POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master of Science in ICT for Smart Societies

Master Degree Thesis

Design of an Application for Patient Monitoring after Coronary Angioplasty

Supervisors
prof.sa Monica Visintin
prof. Marco Giuseppe Ajmone Marsan

Candidate
Marco Cerutti

April 2018

Acknowledgements

Firstly, I would like to thank my supervisor Prof. Monica Visintin for the support and the advices she gave me during my thesis work, guiding me along this path.

A big thank you also to Dr. Matteo Bianco and Dr. Fabrizio D'Ascenzo who follow me with their medical knowledge and expertise.

Thanks to my parents Maurilio and Laura, and to my aunt Raffaella, who have always supported and encouraged me, and who take care of me and never let me lack for anything.

And thank you Serena, who have always been by my side in these years, sharing with me not only the experience of the Politecnico, but also the everyday life, with all the troubles and the joyousness of that.

Innanzitutto, vorrei ringraziare la mia relatrice, la Professoressa Monica Visintin, per il supporto e i consigli che mi ha dato durante il mio lavoro di tesi, guidandomi in questo percorso.

Un grosso grazie anche al Dottor Matteo Bianco e al Dottor Fabrizio D'Ascenzo, che mi hanno seguito con le loro conoscenze mediche e la loro esperienza.

Grazie ai miei genitori Maurilio e Laura, e a mia zia Raffaella, che mi hanno sempre supportato e incoraggiato, e che si sono presi cura di me senza mai farmi mancare nulla.

Infine, grazie Serena, che sei sempre stata al mio fianco in questi anni, condividendo con me non solo l'esperienza del Politecnico ma anche la vita di tutti i giorni, con tutte le preoccupazioni ma soprattutto le gioie che ne derivano.

Summary

Filo Application stems from the need to improve cardiopathic patients' follow-up after leaving the hospital. The application, composed by a **Web** interface for the cardiologist and a smartphone (**Android**) interface for patient, wants to be a *direct link* of communication between them. The system was designed following the requirements and suggestions of cardiologists of Molinette, the main hospital in Torino.

The physician will be able to follow the patient's status in real time, thanks to the data he/she will upload through the application, and will be warned in case of abnormal values or in case of some kind of events, as bleedings or hospitalizations.

The patient will also be able to use the application as therapy calendar, and will be notified from the app when to take which medicine.

The web platform is developed using *HTML* and *Javascript* languages, while the Android one using *XML* and a *Java-based* language and the *Android Studio IDE* (Integrated Development Environment).

The patient is registered into the platform by the physician, trough the web application. Registrations and successive authentications are managed thanks to **Firebase Authentication Service**, interfaced both with the web and the Android app. Users are registered with their email, so their uniqueness is assured. Authentication requires a password that patients can choose and change any time. Once patients have ended their follow-up period, it is possible to delete all their data, if requested.

Information about patients is saved into the **Firebase Realtime Database**, and is retrievable only by its owner and by the physician. Cardiologists can use also the **Firebase Cloud Storage** to save some documents about each patient, as ECG files and examination prospectuses.

With a correct and sincere utilization of the app from the patient, the physician will have more information about the patient's situation to help him/her taking clinical decisions, while the patient will be helped in the first steps with the therapy and changes in the lifestyle.

Contents

List of Figures

Listings

Chapter 1

Filo Application

1.1 Background

Heart diseases represent the principal cause of death in the western world. *Myocardial Infarction* and *Ischemic Heart Disease* represent the most frequent cardiovascular diseases.

The gold standard of care for coronary diseases is the **Coronary Angioplasty** that, through a micro-invasive surgery, allows to treat the disease in a safe and effective way. It consists in a dilation of an obstructed blood vessel by means of a balloon catheter, introduced into the vessel and inflated at the obstruction.

In Italy, every year 140,000 angioplasties are performed. The cost of materials is over €2,000 for each procedure, and the overall cost is in average higher than €5,000. About 35% of patients who have undergone an angioplasty surgery have a relapse within 2 years, in most cases due to a low compliance with the therapy or because they do not change their lifestyle.

Secondary Prevention, i.e. all measures taken to avoid disease relapses, is therefore an absolute priority not only from a medical point of view, but also for social and economic reasons.

1.2 The Project

This project is born to build an innovative welfare model for all those patients suffering of myocardial infarction and/or have undergone a coronary angioplasty. These patients, especially in the phase immediately after the acute event, are really fragile from a clinical point of view and they need medical assistance to avoid relapses. Unfortunately, after the release from the hospital, assistance is limited to occasional meetings with the family doctor and few cardiovascular examinations within 3/6 months or a year.

The adoption of behavioural rules as interruption of smoking, a good diet and the control of blood pressure is given to the autonomous management of the patient. He/she is not always able to act those measures, that often mean radical changes in his/her lifestyle, frequent examinations and the management of complex therapies. Patients not adopting those rules often experience a new cardiovascular event.

Filo starts out from the will to create a direct link between cardiologist and patient after the hospital discharge, producing also a great quantity of clinical data allowing to optimize the secondary prevention reducing the number of re-infarctions. It comes from the necessity to keep under control patients in the out-of-hospital environment and from the consciousness that current assistance is far to be optimal for all the patients.

Filo aims to make better health conditions of cardiopathic patients, optimizing the assistance and the secondary follow-up, to prevent new pathologic events.

1.3 The Service

Filo Application is a service addressed to cardiologists and cardiopathic patients, intended to optimize the domestic follow-up through a **Smartphone Application** and a **Web Portal**.

In case the patient did not have a smartphone or the ability to use it, his/her **caregiver** will guide him/her during the follow-up using **Filo Application**. The caregiver has a fundamental role in the at-home assistance of a patient, especially for elderly ones: the role is usually covered by someone younger, like a child, a nephew or a in-home nurse who usually deals with the therapy and manage all other aspects of the disease. The intimate knowledge of the patient and of his/her needs and the lower age make the caregiver a very good intermediary for a service as **Filo**.

The identification of the subject that will use the service will take place at the moment of the release from the hospital. The patient, or the caregiver, will receive from the hospital staff all the information needed to use the service, as login and data collection modality, or the recover password procedure. In that occasion, the physician will insert in the web portal all clinical data necessary for a correct use of the application.

The patient, at home, will start an exchange of information with the physician, who will be able to offer a more personalized support, based on patient's needs.

1.3.1 Patient Interactions

The patient will use **Filo** with 4 main functions:

- A source of positive reinforcement for the adoption of behavioural rules useful to secondary prevention (interruption of smoke, continued physical activity, etc.).
- A reminder for drug therapy assumption (improvement of therapy compliance).
- Anthropometric data collection (pressure, weight, glycaemia, etc.).
- Clinical data collection (bleedings, hospitalizations, other medical examinations, etc.).

The goal is to make the patient more aware about secondary prevention and the correct therapy compliance. In fact, errors in drug assumption represent even today one of the principal causes of new pathologic events. This service should reduce those errors.

All collected data will be saved on a cloud database (**Firebase Realtime Database**), and they will form temporal series which the physician will be able to see on the web platform.

1.3.2 Physician Interactions

The physician will use **Filo** with 4 main purposes:

- To obtain necessary information to draw a profile of the patient current health status and of his compliance with the therapy.
- To have a solid source of data on which to base clinical decisions.
- To receive earlier alert about potential risk situation for the patient.
- To benefit from data useful for clinical research.

The cardiologist, at the moment of the patient release from the hospital, will be able to load all clinical information, performed examinations and complete history about him/her. He/she will have always under control patients anthropometric parameters, thanks to a series of graphs visible in the web platform.

1.3.3 Overall Description

This section describes the overall functioning of **Filo** platform. It is possible to identify three main entities that interact with each other:

- The *Physician* (through the **Web** application)
- *Firebase* (authentication provider, database and storage)
- The *Patient* (through the **Android** application)

First of all, the *Physician* registers the *Patient* on *Firebase*, through the **Web** application, using the *Patient*'s **email**. In this way, the *Patient* becomes a **user** of the platform. He/she receives a **User Id**, a unique identifier used by the platform, and a dummy **password**. Immediately after the registration, *Firebase* sends to the *Patient* an email message to reset his/her own password. From this moment, the *Patient* is able to log in from the **Android** application and can start using it (Figure 1.1[1]).

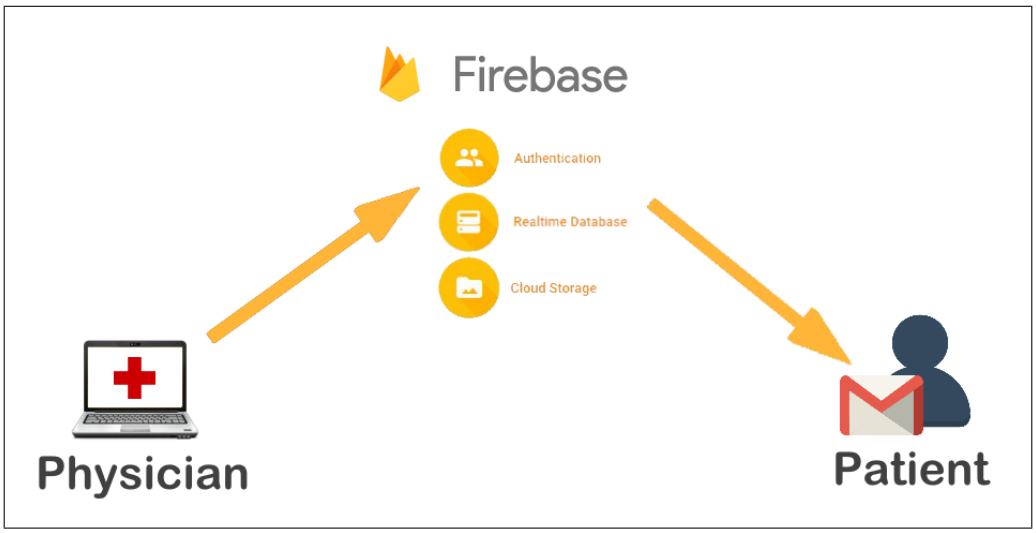

Figure 1.1: *Patient* Registration

From now on, the *Patient* can access the **Firebase Realtime Database**, will be able to self-manage future **password changes** and to receive **notifications** from the app.

[1]Used icons are copyright free [1]

Using the **Android** application, the *Patient* will be able to retrieve different pieces of information about his/her clinical history, and will be able to share with the *Physician* different data to describe his/her real-time situation (Figure 1.2). **Firebase** sends data over an **HTTPS** connection, so it autonomously manage their encryption in the transit phase.

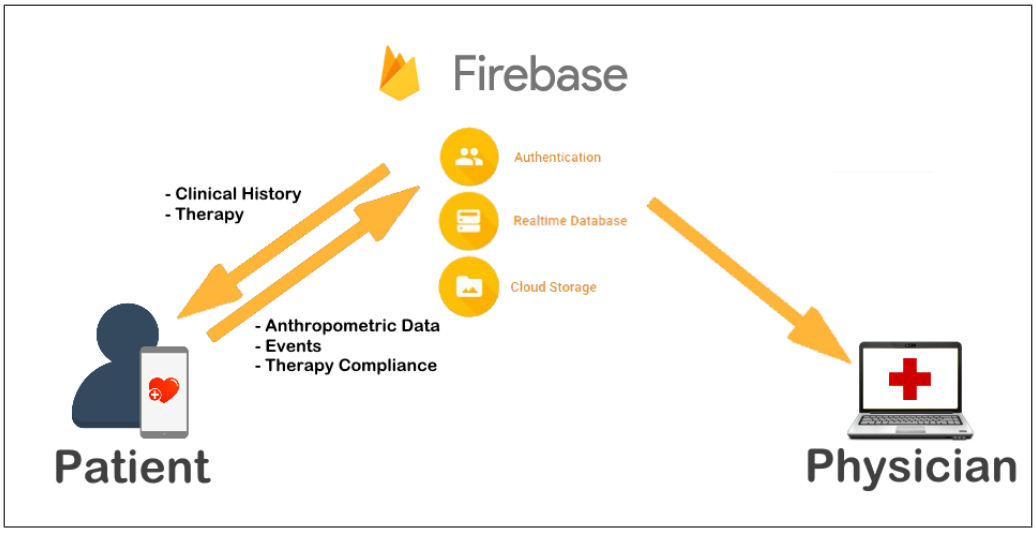

Figure 1.2: Data Communication

The *Patient* will be able to access his/her **Clinical History** and the **Therapy** he/she has to follow, information uploaded by the *Physician*. On the other hand, the *Patient* will be able to upload information as **Anthropometric Data** (pressure, heart rate, glycaemia, etc.) and **Events** (bleedings, therapy changes, other physical examinations, etc.). He/she will also being asked to communicate data about the **Therapy Compliance** day-by-day.

The *Physician* will be able to retrieve all these data from the **Web** application every time, and will be notified in the **Home Page** if a *Patient* has inserted an event or some **alarming anthropometric data**. In fact, the **Web** application will show these *Patients* differently from other ones. The *Physician* will then choose if to cancel the alert or not.

The **Web** application will show data in a user friendly way, plotting them based on the type of data (numerical, boolean, etc.). A specific page is devoted to do this for each *Patient*.

All these functionalities allow the *Patient* to be closer to his/her cardiologist during the follow-up.

Another part of **Filo** is instead only used by the *Physician*. He will be able to access not only the **Realtime Database**, but also the **Cloud Storage**, where to save *Patients*' **ECG** files and **Visit Prospectuses** (Figure 1.3).

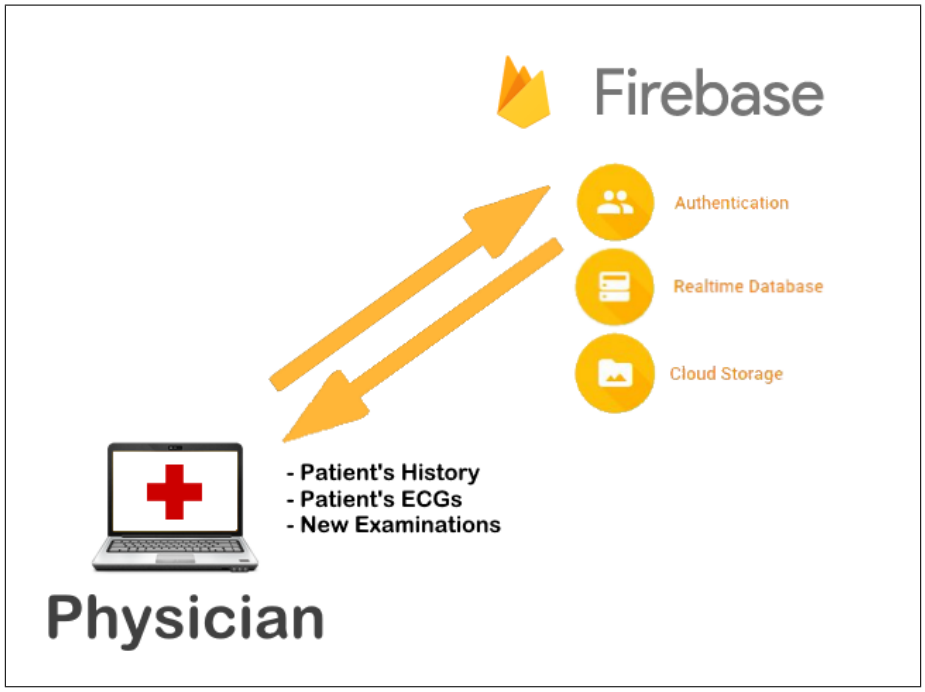

Figure 1.3: *Physician* Functionalities

During examinations, the *Physiscian* can load files in the **ECG directory** of the *Patient*, choosing them from the ones saved on the used computer. He can choose textual files, images, pdfs, spreadsheets, etc., and can load more than one file at the same time. Then, at the end of the examination, the prospectus will be automatically saved in the **visit directory**. Each file is saved in a sub directory named as the date of the examination day.

Files will be always retrievable and downloadable from a specific page, and the ones in a format the browser can open will be also viewable without downloading them, as for pdf files.

Patient history, will be retrievable from the database in the **Patient Data Page** of the web application, and will be editable during each examination.

Chapter 2

Firebase

Firebase is a mobile and web application development platform produced by **Firebase, Inc.** in 2011, then acquired by **Google** in 2014 [2]. It offers a series of services very helpful in developing web and mobile applications, such as **Firebase Authentication**, the **Realtime Database** and the **Firebase Storage**. These services can be easily integrated in the applications using API key for web and SHA-1 key for Android.

2.1 Firebase Auth

Firebase offers a service to manage user authentication, using different paradigms. Besides *Email-Password* paradigm, Firebase authentication provide the possibility to directly authenticate a user using his/her **Google**, **Twitter**, **Facebook** or **GitHub** profiles and more [3].

Nevertheless, for **Filo** application only the *Email-Password* paradigm has been chosen, because being an app with medical related purpose, it was preferred not to link it with social profiles.

Firebase also manages cases in which the user tries to create different accounts using the same e-mail address with different authentication providers. The application can choose if to allow or to avoid it. Moreover, it automatically limits the number of new anonymous or email-based registrations coming from the same IP address, to avoid possible violations.

The **Authentication** service provides also some models to build Email and SMS for communications to users. For example, a model to verify the email used to create an account or to change the user password in case he forgets it.

After the creation, each user automatically receives a *Uid*, a unique identifier not related to the user information or to the authentication provider. That identifier will be used in the database for two main reasons: first, database's rules can manage *Uids* to allow access to a particular branch only to the user that owns it, second, to make the database anonymous.

2.2 Firebase Realtime Database

Firebase Realtime Database is a **NoSQL** ("non SQL", meaning not based on a Structured Query Language) cloud database. Data is synced in real time and stay available also if the application goes offline. Data are stored in **JSON** (JavaScript Object Notation) format, which is characterized by a tree structure [4]. A more detailed description of JSON can be found at the end of this chapter (2.3.7).

It is a *Real-time* database, so instead of classical *HTTP* requests it uses data synchronization. It means that every time data changes, the connected devices receives the update within milliseconds. When the application needs to read a database snapshot, the *onDataChange()* method has to be used. It is triggered once when created and again every time data changes, so information inside application is always updated [5].

Firebase applications remain responsive also when offline, because the **Firebase Realtime Database SDK** (Software Development Kit) persists data to local disk. Then, once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.

Firebase offers also the possibility to custom a set of rules to access the database. Basic rules require users to be logged-in to access data, but it is possible to redefine them using the **Firebase Realtime Database Security Rules**, a flexible expression-based rules language. This allows to define more strictly who and how each database branch can be accessed.

2.2.1 Realtime Database Limits

Firebase Realtime Database is able to manage till 100,000 *Simultaneous Connections*, where a simultaneous connection is one mobile device, browser or server app connected to the database. It is important to understand that it is not the maximum number of users, but the amount of users connected at the same time. To make an example, applications with *10 million* of users have generally fewer than 100 thousands of simultaneous users. However, it is possible to go beyond this limit implementing multiple databases.

The number of responses each database is able to sent is about 100,000/*second*, it means it is the maximum number of contemporaneous reading actions on the database.

The maximum depth of the tree is 32 child nodes, with keys no longer than 768 Bytes with no new lines, no special characters (. , $, # , [,] , /) or any *ASCII* control character (0x00 - 0x1F and 0x7F).

The maximum size of a string is of *10 MB*, with *UTF-8* encoding. UTF-8 uses 1 to 4 bytes to represent each character, but only 1 byte is requested to each one of the 128 characters of ASCII alphabet [6], so each string can be 5/10 *million* characters long.

A single read response, i.e. the amount of data downloaded from the database at a single location, should be less than *256 MB*. In case of need for larger response, Firebase offers other options, but it is not the case of interest of **Filo**.

The size of a single write request should be less than *256 MB* from *REST API* and *16 MB* from the *SDKs* [7]. Again, **Filo Application** perfectly fits these limits.

The amount of memory that can be occupied in total by the entire database varies if it is used as free or under payment. The free version used for this thesis work allows for *1 GB* of memory [8].

2.3 Filo Application Database Structure

Filo Application Database is made of 5 principal branches:

- patientsUid
- patients
- data
- events
- physicianUid

Each branch follows proper rules to be read and written, discussed at the end of this section (2.3.6). Each branch will be illustrated and described in the following paragraphs.

2.3.1 PatientsUid Branch

PatientsUid is the branch used to list all patients in the **Web Application**, the one used by physicians. The structure of *patientsUid* branch can be seen in Figure 2.1.

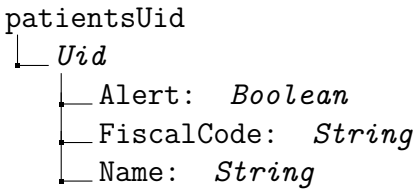

Figure 2.1: Structure of *patientsUid* Branch

Name and *FiscalCode* are used as identifiers, while the *Alert* field is used to diversify patients in the **Home Page** of the web application (section 3.3). Patients with the *Alert* field *true* are shown in a different list with a different colour.

2.3.2 Patients Branch

The information about patient is stored in the *patients* branch. Figure 2.2 shows the first part of the list of parameters, the other is given in Figure 2.3.

```
patients
└── Uid
    ├── PersonalData
    │   ├── Birthplace: String
    │   ├── DateOfBirth: yyyy-mm-dd date as String
    │   ├── Email: String
    │   ├── FiscalCode: String
    │   ├── Gender: String
    │   ├── Name: String
    │   ├── ResidenceCity: String
    │   ├── ResidenceRoad: String
    │   ├── Surname: String
    │   └── Telephone: String
    ├── History
    │   ├── Cardiovascular: String
    │   ├── Advices: String
    │   ├── Ecg: String
    │   ├── General: String
    │   └── Today: String
    ├── Angioplasties
    │   └── index (from 0)
    │       ├── date: yyyy-mm-dd date as String
    │       ├── revasc: String
    │       ├── status: String
    │       └── vessel: String
    ├── EcoURL
    │   └── File Name: yyyy-mm-dd date as String
    └── OldExaminationURL
        └── File Name: yyyy-mm-dd date as String
```

Figure 2.2: Structure of *patients* Branch (1)

Personal Data branch contains all personal information about the patient.

In the *History* branch data about last examination are saved. This information is read at the next visit to the cardiologist and will be overwritten at the end of it.

The *Angioplasties* branch keeps trace of all patient's ill and treated vessels, with the date of registration or treatment and the type of revascularization, if performed.

EcgURL and *OldExaminationURL* branches contains the names of all files stored in the **Firebase Storage** related to the patient. These branches are used to retrieve those files.

```
patients
└── Uid
    ├── RiskFactors
    │   ├── Allergies: String
    │   ├── Diabetic: Boolean as String
    │   ├── Dyslipidemia: Boolean as String
    │   ├── Familiarity: Boolean as String
    │   ├── HIVinTherapy: Boolean as String
    │   ├── Hypertension: Boolean as String
    │   ├── KidneyFailure: Boolean as String
    │   ├── PreviousChemio: Boolean as String
    │   ├── PreviousRadio: Boolean as String
    │   └── Smoker: Boolean as String
    ├── Status
    │   ├── Weight: numerical value as String
    │   ├── Height: numerical value as String
    │   ├── BMI: numerical value as String
    │   ├── LVEF: numerical value as String
    │   ├── STEMI: Boolean as String
    │   └── NSTEMIACS: Boolean as String
    └── Therapies
        └── index (from 0)
            ├── drug: String
            ├── assumptionDays: String
            ├── assumptionHours: String
            ├── infoMeal: String
            └── end: String
```

Figure 2.3: Structure of *patients* Branch (2)

Risk Factors branch contains the information about risk factors affecting the patient. Each key corresponds to a factor, while values can take two possible values as *String*, *YES* or *NO*. The only exception is the field *Allergies*, that can have free text as a value.

Status branch stores the patient status at the moment of the last examination, in terms of physical condition (*Weight*, *Height* and *BMI*), heart condition (*LVEF*) and previous performed heart surgeries (*STEMI* and *NSTEMIACS*).

Therapies branch keeps trace of all drugs the patient has to take and when, with extra information about the assumption related to the meals and its end.

2.3.3 Data Branch

This is the part of the database where patients save data through the **Android Application**. Each data is saved in the proper branch with the date of the selected day. The structure of this branch can be seen in Figure 2.4.

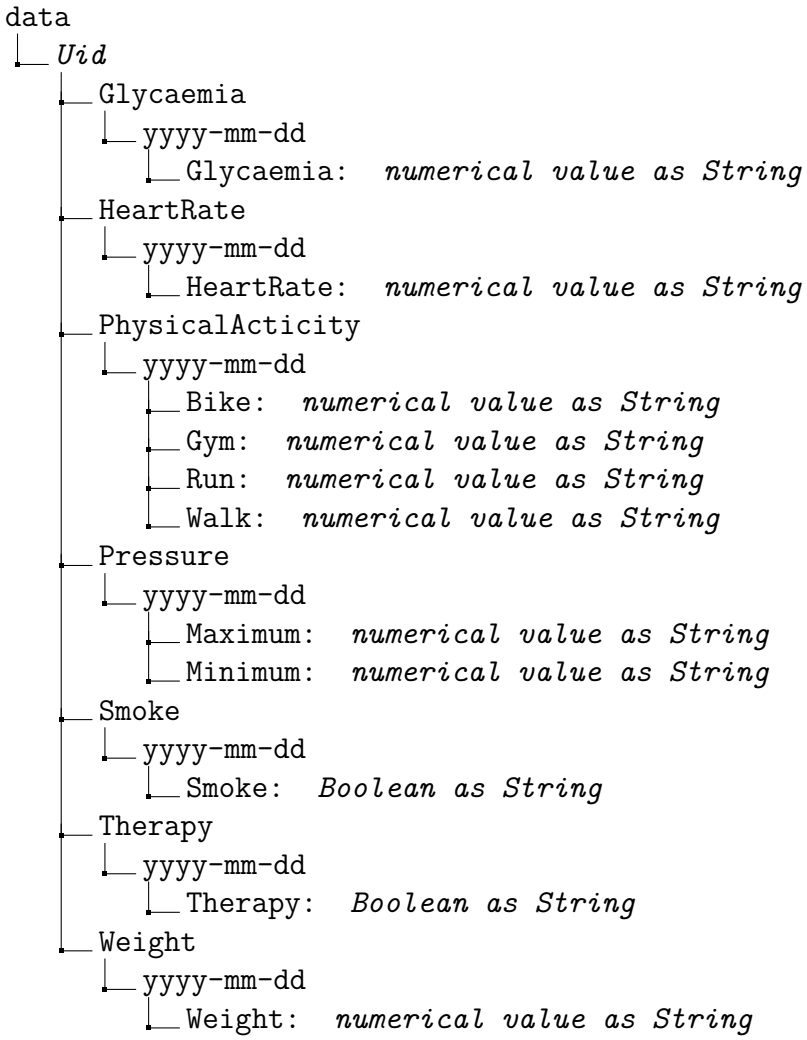

Figure 2.4: Structure of *data* Branch

Pressure branch has always two fields, corresponding to Systolic (maximum) and Diastolic (minimum) pressure.

PhysicalActivity branch instead might not have all the values. It is normal for a person not to do all types of physical activities every day, so the system automatically manages the missing fields.

Smoke and *Therapy* can get two possible values, *YES* and *NO*, while *Glycaemia* and *HeartRate* take numerical values, but saved as *Strings*.

2.3.4 Events Branch

This branch is populated with information related to events that may happen to patients during the follow-up, through the devoted activity of the **Mobile Application** (section 4.6). Events can be of different types, and each one has a different field. The structure of *events* branch is shown in Figure 2.5.

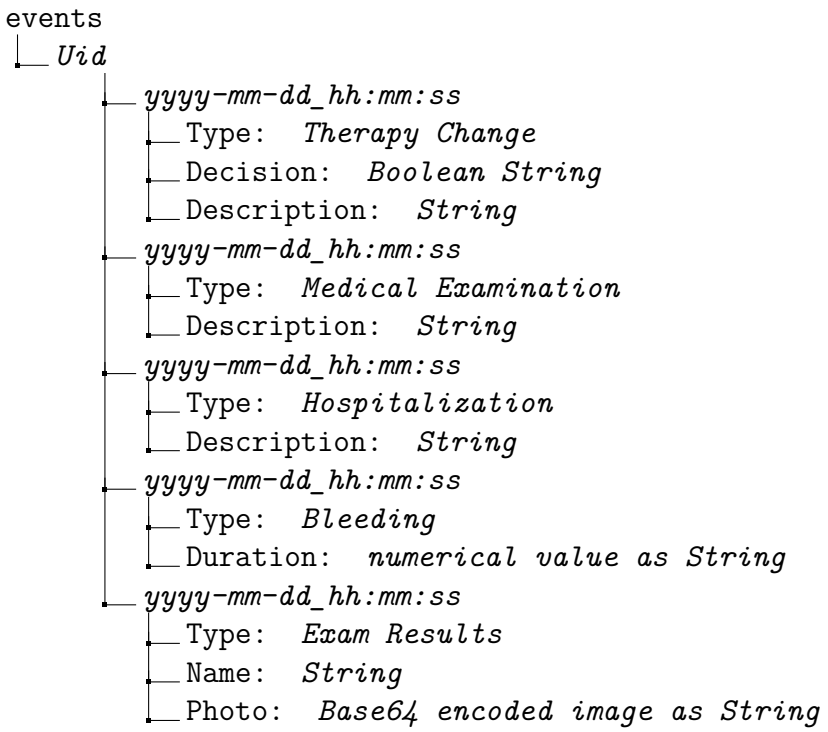

Figure 2.5: Structure of *events* Branch

Events such as bleeding can occur several times in a day, so the time key here specifies also the hour in addition to date. In *Therapy Change* event, *Decision* field can take two values as *String*, *Personal* or *Physician*, to notify who decide the therapy change.

In *Medical Examination* and *Hospitalization* events, *Description* fields can be used to describe type and reason for those events.

Bleeding event has a *Duration* field to specify how long the bleeding has lasted.

The *Exam Result* event is the most complex. It allows the patient to save into the database an image that can be retrieved and decoded by the physicians through the **Web Application**.

2.3.5 PhysicianUid Branch

This branch can be read only from physician's accounts. It is basically used to keep trace of which *Uids* are related to physicians, and to control access permission in the **Web Application**, that only them can use.

2.3.6 Filo Application Database Rules

Code in Listing 2.1 defines the rule structure of Filo Database. It defines who can access and how each database branch can be accessed. Employing users *Uids* as keys into the database allows to define the access to a specific branch only to the user who owns it, thanks to the *$uid* key of the **Firebase Realtime Database Security Rules** language.

As described in the rules, complete access to the *patientsUid* branch is given when the authenticated *Uid* corresponds to the physician one. So each cardiologist can read every patient's branch, while each patient can read only the one corresponding to its own *Uid*.

The patient branch is the one modified through the **Web Application**, so only physicians have writing access, but information inside has to be visible on the **Android Application**, so each patient can read its own.

Data and *events* branches can be read and written both by physicians and patients, obviously each patient its own, while *physicianUid* branch can be accessed only by physicians. This last one is used in the **Web Application** to check if the user has physician permission.

2.3.7 JSON Format

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is a text format completely language independent but that uses conventions familiar to the C-family of languages. These properties make it an ideal data-interchange language.

A **JSON** is built on two structures:

- A collection of Key/Value pairs
- An ordered list of values

These two data structures are universally known, and all modern programming languages support them in one form or another. This makes **JSON** a data structure easily understandable by different languages [9].

```
{
 "rules": {
  "patientsUid":{
   ".write": "auth.uid===physicianUid",
   ".read": "auth.uid===physicianUid",
   "$uid":{
    ".write": "$uid===auth.uid || auth.uid===physicianUid",
    ".read": "$uid===auth.uid || auth.uid===physicianUid"
   }
  },
  "patients":{
   "$uid":{
    ".write": "auth.uid===physicianUid",
    ".read": "$uid===auth.uid || auth.uid===physicianUid"
   }
  },
  "data":{
   "$uid":{
    ".write": "$uid===auth.uid || auth.uid===physicianUid",
    ".read": "$uid===auth.uid || auth.uid===physicianUid"
   }
  },
  "physicianUid":{
   ".write": "auth.uid===physicianUid",
   ".read": "auth.uid===physicianUid"
  },
  "events":{
   "$uid":{
    ".write": "$uid===auth.uid || auth.uid===physicianUid",
    ".read": "$uid===auth.uid || auth.uid===physicianUid"
   }
  }
 }
}
```

Listing 2.1: Filo Database Access Rules

2.4 Firebase Storage

Firebase Storage is a *Cloud Storage* easily accessible by applications interfaced with Firebase. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads regardless of network quality. Uploads and downloads are robust, meaning they restart where they stopped, saving users time and bandwidth.

Firebase SDKs for Cloud Storage integrated with Firebase Authentication give it a very strong security so files access permissions can be managed using filename, size, content type, and other metadata [10].

Similar to the *Realtime Database*, **Firebase Security Rules for Cloud Storage** makes it easy to define access permission to the storage. The basic rule is that users have to be authenticated to access files but it is easy to define specific paths that can be read or written only by specific users [11].

The amount of memory that can be occupied by files in the Cloud Storage depends on the chosen pricing plan for the project, and it is not related to the amount of memory used by the **Realtime Database**. The free plan used for this thesis allows to store until *5 GB* [8].

2.5 Filo Application Cloud Storage Structure

Filo Cloud Storage is used by physicians to save documents about patients. It contains mainly *pdf* files, such as the one produced at the end of each examination. Only physicians have access to the storage.

The structure is really simple and is shown in Figure 2.6.

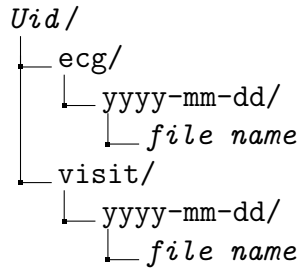

Figure 2.6: Filo Cloud Storage Structure

The two directories contain *ECG* files and hospital discharge documents saved by physicians, as discussed in section 3.5.

Code in Listing 2.2 defines the access rule for the storage:

```
1 service firebase.storage {
2   match /b/{bucket}/o {
3     match /{userId}/{allPaths=**} {
4       allow read, write: if request.auth.uid == phisicianUid;
5     }
6   }
7 }
```

Listing 2.2: Filo Storage Access Rules

Each principal directory is named with the patient's *userId*, then all paths starting from it can be read and written only if the authenticated *Uid* corresponds to the one of the physician.

Chapter 3

The Web Platform

3.1 Introduction

The web application of **Filo** platform is the interface used by the physician to access all patients data. Once the user is logged, if it has physician permission on **Firebase Database**, he/she can see a list of all patients and add new ones.

The registration requires to insert personal data of new patients and gives the possibility to perform the first medical examination. The application allows to insert several data and shows the list of all events communicated by the patient through the **Android Application** at the end of the page.

It also produces hospital discharge documents, saving them into the **Firebase Storage** and making possible the download.

From the list of patients, physicians can open the patient's page by clicking his/her name, where all information about the patient's status are visible. They can also visualize different charts realized using data given by patients.

The user interfaces of the application were written in Italian, because the expected population target is Italian, however, it will be simple to add other languages, and generate for example an English interface.

3.2 The Login Page

The Login Page is the starting page of the physician part of the **Filo Platform**. Any attempt to access, without a previous login, automatically redirects to this page. User's *Email* and *Password* are required to perform login. Once data have been inserted, it is sufficient to click *Enter* button on keyboard, both from password and from email input field, or it is possible to click the *Login* button ("Accedi"). Then *Database* and *Logout* buttons become clickable, and it is no more possible to interact with the login one. Firebase sign-in status is also visible right below the buttons. Once the login has been performed, it is necessary to click *Database* button to access it.

To avoid user's mistakes in typing password, a *Caps Lock* detector is active in the *Password* field, to warn him/her if it is enabled.

Figure 3.1 shows the **Login Page** in the unauthenticated state.

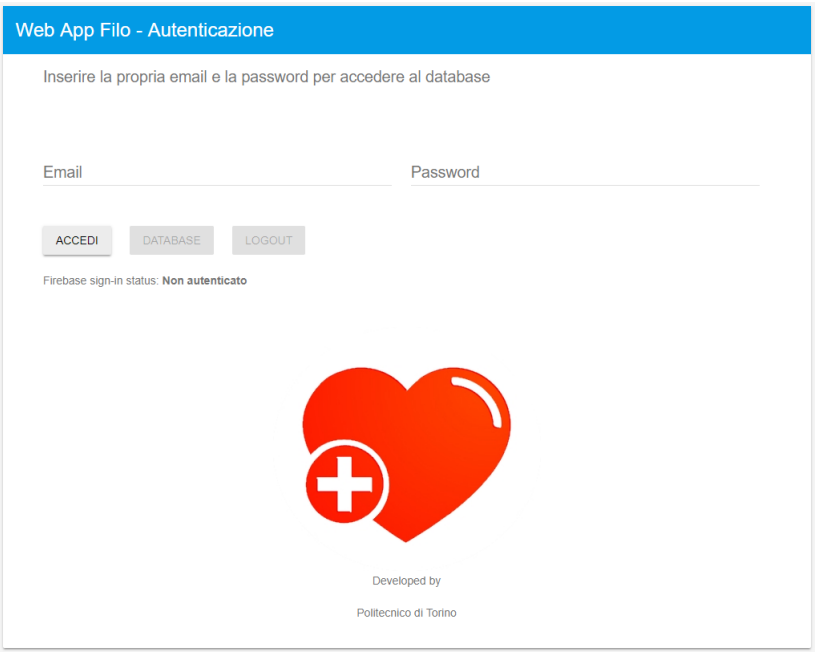

Figure 3.1: The Login Page

3.3 The Home Page

The web application **Home Page** shows the list of all patients registered into the database. They are listed using *Name*, *Surname* and *Fiscal Code* and clicking on them the **Patient Data Page** is opened.

The **Home Page** also gives a panoramic view of the patients' status. In fact, the name of those who added recently an *Event* (section 3.6) or added an alarming value of *Pressure*, *Glycaemia* or *Resting Heart Rate*, is in red colour, and it is listed above other patients. This allows physicians to be more alerted about the general status of all patients.

Figure 3.2 shows the page: the orange list contains "normal" patients, while the red one is the "alerted" patient list.

Figure 3.2: The Home Page

This page also shows two buttons. The first button ("Nuovo Paziente") opens the **Registration Page** to register a new patient. The second button ("Esci") allows to exit the platform and perform log off procedure.

3.4 The New Patient Page

This is the page from which it is possible to add new patients to the database (Figure 3.3). Personal data are requested to perform the operation, but only few are mandatory:

- Name ("Nome")
- Surname ("Cognome")
- Email
- Fiscal Code (Social Security Number) ("Codice Fiscale")
- Date of birth ("Data di nascita")

Name, *Surname* and *Fiscal Code* are used to identify the patient in the **Home Page**. *Email* is used by **Firebase** to create the patient account and to give him access to the Android application. Thanks to the fact that all emails are unique, the identification is unambiguous. *Date of birth* is not used as identifier, because the *Fiscal Code* is already a unique identification key, but it is a useful information about patients.

Not mandatory data are:

- Place of Birth ("Luogo di nascita")
- Residence ("Residenza")
- Telephone ("Telefono")
- Gender ("Sesso")

This page presents also two buttons. The *Register* ("Registra") button performs the patient registration on Firebase, using the email and creating a dummy password, and calls a function to send the Firebase Reset Password Email to the patient, so that he/she can choose his/her own one. The *Cancel* ("Annulla") button allows to cancel registration, redirecting physician to the **Home Page**.

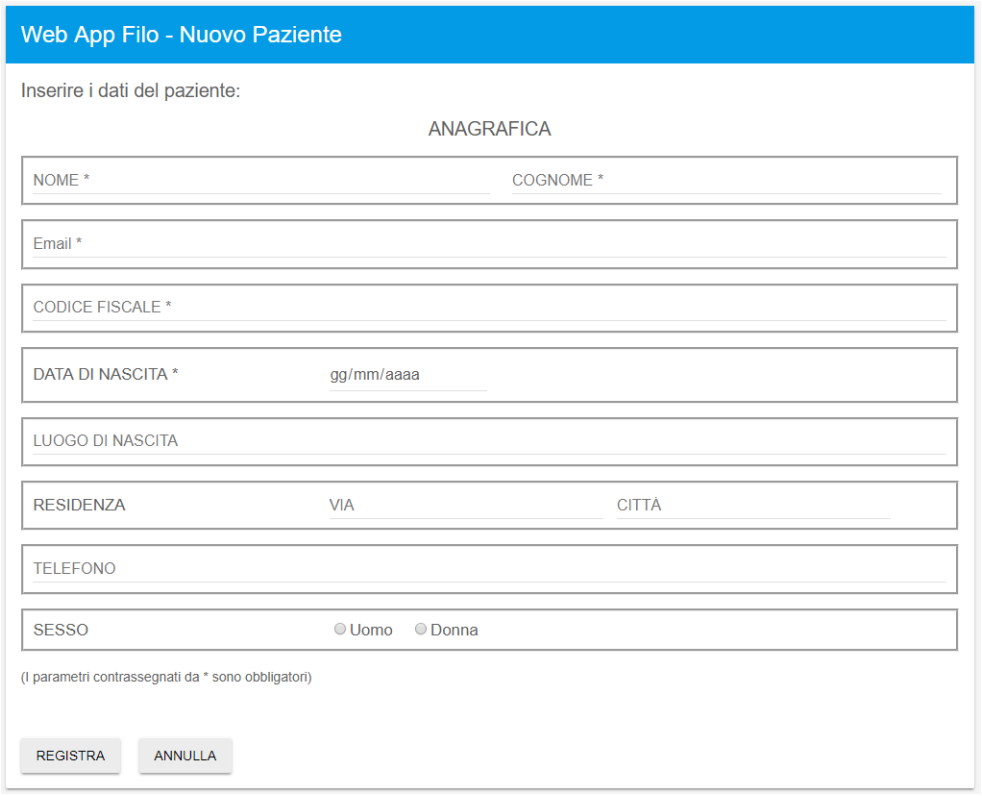

Figure 3.3: The New Patient Page

3.5 The New Visit Page

The **New Visit Page** is directly accessible after the patient registration and from the **Patient Data Page**. It allows to insert different kinds of data that are required during the first visit. Data are inserted and saved on the **Firebase Database**, so they can be retrieved any time. Also the personal data are accessible from this page and they can be modified if necessary. The only not editable data is *Email*, because it is used from Firebase to identify the account.

Data regarding more specifically the medical examination are:

- Risk Factors
- Body Mass Index (BMI)
- Previous Coronary Events Information
- Left Ventricular Ejection Factor
- Performed Angioplasties and still ill Vessels
- Complete History
- Therapy

3.5.1 Risk Factors

Risk factors are elements describing the patient tendency to develop a cardiac disease. Physician can tell if the patient is affected or not by a given risk factor using a radio button structure with double choice, *Yes* or *No*, except for *Allergies*, to which a free text field is given.

Smoking

Smoking greatly increases risk of having heart attack. It reduces the ability of blood in carrying oxygen. If there is less oxygen in blood, the heart has to work harder to pump blood to deliver the correct amount of oxygen to all body.

Tobacco products also make blood clots more likely to form, and increase the amount of plaque build-up in arteries. These two conditions increase the probability of a heart attack.

Smoking also increases the stiffness of blood vessels making harder for them to expand and contract as needed, and more likely to split.

These changes to the arteries can cause not only a heart attack, but also stroke or angina [12].

Hypertension

The excess strain and resulting damage from high blood pressure (*hypertension*) causes the coronary arteries serving the heart to slowly become narrowed from a build-up of fat, cholesterol and other substances that together are called plaque. This slow process is known as atherosclerosis.

When a coronary artery becomes blocked by those substances, the flow of blood through the vessel is interrupted starving the heart of oxygen and nutrients, causing heart attack [13].

Dyslipidaemia

Lipid abnormalities, including high levels of low-density lipoprotein cholesterol (LDL-C), elevated triglycerides and low levels of high-density lipoprotein cholesterol (HDL-C), are associated with an increased risk of Cardio Vascular diseases [14].

Diabetes

People with *diabetes* are more likely to develop heart diseases and have a greater chance of a heart attack. They are also more likely to have other risk factors, as high blood pressure and high cholesterol. Moreover, high blood glucose from diabetes can damage blood vessels and nerves controlling them and the heart.

People with diabetes tend to develop heart diseases at a younger age than people without diabetes [15].

Family history of Coronary Artery Disease

Having *Family History of Coronary Artery Disease* means to have first degree relatives who had a cardiovascular event in young age (less than 55 years in men and less than 65 in women) [16].

Even if the environment and the life style are key elements in causing them, it is proved that cardiovascular diseases are influenced by genetics [17].

Previous Chemotherapy

Chemotherapy side effects may increase the risk of heart disease, including weakening of the heart muscle and rhythm disturbances. Certain types of chemotherapy also may increase the risk of heart attack.

Anthracycline drugs are most commonly linked to changes of the heart muscle but fortunately, heart disease associated with chemotherapy is rare and not all chemotherapy drugs carry the potential side effect of heart damage [18] [19].

Previous Radiotherapy

Radiation therapy to the chest area often is part of the treatment for Hodgkin lymphoma and cancers of the lung, esophagus, or breast. Cardiotoxicity is a risk when a large volume of heart muscle is exposed to a high dose of radiation [20].

Although more modern treatment often involves lower cardiac doses and can therefore result in lower risk, there is evidence that an increased risk in developing cardiovascular disease remains for patients treated for Hodgkin lymphoma and others cancers [21].

HIV in treatment

HIV has been found to directly affect vascular biology resulting in an increased risk of cardiovascular disease compared to uninfected persons. Moreover, some cardiovascular medications can interact with *HIV* drugs and should not be used together.

HIV caused chronic inflammation, hypercoagulability and platelet activation all contribute to endothelial dysfunction, leading to higher risk for cardiovascular events [22].

Kidney Failure

Heart disease is the most common cause of death among people who have *kidney disease*. Moreover, they share two of the same main causes: diabetes and high blood pressure.

High blood pressure is not only a cause of kidney disease; kidney disease is also a cause of high blood pressure. When you have damaged kidneys, they may be unable to filter extra water and salt from your body. The high blood pressure that results can then make kidney disease worse. Worsening kidney disease can raise blood pressure again. A dangerous cycle results as each disease makes the other worse [23].

Allergies

Allergies may have an impact on heart and cardiovascular system. They can trigger a chronic train reaction. Allergies are a well-known cause for asthma. When the mucous membranes are inflamed over a longer period of time, an allergy can also predispose to bronchitis. Chronic lung disease may lead to right ventricular enlargement and failure.

Moreover, the combination of physical stress and a weakened immune system caused by allergies can lead to a weakening of the heart muscle and can give rise to heart muscle inflammation [24].

Figure 3.4: Risk Factors Form

Figure 3.4 shows the form for inserting *Risk Factors*. As explained at the beginning of this section (3.5.1) all choices are of boolean type, except for *Allergies*. In this field, the physician can write free text to better describe the patient's situation, and all will be saved into the **Firebase Database**.

The information stored during an examination can be modified and updated in following ones.

In the form, "Fumatore" stands for *Smoker*, "Iperteso" stands for *Hypertensive*, "Dislipidemico" stands for *Dyslipidemic*, "Diabetico" stands for *Diabetic*. "Familiarità per malattia coronarica" means the patient has a *Family History of Coronary Artery Diseases*, "Pregressa chemioterapia" and "Pregressa radioterapia" mean the patient has been respectively subjected to *Chemotherapy* or *Radiotherapy*. "HIV in terapia" means the patient is actually receiving a treatment for *HIV* virus. "Insufficienza renale severa" stands for *Kidney Failure*, while "Allergie" stands for *Allergies*.

3.5.2 Body Mass Index (BMI)

Body Mass Index is a value derived from the mass (weight) and height of an individual. It is defined as the body mass [kg] divided by the square of the body height [m] (Eq. (3.1)):

$$BMI = \frac{mass}{height^2} \left[\frac{kg}{m^2}\right] \tag{3.1}$$

BMI provides a simple numeric measure of a person's thinness.

CATEGORY	BMI[kg/m^2]	
	from	to
Underweight	-	18.5
Normal	18.5	25
Overweight	25	30
Obese	30	-

Table 3.1: BMI categories [25]

As shown in Table 3.1 if the *BMI* value is greater than 30, a person is considered obese. Obesity is a risk factor for fatal heart attacks even for people who do not have the conditions normally associated with cardiovascular disease [26].

By inserting height ("Altezza", [cm]) and weight ("Peso", [kg]), the platform automatically computes the correct *BMI*.

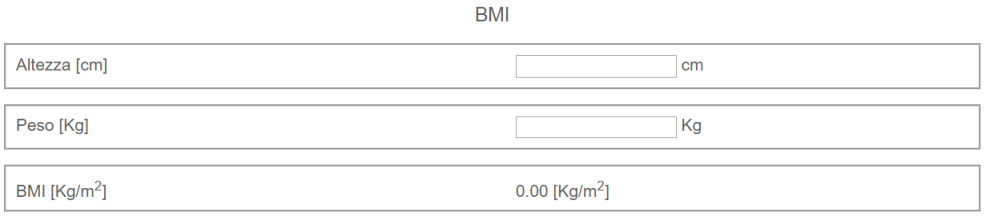

Figure 3.5: BMI Form

Figure 3.5 shows the form to insert patient's height and weight, to retrieve the *BMI* value.

3.5.3 Previous Acute Coronary Events Information

In this form, the physician can insert information about the patient's *Previous Acute Coronary Events*. More specifically, he/she can tell if the patient had a previous STEMI (ST elevation myocardial infarction), or NSTEMI (Non-ST elevation myocardial infarction). In the form, "Pregresso STEMI" stands for *previous STEMI*, while "Pregresso N-STEMI/ACS" stands for *previous N-STEMI/ACS*.

The first, is a coronary obstruction leading to a complete blood flow interruption in the area moistened by that artery, the second is constituted by a temporary blood flow stop [27].

Figure 3.6 shows the form to insert data about *Previous Acute Coronary Events*.

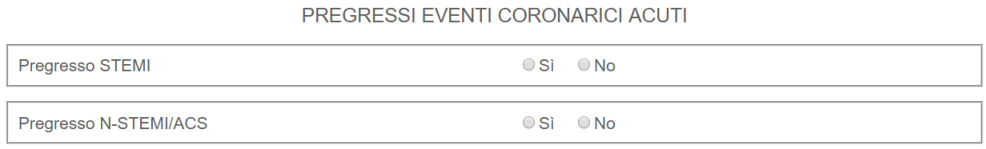

Figure 3.6: Previous Acute Coronary Events Form

3.5.4 Left Ventricular Ejection Factor (LVEF)

Left Ventricular Ejection Factor ("Funzione ventricolare sinistra") is the volumetric fraction ejected from the left ventricle at every heartbeat, with respect to the end-diastolic volume [28] (the ventricle volume at the end of diastolic phase [29]). It is measured as a percentage.

LVEF is used in medicine as an important determinant of the severity of systolic heart failure, due to cardiovascular diseases. This value should be about 67% in healthy people. If *LVEF* value decreases it can be very problematic, becoming really dangerous when it goes below 40%. At a value of 25% or below, the patient will present constant symptomatic manifestations.

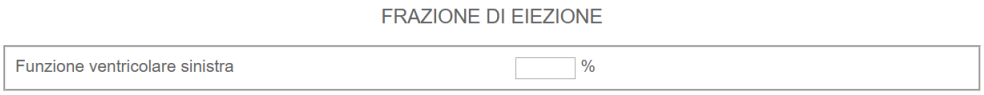

Figure 3.7: LVEF Form

Figure 3.7 shows the form to insert the *Left Ventricular Ejection Factor* value. The platform automatically rejects values below 30% and over 90%, avoiding at least some typing errors or the insertion of nonsense values.

3.5.5 Performed Angioplasties and still ill Vessels

In this form (Figure 3.8) the physician can insert several pieces of information about performed angioplasties, but also note if there are vessels that have not been treated yet.

The first field is the date ("Data"): automatically set to the current day, can be changed by the physician, but the platform itself does not allow to insert a future date for a performed angioplasty, avoiding those kind of errors.

The second field is the ill vessel. Physician can choose from a list of 20 vessels and tell if it is still ill ("Malato") or already treated ("Trattato"). If he/she chooses the *Treated* choice, the third field becomes fillable.

The third field describes the type of treatment performed on the vessel. If the *Treated* button is active, this field cannot be left unfilled, while it is not filled if the *Ill* button is active.

Figure 3.8: The Angioplasty Form

By clicking the *Add* ("Aggiungi") button, the performed angioplasty or still ill vessel is inserted in a list inside the platform, and the list will be shown above the input form. Figure 3.9 shows the form with two listed elements.

Figure 3.9: The Angioplasty Form with the Angioplasty List

Every element of the list can be deleted by clicking the relative *Delete* ("Elimina") button. Then, to complete the saving on the **Firebase Database** it will be necessary to click on the *Register Button* at the end of the page.

3.5.6 Patient History

This is one of the biggest form to fill during the first examination, but it will be half compiled during the following ones. Here the physician can insert the *Patient's Clinical History*, related or not to cardiovascular events and treatments.

All fields apart from the one related to the values of *Blood Pressure* and *Resting Heart Rate* are free text, with no limitations on the number of characters.

The first field is the *General History*, a record of all clinical elements related to the patient life. Generally, these are information about great clinical events, like some surgery performed in the past, or others elements that could be related to the actual situation of the patient.

The second field is the *Cardiovascular History*, information directly regarding the cardiovascular status of the patient end events related to it.

The third field is the *Proximal History*. These are recent events bound or not to cardiovascular disease (a patient could be admitted for other reasons and then develop a cardiovascular disease) but generally strictly related to the reason of the medical examination.

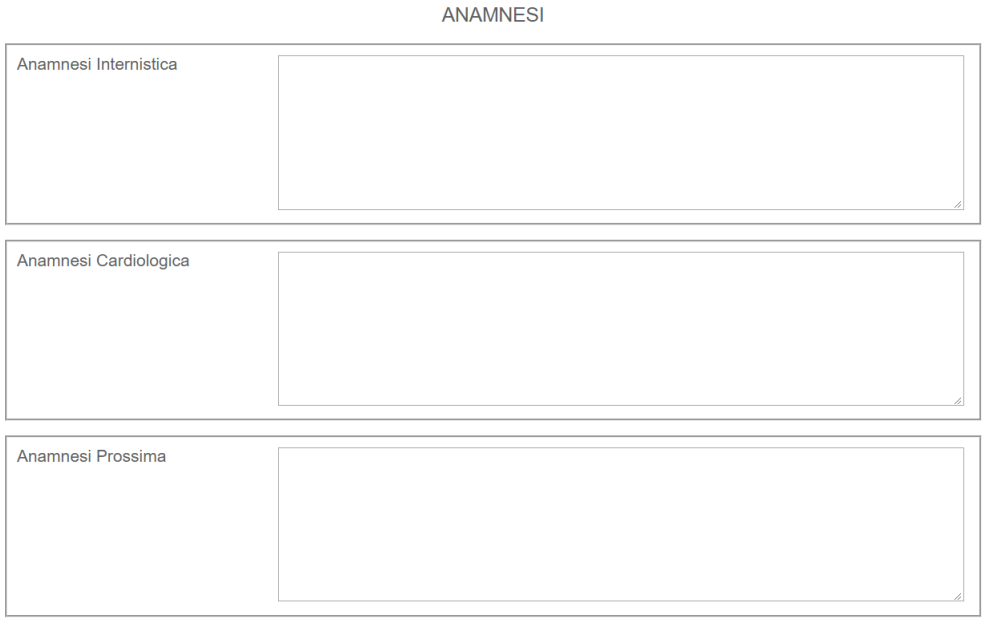

Figure 3.10: The Anamnesis Form - 1

In the form in Figure 3.10, "Anamnesi Internistica" stands for *General History*, "Anamnesi Cardiologica" stands for *Cardiovascular History*, while "Anamnesi Prossima" stands for *Proximal History*.

Then, there are the fields about blood pressure and heart rate. Also here, as in other numeric fields of this platform, there is a control about the range of the inserted value.

Values of *Systolic Pressure* below 60 $mmHg$ and above 270 $mmHg$ have no sense and are rejected by the input box, as values below 20 $mmHg$ and above 150 $mmHg$ for the *Diastolic Pressure*. A similar filter acts on the *Resting Heart Rate* input box, where values below 20 *bpm* and above 220 *bpm* cannot be inserted.

The sixth field is about the *Electrocardiogram* (*ECG*), where, in addition to the free text, physicians can also upload some files, by clicking on the *Choose Files* ("Scegli File") button. These are not limited to be images, but can be also files generated by the ECG machine that can be reviewed later, during future examinations. In fact, *ECG* is a fundamental document of a patient affected by heart disease and it is really useful to have historical records to make a chronological evaluation of their evolution. Knowing how *ECG* changes in time, it becomes easier to perform a good diagnosis.

To make this possible, there are no limitations on the number of files that can be uploaded. They will be saved directly into the **Firebase Storage**, in the patient directory.

The next field is about the current examination and it is a simple free text field.

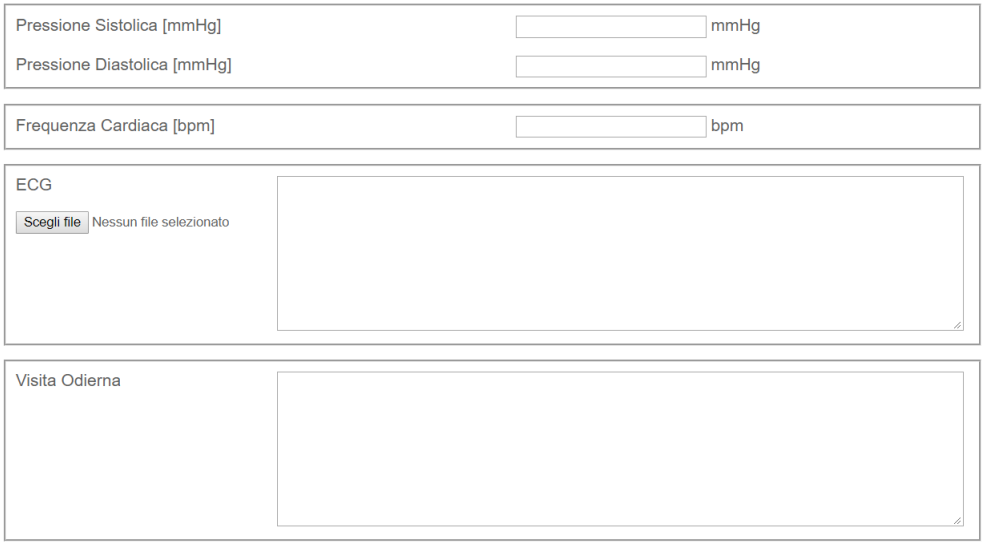

Figure 3.11: The Anamnesis Form - 2

In the form in Figure 3.11, "Pressione Sistolica" and "Pressione Diastolica" stand respectively for *Systolic Pressure* and *Diastolic Pressure*, "Frequenza Cardiaca" stands for *Heart Rate*, "Visita Odierna" means *Today's Visit*.

The eighth field is about examination *Conclusions* ("Conclusioni"), and all text written here will be saved being added with the current date to the *Cardiovascular History*, to better keep all patient information and make them more usable during future examinations.

After *Conclusions*, there is the *Therapy Changes* ("Variazioni nella Terapia") field. Here the physician can write all information about therapy modifications that can be useful to the patient and to his family doctor.

Figure 3.12 shows these last two fields.

Figure 3.12: The Anamnesis Form - 3

All text field of this and other areas in the platform autonomously adapt their dimension to the inserted text.

3.5.7 Therapy

This part of the physician platform is devoted to the definition of patients' drug therapy. As previous forms, it presents various fields to be as complete as possible in defining the therapy.

The first field is a text box to insert the drug name. Then it is possible to define if the drug has to be taken every day or not. If the *Every Day* button is checked, it is not possible to select single days. Selecting the button *Select Days* all check-boxes are unchecked and the physician can choose only the correct ones. Checking all days automatically disables single days check-boxes and selects the *Every Day* button.

Then the physician is able to choose if the drug has to be taken *Before Meal*, *After Meal* or if there are no specific requirements. Last, it is possible to choose if the therapy has an ending date or not by checking or not the *Until* check-box.

As in other similar fields, there is a control function on the ending date, that cannot be selected as a past date. Other control functions are activated if the *Add* ("Aggiungi") button is pressed without having specified the drug name, without having selected some days with the *Select Days* button checked, or without having specified an ending date with the *Until* check-box checked.

Figure 3.13: The Therapy Form

In the form in Figure 3.13, "Farmaco" stands for *Drug*. "Giorni", "Tutti i giorni" and "Seleziona i giorni" stand for *Days*, *All days* and *Select days*, then there is the list of all days of the week, from Monday ("Lunedì") to Sunday ("Domenica"). "Orari" means *Time*; "Dopo i pasti", "A stomaco vuoto" and "Indifferente" mean *After meal*, *On an empty stomach* and *Not specified*. "Fino a" stands for *Until*.

3.5.8 Events

The **New Visit Page** shows the list of all events communicated by the patient. This list, visible at the end of the page, will be obviously empty at the first visit, so it will be discussed in the **Patient Data Page** section (3.6).

3.5.9 Conclude Examination

At the end of this page there are two buttons: the *Register Examination Button* ("Registra Visita") and the *Conclude Examination Button* ("Concludi Visita"). Figure 3.14 shows the end of the page.

Figure 3.14: New Visit Page End

By clicking the *Register* one, if all mandatory data are still inserted (the same required in the **Add New Patient Page**) a **Confirmation Message** will be displayed (see Figure 3.15):

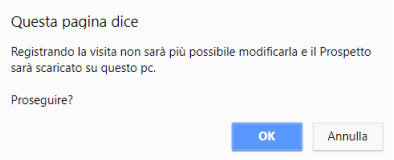

Figure 3.15: The Registration Confirmation Message

The message says: "Registering the visit you will not be ale to modify it and the prospectus will be downloaded on this pc. Continue?".

Accepting (*OK* button), all page elements excepted the other button will be disabled and set in read-only mode, as communicated by the message. This also registers into the **Firebase Database** all information inserted by the physician and saves into the **Firebase Storage** all files selected in the *ECG* form. Moreover, the examination prospectus pdf file will be created, downloaded on the used computer into the *Download* directory and saved into the **Storage**, to keep record of it. Denying (*"Annulla"* button), no action will be performed.

By clicking the *Conclude* button, the page will show a message (Figure 3.16) saying: "Examination completed". Closing it, the platform will redirect the physician to **Patient Data Page**, discussed in the next section.

Figure 3.16: The Conclude Examination Message

If the *Conclude* button is clicked before the *Registration* one, a different **Confirmation Message** is shown (see Figure 3.17). It says: "The visit has not been registered, conclude it anyway?".

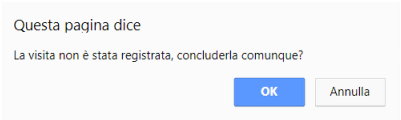

Figure 3.17: The Conclude Confirmation Message without Registration

By confirming, the physician will be redirected to the **Patient Data Page** but all the inserted data will be lost and the examination cancelled. By cancelling, the examination can continue and no effect will be applied on the page.

3.6 The Patient Data Page

In this page (Figure 3.18), all information collected from medical examinations is visible. The order is the same as in the **New Visit Page**. The only difference is that data are in read-only format.

The page top shows four buttons.

- The New Examination Button ("Nuova Visita")
- The Overview Button ("Panoramica")
- The Reset Password Button ("Resetta Password")
- The Delete Alert Button ("Cancella Alert")

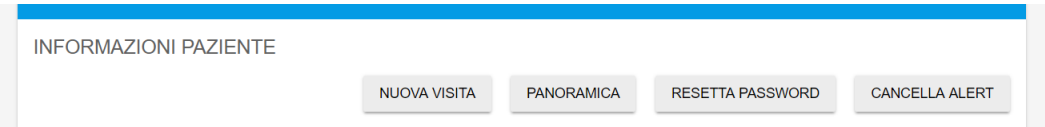

Figure 3.18: The Patient Data Page Top

The *New Examination* button opens in the same *tab* of the **Patient Data** page, the **New Examination** page. The *Overview* button opens the **Patient Overview** page, discussed in section 3.7. The *Reset Password* button, calls a **Firebase** function to send the patient a *Reset Password Email*. Obviously, patients can reset their password also from the **Android Application**, as discussed in chapter 4. The *Delete Alert* button changes the *Alert* field in the *patientsUid* branch to *false*, so that the patient name in the **Home Page** will be no more visualized in red colour and will be moved from the alert list to the normal list (see Section 3.3).

From this page it is also possible to access the list of patient's files stored into the **Firebase Storage**. Below the *History* title ("Anamnesi"), there is a link to access old examinations (Figure 3.19); at the end of the *ECG* form the one to access electrocardiogram files (Figure 3.20).

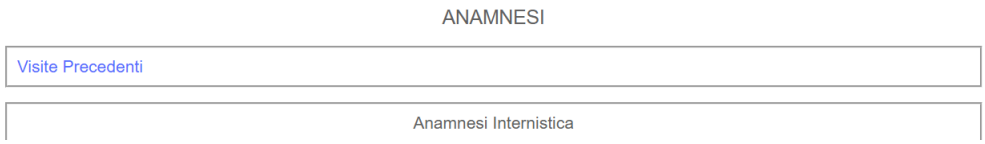

Figure 3.19: The Link to Old Examinations Page

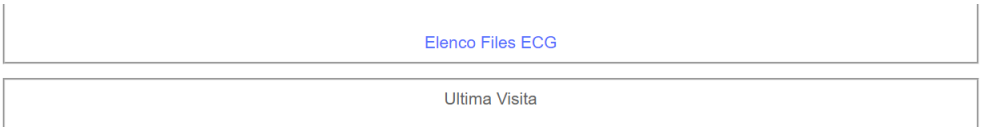

Figure 3.20: The Link to ECG Page

These two links respectively open in a new tab the **List Old Examination** page and the **List ECG** page, which are different but structured in the same way. Both present a simple list, sorted by date, of all files stored respectively in *ECG* and *Examination* directory. By clicking on the file name, it is possible to download the file, as better explained in the related section (3.8).

3.6.1 Events

The end of this page, as the end of the **New Visit** page, shows the list of all *Events* communicated by the patient through the **Android Application** (Figure 3.21). They are listed by date, and if the patient uploaded an image, the event name is also a link to open that image in a new tab.

The photo page will have as title the name of the *Event*.

Figure 3.21: The Events List

3.6.2 Patient Deletion

Below the event list there are still other two buttons (Figure 3.22). The **Back** ("Indietro") button closes the page end redirects to the **Home Page**, the **Delete Patient** ("Cancella Paziente") button, opens a new page entirely dedicated to the patient deletion.

Figure 3.22: Back and Delete Patient Buttons

The Delete Patient Page

From this page the physician can chose if to delete permanently the patient from the platform or not. By clicking on the **Delete Permanently** ("Elimina Definitivamente") button (Figure 3.23) Every information about the patient and every files in the storage related to him/her will be permanently removed.

By clicking on the **Back** ("Indietro") button, the physician will be redirect to the **Patient Data Page**.

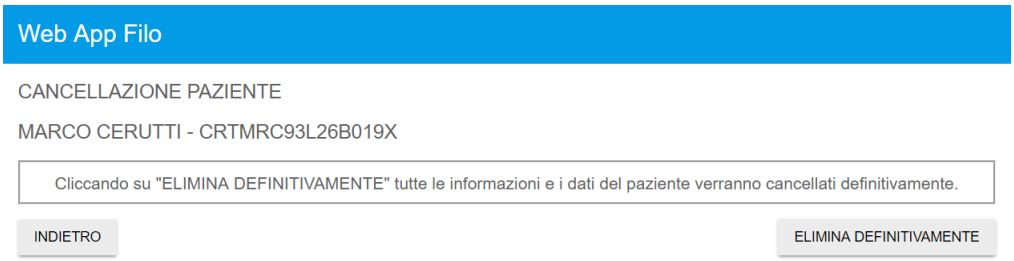

Figure 3.23: Delete Patient Page

3.7 The Patient Overview Page

This page is entirely populated with data provided by patient. These data are divided in three main groups.

- Numerical Data
- Smoke Data
- Physical Activity Data

3.7.1 Numerical Data

This part of the page is devoted to represent the time trend of data provided by the patients. They are *Systolic Pressure*, *Diastolic Pressure*, *Glycaemia*, *Heart Rate* and *Weight*. These data, represented with different colours, are by default plotted from the current date to 15 days before.

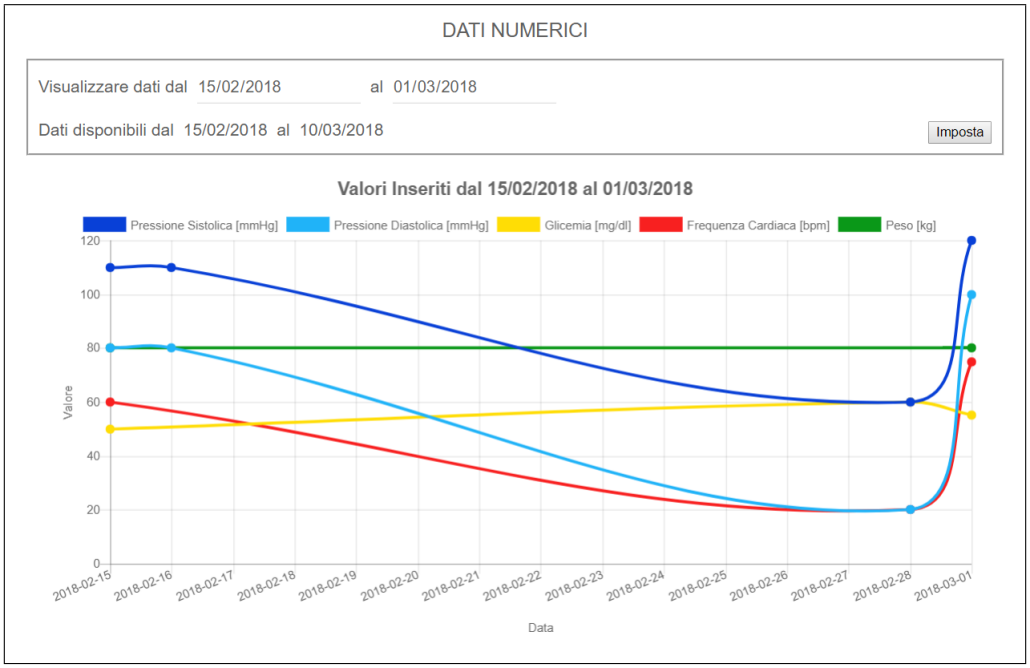

Figure 3.24: Numerical Data

Figure 3.24 shows the plot of numerical data and the form to set the representation interval. "Visualizzare dati dal - al" means *Visualize data from - to*, while "dati disponibili dal - al" stands for *Data available from - to*. "Pressione Sistolica" means *Systolic Pressure*, "Pressione Diastolica" means *Diastolic Pressure*, "Glicemia" means *Glycaemia*, "Frequenza Cardiaca" means *Heart Rate*, "Peso"

means *Weight*. The units of measure are indicated in the legend, on top of the chart.

The physician can choose whatever time interval to visualize data, by defining the first and last dates in the form on the top of the chart, and clicking on the *Set* ("Imposta") button. If the starting date is before the date of the first database entry, the starting boundary is shifted to not leave a white space at the beginning of the graph. If the ending date is after the last database entry, the ending boundary is shifted with the same principle. A message displayed on the page tells the physician the dates of first and last data.

When a value is represented by a dot, it means the data is present for that day. Dots are connected by a line to help visualizing the trend. By clicking on the data name in the legend it is possible to remove that data from the plot, to better visualize the other ones. By re-clicking, the data is plotted again. When a data is removed, the chart automatically rescales to represent the remaining ones as good as possible. All these functionalities are automatically implemented by the open source library used to represent charts, called **Chart.js** [30].

3.7.2 Smoke Data

Smoke data are collected from patient in boolean format, asking him if he has smoked or not day by day. For a better estimation, the platform takes into account also days in which the patient has inserted no data.

The red part represents days in which the patient smoked, the green the amount of days passed without smoking and the grey part represents all the days for which there are no data.

The chart's title tells the physician the date interval in which data are available, considering the first and last entries in the database.

Figure 3.25 shows the pie chart created using smoke data. "Statistiche fumo dal - al" stands for *Smoking statistics from - to*; "Si", "No" and "Nessun Dato" stand for *Yes*, *No* and *No Data*.

As in the previous chart, it is possible to remove some data from the pie chart by clicking on the data name in the legend.

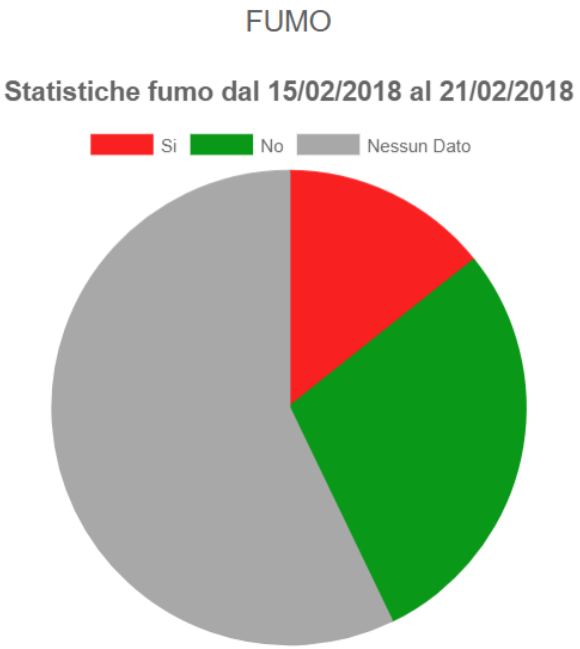

Figure 3.25: Smoke Data

3.7.3 Therapy Data

Data about the patient's compliance with the therapy are collected in boolean format. Also if the patient is notified to give this information every day thanks to **Android Notifications**, it is possible that some day is missing, so the representation mode is the same used for *Smoke Data*.

Figure 3.26 shows the pie chart built from therapy compliance data, where the green colour represents days in which the patient has taken correctly all medicine that were prescribed, the red colour days in which he/she did not, and in grey colour days with no information about therapy compliance.

As in smoke's pie chart, the chart's title tells the physician the date interval in which data are available, considering the first and last entries in the database.

In the chart's title, "Statistiche Assunzione Farmaci dal - al" stands for *Drug Assumption Statistics from - to*; "Si", "No" and "Nessun Dato" stand for *Yes*, *No* and *No Data*.

As in the previous charts, it is possible to remove some data from the pie chart by clicking on the data name in the legend.

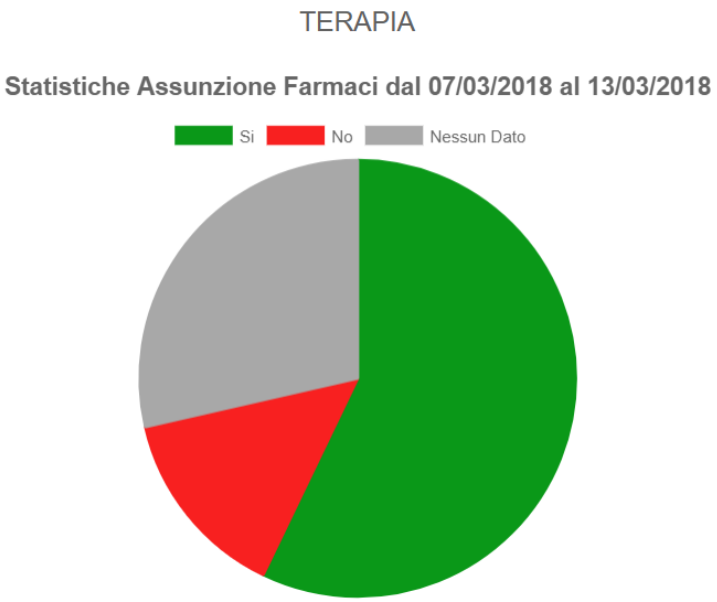

Figure 3.26: Therapy Data

3.7.4 Physical Activity

This part shows the amount of time each day the patient spent doing physical activity. Data regards four types of activities:

- Walking ("Camminata")
- Running ("Corsa")
- Biking ("Bicicletta")
- Gym ("Palestra")

Figure 3.27 shows an example of the generated plot. A bar diagram shows the medium amount of time spent each day doing each kind of activity in minutes. The entire amount of time from the first inserted data is taken into account.

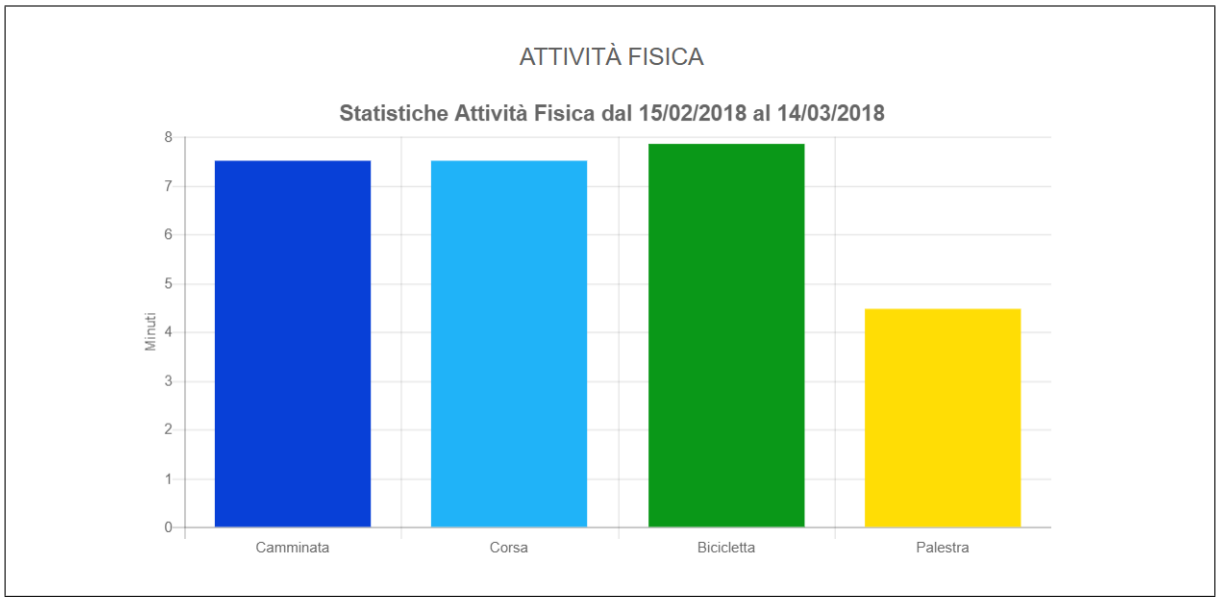

Figure 3.27: Physical Activity Data

In the chart's title, "Statistiche Attività Fisica dal - al" stands for *Physical Activity Statistics from - to*.

The unit of measure used to count the amount of time spent doing physical activity is the *Minute* ("Minuti").

3.8 The Old Examinations and the ECG Pages

These two pages show a list of files stored into the **Firebase Storage**. Depending on the page, two different directories are interrogated. Each element of the list is a link to view and download the related file. Files are listed by date.

If the file is not in a format the browser can read, it is downloaded directly, but if it is viewable in the browser, it is opened in a new tab, where it is possible to read it without downloading. Clearly, the physician can still download it and, in this case, also choose the download directory.

Figure 3.28: File Lists in the two Pages

Figure 3.28 shows the two lists. "Elenco Visite Passate" means *Old Examinations List*, while "Elenco Files ECG" stands for *ECG Files List*.

In the same figure it is possible to see how each file has a proper *Delete* ("Elimina") button. If that button is clicked, a **Confirmation Message** (Figure 3.29) is shown to the user, asking if he/she really wants to delete it, because the file will be permanently removed from the storage. The message says: "The file will be removed permanently. Continue?".

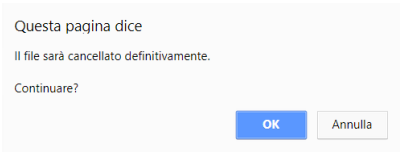

Figure 3.29: Delete File Confirmation Message

By clicking the *Ok* button the file will be deleted, by cancelling ("Annulla" button) no operation will be performed and the dialogue closes.

Chapter 4

The Android Application

4.1 Introduction

The **Android Application** is the patient interface to access the **Filo** platform. Once a patient is registered by the physician using the **Web Application**, he/she can access the application using *Email* and *Password*. Password can be changed any time from the app, if needed.

From the **Android Application** patients have a sort of communication line to physicians. They can insert data about their clinical situation day by day. They can access their history to retrieve information that can be useful, for example, to their family doctor. Patients also receive notifications from the app to remember to take medicines, and they can communicate if they are following the treatment or not.

All these data are collected in the **Firebase Database** and can be accessed by physicians using the **Web Application** (chapter 3).

4.2 The Login Activity

Figure 4.1 shows the first screen a patient sees in the **Android Application**.

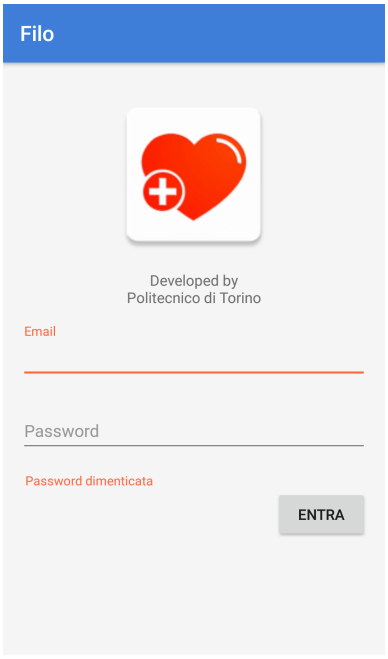

Figure 4.1: The Login Activity

From this page, users can access the application using their personal *email* and *password*. **Login Activity** manages different kinds of errors the user could make during the login phase. In the activity, "Entra" means *Enter*, while "Password dimenticata" means *forgot password*.

4.2.1 User Errors Handling

Missing Data

If the user forgets to insert email or password, he/she is advised of that and the first empty input box is focused (Figure 4.2). The message says "Campo richiesto", that stands for *Missing field*.

Badly Formatted Email

If the inserted Email does not have the correct sequence of characters, so it does not contain the *at* (@) sign before the domain name and the *full stop* inside it, the user is advised of that and the input box is focused (Figure 4.3). The message says "Questa Email non è valida", that stands for *This Email is invalid*.

Incorrect Email

If the Email is formatted correctly but there is no user with that email, usually due to a user typo error, the login fails and a message is shown as a *Toast* at the page bottom (Figure 4.4). The Toast message in the figure stands for: "*There are no users with these credentials. Please, verify email and password have been inserted correctly*".

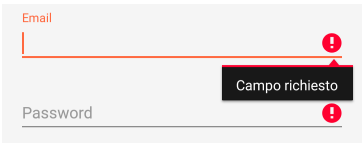

Figure 4.2:
Missing Field

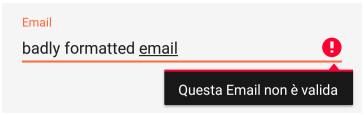

Figure 4.3:
Badly Formatted Email

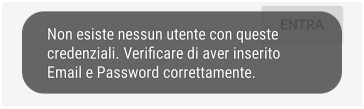

Figure 4.4:
Incorrect Email

Short Password

Firebase password rules accept only passwords having at least 6 characters, so if the user inserts a shorter one for some typo errors, he is advised of that and the input box is focused (Figure 4.5). The message says "Password troppo corta", that means *Too short password*.

Wrong Password

If the *Email* coincide with the one of a registered user and there are no other errors but the password is wrong, a *Toast* message is displayed to the user to advise him of that (Figure 4.6). The Toast message says "Password errata", that stands for *Wrong password*.

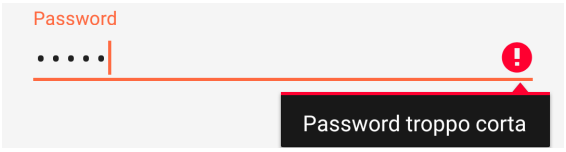

Figure 4.5:
Short Password Error

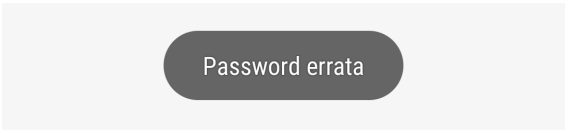

Figure 4.6:
Wrong Password

All these messages are displayed after the *Login* button is pressed. If no errors occur, the login is performed and the user is redirect to the **Main Activity** of the application.

If the user cannot remember his/her password, the application offers the possibility to reset it, by clicking on *Forgot Password*. This opens the **Forgot Password Activity**.

4.3 The Forgot Password Activity

This simple activity is devoted to reset the password in case the user forgot it (Figure 4.7).

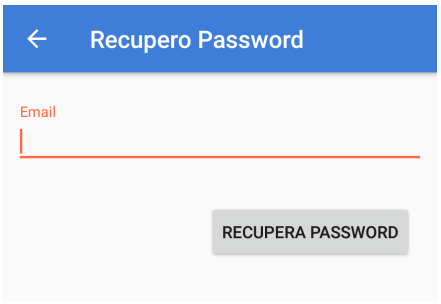

Figure 4.7: Forgot Password Activity

The user has to insert the Email used by the physician during the registration phase, and click the *Recover Password* button. This activates the same **Firebase** function activated by the *Reset Password Button* in the **Patient Data Page** of the **Web Application** (section 3.6).

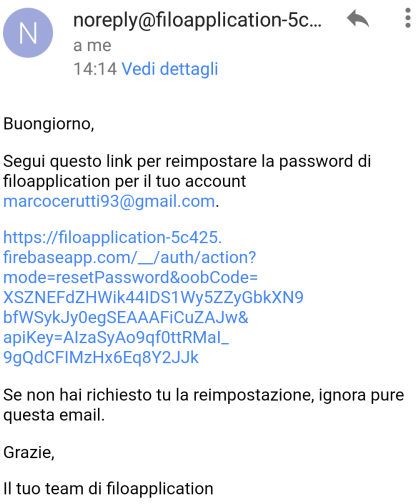

Figure 4.8:
Reset Password Email

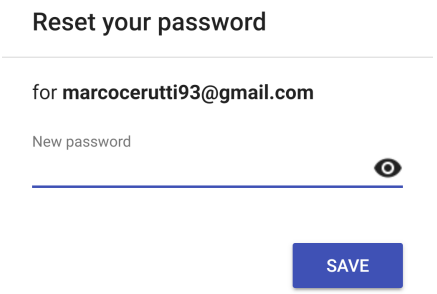

Figure 4.9:
Reset Password Firebase Form

It sends to the inserted Email address an email with a link to reset the password (Figure 4.8). This link opens the **Firebase** form to do it. The only constraint on the new password is to be at least 6 characters long, as requested by **Firebase Security Rules**. Clicking on the *Save* button the action is performed.

4.4 The Main Activity

This is the main screen of the application (Figure 4.10). When the user logs in, it is automatically opened, and if the application is closed without logging out, this screen is shown at the next application reopening.

This activity, built on a two-by-two *Table Layout*, shows four buttons to perform four different operations:

- Data Registration Activity Button ("Registra Dati" stands for *Records Data*)

  It opens the Data Registration Activity, where the user can save data about his/her day-by-day situation.

- Therapy Activity Button ("Terapia" stands for *Therapy*)

  It opens The Therapy Activity, where the user can see his/her prescription, decided by the physician.

- Clinical History Button ("Cartella Clinica" stands for *Clinical History*)

  It opens the Clinical History Button, where the patient can see all his/her personal and clinical data, including the Cardiovascular History.

- Exit Button ("Esci" stands for *Exit*)

  This button perform logout and close the application.

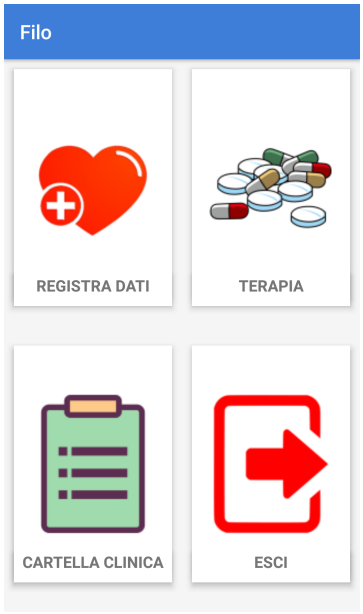

Figure 4.10: The Main Activity

4.5 The Data Registration Activity

This is the activity devoted to patient-to-physician communication. From this screen, the patient chooses which kind of data to save on the **Firebase Database**, by clicking on the related button (Figure 4.11).

"Pressione Sanguigna" stands for *Blood Pressure*, "Peso" stands for *Weight*, "Sigarette Fumate" stands for *Smoked Cigarettes*, "Attività Fisica" stands for *Physical Activity*, "Frequenza Cardica" stands for *Heart Rate*, "Glicemia" stands for *Glycaemia*, "Evento" stands for *Event*.

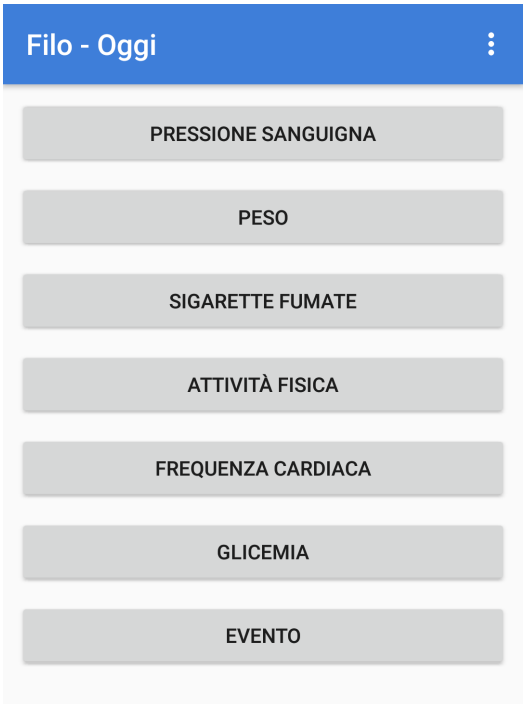

Figure 4.11: The Data Registration Activity

Each button opens a specific dialogue box, devoted to the communication of that specific kind of data, except for the *Event Button*, that opens a dedicated activity. When a dialogue has been opened, it cannot be closed by clicking outside it on the screen, but it can be closed only by completing the operation, by clicking *Ok*, or by deleting it, by clicking *Delete* ("Annulla"). Again, a few adjustments have been implemented to avoid user mistakes.

First, by clicking on the input box of a numerical data, only the numerical keypad is accessible and no other characters are permitted. Moreover, in case of values that have to be integer, also the decimal point is disabled.

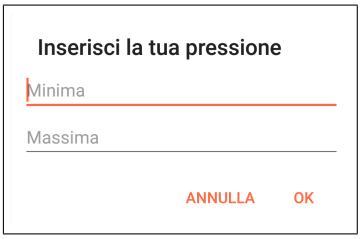

Figure 4.12:
Pressure Dialogue

Figure 4.13:
Pressure Dialogue
Empty Field Error

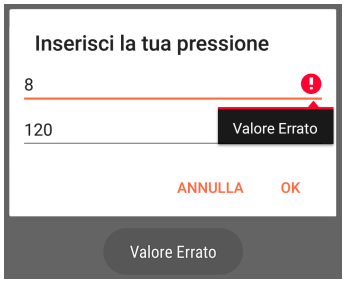

Figure 4.14:
Pressure Dialogue
Wrong Value Error

Pressure Dialogue in Figure 4.12 is used as example to explain the other two adjustments. In the dialogue box, "Inserisci la tua pressione" means *Insert your pressure values*, while "Minima" and "Massima" respectively stand for *Minimum (Diastolic)* and *Maximum (Systolic)*.

The second implemented adjustment avoids to insert empty data: data registration is blocked if a requested field is not filled (Figure 4.13).

Third, a control function avoids the user to insert values completely off the scale, that would be nonsense data (Figure 4.14).

In all these cases, the error field is focused and a Toast message is displayed.

Weight Dialogue is structured in the same way, the only exceptions are that has only one field to be compiled and that the decimal point is accepted (Figure 4.15). In the form, "Inserisci il tuo peso" stands for *Insert your weight*.

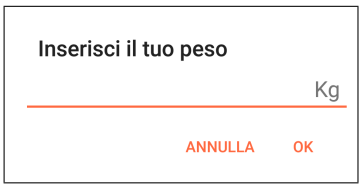

Figure 4.15:
Weight Dialogue

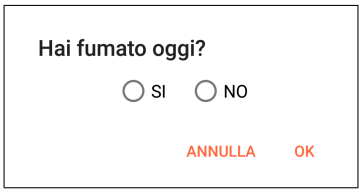

Figure 4.16:
Smoke Dialogue

The *Smoke Dialogue* instead is quite different. It is composed by a single radio button, where the patient can tell if he smoked or not in the considered day (Figure 4.16). In the dialogue box, "Hai fumato oggi?" stands for *Have you smoked today?*, while "Si" and "No" mean respectively *Yes* and *No*.

As in the other two dialogues, the user can close it only by responding or clicking the cancel button. This choice has been made to avoid the user to confirm without giving that piece of information, maybe convinced otherwise.

The fourth button opens the *Physical Activity Dialogue*. This allows the patient to communicate how much time in minutes he/she spends each day doing some physical activities, divided in four categories: *Walking* ("Camminata"), *Running* ("Corsa"), *Biking* ("Bicicletta") and *Gym* ("Palestra") (Figure 4.17). "Attività Fisica" stands for *Physical Activity*, while "Minuti" stands for *Minutes*.

Figure 4.17: Physical Activity Dialogue

In this case the dialogue box behaves slightly differently. In fact, this is the only box allowing to have some field not filled in: if none is filled, the application behaves normally, advising the user in the ordinary way, but it accepts the confirmation if at least a single field is compiled. This because it is normal for a person to do different kinds of physical activity maybe in different days, or in different moments of the day. So the patient can fill each input box when and as he/she wants.

The last dialogues opened by this activity are the *Heart Rate Dialogue* (Figure 4.18) and the *Glycaemia Dialogue* (Figure 4.19). Those dialogue boxes follow the same rules of the *Weight Dialogue*, but the first accepts only integer numbers as input.

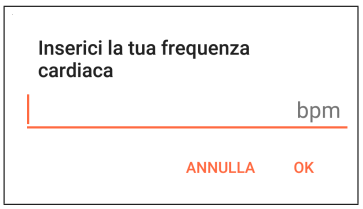

Figure 4.18: Heart Rate Dialogue

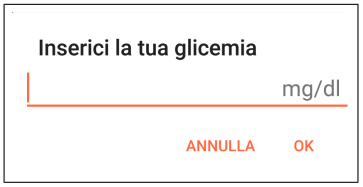

Figure 4.19: Glycaemia Dialogue

All those dialogues, when the *OK* button is pressed, publish the inserted data into the **Firebase Database**. If no other action is performed by the patient, data are published referring the current day, but the user is allowed to insert data also related to previous days. This activity presents a three dots menu on the top right of the view, from which it is possible to change the considered date (Figure 4.20).

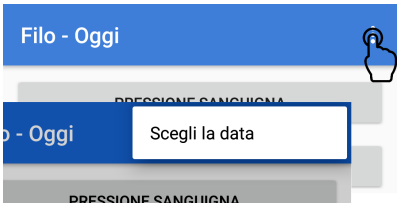

Figure 4.20:
Three Dots Menu [31]

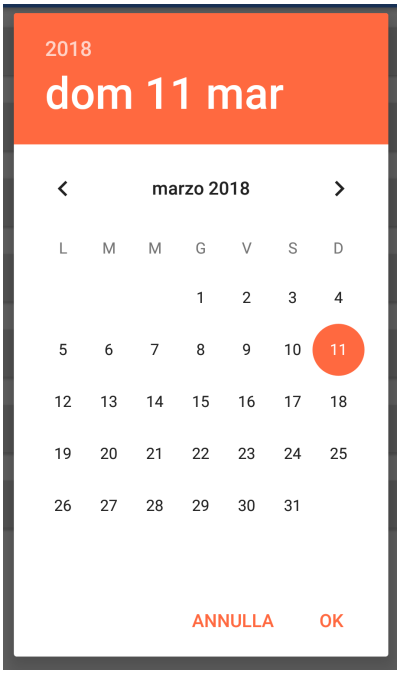

Figure 4.21:
Date Picker

By clicking on the *Choose Date* ("Scegli la data") menu item, an *Android Date Picker* is opened on the screen (Figure 4.21).

At its opening, the date picker is always set on the current date. From this box the patient can choose a different date to insert his/her data. Any past date is accepted, but when the user opens a dialogue box, a message is shown to remember that he/she is modifying a past information (Figure 4.22). This because the user could still make some mistakes in inserting his/her data, or not having the possibility to do that or forget it. The message in red says: "*Attention! You are modifying a past information!*".

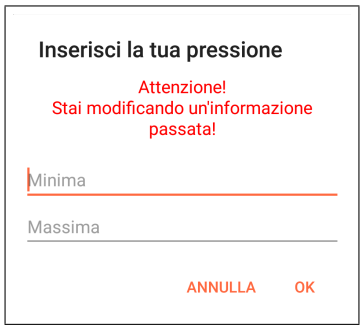

Figure 4.22: Pressure Dialogue Box for a Past Date

When the user presses *OK* from the *Date Picker*, it is closed and the title of the **Data Registration** activity changes from *Filo - Today* to *Filo - chosen date"* (Figure 4.23).

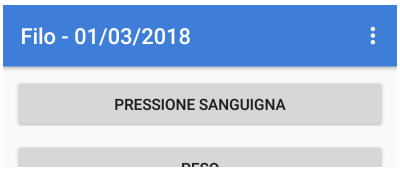

Figure 4.23:
Data Registration Activity
for a Past Date

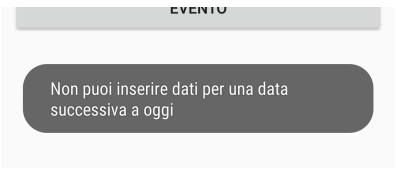

Figure 4.24:
Future Date Toast Message

As for all the other parts of the application, some adjustments have been taken also for the *Date Picker*. If the user selects a future date to insert the data, the application rejects this choice, setting the date to the current day, also changing the title to *Filo - Today*, and displaying a Toast message advising the user that he/she cannot choose a future date (Figure 4.24). The Toast message says: "*You cannot insert data for a future date*".

The last button, the *Event* button, opens the **Event Registration Activity**, discussed in section 4.6.

4.6 The Event Registration Activity

The *Event* button on the **Data Registration Activity** opens a new activity, entirely devoted to the collection of information regarding different events that might affect the patient during the follow-up (Figure 4.25).

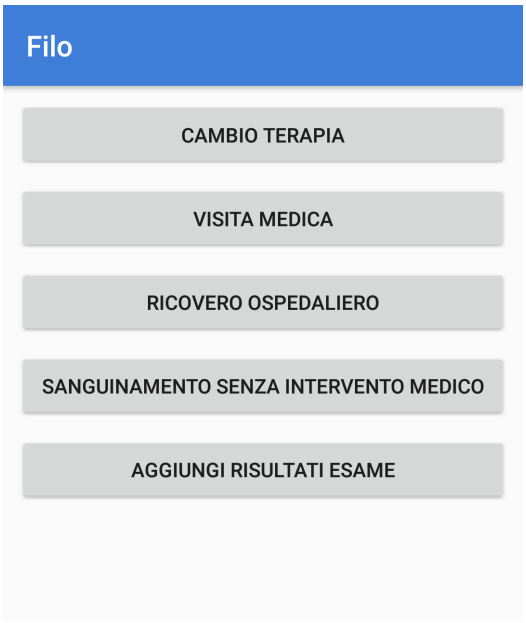

Figure 4.25: The Event Registration Activity

As in the **Data Registration** activity, each button of this page opens a dialogue box to register the specific event. Five different kinds of events have been defined according to physician requests: *Therapy Changes* ("Cambio Terapia"), *Physical Examination* ("Visita Medica"), *Hospitalization* ("Ricovero Ospedaliero"), *Bleeding without Medical Intervention* ("Sanguinamento senza Intervento Medico") and *Examinations Results* ("Aggiungi Risultati Esame").

In the change therapy box (Figure 4.26) the patient has to insert a description of the therapy modification he/she has adopted, and if the choice was made by himself/herself or under the advice of an other physician. In the box, the question below the title asks: "*It has been a personal decision or a medical advice?*". "Personale" and "Medico" mean respectively *Personal* and *Physician*. Then, the form asks for a description of the adopted changes: "*Describe your changes in therapy*" ("Descrivi il cambiamento nella terapia").

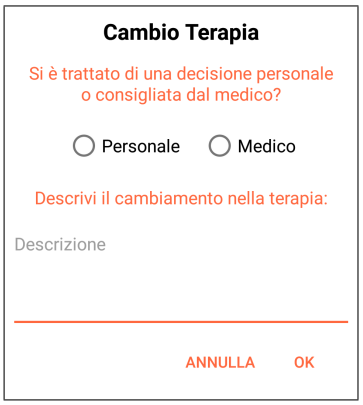

Figure 4.26:
Change Therapy
Dialogue

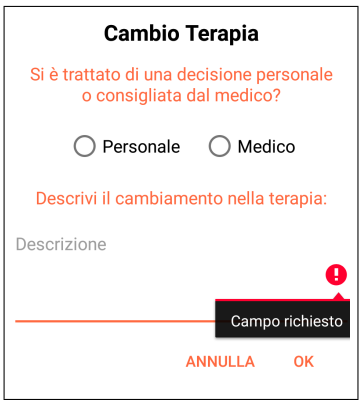

Figure 4.27:
Change Therapy
Dialogue Missing Field

Again the system prevents the user from inserting data with missing fields. If the description field is empty when the *OK* button is pressed the dialogue remains opened, and the *Description* field is focused (Figure 4.27). If the *Choice Radio Button* has not been filled in, a Toast message is displayed to the user (Figure 4.28). The Toast message says: "*Select Personal or Physician*". Only if both the radio button and the description field have been filled the dialogue closes and the information is registered into the database.

Figure 4.28:
Change Therapy Dialogue
Missing Choice Toast Message

Physical Examination and *Hospitalization* dialogues, having only a text box to be filled, have only the control on the empty field, that acts in the same way as the previous one. Here the patient has to insert respectively the reason of the examination, especially if he/she went to another physician, and the cause of the hospitalization.

Figure 4.29 shows the *Physical Examination* dialogue. "Inserisci la ragione della visita" stands for *Insert the reason for the examination.*

Figure 4.30 shows the *Hospitalization* dialogue box, where "Inserisci la ragione del ricovero" means *Insert the reason for the hospitalization.*

Figure 4.29: Physical Examination Dialogue

Figure 4.30: Hospitalization Dialogue

The next kind of event a patient can communicate using the application, is the bleeding event, solved without medical intervention. In fact bleedings are not always so serious to require it, but can also be managed by the patient alone. However, in those cases it could be useful for the physician to know the duration of those bleeding events, so the patient can communicate it with the **Filo** application. As the previous dialogues, if the field is not filled in, the user is warned about it.

Figure 4.31 shows the *bleeding* dialogue box.

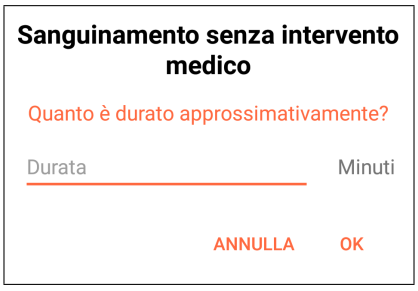

Figure 4.31: Bleeding Dialogue

The box title, "Sanguinamento senza intervento medico", means *Bleeding without medical intervention*. It also asks "Quanto è durato approssimativamente?", that means *How long has lasted approximately?*. The patient has to insert the duration in *minutes* ("Minuti").

The last button of this activity opens a box to communicate exam results. Figure 4.32 shows the dialogue box. The patient can make a photo of the exam result sheet and give it a name. It is converted to a *String* using the *Base64* encoding and it is stored into the **Firebase Database**. In this way this photo can be always retrievable from physicians though the **Web Application**.

The photo can be taken in that specific moment, by clicking the *Camera* button, or picked from the phone memory choosing the *Gallery* or the *Photos* application. When the photo has been chosen, it can be also cropped if the user wants it. Figure 4.33 shows the modality to chose the photo source. "Selezionare un'applicazione per completare l'azione" stands for "*Select an application to complete the action*".

The image cropper has been created using a third part library, developed by ArthurHub [32].

Figure 4.32:
Exam Result Dialogue

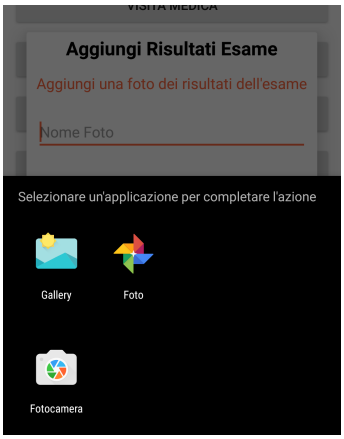

Figure 4.33:
Exam Result Photo Picker

The box title means *Add Examination Results*, while the indication "Aggiungi una foto dei risultati dell'esame" means "*Add a photo of examination results*".

The user is not obliged to insert a photo, but he/she can only communicate that he/she has taken a physical examination. In any case, if he/she does not insert anything, he/she is warned as in all others dialogue boxes.

4.7 The Clinical History Activity

This is a read only activity where the patient can access data given to the physician during the cardiovascular examinations. The patient can read all his/her personal data (Figure 4.34), being advised to the major *Risk Factors* affecting him/her (Figure 4.35), knowing his/her status at the last examination (Figure 4.36), read their proper general and cardiovascular history (Figure 4.37).

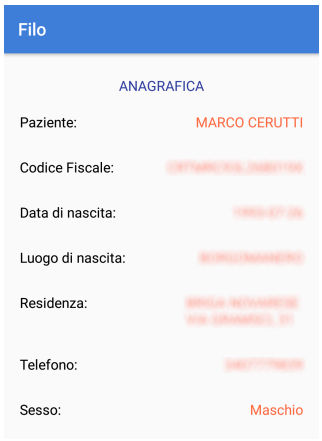

Figure 4.34:
History Activity (1)

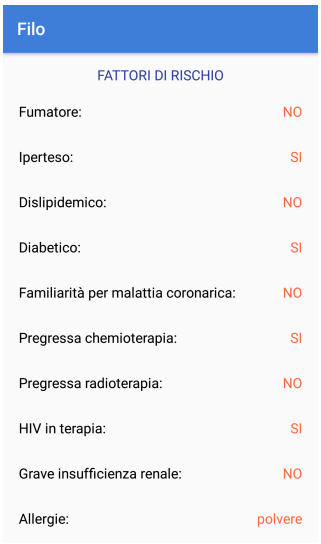

Figure 4.35:
History Activity (2)

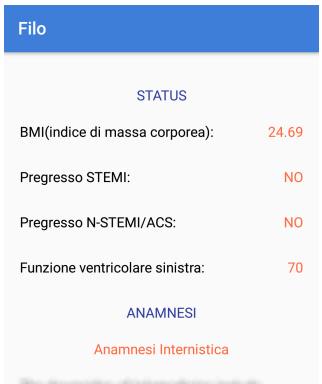

Figure 4.36:
History Activity (3)

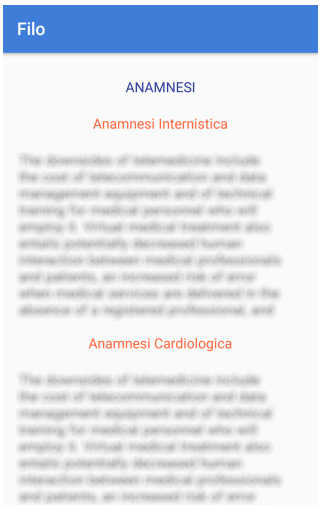

Figure 4.37:
History Activity (4)

The page is build on a scroll-view layout, allowing to scroll the screen up an down to see all information visible on the page. Information is retrieved from the **Firebase Database** when the activity is opened, and during the download a circular progress bar is shown to notify that something is happening. When the view is entirely populated, the progress bar is removed from the screen.

From this page, patient can access the following data:

- Personal Data

  Name, Fiscal Code, Date of Birth, Place of Birth, Residence, Telephone Number and Gender. This allows the patient also to be informed if a datum is missing or wrong, and he/she can do something to communicate the error to the physician.

- Risks Factors

  If the patient is a smoker, if he/she suffers of Hypertension, Dyslipidaemia or Diabetes, if he/she has a family history of coronary diseases, if he/she had previous Chemotherapy or Radiotherapy or if he/she is following a treatment fo HIV, if he/she suffers of kidney failure or has some allergies.

- Actual Status

  The patient BMI value, if he/she had previous STEMI or N-STEMI surgeries, his/her left ventricular ejection factor at the last examination.

- History

  General and Cardiovascular history, with last examination information, treatment and advices.

To have all patient own information in the same place, easy to be retrieved, could be very useful to the patient and to other physicians he/she could meet during the follow-up, but this activity can also be a very effective instrument to avoid and correct data registration errors, because the patient has always under control all his/her own data.

4.8 The Therapy Activity

This is an other very important activity of the application. It is populated with the information related to the therapy the patient has to follow. It is build on a *Fragment Layout*, where each fragment corresponds to a day of the week. Figure 4.38 shows the activity set in the Thursday fragment. Right below the title ("Terapia" stands for *Therapy*), is possible to see the menu to select the day, from Monday ("Lun") to Sunday ("Dom"). "Orario" stands for *Time*, "Medicinale" means for *Drug*.

It is possible to surf fragments both by clicking on one of them in the menu or by sliding to the right ant to the left on the screen.

Figure 4.38: Therapy Activity

Each fragment sheet presents on the left the time of the day a medicine has to be taken and to the right the drug's name, with some extra information about the assumption, if they are given. In particular, it is notified if the medicine has to be taken after a meal or on an empty stomach.

The possible times go from 8.00 in the morning to 22.00, with an interval of one hour. If in a specific hour the treatment requires a medicine assumption, the time is visible, otherwise it remains hidden, so the view is not overloaded with too many arguments. Also this view, as the one of the *Clinical History*, is scrollable so also if the therapy indications become a lot for every day, they are all accessible from each fragment.

The **Therapy Activity** can be accessed in two different ways, through the in-app notifications (section 4.8.1) and through the *Main Activity*. In both cases, when it is opened, the activity shows the fragment corresponding to the current day, to facilitate the user to retrieve the correct information. So the application becomes also a sort of calendar for the patients, to help them remember to follow the therapy.

This activity also presents a three dots menu, where the user can open a dialogue box to communicate that he/she has taken all the drugs for the current day. This helps the physician to know how much the patient is compliant with the therapy, taking for granted that the patient is honest.

4.8.1 The Application Notifications

Filo app notifications are of two types. The first is only to remember the patient to take its medicines, the second is also to retrieve data from the user about its compliance with the therapy.

Figure 4.39 shows the first type of notifications: "Ricordati di prendere i tuoi medicinali" means "*Remember to take your medicines*". Figure 4.40 shows an example of the second type on notifications: "Hai preso le tue medicine oggi?" stands for "*Have you taken your medicines today?*".

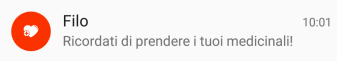

Figure 4.39:
Notification Type 1

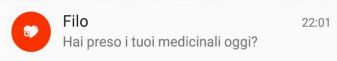

Figure 4.40:
Notification Type 2

Notifications are shown four times in the day: at 10.00 a.m. and at 14.00, 18.00 and 22.00. The first three are of the first type, the last one of the second type.

First type notifications simply opens the therapy activity when clicked, at the current day page, while the second type ones open the activity and also open the compliance dialogue box, to make easier for the user to communicate the requested information.

Notifications of type one can also be dismissed from the Android Notification Overview, to avoid them to be exaggeratedly invasive. Type two not, because it is important to have the compliance information.

Conclusions

The main purpose of this thesis was to develop a working prototype of **Filo Application**, in the form of a **Web** and a **Mobile** (**Android**) platforms. The first have been developed using **HTML/CSS** and **Javascript** languages, to define respectively layout and functionalities of each page, and did not require a specific development platform. Instead, the second was developed using **Android Studio**, the primary **Integrated Development Environment** (**IDE**) for Android-native application development. Principal languages used in Android are **XML** for the layout and a **Java-based language** for the functionalities.

iOS was not considered due to the more restrictive development policies and requirements adopted by **Apple**.

To fulfil the requirements, an **authentication** and **data storage** provider was necessary. **Google Firebase** was chosen due to its flexibility. It is quite easy to interface it with these platforms, and it also offers the possibility to be linked with other ones (iOS, AndroidPhone), that may be requested in the future.

The **Mobile** application purposes are to allow patients to communicate information about their health status and at the same time to remember them the therapy to follow.

The **Web** application offers the physician a way to keep under control all the registered patients from a unique platform, offering also real-time information. Moreover, it offers the possibility to retrieve a great amount of data that can be used in scientific researches.

Both platforms are still in Beta phase and they need to be tested on the field, with a reasonable number of users. Nevertheless, the applications provide all the required functionalities to start the test phase.

Appendix A

Web Application Code

Each page of the **Web Application** is defined by two different files: an **HTML** file and a **Javascript** file.

HTML file defines the page layout, describing aspect and positioning of each element composing it. **Javascript** (**js**) file describes the functionalities of the page itself.

Elements that have to be linked to a function, need to be called in the js file, so an **id** is given to them. In this way they can be retrieved using the **getElementById** method.

A.1 IndexWebApp

A.1.1 HTML

The following (Listing A.1) is the HTML code defining the layout of the starting page of the **Web** application.

```
<form style="display:block">
 <fieldset style="display:block">
  <label>
   <span style="display:inline-block;width:100%">
    <p style="display:inline-block;width:48%"> </p>
        
    <p id="warning"
    style="display:inline-block;width:48%;
    color:red;font-weight:bold;visibility:hidden">
    Attenzione! CapsLock attivato!</p>
   </span>
   <span style="display:inline-block;width:100%">
    <input class="mdl-textfield__input"
    style="display:inline;width:48%;font-size:140%"
    type="email" id="email" name="email"
    placeholder="Email"  />
        
    <input class="mdl-textfield__input"
    style="display:inline;width:48%;font-size:140%"
    type="password" id="password" name="password"
    placeholder="Password" />
   </span>
  </label>
 </fieldset>
</form>

<br/><br/>
<button class="mdl-button@@ mdl-js-button mdl-button--raised"
id="sign-in" name="signin">Accedi</button>
    
<button class="mdl-button@@ mdl-js-button mdl-button--raised"
id="database" name="database" disabled>Database</button>
    
<button class="mdl-button@@ mdl-js-button mdl-button--raised"
id="log-out" name="logout" disabled>Logout</button>
<div class="user-details-container">
 Firebase sign-in status: <span id="sign-in-status">Unknown</
    span>
</div>
<div class="user-details-container" style="visibility:hidden;">
 Firebase user: <span id="firebase-user-details"> null </span>
</div>
```

```
42 <div style="text-align:center">
43  <img src="heart1.jpg">
44  <p>Developed by</p>
45  <p>Politecnico di Torino</p>
46 </div>
```

Listing A.1: IndexWebApp HTML Code

A.1.2 Js

In every **Javascript** file there is the need to define **Firebase** configuration (Listing A.2). This allows to interface the application with it.

```
1 const config = {
2  apiKey: "AIzaSyAo9qf0ttRMaI_9gQdCFIMzHx6Eq8Y2JJk",
3  authDomain: "filoapplication-5c425.firebaseapp.com",
4  databaseURL: "https://filoapplication-5c425.firebaseio.com",
5  storageBucket: "filoapplication-5c425.appspot.com",
6 };
```

Listing A.2: Firebase Configuration Parameters

Then the **initializeApp** method is called (Listing A.3):

```
1 firebase.initializeApp(config);
```

Listing A.3: Firebase Initialization

In this page, a **caps lock** controller is enabled in the **password** field (Listing A.4):

```
1  txtPassword.onkeypress = function(e) {
2   if ((e.which >= 65 && e.which <= 90) ||
3       (e.which >= 97 && e.which <= 122)){
4    var str = String.fromCharCode(e.which);
5    capsLockEnabled = (str.toUpperCase() === str);
6    console.log("CapsLock enabled: "
7        + capsLockEnabled.toString());
8    if(capsLockEnabled){
9     warning.style.visibility = "visible";
10   }else{
11    warning.style.visibility = "hidden";
12   }
13  }
14 };
```

```
15
16 txtPassword.onkeydown = function(e) {
17  if (e.which == 20 && capsLockEnabled !== null) {
18   capsLockEnabled = !capsLockEnabled;
19   console.log("CapsLock enabled: "
20       + capsLockEnabled.toString());
21  } else if (e.which == 20) {
22   console.log("CapsLock initial state not set.");
23  }
24
25  if(capsLockEnabled){
26   warning.style.visibility = "visible";
27  }else{
28   warning.style.visibility = "hidden";
29  }
30 };
```

Listing A.4: Caps Lock Controller

The functionality of the **login button** is defined by the code in Listing A.5:

```
1 btnLogin.addEventListener('click', e => {
2  const email = txtEmail.value;
3  const pass = txtPassword.value;
4  const auth = firebase.auth();
5  const promise = auth.signInWithEmailAndPassword(email, pass);
6  promise.catch(function(error) {
7   console.log(error.message);
8   alert(error.message);
9  });
10 });
```

Listing A.5: IndexWebApp Login Button

When the user logs in, the page layout changes. The listener defined by the code in Listing A.6 waits for this change:

```
1 firebase.auth().onAuthStateChanged(firebaseUser => {
2  btnLogout.disabled = false;
3  if(firebaseUser) {
4   console.log(firebaseUser);
5   var query = firebase.database().ref("physicianUid")
       .orderByKey();
6   query.once("value").then(function() {
7    btnDatabase.disabled = false;
8    btnLogout.disabled = false;
9    btnLogin.disabled = true;
```

```
10      signInStatus.innerHTML = "Autenticato";
11      accountDetails.style.visibility = "visible";
12      accountDetails.innerHTML = firebase.auth()
13          .currentUser.email;
14    }).catch(function(error) {
15      var errorMessage = "Accesso negato! Non sei un medico, non
            puoi accedere al database!";
16      alert(errorMessage);
17      console.log(error);
18      signInStatus.innerHTML = "Accesso al database rifiutato";
19    });
20   } else {
21    btnDatabase.disabled = true;
22    btnLogout.disabled = true;
23    btnLogin.disabled = false;
24    signInStatus.innerHTML = "Non autenticato";
25    accountDetails.style.visibility = "hidden";
26   }
27 });
```

Listing A.6: IndexWebApp Listener

A.2 Home Page Code

A.2.1 HTML

The main elements of the **Home Page** are the two **divs** containing the two different lists for alerted and normal patients, **alert-patient-list-div** and **patient-list-div**, and the two buttons, **add** and **exit**, used to add a new patient and to exit the application. Listing A.7 defines their layout.

```
1  <p style="font-size:140%">Elenco Pazienti</p>
2  <p>Cliccare su un paziente per aprire la pagina relativa.</p>
3  <p>(In rosso i pazienti che da poco hanno comunicato un evento
       o che hanno inseriti dati fuori scala)</p>
4
5  <div id="alert-patient-list-div"></div>
6  <div id="patient-list-div"></div>
7
8  <br/><br/>
9  <button class="mdl-button mdl-js-button mdl-button--raised"
10 id="add" name="add" >Nuovo Paziente</button>
11    
12 <button class="mdl-button mdl-js-button mdl-button--raised"
13 id="exit" name="exit">Esci</button>
```

Listing A.7: Home Page Elements

A.2.2 Js

This page, as all others except the **Index** one, has a controller to verify if the user has physician permissions. If not, it redirects him/her to the **indexWebApp** page. The controller, defined by code in Listing A.8, is the same for all pages.

```
function verifyUserPermission() {
 var bool = false;
 var query = firebase.database().ref("physicianUid")
     .orderByKey();
 query.once("value").then(function(){
  bool = true;
 }).catch(function(error) {
  console.log(error);
 }).then(function(){
  if (bool){
   main.style.visibility = 'visible';
  } else {
   var errorMessage = "Accesso negato! Non sei un medico,
       non puoi accedere al database!";
   alert(errorMessage);
   firebase.auth().signOut();
   window.location.replace("indexWebApp.html");
  }
 });
}
```

Listing A.8: Permission Controller

The core of this page's code is the **populateList** function. This function checks the **Alert** field for each patient and, if it is **true**, adds him/her in the **alertList** instead that in the normal list (Listing A.9).

```
function populateList() {
 var patients = [];
 var list = document.createElement("ul");
 var alertList = document.createElement("ul");
 list.style = "list-style-type:none";
 var query = firebase.database().ref("patientsUid").orderByKey(
     );
 query.once("value").then(function(snapshot) {
  snapshot.forEach(function(childSnapshot){
   patients.push(childSnapshot);
  })
 }).then(function() {
  for(var i = 0; i<patients.length; i++) {
   var textContent = patients[i].val().Name + " - "
       + patients[i].val().FiscalCode;
```

```
15      var itemList = document.createElement("li");
16      var f = document.createElement("form");
17      var fs = document.createElement("fieldset");
18      var a = document.createElement("a");
19      f.style.display = "block";
20      f.style.border = "groove";
21      f.style.overflow = "hidden";
22      fs.style.display = "block";
23      fs.style.margin = "0.1%";
24      a.textContent = textContent;
25      a.setAttribute("href", "patientData.html" + "?"
26          + "patient=" + patients[i].key);
27      fs.appendChild(a);
28      f.appendChild(fs);
29      itemList.appendChild(f);
30      if ("Alert" in patients[i].val()){
31       if (patients[i].val().Alert == true){
32        alertList.appendChild(itemList);
33        a.style.color = "red";
34       } else {
35        list.appendChild(itemList);
36       }
37      } else {
38       list.appendChild(itemList);
39      }
40     }
41    });
42    divAlertPatients.appendChild(alertList);
43    divPatients.appendChild(list);
44  }
```

Listing A.9: Home Page - PopulateList Function

A.3 InsertNewPatient Page

A.3.1 HTML

The following HTML code (Listing A.10) defines the layout of **InsertNewPatient** page.

```
1  <p style="font-size:140%">Inserire i dati del paziente:</p>
2  <p style="font-size:140%" align="center" >ANAGRAFICA</p>
3
4  <form style="display:block;border:groove">
5   <fieldset style="display:block;margin:1%">
6    <label>
7     <input class="mdl-textfield__input" style="display:inline;
        width:50%;text-transform:uppercase" type="text" id="
        inName" placeholder="Nome *" />
```

```
8        
9     <input class="mdl-textfield__input" style="display:inline;
          width:47%;text-transform:uppercase" type="text" id="
          inSurname" placeholder="Cognome *" >
10   </label>
11  </fieldset>
12 </form><form style="display:block;border:groove">
13  <fieldset style="display:block;margin:1%">
14   <label>
15    <input class="mdl-textfield__input" style="
          display:inline;width:100%" type="email" id="inEmail"
          placeholder="Email *" />
16   </label>
17  </fieldset>
18 </form><form style="display:block;border:groove">
19  <fieldset style="display:block;margin:1%">
20   <label>
21    <input class="mdl-textfield__input" style="
          display:inline;width:100%;text-transform:uppercase" type=
          "text" maxlength="16" id="inCode" placeholder="Codice
          Fiscale *" />
22   </label>
23  </fieldset>
24 </form><form style="display:block;border:groove">
25  <fieldset style="display:block;margin:1%">
26   <span style="display: inline-block; width: 30%">DATA DI
        NASCITA *</span>
27      
28   <label>
29    <input class="mdl-textfield__input" style="
          display:inline;width:auto;" type="date" data-date-format=
          "DD MM YYYY" id="inBirthday" placeholder="DD-MM-YYYY"  />
30   </label>
31  </fieldset>
32 </form><form style="display:block;border:groove">
33  <fieldset style="display:block;margin:1%">
34   <label>
35    <input class="mdl-textfield__input" style="
          display:inline;width:100%;text-transform:uppercase" type=
          "text" id="inBirthplace" placeholder="Luogo di Nascita" /
          >
36   </label>
37  </fieldset>
38 </form><form style="display:block;border:groove">
39  <fieldset style="display:block;margin:1%">
40   <span style="display: inline-block; width: 30%">RESIDENZA</
        span>
41      
```

```
42    <label>
43     <input class="mdl-textfield__input" style="
          display:inline;width:30%;text-transform:uppercase" type="
          text" id="inResRoad" placeholder="Via" />
44      
45     <input class="mdl-textfield__input" style="
          display:inline;width:30%;text-transform:uppercase" type="
          text" id="inResCity" placeholder="Citt&agrave" />
46    </label>
47   </fieldset>
48  </form><form style="display:block;border:groove">
49   <fieldset style="display:block;margin:1%">
50    <label>
51     <input class="mdl-textfield__input" style="
          display:inline;width:100%;text-transform:uppercase" type=
          "text" id="inTelephone" placeholder="Telefono" />
52    </label>
53   </fieldset>
54  </form><form style="display:block;border:groove">
55   <fieldset style="display:block;margin:1%">
56    <span style="display:inline-block;width:30%">SESSO</span>
57       
58    <label>
59     <input type="radio" name="gender" id="gender" value="Maschio
          ">Uomo
60    </label>
61      
62    <label>
63     <input type="radio" name="gender" id="gender" value="Femmina
          ">Donna
64    </label>
65   </fieldset>
66  </form>
67  <p>(I parametri contrassegnati da * sono obbligatori)</p>
68
69  <br/><br/>
70  <button class="mdl-button mdl-js-button mdl-button--raised"
       type="submit" id="register" name="register">Registra</button
       >
71     
72  <button class="mdl-button mdl-js-button mdl-button--raised" id=
       "cancel" name="cancel" >Annulla</button>
```

Listing A.10: InsertNewPatient Page Elements

A.3.2 Js

To generate a new user, it is needed to define a dummy password. **GeneratePass** function, defined by code in Listing A.11, is used to do this.

```
1 function generatePass(){
2  var chars = "0123456789abcdefghijklmnopqrstuvwxyz-
      ABCDEFGHIJKLMNOPQRSTUVWXYZ";
3  var pass = "";
4  for (var i = 0; i < 7; i++) {
5   pass += chars[Math.floor(Math.random() * chars.length)];
6  }
7  return pass;
8 };
```

Listing A.11: GeneratePass Function

CheckData function (Listing A.12) is instead devoted to verify that all mandatory data have been inserted by the physician during registration. If the check is passed, **handleSignUp** function is called to perform sign up (Listing A.13).

```
1  function checkData() {
2   var ok = true;
3   var id = null;
4   var email = inEmail.value;
5   var name = inName.value;
6   var surname = inSurname.value;
7   var birthday = inBirthday.value;
8   var code = inCode.value;
9   if (!birthday){
10   ok = false;
11   id = "pBirthday";
12  }
13  if (!code){
14   ok = false;
15   id = "pCode";
16  }
17  if (!email){
18   ok = false;
19   id = "pEmail";
20  }
21  if (!surname){
22   ok = false;
23   id = "pSurname";
24  }
25  if (!name){
26   ok = false;
27   id = "pName";
28  }
29  if (!ok) {
```

```
30    var errorMessage = "Dati Mancanti!";
31    alert(errorMessage);
32    console.log(errorMessage);
33    document.getElementById(id).focus();
34   } else {
35    handleSignUp();
36   }
37  }
```

Listing A.12: CheckData Function

```
1   function handleSignUp(){
2    var email = inEmail.value;
3    var name = inName.value.toUpperCase();
4    var surname = inSurname.value.toUpperCase();
5    var birthday = inBirthday.value;
6    var code = inCode.value.toUpperCase();
7    var birthplace = inBirthplace.value.toUpperCase();
8    var resRoad = inResRoad.value.toUpperCase();
9    var resCity = inResCity.value.toUpperCase();
10   var telephone = inTelephone.value;
11   var inGender = document.querySelector("input[id='gender']:
        checked");
12   var gender = "";
13   var pass = generatePass();
14   var secondaryAuth = firebase
15       .initializeApp(config, "Secondary");
16   if (inGender){
17    gender = inGender.value;
18   }
19   secondaryAuth.auth().createUserWithEmailAndPassword(email,
        pass)
20       .then(function() {
21    var newUser = secondaryAuth.auth().currentUser;
22    newUser.updateProfile({email: email});
23    patient = newUser.uid;
24   }).then(function() {
25    firebase.auth().sendPasswordResetEmail(email).then(function()
         {
26     alert("Email inviata al paziente.");
27    }).then(function(){
28     var postData = {
29      DateOfBirth: birthday,
30      Email: email,
31      FiscalCode: code,
32      Name: name,
33      Surname: surname,
34      Gender: gender,
35      Birthplace: birthplace,
36      ResidencyRoad: resRoad,
```

```
37      ResidencyCity: resCity,
38      Telephone: telephone
39     };
40     var managementData = {
41      FiscalCode: code,
42      Name: name + " " + surname
43     };
44     firebase.database().ref("patients/" + patient + "/
          PersonalData").set(postData);
45     firebase.database().ref("patientsUid/" + patient).set
          (managementData);
46    }).then(function() {
47     console.log("Paziente registrato");
48     secondaryAuth.auth().signOut()
49    }).catch(function(error){
50     var errorMessage = error.message;
51     alert(errorMessage);
52     console.log(error);
53    }).then(function(){
54     secondaryAuth.delete();
55    }).then(function(){
56     var firstVisitMessage = "Paziente registrato con successo.\n
          \nProseguire con la prima visita?";
57     if (window.confirm(firstVisitMessage)){
58      window.location.replace("newVisit.html" + "?" + "patient="
           + patient);
59     } else {
60      window.location.replace("home.html");
61     }
62    });
63   });
64  }
```

Listing A.13: HandleSignUp Function

A.4 NewVisit Page

A.4.1 HTML

The first part of the page shows and allows to modify **Personal Data**. Each field is similar to the ones defined by the code in Listing A.14.

```
1  <p style="font-size:140%" align="center" >ANAGRAFICA</p>
2  <form class="edit-form" style="display: block; border: groove">
3   <fieldset style="display: block; margin: 1%">
4    <label>
5     <span style="display:inline-block;width:49%">
6      NOME*:  
7      <input class="mdl-textfield__input" style="
          display:inline;text-transform:uppercase;width:auto"
          type="text" id="inName" placeholder="Nome *" />
8     </span>
9     <span style="display:inline-block;width:47%">
10     COGNOME*:  
11     <input class="mdl-textfield__input" style="
          display:inline;text-transform:uppercase;width:auto"
          type="text" id="inSurname" placeholder="Cognome *" />
12    </span>
13   </label>
14  </fieldset>
15 </form>
16 [...]
```

Listing A.14: Personal Data Elements

Then, there is the part dedicated to **Risk Factors** (Listing A.15). Boolean data forms are similar to the one about **smoke** ("Fumatore"), the last form is about **allergies** ("Allergie").

```
1  [...]
2  <p style="font-size:140%" align="center">FATTORI DI RISCHIO</p>
3  <form class="edit-form" style="display:block;border:groove">
4   <fieldset style="display:block;margin:1%">
5    <span style="display:inline-block;width:49%">Fumatore</span>
6     <label>
7      <input type="radio" name="smoke" id="smokeY" value="SI">
8      S&igrave;
9     </label>
10      
11    <label>
12     <input type="radio" name="smoke" id="smokeN" value="NO">No
13    </label>
14   </fieldset>
15  </form>
16 [...]
```

```
17 [...]
18  <form class="edit-form" style="display:block;border:groove">
19   <fieldset style="display:block;margin:1%">
20    <span style="
        display:inline-block;vertical-align:top;width:49%">
        Allergie</span>
21    <span style="display:inline-block;width:49%">
22     <textarea style="width:100%;" rows="4" name="allergie" id="
         allergie"></textarea>
23    </span>
24   </fieldset>
25  </form>
26 [...]
```

Listing A.15: Risk Factors Elements

The next one, is the part dedicated to the **BMI**, defined by code in Listing A.16.

```
1  [...]
2  <p style="font-size:140%" align="center" >BMI</p>
3   <form class="edit-form" style="display:block;border:groove">
4    <fieldset style="display:block;margin:1%">
5     <span style="display:inline-block;width:50%">Altezza [cm]</
        span>
6        
7     <input type="number" name="altezza" id="altezza"> cm
8    </fieldset>
9   </form><form class="edit-form" style="
      display:block;border:groove">
10  <fieldset style="display:block;margin:1%">
11   <span style="display:inline-block;width:50%">Peso [Kg]</span>
12      
13   <input type="number" name="peso" id="peso"> Kg
14  </fieldset>
15 </form><form style="display:block;border:groove">
16  <fieldset style="display:block;margin:1%">
17   <span style="display:inline-block;width:50%">BMI [Kg/m<sup>2<
       /sup>]</span>
18      
19   <span id="text-bmi">0.00</span> [Kg/m<sup>2</sup>]
20  </fieldset>
21 </form>
22 [...]
```

Listing A.16: BMI Elements

The following code (Listing A.17) defines the part of the page dedicated to **Previous Acute Cardiovascular Events** and to the registration of the **Left Ventricular Ejection Fraction** value.

```
1  [...]
2  <p style="font-size:140%" align="center" >PREGRESSI EVENTI
       CORONARICI ACUTI</p>
3  <form class="edit-form" style="display:block;border:groove">
4   <fieldset style="display:block;margin:1%">
5    <span style="display:inline-block;width:50%">Pregresso STEMI<
       /span>
6       
7    <input type="radio" name="stemi" id="stemiY" value="SI">S&
       igrave;
8      
9    <input type="radio" name="stemi" id="stemiN" value="NO">No
10  </fieldset>
11 </form>
12 <form class="edit-form" style="display:block;border:groove">
13  <fieldset style="display:block;margin:1%">
14   <span style="display:inline-block;width:50%">Pregresso
       N-STEMI/ACS</span>
15      
16   <input type="radio" name="n-stemi" id="n-stemiY" value="SI">S
       &igrave;
17     
18   <input type="radio" name="n-stemi" id="n-stemiN" value="NO">
       No
19  </fieldset>
20 </form>
21
22 <br/><br/>
23 <p style="font-size:140%" align="center" >FRAZIONE DI EIEZIONE<
     /p>
24 <form class="edit-form" style="display:block;border:groove">
25  <fieldset style="display:block;margin:1%">
26   <span style="display:inline-block;width:50%">Funzione
       ventricolare sinistra</span>
27      
28   <input type="number" name="LVEF" id="lvef" min="30" max="90">
        %
29  </fieldset>
30 </form>
31 [...]
```

Listing A.17: Previous Cardiovascular Events and LVEF Elements

Code in Listing A.18 defines the form to add information about **angioplasties** and **still ill vessels**.

```
1  [...]
2  <p style="font-size:140%" align="center" id="angio-title">
       ANGIOPLASTICHE</p>
3  <form class="edit-form" style="display:block;border:groove" id=
       "angio-form">
4   <fieldset style="display:block;margin:1%">Data:
5    <input class="mdl-textfield__input" style="
         display:inline-block;width:auto;" type="date"
         data-date-format="DD MM YYYY" id="angio-date" placeholder=
         "DD-MM-YYYY"  />
6    <br/><br/>
7    <select style="display:inline-block;width:45%" id="
         vessel-select"></select>
8       
9    <select style="display:inline-block;width:15%" id="
         revasc-select" disabled="true"></select>
10      
11   <input type="radio" name="angio-status" id="angio-status"
         value="Malato" checked="true">Malato
12   <input type="radio" name="angio-status" id="angio-status"
         value="Trattato">Trattato
13   <button style="float:right" type="button" id="add-angio">
         Aggiungi</button>
14  </fieldset>
15 </form>
16 [...]
```

Listing A.18: Angioplasty Elements

Below the **angioplasties** form, the part of the page dedicated to patient's **history** begins. These forms are similar to the "**Anamnesi Internistica**" one, defined by code in Listing A.19. These parts of the page include also a section to insert **systolic** and **diastolic pressure** ("Pressione Sistolica" and "Pressione Diastolica") and **heart rate** ("Frequenza Cardiaca"). The last form defined in these Listing is about **ECG**, that is different from other history forms, because it also contains an **input** element for files uploading (*id="add-ecg"*).

```
1  [...]
2  <p style="font-size:140%" align="center" id="anamnesi-title">
       ANAMNESI</p>
3  <form class="edit-form" style="display:block;border:groove">
4   <fieldset style="display:block;margin:1%">
5    <span style="display:inline-block;vertical-align:top;
6        width:25%">Anamnesi Internistica</span>
```

```
7       
8    <textarea style="width:72%;" rows="10" type="text" name="
        internistica" id="internistica"></textarea>
9   </fieldset>
10 </form>
11 [...]
12 <form class="edit-form" style="display:block;border:groove">
13  <fieldset style="display:block;margin:1%">
14   <span style="display:inline-block;width:50%">Pressione
        Sistolica [mmHg]</span>
15      
16   <input type="number" style="display:inline-block;width:20%"
        name="pressure" id="syst" min="60" max="270"> mmHg
17   <br/><br/>
18   <span style="display:inline-block;width:50%">Pressione
        Diastolica [mmHg]</span>
19      
20   <input type="number" style="display:inline-block;width:20%"
        name="pressure" id="diast" min="20" max="150"> mmHg
21  </fieldset>
22 </form><form class="edit-form"
23     style="display:block;border:groove">
24  <fieldset style="display:block;margin:1%">
25   <span style="display:inline-block;width:50%">Frequenza
        Cardiaca [bpm]</span>
26      
27   <input type="number" style="display:inline-block;width:20%"
        name="freq" id="freq" min="20" max="220"> bpm
28  </fieldset>
29 </form><form class="edit-form"
30     style="display:block;border:groove">
31  <fieldset style="display:block;margin:1%">
32   <span style="display:inline-block;vertical-align:top;
33       width:25%">ECG
34   <br/><br/>
35   <input type="file" id="add-ecg" multiple ></input>
36   </span>
37      
38   <textarea style="width:72%;" rows="10" type="text"
39       name="ecg-text" id="ecg-text"></textarea>
40  </fieldset>
41 </form>
42 [...]
```

Listing A.19: History Elements

The next form is dedicated to insert **therapies** (Listing A.20).

```
1  [...]
2  <p style="font-size:140%" align="center" id="therapy-title">
       TERAPIA</p>
3  <form class="edit-form" style="display:block;border:groove" id=
       "therapy-form">
4   <fieldset style="display:block;margin:1%">
5    <input type="text" style="width:30%" name="drug" id="drug"
         placeholder="FARMACO">
6    <br/><br/>
7    Giorni:
8    <input type="radio" class="default-radio-day" name="radio-day
         " id="radio-day" value="all" checked>Tutti i giorni
9      
10   <input type="radio" name="radio-day" id="radio-day" value="
         select">Seleziona i giorni
11   <br/><br/>
12   <input type="checkbox" class="day" id="mon" value="lun"
         disabled checked>Luned&igrave;
13   <input type="checkbox" class="day" id="tue" value="mar"
         disabled checked>Marted&igrave;
14   <input type="checkbox" class="day" id="wed" value="mer"
         disabled checked>Mercoled&igrave;
15   <input type="checkbox" class="day" id="thu" value="gio"
         disabled checked>Gioved&igrave;
16   <input type="checkbox" class="day" id="fri" value="ven"
         disabled checked>Venerd&igrave;
17   <input type="checkbox" class="day" id="sat" value="sab"
         disabled checked>Sabato
18   <input type="checkbox" class="day" id="sun" value="dom"
         disabled checked>Domenica
19   <br/><br/>
20   Orari:
21   <select style="display:inline-block;width:auto" name="
         time-select" id="time1"></select> 
22   <select style="display:inline-block;width:auto" name="
         time-select" id="time2"></select> 
23   <select style="display:inline-block;width:auto" name="
         time-select" id="time3"></select> 
24   <select style="display:inline-block;width:auto" name="
         time-select" id="time4"></select> 
25   <select style="display:inline-block;width:auto" name="
         time-select" id="time5"></select>
26   <br/><br/>
27   <input type="radio" name="meal" id="meal" value="dp"> Dopo i
         pasti
28     
29   <input type="radio" name="meal" id="meal" value="sv"> A
         stomaco vuoto
```

```
30      
31    <input type="radio" name="meal" id="meal" value="ind"
          checked="true"> Indifferente
32    <br/><br/>
33    <input type="checkbox" id="till">Fino a: 
34    <input class="mdl-textfield__input" style="
          display:inline-block;width:auto;" type="date"
          data-date-format="DD MM YYYY" id="therapy-end"
          placeholder="DD-MM-YYYY" disabled/>
35    <button style="float:right" type="button" id="add-therapy">
          Aggiungi</button>
36   </fieldset>
37  </form>
38  [...]
```

Listing A.20: Therapy Elements

At the end of the page, there is the form used to represent **events**, and below it, there are the two buttons to register the visit and conclude it. (Listing A.21)

```
1  [...]
2  <p style="font-size:140%" align="center" id="events-title">
       EVENTI REGISTRATI</p>
3
4  <br/><br/>
5  <button class="mdl-button mdl-js-button mdl-button--raised"
       type="submit" id="register" name="register">Registra Visita<
       /button>
6
7  <button style="float:right" class="mdl-button mdl-js-button
       mdl-button--raised" id="conclude" name="conclude" >Concludi
       Visita</button>
```

Listing A.21: Event Elements and End Page Buttons

A.4.2 Js

Except for the first visit, **NewVisit** page is populated with information stored in the database during previous examinations. **PopulatePersonalData** function is used to populate fields containing patient's personal data (Listing A.22). All *populate* functions are similar to this one: they queries the database for data and populate fields with them. The variable *patient* contains the **Uid** of the patient.

```
function populatePersonalData() {
 var completeResidency = "";
 var query = firebase.database().ref("patients/" + patient +
     "/PersonalData").orderByKey();
 query.once("value").then(function(snapshot) {
  if (snapshot.hasChild("Name")){
   inName.value = snapshot.val().Name;
  }
  if (snapshot.hasChild("Surname")){
   inSurname.value = snapshot.val().Surname;
  }
  if (snapshot.hasChild("Email")){
   email.textContent = snapshot.val().Email;
  }
  if (snapshot.hasChild("FiscalCode")){
   inCode.value = snapshot.val().FiscalCode;
  }
  if (snapshot.hasChild("DateOfBirth")){
   inBirthday.value = snapshot.val().DateOfBirth;
  }
  if (snapshot.hasChild("Birthplace")){
   inBirthplace.value = snapshot.val().Birthplace;
  }
  if (snapshot.hasChild("ResidencyCity")){
   inResCity.value = snapshot.val().ResidencyCity;
  }
  if (snapshot.hasChild("ResidencyRoad")){
   inResRoad.value = snapshot.val().ResidencyRoad;
  }
  if (snapshot.hasChild("Telephone")){
   inTelephone.value = snapshot.val().Telephone;
  }
  if (snapshot.hasChild("Gender")){
   if (snapshot.val().Gender == 'Maschio'){
    valGenderM.checked = true;
   } else if (snapshot.val().Gender == 'Femmina'){
    valGenderF.checked = true;
   }
  }
 });
}
```

Listing A.22: PopulatePersonalData Function

Function to populate **angioplasties**, **therapies** and **Events** are slightly different. In these cases in fact, there are not some fields to fill up, but they have to be created based on the number of element present in the database. Code in Listing A.23 defines the **populateAngioplasties** function. The other two are of the same type. As in the previous function, *patient* variable contains the patient's **Uid**. In addition to the fields creation, these functions also save these elements in local variables, so that they can be deleted or update by the physician (*patientAngioList* for this example).

```
1  function populateAngioplasties() {
2   var query = firebase.database().ref("patients/" + patient + "/
       Angioplasties").orderByKey();
3   query.once("value").then(function(snapshot){
4    if (snapshot.val()) {
5     var index = 0;
6     snapshot.forEach(function(childSnapshot) {
7      var dateOldFormat = childSnapshot.val().date.split("-");
8      var dateNewFormat = dateOldFormat[2] + "/" + dateOldFormat
          [1] + "/" + dateOldFormat[0];
9      var f = document.createElement("form");
10     var fs = document.createElement("fieldset");
11     var d = document.createElement("div");
12     var b = document.createElement("button");
13     f.className = "edit-form";
14     f.style.display = "block";
15     f.style.border = "groove";
16     f.style.overflow = "hidden";
17     f.id = "f-" + String(index);
18     fs.style.display = "block";
19     fs.style.margin = "1%";
20     d.style.display = "inline-block";
21     d.style.textAlign = "left";
22     d.style.width = "100%";
23     if (childSnapshot.val().status = "Malato"){
24      d.textContent = dateNewFormat + ";\xa0\xa0\xa0" +
           childSnapshot.val().vessel + ";\xa0\xa0\xa0" +
           childSnapshot.val().status;
25     } else {
26      d.textContent = dateNewFormat + ";\xa0\xa0\xa0" +
           childSnapshot.val().vessel + ";\xa0\xa0\xa0" +
           childSnapshot.val().revasc + ";\xa0\xa0\xa0" +
           childSnapshot.val().status;
27     }
28     b.style.float = "right";
29     b.type = "button";
30     b.textContent = "Elimina";
31     b.id = index;
32     b.addEventListener('click', deleteAngioButton, false);
```

```
33      angioListElement = {
34       date: childSnapshot.val().date,
35       vessel: childSnapshot.val().vessel,
36       revasc:  childSnapshot.val().revasc,
37       status: childSnapshot.val().status
38      }
39      patientAngioList.push(angioListElement);
40      document.getElementById("angio-title").after(f);
41      f.appendChild(fs);
42      fs.appendChild(d);
43      d.appendChild(b);
44      index++;
45     });
46    }
47   });
48  }
```

Listing A.23: PopulateAngioplasties Function

In addition to the population, data as **therapies** and **angioplasties** must be editable by the physician. Function **addTherapy** in Listing A.24 is devoted to generate a new **therapyListElement**, to add it to the **patientTherapyList** and on the page view. Moreover, it controls that all mandatory fields have been filled up. A similar function has been defined for angioplasties.

```
1  function addTherapy() {
2   var buttonId = patientTherapyList.length;
3   var therapyDrug = inDrug.value;
4   var therapyDaysRadio = document.querySelector("input[id='radio
       -day']:checked").value;
5   var therapyDays = "";
6   var therapyTimesList = [];
7   var therapyTimes = "";
8   var therapyMeal = document.querySelector("input[id='meal']:
       checked").value;
9   var therapyEnd = "-";
10  var therapyTextContent = "";
11  var therapyListElement = {};
12  var f = document.createElement("form");
13  var fs = document.createElement("fieldset");
14  var d = document.createElement("div");
15  var b = document.createElement("button");
16  if (!therapyDrug){
17   alert("Inserire il nome del farmaco");
18   return
19  }
20  therapyTextContent = therapyTextContent + therapyDrug + ";\xa0
       \xa0\xa0Assunzione:\xa0";
```

```
21  if (therapyDaysRadio == "all"){
22   therapyDays = "lun,mar,mer,gio,ven,sab,dom";
23   therapyTextContent = therapyTextContent + "Tutti i giorni;\
        xa0\xa0\xa0";
24  } else if (therapyDaysRadio == "select"){
25   for (i=0; i<checkDayList.length; i++){
26    if (checkDayList[i].checked){
27     therapyDays = therapyDays + checkDayList[i].value + ",";
28    }
29   }
30   therapyDays = therapyDays.substring(0, therapyDays.length -
        1);
31   therapyTextContent = therapyTextContent + therapyDays + ";\
        xa0\xa0\xa0";
32   }
33   if (!therapyDays){
34    console.log(therapyDays);
35    alert("Inserire almeno un giorno per l'assunzione del
         farmaco");
36    return
37   }
38   timeSelectList.forEach(function(timeSelect) {
39    if (timeSelect.options[timeSelect.selectedIndex].text != "")
         {
40     therapyTimesList.push(timeSelect.options[timeSelect
          .selectedIndex].text);
41    }
42   });
43   if (therapyTimesList.length == 0){
44    alert("Inserire almeno un orario per l'assunzione del
         farmaco");
45    return
46   }
47   therapyTimesList.sort();
48   therapyTimesList.forEach(function(timeNumber) {
49    therapyTimes = therapyTimes + timeNumber + ",";
50   });
51   therapyTimes = therapyTimes.substring(0, therapyTimes.length
        - 1);
52   therapyTextContent = therapyTextContent + therapyTimes;
53   if (therapyMeal == "dp"){
54    therapyTextContent = therapyTextContent + "\xa0\xa0\xa0Dopo
         i pasti";
55   } else if (therapyMeal == "sv"){
56    therapyTextContent = therapyTextContent + "\xa0\xa0\xa0A
         stomaco vuoto";
57   }
58   if (therapyEndCheck.checked){
59    therapyEnd = therapyEndDate.value;
60    if (!therapyEnd){
```

```
61      alert("Inserire la data di fine terapia o deselezionare il
            campo.");
62      return
63     }
64    if (therapyEnd < dateDB){
65     console.log("end:",therapyEnd);
66     console.log("min:",dateDB);
67     alert("Impossibile selezionare una data passata come fine
           della terapia.");
68     return
69    }
70    therapyTextContent = therapyTextContent + ";\xa0\xa0\xa0Fino
          al\xa0" + therapyEnd;
71   }
72   f.className = "edit-form";
73   f.style.display = "block";
74   f.style.border = "groove";
75   f.style.overflow = "hidden";
76   f.id = "fther-" + String(buttonId);
77   fs.style.display = "block";
78   fs.style.margin = "1%";
79   d.style.display = "inline-block";
80   d.style.textAlign = "left";
81   d.style.width = "100%";
82   d.textContent = therapyTextContent;
83   b.style.float = "right";
84   b.type = "button";
85   b.textContent = "Elimina";
86   b.id = "ther-" + String(buttonId);
87   b.addEventListener('click', deleteTherapyButton, false);
88   therapyListElement = {
89    drug: therapyDrug,
90    assumptionDays: therapyDays,
91    assumptionHours: therapyTimes,
92    infoMeal: therapyMeal,
93    end: therapyEnd
94   }
95   patientTherapyList.push(therapyListElement);
96   document.getElementById("therapy-title").after(f);
97   f.appendChild(fs);
98   fs.appendChild(d);
99   d.appendChild(b);
100 }
```

Listing A.24: AddTherapy Function

The following function (Listing A.25) is devoted to evaluate the **BMI** value based on the inserted values of **weight** and **height**, and to populate the related field with the computed value.

```
1 function updateBMI() {
2 txtBMI.textContent = (inWeight.value/((inHeight.value/100.0)*
      (inHeight.value/100.0))).toFixed(2);
3 }
```

Listing A.25: UpdateBMI Function

The next functions (Listing A.26) are used to adapt **<textarea>** fields according to the amount of text they contain.

```
1  function adjustTextArea(){
2   var tx = document.getElementsByTagName("textarea");
3   for (var i = 0; i < tx.length; i++) {
4    tx[i].style.height = tx[i].scrollHeight + "px";
5    tx[i].style.overflowY = "hidden";
6    tx[i].addEventListener("input", adjust, false);
7   }
8  }
9  function adjust() {
10  this.style.height = "auto";
11  this.style.height = (this.scrollHeight) + "px";
12 }
```

Listing A.26: AdjustTextArea and Adjust Functions

The following function (Listing A.27) is called to register the examination. It is called after the **checkData** function (see Listing A.12). It reads all data inserted in the page and saves them into the database, it generates the **examination prospectus** ("Prospetto"), downloading it and saving it into the **cloud storage**.

```
1  function handleNewVisitData() {
2   var name = inName.value.toUpperCase();
3   var surname = inSurname.value.toUpperCase();
4   var birthday = inBirthday.value;
5   var code = inCode.value.toUpperCase();
6   var birthplace = inBirthplace.value.toUpperCase();
7   var resRoad = inResRoad.value.toUpperCase();
8   var resCity = inResCity.value.toUpperCase();
9   var telephone = inTelephone.value;
10  var inGender = document
11      .querySelector("input[name='gender']:checked");
```

```
12  var gender = "";
13  if (inGender){
14   gender = inGender.value;
15  }
16  var postData = {
17   DateOfBirth: birthday,
18   FiscalCode: code,
19   Name: name,
20   Surname: surname,
21   Gender: gender,
22   Birthplace: birthplace,
23   ResidencyRoad: resRoad,
24   ResidencyCity: resCity,
25   Telephone: telephone
26  };
27  var personalDataText = "Paziente: " + name + " " + surname +
28  "\n" + "Nato a " + birthplace + " il " +
29  birthday.split('-')[2] + "/" + birthday.split('-')[1] + "/" +
30  birthday.split('-')[0] + "\n" + "Codice Fiscale: " + code +
31  "\n" + "Residente in " + resCity + ", " + resRoad + "\n" +
32  "Telefono: " + telephone + "\nE-mail: " + email.innerHTML;
33  var y = 20;
34  firebase.database().ref("patients/" + patient + "/PersonalData
       ").update(postData);
35  Prospetto.setFontType("bold");
36  Prospetto.text(105, y, "Azienda Ospedaliera Citt\xE0 della
       Salute e della Scienza di Torino", null, null, 'center');
37  Prospetto.text(105, y+=20, "Cardiologia Universitaria", null,
       null, 'center');
38  Prospetto.text(105, y+=10, "(Direttore Prof. F. Gaita)", null,
        null, 'center');
39  Prospetto.text(105, y+=5, "Corso Bramante, 88 - 10126 TORINO",
        null, null, 'center');
40  Prospetto.text(105, y+=10, "AMBULATORIO DIVISIONALE", null,
       null, 'center');
41  Prospetto.setFontType('normal');
42  Prospetto.text(105, y+=5,"Telefono 011/6335538", null, null, '
       center');
43  Prospetto.setFontSize(12);
44  Prospetto.text(20, y+=10, personalDataText);
45 //-------------------------------------------------------------
46  var riskFactors = {};
47  var status = {};
48  var angioplasties = {};
49  var anamnesis = {};
50  var therapies = {};
51  var generalInfoText = '';
52  var anamnesisText = [];
53  var smoke = document
        .querySelector("input[name='smoke']:checked");
54
```

```
55  var ipert = document
56      .querySelector("input[name='ipert']:checked");
57  var dislip = document
58      .querySelector("input[name='dislip']:checked");
59  var diab = document
60      .querySelector("input[name='diabete']:checked");
61  var fam = document
62      .querySelector("input[name='familiarita']:checked");
63  var chemio = document
64      .querySelector("input[name='chemio']:checked");
65  var radio = document
66      .querySelector("input[name='radio']:checked");
67  var hiv = document
68      .querySelector("input[name='hiv']:checked");
69  var insRen = document
70      .querySelector("input[name='ins-ren']:checked");
71  var stemi = document
72      .querySelector("input[name='stemi']:checked");
73  var nstemi = document
74      .querySelector("input[name='n-stemi']:checked");
75  if (smoke){
76   riskFactors.Smoker = smoke.value;
77  }
78  if (ipert){
79   riskFactors.Hypertension = ipert.value;
80  }
81  if (dislip){
82   riskFactors.Dyslipidemia = dislip.value;
83  }
84  if (diab){
85   riskFactors.Diabetic = diab.value;
86  }
87  if (fam){
88   riskFactors.Familiarity = fam.value;
89  }
90  if (chemio){
91   riskFactors.PreviousChemio = chemio.value;
92  }
93  if (radio){
94   riskFactors.PreviousRadio = radio.value;
95  }
96  if (hiv){
97   riskFactors.HIVinTherapy = hiv.value;
98  }
99  if (insRen){
100  riskFactors.KidneyFailure = insRen.value;
101 }
102 if (valAllergies){
103  riskFactors.Allergies = valAllergies.value;
104 }
```

```
105   if (inWeight.value != "") {
106    status.Weight = inWeight.value;
107   }
108   if (inHeight.value != "") {
109    status.Height = inHeight.value;
110   }
111   if (stemi){
112    status.STEMI = stemi.value;
113   }
114   if (nstemi){
115    status.NSTEMIACS = nstemi.value;
116   }
117   if (valLVEF.value != ""){
118    status.LVEF = valLVEF.value;
119   }
120   status.BMI = txtBMI.innerHTML;
121   firebase.database().ref("patients/" + patient +
122       "/RiskFactors").update(riskFactors);
123   firebase.database().ref("patients/" + patient + "/Status")
124       .update(status);
125   generalInfoText = "Peso: " + inWeight.value + "kg\tAltezza: "
        + inHeight.value + "cm\tBMI: " + txtBMI.innerHTML + "\n";
126  //-------------------------------------------------------------
127   for (angioListElement in patientAngioList) {
128    angioplasties[angioListElement] = patientAngioList[
         angioListElement];
129   }
130   firebase.database().ref("patients/" + patient +
131       "/Angioplasties").set(angioplasties);
132  //-------------------------------------------------------------
133   if (sInternistica.value != ""){
134    anamnesis.Internistica = sInternistica.value;
135    anamnesisText.push("ANAMNESI INTERNISTICA\n" + sInternistica
         .value + "\n\n");
136   }
137   if (sCardiologica.value != ""){
138    var cardiologica = sCardiologica.value;
139    if (sConclusioni.value != ""){
140     var txtCon = "\n\nConclusioni " + date;
141     txtCon = txtCon + "\n\n" + conclusioni.value;
142     cardiologica = cardiologica + "\n" + txtCon;
143    }
144    anamnesis.Cardiologica = cardiologica;
145    anamnesisText.push("ANAMNESI CARDIOLOGICA\n" + cardiologica +
         "\n\n");
146   }
147   if (sEcg.value != ""){
148    anamnesis.Ecg = sEcg.value;
149    anamnesisText.push("ECG\n" + sEcg.value + "\n\n");
150   }
```

```
151   if (sOdierna.value != ""){
152    var pressure = "";
153    var pressureDB = {
154     Maximum: "",
155     Minimum: ""
156    };
157    var freq = "";
158    var freqDB = {HeartRate: ""};
159    if (inSyst.value != "" && inDiast.value != ""){
160     var syst = inSyst.value;
161     var diast = inDiast.value;
162     pressure = "Pressione: " + syst + "/" + diast + " mmHg\n";
163     pressureDB.Maximum = syst;
164     pressureDB.Minimum = diast;
165     firebase.database().ref('data/' + patient + '/Pressure/' +
          dateDB).set(pressureDB);
166    }
167    if (inFreq.value != ""){
168     var hr = inFreq.value;
169     freq = "Frequenza Cardiaca: " + hr + "bpm\n";
170     freqDB.HeartRate = hr;
171     firebase.database().ref("data/" + patient + "/HeartRate/" +
          dateDB).set(freqDB);
172    }
173    anamnesis.Odierna = date + "\n" + pressure + "\n" + freq + "\
       n" + sOdierna.value;
174    anamnesisText.push("VISITA ATTUALE\n" + date + "\n" +
          pressure + "\n" + freq +
175       "\n" + sOdierna.value + "\n\n");
176   }
177   if (sConsigli.value != ""){
178    anamnesis.Consigli = sConsigli.value;
179    anamnesisText.push("SI CONSIGLIA\n" + sConsigli.value +
180        "\n\n");
181   }
182   firebase.database().ref("patients/" + patient + "/Anamnesis")
        .update(anamnesis);
183  //-------------------------------------------------------------
184   for (therapyListElement in patientTherapyList) {
185    therapies[therapyListElement] = patientTherapyList[
          therapyListElement];
186   }
187   firebase.database().ref("patients/" + patient + "/Therapies")
        .set(therapies);
188  //-------------------------------------------------------------
189   for (ev in eventsToRemove) {
190    firebase.database().ref("events/" + patient + "/" +
          eventsToRemove[ev]).remove();
191   }
192  //-------------------------------------------------------------
```

```
193   if (ecgFileList){
194    for (i=0; i<ecgFileList.length; i++){
195     var file = ecgFileList.item(i);
196     var fileNameKey = file.name.replace(".","*");
197     var fileURL = patient + "/ecg/" + dateDB + "/" + file.name;
198     var fileRef = storageRef.child(fileURL);
199     fileRef.put(file);
200     firebase.database().ref("patients/" + patient +
201         "/EcgURL").update({[fileNameKey]: dateDB});
202    }
203   }
204  //-------------------------------------------------------------
205   Prospetto.text(20, y+=30, generalInfoText);
206   y+=5;
207   var pageHeight= Prospetto.internal.pageSize.height;
208   console.log(pageHeight);
209   for (i=0; i<anamnesisText.length; i++){
210    var textArray = Prospetto
211        .splitTextToSize(anamnesisText[i],170);
212    for (j=0; j<textArray.length; j++){
213     var lineHeight = Prospetto
214         .getTextDimensions(textArray[j]).h;
215     console.log(lineHeight);
216     console.log(textArray[j]);
217     if (y + lineHeight >= pageHeight){
218      Prospetto.addPage();
219      y = 20;
220     }
221    Prospetto.text(20, y+=5, textArray[j]);
222    }
223   }
224   var blob = Prospetto.output('blob');
225   var prospettoName = inSurname.value + "_" + inName.value + "_"
        + inCode.value + "_" + dateDB + ".pdf";
226   var prospettoNameKey = prospettoName.replace(".","*");
227   var visitRef = storageRef.child(patient + "/visit/" + dateDB +
        "/" + prospettoName);
228   visitRef.put(blob);
229   firebase.database().ref("patients/" + patient +
230      "/OldExaminationURL").update({[prospettoNameKey]: dateDB});
231   registered = true;
232   disableModification();
233   Prospetto.save(prospettoName);
234  }
```

Listing A.27: HandleNewVisitData Function

After this, the **disableModification** function is called (Listing A.28).

```
1  function disableModification(){
2   var forms = document.getElementsByClassName("edit-form");
3   for (i=0; i<forms.length; i++){
4    var elements = forms[i].elements;
5    for (j=0; j<elements.length; j++) {
6     elements[j].readOnly = true;
7     elements[j].disabled = true;
8    }
9   }
10  btnRegister.disabled = true;
11 }
```

Listing A.28: DisableModification Function

Then, by clicking on the **Conclude Visit** button, the **concludeVisit** function is called (Listing A.29).

```
1  function concludeVisit(){
2   if (registered){
3    var message = "Visita conclusa";
4    alert(message);
5    window.open("patientData.html" + "?" + "patient=" + patient,
        '_self', 'false');
6   } else {
7    var message = "La visita non \xE8 stata registrata,
        concluderla comunque?";
8    if (window.confirm(message)){
9     window.open("patientData.html" + "?" + "patient=" + patient,
         '_self', 'false');
10   }
11  }
12 }
```

Listing A.29: ConcludeVisit Function

A.5 PatientOverview Page

A.5.1 HTML

The following HTML code (Listing A.30) defines the containers of graphs shown in the **PatientOverview** page and their position.

```
<p style="font-size:140%" id="patient-title"></p>
<p align="right">
 <button class="mdl-button mdl-js-button mdl-button--raised"
    id="database" name="database" >Torna al Database</button>
</p>
<p style="font-size:140%" align="center">DATI NUMERICI</p>
<form class="edit-form" style="display:block;border:groove">
 <fieldset style="display: block; margin: 1%">
  Visualizzare dati dal 
  <input class="mdl-textfield__input" style="
     display:inline-block;width:auto;" type="date"
     data-date-format="DD MM YYYY" id="start" placeholder="
     DD-MM-YYYY"/>
   al 
  <input class="mdl-textfield__input" style="
     display:inline-block;width:auto;" type="date"
     data-date-format="DD MM YYYY" id="end" placeholder="
     DD-MM-YYYY" />
  <br/><br/>
  Dati disponibili dal 
  <div style="display:inline-block;width:auto;" id="first-entry
     "></div>
   al 
  <div style="display:inline-block;width:auto;" id="last-entry"
     ></div>
  <button style="float:right" type="button" id="set">Imposta</
     button>
 </fieldset>
</form>
<div style="margin:auto;width:95%">
 <canvas id="line-chart" width="auto" height="auto" ></canvas>
 <br/><br/>
</div>
<p style="font-size:140%" align="center">FUMO</p>
<div style="margin:auto;width:80%">
 <canvas id="pie-chart-smoke" width="auto" height="auto" ></
    canvas>
 <br/><br/>
</div>
<p style="font-size:140%;" align="center">TERAPIA</p>
<div style="margin:auto;width:80%">
```

```
31   <canvas id="pie-chart-therapy" width="auto" height="auto" ></
        canvas>
32   <br/><br/>
33  </div>
34  <p style="font-size:140%" align="center" >ATTIVIT&Agrave;&
       nbsp;FISICA</p>
35  <div style="margin:auto;width:80%">
36   <canvas id="bar-chart" width="auto" height="auto"></canvas>
37  </div>
38  <br/><br/>
39  <p>
40   <button class="mdl-button mdl-js-button mdl-button--raised"
        id="back" name="back" >Indietro</button>
41  </p>
```

Listing A.30: PatientOverview Page Elements

A.5.2 Js

The following function (Listing A.31) is devoted to populate the canvas containing the line chart. It retrieves data from the database, checks the larger time interval for which a datum is available and gives **null** values to each data when it is not available for a given time. In this way the graph has not gaps. Then, it sets the time interval on the graph equals to 15 days the first time, then equal to the one manually set by the physician. This function, in fact, is called once when the page is opened and then each time the **btnSet** is pressed (Listing A.30, *id=set*).

Other functions used to populate graphs are of the same type of this, but they populate **pie** and **bar** charts.

```
1  function populateLineCanvas(){
2   var maximum = [];
3   var orderedMax = [];
4   var minimum = [];
5   var orderedMin = [];
6   var glycemia = [];
7   var orderedGly = [];
8   var heartRate = [];
9   var orderedHR = [];
10  var weight = [];
11  var orderedWei = [];
12  var t = [];
13  var newT = [];
14  var minDate = new Date(startLineChart.value.replace("-",","));
15  var maxDate = new Date (endLineChart.value.replace("-",","));
16  var date1;
17  var date2;
18  var dateTitle1;
19  var dateTitle2;
```

```
20  var pressureQuery = firebase.database().ref("data/" + uId +
        "/Pressure").orderByKey();
21  var glycemiaQuery = firebase.database().ref("data/" + uId +
        "/Glycemia").orderByKey();
22  var heartRateQuery = firebase.database().ref("data/" + uId +
        "/HeartRate").orderByKey();
23  var weightQuery = firebase.database().ref("data/" + uId +
        "/Weight").orderByKey();
24  pressureQuery.once("value").then(function(snapshot) {
25   if (snapshot.val()) {
26    snapshot.forEach(function(childSnapshot) {
27     t.push(childSnapshot.key);
28     maximum[childSnapshot.key] = childSnapshot.val().Maximum;
29     minimum[childSnapshot.key] = childSnapshot.val().Minimum;
30    });
31   }
32  }).then(function(){
33   glycemiaQuery.once("value").then(function(snapshot) {
34    if (snapshot.val()) {
35     snapshot.forEach(function(childSnapshot) {
36      if (t.indexOf(childSnapshot.key) === -1){
37       t.push(childSnapshot.key);
38      }
39      glycemia[childSnapshot.key] = childSnapshot.val()
            .Glycemia;
40     });
41    }
42   }).then(function(){
43    heartRateQuery.once("value").then(function(snapshot) {
44     if (snapshot.val()) {
45      snapshot.forEach(function(childSnapshot) {
46       if (t.indexOf(childSnapshot.key) === -1){
47        t.push(childSnapshot.key);
48       }
49       heartRate[childSnapshot.key] = childSnapshot.val()
            .HeartRate;
50      });
51     }
52    }).then(function(){
53     weightQuery.once("value").then(function(snapshot) {
54      if (snapshot.val()) {
55       snapshot.forEach(function(childSnapshot) {
56        if (t.indexOf(childSnapshot.key) === -1){
57         t.push(childSnapshot.key);
58        }
59        weight[childSnapshot.key] = childSnapshot.val().Weight;
60       });
61      }
62     }).then(function(){
63      var maxKeys = [];
```

```
64        var minKeys = [];
65        var glyKeys = [];
66        var hrKeys = [];
67        var weiKeys = [];
68        var firstDate = new Date(t.sort()[0].replace("-",","));
69        var lastDate = new Date(t.sort()[t.length-1]
              .replace("-",","));
70        date1 = ("0" + firstDate.getDate()).slice(-2) + "/" +
              ("0" +  (firstDate.getMonth()+1)).slice(-2) + "/" +
              firstDate.getFullYear();
71        date2 = ("0" + lastDate.getDate()).slice(-2) + "/" +
              ("0" + (lastDate.getMonth()+1)).slice(-2) + "/" +
              lastDate.getFullYear();
72        firstEntry.textContent = date1;
73        lastEntry.textContent = date2;
74        if (minDate < firstDate){
75         minDate = firstDate;
76        }
77        if (maxDate > lastDate){
78         maxDate = lastDate;
79        }
80        dateTitle1 = ("0" + minDate.getDate()).slice(-2) + "/" +
              ("0" + (minDate.getMonth()+1)).slice(-2) + "/" +
              minDate.getFullYear();
81        dateTitle2 = ("0" + maxDate.getDate()).slice(-2) + "/" +
              ("0" + (maxDate.getMonth()+1)).slice(-2) + "/" +
              maxDate.getFullYear();
82        while(minDate <= maxDate){
83         var d = minDate.getFullYear() + "-" +
               ("0" + (minDate.getMonth()+1)).slice(-2) + "-" +
               ("0" + minDate.getDate()).slice(-2);
84         newT.push(d)
85         minDate = new Date(minDate
              .setDate(minDate.getDate() + 1));
86        }
87        for (i=0; i<newT.length; i++){
88         if (newT[i] in maximum){
89          orderedMax[newT[i]] = maximum[newT[i]];
90         } else {
91          orderedMax[newT[i]] = null;
92         }
93         if (newT[i] in minimum){
94          orderedMin[newT[i]] = minimum[newT[i]];
95         } else {
96          orderedMin[newT[i]] = null;
97         }
98         if (newT[i] in glycemia){
99          orderedGly[newT[i]] = glycemia[newT[i]];
100        } else {
101         orderedGly[newT[i]] = null;
```

```
102         }
103         if (newT[i] in heartRate){
104          orderedHR[newT[i]] = heartRate[newT[i]];
105         } else {
106          orderedHR[newT[i]] = null;
107         }
108         if (newT[i] in weight){
109          orderedWei[newT[i]] = weight[newT[i]];
110         } else {
111         orderedWei[newT[i]] = null;
112         }
113        }
114       }).then(function(){
115        var ctx = lineChartRef.getContext("2d");
116        var lineChart = new Chart(ctx, {
117         type: "line",
118         data: {
119          labels: newT,
120          datasets: [{
121           label: "Pressione Sistolica [mmHg]",
122           data: Object.values(orderedMax),
123           fill: false,
124           backgroundColor: ["rgba(8, 64, 215, 1)"],
125           borderColor: ["rgba(8, 64, 215, 1)"],
126           borderWidth: 3
127          },{
128           label: "Pressione Diastolica [mmHg]",
129           data: Object.values(orderedMin),
130           fill: false,
131           backgroundColor: ["rgba(32, 179, 248,1)"],
132           borderColor: ["rgba(32, 179, 248,1)"],
133           borderWidth: 3
134          },{
135           label: "Glicemia [mg/dl]",
136           data: Object.values(orderedGly),
137           fill: false,
138           backgroundColor: ["rgba(255, 221, 5, 1)"],
139           borderColor: ["rgba(255, 221, 5, 1)"],
140           borderWidth: 3
141          },{
142           label: "Frequenza Cardiaca [bpm]",
143           data: Object.values(orderedHR),
144           fill: false,
145           backgroundColor: ["rgba(248, 32, 32, 1)"],
146           borderColor: ["rgba(248, 32, 32, 1)"],
147           borderWidth: 3
148          },{
149           label: "Peso [kg]",
150           data: Object.values(orderedWei),
151           fill: false,
```

```
152           backgroundColor: ["rgba(10, 152, 24, 1)"],
153           borderColor: ["rgba(10, 152, 24, 1)"],
154           borderWidth: 3
155         }]
156       },
157       options: {
158         spanGaps: true,
159         scales: {
160           yAxes: [{
161             ticks: {
162               beginAtZero:true
163             },
164             scaleLabel: {
165               display:true,
166               labelString: "Valore"
167             }
168           }],
169           xAxes: [{
170             scaleLabel: {
171               display:true,
172               labelString: "Data"
173             }
174           }]
175         },
176         title: {
177           display: true,
178           position: "top",
179           text: "Valori Inseriti dal " + dateTitle1 +
                  " al " + dateTitle2,
180           fontSize: 18,
181         }
182       }
183     });
184   });
185  });
186 });
187 });
188 }
```

Listing A.31: PopulateLineCanvas Function

A.6 ListEcg and ListOldExamination Pages

A.6.1 HTML

The HTML layout of these two pages is really simple. They are entirely populated by functions that attach elements to the **list-title <p>** element of each page (Listing A.32 and A.33).

```
1 <p style="font-size:140%" id="list-title">ELENCO FILES ECG</p>
```

Listing A.32: ListEcg Page *list-title* Element

```
1 <p style="font-size:140%" id="list-title">ELENCO VISITE PASSATE
    </p>
```

Listing A.33: ListOldExamination Page *list-title* Element

A.6.2 Js

The following **populateList** function (Listing A.34) retrieves from the database the **url** of each **ECG** file for the considered patient and makes them available for downloading. A similar function is defined for the **Old Examinations** files. For each file, a **<form>** element is created, it is populated with the file name and attached to the page below the **list-title <p>** element (see Listing A.32 and A.33).

```
1  function populateList(){
2   var query = firebase.database().ref("patients/" + patient +
        "/EcgURL").orderByValue();
3   query.on("value", function(snapshot){
4    snapshot.forEach(function(data) {
5     var storageRef = firebase.storage().ref();
6     var downURL;
7     var spltDate = data.val().split("-");
8     var date = spltDate[2] + "/" + spltDate[1] +
          "/" + spltDate[0];
9     var fileName = data.key.replace("*",".");
10    var fileDirectory = data.val();
11    var fileURL = patient + "/ecg/" + fileDirectory +
          "/" + fileName;
12    storageRef.child(fileURL).getDownloadURL()
         .then(function(url) {
13     var f = document.createElement("form");
14     var fs = document.createElement("fieldset");
15     var a = document.createElement("a");
```

```
16      var b = document.createElement("button");
17      f.style.display = "block";
18      f.style.border = "groove";
19      f.style.overflow = "hidden";
20      fs.style.display = "block";
21      fs.style.margin = "1%";
22      a.style.display = "inline-block";
23      a.style.textAlign = "left";
24      a.style.width = "auto";
25      a.href = url;
26      a.target = "_blank";
27      a.textContent = date + " - " + fileName;
28      b.style.float = "right";
29      b.type = "button";
30      b.textContent = "Elimina";
31      b.id = fileURL;
32      b.addEventListener('click', deleteFileButton, false);
33      document.getElementById("list-title").after(f);
34      f.appendChild(fs);
35      fs.appendChild(a);
36      fs.appendChild(b);
37     });
38    });
39   });
40  }
```

Listing A.34: PopulateList Function

Appendix B

Android Application Code

Each **Android Activity** is basically defined by two files, a **XML** file that defines the layout and a **Java** file that defines the functionalities. Activities are then managed by the Operating System thanks to the **Manifest.XML** file. It defines the **package** name, the application **permissions** and on which versions of Android OS the app can run. In this file, **launch icon** and all activities are also called, to define their titles and which one is the *main* (the one opened when the app starts). It also defines **services** and **receivers** used to manage notifications (Listing B.1).

```
1  <manifest
2    xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.tesi.marco.filo">
4
5    <uses-permission android:name="android.permission
      .MANAGE_DOCUMENTS" />
6    <uses-permission android:name="android.permission
      .READ_EXTERNAL_STORAGE" />
7    <uses-permission android:name="android.permission
      .WRITE_EXTERNAL_STORAGE" />
8    <uses-permission
9      android:name="android.permission.VIBRATE"
10     android:maxSdkVersion="18" />
11
12   <application
13     android:allowBackup="true"
14     android:icon="@mipmap/heart_login"
15     android:label="@string/app_name"
16     android:roundIcon="@mipmap/heart_login"
17     android:supportsRtl="true"
18     android:theme="@style/AppTheme">
19
20     <activity android:name=".StartingActivity">
21       <intent-filter>
22         <action android:name="android.intent.action.MAIN" />
23         <category android:name="android.intent.category.LAUNCHER" />
```

```
24          </intent-filter>
25      </activity>
26      <activity
27          android:name=".LoginActivity"/>
28      <activity
29          android:name=".MainActivity"
30          android:screenOrientation="portrait"/>
31      <activity
32          android:name=".PatientDataActivity"
33          android:label="@string/history_activity_title"
34          android:screenOrientation="portrait"/>
35      <activity
36          android:name=".AddDataActivity"
37          android:screenOrientation="portrait"/>
38      <activity
39          android:name=".ForgotPasswordActivity"
40          android:label="@string/forgot_password_activity_title"
41          android:parentActivityName=".LoginActivity" />
42      <activity
43          android:name=".AddEventActivity"
44          android:screenOrientation="portrait"/>
45      <activity
46          android:name=".TherapyActivity"
47          android:label="@string/therapy_activity_title"
48          android:screenOrientation="portrait"
49          android:theme="@style/AppTheme.NoActionBar" />
50      <activity
51          android:name="com.theartofdev.edmodo.cropper.CropImageActivity"
52          android:theme="@style/Base.Theme.AppCompat" />
53
54      <receiver
55          android:name=".NotificationReceiver"
56          android:enabled="true"
57          android:exported="false"/>
58      <receiver
59          android:name=".ConfirmingNotificationReceiver"
60          android:enabled="true"
61          android:exported="false"/>
62
63      <service
64          android:name=".NotificationIntentService"
65          android:exported="false"/>
66      <service
67          android:name=".ConfirmingNotificationIntentService"
68          android:exported="false"/>
69    </application>
70  </manifest>
```

Listing B.1: Manifest File

B.1 LoginActivity

B.1.1 XML

Login Activity is principally composed by two **EditText** fields, where the user has to insert his/her email and password. Below them, there are the clickable **TextView**, that opens the **ForgotPassword** activity, and the **Button** to perform login. All these objects are positioned inside a **ScrollView**, to allow the user to scroll up and down and visualize all elements also with the **soft keyboard** opened. Code in Listing B.2 defines the activity layout.

```
1  <LinearLayout xmlns:android="http://schemas.android.com/
     apk/res/android"
2    xmlns:tools="http://schemas.android.com/tools"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    android:gravity="center_horizontal"
6    android:background="@color/grey_100"
7    android:padding="@dimen/activity_margin"
8    android:orientation="vertical"
9    tools:context=".LoginActivity">
10
11   <ProgressBar
12     android:id="@+id/login_progress_bar"
13     android:layout_width="wrap_content"
14     android:layout_height="wrap_content"
15     android:layout_gravity="center"
16     android:visibility="gone" />
17
18   <ScrollView
19     android:layout_width="match_parent"
20     android:layout_height="match_parent">
21
22     <LinearLayout
23       android:layout_width="match_parent"
24       android:layout_height="wrap_content"
25       android:orientation="vertical">
26
27       <ImageView
28         android:id="@+id/login_logo"
29         android:layout_width="160dp"
30         android:layout_height="160dp"
31         android:layout_margin="@dimen/margin"
32         android:layout_gravity="center_horizontal"
33         android:src="@mipmap/heart_login"/>
34
35       <TextView
36         android:layout_width="wrap_content"
37         android:layout_height="wrap_content"
38         android:layout_gravity="center_horizontal"
```

```
39          android:text="@string/developed_by"
40          android:textAlignment="center"/>
41
42        <android.support.design.widget.TextInputLayout
43          android:layout_width="match_parent"
44          android:layout_height="wrap_content"
45          android:paddingTop="@dimen/padding">
46
47          <AutoCompleteTextView
48            android:id="@+id/field_email"
49            android:layout_width="match_parent"
50            android:layout_height="wrap_content"
51            android:hint="@string/prompt_email"
52            android:textSize="@dimen/text"
53            android:maxLines="1"
54            android:inputType="textEmailAddress"
55            android:imeOptions="actionNext"/>
56        </android.support.design.widget.TextInputLayout>
57
58        <android.support.design.widget.TextInputLayout
59          android:layout_width="match_parent"
60          android:layout_height="wrap_content"
61          android:paddingTop="@dimen/padding">
62
63          <EditText
64            android:id="@+id/field_password"
65            android:layout_width="match_parent"
66            android:layout_height="wrap_content"
67            android:hint="@string/prompt_password"
68            android:textSize="@dimen/text"
69            android:maxLines="1"
70            android:inputType="textPassword"/>
71
72        </android.support.design.widget.TextInputLayout>
73
74        <RelativeLayout
75          android:layout_width="match_parent"
76          android:layout_height="wrap_content"
77          android:paddingTop="@dimen/padding">
78
79          <TextView
80            android:id="@+id/forgot_password"
81            android:layout_width="wrap_content"
82            android:layout_height="wrap_content"
83            android:layout_marginLeft="@dimen/half_margin"
84            android:layout_marginStart="@dimen/half_margin"
85            android:textColor="@color/colorAccent"
86            android:text="@string/forgot_password"
87            android:textSize="@dimen/reduced_text"
88            android:clickable="true"/>
```

```
89
90          <Button
91            android:id="@+id/email_sign_in_button"
92            android:layout_width="wrap_content"
93            android:layout_height="wrap_content"
94            android:layout_below="@id/forgot_password"
95            android:layout_alignParentRight="true"
96            android:layout_alignParentEnd="true"
97            android:text="@string/action_sign_in"/>
98
99        </RelativeLayout>
100     </LinearLayout>
101   </ScrollView>
102 </LinearLayout>
```

Listing B.2: LoginActivity Layout

B.1.2 Java

When the login button is pressed, the **listener** in Listing B.3 is activated.

```
1   findViewById(R.id.email_sign_in_button)
     .setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View v) {
4       email = mEmailField.getText().toString();
5       password = mPasswordField.getText().toString();
6       if (checkDataForLogin()){
7         focusView.requestFocus();
8       } else {
9         mProgressView.setVisibility(View.VISIBLE);
10        InputMethodManager imm =
           (InputMethodManager)getSystemService
           (LoginActivity.INPUT_METHOD_SERVICE);
11        imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
12        signIn();
13    }
14  }
15 });
```

Listing B.3: Login Button Listener

The **CheckDataForLogin** function (Listing B.4), called by the listener in Listing B.3, is used to perform preliminary checks before trying to log in, thanks to the devoted **Firebase** function.

```
1  public boolean checkDataForLogin() {
2
3    boolean check = false;
4    if (email.isEmpty() || password.isEmpty()){
5      if (password.isEmpty()) {
6        mPasswordField.setError(getString(R.string
   .error_field_required));
7        focusView = mPasswordField;
8      }
9      if (email.isEmpty()) {
10       mEmailField.setError(getString(R.string.error_field_required));
11       focusView = mEmailField;
12     }
13     check = true;
14   } else if (!isEmailValid(email)){
15     mEmailField.setError(getString(R.string.error_invalid_email));
16     focusView = mEmailField;
17     check = true;
18   } else if (!isPasswordValid(password)) {
19     mPasswordField.setError(getString(R.string
   .error_invalid_password));
20     focusView = mPasswordField;
21     check = true;
22   }
23   return check;
24 }
25
26 public static final Pattern VALID_EMAIL_ADDRESS_REGEX =
27   Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$",
    Pattern.CASE_INSENSITIVE);
28
29 private boolean isEmailValid(String email) {
30   Matcher matcher = VALID_EMAIL_ADDRESS_REGEX .matcher(email);
31   return matcher.find();
32 }
33
34 private boolean isPasswordValid(String password) {
35 // This is the constraint required by firebase.
36   return password.length() > 5;
37 }
```

Listing B.4: CheckDataForLogin Function

If data had been inserted correctly, the **signIn** function is called, defined by Code in Listing B.5.

```
1  private void signIn() {
2    mAuth.signInWithEmailAndPassword(email, password)
3    .addOnCompleteListener(this,
     new OnCompleteListener<AuthResult>() {
4      @Override
5      public void onComplete(@NonNull Task<AuthResult> task) {
6        if (task.isSuccessful()) {
7          Intent intent = new Intent(LoginActivity.this,
           MainActivity.class);
8          startActivity(intent);
9          finish();
10       } else {
11         try {
12           throw task.getException();
13         } catch(FirebaseAuthInvalidUserException e) {
14           Toast.makeText(LoginActivity.this,
              getString(R.string.invalid_user_exception),
15           Toast.LENGTH_LONG).show();
16         } catch(FirebaseAuthInvalidCredentialsException e) {
17           Toast.makeText(LoginActivity.this,
              getString(R.string.invalid_credential_exception),
18           Toast.LENGTH_LONG).show();
19         } catch(Exception e) {
20           Toast.makeText(LoginActivity.this,
              e.getLocalizedMessage(),
21           Toast.LENGTH_LONG).show();
22         }
23         mProgressView.setVisibility(View.GONE);
24       }
25     }
26   });
27 }
```

Listing B.5: SignIn Function

B.2 ForgotPasswordActivity

B.2.1 XML

Code in Listing B.6 defines the **ForgotPassword** activity layout.

```
1  [...]
2  <RelativeLayout
3    android:layout_width="match_parent"
4    android:layout_height="match_parent">
5
6    <android.support.design.widget.TextInputLayout
7      android:id="@+id/email"
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content"
10     android:paddingTop="@dimen/padding"
11     android:layout_margin="@dimen/margin">
12
13     <AutoCompleteTextView
14       android:id="@+id/field_email"
15       android:layout_width="match_parent"
16       android:layout_height="wrap_content"
17       android:hint="@string/prompt_email"
18       android:textSize="15sp"
19       android:maxLines="1"
20       android:inputType="textEmailAddress"
21       android:imeOptions="actionNext"/>
22   </android.support.design.widget.TextInputLayout>
23
24   <Button
25     android:id="@+id/recover_password_button"
26     android:layout_width="wrap_content"
27     android:layout_height="wrap_content"
28     android:layout_below="@id/email"
29     android:layout_alignParentRight="true"
30     android:layout_alignParentEnd="true"
31     android:layout_margin="@dimen/double_margin"
32     android:text="@string/action_recover_password"/>
33
34   <ProgressBar
35     android:id="@+id/login_progress_bar"
36     style="?android:progressBarStyleLarge"
37     android:layout_width="wrap_content"
38     android:layout_height="wrap_content"
39     android:layout_centerInParent="true"
40     android:layout_gravity="center"
41     android:visibility="gone"/>
42
43 </RelativeLayout>
```

Listing B.6: ForgotPasswordActivity Layout

B.2.2 Java

On this page, a controller equal to the one of the **LoginActivity** (see Listing B.4), acts on the **Email** field. If the check is passed, the keyboard hides, and a **progress bar** is shown until the **reset password email** is sent. Then the activity is *finished* and the login activity is *resumed*. Code in Listing B.7 defines the **sendEmail** function, that manages the **sendPasswordResetEmail** Firebase method.

```
1  mEmailField = (AutoCompleteTextView)
    findViewById(R.id.field_email);
2  sendEmail = (Button) findViewById(R.id.recover_password_button);
3  mProgressView = findViewById(R.id.login_progress_bar);
4
5  sendEmail.setOnClickListener(new View.OnClickListener() {
6    @Override
7    public void onClick(View v) {
8      email = mEmailField.getText().toString();
9      if (checkEmail()) {
10       focusView.requestFocus();
11     } else {
12       mProgressView.setVisibility(View.VISIBLE);
13       InputMethodManager imm =
          (InputMethodManager)getSystemService
          (ForgotPasswordActivity.INPUT_METHOD_SERVICE);
14       imm.hideSoftInputFromWindow(mEmailField
          .getWindowToken(),0);
15       mAuth.sendPasswordResetEmail(email)
16       .addOnCompleteListener(new OnCompleteListener<Void>() {
17         @Override
18         public void onComplete(@NonNull Task<Void> task) {
19           if (task.isSuccessful()){
20             Toast.makeText(ForgotPasswordActivity.this,
                R.string.email_sent, Toast.LENGTH_SHORT).show();
21             finish();
22           } else {
23             Toast.makeText(ForgotPasswordActivity.this,
                R.string.error_password_reset,
                 Toast.LENGTH_SHORT).show();
24           }
25           mProgressView.setVisibility(View.GONE);
26         }
27       });
28     }
29   }
30 });
```

Listing B.7: ForgotPassword Activity

B.3 MainActivity

B.3.1 XML

The **Main** activity is built on a 2x2 **TableLayout** (Listing B.8). Inside each **TableRow**, a **RelativeLayout** contains the **ImageButton** and the **TextView** with the button name.

```
1  <TableLayout xmlns:android="http://schemas.
    android.com/apk/res/android"
2    xmlns:tools="http://schemas.android.com/tools"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    tools:context="com.tesi.marco.filo.MainActivity"
6    android:orientation="vertical"
7    android:background="@color/grey_100">
8
9    <TableRow
10     android:layout_width="fill_parent"
11     android:layout_height="0dp"
12     android:layout_weight="1"
13     android:layout_marginBottom="@dimen/double_margin">
14
15     <RelativeLayout
16       android:layout_width="0dp"
17       android:layout_height="fill_parent"
18       android:layout_weight="1"
19       android:background="@color/grey_100"
20       android:padding="@dimen/margin"
21       android:clipToPadding="false" >
22
23       <ImageButton
24         android:id="@+id/add_data_button"
25         android:scaleType="fitCenter"
26         android:layout_width="match_parent"
27         android:layout_height="match_parent"
28         android:src="@mipmap/heart"
29         android:background="@color/white"
30         android:elevation="@dimen/elevation_main"/>
31
32       <TextView
33         android:id="@+id/add_data_button_text"
34         android:layout_width="match_parent"
35         android:layout_height="wrap_content"
36         android:layout_alignBottom="@id/add_data_button"
37         android:layout_marginBottom="@dimen/margin"
38         android:text="@string/add_data_button_text"
39         android:textAlignment="center"
40         android:textStyle="bold"
```

```
41          android:textSize="@dimen/text"
42          android:background="@color/white"
43          android:elevation="@dimen/elevation_main"/>
44
45      </RelativeLayout>
46
47
48      <RelativeLayout
49        android:layout_width="0dp"
50        android:layout_height="fill_parent"
51        android:layout_weight="1"
52        android:background="@color/grey_100"
53        android:padding="@dimen/margin"
54        android:clipToPadding="false">
55
56
57        <ImageButton
58          android:id="@+id/medicine_button"
59          android:layout_width="match_parent"
60          android:layout_height="match_parent"
61          android:background="@color/white"
62          android:scaleType="fitCenter"
63          android:src="@mipmap/ic_medicine"
64          android:elevation="@dimen/elevation_main"/>
65
66        <TextView
67          android:id="@+id/medicine_button_text"
68          android:layout_width="match_parent"
69          android:layout_height="wrap_content"
70          android:layout_marginBottom="@dimen/margin"
71          android:layout_alignBottom="@id/medicine_button"
72          android:text="@string/medicine_button_text"
73          android:textAlignment="center"
74          android:textStyle="bold"
75          android:textSize="@dimen/text"
76          android:background="@color/white"
77          android:elevation="@dimen/elevation_main"/>
78
79      </RelativeLayout>
80
81    </TableRow>
82
83    <TableRow
84      android:layout_width="fill_parent"
85      android:layout_height="0dp"
86      android:layout_weight="1"
87      android:layout_marginBottom="@dimen/double_margin" >
88
89      <RelativeLayout
90        android:layout_width="0dp"
```

```
91          android:layout_height="fill_parent"
92          android:layout_weight="1"
93          android:background="@color/grey_100"
94          android:padding="@dimen/margin"
95          android:clipToPadding="false">
96
97          <ImageButton
98            android:id="@+id/patient_information_button"
99            android:layout_width="match_parent"
100           android:layout_height="match_parent"
101           android:background="@color/white"
102           android:scaleType="fitCenter"
103           android:src="@mipmap/ic_clinic_information"
104           android:elevation="@dimen/elevation_main"/>
105
106         <TextView
107           android:id="@+id/patient_information_button_text"
108           android:layout_width="match_parent"
109           android:layout_height="wrap_content"
110           android:layout_alignBottom=
               "@id/patient_information_button"
111           android:layout_marginBottom="@dimen/margin"
112           android:background="@color/white"
113           android:text="@string/patient_information_button_text"
114           android:textAlignment="center"
115           android:textSize="@dimen/text"
116           android:textStyle="bold"
117           android:elevation="@dimen/elevation_main"/>
118
119       </RelativeLayout>
120
121       <RelativeLayout
122         android:layout_width="0dp"
123         android:layout_height="fill_parent"
124         android:layout_weight="1"
125         android:background="@color/grey_100"
126         android:padding="@dimen/margin"
127         android:clipToPadding="false">
128
129         <ImageButton
130           android:id="@+id/logout_button"
131           android:layout_width="match_parent"
132           android:layout_height="match_parent"
133           android:background="@color/white"
134           android:scaleType="fitCenter"
135           android:src="@mipmap/logout"
136           android:elevation="@dimen/elevation_main"/>
```

```
137
138         <TextView
139           android:id="@+id/logout_button_text"
140           android:layout_width="match_parent"
141           android:layout_height="wrap_content"
142           android:layout_alignBottom="@id/logout_button"
143           android:layout_marginBottom="@dimen/margin"
144           android:background="@color/white"
145           android:text="@string/logout_button_text"
146           android:textAlignment="center"
147           android:textSize="@dimen/text"
148           android:textStyle="bold"
149           android:elevation="@dimen/elevation_main" />
150
151       </RelativeLayout>
152     </TableRow>
153   </TableLayout>
```

Listing B.8: MainActivity Layout

B.3.2 Java

Code in Listing B.9 adds functionalities to the four **ImageButtons**. **AddData** and **info** open the related activities. **Medicine** opens the **Therapy Activity**, but passes the current day as an **Extra** in order to open the activity in the correct tab. **SignOutButton** calls the **signOut** function (Listing B.10), that performs **Firebase Sign Out** and finishes the **Main** activity.

```
1  addData = (ImageButton) findViewById(R.id.add_data_button);
2  addData.setOnClickListener(new View.OnClickListener() {
3    @Override
4    public void onClick(View v) {
5      startActivity(new Intent(MainActivity.this,
           AddDataActivity.class));
6    }
7  });
8  medicine = (ImageButton) findViewById(R.id.medicine_button);
9  medicine.setOnClickListener(new View.OnClickListener() {
10   @Override
11   public void onClick(View v) {
12     Calendar cal = Calendar.getInstance();
13     day = cal.get(Calendar.DAY_OF_WEEK);
14     switch (day){
15       case Calendar.SUNDAY:
16         tab = 6;
17         break;
18       case Calendar.MONDAY:
19         tab = 0;
20         break;
```

```
21          case Calendar.TUESDAY:
22            tab = 1;
23            break;
24          case Calendar.WEDNESDAY:
25            tab = 2;
26            break;
27          case Calendar.THURSDAY:
28            tab = 3;
29            break;
30          case Calendar.FRIDAY:
31            tab = 4;
32            break;
33          case Calendar.SATURDAY:
34            tab = 5;
35            break;
36        }
37        Intent intent = new Intent(MainActivity.this,
              TherapyActivity.class);
38        intent.putExtra("tab",tab);
39        startActivity(intent);
40      }
41  });
42  info = (ImageButton) findViewById(R.id
      .patient_information_button);
43  info.setOnClickListener(new View.OnClickListener() {
44    @Override
45    public void onClick(View v) {
46      startActivity(new Intent(MainActivity.this,
              PatientDataActivity.class));
47    }
48  });
49  signOutButton = (ImageButton) findViewById(R.id.logout_button);
50  signOutButton.setOnClickListener(new View.OnClickListener() {
51    @Override
52    public void onClick(View v) {
53      signOut();
54    }
55  });
```

Listing B.9: MainActivity Buttons

```
1  private void signOut() {
2    mAuth = FirebaseAuth.getInstance();
3    mAuth.signOut();
4    Intent intent = new Intent(MainActivity.this,
       LoginActivity.class);
5    startActivity(intent);
6    finish();
7  }
```

Listing B.10: MainActivity signOut Function

B.4 AddDataActivity

B.4.1 XML

AddDataActivity is composed by a list of buttons, positioned one below the other inside a **RelativeLayout**. The overall **RelativeLayout** is contained by a **ScrollView** so it can be easily visualized also on smaller screen and it is easily adaptable in case of future addition of more buttons. Code in Listing B.11 defines the activity layout.

```
1  <ScrollView xmlns:android="http://schemas
    .android.com/apk/res/android"
2    xmlns:tools="http://schemas.android.com/tools"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    tools:context="com.tesi.marco.filo.AddDataActivity">
6
7    <RelativeLayout
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content">
10
11     <Button
12       android:id="@+id/add_pressure_data"
13       android:layout_width="match_parent"
14       android:layout_height="wrap_content"
15       android:layout_marginTop="@dimen/margin"
16       android:layout_marginLeft="@dimen/margin"
17       android:layout_marginStart="@dimen/margin"
18       android:layout_marginRight="@dimen/margin"
19       android:layout_marginEnd="@dimen/margin"
20       android:layout_marginBottom="@dimen/half_margin"
21       android:text="@string/add_pressure_data" />
22
23     <Button
24       android:id="@+id/add_weight_data"
25       android:layout_below="@id/add_pressure_data"
26       android:layout_width="match_parent"
27       android:layout_height="wrap_content"
28       android:layout_marginTop="@dimen/half_margin"
29       android:layout_marginLeft="@dimen/margin"
30       android:layout_marginStart="@dimen/margin"
31       android:layout_marginRight="@dimen/margin"
32       android:layout_marginEnd="@dimen/margin"
33       android:layout_marginBottom="@dimen/half_margin"
34       android:text="@string/add_weight_data" />
35
36     <Button
37       android:id="@+id/add_smoke_data"
38       android:layout_below="@id/add_weight_data"
```

```
39          android:layout_width="match_parent"
40          android:layout_height="wrap_content"
41          android:layout_marginTop="@dimen/half_margin"
42          android:layout_marginLeft="@dimen/margin"
43          android:layout_marginStart="@dimen/margin"
44          android:layout_marginRight="@dimen/margin"
45          android:layout_marginEnd="@dimen/margin"
46          android:layout_marginBottom="@dimen/half_margin"
47          android:text="@string/add_smoke_data" />
48
49      <Button
50          android:id="@+id/add_physical_activity_data"
51          android:layout_below="@id/add_smoke_data"
52          android:layout_width="match_parent"
53          android:layout_height="wrap_content"
54          android:layout_marginTop="@dimen/half_margin"
55          android:layout_marginLeft="@dimen/margin"
56          android:layout_marginStart="@dimen/margin"
57          android:layout_marginRight="@dimen/margin"
58          android:layout_marginEnd="@dimen/margin"
59          android:layout_marginBottom="@dimen/half_margin"
60          android:text="@string/add_physical_activity_data"/>
61
62      <Button
63          android:id="@+id/add_heart_rate_data"
64          android:layout_below="@id/add_physical_activity_data"
65          android:layout_width="match_parent"
66          android:layout_height="wrap_content"
67          android:layout_marginTop="@dimen/half_margin"
68          android:layout_marginLeft="@dimen/margin"
69          android:layout_marginStart="@dimen/margin"
70          android:layout_marginRight="@dimen/margin"
71          android:layout_marginEnd="@dimen/margin"
72          android:layout_marginBottom="@dimen/half_margin"
73          android:text="@string/add_heart_rate_data"/>
74
75      <Button
76          android:id="@+id/add_glycemia_data"
77          android:layout_below="@id/add_heart_rate_data"
78          android:layout_width="match_parent"
79          android:layout_height="wrap_content"
80          android:layout_marginTop="@dimen/half_margin"
81          android:layout_marginLeft="@dimen/margin"
82          android:layout_marginStart="@dimen/margin"
83          android:layout_marginRight="@dimen/margin"
84          android:layout_marginEnd="@dimen/margin"
85          android:layout_marginBottom="@dimen/half_margin"
86          android:text="@string/add_glycemia_data"/>
87
```

```
88      <Button
89        android:id="@+id/add_event"
90        android:layout_below="@id/add_glycemia_data"
91        android:layout_width="match_parent"
92        android:layout_height="wrap_content"
93        android:layout_marginTop="@dimen/half_margin"
94        android:layout_marginLeft="@dimen/margin"
95        android:layout_marginStart="@dimen/margin"
96        android:layout_marginRight="@dimen/margin"
97        android:layout_marginEnd="@dimen/margin"
98        android:layout_marginBottom="@dimen/half_margin"
99        android:text="@string/add_event"/>
100
101   </RelativeLayout>
102 </ScrollView>
```

Listing B.11: AddDataActivity Layout

B.4.2 Java

Code in Listing B.12 describes the behaviour of the **PressureButton**. It is taken as an example to describe all buttons of this page opening dialogue boxes to add some kind of data. All buttons work like this, except the last one which opens a new activity. **AddPressureData** function is defined by code in Listing B.13.

The first three lines are used to retrieve information about the current date, that is used to save data and to allow the user to choose a different date in the **datePicker** function (Listing B.14).

```
1  SimpleDateFormat dateFormat =
     new SimpleDateFormat("yyyy-MM-dd", Locale.ITALIAN);
2  myDate = dateFormat.format(Calendar.getInstance().getTime());
3  todayDate = dateFormat.format(Calendar.getInstance().getTime());
4
5  PressureButton = (Button) findViewById(R.id.add_pressure_data);
6
7  PressureButton.setOnClickListener(new View.OnClickListener() {
8    @Override
9    public void onClick(View v) {
10     AddPressureData();
11   }
12 });
```

Listing B.12: PressureButton

Before the dialogue box is dismissed, data are checked, both for their presence and for their values (Listing B.13, lines 55 to 68 and lines 70 to 83 respectively). Moreover, if data are worrying, the **Alert** field is set as **true**.

```
1  private void AddPressureData() {
2    AlertDialog.Builder builder = new AlertDialog.Builder(this);
3    LayoutInflater inflater = getLayoutInflater();
4    View pressureDialogueView = inflater
       .inflate(R.layout.pressure_dialogue,null);
5    builder.setTitle(R.string.insert_pressure);
6    builder.setView(pressureDialogueView);
7    ePressureMax = (EditText) pressureDialogueView
       .findViewById(R.id.edit_max_pressure);
8    ePressureMax.setRawInputType(Configuration.KEYBOARD_12KEY);
9    ePressureMin = (EditText) pressureDialogueView
       .findViewById(R.id.edit_min_pressure);
10   ePressureMin.setRawInputType(Configuration.KEYBOARD_12KEY);
11   tAlertOldDate = (TextView) pressureDialogueView
      .findViewById(R.id.alert_old_date);
12   if (Integer.parseInt(myDate
      .replace("-","")<Integer.parseInt(todayDate
       .replace("-",""))){
13     tAlertOldDate.setVisibility(View.VISIBLE);
14   }
15
16   builder.setCancelable(false);
17   builder.setPositiveButton(R.string.ok,null);
18   builder.setNegativeButton(
19     R.string.cancel,
20     new DialogInterface.OnClickListener() {
21       public void onClick(DialogInterface dialog, int id) {
22         dialog.cancel();
23     }
24   });
25
26   final AlertDialog pressureDialogue = builder.create();
27   pressureDialogue.show();
28   Button ok = pressureDialogue
     .getButton(DialogInterface.BUTTON_POSITIVE);
29   ok.setOnClickListener(new View.OnClickListener() {
30     @Override
31     public void onClick(View v) {
32       String pressureMax = ePressureMax.getText().toString();
33       String pressureMin = ePressureMin.getText().toString();
34       if (checkPressureData(pressureMax,pressureMin)){
35         if (checkPressureValue(pressureMax,pressureMin)){
36           if (Double.parseDouble(pressureMax)>=180.0 ||
              Double.parseDouble(pressureMax)<=90.0 ||
               Double.parseDouble(pressureMin)>=100.0 ||
                Double.parseDouble(pressureMin)<=50.0){
37             Map<String,Object> update = new HashMap<>();
```

```
38              update.put("Alert", true);
39              alertRef.updateChildren(update);
40            }
41            DatabaseReference pRef = myRef.child("Pressure")
                .child(myDate);
42            pRef.child("Maximum").setValue(pressureMax);
43            pRef.child("Minimum").setValue(pressureMin);
44            pressureDialogue.cancel();
45          } else {
46            Toast.makeText(AddDataActivity.this,
               getString(R.string.error_out_of_scale),
               Toast.LENGTH_SHORT).show();
47          }
48        } else {
49          Toast.makeText(AddDataActivity.this,
             getString(R.string.error_dialog_ok),
              Toast.LENGTH_SHORT).show();
50        }
51      }
52    });
53  }
54
55  private boolean checkPressureData(String max, String min){
56    boolean check = true;
57    if (max.isEmpty()){
58      ePressureMax.setError(getString
         (R.string.error_field_required));
59      ePressureMax.requestFocus();
60      check = false;
61    }
62    if (min.isEmpty()){
63      ePressureMin.setError(getString
         (R.string.error_field_required));
64      ePressureMin.requestFocus();
65      check = false;
66    }
67    return check;
68  }
69
70  private boolean checkPressureValue(String max, String min){
71    boolean check = true;
72    if (Double.parseDouble(max)<=60.0 ||
        Double.parseDouble(max)>=270.0){
73      ePressureMax.setError(getString
         (R.string.error_out_of_scale));
74      ePressureMax.requestFocus();
75      check = false;
76    }
77    if (Double.parseDouble(min)<=20.0 ||
        Double.parseDouble(min)>=150.0){
```

```
78      ePressureMin.setError(getString
          (R.string.error_out_of_scale));
79      ePressureMin.requestFocus();
80      check = false;
81    }
82    return check;
83  }
```

Listing B.13: AddPressureData Functions

The **DatePicker** is opened from the menu, and allows to choose a past date to register data (Listing B.14).

```
1  private void datePicker(){
2    Calendar calendar = Calendar.getInstance();
3    final  SimpleDateFormat dateFormatter =
       new SimpleDateFormat("yyyy-MM-dd", Locale.ITALIAN);
4    final  SimpleDateFormat dateTitleFormatter =
      new SimpleDateFormat("dd/MM/yyyy", Locale.ITALIAN);
5    DatePickerDialog datePickerDialog =
       new DatePickerDialog(this, new DatePickerDialog
      .OnDateSetListener() {
6      public void onDateSet(DatePicker view, int year,
          int monthOfYear, int dayOfMonth) {
7        Calendar nDate = Calendar.getInstance();
8        nDate.set(year, monthOfYear, dayOfMonth);
9        myDate = dateFormatter.format(nDate.getTime());
10       if (Integer.parseInt(myDate
          .replace("-",""))>Integer.parseInt(todayDate
           .replace("-","")){
11         myDate = todayDate;
12         Toast.makeText(AddDataActivity.this,
              R.string.error_future_date,
              Toast.LENGTH_SHORT).show();
13       }
14       String title;
15       if (myDate.equals(todayDate)) {
16         title = getString(R.string.add_data_title_today);
17       } else {
18         title = getString(R.string.app_name) + " - " +
              dateTitleFormatter.format(newDate.getTime());
19       }
20       getSupportActionBar().setTitle(title);
21     }
22   }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
       calendar.get(Calendar.DAY_OF_MONTH));
23   datePickerDialog.show();
24 }
```

Listing B.14: DatePicker

B.5 AddEventActivity

AddEvent activity is structured in the same way of **AddData** one. The only difference is that a dialogue box allows to get an image from camera or from memory. Once the image **uri** has been retrieved, the image is set as background of the **ImageButton** used to activate this function, so the user can see it. Then it is encoded to a **String** with the **Base64 encoding** and saved into the database as a string. (Listing B.15).

```
1 imageBitmap = BitmapFactory
   .decodeStream(getContentResolver().openInputStream(resultUri));
2 BitmapDrawable bDrawable =
     new BitmapDrawable(getApplicationContext()
   .getResources(), imageBitmap);
3 bAddPhoto.setBackgroundDrawable(bDrawable);
4 ByteArrayOutputStream baos = new ByteArrayOutputStream();
5 imageBitmap.compress(Bitmap.CompressFormat.JPEG, 25, baos);
6 byte[] byteArrayImage = baos.toByteArray();
7 encodedImage = Base64.encodeToString(byteArrayImage,
     Base64.DEFAULT);
```

Listing B.15: Image Encoding

B.6 PatientDataActivity

B.6.1 XML

This activity is a list of fields that are populated with data taken from the database. At the activity opening, a progress bar is shown to notify the user that something is loading, then it disappears and the **NestedScrollView** behind it becomes visible. Each field has a name positioned on the left hand side of the screen, while the value is aligned on the right hand side. **History** data are an exception because they are written in a **TextView** below their title, due to the fact that they are made of long text. In Listing B.16 it is defined the first field, which contains the patient's name. The other fields are of the same type.

```
1  <LinearLayout xmlns:android="http://schemas
       .android.com/apk/res/android"
2    xmlns:tools="http://schemas.android.com/tools"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    android:paddingTop="@dimen/padding"
6    android:paddingBottom="@dimen/padding"
7    android:paddingStart="@dimen/reduced_padding"
8    android:paddingEnd="@dimen/reduced_padding"
9    android:orientation="vertical"
10   tools:context="com.tesi.marco.filo.PatientDataActivity">
11
12   <LinearLayout
13     android:id="@+id/data_progress_bar"
14     android:layout_width="match_parent"
15     android:layout_height="match_parent"
16     android:gravity="center"
17     android:visibility="gone" >
18
19     <ProgressBar
20       android:layout_width="wrap_content"
21       android:layout_height="wrap_content"
22       style="?android:progressBarStyleLarge">
23     </ProgressBar>
24   </LinearLayout>
25
26   <android.support.v4.widget.NestedScrollView
27     android:layout_width="match_parent"
28     android:layout_height="match_parent">
29
30     <RelativeLayout
31       android:layout_width="match_parent"
32       android:layout_height="wrap_content">
33
```

```
34          <TextView
35            android:id="@+id/title_personal_data"
36            android:layout_width="match_parent"
37            android:layout_height="wrap_content"
38            android:gravity="center_horizontal"
39            android:layout_marginTop="@dimen/margin"
40            android:text="@string/personal_data"
41            android:textSize="@dimen/text"
42            android:textColor="@color/colorPrimaryDark"/>
43
44          <RelativeLayout
45            android:id="@+id/name_surname"
46            android:layout_width="match_parent"
47            android:layout_height="wrap_content"
48            android:layout_below="@id/title_personal_data"
49            android:layout_marginTop="@dimen/margin"
50            android:orientation="horizontal">
51
52            <TextView
53              android:id="@+id/patient"
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:textSize="@dimen/text"
57              android:textColor="@color/black"
58              android:text="@string/patient"
59              android:layout_margin="@dimen/margin"/>
60
61            <TextView
62              android:id="@+id/patient_name"
63              android:layout_width="wrap_content"
64              android:layout_height="wrap_content"
65              android:layout_alignParentRight="true"
66              android:layout_alignParentEnd="true"
67              android:textSize="@dimen/text"
68              android:text="@string/info_not_present"
69              android:textColor="@color/colorAccentDark"
70              android:layout_margin="@dimen/margin" />
71
72          </RelativeLayout>
73          [...]
74        </RelativeLayout>
75      </android.support.v4.widget.NestedScrollView>
76  </LinearLayout>
```

Listing B.16: PatientData Activity Layout

B.6.2 Java

Code in Listing B.17 defines the function to populate the **Personal Data** fields of **PatientData** activity. Other functions populating remaining fields are of the same type.

```
1  tvName = (TextView) findViewById(R.id.patient_name);
2  tvCode = (TextView) findViewById(R.id.fiscal_code_data);
3  tvDob = (TextView) findViewById(R.id.date_of_birth_data);
4  tvPob = (TextView) findViewById(R.id.place_of_birth_data);
5
6  db = FirebaseDatabase.getInstance();
7  userRef = db.getReference().child("patients").child(Uid);
8  personalDataRef = userRef.child("PersonalData");
9
10 personalDataRef
    .addListenerForSingleValueEvent(new ValueEventListener() {
11   @Override
12   public void onDataChange(DataSnapshot dataSnapshot) {
13     code = dataSnapshot.child("FiscalCode")
        .getValue().toString();
14     tvCode.setText(code);
15     dateOfBirth = dataSnapshot.child("DateOfBirth")
        .getValue().toString();
16     tvDob.setText(dateOfBirth);
17     if (dataSnapshot.hasChild("Birthplace")) {
18       placeOfBirth = dataSnapshot.child("Birthplace")
          .getValue().toString();
19       tvPob.setText(placeOfBirth);
20     }
21     if (dataSnapshot.hasChild("ResidencyCity") &&
         !dataSnapshot.hasChild("ResidencyRoad")) {
22       residency = dataSnapshot.child("ResidencyCity")
          .getValue().toString();
23       tvRes.setText(residency);
24     }else if (dataSnapshot.hasChild("ResidencyCity") &&
         dataSnapshot.hasChild("ResidencyRoad")) {
25       residency = dataSnapshot.child("ResidencyCity")
          .getValue().toString() + "\n" +
26       dataSnapshot.child("ResidencyRoad").getValue().toString();
27       tvRes.setText(residency);
28     }
29     if (dataSnapshot.hasChild("Telephone")) {
30       telephone = dataSnapshot.child("Telephone")
          .getValue().toString();
31       tvTel.setText(telephone);
32     }
```

```
33      if (dataSnapshot.hasChild("Gender")) {
34        gender = dataSnapshot.child("Gender")
           .getValue().toString();
35        tvGender.setText(gender);
36      }
37      mProgressView.setVisibility(View.GONE);
38    }
39
40    @Override
41    public void onCancelled(DatabaseError databaseError) {
42      Toast.makeText(PatientDataActivity.this,
           databaseError.getMessage(), Toast.LENGTH_SHORT).show();
43    }
44  });
```

Listing B.17: Personal Data Population

B.7 TherapyActivity

B.7.1 XML

Therapy Activity is built on a **TabLayout**. There are seven tabs, one for each day of the week. Code in Listing B.18 defines the container for the layout defined by code in Listing B.19. This layout is the same for each tab, but it is populated in different ways depending on the selected one.

```
1  <LinearLayout xmlns:android="http://schemas.android.com/
     apk/res/android"
2    xmlns:app="http://schemas.android.com/apk/res-auto"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    android:orientation="vertical">
6
7    <android.support.design.widget.AppBarLayout
8      android:id="@+id/appbar"
9      android:layout_width="match_parent"
10     android:layout_height="wrap_content"
11     android:paddingTop="8dp"
12     android:theme="@style/AppTheme.AppBarOverlay">
13
14     <android.support.v7.widget.Toolbar
15       android:id="@+id/toolbar"
16       android:layout_width="match_parent"
17       android:layout_height="?attr/actionBarSize"
18       android:background="?attr/colorPrimary"
19       app:layout_scrollFlags="scroll|enterAlways"
20       app:popupTheme="@style/AppTheme.PopupOverlay" />
21   </android.support.design.widget.AppBarLayout>
```

```
22    <android.support.design.widget.TabLayout
23      android:id="@+id/sliding_tabs"
24      android:layout_width="match_parent"
25      android:layout_height="wrap_content"
26      app:tabMode="fixed" />
27
28    <android.support.v4.view.ViewPager
29      android:id="@+id/viewpager"
30      android:layout_width="match_parent"
31      android:layout_height="0px"
32      android:layout_weight="1"
33      android:background="@android:color/white"/>
34
35  </LinearLayout>
```

Listing B.18: Therapy Activity Layout

The layout of each tab is composed by a **RelativeLayout** containing the page list title (*Time* on the left hand side, *Drug* on the right hand side), and by a **NestedScrollView** containing the RelativeLayouts for each time (from 8.00 to 22.00). These RelativeLayouts are not visible unless they are populated, to avoid a view overloading.

```
1  <LinearLayout xmlns:android="http://schemas
     .android.com/apk/res/android"
2    android:orientation="vertical"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent">
5
6    <RelativeLayout
7      android:id="@+id/title_therapy"
8      android:layout_width="match_parent"
9      android:layout_height="wrap_content"
10     android:orientation="horizontal">
11
12     <TextView
13       android:id="@+id/title_time"
14       android:layout_width="wrap_content"
15       android:layout_height="wrap_content"
16       android:textSize="@dimen/text"
17       android:textColor="@color/black"
18       android:text="@string/time"
19       android:layout_margin="@dimen/margin"/>
20
21     <TextView
22       android:id="@+id/title_drugs"
23       android:layout_width="wrap_content"
24       android:layout_height="wrap_content"
25       android:layout_alignParentRight="true"
26       android:layout_alignParentEnd="true"
```

```
27          android:text="@string/drug"
28          android:textSize="@dimen/text"
29          android:textColor="@color/colorAccent"
30          android:gravity="end"
31          android:layout_margin="@dimen/margin" />
32
33      </RelativeLayout>
34
35      <android.support.v4.widget.NestedScrollView
36        android:layout_width="match_parent"
37        android:layout_height="match_parent">
38
39        <RelativeLayout
40          android:layout_width="match_parent"
41          android:layout_height="wrap_content">
42
43          <RelativeLayout
44            android:id="@+id/time8"
45            android:layout_width="match_parent"
46            android:layout_height="wrap_content"
47            android:layout_marginTop="@dimen/margin"
48            android:orientation="horizontal"
49            android:visibility="gone">
50
51            <TextView
52              android:id="@+id/time8_time"
53              android:layout_width="wrap_content"
54              android:layout_height="wrap_content"
55              android:textSize="@dimen/text"
56              android:textColor="@color/black"
57              android:text="@string/hour8"
58              android:layout_margin="@dimen/margin"/>
59
60            <TextView
61              android:id="@+id/time8_drugs"
62              android:layout_width="wrap_content"
63              android:layout_height="wrap_content"
64              android:layout_alignParentRight="true"
65              android:layout_alignParentEnd="true"
66              android:textSize="@dimen/text"
67              android:textColor="@color/colorAccent"
68              android:gravity="end"
69              android:layout_margin="@dimen/margin" />
70
71          </RelativeLayout>
72  [...]
73        </RelativeLayout>
74      </android.support.v4.widget.NestedScrollView>
75  </LinearLayout>
```

Listing B.19: Therapy Tab Layout

B.7.2 Java

Code in Listing B.20 defines the **viewPager** that is necessary to manage the **TabLayout**, and also to manage the opening of the correct tab based on the **Extra** got from the **Intent**. Code in Listing B.21 defines the **TherapyFragmentPageAdapter**, used to manage the structure of the seven tabs.

```
1 viewPager = (ViewPager) findViewById(R.id.viewpager);
2 viewPager.setAdapter(new TherapyFragmentPagerAdapter
    (getSupportFragmentManager(), TherapyActivity.this));
3
4 TabLayout tabLayout = (TabLayout)
    findViewById(R.id.sliding_tabs);
5 tabLayout.setupWithViewPager(viewPager);
6 TabLayout.Tab tab = tabLayout
   .getTabAt(getIntent().getIntExtra("tab",0));
7 tab.select();
```

Listing B.20: Therapy Activity ViewPager

```
1  public class TherapyFragmentPagerAdapter
    extends FragmentPagerAdapter {
2    final int PAGE_COUNT = 7;
3    private String tabTitles[] =
      new String[]{getString(R.string.mon),
      getString(R.string.tue), getString(R.string.wed),
      getString(R.string.thu), getString(R.string.fri),
      getString(R.string.sat), getString(R.string.sun),};
4    private Context context;
5
6    public TherapyFragmentPagerAdapter(FragmentManager fm,
        Context context) {
7      super(fm);
8      this.context = context;
9    }
10   @Override
11   public int getCount() {
12     return PAGE_COUNT;
13   }
14   @Override
15   public Fragment getItem(int position) {
16     return PageFragment.newInstance(position + 1);
17   }
18   @Override
19   public CharSequence getPageTitle(int position) {
20     return tabTitles[position];
21   }
22 }
```

Listing B.21: Therapy Tabs Definition

The following code (Listing B.22) is devoted to populate the tab view. **Text View** "**tv8**" and **Relative Layout** "**rl8**" have been taken as an example to show how each time layout is populated and made visible. Basically, the function reads the database and, for each therapy, checks the assumption days and hours. Then, if the day corresponds to the one of the tab, it keeps the data, otherwise not. In the end, it adds the therapy to all those views matching the assumption hours, also changing their **visibility** attribute to *visible*.

```
1  public static PageFragment newInstance(int page) {
2    Bundle args = new Bundle();
3    args.putInt(ARG_PAGE, page);
4    PageFragment fragment = new PageFragment();
5    fragment.setArguments(args);
6    return fragment;
7  }
8
9  @Override
10 public View onCreateView(LayoutInflater inflater,
     ViewGroup container,  Bundle savedInstanceState) {
11   View view = inflater
       .inflate(R.layout.therapy_day, container, false);
12   tv8 = (TextView) view.findViewById(R.id.time8_drugs);
13   rl8 = (RelativeLayout) view.findViewById(R.id.time8);
14
15 [...]
16
17   auth = FirebaseAuth.getInstance();
18   Uid = auth.getCurrentUser().getUid();
19   db = FirebaseDatabase.getInstance();
20   therapiesRef = db.getReference().child("patients").child(Uid)
      .child("Therapies");
21   therapiesRef.addListenerForSingleValueEvent(
       new ValueEventListener() {
22     @Override
23     public void onDataChange(DataSnapshot dataSnapshot) {
24       for (DataSnapshot childSnapshot : dataSnapshot
           .getChildren()){
25         String[] assumptionDays = childSnapshot
            .child("assumptionDays").getValue().toString()
            .split(",");
26         String[] assumptionHours = childSnapshot
            .child("assumptionHours").getValue().toString()
           .split(",");
27         if (Arrays.asList(assumptionDays)
           .contains(days[mPage-1])){
28           for (int i=0; i<assumptionHours.length; i++){
29             if (assumptionHours[i].equals("8.00")){
30               tv8.append(childSnapshot.child("drug")
                 .getValue().toString());
```

```
31                  String extra = getInfoMeal(childSnapshot);
32                  if (!extra.isEmpty()){
33                    tv8.append(", " + extra + "\n");
34                  } else {
35                    tv8.append("\n");
36                  }
37                  rl8.setVisibility(View.VISIBLE);
38                }
39
40  [...]
41
42              }
43            }
44          }
45        }
46
47        @Override
48        public void onCancelled(DatabaseError databaseError) {
49          Toast.makeText(TherapyActivity.this,
             databaseError.getMessage(), Toast.LENGTH_SHORT).show();
50        }
51      });
52      return view;
53  }
```

Listing B.22: Therapy Fragment Page

Bibliography

[1] iconfinder.com https://www.iconfinder.com

[2] Wikipedia.org, *Firebase*, https://en.wikipedia.org/wiki/Firebase

[3] Firebase.google.com, *Firebase Authentication*, https://firebase.google.com/docs/auth/

[4] Firebase.google.com, *Firebase Realtime Database*, https://firebase.google.com/docs/database/

[5] Firebase.google.com, *Read and Write Data on Android*, https://firebase.google.com/docs/database/android/read-and-write

[6] Wikipedia.org, *UTF-8*, https://it.wikipedia.org/wiki/UTF-8

[7] Firebase.google.com, *Realtime Database Limits*, https://firebase.google.com/docs/database/usage/limits

[8] https://firebase.google.com/pricing/

[9] *Introducing JSON*, https://www.json.org/index.html

[10] Firebase.google.com, *Cloud Storage*, https://firebase.google.com/docs/storage/

[11] Firebase.google.com *Get Started with Storage Security Rules*, https://firebase.google.com/docs/storage/security/start

[12] heartfoundation.org, *Smoking and your heart*, https://www.heartfoundation.org.au/your-heart/know-your-risks/smoking-and-your-heart

[13] American Heart Association, *How High Blood Pressure Can Lead to a Heart Attack*, http://www.heart.org/HEARTORG/Conditions/HighBloodPressure/LearnHowHBPHarmsYourHealth/How-High-Blood-Pressure-Can-Lead-to-a-Heart-Attack_UCM_301823_Article.jsp#.Wplz8ujOVPZ

[14] M. Miller, *Dyslipidemia and cardiovascular risk: the importance of early prevention*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2729130/

[15] National Institute of Diabetes and Digestive and Kidney Diseases, *Diabetes, Heart Disease, and Stroke*, https://www.niddk.nih.gov/health-information/diabetes/overview/preventing-problems/heart-disease-stroke

[16] Il Progetto Cuore, *Prevenzione e Stili di Vita*, http://www.cuore.iss.it/

prevenzione/prevenzione.asp
[17] Dr. Massimo Gualerzi, *Esiste una Familiarità nei casi di infarto*, https://www.pazienti.it/risposte/infarto-familiarita
[18] Timothy J. Moynihan, *Can chemotherapy side effects increase the risk of heart disease?*, https://www.mayoclinic.org/diseases-conditions/cancer/expert-answers/chemotherapy-side-effects/faq-20058319
[19] Canadian Cance Society, *Heart damage and chemotherapy*, http://www.cancer.ca/en/cancer-information/diagnosis-and-treatment/chemotherapy-and-other-drug-therapies/chemotherapy/side-effects-of-chemotherapy/heart-damage-and-chemotherapy/?region=on
[20] Cardio-Oncology and Radiation Heart Disease Specialty Centers & Clinics, *Radiation Heart Disease Clinic*, https://my.clevelandclinic.org/departments/heart/depts/radiation-heart-disease-clinic
[21] David J. Cutter, Sarah C. Darby and Syed W. Yusuf, *Risks of Heart Disease after Radiotherapy*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3113133/
[22] Birgitt Dau and Mark Holodniy, *The Relationship Between HIV Infection and Cardiovascular Disease*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2780822/
[23] National Institute of Diabetes and Digestive and Kidney Diseases, *Heart Disease & Kidney Disease*, https://www.niddk.nih.gov/health-information/kidney-disease/chronic-kidney-disease-ckd/heart-disease
[24] CardioSecur, *Allergies and Cardiovascular Disease*, https://www.cardiosecur.com/en/your-heart/specialty-articles/allergies-and-cardiovascular-disease/
[25] Wikipedia.org, *Body mass index*, https://en.wikipedia.org/wiki/Body_mass_index
[26] Bill Hendrick, *Obesity Increases Risk of Deadly Heart Attacks*, https://www.webmd.com/heart-disease/news/20110214/obesity-increases-risk-of-deadly-heart-attacks#1
[27] Wikipedia.org, *Infarto miocardico acuto*, https://it.wikipedia.org/wiki/Infarto_miocardico_acuto
[28] Wikipedia.org, *Ejection fraction*, https://en.wikipedia.org/wiki/Ejection_fraction
[29] Wikipedia.org, *End-diastolic volume*, https://en.wikipedia.org/wiki/End-diastolic_volume
[30] http://www.chartjs.org/
[31] Touch Icon made by http://www.freepik.com/ from www.flaticon.com
[32] ArthurHub, *Android-Image-Cropper*, https://github.com/ArthurHub/Android-Image-Cropper