "andreiQueue" queue after converted to a Json and furthermore to an array of bytes. We will further see depending on what the connection channel is opened or closed.
The onclick function implements different type of actions when different buttons are pressed. We have two buttons present on the layout corresponding to the current activity. One should be pressed when starting the execution of the exercise whereas the other one should be pressed after finishing the execution. When the first one is pressed, the defaultExchangeAndQueue method from the above defined rabbitMQ class is called, which creates a connection and initialises a channel with his queues. When this happens, the objects created in the onSensorChanged method, containing data about the performed exercise, will start to be sent to the message broker, where will wait until received by a consumer. On the other side, when the user presses the button which indicates that the execution is over, several things happen. One of them is that the rabbitMq

channel is closed, in order to not send dummy data to the server. But just before closing the channel a request response type of procedure is approached. The system first sends to the "finishSending" queue of the rabbitMq channel the message "Done sending". This message traverses through the pipeline to the python application where the repetition count is computed. Because this computation might take a while, the system is put 2 seconds to wait. After two seconds, the application hopes that it will find the requested result in the "repCountResult" queue of the rabbitMq channel. Hopefully the result waits there to be consumed and the application takes and displays it to the user. Lastly, the connection is finally closed.
Concluding, this class gathers data from the sensor and combines it with other variabls in order to create a DAO. At each change of one of the sensors, this data will be send to be saved into the database, depending on which one from the two provided buttons was lastly pressed.

GraphsActivity: This activity is displayed to the user when he presses the icon representing a graph from the HomeView layout and a new intent is created. The message passed between the two activities represents the name of the chosen exercise. This value is firstly retrieved with the getExtra method and saved into the chosenExercise variable. This variable is being assigned to a textView representing the title of the current layout. This layout has the role of presenting to the user his personal records over a chosen period of time. A spinner is again used for displaying options from which the user can choose. This time the options represent the year and the month interval in which the personal records will be shown. In order to display the maximum used weight for a specific exercise in a day at a certain set, along with the number of repetitions for that specific execution set, an android chart was used. The set containing this charts has been added also added at the project dependencies and can be found at "com.github.PhilJay:MPAndroidChart:v2.2.4". Among all the different charts, the bar chart was chosen. In order to get the values which will be displayed, when the layout is firstly loaded, the setBarChatValues function is called. The way this function works is the following. It uses the request response procedure, described at receiving the number of repetitions, in order to receive the information needed to display. It firstly calls the defaultExchangeAndQueue method from the RabbitMq class in order to create a connection and a channel with its queues. The next step is to publish to the "finishSending" queue a message containing the name of the chosen exercise which was selected in the previous layout. After waiting for 2 seconds, the system consumes all the messages contained in the "repCountResult" queue, which should resemble the requested data, namely an array containing the date, the weight and the number of repetitions of all the performed exercises of the requested type. This data will be then split into three separate arrays, in order to be used for plotting them on the graph. Initially, all the data is shown displayed to the user. If he chooses he can use the spinners in order to filter them by date. Whenever a spinner value changes, being it the year or the month, the whole data is filtered and only the personal records of the specified date are being shown.

As we have seen, the request response requires from both applications to trust each other. The one responding trusts the other one that at some point it will send a message to the queue to which it listens to whereas the one requesting trusts the other one that it will provide the correct response to the queue it listens to.

WorkoutActivity
This activity is displayed to the user when he presses the "Workout Info" button from the HomeView layout and a new intent is created. The scope of this activity is to present to the user a proper feedback of the performed exercises for a specific day. This class has similar functionality with the previous one but different implementation. It also uses a filtering method in order to get specific information from the database, with the help of the message broker, and display it to the user. In order for this to happen, the user should firstly choose a date from the calendar. A DatePickerDialog was used in order to create the calendar interface. After the user successfully chooses a date, consisting of day, month and year, the following steps happen. Another request response approach is being used in order to obtain the workout information. After the rabbitMq connection along with the channel and its queues are created, the application publishes to the "finishSending" queue the chosen date. After waiting for two seconds, it consumes the message from the "repCountResult" queue which should contain a list with the performed exercises from that day along with more specific information. After successfully retrieving the data, which is contained into a single object, it is converted into separate arrays, each one containing at position i a specific attribute of the i-th exercise. After converting the data into the needed resources, it is displayed to the user as a list of exercises, each one containing the exercise name, the number of repetitions, the used weight, and the time spent for performing that exercise. If no data is found in the queue, the "No workout found on specified date" message is displayed. This process repeats itself every time the user picks a different date.
Another functionality provided by this activity is the one which allows the user to download to the external storage of his device the previewed workout. A download button was created to which this functionality was assigned. Moreover, the user has the possibility of choosing the name of the created file by entering it into the plain text component next to the button. The application takes this string from the component and creates a new file with its value, with the .txt extension, and writes into it the content of the displayed workout. The makeText function of the Toast class is finally used in order to display to the user the location of the saved file.

ExerciseInfoActivity
This activity is displayed to the user when he presses the icon representing a questionmark from the HomeView layout and a new intent is created. The message passed between the two activities represents the name of the chosen exercise. This value is firstly retrieved with the getExtra method and saved into the chosenExercise variable. This variable is being assigned to a textView

representing the title of the current layout. This page has the purpose of supplying the user with significant information about each supported exercise. The information is divided in two main parts:

1. **Description**
   This part provides information regarding the importance of the exercise along with the proper time to execute it and the exact targeted muscle group.
2. **Video**
   This part provides a video, saved into the raw directory of the resource package, in mp4 format, in which the proper execution of the exercise is presented.

A switch approach is being used in order to determine which description and which video to use, depending on the name of the chosen exercise.

An important mention would be that the "android.permission.INTERNET" command was used into the manifest xml file in order to allow connection to the internet. Without this command, a connection to the rabbitMq server cannot be established.
A last specification would be that the android applications requires the two other application to work as well in order to be able to send and receive data. One of the applications is the one created in the Visual Studio environment which will be explained next.

5.5.2 Visual Studio

This application uses the C language in order to build a path between the android application and the MySQL database. Regarding the class diagram, it has only three classes, from which 2 Data Transfer Object models and one main class where the main function is implemented. The two DAOs have the same attributes as the ones defined in the android application because they are meant to receive the information contained in them. As explained before, only one is used at a time, depending on the scope of the application, being it used for testing/running or for analysing.
The code for receiving and sending the information is as simple as it could be. First of all, two connections must be established, one with the RabbitMq server and one with the MySQL database. In order to such a thing, the IDE requires the following NuGet packages to be installed: RabbitMQ.Client, MySql.Data, MySQLConnection, MySqlConnector, System.Data.SqlClient. This packages are mostly libraries which provide different functionalities. After successfully installed the libraries, with their help, certain functions can be used to create the desired connections. The ConnectionFactory instance creates a factory with the URI provided by the CloudAMQP server as parameter. Furthermore the CreateConnection and CreateModel functions are used in order to create a channel and bind it to a connection. The QueueDeclare method binds the queue with

the name ”andreiQueue” to the channel. Furthermore, an infinite loop starts in which the system continuously reads the incoming queued messages and performs and insertion query into the MySQL database. In order to create the connection to the database, a connection string, containing the server, the user id, the password and the name of the desired database, was used. In addition to the data retrieved by the android application, a date variable is created for each received object representing the current execution date.
As well as for the android application, a working internet connection is required in order for the communication with the CloudAMQP server to work.

5.5.3 PyCharm

This application uses the python language in order to facilitate communication between different system components and in order to compute different results before sending them to be displayed to the user interface. This programming language was used because it allows the programmer to perform complex tasks with ease. Furthermore, establishing connections to the needed servers, was turned into simple tasks with the help of certain libraries. In order to optimise the readability and understanding, the application code was divided in different modules, each one concerning with functions regarding a common concept, the main module being the one where all of the others are put together in order to create the desired functionality. We will briefly describe the role of each module alongside with the implemented functions.

database.py
This module deals with two main things. One of them is establishing the connection to the MySQL Workbench database. The other scope of this module is to perform queries in order to insert, delete or retrieve specific data from the database tables. Because of the mysql library, a connection instance could be easily created just by specifying the database credentials and applying the connect method. Following we have different methods for CRUD operations on each of the existing tables of the connected database. These methods mostly look the same, the main difference being the executed queries. In order to perform a query, a cursor instance is retrieved by calling the cursor function on the database we are working with. Following, the execute method has to be called on the retrieved cursor with the query string given as parameter. If the operation is one which should retrieve some data from the database, this data can be stored into a variable by calling the fetchall method on the same cursor instance. The methods for deleting or retrieving all of the data are pretty basic and self explanatory. The most interesting functions are the one which have to retrieve specific data or to do mixed operations and to process the result before returning it to the module which call it.
One example of a function like this is the addFeedbackData method. It it used in order to update the table containing the daily personal records. The method takes as parameters the current database along with the number of repetitions, weight, date and the type of the current exercise. Because of the fact that only

the exercise with the most used weight per day must be stored in the database we first have to first retrieve the currently stored exercise with this properties, if there exists one. In order to do such a thing we perform a query in which we try to retrieve every exercise which was performed on the specific date and has the specified type, which should return one object or none. In order to keep the explained personal record property, three cases follow:

1. 1.If none is returned, the current exercise is simply added to the database.
2. 2.If the weight of the current exercise is less than the weight of the retrieved exercise, nothing is done.
3. 3.If the weight of the current exercise is more or equal than the weight of the retrieved exercise, a query is created in order to delete the retrieved exercise from the database and another query is performed in order to add the current one, which has a bigger weight.

Another type of more complicated functions, are the one which retrieve all the specific type of exercises and return them into an array touple of exercises. This type of exercises do not perform any queries, they just transform the input data into the desired output data. For example if the input is a list of all the performed exercises, the function first selects just the data which corresponds to a specific exercise type. Following there could be multiple executions of the same exercises, that's why the system has to divide it by sets. For example, if there were 3 different sets performed, of the same exercise, the output would be three touples, each one containing an array of the data which builds that exercise, at each index containing. The data returned as a touple would be the three accelerometer axis data, the milliseconds corresponding to each of that data and the type of the exercise. So if we take a look at the first touple at index i at each of the returned values, we should find the x, y and z value of the accelerometer, the millisecond at which the data was recorded and the type of the exercise for the i-th entry of the database for the current exercise set. Concluding, this function let us divide a long array of exercise data into different sets of the same exercise type.

rabbitMq.py
This module occupies with establishing a connection with the CloudAMQP server and with publishing or consuming messages from it. In order to do such a thing, only two functions are necessary. One in order to send and one in order to receive a message. Both functions have to firstly use the URI provided by the CloudAMQP management server interface in order to connect to the online server. Similarly to the other applications, this step requires the device on which the app is run to have a working internet connection in order to successfully connect to the online message broker server. The pika library is the one which provides the functionality which help with this step. The first method, which is used for sending a message, takes as parameter the message to be send, publishes the send result on the top of the "repCountResult" queue. On the other

side, the second function, which is used in order to consume a message, reads from the ”finishSending” queue until a message arrives, and returns it to the module where it was called.

charts.py
This module uses the pyplot collection of functions from the matplotlib library in order to plot the sensor gathered data on a graph. The x axis of the graph represent the time passed and the y axis the values of the accelerometer. We are using a 3-axis accelerometer, hence we have three y axis, which are plotted on the same graph, each one having a different color. Besides that, on one of the three axis, the minima and the maxima points are indicated as small points at the specific location. Lastly, the legend function is used in order to provide explanation about the meaning of each color used for drawing the different figures on the graph.

signalProcessing.py
This is the module where all the complex computation happen. The main scope is to successfully calculate the number of repetitions for a specific exercise type. Each different exercise has a similar implementation with slightly changes. The main function calls the appropriate method depending on the type of the exercise in order to find the number of repetitions. As function parameters, the accelerometer data along with the milliseconds, calculated in the database module, are passed. Because of the big similarity between the counting functions, we will describe only one, namely the one which counts the repetitions for the Bench Press exercise.
The function used for calculating the number of repetitions is the find peaks function from the signal collection of the scipy library. As explained in the section -to be updated-, the function is used in order to calculate the peaks of the provided signal between a specific height. The only difference between different exercise types is the height between which the peaks are searched, and the axis on which this peaks appear, which is given as parameter to the described function. For the bench press exercise, the peaks should be searched on the signal provided by the z axis of the accelerometer and between a height of -10 and 12. The height has the same measurement as the values from the y axis of the graph.

6 Experiments and Validation

6.1 Experimental Setup

6.1.1 Data sources for data used in experiments (real or simulated)

6.1.2 Metrics used to evaluate the solution

6.1.3 Experimental setup description

6.1.4 Algorithms and parameter values used in experiments

6.1.5 Classical Algorithms/Methods used in comparisons

6.2 Experimental Results

7 User Manual

7.1 Installation Prerequisites

7.2 User Interaction

8 Conclusion

References

[1] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15:31314–31338, 12 2015.

[2] Oresti Banos, Máté Tóth, Miguel Damas, Hector Pomares, and Ignacio Rojas. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors (Basel, Switzerland)*, 14:9995–10023, 06 2014.

[3] Sourav Bhattacharya and Nicholas Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. pages 1–6, 03 2016.

[4] Kilian Forster, Pascal Brem, Daniel Roggen, and Gerhard Troster. Evolving discriminative features robust to sensor displacement for activity recognition in body area sensor networks. pages 43 – 48, 01 2010.

[5] Heli Koskimäki and Pekka Siirtola. Recognizing gym exercises using acceleration data from wearable sensors. pages 321–328, 12 2014.

[6] Marcel Luzar, Andrzej Czajkowski, Rafał Kasperowicz, Maciej Rot, Mateusz Semegen, and Józef Korbicz. Sensor-based weightlifting workout assisting system design and its practical implementation. *Journal of Sensors*, 2019:1–14, 12 2019.

[7] Mohd Halim Mohd Noor, Zoran Salcic, and Kevin Wang. Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer. *Pervasive and Mobile Computing*, 38, 09 2016.

[8] Dan Morris, T. Saponas, Andrew Guillory, and Ilya Kelner. Recofit: Using a wearable sensor to find, recognize, and count repetitive exercises. *Conference on Human Factors in Computing Systems - Proceedings*, 04 2014.

[9] Bobak Mortazavi, Mohammad Pourhomayoun, Gabriel Alsheikh, Nabil Alshurafa, Sunghoon Lee, and Majid Sarrafzadeh. Determining the single best axis for exercise repetition recognition and counting on smartwatches. pages 33–38, 06 2014.

[10] Igor Pernek, Gregorij Kurillo, Gregor Stiglic, and Ruzena Bajcsy. Recognizing the intensity of strength training exercises with wearable sensors. *Journal of biomedical informatics*, 58, 10 2015.

[11] Jun Qi, Po Yang, Martin Hanneghan, Stephen Tang, and Bo Zhou. A hybrid hierarchical framework for gym physical activity recognition and measurement using wearable sensors. *IEEE Internet of Things Journal*, PP, 05 2018.

[12] Wen qi, Hang Su, Chenguang Yang, Giancarlo Ferrigno, Elena De Momi, and Andrea Aliverti. A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone. *Sensors*, 19, 08 2019.

[13] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. myhealthassistant: A phone-based body sensor network that captures the wearertextquoterights exercises throughout the day. 01 2011.

[14] Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16:426, 03 2016.

[15] Andrea Soro, Gino Brunner, Simon Tanner, and Roger Wattenhofer. Recognition and repetition counting for complex physical exercises with deep learning. *Sensors*, 19:714, 02 2019.

[16] Terry Um, Vahid Babakeshizadeh, and Dana Kulic. Exercise motion classification from large-scale wearable sensor data using convolutional neural networks. pages 2385–2390, 09 2017.

[17] Eduardo Velloso, Andreas Bulling, Hans Gellersen, Wallace Ugulino, and Hugo Fuks. Qualitative activity recognition of weight lifting exercises. *ACM International Conference Proceeding Series*, 03 2013.

[18] Zheng-An Zhu, Yun-Chung Lu, Chih-Hsiang You, and Chen-Kuo Chiang. Deep learning for sensor-based rehabilitation exercise recognition and evaluation. *Sensors*, 19:887, 02 2019.

[19] Zhendong Zhuang and Yang Xue. Sport-related human activity detection and recognition using a smartwatch. *Sensors*, 19:5001, 11 2019.