POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering

**Master of Science in Biomedical Engineering**

# Design of a Web Application for Patient Monitoring After Coronary Angioplasty

*Supervisor:*
Prof. **Monica Visintin**

*Candidate:*
**Sara Prete**

Academic Year 2018-2019

# Acknowledgements

Prima di procedere con la stesura di questa tesi, e' doveroso ringraziare tutte le persone che mi hanno supportato e che sono state sempre al mio fianco.

Innanzitutto vorrei ringraziare la Professoressa Monica Visintin per avermi dato fiducia e per essere stata sempre disponibile, gentile e attenta nello svolgimento del mio lavoro di tesi.

Ringrazio Paolo Doz per avermi accolto nella famiglia Abinsula e avermi permesso di crescere in un ambiente che per me era del tutto nuovo. Insieme a lui ringrazio Alberto, Gabriele, Pietro e MariaChiara, i 'maestri' che in questi mesi mi hanno sopportato e hanno ascoltato pazientemente le mie richieste. Con loro spero di continuare a collaborare con il solito entusiasmo. Non posso non ringraziare tutti i ragazzi di Abinsula: Paola, Luca, Giulio, David, Ilario, Riccardo, Gianluca, Massimo, Gianfranco, Matteo, Renato e Claudio. Li ringrazio per i sorrisi che mi regalano ogni giorno e per essere sempre presenti quando ho bisogno d'aiuto.

Ringrazio Matteo Bianco. Senza la sua grinta e forza di volonta' CardioFilo non sarebbe la stessa. Ringrazio Fabrizio D'Ascenzo per la gentilezza dimostratami quando ho intrapreso questo percorso.

Il grazie piu' grande va a i miei genitori, Maria e Salvatore. Non ho parole per esprimere la mia gratitudine nei loro confronti. Immaginando la donna che vorrei diventare, la immagino come loro: forte, determinata, paziente, simpatica, giocherellona, affettuosa e un po' permalosa. Grazie per tutti i sacrifici di questi anni. Ringrazio mio fratello Giulio per l'affetto e la pazienza. Parliamo poco ma ci vogliamo tanto bene. E poi ovviamente...grazie Nerone!

Grazie a Claudia e Lucia, amiche da sempre. Mi stupisco ogni volta nel vedere quanta fiducia riescano a riporre nei miei confronti. Senza il loro supporto costante, non sarei la persona che sto diventando. Spero di non perderle mai.

Ringrazio Greca, sorella maggiore, amica e complice di tante risate. Grazie per l'attenzione affettuosa che mi regala. Anche se lontane, sono sicura che continueremo a crescere insieme.

Grazie Alessia e Silvia, amiche, colleghe, confidenti, compagne di figuracce e pianti (di Silvia). Solo loro e pochi altri possono capire quanto costi portare avanti questo percorso. Gli anni con voi sono stati i piu' belli. Spero che i prossimi lo siano altrettanto.

Ringrazio tutti gli zii, i cugini, gli amici, le mie coinquiline, i colleghi di univerista'. Non c'e' una sola persona tra loro che non abbia creduto in me.

# Summary

CardioFilo project aims to create an healthcare model for patients affected by atrial fibrillation, myocardial infarction and/or undergoing coronary angioplasty or other revascularisation procedures. Patients, in particular those who have been affected by infarction, in the phases following the acute event, are particularly fragile from a clinical point of view and they need medical assistance to prevent the risk of disease recurrence. Unfortunately, after discharge from the hospital, assistance is often limited to a few cardiological visits at a distance of 3-6 months or a year. Habits such as controlling blood pressure and weight or smoking cessation are actions that the patient can be easily get used to. However he/she is not always able to manage radical changes of habits, too frequent controls and complex therapies. The risk involved in patients adopting behaviors for effective secondary prevention of cardiovascular disease is that patients themselves could experience a new cardiovascular adverse event. The aim of the CardioFilo application is to provide an additional tool for the secondary prevention of cardiovascular diseases through the smartphone, the most used objects in almost everyone's life. The name CardioFilo stems indeed from the desire to create a "direct line" ("filo" in Italian) between the cardiologist and the patient. This thesis work was carried out at Abinsula s.r.l., a Sardinian company that operates in the information technology sector. The main requirements of the application have been suggested by cardiologists of Ospedale San Giovanni Molinette who requested the implementation of the application. In a previous thesis work [1], a first version of both web and mobile application was developed. The information obtained from the developer of these applications was decisive in the phase of gathering the requirements. On the other hand, from an implementative point of view, the application has been completely replaced since the previous version used the cloud-based storage system 'Firebase', a system that, even according to the ethics committee of the Molinette hospital, does not guarantee total security of patient data: a more secure and robust server-based data storage system is needed.

The web application was implemented jointly with the Abinsula web development team and all the graphics are still in development together with the graphic team. Both teams are located in Sassari and the collaboration with them took place remotely. The applica-

tion consists in a web interface for cardiologists and a mobile interface for patients. In this master thesis work, together with the definition of the requirements and graphics for both web and mobile applications, the web platform was implemented using the Python-based framework Django. For the modeling of clinical data, an open source clinical standard called openEHR was respected. The definition of the clinical data model and the creation of an openEHR repository are currently under development.

# List of acronyms

**AF** Atrial Fibrillation

**BMI** Body Mass Index

**ECG** Electrocardiogram

**HIV** Human Immunodeficiency Virus

**PCI** Percutaneous coronary intervention

**CABG** Coronary Artery Bypass Graft surgery

**UML** Unified Modeling Language

**FV** First Visit

**NV** New Visit

**NP** New Patient

**PP** Patient Page

**API** Application Programming Interface

**JSON** JavaScript Object Notation

**GDPR** General Data Protection Regulation

**DPO** Data Protection Officer

**DPIA** Data Protection Impact Assessment

**EHR** Electronic Health Record

**CKM** Clinical Knowledge Manager

**AE** Archetype Editor

**ADL** Archetype Definition Language

**TD** Template Designer

**SQL** Structured Query Language

**DBMS** DataBase Management System

**RDBMS** Relational DataBase Management System

**MD** Medical Device

**MDD** Medical Device Directive

**SaMD** Software as a Medical Device

# Contents

# Chapter 1

# The CardioFilo Project

CardioFilo is a service aimed at optimizing cardiac patient home follow-up by integrating a smartphone app and a web platform. The web application aims to remotely monitor cardiac patients and it's used by cardiologists. Patients will use the mobile application to enter personal and clinical data. However mobile application can be used also by the cardiologist to have always on the smartphone an overview of the health status of the patients. Both web and mobile application communicate with the server via Internet connecting by Wi-Fi/ADSL, Ethernet or 3G/4G (Fig. 1.4)

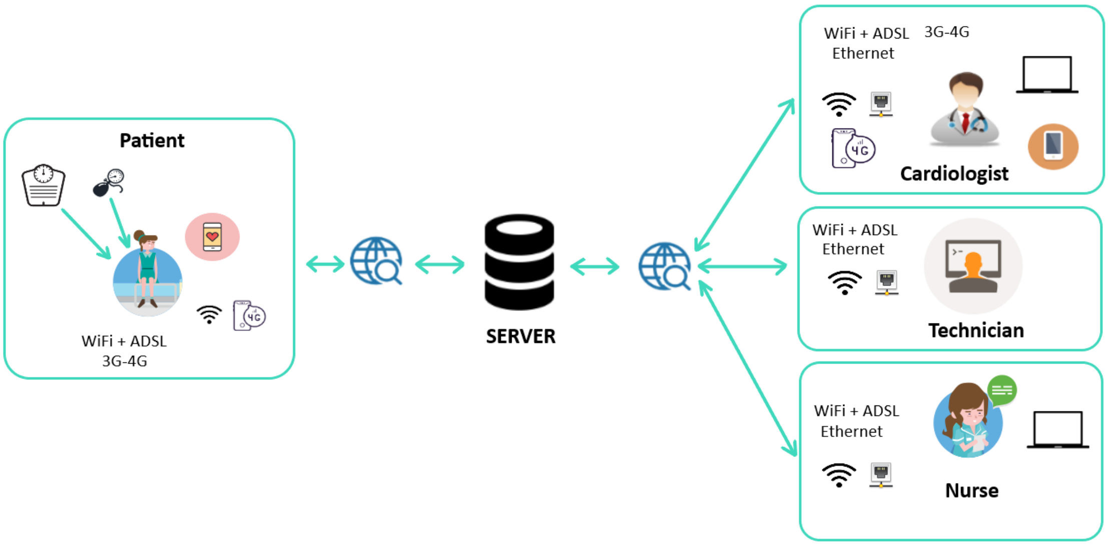


Figure 1.1: Schematic representation of the CardioFilo hardware architecture

The pathologies of interest for the CardioFilo project are ischemic cardiopathies and atrial fibrillation, both the most serious and most probable complications that arise after heart attack.

## 1.1 Pathologies of interest

### 1.1.1 Ischemic Heart Disease

Ischemic heart disease is one of the main causes of death in the western world, together with atherosclerosis. Ischemic heart disease includes pathological conditions for which the heart does not receive the proper amount of blood and oxygen. The heart works thanks to a balance between the need for oxygen, which is used for cardiac metabolism, and the right supply of blood. In some pathological conditions, the oxygen supply is not sufficient and the heart risks damages that reduce its functionality and lead to heart failure. Ischemic heart disease can occur with angina pectoris or acute myocardial infarction. The latter occurs following a sudden obstruction of the coronary arteries consequent to the formation of atherosclerotic plaques in the blood vessel walls or (more rarely) to coronary spasms. As a result of an infarction, the heart undergoes morpho-conformational variations that limit its normal functions and can trigger heart failure if no proper action is taken. The treatment for ischemic heart disease plans to restore the blood flow to the heart by drug therapy or a revascularisation intervention. The revascularisation intervention may be either a Percutaneous transluminal Coronary Angioplasty (PCI) or a Coronary Artery Bypass Graft surgery (CABG) (Fig. 1.2). PCI is a minimal invasive intervention that involves the insertion in the blocked coronary of a small balloon, connected to a stent, which is inflated and expanded at the obstruction of the artery. CABG is an intervention that occurs through sternotomy. The surgeon draws a healthy vein or artery from the patient's body and connects it to the blocked coronary artery. By doing so, the grafted artery or vein bypasses the clogged coronary allowing the blood to re-infuse the heart.

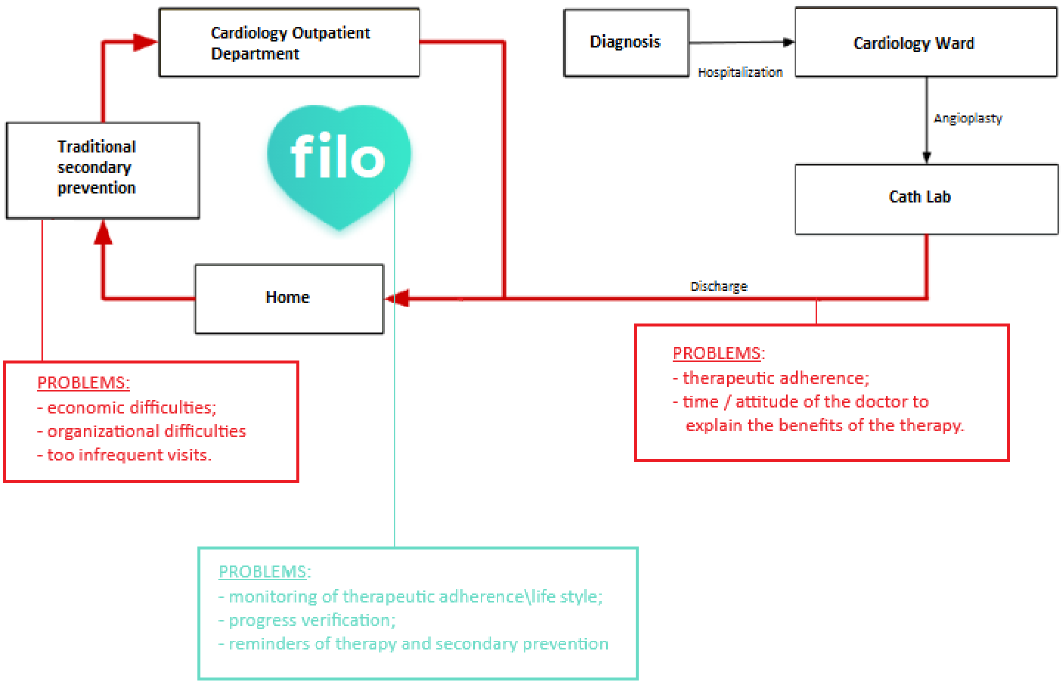


Figure 1.2: Schematic representation of the current post-angioplasty/heart attack follow-up modalities

### 1.1.2 Heart Failure

Heart failure is a condition in which the heart does not receive the right amount of blood in order to satisfy its energy needs. Moreover it is one of the main causes of mortality and morbidity in our country and it is also an important burden on the economy of the health system. Other causes are coronary heart diseases, hypertension, previous myocardial infarction, diabetes, arrhythmia and some congenital diseases. There are two types of heart failure: acute, when it occurs after a heart attack or a hypertensive crisis, and chronic, in response to damage (even asymptomatic) accumulated over time. Many patients do not know they are suffering from heart failure but in the last decades it's one of the most common disease in the western world. So it is a very alarming problem. According to [2]: "Heart failure is [...] an Italian problem and concerns about six hundred thousand people. Around 65 years, it is the first cause of hospitalization but after this threshold, the frequency doubles in every decade of age up to even tips of about ten percent after seventy years.[...] ". It is clear that this is therefore a problem destined to grow as the population is aging due to the improvement of therapies and sanitary conditions.

### 1.1.3 Atrial Fibrillation and Atrial Flutter

Atrial fibrillation consists in an irregular and often rapid heart rate that can increase risk of stroke, heart failure and other heart-related complications. During atrial fibrillation, the heart beats irregularly and suddenly. Symptoms usually are palpitations, shortness of breath and weakness. Episodes of atrial fibrillation can be isolated, but some patients develop permanent atrial fibrillation that requires treatment. Atrial fibrillation can lead to the formation of atherosclerotic plaques in the blood vessel wall that may circulate to other organs and lead to blocked blood flow (ischemia). Treatment for atrial fibrillation consists in anticoagulant therapy. Atrial fibrillation can be:

- Valvular. Valvular AF affects patients with mitral valve disease or with an implanted mechanical valve. Valvular AF therapy consists in the assumption of Warfarin drugs.
- Non Valvular. It has not very known causes. Some can be high blood pressure or thyroid problems. The therapy for this type of AF is DOAC Therapy (Direct-acting Oral Anticoagulants).

Atrial flutter is similar to atrial fibrillation, but the heart rate is more organized and less chaotic than atrial fibrillation. However atrial flutter can develop into atrial fibrillation and vice versa. As with atrial fibrillation, atrial flutter is usually not dangerous for life if properly treated.

## 1.2 Actors

The CardioFilo system foresees the following user actors (Fig. 1.3):

- Cardiologist: user of both web and mobile applications;
- Patient/Caregiver: user of mobile application;
- Nurse: user of web application with restrictions;
- Doctors from scientific societies: users of the web application in which all the data have been anonymized.
- Technicians: developers of both web and mobile application, maintainers of the software

The patient, or who takes care of him (caregiver), will use Cardiofilo mobile application with three main functions:

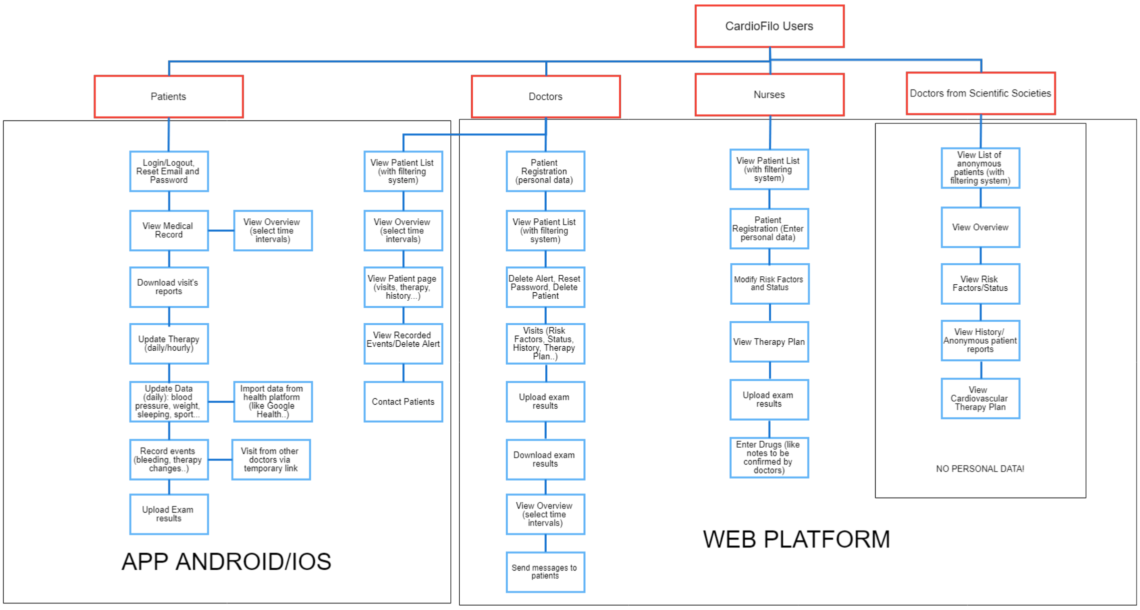


Figure 1.3: CardioFilo Users

- Collection of clinical measurements (e.g. : blood pressure, heart rate, weight, adverse events);
- Improvement of therapeutic compliance through an active monitoring system of the therapy intake;
- Direct communication with the health care provider of changes in the health status.

The cardiologist will use both the CardioFilo web platform and the smartphone app to:

- Always have updated information on the patient;
- Collect data in a simple and effective way;
- Optimize the therapeutic measures.

Advantages for the patient:

- Continuous remote monitoring of health conditions by the reference cardiologist;
- Feeling of security arising from knowing that there is a doctor who monitors patient's health data;
- Collection of clinical data aimed at optimizing care;
- Optimization of therapeutic compliance;

- Increased awareness of his health status (Patient Empowerment).

Disadvantages for the patient:

- Stress caused by all the reminders automatically generated by the mobile application;
- Feeling of being constantly observed;
- Anxiety caused by a parameter out of the ideal range.

Advantages for the doctor:

- Real-time access to patient data;
- Timely identification of the most critical patients;
- System of 'alarm filtering': the system will sent to the doctor the most critical notifications about events registered by the patient.
- Deeper knowledge of the patient's health status;
- Better management of resources and costs without sacrificing the quality of service offered to patients.

Disadvantages for the doctor:

- Legal issues about responsibilities towards the patients;
- Amount of constant notifications for alarms that he believes are not serious.

### 1.2.1 Cardiologists

The cardiologist actively interacts with the web application. He/she can also access the CardioFilo mobile application with a username and a password but, in this case, without the possibility of making changes. Every physician has his/her own password if he/she has to/prefers to work independently from other colleagues, otherwise it's possible to set a unique password for different accounts. When the cardiologist tries to log in with an invalid username or password an alert automatically appears. After the first login, the physician will be asked to change the password. At every attempt to log out an alert appears ('Are you sure you want to leave?'). Once logged in in the web application, the cardiologist is able to:

- Interact with the mobile application;
- Create a New Patient Page;

- Conduct Visits.

Through the mobile applications, the cardiologist is able to:

- View patients data (e.g. Overview, Status, Personal data etc);
- Get notifications about the events registered by the patients.
- Contact patients.

**First Visit**

Patient and cardiologist meet each other for the first time. The physician asks the patient for personal data in order to insert him/her in the Patient List. This task could also be done by a secretary or a nurse. Required patient personal data:

- Name and surname;
- Email to enable the patient to access the mobile application (ID and password automatically generated);
- Date and place of birth;
- Address;
- Phone number;
- Gender.

If the patient is accompanied by a caregiver, the caregiver must also provide personal data. The physician explains the patient how the mobiles app works and guides him/her through installation. For patients there will be some video tutorials that:

- guide them through the installation of the mobile app;
- illustrate procedures to patients (e.g. angioplasty).

Now the first visit can start and the following is required:

- Risk factors, i.e. smoke (how many cigarettes per day and pack per year), diabetes, hypertension, familiarity with coronary heart diseases, previous chemotherapy/radiotherapy, HIV in treatment, several kidney failures.
- Patient's status, i.e. height, weight, BMI, previous acute coronary events, ejection fraction, angioplasty (date, treatment, coronary district), allergies, implanted active devices.

- Patient History: recent and passed medical history, cardiovascular history, ECG files and other result exams files uploaded by physicians himself or by the patient using the CardioFilo mobile app.
- Report visit.
- Establish a therapy plan;
- Report conclusions and recommendations.

If the patient already exists in the database, an alert will be displayed. At the end of every visit, the system automatically generates a pdf file which contains the report of the visit.

**Patient List**

It is here assumed that the physician has already registered some patients and done a first visit. He/she's now able to access all patients' personal pages. In the patient page the physician can:

- Start a new visit.
- See an overview of the data entered by the patient (sport, smoke, therapy, systolic blood pressure, diastolic blood pressure, glycemia, heart rate, weight and recorded events).
- Access to previous visit's reports, a list of ECG files and result of the exams.
- Reset patient's password (maybe the patient forgot it) and cancel alerts (physician cancels patient's alert when they meet each other).

If the patient records a traumatic event (continuous bleeding, accident, hospitalization) or pressure/heart rate/glycemia values outside a physiological range, his/her name in the patient list is marked with an alert or highlighted in red. It's possible to send an email (specifying the severity of the event) to the physician who can immediately call the patient/caregiver or send a notification to the cardiologist mobile application. More cardiologists in the same hospital can treat the same patient (not at the same time). It is possible to obtain the personal data of the patient having only his fiscal code (e.g. a few data like name, surname, date and place of birth). Nurse or secretary can do this. The web platform is equipped with a data filtering system so that the physician can choose to see small groups of patients on the basis of severity, gender or pathology/diagnosis. It is also necessary to report if the patient has an active implantable device like pacemaker, implantable cardiac defibrillator or implantable loop recorder and when/where it has been implanted. The cardiologist may also decide to schedule visits to these devices with the patient.

### 1.2.2 Patients

Target patients: women around 75 years old and men around 65 years old. After receiving a username and a password, the patient logs in for the first time. He/she has to change immediately the automatically generated password. After the first visit, the patient starts to use the mobile application. Patients and physicians have to schedule device's periodic checks and report them. The patient is able to enter:

- Blood pressure;
- Weight;
- Smoked cigarettes;
- Sport activity;
- Heart rate;
- Glycemia;
- Sleep monitoring;
- Events: change of therapy (personal decision or recommended by the physician), other visits, hospitalization, bleeding without medical intervention (minutes), loading of exam results;
- Physician email and hospital phone number.

If the patient enters out of range values the system displays a message and the patient has to enter reasonable values. On the app, the patient can see:

- Personal data, risk factors, status, History, Therapy (can see previous reports and exam files);
- A calendar and the Therapy plan;
- Notifications ('Don't forget to take your pills!').

When the patient logs out, an alert message must appear ('Are you sure you want to leave?'). For patients there will be a video tutorial that:

- guides patients through installation of the Android/Ios app;
- illustrates procedures to patients (e.g. angioplasty video-tutorial).

#### Caregiver

Since the patients are elderly and middle-aged people, it is necessary to consider that they may have cognitive or visual impairments, that they may have difficulties in using smartphones or tablets. For this reasons, it is reasonable to provide a figure that supports the patient in daily actions such as using a smartphone. Following is a more precise definition of caregiver: [3] "Person who takes primary responsibility for someone who cannot care fully for himself or herself. The primary caregiver may be a family member, a trained professional or another individual. A person may need care due to loss of health, loss of memory, the onset of illness, an incident (or risk) of falling, anxiety or depression, grief, or a disabling condition.". The caregiver can use the patient's personal page with patient's ID and password on patient's smartphone or on his/her own if the patient doesn't have one.

### 1.2.3 Nurses

Also nurses can access the CardioFilo web platform. A nurse, however, has access only to certain sections of the web application because, as they are not doctors, nurses are not enabled to perform visits. However, the nurse can be a decisive figure in the patient's acceptance phase, i.e. before the patient meets the doctor. In fact, a nurse could speed up the registration process of the patient, for example by entering his personal data, and, being a specialized professional figure, performing a 'pre-visit' on the patient by entering simple clinical measures such as blood pressure, risk factors drugs habitually assumed and previous interventions or loading reports of previous visits and results of tests performed by the patient. Obviously the nurse will not access the history section or the section that allows doctors to modify the patient's cardiological treatment plan. Nurses will only be able to consult the data that they have personally entered but they will not be able to visualize or modify the data entered by the doctor.

### 1.2.4 Doctors from Scientific Societies

Clinical data from cardiological patients are of interest to scientific societies that conduct statistical and scientific studies on diseases such as ischemic heart disease or atrial fibrillation. It was decided to allow doctors belonging to scientific societies to use the CardioFilo web platform. However, the sensitive personal data of patients can not be provided to a doctor who is not the personal cardiologist of the patient and, for this reason, the data will be displayed in an anonymous and encrypted manner. It's possible to create an anonymous patient database to develop algorithms useful in the prognostic definition of the patient suffering from cardiological diseases.

### 1.2.5 Technician

The technician:

- develops both web and mobile applications.
- is the only user able to create login credentials for the physician (which will be changed right after the first login).
- is the only user able to modify both web and Android/Ios applications.
- has to manage the app interaction with the Health Apple/Google Apps.
- is responsible for bug fixing;
- is responsible of the GDPR requirements for privacy and data protection.

## 1.3 Collection of Requirements

What has been described so far is the result of a joint work with Molinette doctors. At first, indeed, numerous meetings were held in which all the requirements of the web and mobile application were defined, both from the point of view of contents and from the point of view of graphics. The phase of gathering the requirements was also necessary in order to precisely define the application architecture and then proceed with the implementation in Django. At first, all the information was summarized in Use Case Diagrams and Activity Diagrams using Visual Paradigm and Cacoo ([4], [5]). See Figure 1.4 for CardioFilo software architecture.

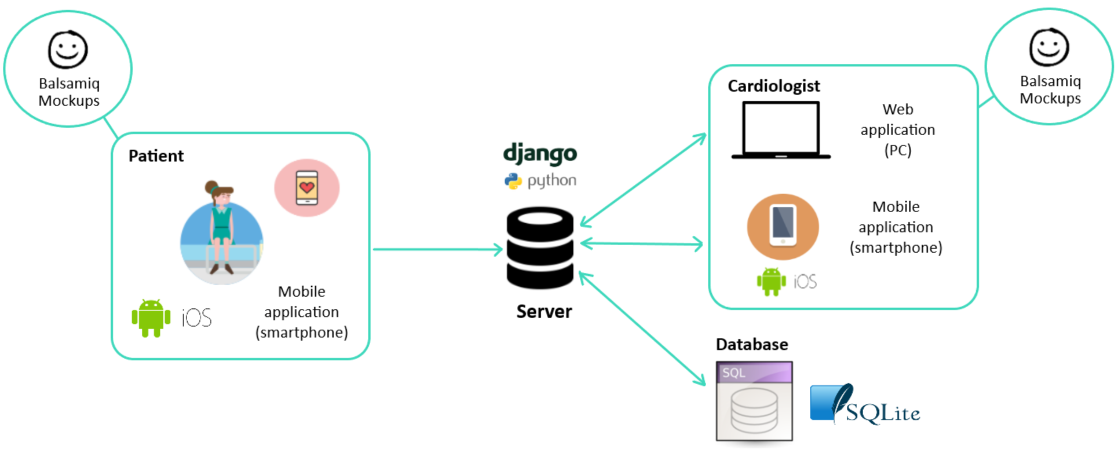


Figure 1.4: Schematic representation of the CardioFilo software architecture

### 1.3.1 UML: Use Case Diagrams

To carry out the collection of requirements in a comprehensive and unambiguous manner, in order to produce quality software, Use Case Diagrams have been realized (see Fig. 1.5 and 1.6).

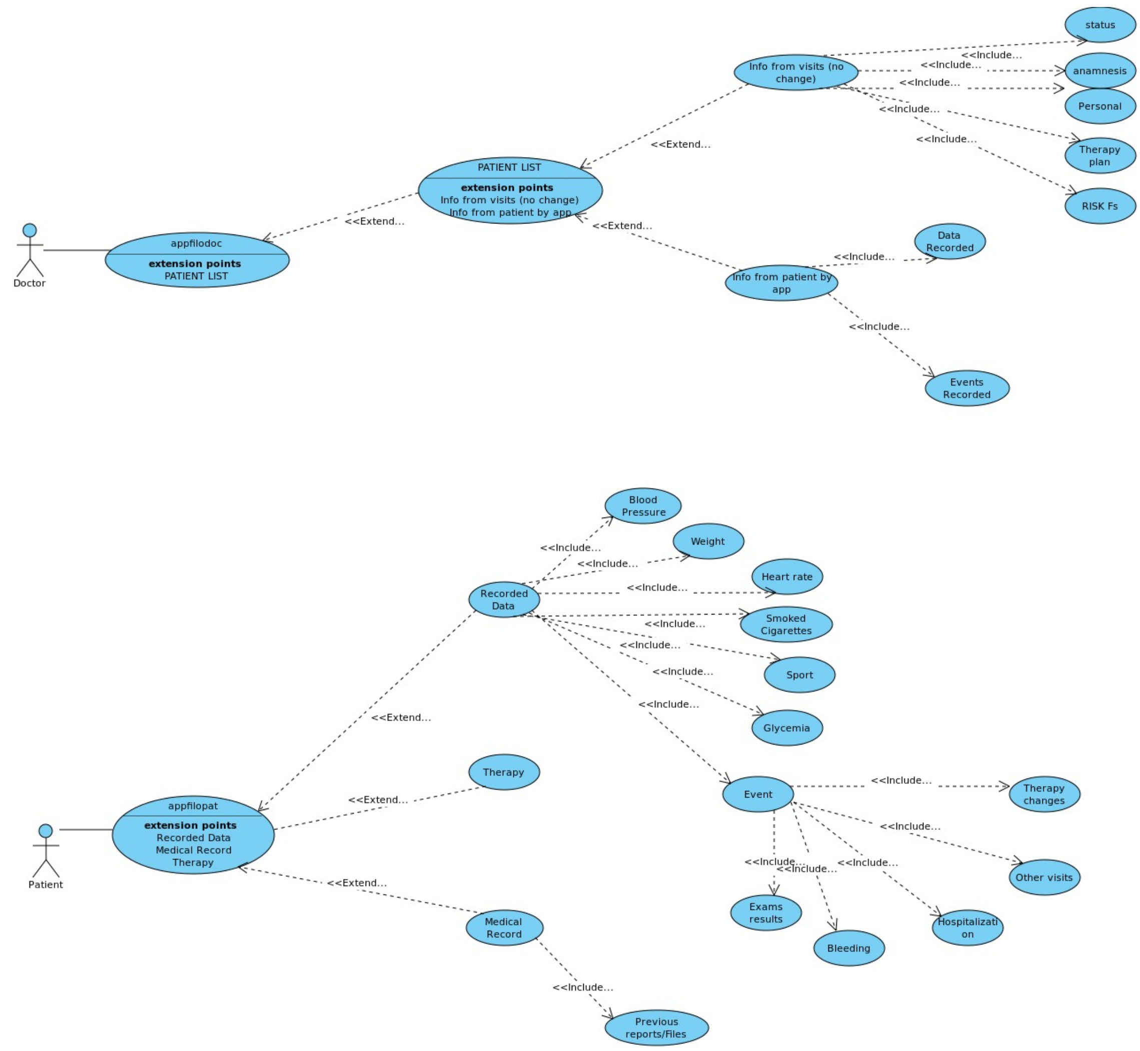


Figure 1.5: Schematic representation of the Use Case Diagram relative to the use of the mobile application by doctors and patients

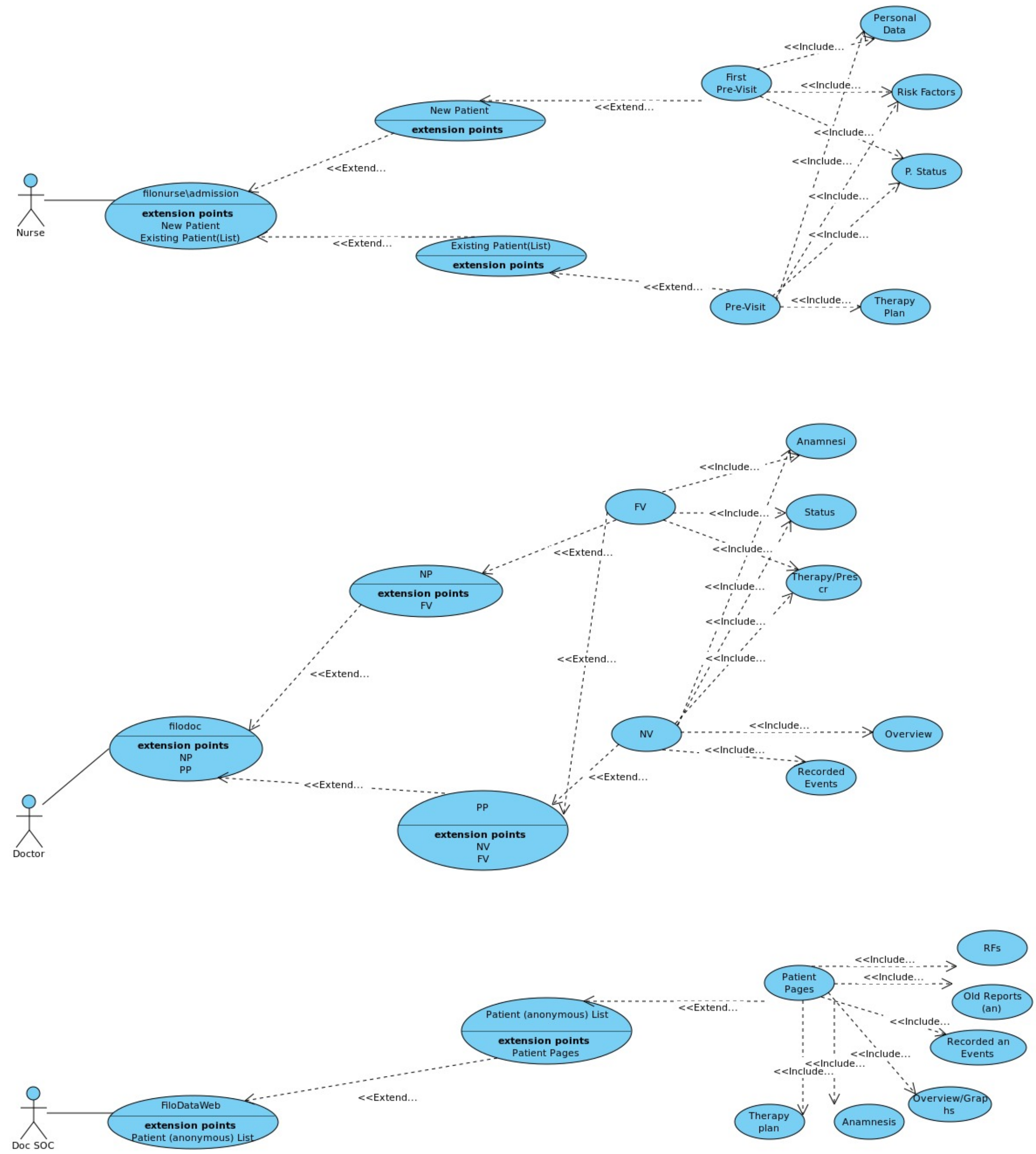


Figure 1.6: Schematic representation of the Use Case Diagram relative to the use of the web application by doctors, nurses and doctors from scientific societies

### 1.3.2 UML: Activity Diagrams

For a more precise design of the web application architecture, activity diagrams have been made. Each diagram represents a single screen of the CardioFilo web application and contains more detailed information about the logical flow of data.

### Visit

The core of CardioFilo application is the 'Visit' section whose activity diagram is shown in Fig. 1.7. It is during the visit that the doctor can enter the most significant clinical data regarding the patient, perform specific examinations and agree with the patient the optimal therapeutic plan.

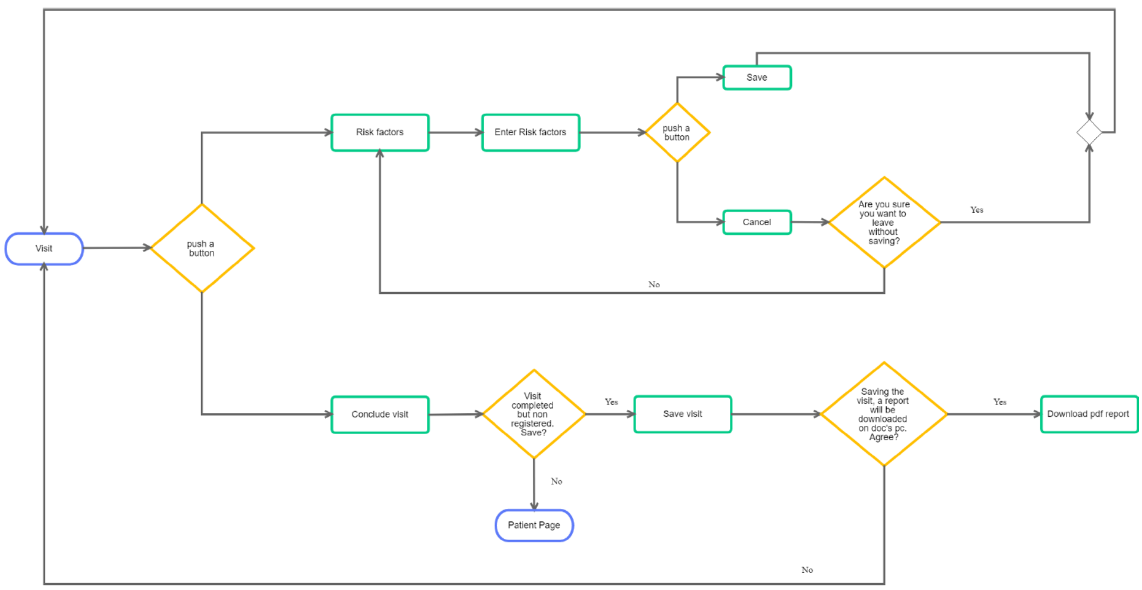


Figure 1.7: Activity Diagram relative to the visit flow

Depending on the pathology, the therapy can vary between anti-platelet and anti-coagulant therapy. In some cases, patients may have to follow both (see Fig. 1.8).

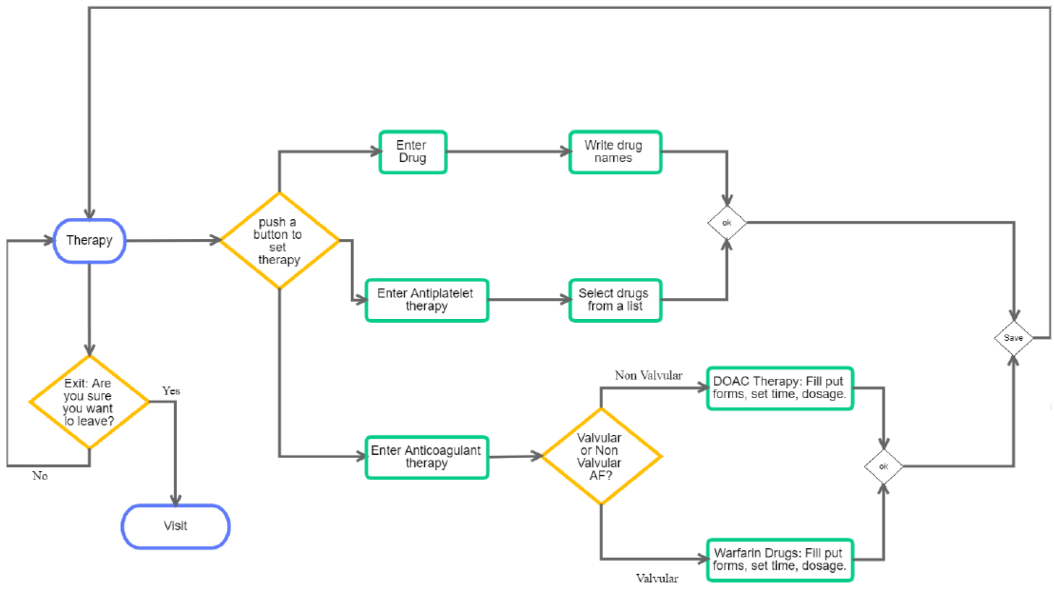


Figure 1.8: Activity Diagram relative to the therapy setting

### Overview

Patient's Overview is one of the most important tools within the CardioFilo Application. On web and mobile platforms, both patients and doctors can have a detailed report of patient's conditions via curves, charts and bar diagrams.

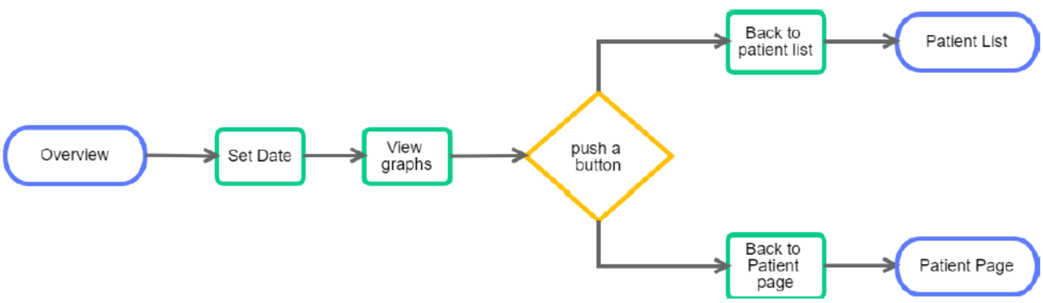


Figure 1.9: Activity Diagram relative to the patient overview

## 1.4 CardioFilo Graphics: creation of mockups and screens

Having defined the UML diagrams, the next step was the implementation of the application mockups. For this task Balsamiq [6] was used, a graphic tool for developers and designers that allows to "sketch" user interfaces and screens for websites and applications. The mockups realized in Balsamiq were supplied to Abinsula's Graphic team located in Sassari, which developed the entire graphics of the web platform and the mobile application. Some mockups and screens of the CardioFilo application are reported in Fig. 1.10-1.18.

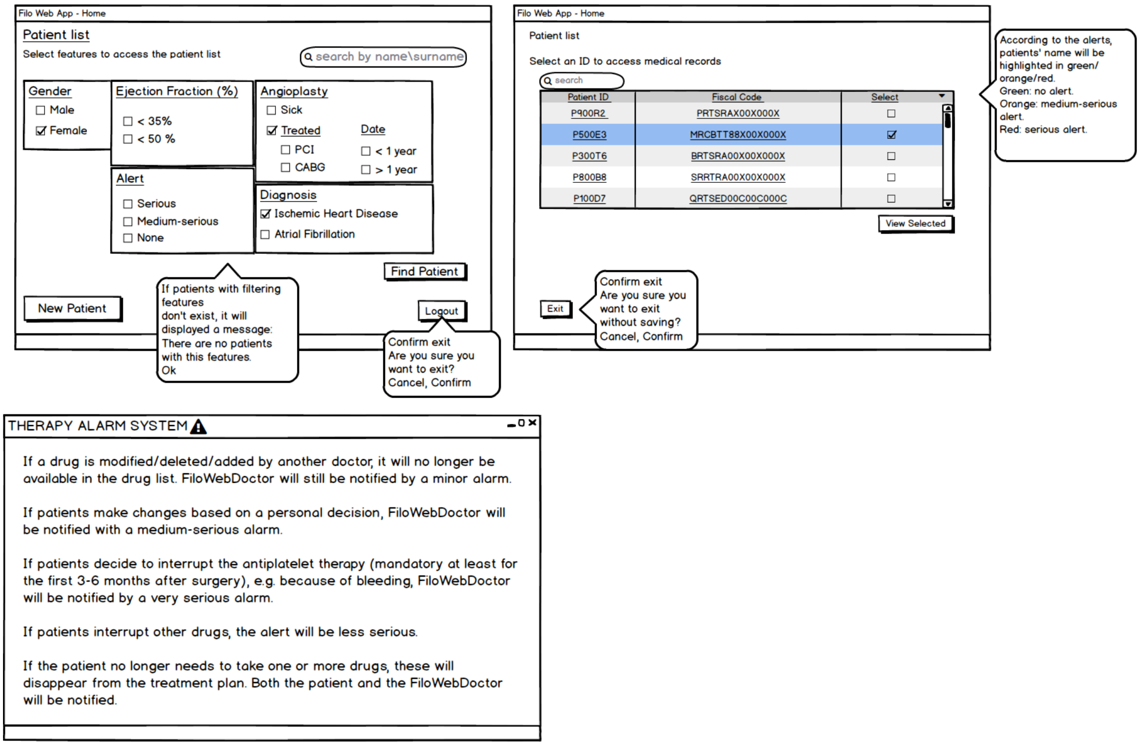


Figure 1.10: Home Page: select features (Mockup)

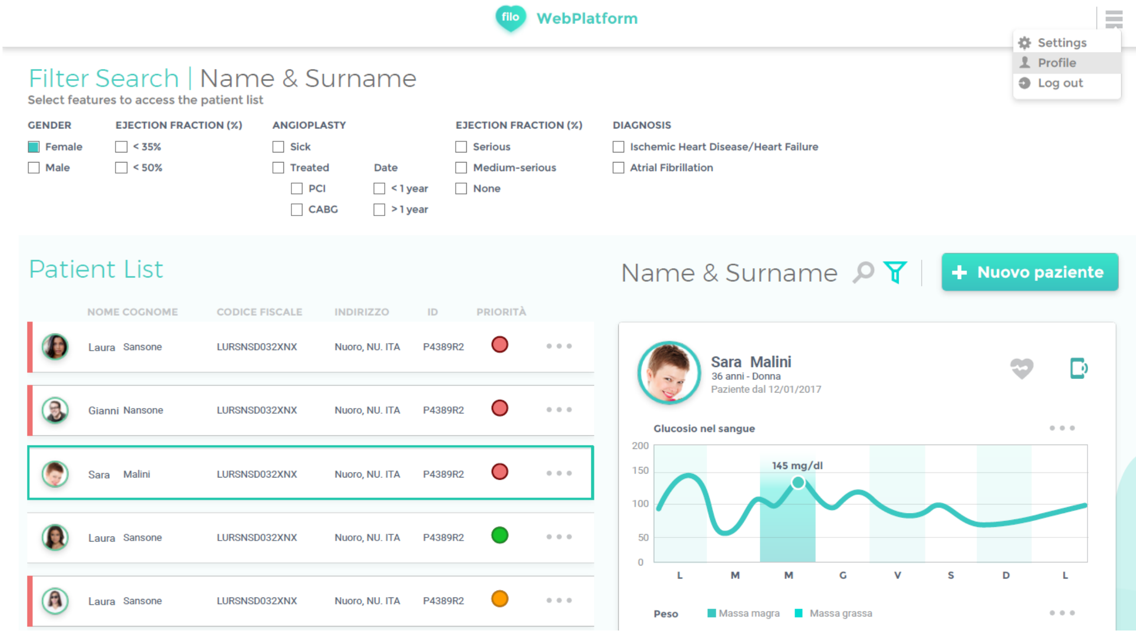


Figure 1.11: Home Page: select features (Screen)

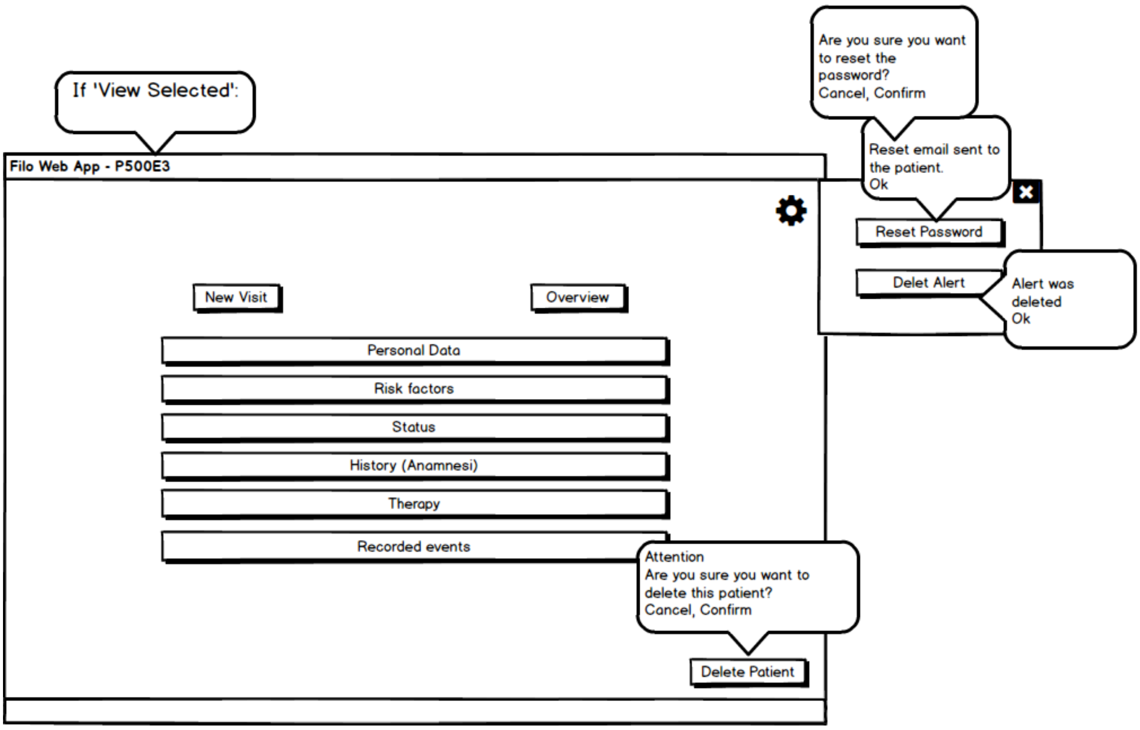


Figure 1.12: Patient Page (Mockup)

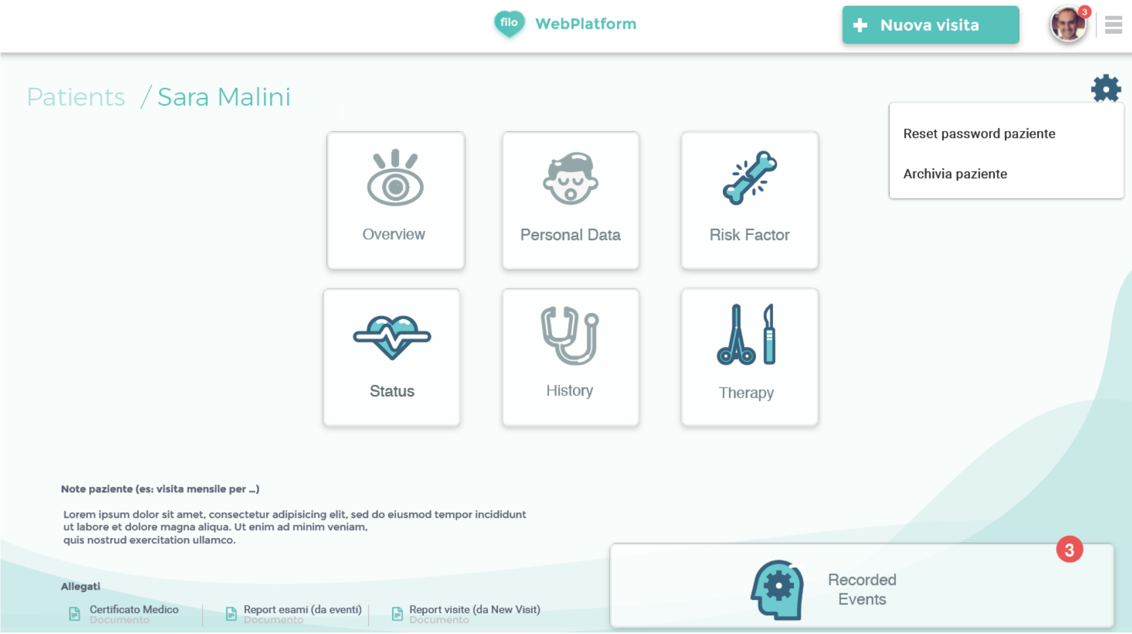


Figure 1.13: Patient Page (Screen)

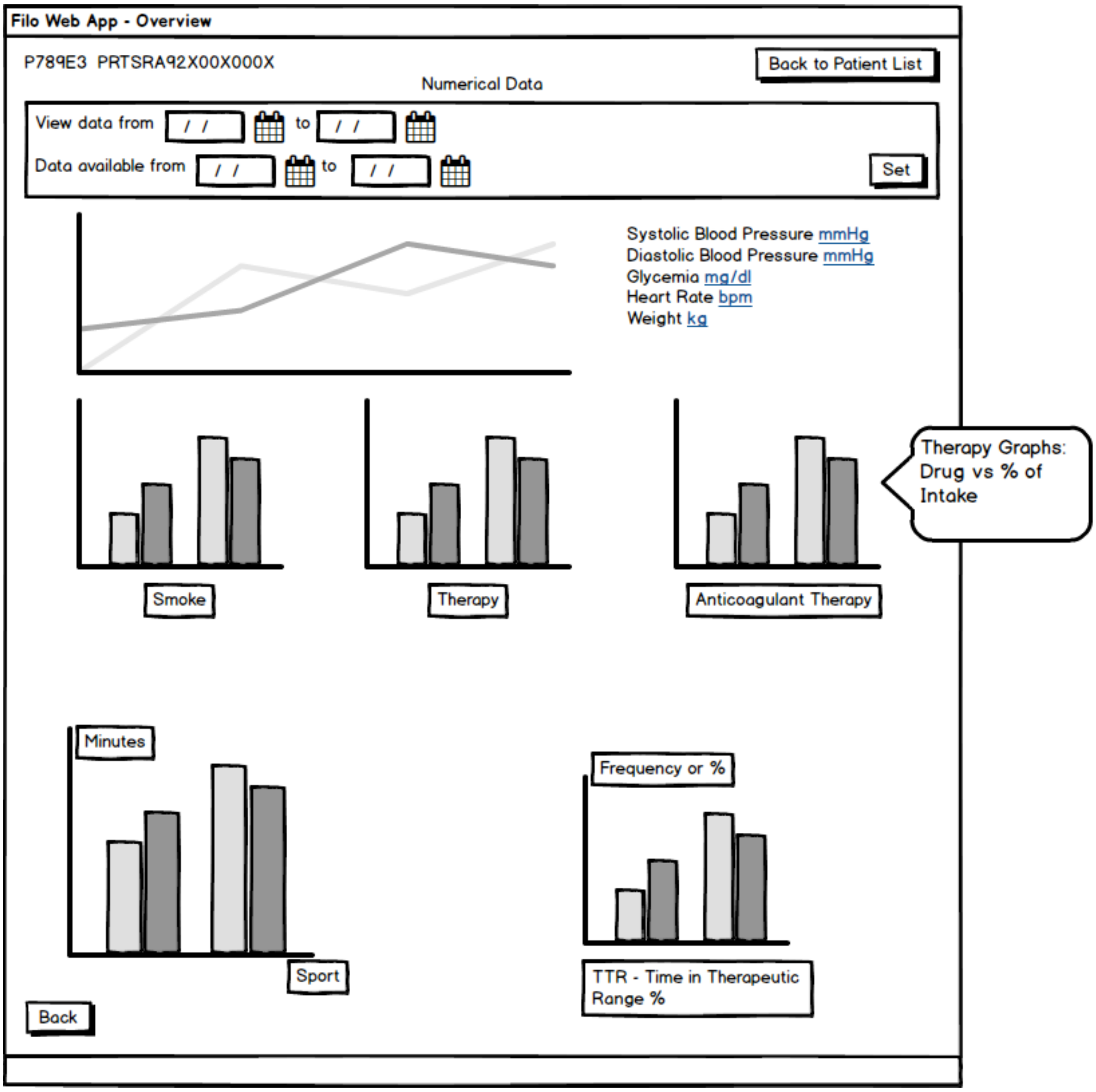


Figure 1.14: Overview (Mockup)

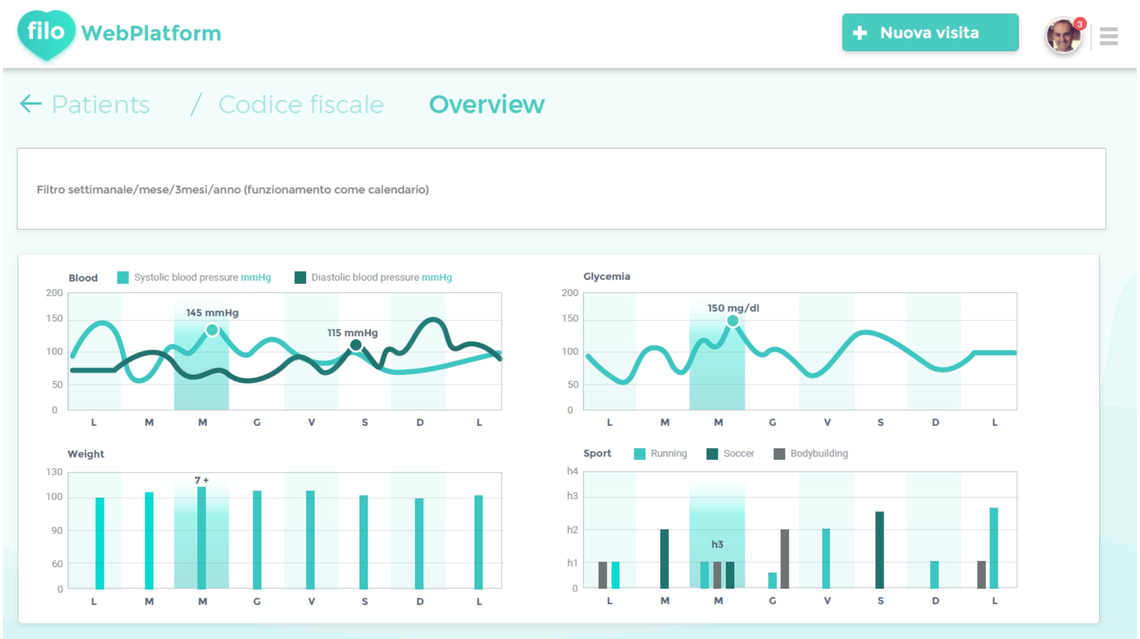


Figure 1.15: Overview (Screen)

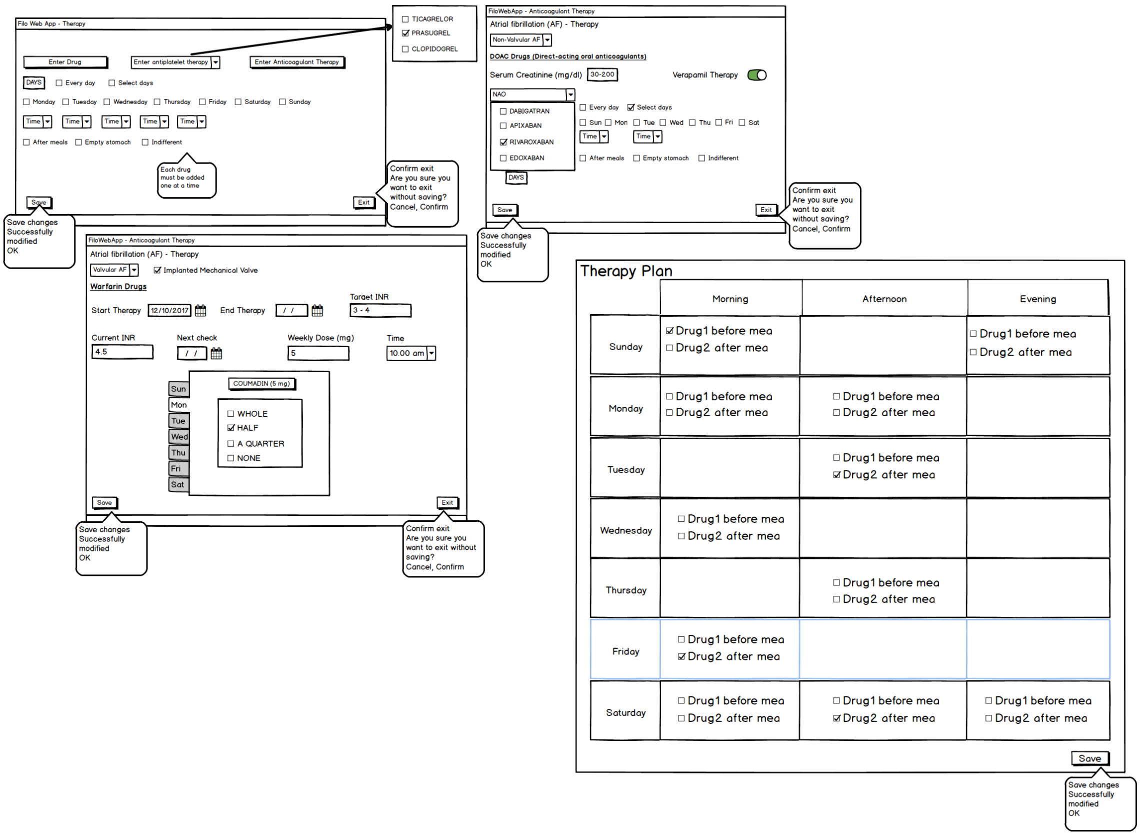


Figure 1.16: Therapy (Mockup)

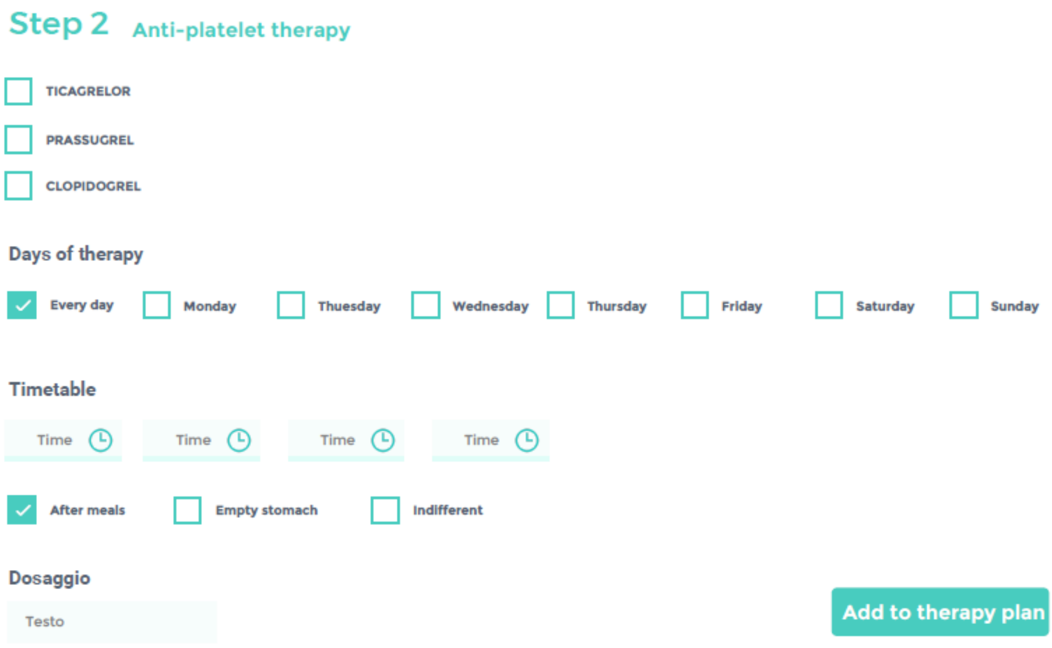


Figure 1.17: Therapy (Screen): anti-platelet therapy

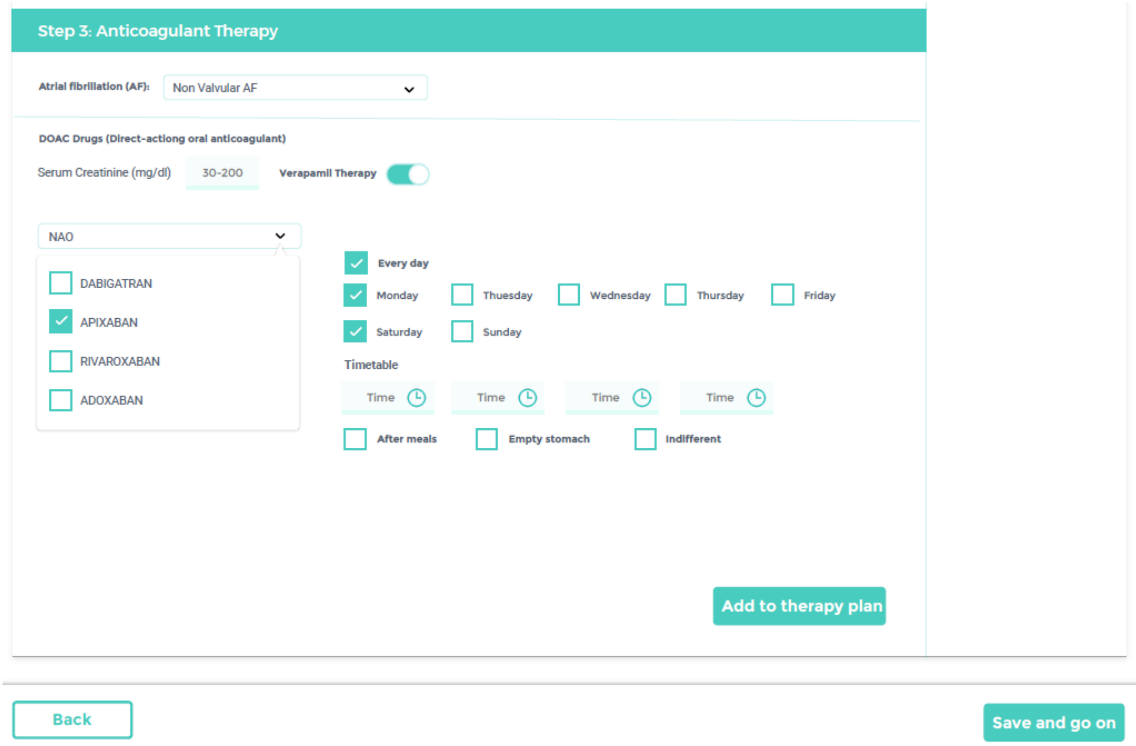


Figure 1.18: Therapy (Screen): anti-coagulant therapy

# Chapter 2

# Data security and GDPR

The security of the system is very important as without it its whole realization could not take place both as regards the mandatory legal aspect to be followed and respected, as the reliability aspect. If the minimum security and data protection requirements are not satisfied, it is very likely that it will be rejected by the ethics committee or not considered as a solution to the problems concerning the transmission of data by private or public hospitals. According to art. 5 of the [7], it is important to guarantee:

- confidentiality;
- integrity of the security system;
- availability and authenticity of data;
- transparency of operations.

### 2.0.1 State of the art: why not Firebase?

In the previous thesis work, the data storage system was based on Google Firebase [8]. The Google Firebase platform is one of the most popular for the development of mobile and web applications since it offers the possibility of using APIs and a cloud database and stores data in JSON format, allowing it to be synchronized in real time with all connected clients. The advantage is that it is enough to sign up for an account to have a NoSQL database available that can be quickly integrated by the developers. The problem with Firebase, however, concerns security measures. When developers fail in establishing database authentication or the cloud instance that supports their applications, in fact, the data breach danger becomes very high. Firebase, in particular, is dedicated to supporting mobile apps and contains detailed information about the user. According to [9]: "Over the years, Appthority analysts have detected 2,300 unprotected Firebase instances, totaling over 100 million user records exposed. Researcher-mapped photography included highly sensitive data, such as personal identification information, medical records and over 2.6 million plaintext passwords. " If these data were compromised by hackers, it would be a great danger to companies involved considering the extent of possible administrative sanctions and the damage to reputation that would ensue. Firebase is not natively protected: when it is installed for the first time, developers have to take care of protecting every single table and row of the database. Although this is not a particularly complex activity, programmers may not have time to follow the development life cycle, applying the correct security controls. However, in this way, it is easier to expose sensitive data, like the clinical one, because developers use a cloud-based infrastructure. The exposure of data to indiscriminate access through the Internet is one of the typical risks of cloud services. This is a risk that can be mitigated through adequate authentication. In the case of Google Firebase, authentication is the questionably implemented part. It is very important to consider the encryption of personal data during transfer through the internet when stored on mobile devices such as smartphones. In conclusion, unprotected Firebase databases show that even if there is greater awareness of cybersecurity issues, sensitive data can be compromised by the most trivial errors. Every day new threats are discovered that require significant effort or technical expertise. Unprotected Firebase instances are simple to exploit and therefore are much more likely to degenerate into a data breach that impacts the organizations involved. It is thus very important to invest more in programming activities, also dedicating time to the topic of secure development to ensure that threats are identified and resolved before the data is exposed, minimizing the risk of data breaches. Considering that in the case of CardioFilo the data concerns the health of patients, the problem is really important. In fact, one of the future developments of this thesis will concern the definition of an effective 'privacy system' to secure the data storage system both in the short and long term.

### 2.0.2 GDPR. General Data Protection Regulation

The GDPR [7] is the main European law on data protection and natural person privacy rights in force since may 24th 2016 and applicable since may 25th 2018 (art. 99 [7]). The GDPR is intended for all entities that process personal data of European citizens. The new legislation applies to companies in order to offer direct protection to citizens. With the GDPR, the definition of personal data and special categories of data is expanded. The GDPR defines those that we are used to call "sensitive data" as "special categories of data" The Regulation, in fact, identifies four categories of interest:

- personal data: any information that can identify the person, including name, surname, physical characteristics and even online identification;

- genetic data: data obtained by DNA or RNA analysis from a biological sample;
- biometric data: any physical identifying characteristic of the person, such as the fingerprint or the facial image;
- health data: any data relating to health, both physical and mental, present, past or future.

In order to be compliant to the new regulation, companies must:

- prepare a disclosure on the processing of personal data that is transparent, clear and easily accessible;
- allow the data subject to give his consent explicitly and trackably;
- ensure that the data collected is relevant, adequate and limited to the purposes for which it is requested and processed.

Therefore, in order to process personal data, companies must receive explicit user consent. But consent is only one of the conditions of lawfulness defined by art. 6 and, with regards to the particular data, from the art. 9. Moreover, in Italy, the new criteria of Legislative Decree 196/2003, amended by Legislative Decree 101/2018, are relevant to the subject. In particular the articles 2-sexies and 2-septies. On the other hand, in the medical field, the provisions on informed consent and the anticipated treatment provisions must also be considered, as set out in L 219/2017. In the request for consent, the company must clearly express what the purposes of data processing are and whether these are used by third-party companies (for example for advertising purposes). Therefore, the request must be clear, understandable and must be presented on an easily recognizable screen. The data subject must be guaranteed the right to withdraw his consent at any time and to express again the explicit consent in case of modification of the data processing or addition of new services to the program used and that involves the use of the data subjects' personal data. Therefore the data shall be collected and used only for the purposes explicitly indicated in the consent. Furthermore, the principle of 'accountability' is introduced for companies in the process of adaptation to the GDPR, having to guarantee maximum security from the collection stage up to the processing and storage of data.Not only: the responsibility for data protection extends to their complete destruction or anonymization. The regulation introduces the right to the "data portability" of one's personal data, to give the possibility to any user to transfer data from one data controller to another. The obligation for the data controller to communicate any violations of personal data (data breach) to the supervisory authority has been regulated. Failure or delayed communication exposes to the possibility of administrative sanctions.

### The Data Protection Officer (DPO)

Among the main obligations under the new regulation is that, for some companies, of inserting the figure of the DPO, the Data Protection Officer. The main task of the DPO is the observation, evaluation and management of the processing of personal data in order to ensure compliance with European and national regulations on privacy. The DPO is a professional figure, with particular skills in information technology, law, risk assessment and process analysis and with specialized knowledge of the rules and administrative procedures that characterize the sector in which it operates. Especially in sensitive sectors such as health, it must also demonstrate specific skills with respect to the types of treatment performed by the owner.

### The right to be forgotten

The right to the deletion of data is a particular form of guarantee that provides for the non-diffusion, without particular reasons, of previous prejudices for the honor of a person. Consequently, the data owners must be guaranteed the right to request that their personal data be deleted and no longer processed if they are no longer necessary for the purposes for which they were collected and processed. For its part, the data controller has the obligation to cancel, without unjustified delay, the personal data of those requesting it. The rights, in addition to those defined by the articles from 15 to 22, are also the right to be informed, to modify/withdraw the consent, to make a complaint to the supervisory authority and to obtain compensation for damages (if there have been damages).

### Privacy and transparency

The new regulation also deals with the complicated link between privacy (the right to privacy) and transparency (the right to always be able to access information). The legislation does not directly modify the national rules on access to administrative documents, but underlines the absence of a contradictory relationship between the two aspects, as both the values of "transparency" and those of "effective protection of confidentiality" they are considered worthy of protection.

## 2.0.3 Future developments

In order to effectively protect CardioFilo patient data, the European regulation now in force will be studied in depth. To do this, in all companies the so-called Data Protection Impact Assessment (DPIA) should be developed. A DPIA is the most important process to affirm to comply with the GDPR and is designed to achieve three targets:

- ensure compliance with regulations, and applicable legal policy requirements for privacy;
- determine the risks and the resulting effects;
- evaluate protections and any alternative processes to mitigate potential risks on rights and freedoms of natural person.

A DPIA creates an alarm system to detect privacy problems by building guarantees in advance and avoiding subsequent problems; avoids costly privacy errors; provides proof that the organization has tried to avoid privacy risks (negative publicity, damage to reputation); improves decision-making; increases customer confidence; shows that the organization takes privacy seriously. A successful DPIA should:

- predict possible risk situations based on the type of data breach, the resources involved, the probability of the event and the severity of the consequences;
- set up a continuous monitoring system able to promptly report events related to privacy risk;
- prepare appropriate corrective measures in the event of an accident by informing the interested parties and the competent authority.

These last aspects will be discussed in the near future in the development of an effective security system for the protection of CardioFilo data.

# Chapter 3

# Introduction to openEHR

## 3.1 Understanding openEHR: a multi-disciplinary approach

Population aging and an increase in life expectancy are some of the most relevant issues for the health system worldwide. The prevalence of chronic diseases is inevitably increasing. The elderly population is the category most affected by comorbidity, i.e. the coexistence of multiple chronic diseases at the same time. In particular, patients of the CardioFilo application not only belong to the elderly population but are also affected by diseases that often involve (or coexist with) chronic diseases. For these patients, health care should consist in a collaboration not only among all health care professionals, such as cardiologists, family doctors, nurses and pharmacists but also computer and biomedical engineers. A multidisciplinary exchange of information, knowledge and medical and technical evaluations can be decisive in the successful management of the health of the elderly population with consequent reduction in morbidity and mortality. For this exchange to be successful, a health information system is required that provides an efficient representation of clinically relevant information and effective computerized decision support to healthcare professionals and patients. It is in this context that the need arises of a "common language" that allows all the actors involved to interact effectively. According to [10]: "[...]openEHR is feasible and easy-to-apply IT platform that possibly provide several benefits over single-level information systems in the care of elderly, comorbid population". openEHR is an open health standard that concerns electronic medical records and the standardization of the data contained in them (independent of vendors and technologies). Data is interoperable regardless of the programming language and database.
The keyword for the success of an openEHR-based system is 'semantic interoperability' necessary to realize an electronic health record (EHR) shared among all the various devices and organizations that may use it.

**Who is using openEHR?**

openEHR is now used worldwide, particularly by non-profit and open source organizations, governments and academic research who operate in medical research and public health [11].

## 3.2 openEHR systems

A system based on openEHR consists of an EHR repository, an archetype repository, terminology, and demographic information.
One of the most important features of openEHR is the possibility to completely separate the clinical data model from the demographic data model. openEHR is based on a 'two-level approach': clinical information is separated from technical knowledge. This allows for self-adapting and more maintainable systems.
The system architecture consists indeed of:

- Reference Model
- Archetype and Template Model

See Fig. 3.1

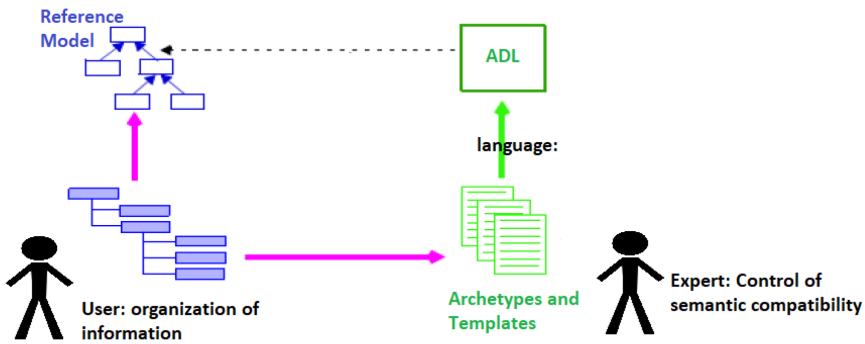


Figure 3.1: openEHR Models Interaction

### 3.2.1 The Reference Model

The openEHR Reference Model is basic information model that defines how the data must be represented in the medical record. For example, the openEHR RM model contains the definition of the data type or the medico-legal requirements.

### 3.2.2 The Archetype and Template Model

Archetypes are groups of data representing all the clinical knowledge. The same archetype can be used in several different cases giving the possibility to model data only once and to do it according to the standard. The international archetype library is the Clinical Knowledge Manager (CKM) and contains about 500 archetypes. An archetype defines a complete, precise and clinically meaningful concept. This provides the basis for the semantic interoperability of clinical knowledge. Archetypes can be divided into classes according to their content (Fig. 3.2):

- Entry archetypes. They divided into: Observation archetypes, Evaluation archetypes, Instruction archetypes and Action archetypes.
- Composition Archetypes (actual documents containing clinical measures, e.g. 'Encounter');
- Section Archetypes (document headings);
- Structure Archetypes. A structure archetype models a structure such as a tree or a list of items and allows reuse in other archetypes;
- Cluster Archetypes: they group items together. Cluster archetypes can be added to other archetypes;
- Element Archetypes. An element archetype represents a single item (that can be reused).

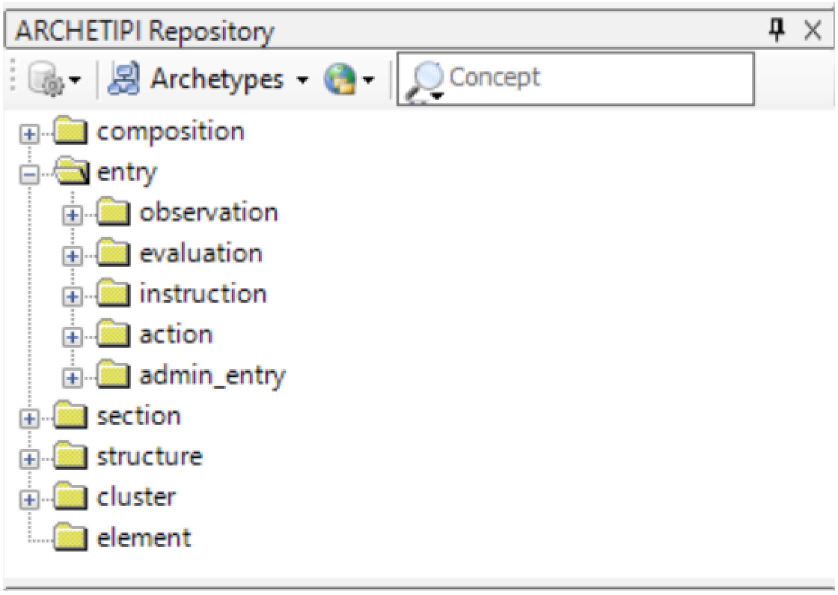


Figure 3.2: Schematic representation of structure of an Archetype Repository

An example of the observation archetype Blood Pressure is shown below:

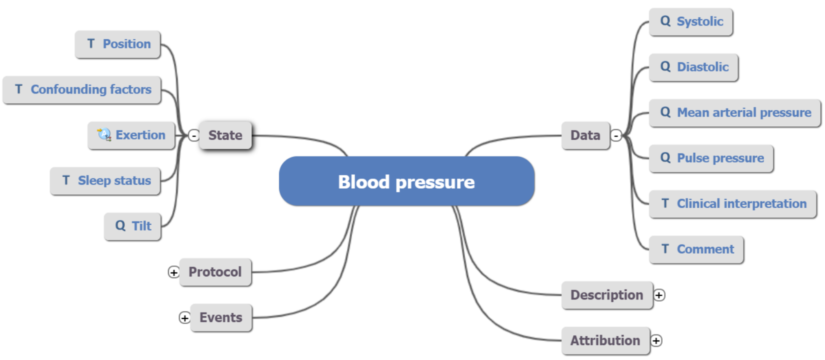


Figure 3.3: Blood Pressure Archetype Structure

Archetypes can be grouped together to create an openEHR Template, a third-level structure in openEHR. Templates are the data set of the openEHR model to be realized.

### 3.2.3 The EHR Server

Once the templates have been created, i.e. after having clearly defined the data model to be used in the openEHR system, it is necessary to generate 'Compositions' (different from the compositions in the archetype repository), the actual clinical documents that will contain the patient's clinical data. To design a health record starting from templates, a system is required that manages the sending and receiving of clinical data. There are proprietary systems and open source systems that perform these tasks. In particular, an example of openEHR Industry Partner is Marand [12], a Slovenian company operating in the ICT field and in the healthcare field. Their Think!EHR platform is an archive of health data useful for the development of health applications. An opensource system was used compliant with openEHR, called Cabolabs [13]. Cabolabs system was created by a community of developers and is based on Java technology. By downloading an instance of Cabolabs, it is possible to upload the templates on the Cabolabs EHR server and create a medical record. The Cabolabs EHR Server allows clinical data management and sharing according the openEHR standard. The operation of the system is based on the HTTP protocol. Once the templates are loaded on the platform, it is possible to create the actual health records through APIs (Application Programming Interface). APIs are a tool that allows programmers mot to rewrite every time all the functions necessary for the program. It is possible to add documents to the health record later. All happens through simple CRUD operations (Create, Read, Update, Delete). It is possible to upload a document in the EHR server and create a Composition using the HTTP APIs. Once the clinical record has been created, it is possible to modify it. All changes are stored in contributions. Contributions show all the changes made. Each contribution refers to one or more items in the health record that have been reviewed by the user. However, the EHR platform used during this thesis work is still under development. openEHR world is still evolving and growing and it will take some time to have a complete, standardized and functional system that can be used.

## 3.3 CardioFilo and openEHR

### 3.3.1 Tools

In the development of an openEHR system, the following tools are required:

- Clinical Knowledge Management (CKM): the public openEHR repository for archetype and template that has been developed by the community of openEHR [14];
- Archetype Editor (AE) of Ocean Health Systems. A tool for creating archetypes base on the Archetype Definition Language (ADL) language[15];
- Template Designer (TD) of Ocean Health Systems. A tool for grouping archetypes for creating custom templates [16].

### 3.3.2 Cardiologists (Web Platform)

To model CardioFilo web application data, the following templates were realized:

- Risk Factors Template (see Figure 3.4);
- Blood Pressure Entry Template (see Figure 3.5);
- Doctor Summary Template (see Figure 3.6).

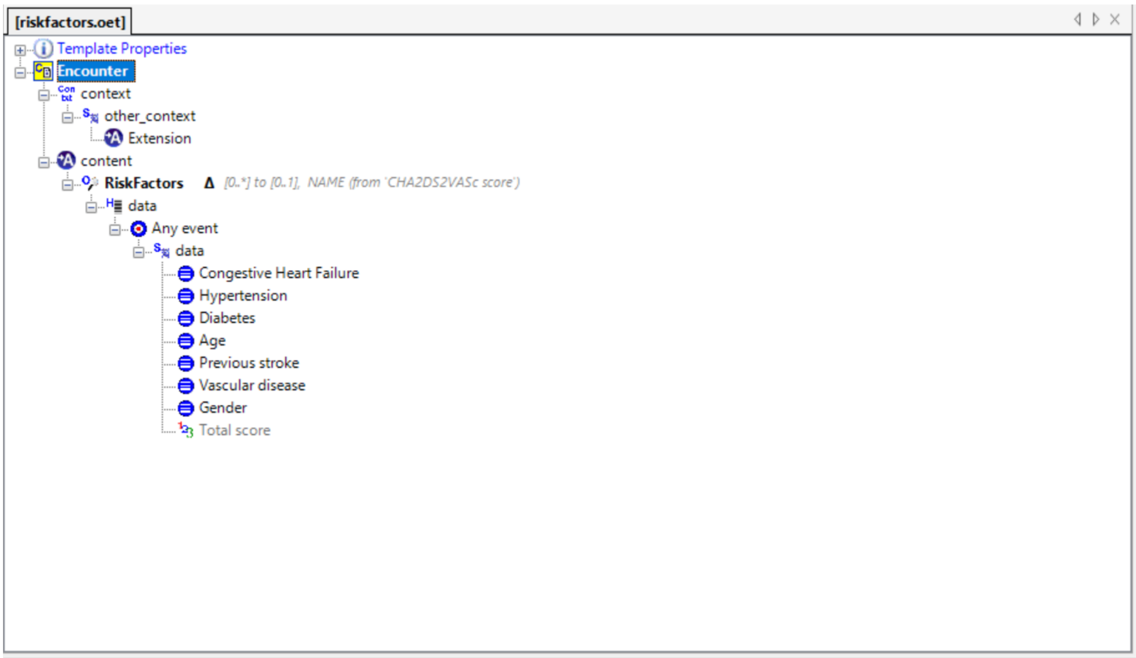


Figure 3.4: Risk Factors Template

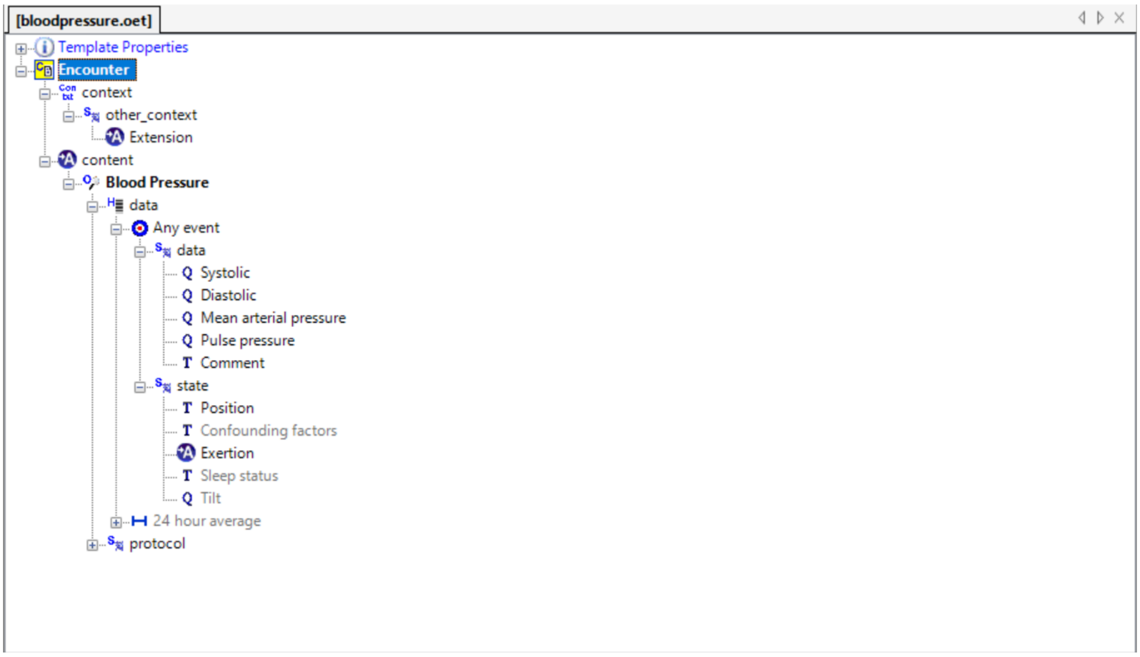


Figure 3.5: Blood Pressure Template; Q stands for Quantity field and T for Text field.


Figure 3.6: Doctor Summary Template; E stands for Evaluation, O stands for Observation.

Each template is composed by archetypes. The archetypes were downloaded from the

CKM and analyzed. Those compliant with the CardioFilo use case were chosen to compose the templates.

**Archetypes for Risk Factors template**

- COMPOSITION: Encounter;
- OBSERVATION: Chadsvas score (renamed RiskFactors).

**Archetypes for Blood Pressure Entry template**

- COMPOSITION: Encounter;
- OBSERVATION: Blood pressure.

**Archetypes for Doctor Summary template**

- COMPOSITION: Encounter;
- SECTION: Patients background (renamed Doctor summary);
- SECTION: Medication order list
- EVALUATION: Reason for encounter;
- EVALUATION: Clinical synopsis;
- EVALUATION: Problem diagnosis;
- EVALUATION: Container;
- OBSERVATION Story;
- OBSERVATION: Imaging exam;
- OBSERVATION: Ecg result;
- CLUSTER: Medication;
- CLUSTER: Timing nondaily.

### 3.3.3 Patient (mobile Application)

To model CardioFilo web application data, the following templates were realized:

- 'selfmeasure' Template (see Figure 3.7);
- 'patientdata' Template (see Figure 3.8).

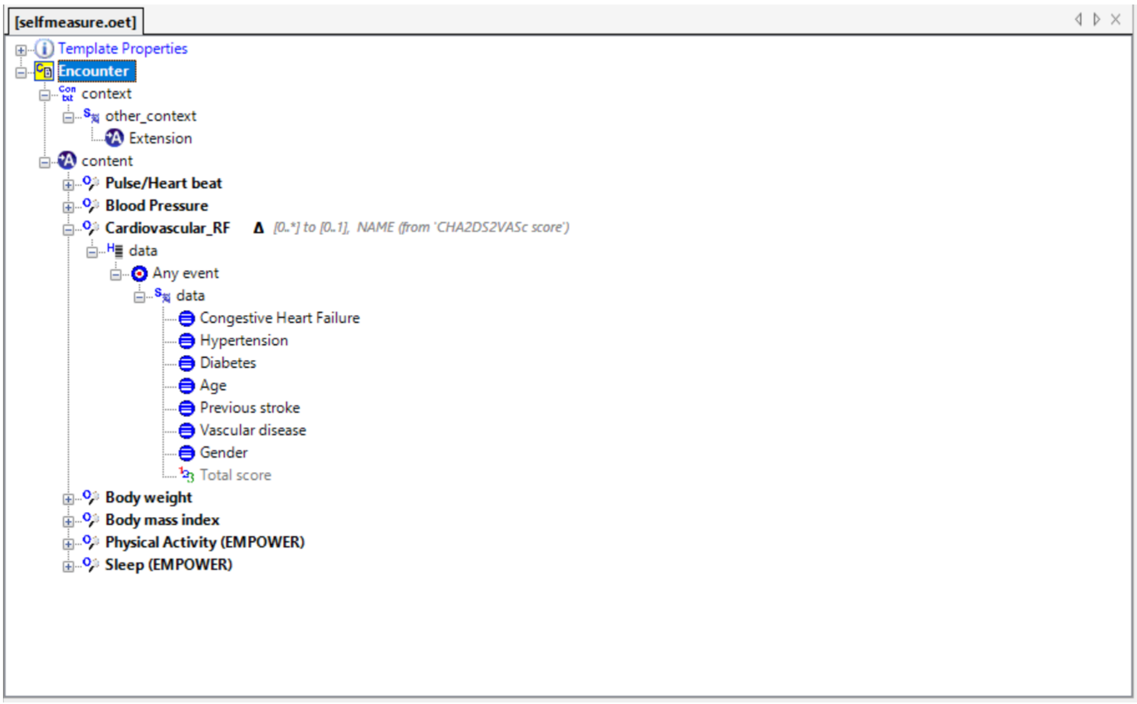


Figure 3.7: Clinical Data Entry (1)


Figure 3.8: Clinical Data Entry (2)

#### Archetypes for 'selfmeasure' template

- OBSERVATION: Pulse;
- OBSERVATION: Blood pressure;
- OBSERVATION: Chadsvas score (renamed Cardiovascular RF);
- OBSERVATION: Body weight;
- OBSERVATION: Body mass index (BMI);
- OBSERVATION: Empower physical exercises;
- OBSERVATION: Empower sleep;
- OBSERVATION: Heart failure symptom questionnaire.

#### Archetypes for 'patientdata' template

- SECTION: Adhoc (renamed Adverse Reaction);
- EVALUATION: Adverse reaction risk;
- EVALUATION: Health risk;

- EVALUATION: Medication safety event;
- EVALUATION: Tobacco smoking;
- OBSERVATION: Substance use;
- OBSERVATION: Alcohol intake.

### 3.3.4 Problems encountered in the design of clinical models

During templates design, it was not always possible to find the archetypes that perfectly match the features defined in the requirements collection phase. Sometimes the "hidden" option was used to omit unnecessary clinical information. However, looking for alternative solutions, some "ad hoc" archetypes are available on the public repository CKM. To make up for the lack of an "Allergies" archetype, for example, there is a SECTION archetype called "Adverse Reaction List" and an EVALUATION archetype ("Adverse Reaction risk") in which physicians and patients can describe in detail problems related to allergic reactions. Regarding risk factors like Ejection Fraction, HIV in treatment, Chemotherapy/Radiotherapy, Dyslipidemia and Kidney failure, not available on the CKM, the "Health Risk Assessment" EVALUATION archetype allows to insert additional risk factors. For a precise and exhaustive description of all the surgical interventions undergone by the patient such as PCI or CABG, there are many EVALUATION archetypes (e.g. "Clinical Synopsis") that allow physicians to insert all the necessary information. Unfortunately, CKM has not yet provided archetypes that model some health data (such as HIV, radiotherapy or chemotherapy) important to the risk factors entry in the CardioFilo application. The task, indeed, was not easy either from a technical or a clinical point of view. Archetypes are defined as complete sets of data and therefore should include all possible use cases. Often the risk is that the archetype is inadequate. Integration and interoperability are important features, but sometimes the ideal separation between the clinical world and the technical aspects is not clear. It is necessary to build a system that integrates the knowledge of all professional fields together, in order to create a 'techno-clinical' team that develops the platforms, starting from the creation of specific and working archetypes, up to the birth of an electronic medical record that is good for everyone. [17]: 'Is not such a clear separation of technical and clinical concerns. The boundary between the technical and the clinical was rather blurry, where key actors on both sides were dependent on each other and each other's competence. [...] Developers and clinicians thus found themselves working together in both arenas where they could exploit and challenge each other's competence. Developing archetypes is in reality a complex issue that needs to involve many stakeholders.' The figure of the biomedical engineer could be included in this area. A biomedical engineer could be the right intermediary between the professional figure who takes care of technical aspects, important for the realization of the system's architecture, and the healthcare staff who has the clinical knowledge necessary for the correct implementation of the data models that will create the medical record.

# Chapter 4

# CardioFilo Web Platform

## 4.1 Django: the web framework

Django is a Python-based web framework used for the development of web applications [18]. In general, creating a website, the following components are always necessary:

- A system to upload files:
- An administration panel
- A system that manages users authentication (signup, login, logout).

A web framework like Django offers all this components ready to be used. To understand Django, it's necessary to talk about HTTP requests and responses (Fig. 4.1). When a request arrives to the web server, the request is sent to Django, which tries to understand what to do. The Django URLresolver searchs matches for the required URLs. A series of schema, i.e. models, are necessary in the attempt of matching request URLs. Django checks all models and if a match is found, it passes the request to an associated function called view. If the request is a letter in the mailbox (i.e. the web server), Django URLresolver is the postman who reads the address on the letter and checks every house number until it matches with the one on the letter. If the house number matches, the postman mails the letter.

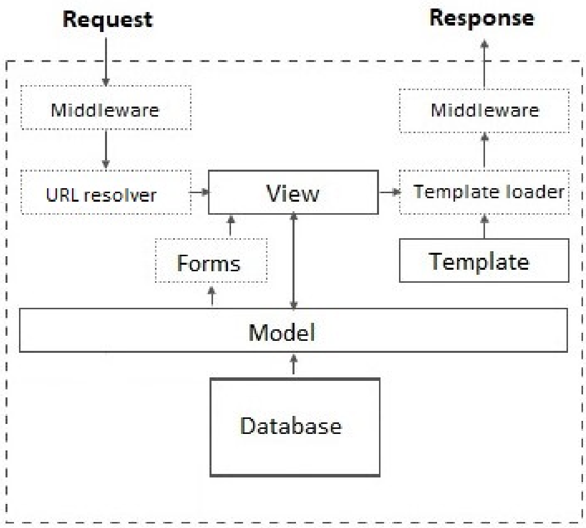


Figure 4.1: Django Request and Response Flow

A model in Django describes the logical structure of the data and how the various components relate to each other. A form describes how data are presented to end users. A form in Django is written in HTML language. As the fields of a Django model are associated with tables in the database, the fields of a form are associated with HTML forms. In the view function, it is possible to query the database searching for informations. Maybe the user asked for an update or a change in his data. The view checks if the user has permissions to modify data. If the user has the proper permissions, the view updates the changes and creates a response. Then Django sends it to the browser. The developer provides the model, the view and forms.The template then maps it to a URL and Django serves it to the user.

### Examples of forms and views for personal data entry

visit.py

```python
from django.shortcuts import render, redirect, render_to_response
from django.views import generic
from django.template import RequestContext
from .forms import VisitForm
from .models import Visit
from apps.Anagraphic.models import Registry

def anagraphic_show(request):

    context = RequestContext(request)

    if request.method == 'GET':

        data = Registry.objects.all()
    else:
        pass

    return render(request,'registry.html',{'data':data})
```

Figure 4.2: Django View

forms.py

```python
from django import forms
from .models import Visit
from apps.Anagraphic.models import Registry
from apps.History.models import History
from apps.RiskFactors.models import RiskFactors

class VisitForm(forms.ModelForm):

    class Meta:
        model = Registry
        fields = [
            'first_name', 'last_name', 'email',
            'fiscal_code', 'birthday', 'birth_place',
            'residence_city','address','city',
            'country','phone','gender',
            ]
```

Figure 4.3: Django Form

registry.html

```
{% block content %}
<h2>Patients Registry</h2>
<ul>
  {% for registry in data %}
    <li>
        <p>First Name: {{ registry.first_name }}</p>
        <p>Last Name: {{ registry.last_name }}</p>
        <p>Email: {{ registry.email }}</p>
        <p>Fiscal Code: {{ registry.fiscal_code }}</p>
        <p>Birthday: {{ registry.birthday }}</p>
        <p>Birth Place: {{ registry.birth_place }}</p>
        <p>Residence City: {{ registry.residence_city }}</p>
        <p>Address: {{ registry.address }}</p>
        <p>City: {{ registry.city }}</p>
        <p>Phone: {{ registry.phone }}</p>
        <p>Gender: {{ registry.gender }}</p>
    </li>
  {% empty %}
    <li>No data available</li>
  {% endfor %}
</ul>
    <a href="{% url 'patients' %}">Back to Patient List</a>
    <a href="{% url 'home' %}">Back to Home Page</a>
{% endblock %}
```

Figure 4.4: HTML Form

The architecture of every Django project is based indeed on the MVT Model: Model-View-Template Model(see Fig. 4.5). The components of an MVT Model are:

- Model Layer: an abstraction layer used to structure and manipulate data.
- View Layer: logic responsible for request/response processing. In Django, web pages and other contents are available thanks to views. The view recovers data from the database using models, it formats and groups data in a HTTP response which is sent to the client (browser).
- Template Layer: series of designer-friendly commands to show informations to the user.

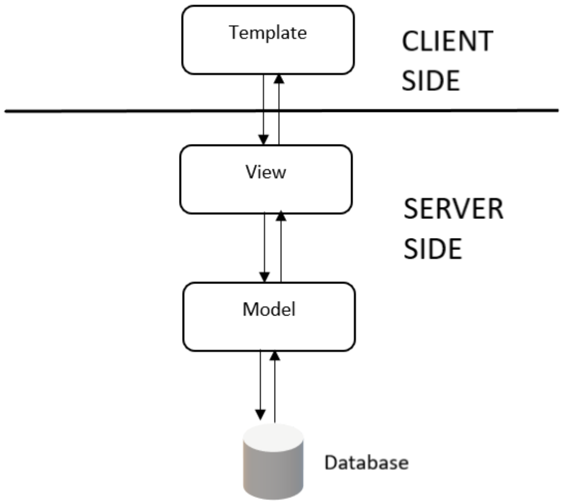


Figure 4.5: Django Model Architecture

The fondamental unit of a Django web application is the Django project. A Django project consists of one or more Django apps, i.e. autonomous packages. Django itself is a group of apps, each one is developed to do one precise task.

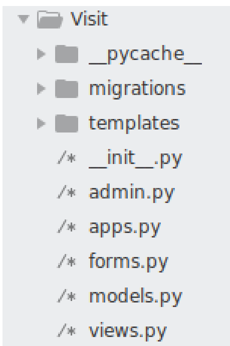


Figure 4.6: CardioFilo App Folder Structure

In particular, CardioFilo apps are:

- Anagraphic: contains all the objects for the insertion of patients personal data;
- History: contains all the objects for the insertion of pressure measurements and text fields that describe the patient's medical history;

- Pathologies: contains all the objects for insertion of the type of pathology from which the patient is suffering

- Risk Factors: contains all the objects for the insertion of patients risk factors;

- Status: contains all the objects for the insertion of informations about clinical measurements (height, weight, ejection fraction etc) and surgical procedures.

- Therapy: contains all the objects for the insertion of the therapeutic plan and informations related to the therapy (dosage, time, type of drug, etc.)

- Visit: recalls all the objects contained in the other apps and refers to the flow of operations performed by the cardiologist during a visit

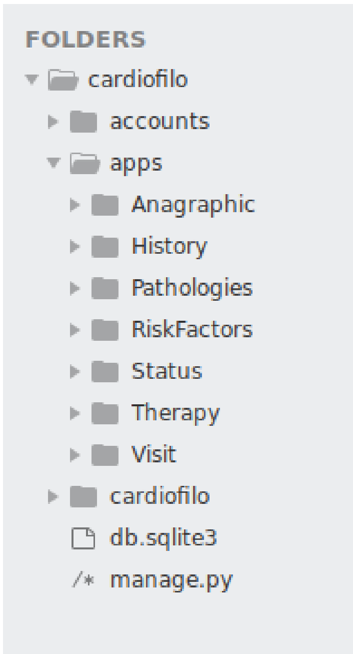


Figure 4.7: CardioFilo Project Folder Structure

The built-in apps, already included in Django, are not visible in the project.

## 4.2 Notes on databases: SQL

For systems like CardioFilo it is essential to use a database. The database is a complex software structure able to manage access and modification of large amounts of data stored on a storage device. In the specific case of this thesis work, a relational database was used. To interact with this type of database there is a language called SQL (Structured Query Language). SQL is a language considered in all respects a standard A database is a collection of data that is managed and organized by a specific software, the DBMS (DataBase Management System). A DBMS is basically a layer between the user and the actual data. Through this intermediate layer the user and the applications do not access the data as they are actually stored, but they only see a logical representation of data. This allows independence between applications and the physical storage of data. It is possible to decide whether to store the data or even to change the DBMS without the applications being affected. The logical representation of that data is not changed. This logical representation is called the 'Database schema'. The relational model (RDMS) has become the most used for the production of DBMS. The fundamental structure of the relational model is the "relation", that is a two-dimensional table made up of lines (tuples) and columns (attributes). Tuples in a relation are an unordered collection of different elements. To distinguish one tuple from another, the concept of "primary key" is used, that is a set of attributes that make it possible to univocally identify a tuple in a relation. Each attribute of a relation is characterized by a name and a domain. The domain indicates which values can be assumed by a column of the relationship. The DBMS will check if only the values allowed by their domains are entered in the attributes of the relations. The fundamental characteristic of the domains of a RDBMS is that they are "atomic": the values contained in the columns can not be separated into values of simpler domains.

#### Use of a database

Thanks to databases, developers can:

- create and modify database schema;
- enter, edit and manage stored data;
- query the stored data;
- create and manage data control and access tools.

In CardioFilo project SQLite [19] was used, a software library written in C language that implements an RDBMS within applications. It allows developers to create a database embedded in a single file.

# Chapter 5

# Software as a Medical Device (SaMD)

Mobile health constitutes a new model of health whose success is linked to the ability to make significant improvements to public and individual health, thanks to the possibility of providing services to anyone, anywhere, anytime and with any device. However most of the web and mobile applications have not been even subjected to any type of control by the competent authorities, being available for download in the common app stores, sometimes responding to generic well-being needs, in other cases representing a real tool for diagnosis and treatment. The need for some regulation therefore becomes a priority and this need will become more and more pressing with the further developments of technology and the greater use of smartphones, tablets and apps.

## 5.1 Overview on Medical Device Regulation

### Definition of Medical Device

Article 1(2)a of Directive 93/42/EEC [20]: "'medical device' means any instrument, apparatus, appliance, software, material or other article, whether used alone or in combination, including the software intended by its manufacturer to be used specifically for diagnostic and/or therapeutic purposes and necessary for its proper application, intended by the manufacturer to be used for human beings for the purpose of:

- diagnosis, prevention, monitoring, treatment or alleviation of disease,
- diagnosis, monitoring, treatment, alleviation of or compensation for an injury or handicap,
- investigation, replacement or modification of the anatomy or of a physiological process,

- control of conception,

and which does not achieve its principal intended action in or on the human body by pharmacological, immunological or metabolic means, but which may be assisted in its function by such means;[...]"

**Current Legislation**

- Council Directive 93/42/EEC on Medical Devices (MDD) (1993);
- Legislative Decree 24/02/1997, No 46 Implementation of Council Directive 93/42/CEE on Medical Devices.

New Regulation on Medical Devices:

- Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices, amending Directive 2001/83/EC, Regulation (EC) No 178/2002 and Regulation (EC) No 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC.

Others:

- MEDDEV 2.1/6 July 2016 Guidelines on the Qualification and classification of stand alone software used in healthcare within the regulatory framework of medical devices.
- Manual on borderline and classification in the community regulatory framework from medical devices 12/2017.

## 5.2 SAMD

### 5.2.1 CardioFilo: software as a Medical Device?

According to the the International Medical Device Regulators Forum (IMDRF) [21]:

> "The term "Software as a Medical Device" (SaMD) is defined as software intended to be used for one or more medical purposes that perform these purposes without being part of a hardware medical device."

Already in the previous legislation on medical devices (EC Directive 93/42 and subsequent amendments, implemented in Italy with Legislative Decree 46/1997), the software was expressly included among the medical devices if it had a medical use, according to what specified in the art. 1 of the Directive.
The qualification of a software as a medical device involved the application of the Directive

with reference to the general safety requirements, the classification of the devices and the specific rules laid down for their certification. Furthermore, Annex II was applied, in the part in which it provided for some rules concerning the development of this type of product. With the ruling of 7 December 2017, the Court of Justice of the European Union established that a software in itself constitutes a medical device if it is specifically intended by the manufacturer to be used for medical diagnostic purposes. The software can be considered a medical device even without human use. Figure 5.1 shows the necessary steps to qualify a software as medical device according to [22].

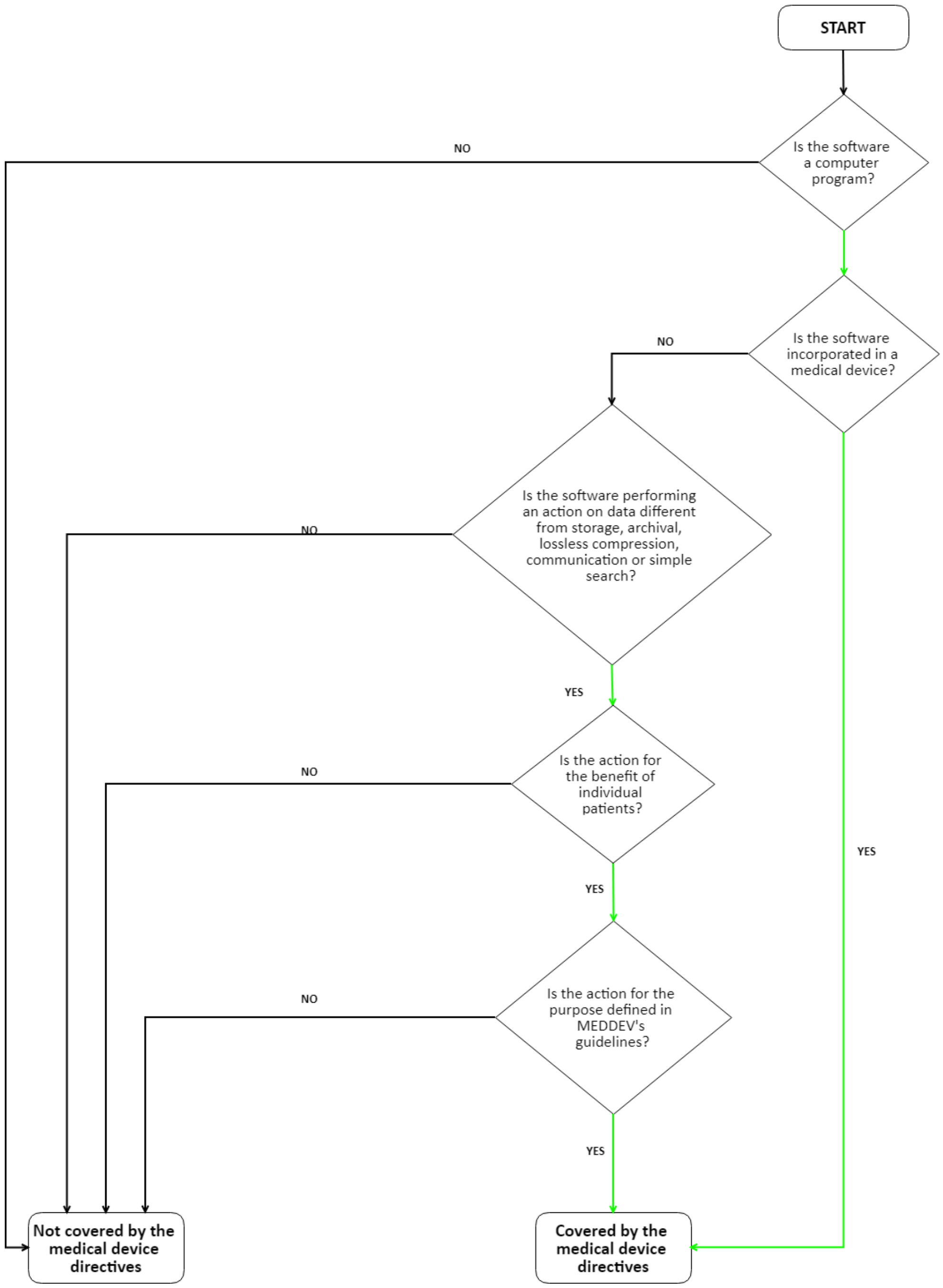


Figure 5.1: Decision Diagram for qualification of software as a medical device

In detail, the question is whether Directive 93/42 should be interpreted as meaning that software that has at least one functionality that allows the use of a patient's personal data in order to help doctors in preparing a therapy plan (in particularly contraindications, interactions with other drugs and excessive dosages) constitutes a medical device even if it does not act in or on the human body. Article 1, paragraph 2, letter a) of the MD directive states that a software constitutes a medical device when it is intended by the manufacturer to be used on humans for the purposes of diagnosis, prevention, control, therapy or mitigation of a disease. According to this directive, therefore, to be qualified as a medical device, a software must satisfy two conditions: to be a "medical device" it is not sufficient that the software is used in medical field, but it must be intended by the manufacturer to be used for medical purposes. In fact an application like CardioFilo, that cross-checks the patient's personal data with the drugs that the doctor intends to prescribe and that is used for prevention, control and therapy or mitigation of a disease, pursues a specifically medical purpose. This circumstance therefore makes it a medical device within the meaning of the MD Directive. This does not happen, however, in the case of software that, although intended to be used in a medical context, has the only purpose of storing and transmitting data, such as software that stores the patient's health data, or a software intended to indicate the contraindications mentioned by the manufacturer of drugs. Regarding the action produced by the device in or on the human body, once the existence of the medical purposes has been assessed, accordingly to the Court, to qualify a software as a medical device it is necessary to concentrate on the purpose of its use and not on how the effect it can produce on or in the human body can occur. In conclusion, it should be considered medical device that software used to create or modify the clinical data collected, through processes of calculation or comparison of the data, in order to provide information on that patient. CardioFilo for example allows to monitor the blood pressure or the glycemia, automatically processing information and providing real-time responses to the patient. On the contrary, software should not be considered medical devices that do not perform any action on the data or whose action is in any case limited to storage, archive or compression without loss of data. On this point, denying a device that does not act directly in or on the human body the quality of a medical device would in practice exclude from the field of application of Directive 93/42 the software that is specifically intended to be used for one or more medical purposes included in the definition of a medical device. For the purpose of qualifying as a medical device, it is therefore irrelevant the fact that the software acts directly or does not act directly on or in the human body, being essential that software has been designed to be used for one or more of the established medical purposes. It follows that, among other functionalities, allows the use of sensitive patient data for the purpose of treatment, diagnosis and rehabilitation, constitutes, as regards these functions, a medical device pursuant to article 1 , paragraph 2, letter a) of Directive 93/42, and this even if the said software does not act directly in or on the human body. Consequently, and as a medical device, this software - pursuant to art. 17, paragraph 1, of the Medical Devices Directive - it must obligatorily bear the CE conformity marking at the time of its marketing. In summary, all medical applications and software for medical diagnostic purposes must now be CE marked as a medical device pursuant to the Directive 93/42/EEC, and soon as a class IIa medical device in accordance with Regulation 2017/745, regardless of whether they are used in or on the human body.

# Chapter 6

# Conclusions

Health care should be safe, effective, patient-focused, timely, efficient and fair. To achieve all these goals it would be preferable to apply care and support services to patients directly in their homes and through the use of tools based on new technologies such as sensors, communication platforms and smartphones. Telemedicine is a valid innovative means that is a link between a qualified medical assistance service and the use of a technology that can increase, improve or simply maintain the functional abilities of patients with disabilities. It only requires an interdisciplinary organization to take into account patient information and awareness, the choice of the best therapy for the individual, the education of relatives or caregivers, the management of complications and follow-up care etc. Demographic factors such as population aging and others will lead to an ever increasing demand for home care. Technological innovations can only accelerate and optimize this process. CardioFilo aims to the development of a system that is simple to manage and brings significant benefits to patients and doctors in the processing of data. In the near future, the CardioFilo project will be continued in order to optimize and simplify the doctor-patient relationship. The realization of software and in general of systems for the management of public and private health in a telematic way concerns an ever closer future, where economic and physical barriers (e.g. distance from hospital centers) will be overcome, and the quality of life of patients will be improved.

# Part I

# Appendix

# Appendix A

# Appendix A

## A.1 UML Diagrams: Cardiologists

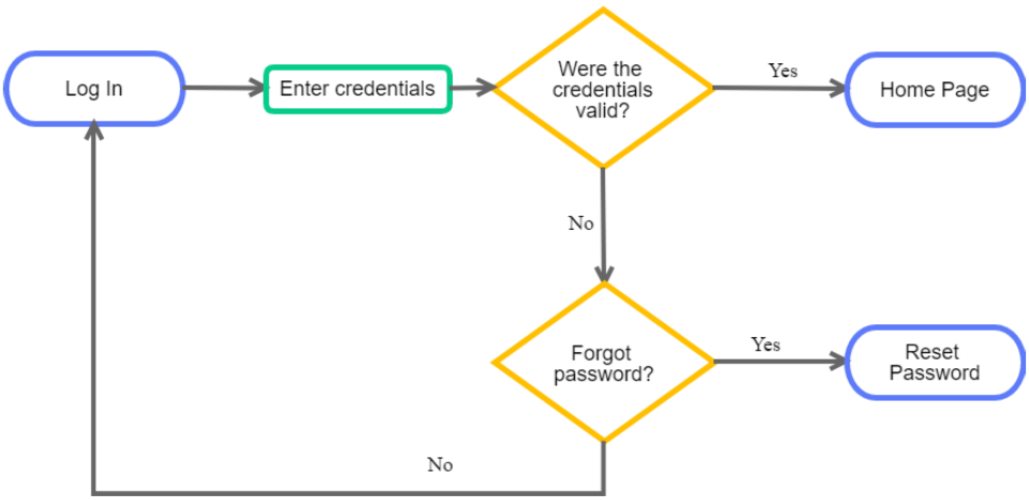


Figure A.1: Login


Figure A.2: Logout

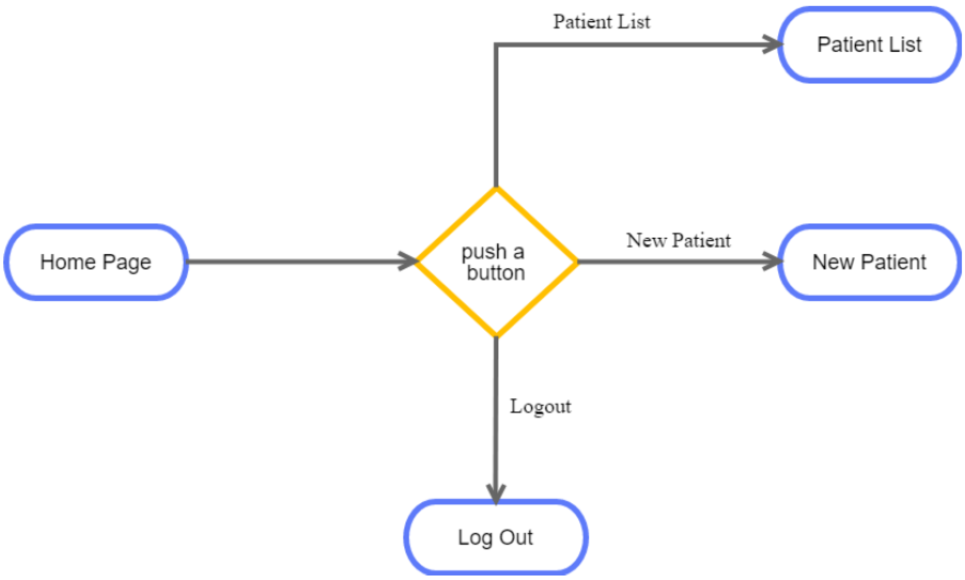


Figure A.3: Access to HomePage

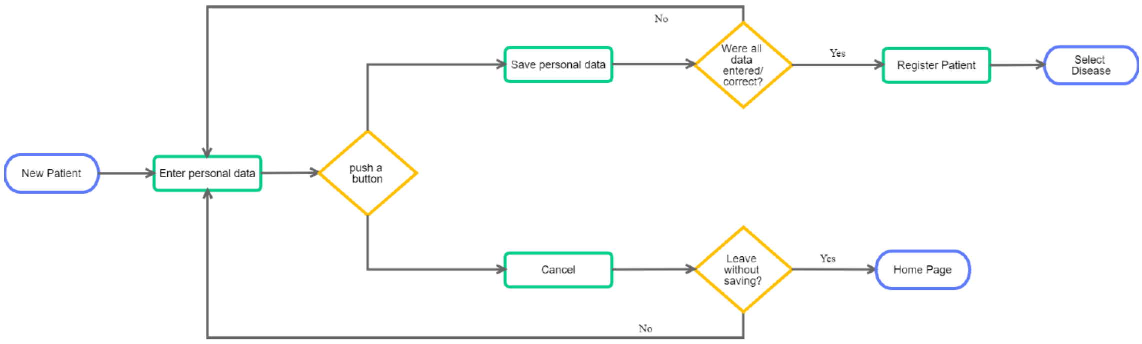


Figure A.4: New Patient Page

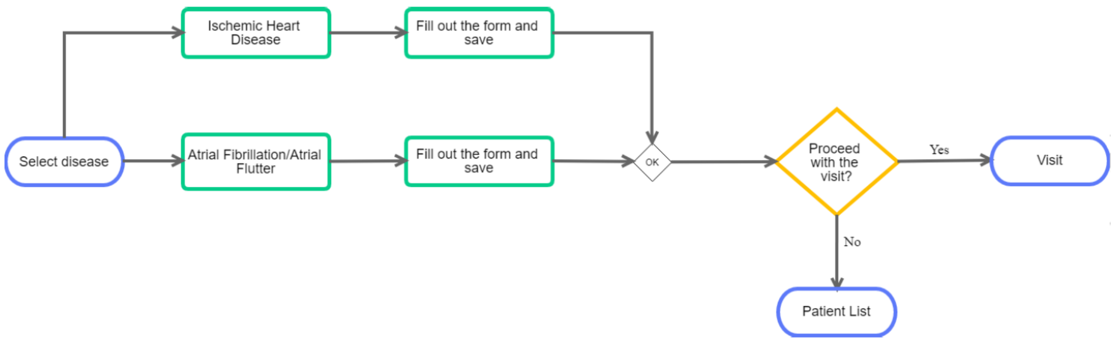


Figure A.5: Select Disease

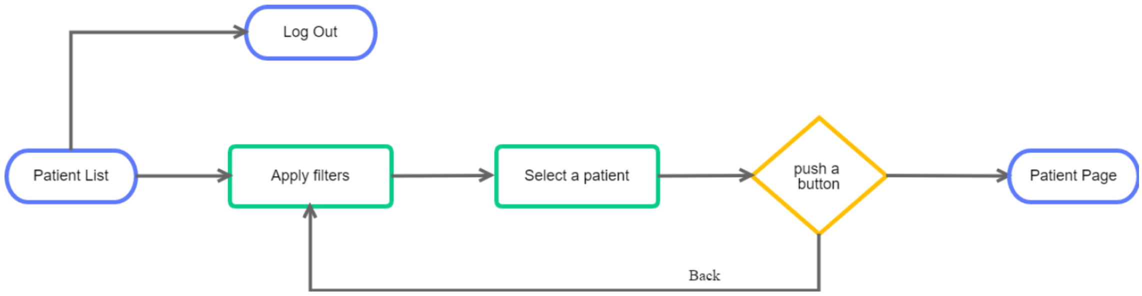


Figure A.6: Patient List Page

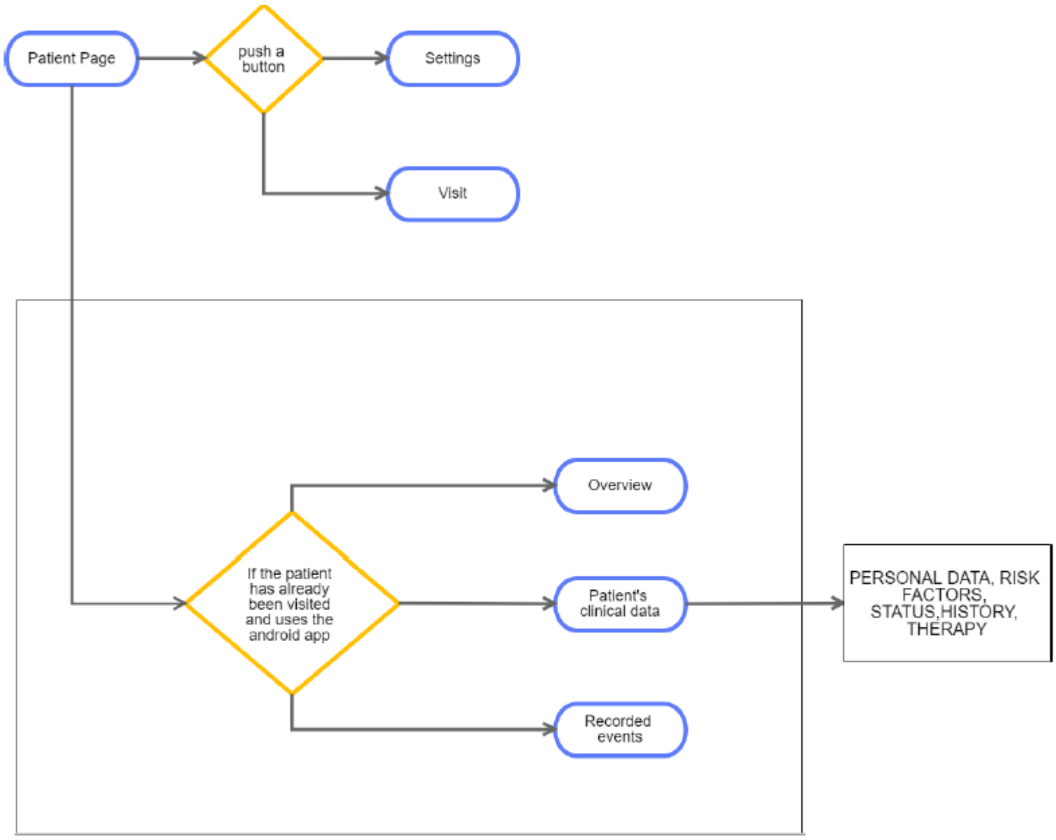


Figure A.7: Personal Patient Page

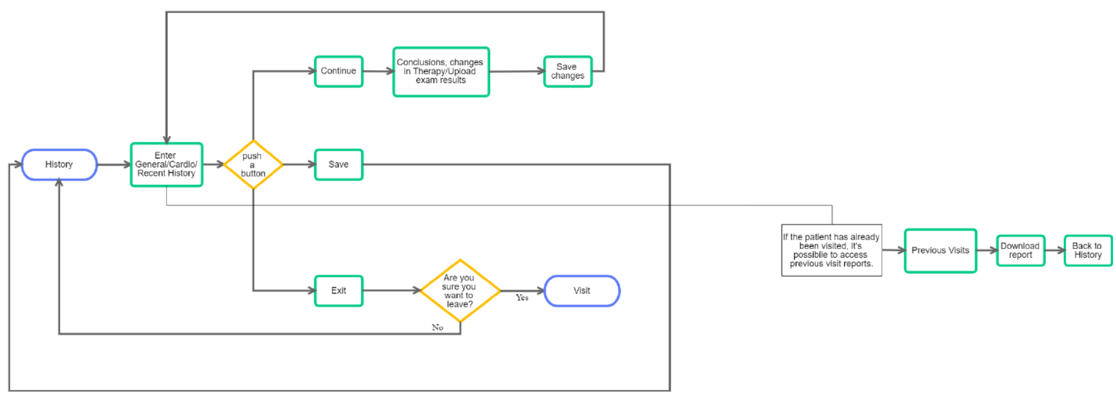


Figure A.8: History

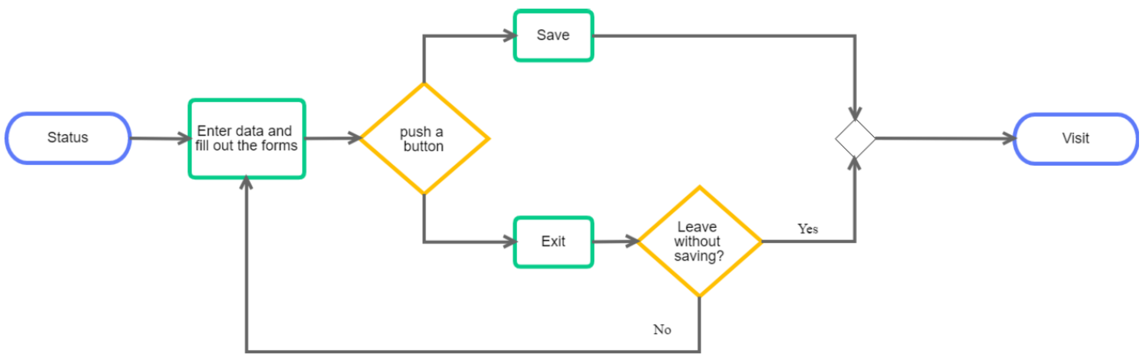


Figure A.9: Status

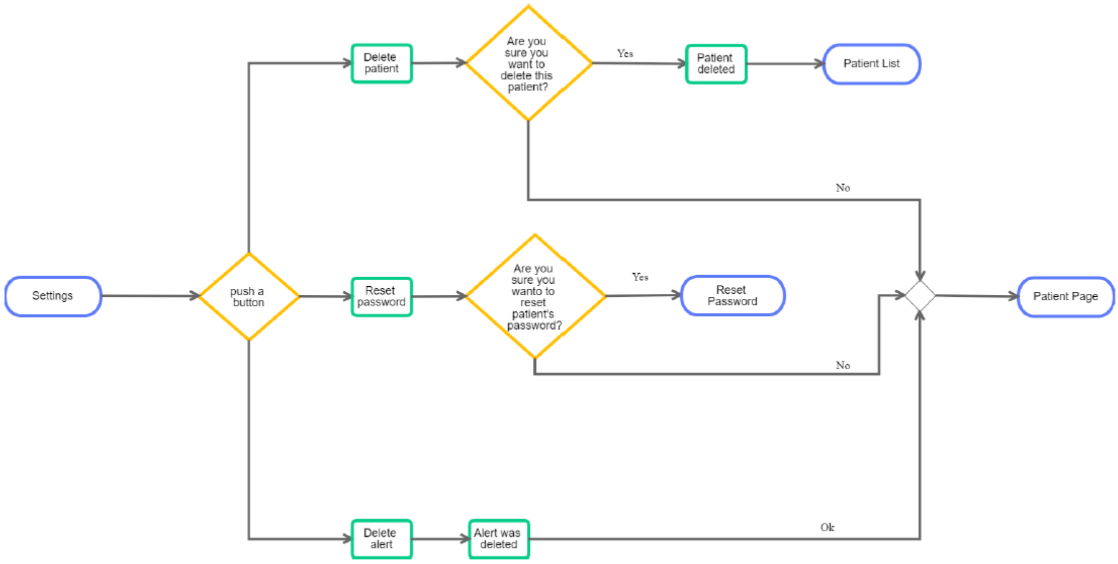


Figure A.10: Settings


Figure A.11: Reset Password

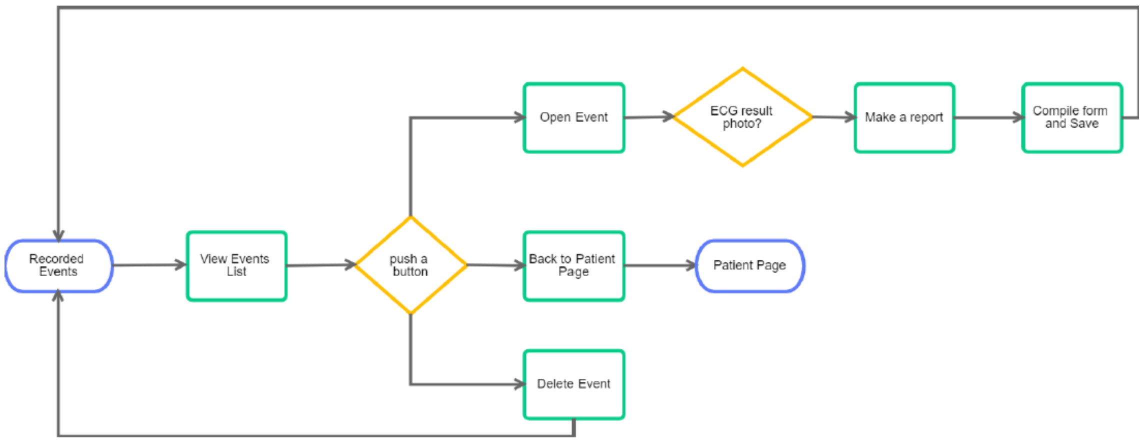


Figure A.12: Recorded Events

# Appendix B

# Appendix B

## B.1 CardioFilo Mockups: Cardiologists

### B.1.1 Web Platform

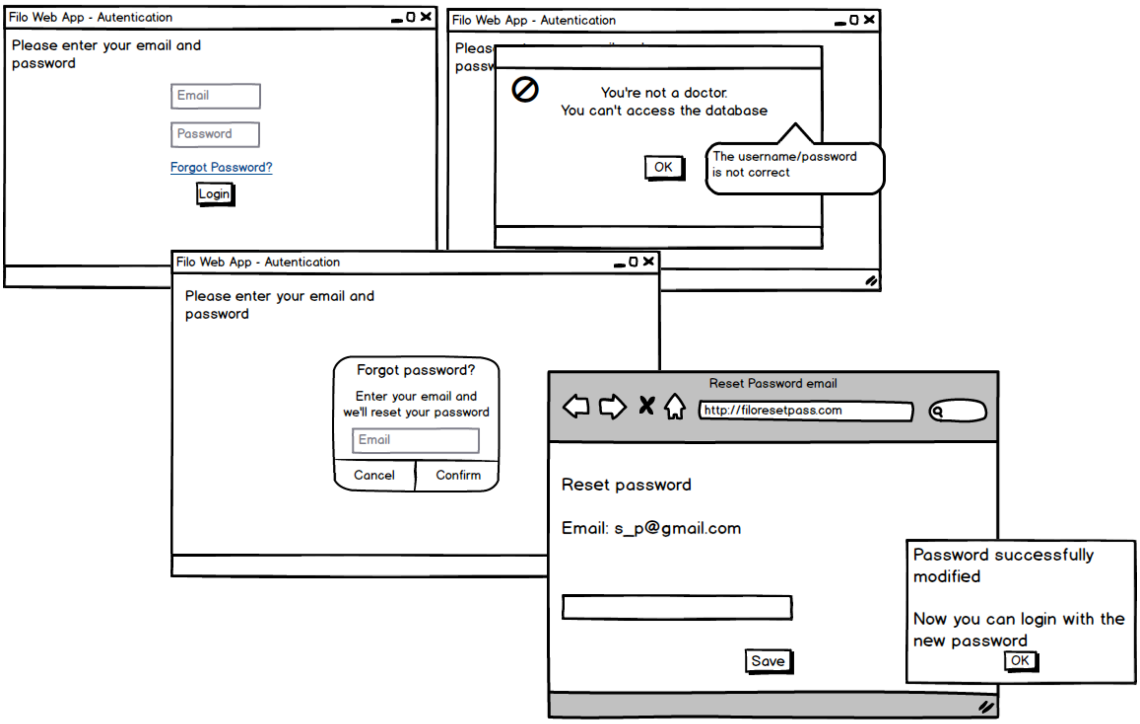


Figure B.1: Login

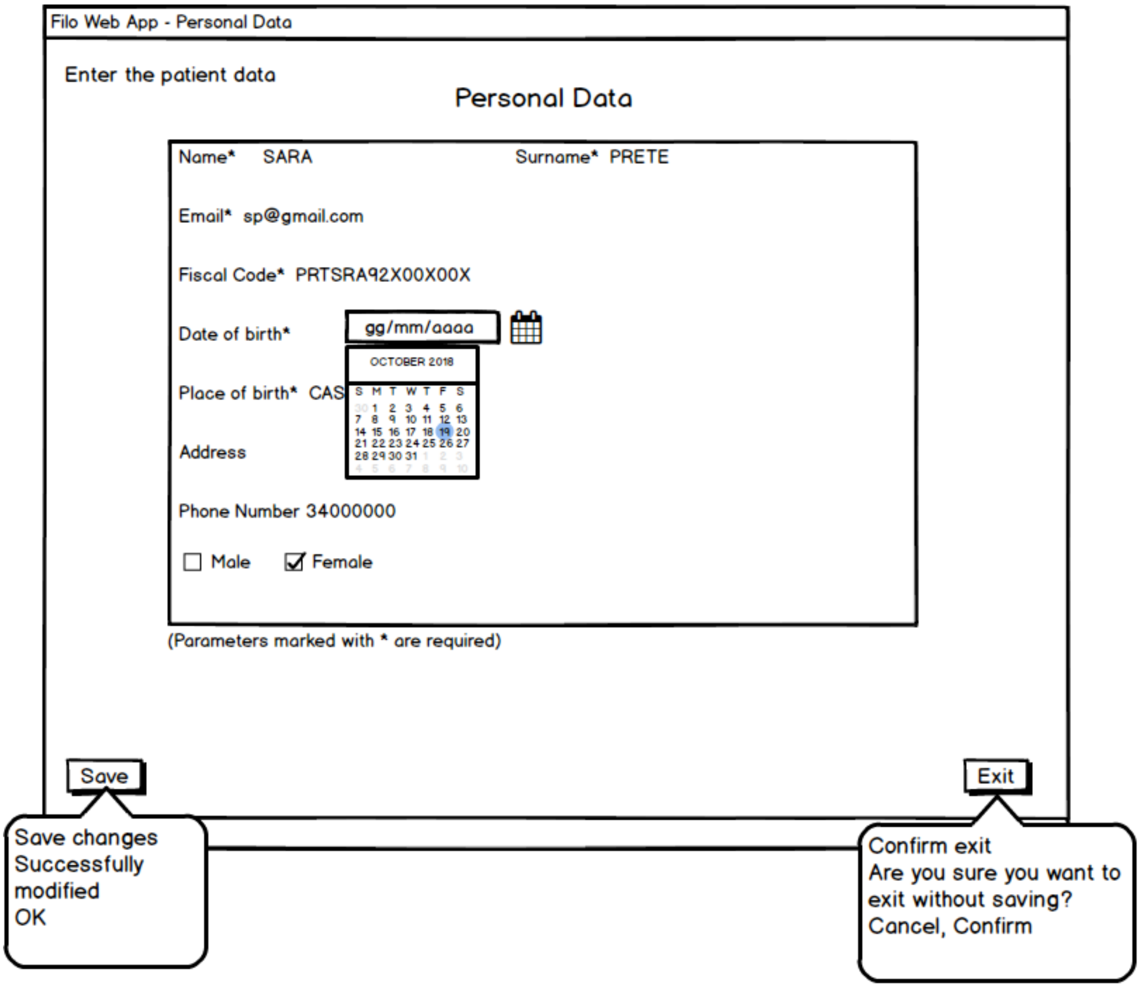


Figure B.2: Personal Data

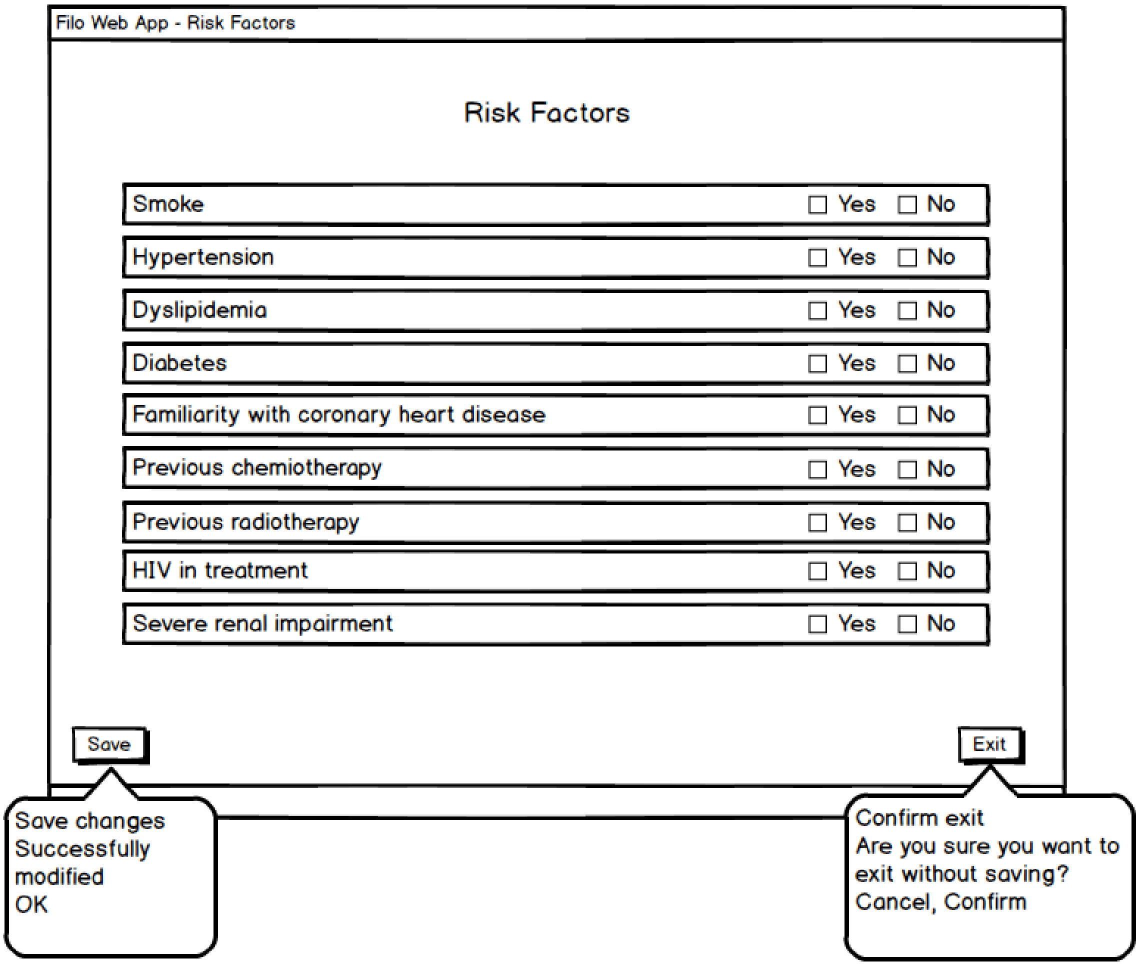


Figure B.3: Risk Factors

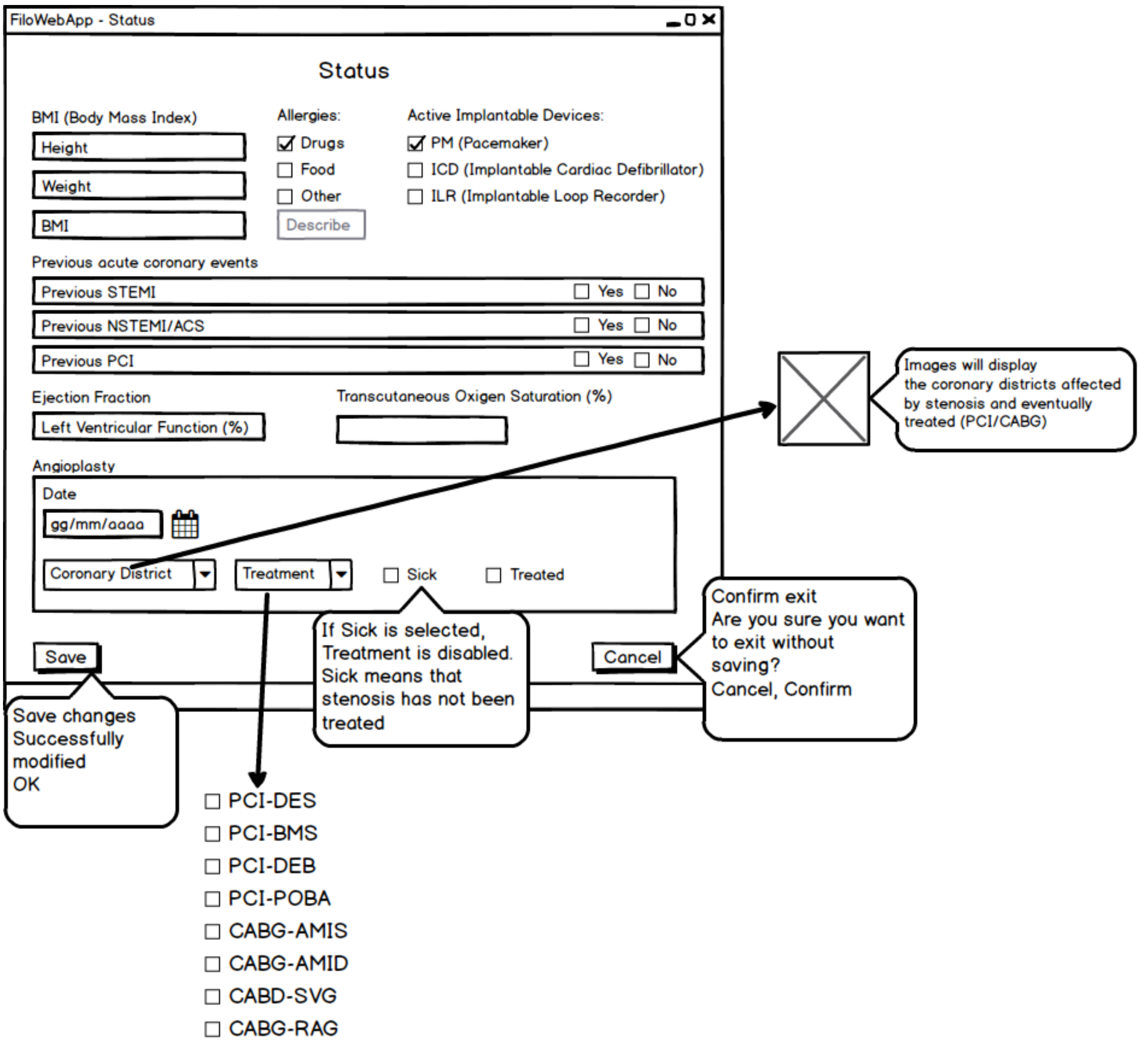


Figure B.4: Status

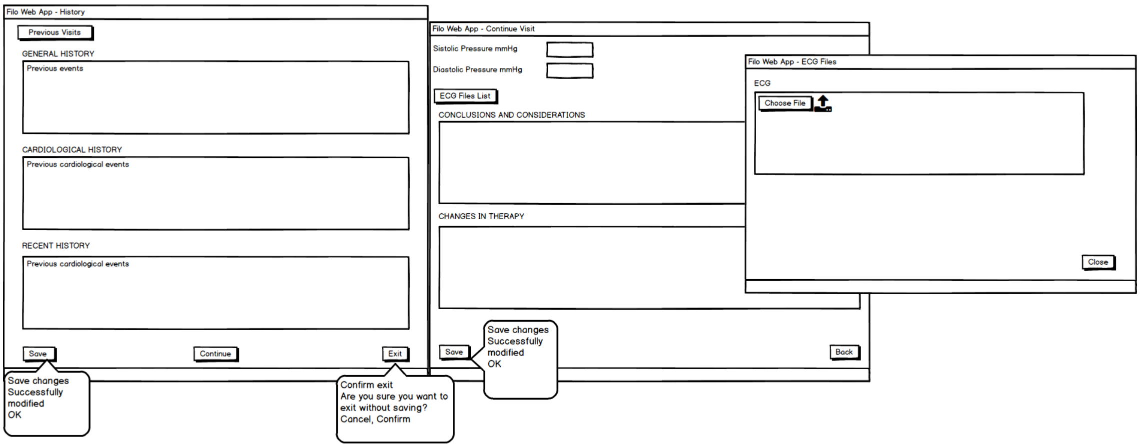


Figure B.5: History

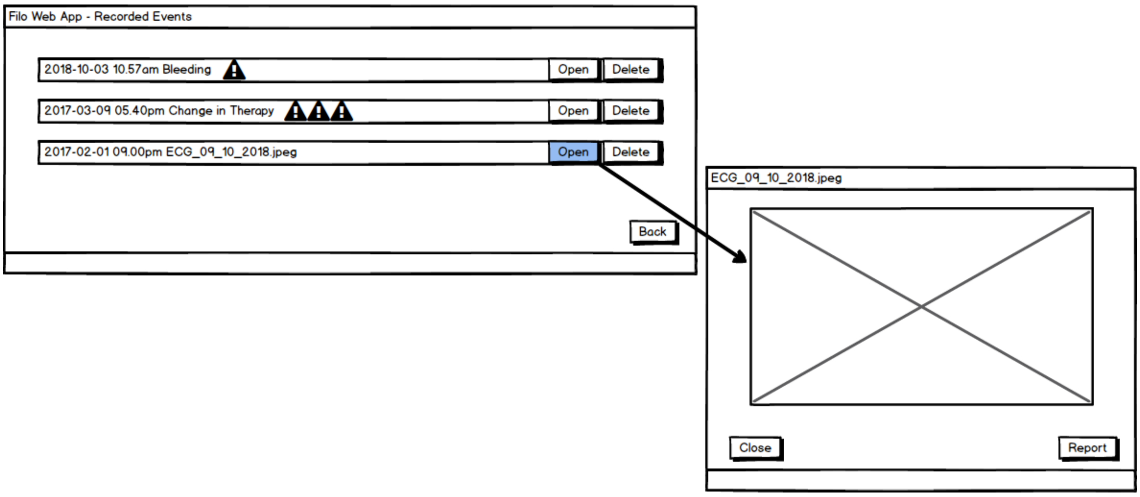


Figure B.6: Recorded events

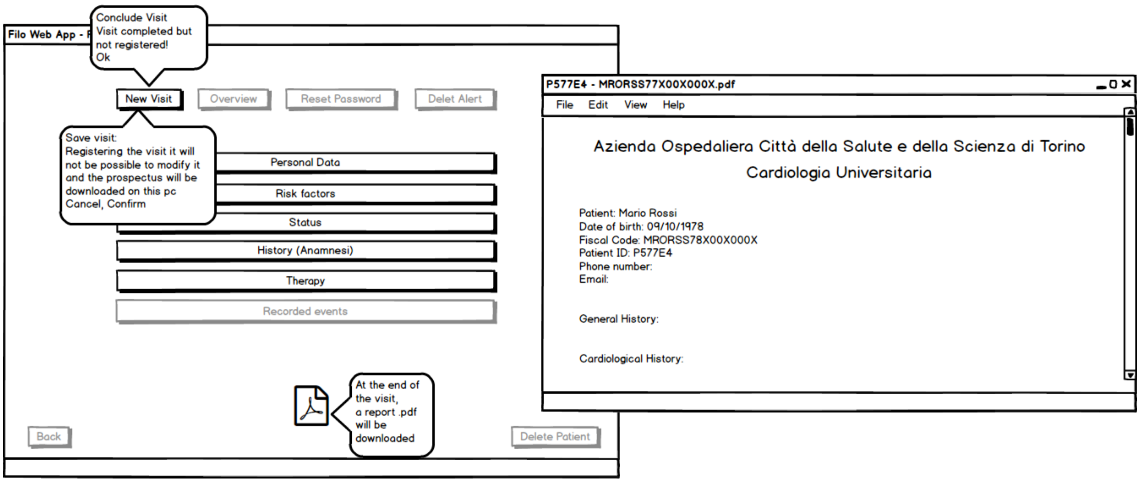


Figure B.7: New Visit and report

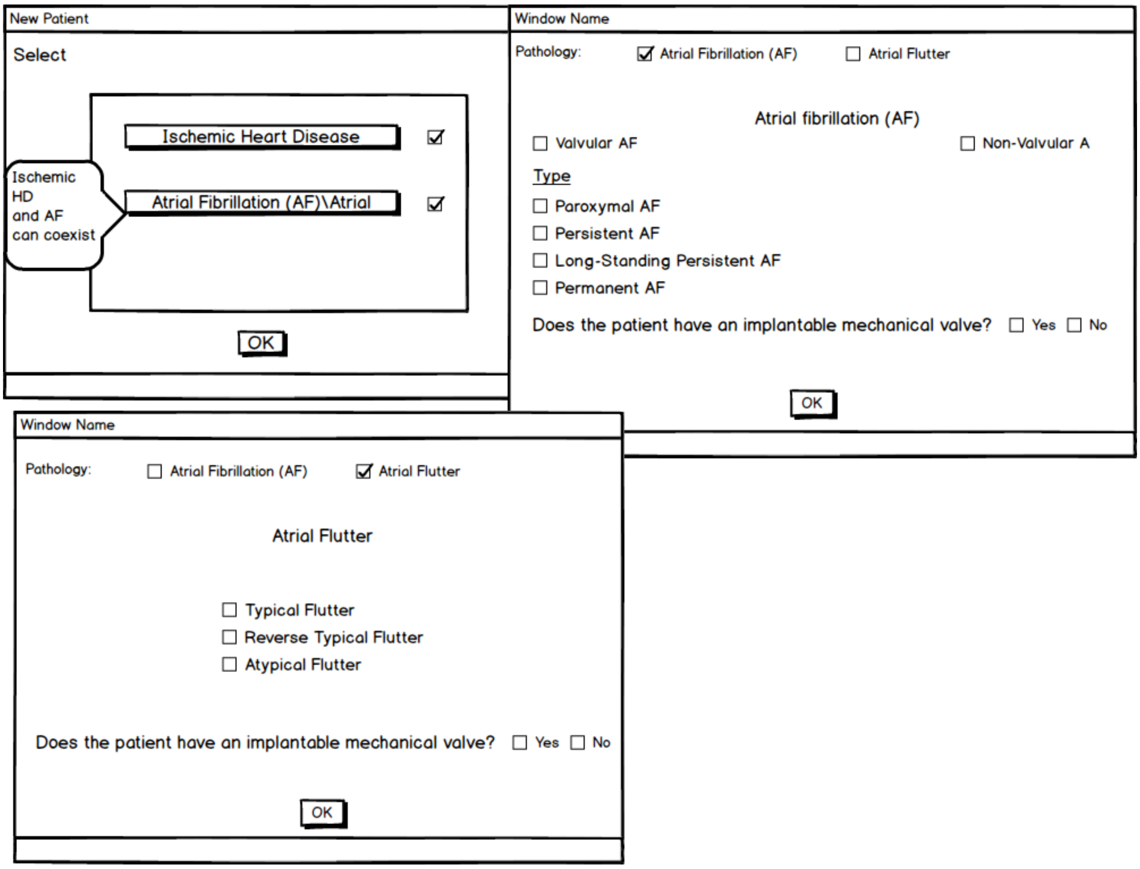


Figure B.8: New Patient Pathology setting

### B.1.2 Mobile Application

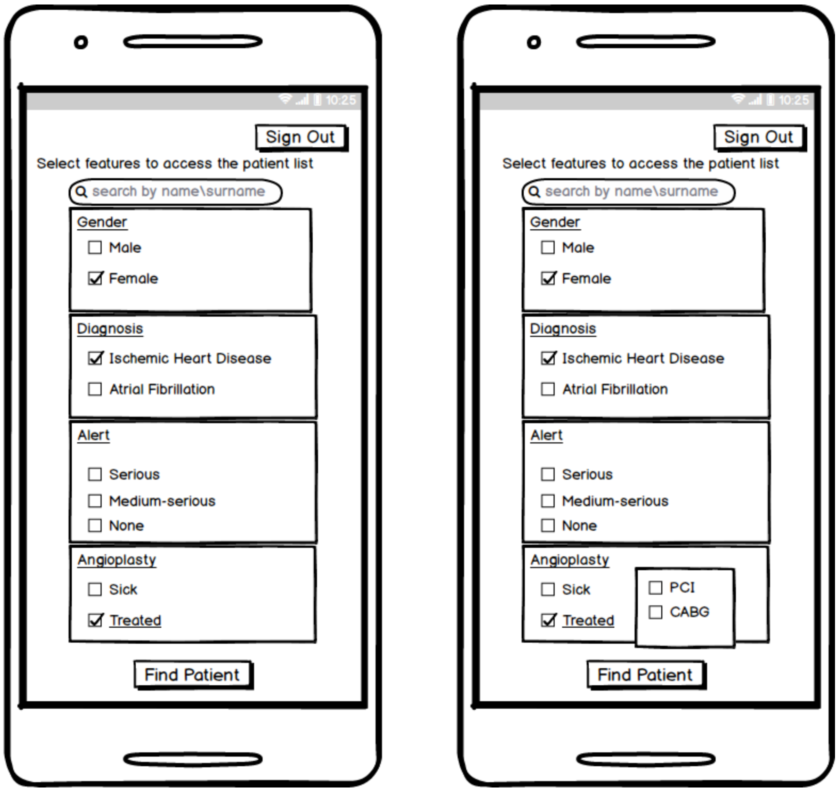


Figure B.9: Select features

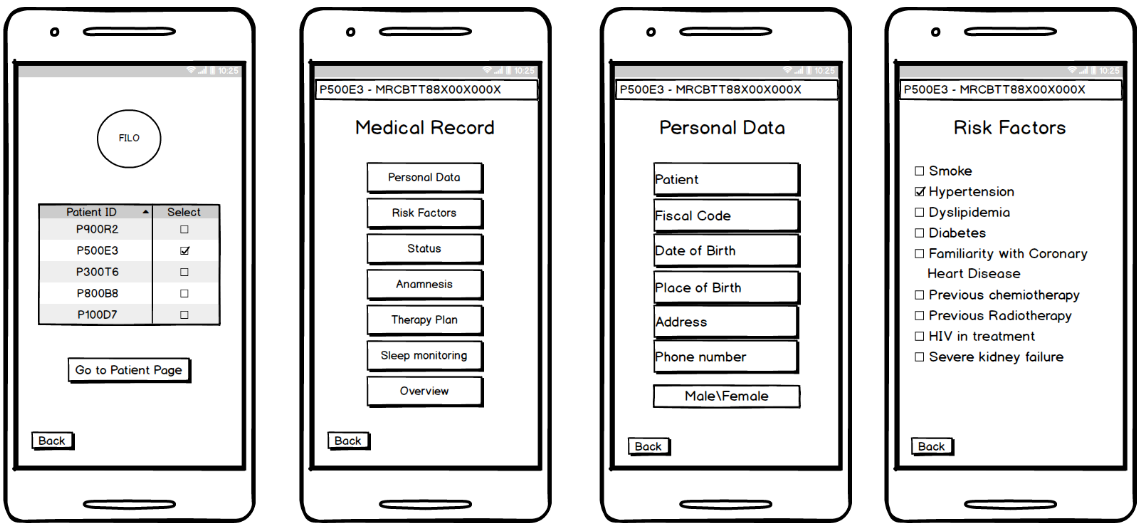


Figure B.10: Patient Data (1)

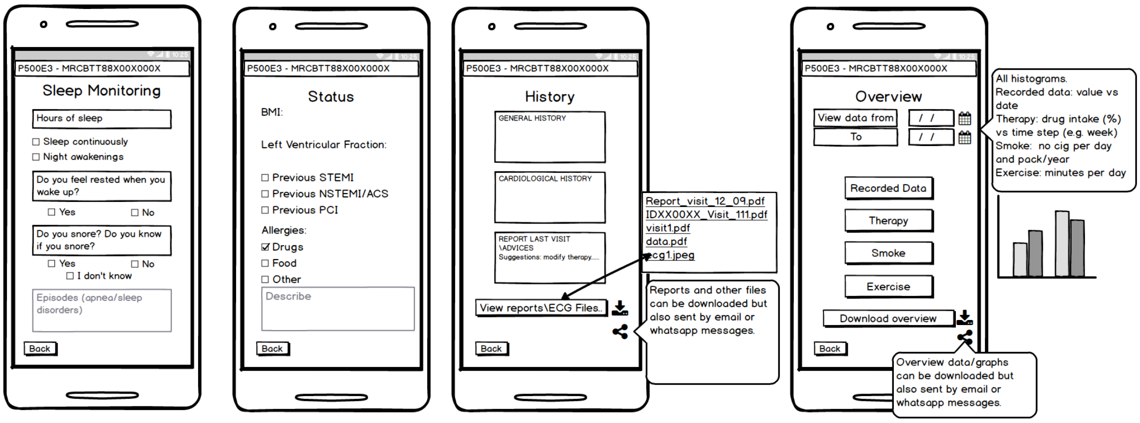


Figure B.11: Patient Data (2)

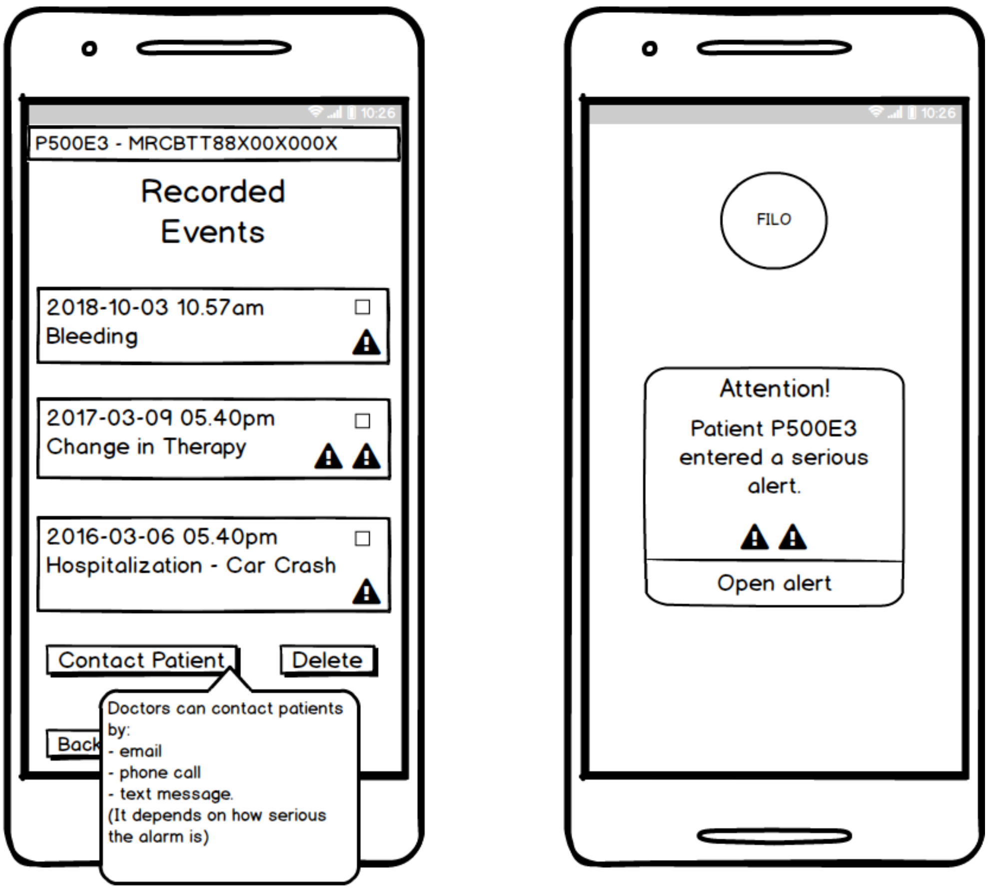


Figure B.12: Patient Data (3)

## B.2 CardioFilo Mockups: Patients

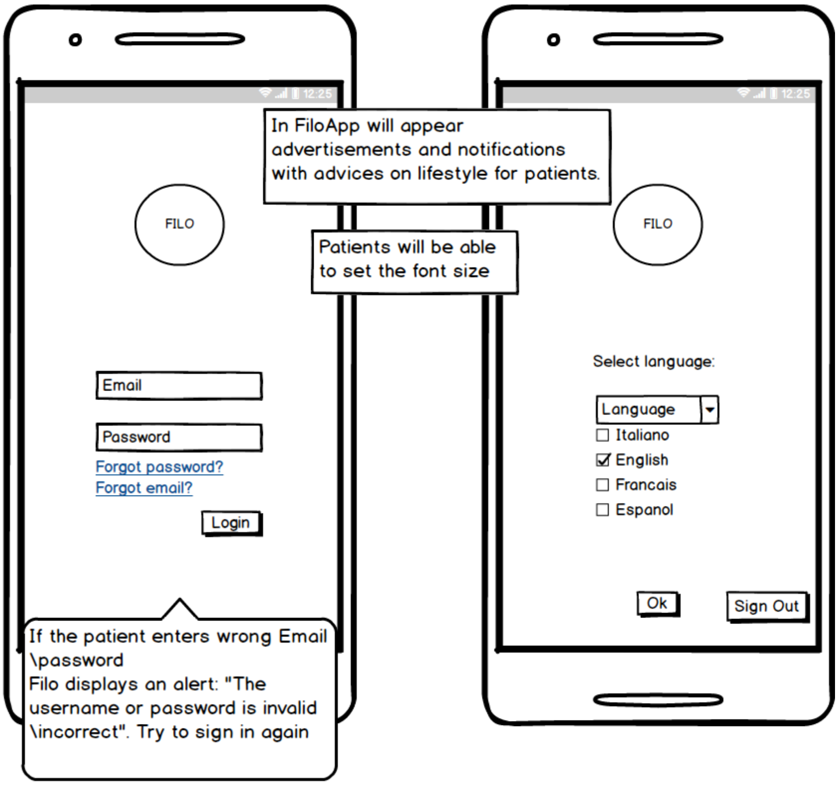


Figure B.13: Login and language selection

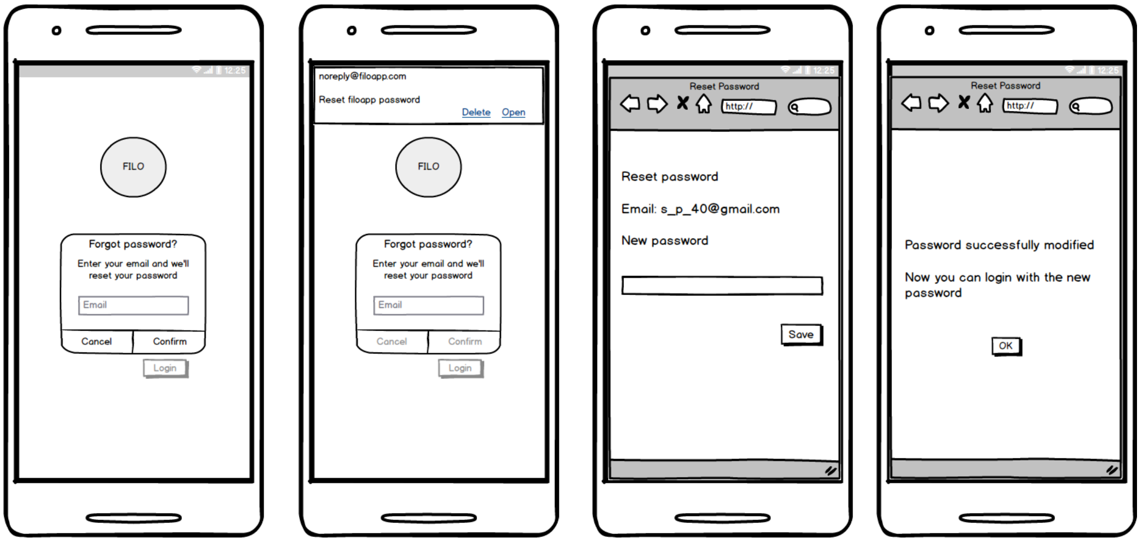


Figure B.14: Reset Password

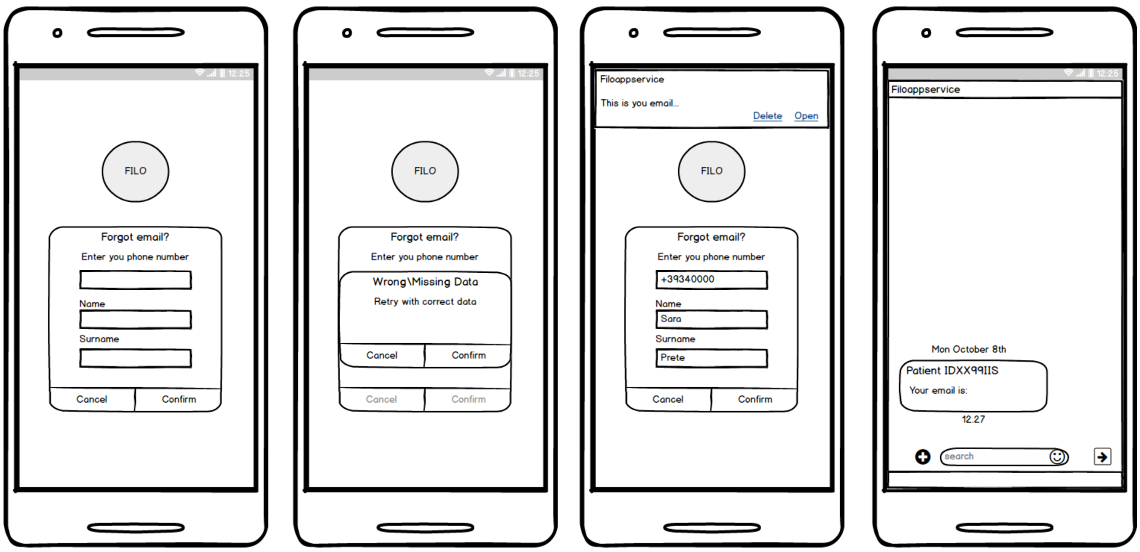


Figure B.15: Reset Email

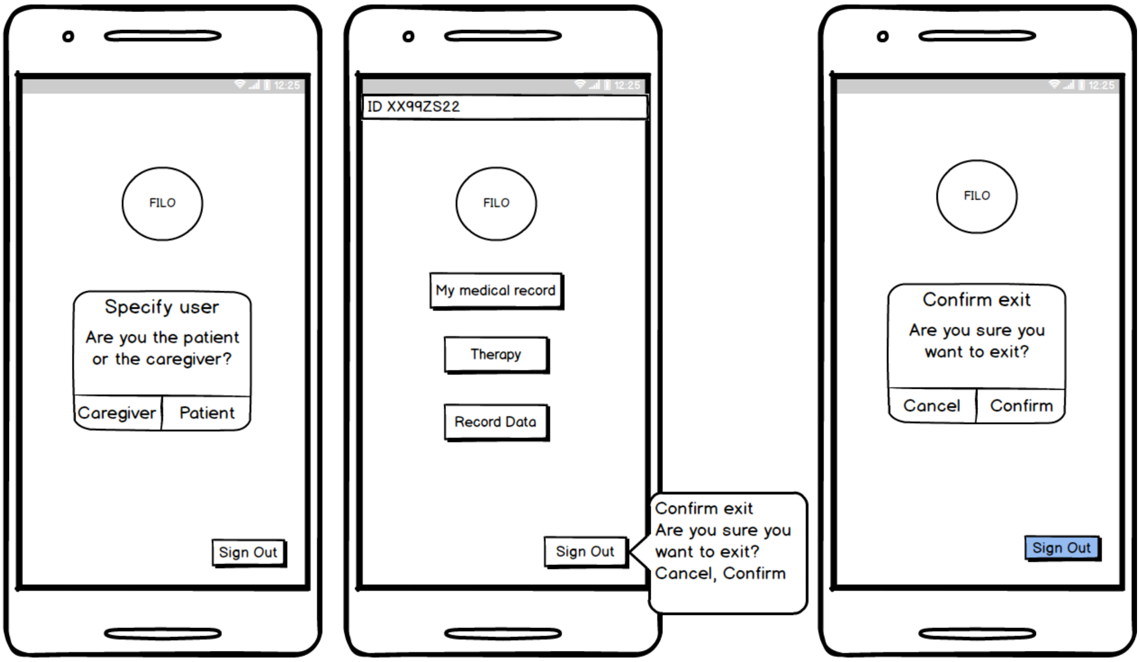


Figure B.16: HomePage

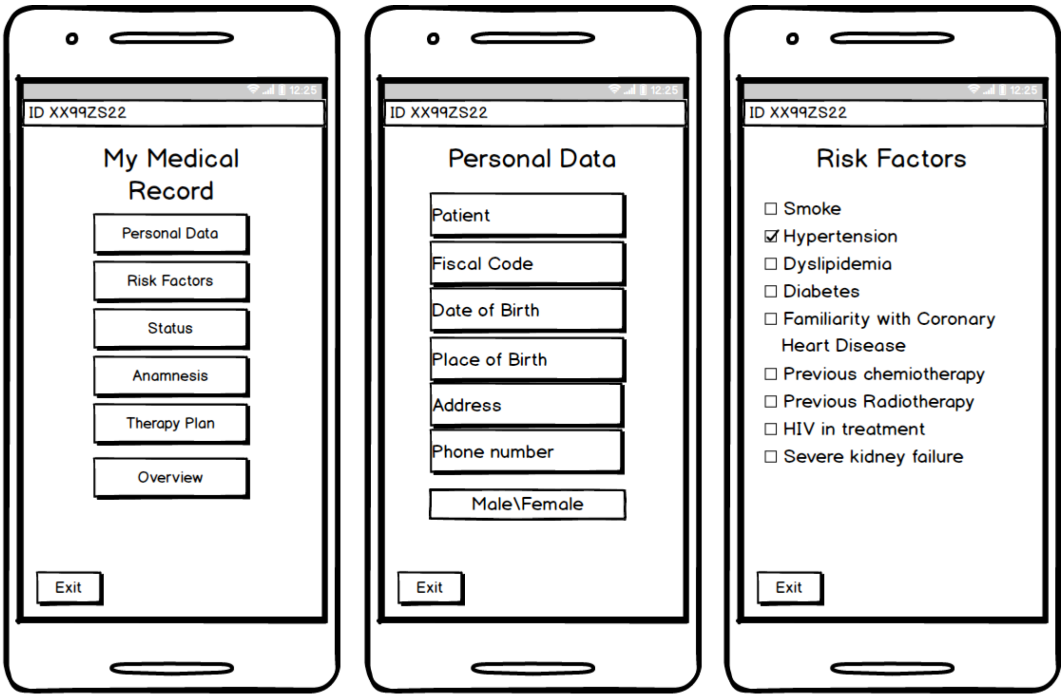


Figure B.17: Medical Record: Personal Data and Risk Factors

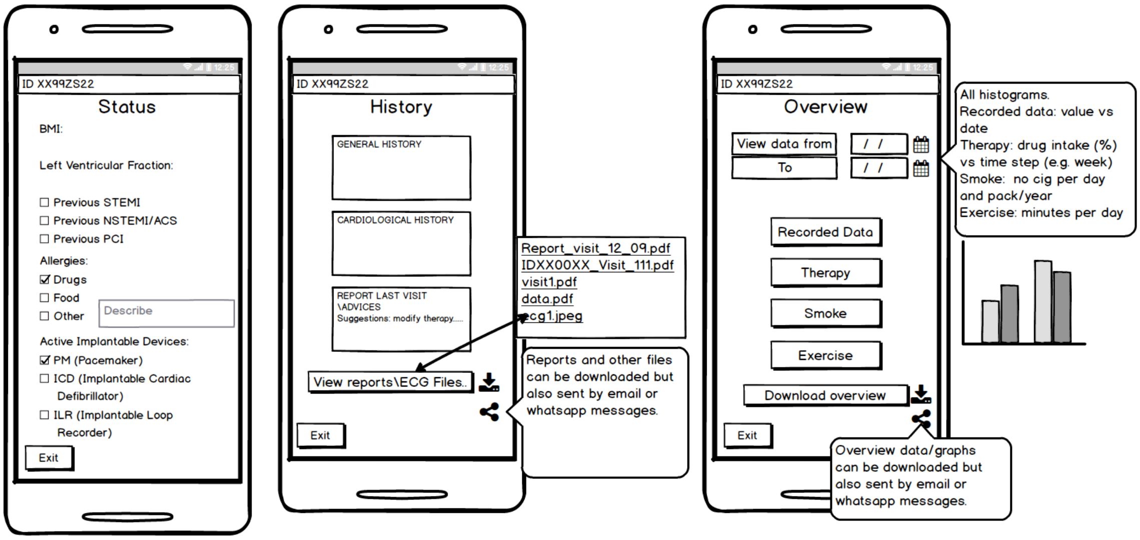


Figure B.18: Medical Record: Status, History and Overview

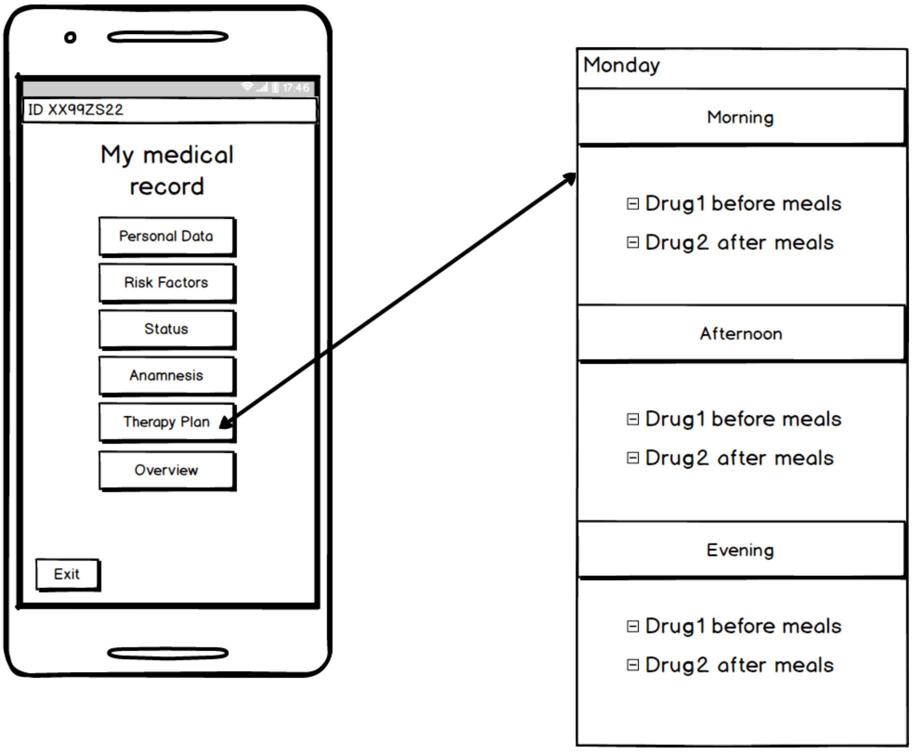


Figure B.19: Medical Record: Therapy Plan

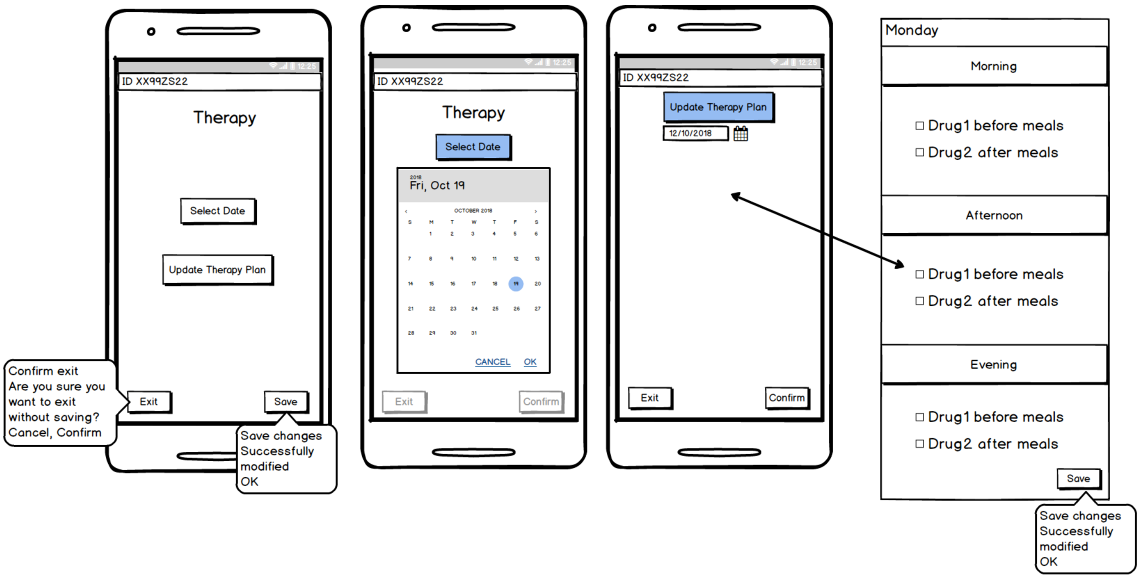


Figure B.20: Update Therapy Plan Page

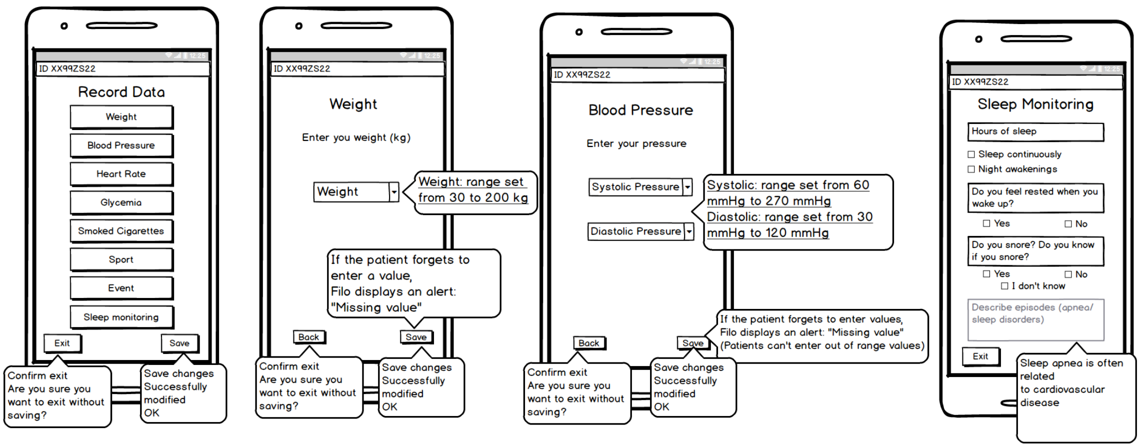


Figure B.21: Record Data Page: Weight, Blood Pressure and Sleep Monitoring

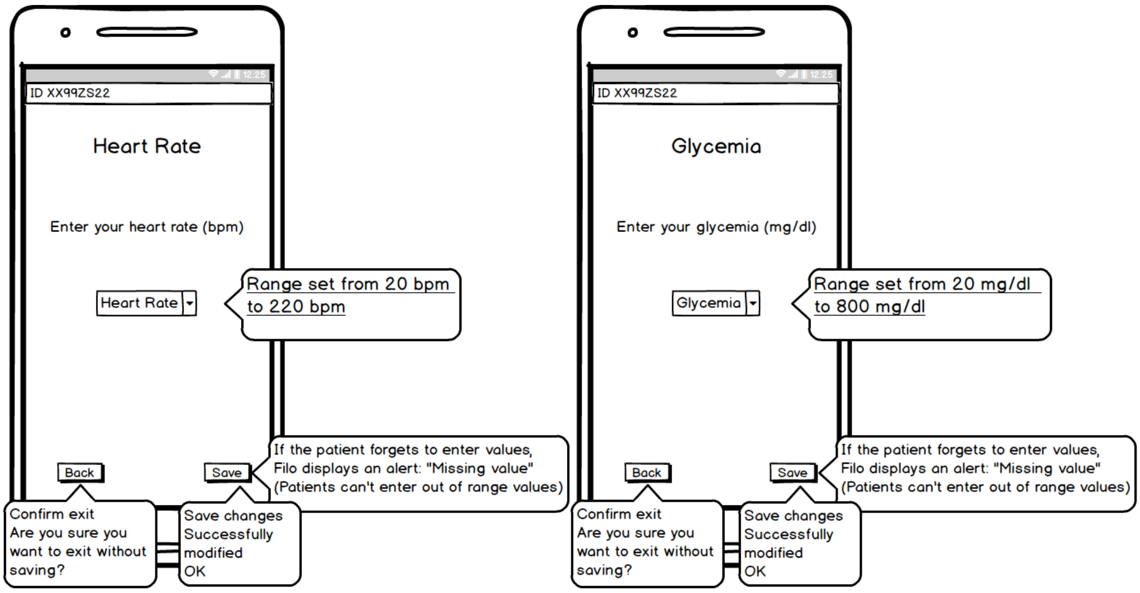


Figure B.22: Record Data Page: Heart Rate and Glycemia

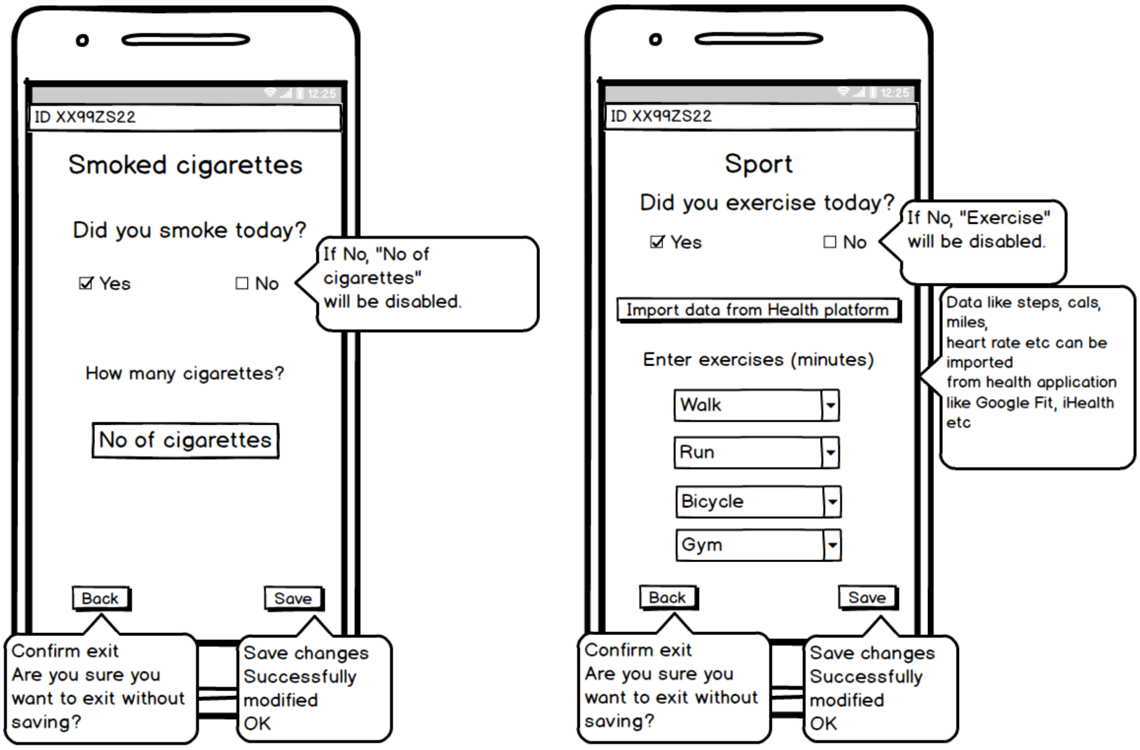


Figure B.23: Record Data Page: Smoked cigarettes and Sport

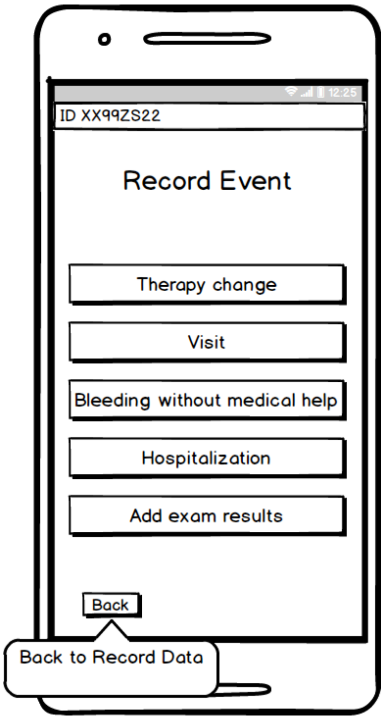


Figure B.24: Record Data Page: Record Event

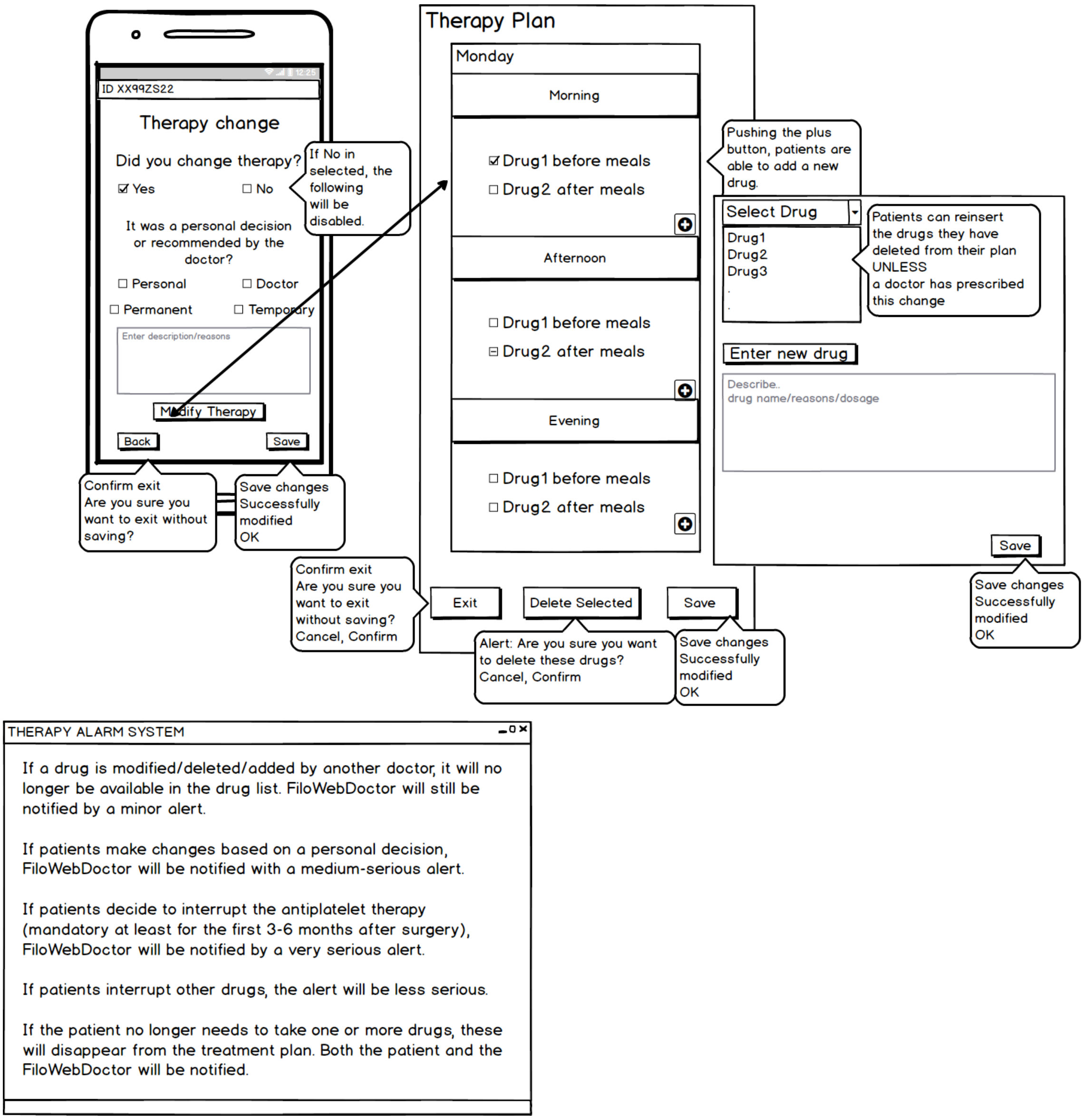


Figure B.25: Record Data Page: Therapy changes

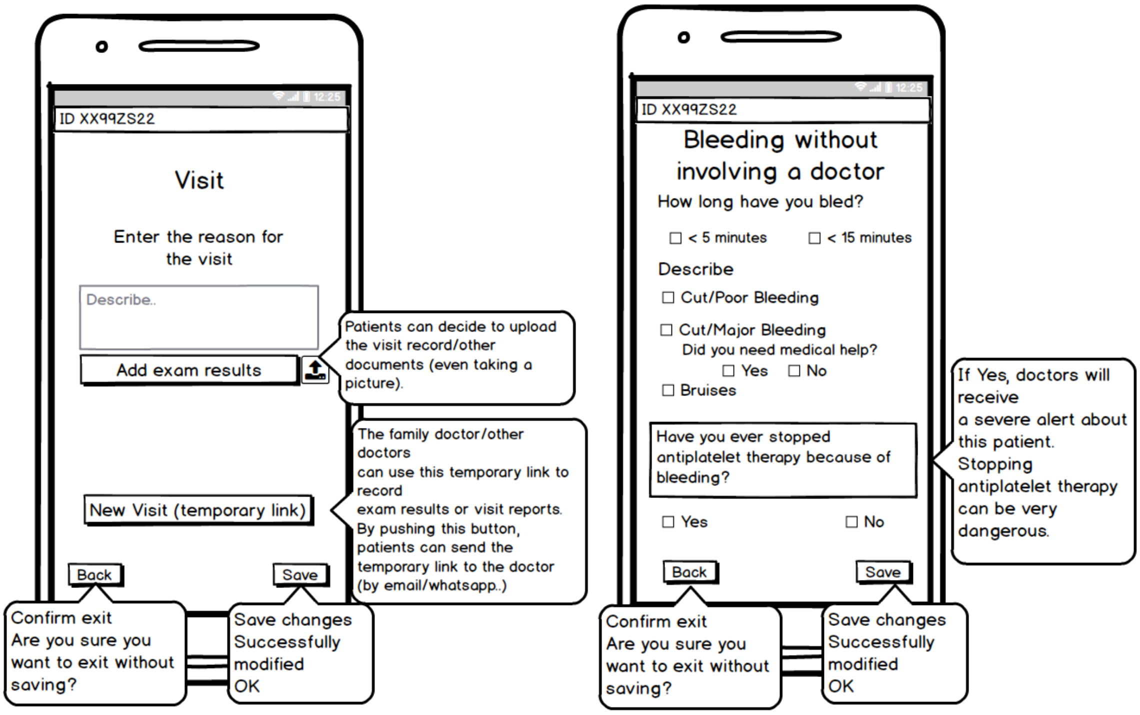


Figure B.26: Record Data Page: Other visits and Bleeding Events

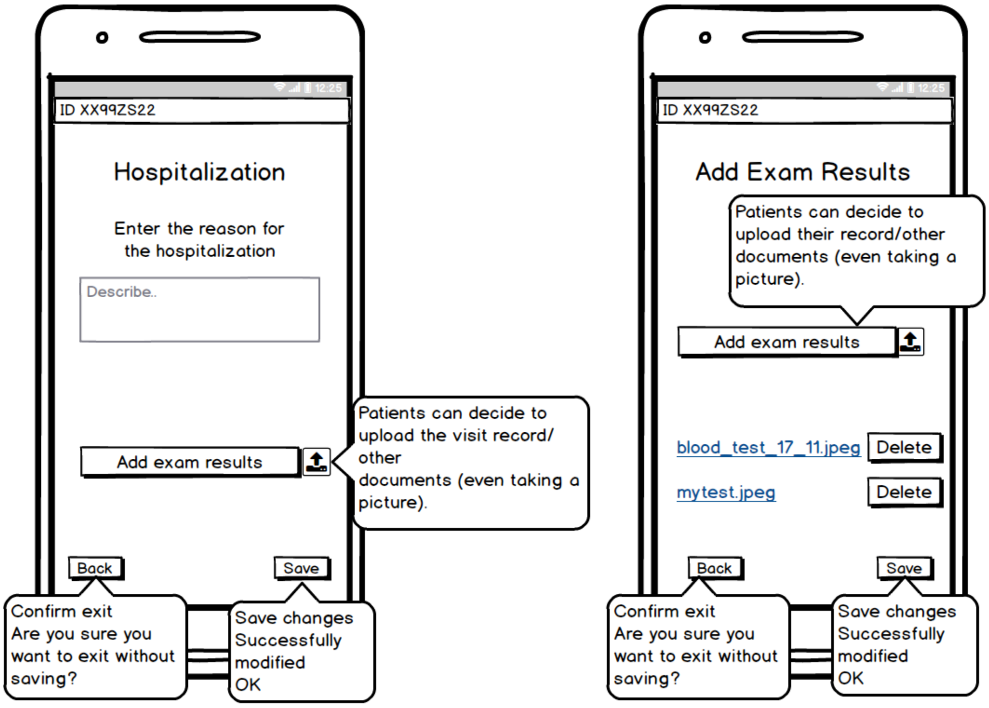


Figure B.27: Record Data Page: Hospitalization and Upload of Exam Results

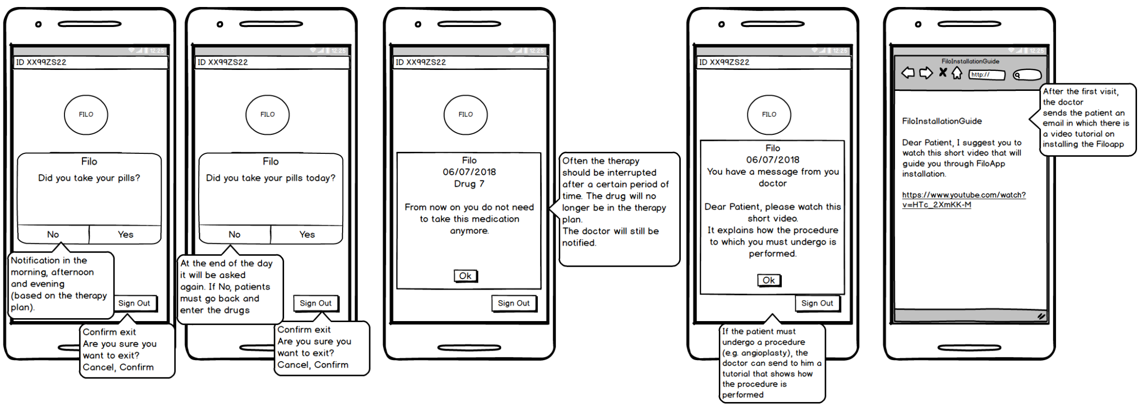


Figure B.28: Notification models

## B.3 CardioFilo Mockups: Nurses

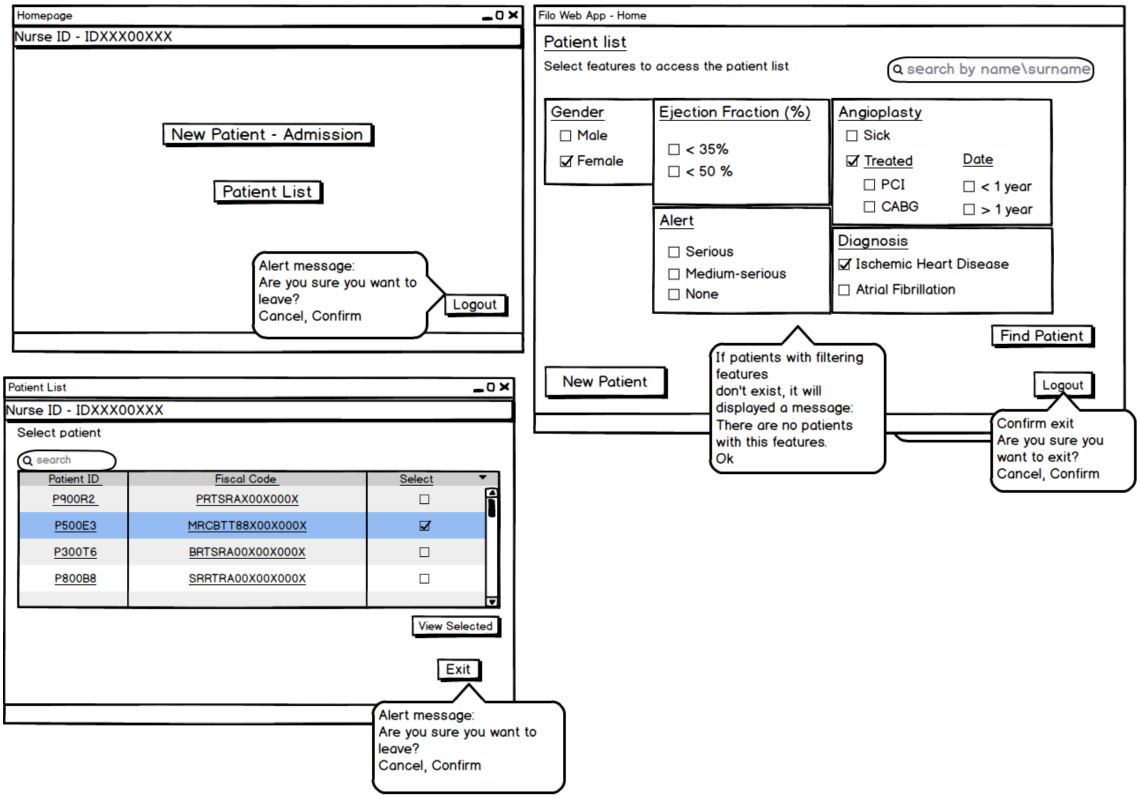


Figure B.29: Home Page and Patient List

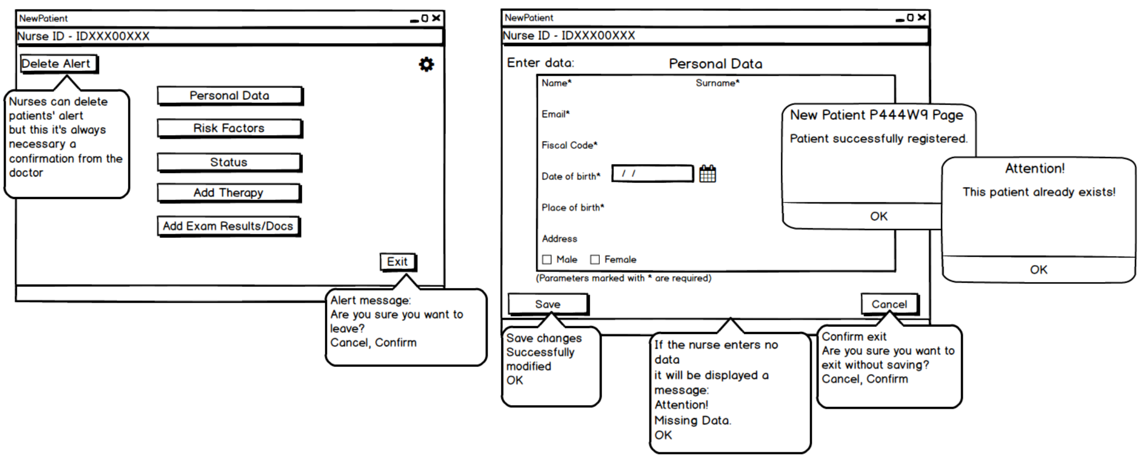


Figure B.30: Patient Page: Personal Data

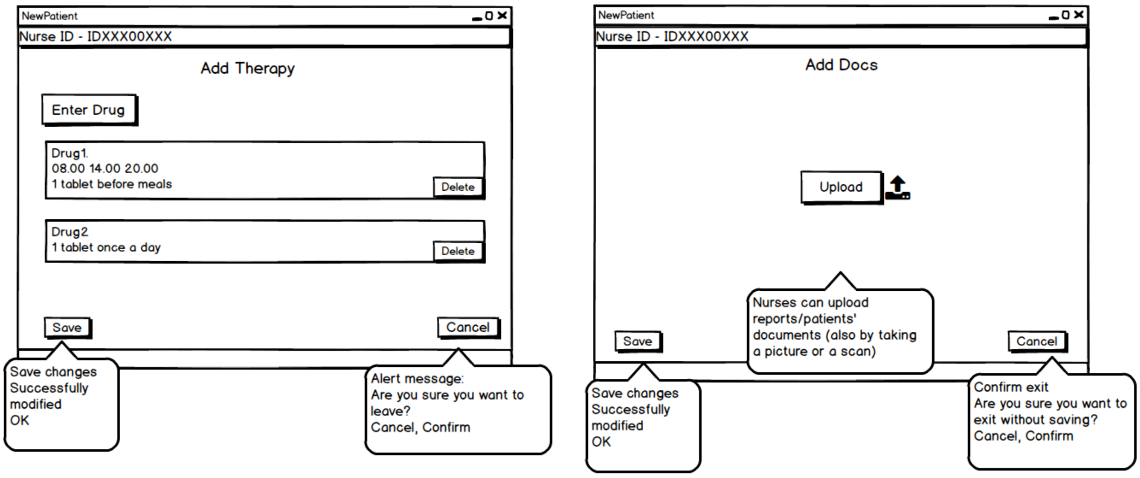


Figure B.31: Patient Page: add drugs and upload exam results

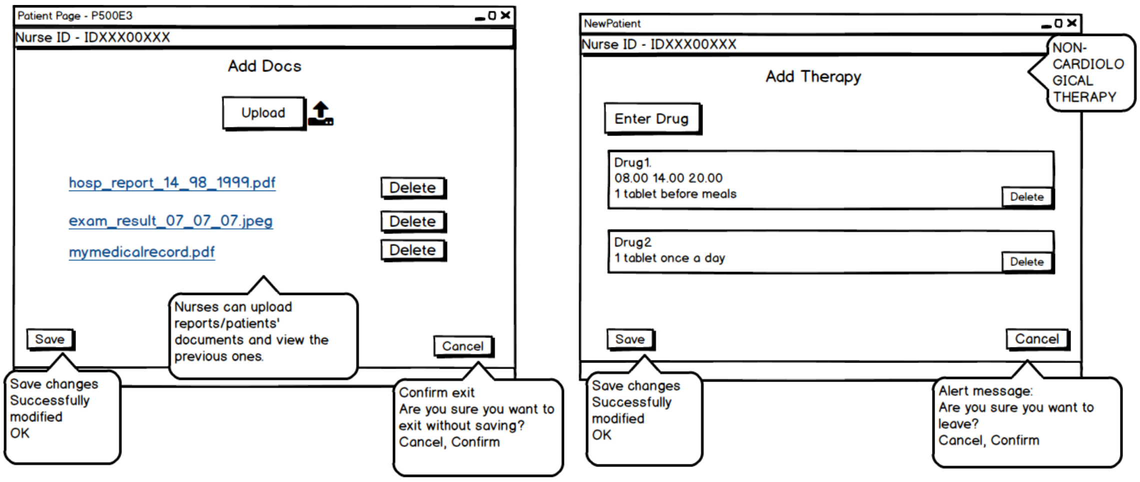


Figure B.32: Consulting Patient Page (4)

## B.4 CardioFilo Mockups: Scientific Societies

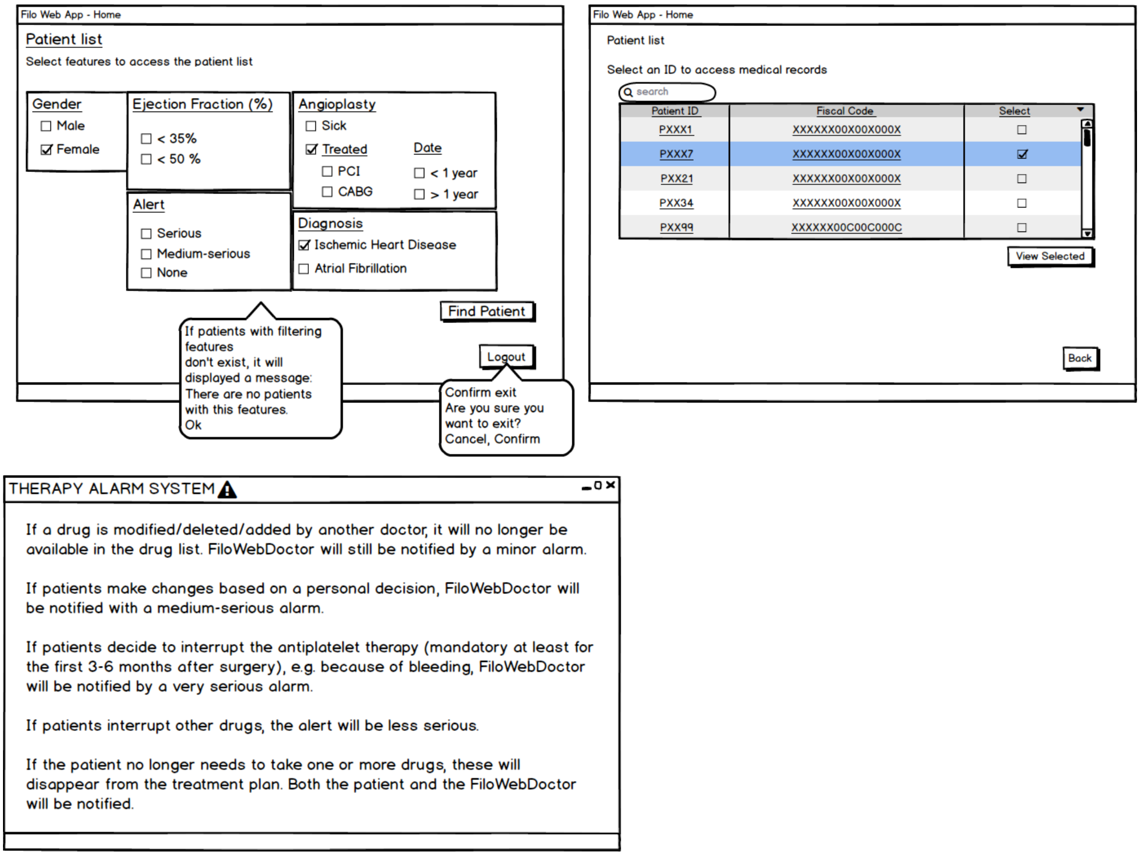


Figure B.33: Patient List

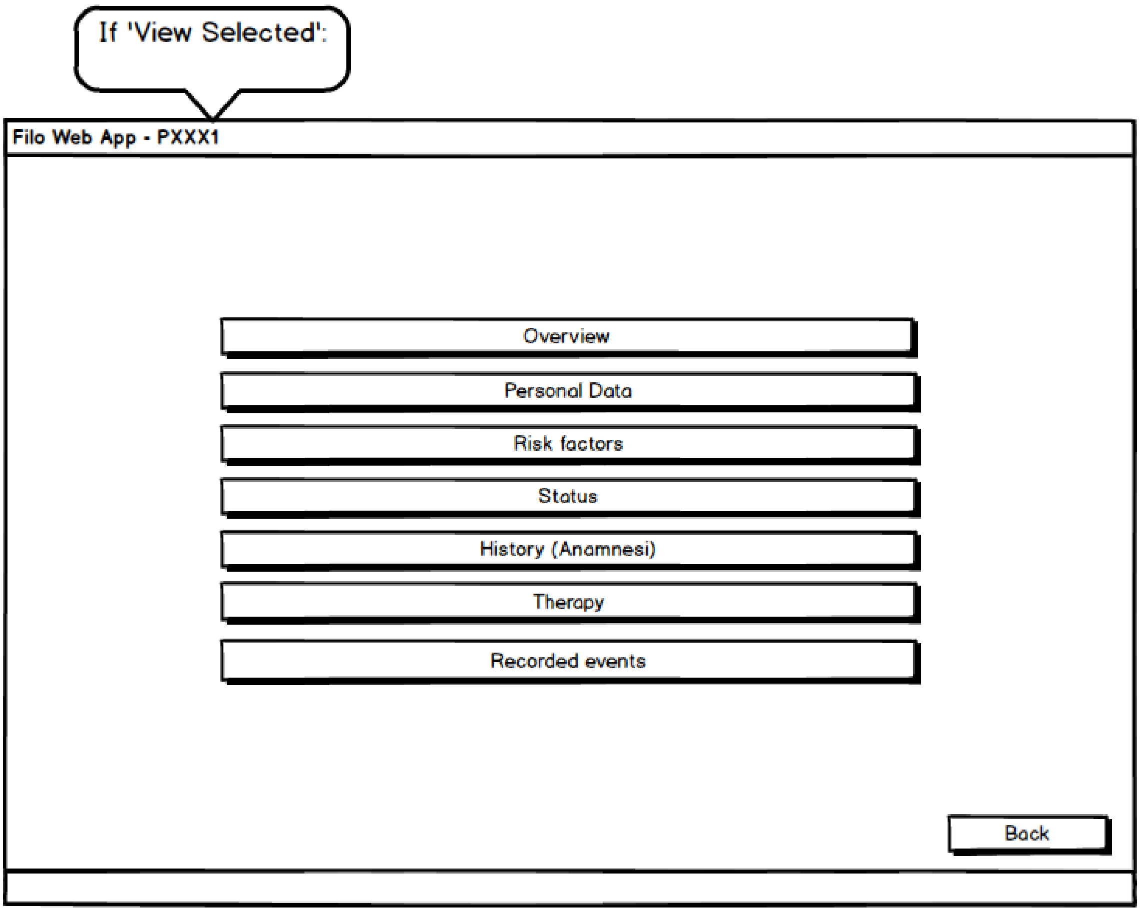


Figure B.34: Patient Page

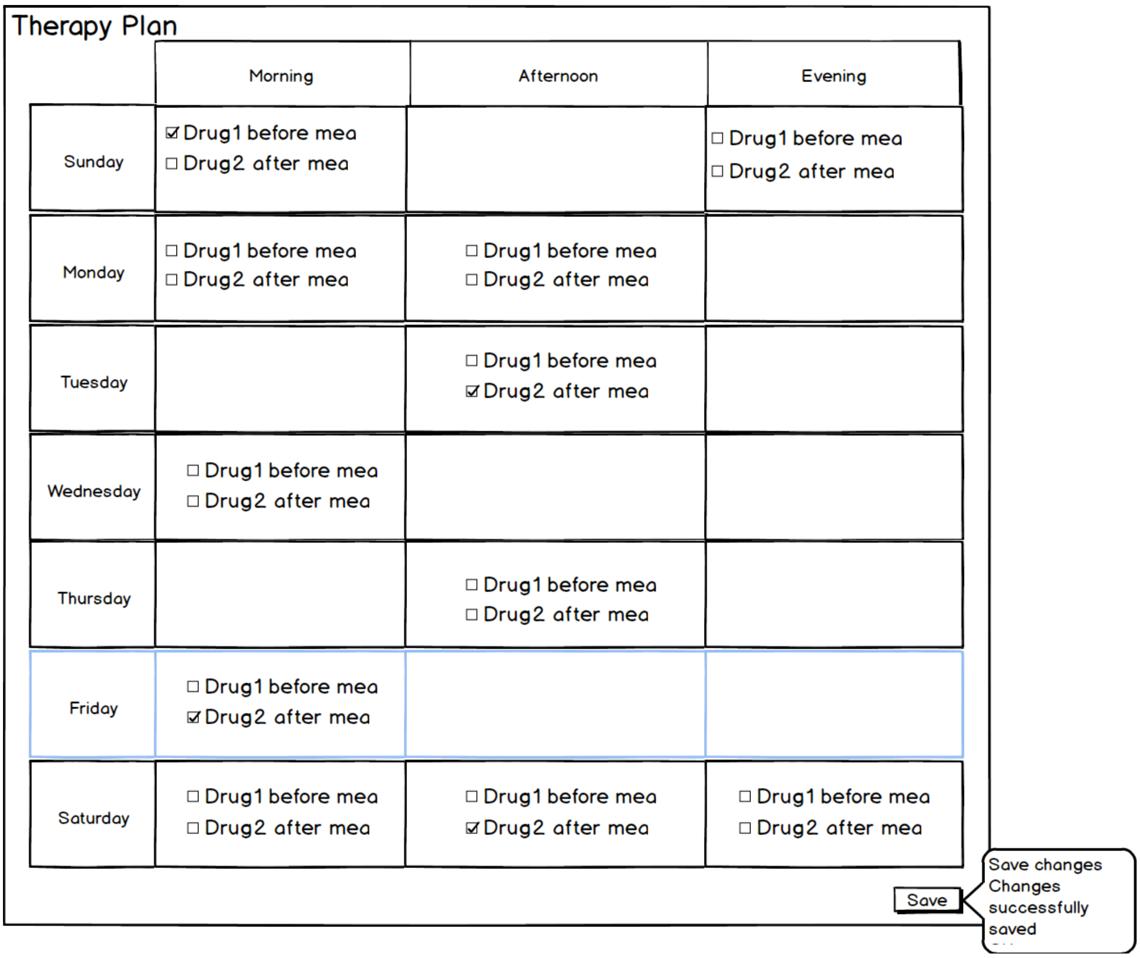


Figure B.35: Therapy Plan

# Appendix C

# Appendix C

## C.1 CardioFilo Graphics: Web Application

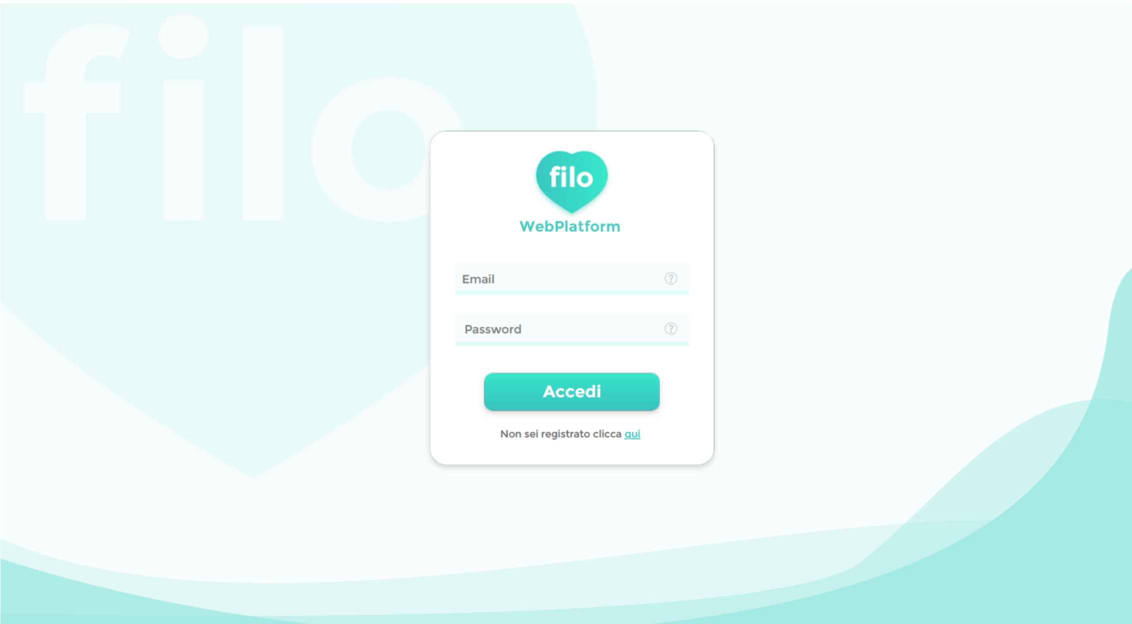


Figure C.1: Login

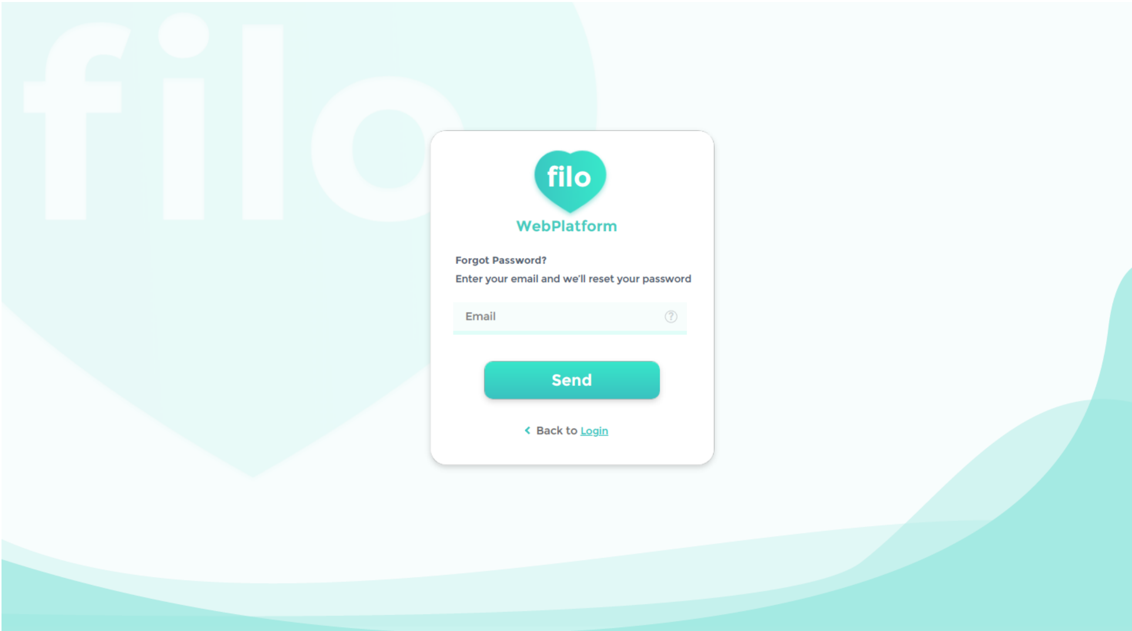


Figure C.2: Reset Password

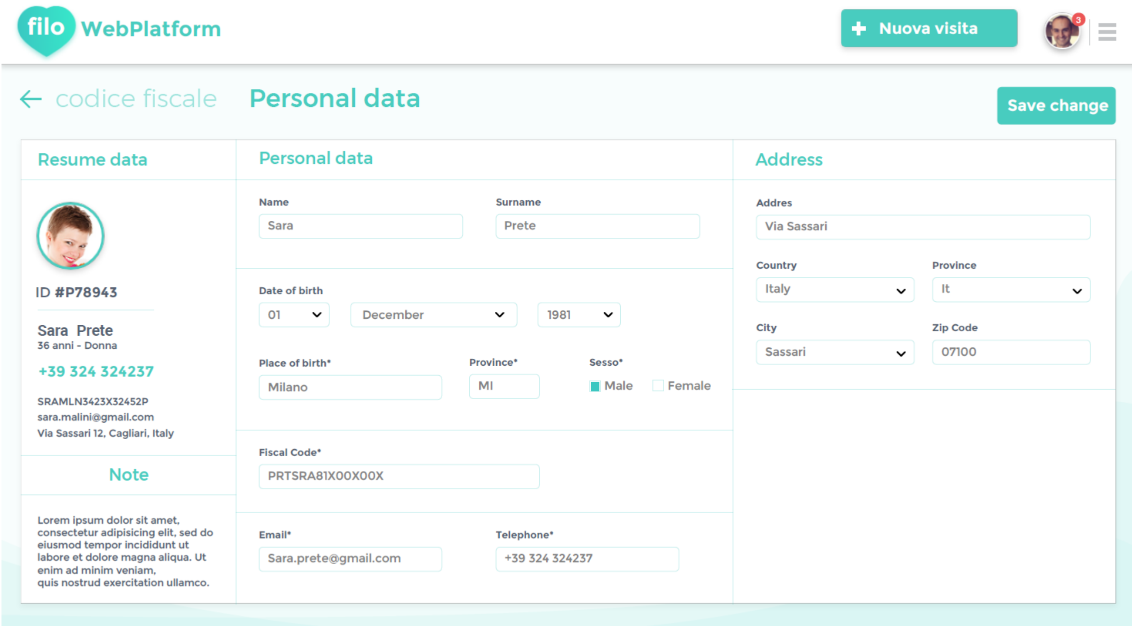


Figure C.3: Patient Page: Personal Data

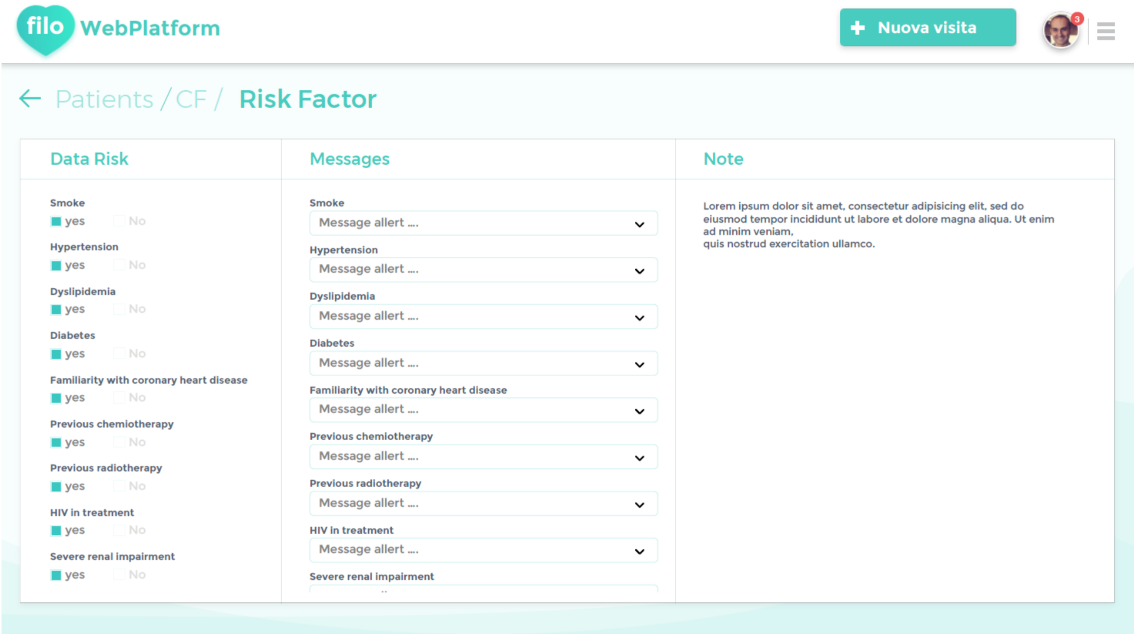


Figure C.4: Patient Page: Risk Factors

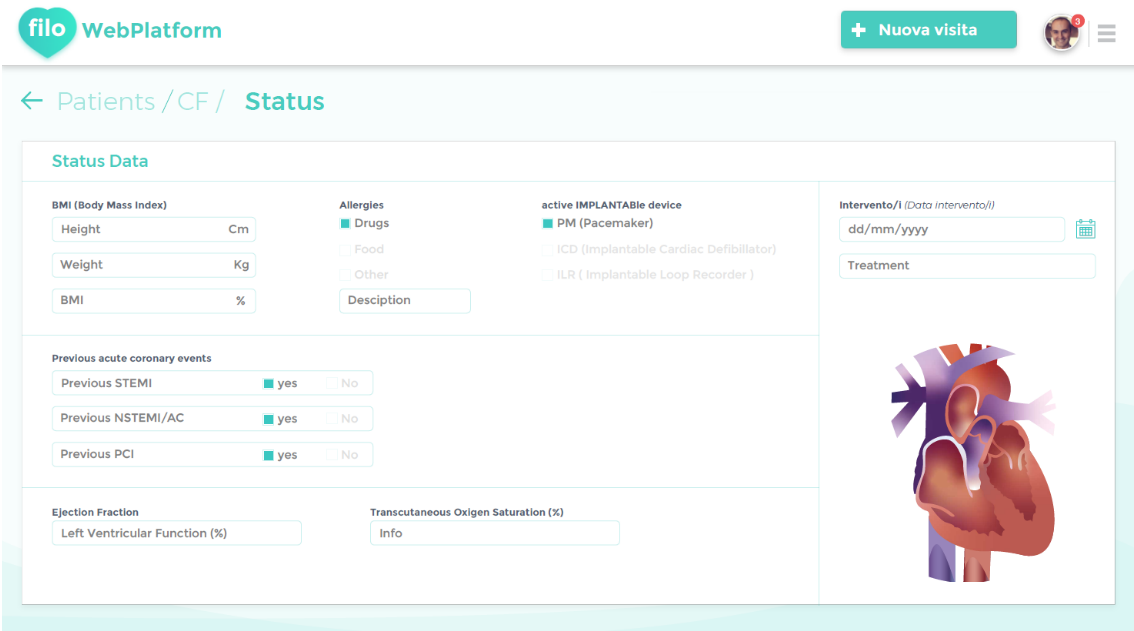


Figure C.5: Patient Page: Status

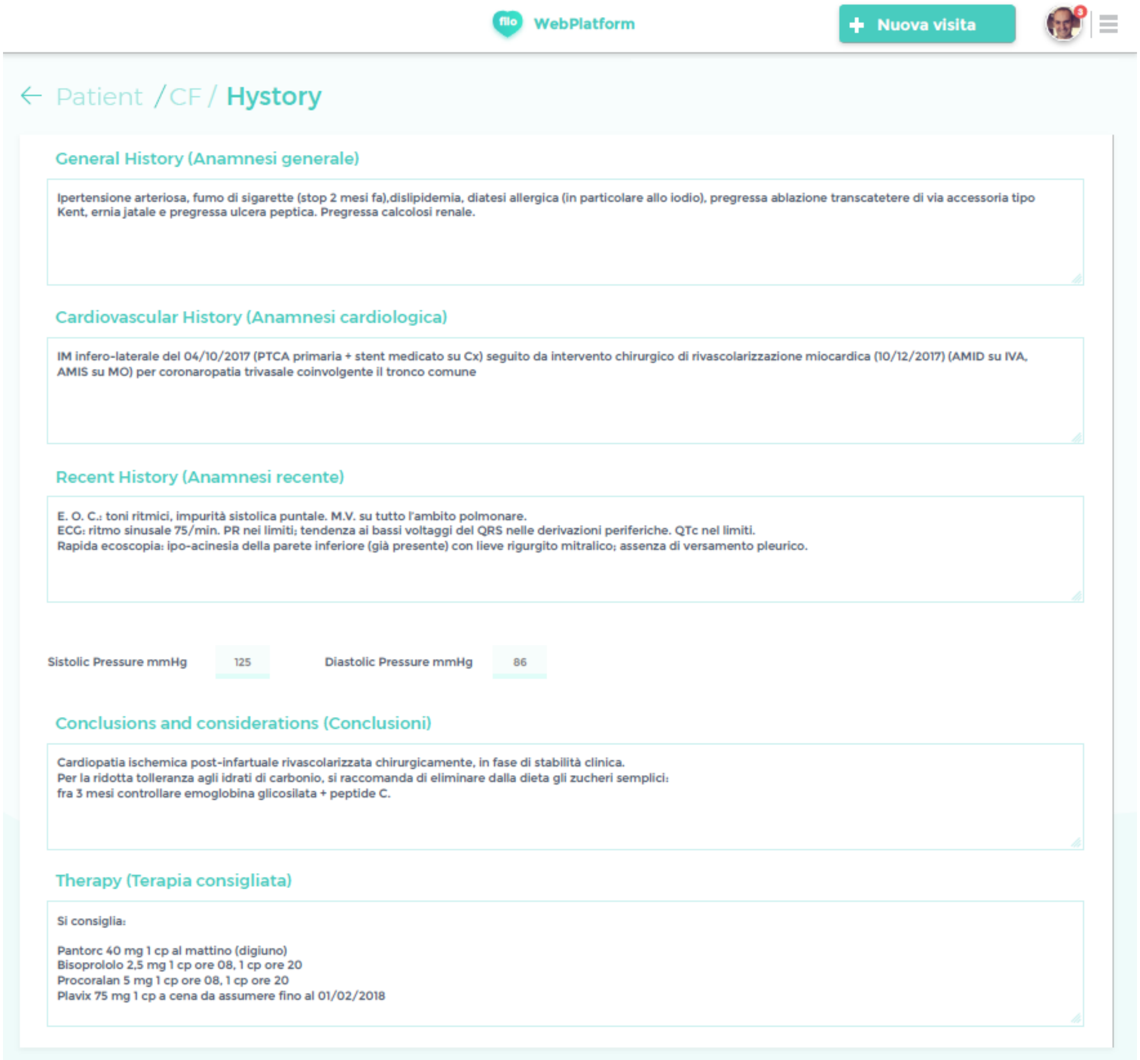


Figure C.6: Patient Page: History

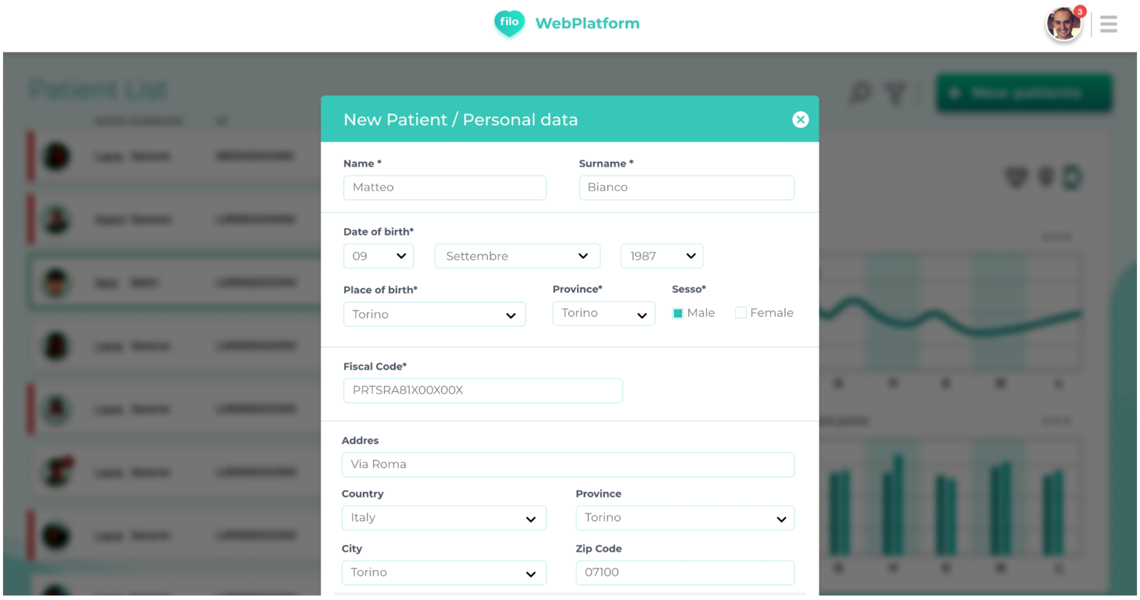


Figure C.7: Enter New Patient (1)

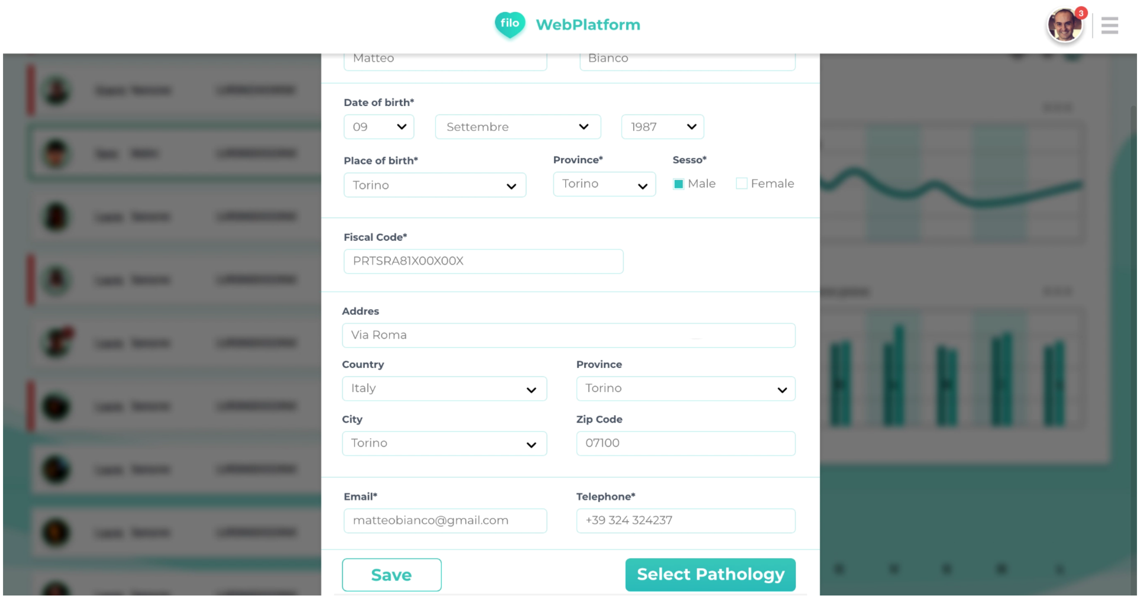


Figure C.8: Enter New Patient (2)

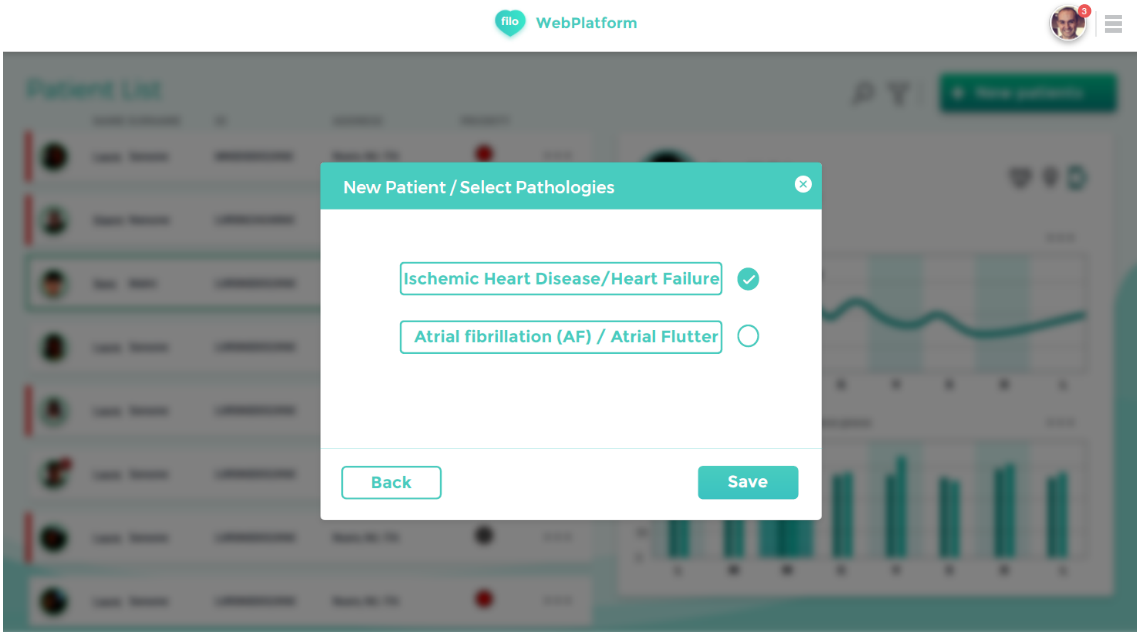


Figure C.9: Select Pathology

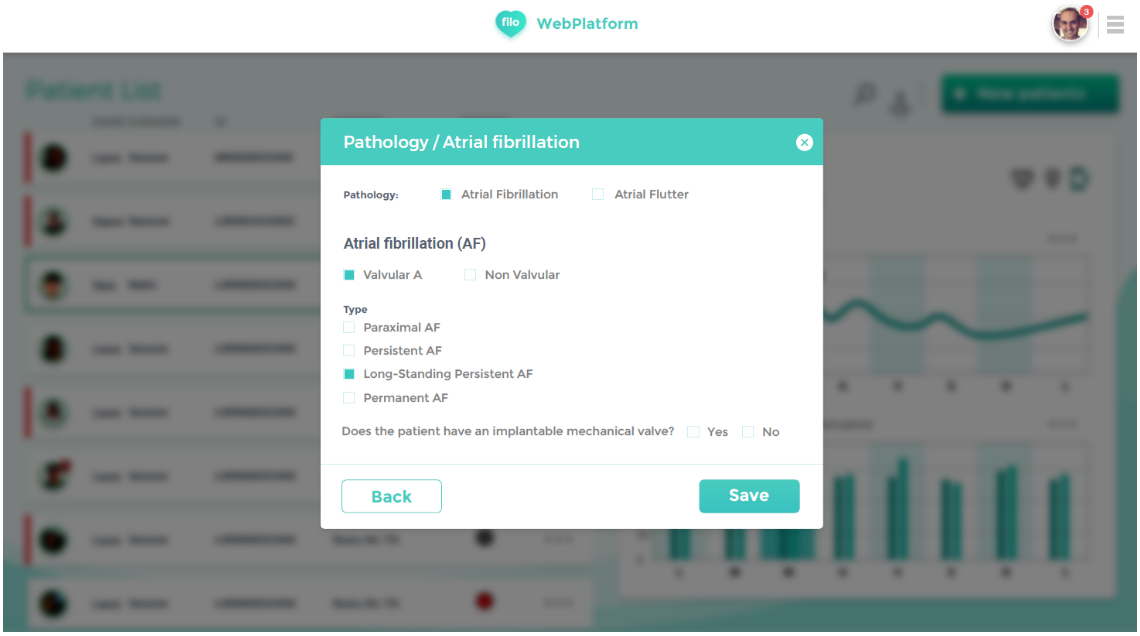


Figure C.10: Select Pathology: Atrial Fibrillation

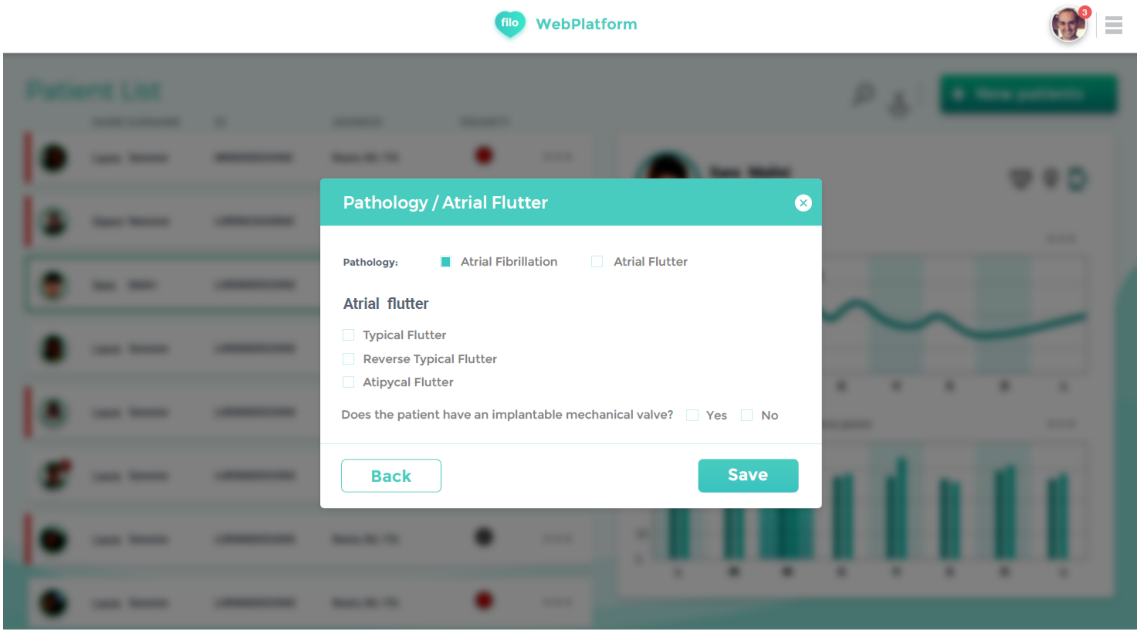


Figure C.11: Select Pathology: Atrial Flutter

## C.2 CardioFilo Graphics: Mobile Application

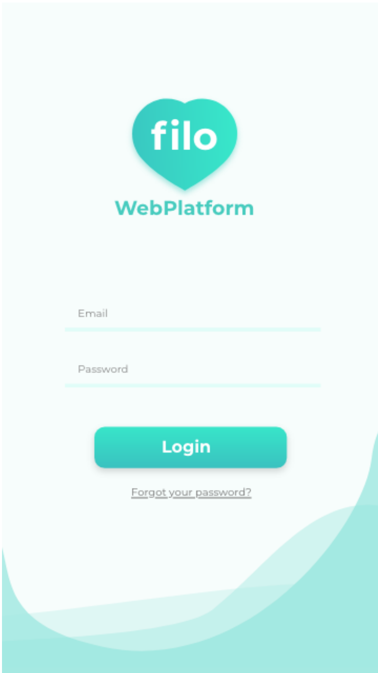


Figure C.12: Login

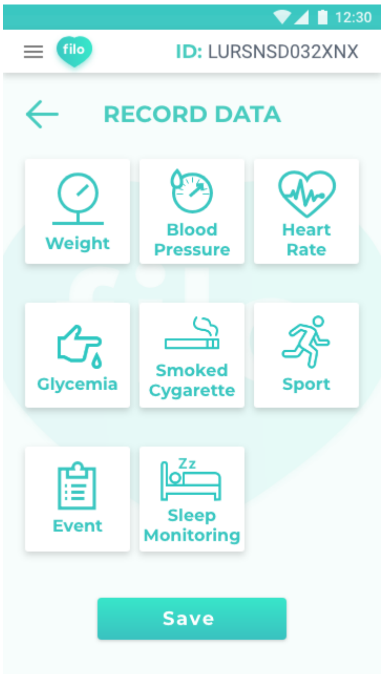


Figure C.13: Record Data

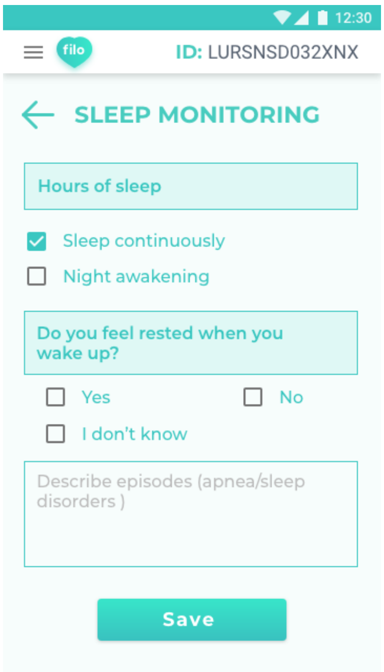


Figure C.14: Sleep Monitoring

# Appendix D

# Appendix D

## D.1 CardioFilo Web Application Code

### D.1.1 Data Models

#### Anagraphic

```
from django.db import models
from viewflow.models import Process, Task
import datetime


class Registry(models.Model):

    first_name = models.CharField(
        max_length=150
    )
    last_name = models.CharField(
        max_length=150
    )

    email = models.EmailField()

    fiscal_code = models.CharField(
        max_length=30
    )

    birthday = models.DateField(
        auto_now=False,
```

```
        auto_now_add=False,
        max_length=10
    )
    birth_place = models.CharField(
        max_length=150
    )

    residence_city = models.CharField(
        max_length=150
    )

    address = models.CharField(
        max_length=150
    )
    city = models.CharField(
        max_length=150
    )
    country = models.CharField(
        max_length=150
    )
    phone = models.CharField(
        max_length=50
    )

    # gender
    GENDER = (
        ('M', 'Male'),
        ('F', 'Female'),
    )

    gender = models.CharField(
        max_length=1,
        choices=GENDER)

    def __str__(self):
        return "Patient␣" + self.fiscal_code
```

```python
    class Meta:
        verbose_name = 'Registry'
        verbose_name_plural = 'Registry'
        permissions = (
            ('can_register_patient', 'To␣add␣the␣patient'),
            ('can_cancel', 'To␣cancel␣the␣patient␣registration')
        )


class RegistryItem(models.Model):
    patient_data = models.ForeignKey(
        Registry,
        null=True,
        on_delete=models.SET_NULL)


class RegistryProcess(Process):

    registry = models.ForeignKey(
        Registry,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
    )

    class Meta:
        verbose_name = 'Registry␣Process'
        verbose_name_plural = 'Registry␣Processes'


class RegistryTask(Task):

    class Meta:
        proxy = True
        verbose_name = 'Registry␣Task'
        verbose_name_plural = 'Registry␣Tasks'
```

**History**

```python
from django.db import models
from viewflow.models import Process, Task

import datetime
from django.core.validators import MaxValueValidator, MinValueValidator


from apps.Anagraphic.models import Registry


class History(models.Model):

    patient = models.ForeignKey(
        Registry,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
    )

    general_history = models.TextField()

    cardiovascular_history = models.TextField()

    recent_history = models.TextField()

    sistolic_blood_pressure = models.IntegerField(
        validators = [
        MinValueValidator(60),
        MaxValueValidator(270)
        ],
        blank=True,
        null=True
        )
    diastolic_blood_pressure = models.IntegerField(
        validators = [
        MinValueValidator(30),
        MaxValueValidator(120)
```

```
        ],
        blank=True,
        null=True
        )



    conclusions = models.TextField()
    changes_in_therapy = models.TextField()


    class Meta:
        verbose_name = 'History'
        verbose_name_plural = 'History'

    def __str__(self):
        return 'Patient' +
        '␣' +
        self.patient.fiscal_code +
         '␣' +
         'History'


class HistoryProcess(Process):

    history = models.ForeignKey(
        History,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
    )

    class Meta:
        verbose_name = 'History␣Process'
        verbose_name = 'History␣Processes'
```

```
class HistoryTask(Task):

    class Meta:
        proxy = True
        verbose_name = 'History Task'
        verbose_name_plural = 'History Tasks'
```

**Pathologies**

```
from django.db import models
from viewflow.models import Process, Task
import datetime
from django.core.validators import MaxValueValidator, MinValueValidator
from apps.Anagraphic.models     import Registry
from django.utils.translation import gettext as _

class Pathology(models.Model):

        patient_data = models.ForeignKey(
                Registry,
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )



        SELECT = (
                ('IS','Ischemic Heart Disease'),
                ('AF','Atrial Fibrillation/Atrial Flutter'),
                ('HF','Heart Failure'),
                ('A','All'),
                )

        disease = models.CharField(
                max_length=2,
                choices=SELECT
                )
```

```
'''
Atrial Fibrillation
'''
VALV = (
        ('V','Valvular'),
        ('NV','Non Valvular'),
        ('N','None'),
        )
fibrillation = models.CharField(
        max_length=2,
        choices=VALV
        )

TYPE = (
        ('PARO','Paroxymal AF'),
        ('PERS','Persistent AF'),
        ('LS','Long-Standing Persistent AF'),
        ('PERM', 'Permanent AF'),
        ('N','None'),
        )
type_of_Atrial_Fibrillation = models.CharField(
        max_length=4,
        choices=TYPE
        )
'''
Atrial Flutter
'''
FLUTTER = (
        ('TF','Typical Flutter'),
        ('RTP','Reverse Typical Flutter'),
        ('AF','Atypical Flutter'),
        ('N','None'),
        )
type_of_Atrial_Flutter = models.CharField(
        max_length=3,
        choices=FLUTTER
        )
```

```
        # For both AF and Atrial Flutter
        VALVE = (
                ('Y','Yes'),
                ('N','No'),
                )
        mechanical_valve = models.CharField(
                _(u'Does the patient have an implanted mechanical valve?'),
                max_length=1,
                choices=VALVE
                )



        def __str__(self):
                return 'Patient' +
                ' ' +
                self.patient_data.fiscal_code +
                ' ' +
                'pathology selection'

        class Meta:
                verbose_name = 'Pathology'
                verbose_name_plural = 'Pathologies'

class PathologyProcess(Process):

        pathology = models.ForeignKey(
                Pathology,
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )

        class Meta:
```

```
        verbose_name = 'Pathology␣Process'
        verbose_name_plural = 'Pathology␣Processes'


class PathologyTask(Task):

    class Meta:
        proxy = True
        verbose_name = 'Pathology␣Task'
        verbose_name_plural = 'Pathology␣Tasks'
```

**Risk Factors**

```
from django.db import models
from viewflow.models import Process, Task
import datetime
from apps.Anagraphic.models    import Registry


class RiskFactors(models.Model):

    patient_data = models.ForeignKey(
        Registry,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

    SMOKE = (
        ('Y','Yes'),
        ('N','No'),
        )
    smoke = models.CharField(
        max_length=1,
        choices=SMOKE
        )

    HYPERTENSION = (
```

```
        ('Y', 'Yes'),
        ('N', 'No'),
        )
hypertension = models.CharField(
        max_length=1,
        choices=HYPERTENSION
        )

DYSLIPIDEMIA = (
        ('Y', 'Yes'),
        ('N', 'No'),
        )
dyslipidemia = models.CharField(
        max_length=1,
        choices=DYSLIPIDEMIA
        )

DIABETES = (
        ('Y', 'Yes'),
        ('N', 'No'),
        )
diabetes = models.CharField(
        max_length=1,
        choices=DIABETES
        )

FAMILIARITY_WITH_CORONARY_HEART_DISEASE = (
        ('Y', 'Yes'),
        ('N', 'No'),
        )
familiarity_with_coronary_heart_disease = models.CharField(
        max_length=1,
        choices=FAMILIARITY_WITH_CORONARY_HEART_DISEASE
        )

PREVIOUS_CHEMIOTHERAPY = (
        ('Y', 'Yes'),
```

```
            ('N', 'No'),
            )
    previous_chemiotherapy = models.CharField(
            max_length=1,
            choices=PREVIOUS_CHEMIOTHERAPY
            )

    PREVIOUS_RADIOTHERAPY = (
            ('Y', 'Yes'),
            ('N', 'No'),
            )
    previous_radiotherapy = models.CharField(
            max_length=1,
            choices=PREVIOUS_RADIOTHERAPY
            )

    HIV_IN_TREATMENT = (
            ('Y', 'Yes'),
            ('N', 'No'),
            )
    HIV_in_treatment = models.CharField(
            max_length=1,
            choices=HIV_IN_TREATMENT
            )

    SEVERAL_RENAL_IMPAIRMENT = (
            ('Y', 'Yes'),
            ('N', 'No'),
            )
    several_renal_impairment = models.CharField(
            max_length=1,
            choices=SEVERAL_RENAL_IMPAIRMENT
            )

    def __str__(self):
            return 'Patient' +
            ' ' +
```

```
                    self.patient_data.fiscal_code +
                    ': ' +
                    'Fattori di Rischio'

        class  Meta:
                    verbose_name = 'Risk Factors'
                    verbose_name_plural ='Risk Factors'

class RiskFactorsProcess(Process):

        factors = models.ForeignKey(
                    RiskFactors,
                    blank=True,
                    null=True,
                    on_delete=models.SET_NULL
                    )

        class Meta:
                    verbose_name = ' Risk Factors Process'
                    verbose_name_plural = 'Risk Factors Processes'


class RiskFactorsTask(Task):

        class Meta:
                    proxy = True
                    verbose_name = 'Risk Factors Task'
                    verbose_name_plural = 'Risk Factors Tasks'
```

**Status**

```
from django.db import models
from viewflow.models import Process, Task
import datetime
from django.core.validators import MaxValueValidator, MinValueValidator
from apps.Anagraphic.models      import Registry
```

```
class Status(models.Model):

    patient_data = models.ForeignKey(
        Registry,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )


    height = models.FloatField(
        validators=[
        MinValueValidator(0),
        MaxValueValidator(300)
        ]
        )

    weight = models.FloatField(
        validators=[
        MinValueValidator(30),
        MaxValueValidator(200)
        ]
        )


    ALLERGIES = (
        ('D','Drugs'),
        ('F','Food'),
        ('O', 'Others')
        )
    allergies = models.CharField(
        max_length=1,
        choices = ALLERGIES
        )

    AID = ( # Implantable Active Devices
```

```
        ('PM', 'Pacemaker'),
        ('ICD','Implantable Cardiac Defibrillator'),
        ('ILR','Implantable Loop Recorder'),
        )
active_implantable_devices = models.CharField(
        max_length=3,
        choices=AID
        )

# Previous acute coronary events

STEMI = (
        ('Y','Yes'),
        ('N','No'),
        )
previous_STEMI = models.CharField(
        max_length=1,
        choices=STEMI
        )

NSTEMI_ACS = (
        ('Y','Yes'),
        ('N','No'),
        )
previous_NSTEMI_ACS = models.CharField(
        max_length=1,
        choices=NSTEMI_ACS
        )

PCI = (
        ('Y','Yes'),
        ('N','No'),
        )
previous_PCI = models.CharField(
        max_length=1,
        choices=PCI
        )
```

```
ejection_fraction = models.IntegerField(
        validators=[
        MinValueValidator(0),
        MaxValueValidator(100)
        ]
        )

transcutaneous_oxigen_saturation = models.IntegerField(
        validators=[
        MinValueValidator(0),
        MaxValueValidator(100)
        ]
        )

ANGIO = (
        ('Y', 'Yes'),
        ('N', 'No'),
        )
angioplasty = models.CharField(
        max_length=1,
        choices=ANGIO
        )


TREAT = (
        ('Y', 'Yes'),
('N', 'No'),
)
treated = models.CharField(
        max_length=1,
        choices=TREAT
        )

TREATMENT = (
        ('DES', 'PCI-DES'),
('DES', 'PCI-DES'),
```

```
            ('DEB', 'PCI-DEB'),
            ('POBA', 'PCI-POBA'),
            ('AMIS', 'CABG-AMIS'),
            ('AMID', 'CABG-AMID'),
            ('SVG', 'CABG-SVG'),
            ('RAG', 'CABG-RAG'),
            ('N', 'None')
                )

    treatment = models.CharField(
                max_length=4,
                choices=TREATMENT
                )

    def __str__(self):
                return 'Patient' +
                ' ' +
                self.patient_data.fiscal_code +
                ' ' +
                'Status'

    class Meta:
                verbose_name = 'Status'
                verbose_name_plural = 'Status'


class StatusProcess(Process):

    patient_status = models.ForeignKey(
                Status,
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )

    class Meta :
```

```
            verbose_name = 'Status Process'
            verbose_name_plural = 'Status Processes'

class StatusTask(Task):

        class Meta:
            proxy = True
            verbose_name = 'Status Task'
            verbose_name_plural = 'Status Tasks'
```

**Therapy**

```
from django.db import models
from viewflow.models import Process, Task
import datetime
from django.core.validators import MaxValueValidator, MinValueValidator
from apps.Anagraphic.models     import Registry


class Therapy(models.Model):

        patient_data = models.ForeignKey(
            Registry,
            blank=True,
            null=True,
            on_delete=models.SET_NULL
            )

        THERAPY = (
            ('0','Anti-platelet Therapy'),
            ('1','Anti-coagulant Therapy'),
            ('2', 'Cardiovascular Therapy'),
            )

        therapy_type= models.CharField(
            max_length=1,
            choices= THERAPY
            )
```

```
ANTI_PLATELET = (
        ('T','Ticagrelor'),
        ('P','Prasugrel'),
        ('C','Clopidogrel'),
        )

anti_platelet_therapy = models.CharField(
        max_length=1,
        choices=ANTI_PLATELET
        )

ANTI_COAGULANT = (
        ('V','Valvular'),
        ('NV','Non Valvular'),
        ('N','None'),
        )
anti_coagulant_therapy = models.CharField(
        max_length=2,
        choices=ANTI_COAGULANT
        )


'''
DOAC Drugs (Direct-acting Oral AntiCoagulants)
'''
serum_creatinine = models.FloatField(
        validators=[
        MinValueValidator(30),
        MaxValueValidator(200)
        ]
        )

VERAPAMIL = (
        ('Y','Yes'),
        ('N','No'),
        )
```

```
verapamil_therapy = models.CharField(
        max_length=1,
        choices=VERAPAMIL
        )

NAO = (
        ('D','Dabigatran'),
        ('A','Apixaban'),
        ('R','Rivaroxaban'),
        ('E','Edoxaban'),
        ('N','None'),
        )
oral_anticoagulants_NAO = models.CharField(
        max_length=1,
        choices=NAO
        )

'''
Warfarin Drugs
'''
start_therapy = models.DateField(
        auto_now=False,
        auto_now_add=False,
        max_length=10
        )

end_therapy = models.DateField(
        auto_now=False,
        auto_now_add=False,
        max_length=10
        )


current_INR = models.FloatField (
        validators=[
        MinValueValidator(0),
        MaxValueValidator(100)
```

```
        ]
        )

next_INR_check = models.DateField(
        auto_now=False,
        auto_now_add=False,
        max_length=10
        )

weekly_dose_mg = models.FloatField(
        validators= [
        MinValueValidator(0),
        MaxValueValidator(100)
        ]
        )

time = models.TimeField(
        auto_now_add=False,
        blank=True
        )


enter_drug = models.TextField(
        max_length=150
        )


def __str__(self):
        return 'Patient' +
        '␣' +
        self.patient_data.fiscal_code +
    '␣' +
        'Therapy␣Plan'


class Meta:
        verbose_name = 'Terapia'
```

```
            verbose_name_plural = 'Terapie'

class TherapyProcess(Process):

        therapy = models.ForeignKey(
            Therapy,
            blank=True,
            null=True,
            on_delete=models.SET_NULL
            )

        class Meta:
            verbose_name = 'Therapy␣Process'
            verbose_name_plural = 'Therapy␣Processes'



class TherapyTask(Task):

        class Meta:
            proxy = True
            verbose_name = 'Therapy␣Task'
            verbose_name_plural = 'Therapy␣Tasks'
```

**Visit**

```
from django.db import models
from viewflow.models import Process, Task
import datetime
from apps.Anagraphic.models     import Registry
from apps.RiskFactors.models import RiskFactors
from apps.History.models import History
from apps.Status.models import Status
from apps.Therapy.models import Therapy
from apps.Pathologies.models import Pathology

class Visit(models.Model):
```

```
patient = models.ForeignKey(
        Registry,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

select_pathology = models.ForeignKey(
        Pathology,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

risk_factors = models.ForeignKey(
        RiskFactors,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

history = models.ForeignKey(
        History,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

status = models.ForeignKey(
        Status,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )

therapy = models.ForeignKey(
        Therapy,
```

```
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )


    def __str__(self):
                return 'Patient' +
                '␣' +
                self.patient.fiscal_code +
                '␣' +
                'Visit'

    class Meta:
                verbose_name = 'Visit'
                verbose_name_plural = 'Visits'

class VisitProcess(Process):

    visit = models.ForeignKey(
                Visit,
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )
    class Meta:

                verbose_name = 'Visit␣Process'
                verbose_name_plural = 'Visit␣Processes'

class VisitTask(Task):

    class Meta:
                proxy = True
                verbose_name = 'Visit␣Task'
                verbose_name_plural = 'Visit␣Tasks'
```

### D.1.2 User Models and User Permissions

**Doctors**

```
from django.db import models
from viewflow.models import Process, Task


class Doctor(models.Model):
        name = models.CharField(
                max_length=255
                )
        surname = models.CharField(
                max_length=255
                )
        specialization = models.CharField(
                max_length=255
                )
        visit = models.BooleanField(
        default=True
        )
        email = models.EmailField()


        def __str__(self):
                return self.name +
                "␣" +
                self.surname +
                "␣" +
                self.specialization

        class Meta:
                verbose_name = 'Doctor'
                verbose_name_plural = 'Doctors'
                permissions = (
                        ("can_login",
                        "To␣enter␣the␣CardioFilo␣website"),
                ("can_reset_own_password",
```

```
        "To␣change␣personal␣password"),
("can_reset_pat_password",
        "To␣reset␣patient's␣password"),
("can_delete_alert",
        "To␣remove␣alarms"),
("can_plan_therapy",
        "To␣provide␣a␣therapy␣plan"),
("can_logout",
        "To␣exit␣the␣CardioFilo␣website"),
("can_add_patient",
        "To␣add␣a␣patient␣on␣the␣list"),
("can_personal_data",
        "To␣register␣patients"),
("can_remove_patient",
        "To␣remove␣a␣patient␣from␣a␣personal␣patient␣list"),
        ("can_patient_list",
        "To␣select␣patients␣from␣the␣list"),
        ("can_visit",
        "To␣perform␣a␣visit"),
        ("can_select_disease",
        "To␣enter␣patient's␣pathologies"),
        ("can_risk_factors",
        "To␣enter␣patient's␣risk␣factors"),
        ("can_status",
        "To␣modify␣patient'status␣info"),
        ("can_history",
        "To␣write␣considerations␣about␣the␣patient"),
        ("can_overview",
        "To␣see␣patient␣health␣trend␣over␣time"),
        ("can_upload",
        "To␣upload␣exam␣results"),
        ("can_download",
        "To␣download␣exam␣results"),
        ("can_contact",
        "To␣contact␣patients␣via␣e-mail␣or␣phone␣number"),
        ("can_report",
        "To␣report␣exam␣results,␣like␣an␣ECG")
```

```
                        )


        @property
        def can_visit(self):
                return self._can_visit

class DoctorsListProcess(Process):

        doctor = models.ForeignKey(
                Doctor,
                blank=True,
                null=True,
                on_delete=models.SET_NULL
                )

        class Meta:
                verbose_name = 'Doctors␣List␣Process'
                verbose_name_plural = 'Doctors␣List␣Processes'

class DoctorsListTask(Task):
        class Meta:
                proxy = True
                verbose_name = 'Doctors␣List␣Task'
                verbose_name_plural = 'Doctors␣List␣Tasks'
```

**Patients**

```
from django.db import models
from viewflow.models import Process, Task
from apps.Anagraphic.models     import Registry


class Patient(models.Model):


        personal_data = models.ForeignKey(
                Registry,
```

```
            blank=True,
            null=True,
            on_delete=models.SET_NULL
            )

DISEASE = (
            ('IS', 'Ischemic heart disease'),
('AF', 'Atrial Fibrillation'),
('B', 'Both'),
('N', 'None'),
)
disease = models.CharField(
            max_length=2,
            choices=DISEASE
            )
event = models.BooleanField(
            default=True
            )
def __str__(self):
            return 'Patient' + ' ' + self.personal_data.fiscal_code


class Meta:

            verbose_name = "Patient"
            verbose_name_plural = "Patients"
            permissions = (
                        ("can_login",
                        "To enter the CardioFilo app"),
      ("can_reset_own_password",
                        "To change personal password"),
      ("can_therapy_plan",
                        "To update therapy plan"),
      ("can_logout",
                        "To exit the CardioFilo website"),
      ("can_record_events",
                        "To enter an event like bleeding or hospitalization")
```

```
        )

    @property
    def can_record_events(self):
        return self._can_record_events

class PatientListProcess(Process):

    patient = models.ForeignKey(
        Patient,
        blank=True,
        null=True,
        on_delete=models.SET_NULL
        )
    class Meta:
        verbose_name = 'Patient␣List␣Process'
        verbose_name_plural = 'Patient␣List␣Processes'

class PatientListTask(Task):

    class Meta:
        proxy = True
        verbose_name = 'Patient␣List␣Task'
        verbose_name_plural = 'Patient␣List␣Tasks'
```

**Nurses**

```
from django.db import models
from viewflow.models import Process, Task


class Nurse(models.Model):
    nurse_id = models.CharField(max_length=255)
    visit = models.BooleanField(default=True)

    def __str__(self):
        return self.nurse_id
```

```
    class Meta:
        verbose_name = "Nurse"
        verbose_name_plural = "Nurses"
        permissions = (
            ("can_login",
            "To␣enter␣the␣CardioFilo␣website"),
        ("can_reset_own_password",
            "To␣change␣personal␣password"),
        ("can_add_drug",
            "To␣enter␣a␣drug␣not␣in␣the␣cardiovascular␣therapy"),
        ("can_logout",
            "To␣exit␣the␣CardioFilo␣website"),
        ("can_add_patient",
            "To␣add␣a␣patient␣on␣the␣list"),
        ("can_personal_data",
            "To␣register␣patients"),
            ("can_select_disease",
            "To␣enter␣patient's␣pathologies"),
            ("can_risk_factors", "To␣enter␣patient's␣risk␣factors"
            ("can_status",
            "To␣modify␣patient'status␣info"),
            ("can_upload",
            "To␣upload␣exam␣results"),
            ("can_visit",
            "To␣perform␣a␣visit")
            )


class NurseListProcess(Process):
    nurse = models.ForeignKey(
        Nurse,
        null=True,
        blank=True,
        on_delete=models.SET_NULL
    )

    class Meta:
```

```
            verbose_name = "Nurse␣List␣Process"
            verbose_name_plural = "Nurse␣List␣Process"


class NurseListTask(Task):
    class Meta:
            proxy = True
            verbose_name = 'Nurse␣List␣Task'
            verbose_name_plural = 'Nurse␣List␣Tasks'
```

**Doctors from Scientific Societies**

```
from django.db import models
from viewflow.models import Process, Task


class ScientificSociety(models.Model):

    doctor_name = models.CharField(
            max_length=255
            )
    doctor_surname = models.CharField(
            max_length=255
            )
    society_name = models.CharField(
            max_length=255
            )

    email = models.EmailField()


    def __str__(self):
            return self.doctor_name +
            "␣" +
            self.doctor_surname +
            "␣" +
            self.society_name
```

```
    class Meta:

        verbose_name = 'Scientific␣Society'
        verbose_name_plural = 'Scientific␣Societies'

        permissions = (
            ("can_login",
            "To␣enter␣the␣CardioFilo␣website"),
            ("can_see_some_info",
            "Can␣only␣read␣a␣few␣patient's␣personal␣data"),
        ("can_reset_own_password",
            "To␣change␣personal␣password"),
        ("can_logout",
            "To␣exit␣the␣CardioFilo␣website"),
            ("can_see_patient_list",
            "To␣see␣anonymous␣patients␣from␣the␣list"),
            ("can_select_disease",
            "To␣choose␣patient's␣pathologies"),
            ("can_see_risk_factors",
            "To␣see␣anonymous␣patient's␣risk␣factors"),
            ("can_see_status",
            "To␣see␣anonymous␣patient'status␣info"),
            ("can_see_history",
            "To␣read␣general␣considerations␣about␣the␣patient"),
            ("can_see_overview",
            "To␣see␣patient␣health␣trend␣over␣time"),
            ("can_download",
            "To␣download␣anonymous␣reports/exam␣results"),
            ("can_see_therapy_plan",
            "To␣see␣anonymous␣patient's␣therapy"),
            )

class ScientificSocietyProcess(Process):

    society = models.ForeignKey(
        ScientificSociety,
        blank=True,
```

```
                null=True,
                on_delete=models.SET_NULL
                )


        class Meta:

                verbose_name = 'Scientific Society Process'
                verbose_name_plural = 'Scientific Society Processes'

class ScientificSocietyTask(Task):

        class Meta:
                proxy = True
                verbose_name = 'Scientific Society Task'
                verbose_name_plural = 'Scientific Society Tasks'
```

# Bibliography

[1] M. Cerutti, "Design of an application for patient monitoring after coronary angioplasty," 4 2018.

[2] Scompenso cardiaco. [Online]. Available: https://www.fondazioneveronesi.it/magazine/tools-della-salute/glossario-delle-malattie/scompenso-cardiaco

[3] Caregiver. [Online]. Available: https://en.wikipedia.org/wiki/Caregiver

[4] Visual Paradigm. [Online]. Available: https://www.visual-paradigm.com/

[5] Cacoo. [Online]. Available: https://cacoo.com/

[6] Balsamiq Mockups. [Online]. Available: https://balsamiq.com/

[7] T. E. PARLIAMENT and T. C. O. T. E. UNION, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," *Official Journal of the European Union*, pp. 1–88, 4 2016.

[8] Firebase. [Online]. Available: https://firebase.google.com/

[9] Firebase sicuro o insicuro? Attenzione: gli sviluppatori non mettano a rischio i dati critici. [Online]. Available: https://www.zerounoweb.it/techtarget/searchsecurity/iot-automated-patch-management-e-altre-sfide-di-sicurezza/

[10] L. Karni, "openehr approach for the evauation and management of elderly patients with multi-morbidity," *Research Gate*, pp. 1–17, 5 2016.

[11] Who is using openEHR? [Online]. Available: https://www.openehr.org/community/membership/

[12] openEHR systems. [Online]. Available: http://www.marand.com/thinkehr/

[13] Cabolabs. [Online]. Available: https://www.cabolabs.com/products

[14] Clinical Knowledge Manager. [Online]. Available: https://openehr.org/ckm/

[15] Archetype Editor. [Online]. Available: http://oceanhealthsystems.com/products/archetype-editor

[16] Template Designer. [Online]. Available: http://oceanhealthsystems.com/products/template-designer

[17] G. E. Bente Christensen, "Evaluating model-driven development for large-scale ehrs through the openehr approach," *International Journal of Medical Informatics*, pp. 43–54, 2 2016.

[18] Django Documentation. [Online]. Available: https://docs.djangoproject.com/en/2.1/

[19] SQLite. [Online]. Available: https://www.sqlite.org/index.html

[20] T. C. O. T. E. COMMUNITIES, "Council directive 93/42/eec of 14 june 1993 concerning medical devices," *Official Journal of the European Union*, pp. 1–60, 6 1993.

[21] Software as a Medical Device (SaMD). [Online]. Available: http://www.imdrf.org/

[22] E. COMMISSION, "Guidelines on the qualification and classification of stand alone software used in healthcare within the regulatory framework of medical devices," pp. 1–29, 7 2016.

[23] Cardiopatia Ischemica. [Online]. Available: https://www.humanitas.it/malattie/cardiopatia-ischemica

[24] X. L. J. A. Lingtong Min, Qi Tian and H. Duann, "An openehr based approach to improve the semantic interoperability of clinical data registry," *Research Gate*, pp. 50–56, 11 2017.

[25] P. Pazos, "Towards the implementation of an openehr-based open source ehr platform (a vision paper)," *Research Gate*, pp. 45–50, 8 2015.

[26] V. Y. Georgy Kopanitsa, Hasan Veseli, "Development, implementation and evaluation of an information model for archetype based user responsive medical data visualization," *Journal of Biomedical Informatics*, pp. 196–205, 4 2015.

[27] P. F. S. L. M. R. Diego A. Orellana, Alberto A. Salas and V. I. Rotger, "Evaluation of a framework to implement electronic health record systems based on the openehr standard," *Journal of Physics*, pp. 1–10, 8 2018.

[28] H. H. Vaclav Papez, Spiros Denaxas, "Evaluating openehr for storing computable representations of electronic health record phenotyping algorithms," *Research Gate*, pp. 1–7, 4 2017.

[29] N. E. H. C. Fadoua Khennoua, Youness Idrissi Khamlichib, "Improving the use of big data analytics within electronic health records: A case study based openehr," *Procedia Computer Science*, pp. 60–68, 3 2018.

[30] C. C. Stefan Kropf and K. Denecke, "A short briefing: Domain modeling of ehrs using the openehr tools clinical knowledge manager (ckm), archetype editor (ae) and template designer (td)," *Research Gate*, pp. 1–18, 1 2018.

[31] T. E. PARLIAMENT and T. C. O. T. E. UNION, "Regulation (eu) 2017/745 of the european parliament and of the council of 5 april 2017 on medical devices, amending directive 2001/83/ec, regulation (ec) no 178/2002 and regulation (ec) no 1223/2009 and repealing council directives 90/385/eec and 93/42/eec," *Official Journal of the European Union*, pp. 1–175, 5 2017.

[32] App medicali e nuovo Regolamento UE sui dispositivi medici. [Online]. Available: https://www.filodiritto.com/articoli/2017/05/app-medicali-e-nuovo-regolamento-ue-sui-dispositivi-medici.html

[33] Software e App mediche: quando diventano medical devices secondo la Corte di Giustizia. [Online]. Available: http://www.lab-ip.net/software-e-app-mediche-quando-diventano-medical-devices-secondo-la-corte-di-giustizia/

[34] Django Forms. [Online]. Available: https://djangobook.com/django-forms/

[35] Django Forms. [Online]. Available: https://djangobook.com/django-forms/

[36] Django Views and Template. [Online]. Available: http://www.allafinedelpalo.it/python-django-2-view-e-template/

[37] Django Tutorials. [Online]. Available: https://tutorial.djangogirls.org/en/