

POLITECNICO DI TORINO

DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

Master of Science in Biomedical Engineering



FrontEnd Development of a Web Application for Patient Monitoring after Coronary Angioplasty

Supervisor:

Prof. Monica Visintin

Co-rapporteur:

Ing. Sara Prete

Candidate

Federica Lorusso

ACADEMIC YEAR 2019-2020

Acknowledgements

Prima di procedere con la stesura di questa tesi, e' doveroso ed essenziale ringraziare tutte le persone che mi hanno supportato in questo lavoro e in questi lunghi 6 anni di percorso universitario.

In primis vorrei ringraziare la Professoressa Monica Visintin per avermi dato fiducia e per essere stata sempre disponibile, nonostante il periodo disastroso attraversato negli ultimi mesi, gentile e attenta nello svolgimento del mio lavoro di tesi.

Vorrei ringraziare anche l'ing. Sara Prete per avermi guidata nel lavoro di tesi ed avermi ascoltata in ogni momento. E' stata un punto di riferimento sostanziale per me in questi mesi.

Grazie inoltre a tutto il team Abinsula per avermi dato fiducia in questo percorso e per avermi reso parte di questo meraviglioso progetto nella loro azienda.

Un grazie lungo una vita intera alla mia famiglia, che fin da subito ha creduto in me ed ha sacrificato sempre tutto pur di rendermi felice nel raggiungere il mio obiettivo.

Grazie mamma per essere il mio punto di riferimento e per aver passato giornate intere ad ascoltare tutte le mie paturnie, sei la mia ombra nonostante i mille chilometri che ci distanziano.

Grazie papà', per essere sempre stato il primo a difendermi in ogni mia decisione, per avermi aiutato sempre a gestire ogni situazione e per essere sempre pronto a mandarmi i famosi pacchi da giu' come sostegno umanitario e morale.

Grazie Francesco, il mio fratellino che e' sempre stato il mio punto di forza maggiore e a cui voglio un mondo di bene e a cui auguro tantissima felicità e fortuna.

Grazie al Collegio Einaudi che mi ha ospitato per ben 5 anni ed e' stata la mia casa, non mi ha fatto mai sentire sola e mi ha permesso di conoscere tantissime persone. Ha reso il mio percorso universitario piu' divertente e meno monotono.

Grazie a Giusi, la mia sorella mancata, a Carlotta, un tripudio di dolcezza, amore e pazienza, ad Alessio, presenza costante e che riesce sempre a farti sorridere e a tutti i miei amici e amiche che in ogni momento riescono a sorprendermi e a farmi capire di avere una famiglia bis anche qui a Torino.

Grazie Andrea, anche per te un grazie lungo una vita intera. Grazie perche' sei sempre li'

pronto ad aiutarmi anche e soprattutto in periodi che mai avrei pensato di attraversare. Grazie per il tuo amore, dolcezza, pazienza, comprensione e ventata di positività che ogni giorno sei capace a trasmettermi.

Un grazie a tutta la tua famiglia per avermi in qualche modo adottata in questi tre anni, vi voglio davvero bene e vi porterò sempre nel mio cuore.

Un piccolo grazie alla determinazione, perseveranza e grinta con cui ho sempre colorato questi anni non sempre facili.

Concludo con il ringraziare Aurora e mio fratello per avermi regalato la gioia più grande in questi mesi, dedico il mio lavoro di tesi alla mia nipotina che fin da subito ha dimostrato d'essere forte.

Ti aspettiamo con ansia e spero che un giorno anche tu possa essere fiera della tua zia.

Summary

Cardiovascular diseases (ischemic heart diseases, such as acute myocardial infarction and angina pectoris, and cerebrovascular diseases, such as ischemic and haemorrhagic stroke) are among the main causes of invalidity and mortality in Italy, being responsible for the 35.8% of all deaths (32.5% in males and 38.8% in females). In particular, according to Istat (The National Institute of Statistics) 2017 data, ischemic heart disease is responsible for 10.4% of all deaths (11.3% in males and 9.6% in females), while cerebrovascular accidents by 9.2% (7, 6% in males and 10.7% in females).

However, these deaths are largely preventable as it is possible to monitor risk factors related to behaviors and lifestyles (smoking, alcohol consumption, poor diet, sedentary lifestyle) often present due to diabetes, obesity, hypercholesterolemia, hypertension.

The Cardiofilo project was created in 2017 with the aim of preventing this type of death by providing a health model for patients suffering from atrial fibrillation, myocardial infarction and / or undergoing coronary angioplasty or other catheterization revision-procedures. In particular, attention is focused on the post-operative course of heart attack patients and an attempt is made to provide an additional tool for secondary prevention of cardiovascular diseases. In general, post-operative health care guarantees few cardiological visits (approximately one visit after 3-6 months or even annually.)

The Cardiofilo project (the name derives from the idea of wanting to recreate a "filo" between the cardiologist and the patient) will allow easier, direct and frequent communication between patient and doctor, through a mobile app (managed directly by the patient's smartphone and / o caregiver) and a web app (managed by the cardiologist and / or nurses). It can be customary for the patient to check values such as blood pressure, weight, blood sugar, which he can then easily send to the doctor, so that he can keep track of any disturbing changes in the parameters. In a previous thesis work, a first version of the web and mobile application was developed[1].

However, improvements have been introduced in the work of this thesis both in the Front-End of the web application and improvements made with the aim of making the application marketable: having an adequate processing of personal data and understanding what are the bases for proceed with a possible CE marking of the product.

There are therefore two servers: a server where all the personal data of the patients will be stored and another OpenEHR repository for health data (a clinical gateway will be needed to interface with the repository).

This thesis work was carried out at Abinsula s.r.l., a Sardinian company that operates in the IT sector and in software design consultancy services.

The main requirements of the application were suggested by the cardiologists of the San Giovanni Molinette Hospital and San Luigi (Orbassano) who requested the implementation of the application.

List of acronyms

EHR Electronic Health records

FE Front-end

HTML HyperTextMarkup Language

CSS Cascading Style Sheets

SGML Standard Generalized Markup Language

W3C World Wide Web Consortium

XML eXtensible Markup Language

XHTML eXtensible HyperText Markup Language

ECMA European Computer Manufacturers Association

ISO International Organization for Standardization

IEC International Electrotechnical Commission

DOM Document Object Model

MVC Model View Controller

UI User Interface

API Application Programming Interface

GDPR General Data Protection Regulation

IP Internet Protocol

DNA DeoxyriboNucleic Acid

RNA RiboNucleic Acid

DPA Data Protection Authorities

DPO Data Protection Officer

EEC European Economic Community

GUCE Official Journal of the European Union

CE European Conformity

SaMD Software as a Medical Device

PR Public relations

DM Medical devices

List of Figures

2.1	Cardiofilo system, with a schematic representation of the web app interface	15
2.2	Original Screen of Risk Factor	20
2.3	Proposed Screen of Risk Factor	21
2.4	Proposed Screen of Risk Factor	21
2.5	Original Screen of Anamnesi	22
2.6	Proposed Screen of Anamnesi	23
2.7	Original Screen of Therapy	24
2.8	Proposed Screen of Therapy	25
2.9	Proposed Screen of Therapy	25
2.10	Proposed Screen of Therapy	26
3.1	Example of a simple HTML page	31
3.2	Example of DOM tree generated by the web browser in the interpretation of the HTML file	32
3.3	Example of a simple HTML page with CSS style	33
3.4	Example of a simple HTML page with CSS style and Javascript syntax . .	35
3.5	Example of a simple HTML page with CSS style and Javascript syntax . .	36
3.6	General scheme relating to the operation of React	39
3.7	Script and the terminal integrated into Visual Studio Code	42
3.8	Installation through node.js npm package and creation of a new React project folder	43
3.9	Folders present in the created "my-app" project	44
3.10	Contents of src folder	46
3.11	Contents present in App.js	47
3.12	Index.js script	49
3.13	Creating a function component in App.js	50
3.14	Creating a class component in file Intestazione.js	51
3.15	Intestazione component	52
3.16	Creating a class component in file Button.js	52
3.17	Path of Patient and Form component	52
3.18	Creating a class component in file Form.js	53
3.19	Overview interface	54
3.20	Link to overview screen's script	55
3.21	Breadcrumbs component script	56
3.22	Filter component script	57
3.23	Calendar component script	58
3.24	Overview script	59
3.25	History interface	60
3.26	Link to History screen's script	61
3.27	History script	62
3.28	Dashboard components	63
3.29	UploadButtons component	64
3.30	Component from Material UI	65
3.31	Terminal execution of the application	66

Contents

Acknowledgements	0
Summary	3
List of Figures	7
1 Introduction	10
2 Cardiofilo Web App	13
2.1 Pathologies of interest	16
2.1.1 Ischemic Heart Disease	16
2.1.2 Atrial fibrillation	16
2.1.3 Atrial flutter	17
2.2 Cardiofilo Web app	18
2.3 Future Design Improvements	19
3 Frontend Development of Cardiofilo	27
3.1 Front-End development tools	30
3.2 Framework for user interface creation	36
3.2.1 React	38
3.3 Front-End development of Cardiofilo	42
3.3.1 Create a new project with function "Create React App"	43
3.3.2 Components present in Cardiofilo Web App	47
4 General Data Protection Regulation and Cardiofilo	67
4.1 Processing of personal data including biomedical data	67
4.2 Contents of the GDPR regulation	68
4.3 Data	69
4.4 GDPR application in Europe	70
4.5 Health data treatment	71
4.6 How to treat Cardiofilo data	73

5	Classification of Cardiofilo as a medical device	74
5.1	Introduction of medical devices	74
5.2	Directive CEE 93/42	75
5.3	CE Marking	81
5.4	Classification of Cardiofilo as a SaMD	83
6	Future improvements	85
7	Conclusions	88
I	Appendix	90
	Bibliography	120

Chapter 1

Introduction

The introduction of new information and telecommunication technologies has radically changed people's daily activities, improving the quality of life and work of people in various fields.

One of the sectors that has benefited most with the advent of the merger between information technology and biomedicine is healthcare.

In this context, the birth of telemedicine is inserted, with which it was possible to conceive the doctor's work in a different way. With the help and technological support, the doctor has the ability to control and monitor patients even if they are physically distant. In this way the doctor is able to optimize time and resources, as well as facilitating the weakest groups of patients who are unable to move with hospital checks.

The doctor will therefore proceed with making the diagnosis or simply a clinical opinion on a patient through the remote transmission of data produced by diagnostic tools or manually entered by the patient.

However, it should be noted that the purpose of telemedicine is not to replace the figure of the doctor, but to support his activity without distorting the doctor-patient interpersonal relationship that must be created in the therapeutic act, the patient must always consider the doctor as an active part of the care process.

Furthermore, telemedicine has an important psychological impact in the life of the patient and his family, since the continuous remote monitoring will allow patients to always feel supervised by the doctor without having to leave the family environment and the moral support of their loved ones[2].

The first experiments in the field of telemedicine were conducted with the aim of providing adequate assistance in the most remote geographical areas or in disadvantaged situations. Subsequently, with the diffusion of more effective data compression techniques and ever faster networks, even voluminous data, such as the images of a computed tomography (CT), were sent via the fixed network.

In Italy, one of the first telemedicine applications consisted in the experimental transmission of electrocardiograms at a distance, started in 1976, using normal telephone lines. Since then, research institutions, universities, scientific societies, the National Research Council (CNR) and the Ministry of Health, working on various projects have guided research towards remote assistance techniques that could in some way improve the patient's life by optimizing the resources and time available to doctors. The current situation in our country regarding telemedicine has not fully taken off, in fact there was a first attempt at regulation with the National Guidelines in Telemedicine sanctioned in the State Regions Conference on February 20, 2014 [3].

These guidelines were then implemented by the various Regions, however very little has been done at the application level. Apart from some glaring examples in Lombardy and Veneto where telemedicine is applied for the management of chronic patients and some cases scattered in other regions in Italy, there is no single national project, which is coherent and interoperable.

Telemedicine could really make a difference especially if it is used in all Regions, because at the ideal level the consultation can take place from any part of the country for any patient: the importance and rediscovery of telemedicine has been there with the onset of the pandemic Covid-19 worldwide.

Thanks to the remote support in the medical field, hospitalizations have been reduced by 25% [3], fewer accesses to hospitals, efficient monitoring of chronic patients and optimal care of traditional patients, virtual triages that could have avoided the contagions of doctors and health professionals in this pandemic period, real-time monitoring of patients affected by Covid-19 and in home isolation and substantially fewer deaths were recorded. With the help of telemedicine, therefore, it was possible to remotely monitor, when possible, patients suffering from Covid-19 but at the same time it made it possible not to neglect the monitoring of chronic patients who normally carry out checks on a weekly or monthly basis, depending on the pathology.

Cardiofilo's project is inserted in the context of telemedicine with the aim, from the beginning, of providing the doctor with technological help for the diagnostic-therapeutic process, while at the same time preventing all the risks associated with patients suffering from cardiovascular diseases. The project was therefore conceived as a real social and cultural innovation in the health field that facilitates remote communication between doctor and patient (especially if they are not in the same location) and facilitates the provision of health services, from diagnosis to therapy.

Telemedicine, in this case, therefore, has the task of offering new solutions and perspectives to the constant demand for health care by the population who would like health care that is more efficient, more technological, more dynamic and closer to people.

Furthermore, it is responsible for:

- providing the tools to facilitate communication and interaction between the doctor and the patient and between the doctors themselves;
- breaking down geographical and temporal barriers, compensating for the uneven distribution of health services throughout the territory;
- reach a greater number of people (especially those who live in remote areas or with poor health facilities);
- speed up bureaucratic-administrative procedures;
- help the patient in research and advice g the doctor;
- simplify the online submission of diagnostic tests;
- allow online viewing of exams without loss of image quality;
- eliminate long waiting lists, guaranteeing the quality of the service and ensuring the protection of the processing of sensitive personal data.

The hope that this period of crisis and global fear may have made people appreciate the importance of a world as large and unexplored as telemedicine is really great.

Investment in the healthcare world assisted with the technological innovations still existing today is never a grant.

This thesis project is therefore a hymn to innovation, prevention, computerization of digital processes in the medical field and health practice through the support of Internet Technologies tools, specialized personnel and specific doctor-patient communication techniques.

Chapter 2

Cardiofilo Web App

Cardiovascular diseases still represent the main cause of death in our country, in particular more than 230 thousand people die a year from ischemias, heart attacks, heart and cerebrovascular diseases.

Those who survive a heart attack become chronically ill. The disease changes the quality of life and the patient appears to be much more sensitive and vulnerable. This vulnerability needs subsequent follow-up checks and visits.

Cardiofilo was born with the aim of making communications between doctor and patient faster and more frequent (otherwise the patient would have a follow-up visit every 3/6 months).

The doctor, through a web app, will be able to keep track of the evolution of the parameters through a mobile app by the patient or by the caregiver (for example blood pressure, heart rate, weight or adverse events) and to highlight in time any important and risky changes. Both the Cardiofilo web app and the Cardiofilo mobile app must:

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

The web app and the mobile app interface via the Internet (connecting via Wi-Fi/ADSL, Ethernet or 3G / 4G) with two servers, one dedicated to personal data storage, the other dedicated to clinical data storage.

A server receives and stores personal data (for example personal data acquired through the "new patient" procedure in the web app and then sent to the mobile app or simply kept in the repository). The server turns out to be a very safe and secure data retention method and this will ensure that the processing of personal data is carried out successfully according to GDPR.

The other repository is dedicated to storing clinical data related to patients (for example

glycemic, blood pressure, weight data) entered either by the patient through the mobile app or by the doctor through the web app. However, repositories dedicated to managing health data must meet certain standard specifications and in this regard, OpenEHR is introduced.

OpenEHR is a technology introduced for e-health, composed of a specific open standard in health informatics that describes how to manage, store, retrieve and exchange clinical data through electronic health records (EHR).

The repository dedicated to the management of health data in Cardiofilo therefore complies with the specifications of OpenEHR and also the repository is connected to the web app through a clinical gateway. The clinical gateway is a necessary component useful in interfacing with an OpenEHR repository and allows you to save clinical data entered by Cardiofilo users, transforming them into OpenEHR compliant compositions.

Cardiofilo web app will be managed by doctors, assisted by nurses or health workers and provides the cardiologist with the opportunity:

- to monitor the parameters in real time;
- have a more in-depth knowledge of the patient's clinical status;
- modify the therapy so that it is efficient and suitable to the patient's current situation;
- well-timed identification of the most critical patients.

Patients will only 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. 2.1)

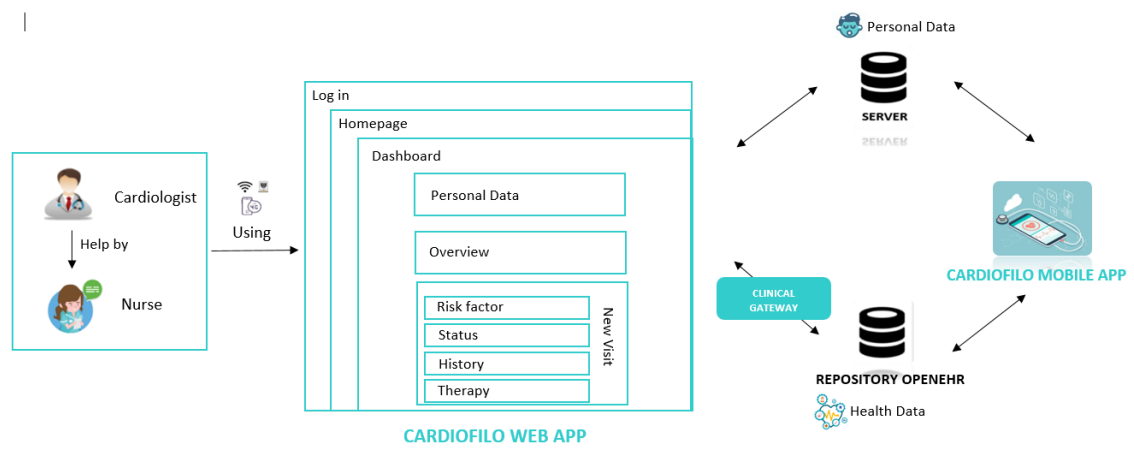


Figure 2.1: Cardiofilo system, with a schematic representation of the web app interface

2.1 Pathologies of interest

Cardiofilo project is meant for all patients who have already had a heart attack and who are therefore subject to cardiovascular diseases.

The main cardiovascular pathologies that Cardiofilo intends to treat are described in the next subsection, highlighting the importance of monitoring to prevent recurrences.

2.1.1 Ischemic Heart Disease

Ischemic heart disease includes all conditions in which there is insufficient balance of blood and oxygen to the heart. The most common cause is atherosclerosis, characterized by the presence of high-cholesterol plaques in the coronary arteries, which can prevent or decrease blood flow. Ischemic heart disease has various clinical manifestations such as stable and unstable angina pectoris and myocardial infarction. Ischemic heart disease includes, for example, heart attack and angina pectoris. The people most prone to this pathology can be classified according to the risk factors. Biological, genetic, pathological and lifestyle factors can be considered. As for biological factors, elderly patients and males are at greater risk. In women, hormone production is a protective element, which is why the risk increases with menopause. Regarding genetic factors, it is advisable that those with a history of familiarity with ischemic heart disease pay particular attention to prevention, as they may have a genetic predisposition. Pathologies such as hypercholesterolemia, hypertension, obesity and diabetes also represent risk factors. Finally, lifestyle: stress, smoking, a sedentary lifestyle and drug use increase the risk of developing ischemic heart disease.

Consequently, doctors with the help of Cardiofilo can monitor patients subject to this type of pathology paying more attention to weight, the percentage of physical activity performed, systolic and diastolic pressure. In addition, they can check, for example, if the patient has stopped smoking or if he still has smoking addiction among the risk factors. [4]

2.1.2 Atrial fibrillation

Atrial fibrillation is one of the cardiovascular pathologies in which a supra ventricular arrhythmia occurs, triggered by electrical impulses coming from myocardial muscle cells present at the junction between the four pulmonary veins and the left atrium. In atrial fibrillation, the electrical activity of the atria is completely disorganized and does not correspond to effective mechanical activity. The atrial depolarization waves are of small amplitude and have a very high frequency. In these conditions the atrioventricular node

receives many more impulses from the atrium than it is able to conduct. This variability of atrioventricular conduction causes the ventricles to contract irregularly. In the absence of pathologies and at rest, the heart rate of an adult is 60-100 beats per minute while the occurrence of atrial fibrillation, the heart rate is between 100 and 175 beats per minute. Atrial fibrillation, from a clinical point of view, is divided into:

- Paroxysmal: when the episodes occur and resolve spontaneously in less than a week;
- Persistent: when the arrhythmic episode does not stop spontaneously but only as a result of external therapeutic interventions;
- Permanent: when cardioversion attempts are not considered appropriate, or the therapeutic interventions have proved ineffective.

Also in this the doctor to prevent the pathological state, through Cardiofilo, can check the patient's state of health by observing the data related to his body weight and his frequency of physical activity carried out: a healthy style and a balanced diet are the first rules to prevent atrial fibrillation. The physician's assessment of systolic and diastolic pressure is also of vital importance since very often patients with arterial hypertension cover a high percentage of cases of atrial fibrillation. [5]

2.1.3 Atrial flutter

Atrial flutter is much less common than atrial fibrillation, but the causes and hemodynamic consequences are similar. Typical atrial flutter is an arrhythmia due to a reentry circuit involving part of the right atrium. The atria depolarize at a rate of 250-350 beats / min (typically 300 beats / min). Since the atrioventricular node is usually unable to conduct at this rate, typically only half of the pulses are conducted, resulting in a regular ventricular rate of 150 beats / min. During this arrhythmia, the atria activate with a high frequency, generating an irregular and often rapid heartbeat (tachycardia). In most subjects, atrial flutter occurs in the presence of acquired heart disease (previous heart attack, heart failure, valvular disease, etc.), congenital heart disease corrected by cardiac surgery and in the presence of arterial hypertension. Also in this case the doctor, with the help of Cardiofilo has the ability to monitor the patient with the aim of preventing any secondary cardiovascular diseases. It can therefore monitor blood pressure, blood sugar, weight and frequency of physical activity. In addition to these data, the doctor always has the possibility of associating the observation of these data with the risk factors present in the patient's clinical history.[6]

2.2 Cardiofilo Web app

Cardiofilo web App was created to support the work of cardiologists, therefore the main actors who will use this application are the doctors, helped by the health staff (nurses, health workers).

The cardiologist will therefore be able to carry out the following actions neatly, each having a corresponding graphic interface:

- **LOG-IN:** as a first step, the doctor authenticates himself during the log-in phase with username and password: if this step is correctly done, then he will enter the app. If for example the password is forgotten, the doctor can recover it;
- **HOMEPAGE:** the first screen that will appear to the doctor will be the home page where there will be a list of patients (with their respective tax code, city and medical priority: white, green, yellow, red code). In addition, on the homepage there will be the "New patient" button with which the doctor can add a new patient to his list of patients;
- **PATIENT DASHBOARD:** if the doctor clicks on a patient's name, one of those present in the patient list, he will access the patient dashboard. In this screen there are various buttons that will allow the doctor to:
 - Access to patient's Overview (a page showing the trends over the days / months / years of some parameters such as: pressure, blood glucose level, weight and hours of sports activities);
 - Access to personal data;
 - Access to risk factors (a page where the doctor can track the particular risk factors of that particular patient such as smoking, hypertension, hyperglycaemia and possibly send an alert message to the patient);
 - Access to patient's status page (a page where some patient parameters are tracked such as: Body Mass Index, allergies, if there are active implanted devices and when the intervention was performed, previous acute coronary events, Ejection Fraction, Transcutaneous Oxygen Saturation (%));
 - Access to patient's History page (Anamnesis);
 - Access to patient's Therapy page;
 - Access to events recorded by the patient through the smartphone app;
 - Send a notification to the patient;
 - Access to attachments in the archive of that patient data;

- Reset the password of the patient;
 - Archive the patient;
 - Enter a new visit.
- **NEW VISIT:** In addition to having a view of the data previously saved in the application, the cardiologist will be able to make a new visit to the patient, in which he will have the opportunity to enter new data. The process of the new visit takes place in various steps:
 1. Entering data on the Risk Factor page;
 2. Entering data on the Status page;
 3. Entering data on the History page;
 4. Data entry on the Therapy page;

2.3 Future Design Improvements

The Abinsula Graphics team, located in Sassari, previously developed the entire graphics of the web and mobile platform (mockup).

Clearly these first results are not the definitive ones, therefore we have tried to improve the interface design of Cardiofilo Web App.

The design of the app must be:

- functional;
- appealing;
- intuitive and ease of use;
- serves to capture the attention of people, old and new users;
- innovative;
- innovative by exploiting new trends in the world of User Interaction & Design.

For example, the graphic interface design must ensure that every time the App shows a message (for example a pop-up) asking the user for an input (such as the request to enter the patient's systolic and diastolic pressure), all the necessary information is present, so that the user can make a decision on what he must or wants to do.

Otherwise the user will be disoriented and will not be able to make a quick decision. This is undoubtedly a friction in terms of Usability (degree of ease and satisfaction with which the interaction between man and an application takes place).

The design phase is therefore essential for the success of the application and a well-considered use of all the graphic components of the app is therefore important and to determine their usability are above all the position, size and structure of the application contents.

Figures 1.2-1.10 show the proposed changes for future improvements of the app.

Cardio Fito

WebPlatform

+ New visit

← Patient / Sara Malini / Risk Factor

Note: the fields are read-only, to change go from New Visit

Data Risk	Messages	Note
<div>Smoke</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Smoke</div> <div>Message alert</div>	<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</div>
<div>Hypertension</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Hypertension</div> <div>Message alert</div>	
<div>Dyslipidemia</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Dyslipidemia</div> <div>Message alert</div>	
<div>Diabetes</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Diabetes</div> <div>Message alert</div>	
<div>Familiarity with coronary heart disease</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Familiarity with coronary heart disease</div> <div>Message alert</div>	
<div>Previous chemotherapy</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Previous chemotherapy</div> <div>Message alert</div>	
<div>Previous radiotherapy</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Previous radiotherapy</div> <div>Message alert</div>	
<div>HIV in treatment</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>HIV in treatment</div> <div>Message alert</div>	
<div>Severe renal impairment</div> <div><input checked="" type="checkbox"/> yes <input type="checkbox"/> No</div>	<div>Severe renal impairment</div> <div></div>	

Figure 2.2: Original Screen of Risk Factor

Cardio Filo

WebPlatform

+ New visit

← Patient / Sara Malini / Risk Factor

Note: the fields are read-only, to change go from New Visit

DATA RISKS

◀

APRIL 2020

📅

▶

	yes	no		Note
Smoke	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Hypertension	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Diabetes	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Previous chemotherapy	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Severe renal impairment	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Familiarity with coronary heart <u>disease</u>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
HIV in treatment	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

Figure 2.3: Proposed Screen of Risk Factor

WebPlatform

+ New visit

Patient / Sara Malini / Risk Factor

Note: the fields are read-only, to change go from New Visit

DATA RISKS

APRIL 2020

Risk Factor	yes	no	Action	Note
Smoke	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	
Hypertension	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	We recommend that you quit smoking to avoid cardiovascular problems. Dott. Rossi <button>SEND</button>
Diabetes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	
Previous chemotherapy	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	
Severe renal impairment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	
Familiarity with coronary heart disease	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	
HIV in treatment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[Envelope icon]	

Figure 2.4: Proposed Screen of Risk Factor

Figures 2.3 and 2.4 show the proposed screens with improvements both graphically

and related to the management of the components on the page. In particular, a temporal reference has been added to the screens with the aim of giving the cardiologist a temporal reference to which we refer, thus giving the possibility to view the pages of the visits made in the previous days / weeks / months.

The time reference is also added in all the other interfaces that follow. Another improvement is related to the part dedicated to the sending of alert messages to the patient, the table of the various messages proposed in 2.2 has condensed, in a component that with its shape suggests intuitively to the user, to send alert messages to the patient.

Cardio Filo WebPlatform + New visit

← Patient / Sara Malini / History Note: the fields are read-only, to change go from New Visit

General History (Anamnesi generale)

Ipertensione arteriosa, fumo di sigarette (stop 2 mesi fa), dislipidemia, diatesi allergica (in particolare allo iodio), pregressa ablazione transcateretere di via accessoria tipo Kent, ernia iatale e pregressa ulcera peptica. Pregressa calcolosi renale.

Cardiovascular History (Anamnesi cardiologica)

IM infero-laterale del 04/10/2017 (PTCA primaria + stent medicato su Cx) seguito da intervento chirurgico di rivascolarizzazione miocardica (10/12/2017) (AMID su IVA, AMIS su MO) per coronaropatia trivale coinvolgente il tronco comune

Recent History (Anamnesi recente)

E. O. C.: toni ritmici, impurità sistolica puntale. M.V. su tutto l'ambito polmonare.
ECG: ritmo sinusale 75/min. PR nei limiti; tendenza ai bassi voltaggi del QRS nelle derivazioni periferiche. QTc nei limiti.
Rapida ecocopia: ipo-acinesia della parete inferiore (già presente) con lieve rigurgito mitralico; assenza di versamento pleurico.

Systolic Pressure mmHg **125** Diastolic Pressure mmHg **86**

Conclusions and considerations (Conclusioni)

Cardiopatia ischemica post-infartuale rivascolarizzata chirurgicamente, in fase di stabilità clinica.
Per la ridotta tolleranza agli idrati di carbonio, si raccomanda di eliminare dalla dieta gli zuccheri semplici:
fra 3 mesi controllare emoglobina glicosilata + peptide C.

Therapy (Terapia consigliata)

Si consiglia:

Pantorc 40 mg 1 cp al mattino (digiuno)
Bisoprololo 2.5 mg 1 cp ore 08, 1 cp ore 20
Procoralan 5 mg 1 cp ore 08, 1 cp ore 20
Plavix 75 mg 1 cp a cena da assumere fino al 01/02/2018

Figure 2.5: Original Screen of Anamnesi

Cardio

WebPlatform

+ New visit

← Patient / Sara Malini / History

APRIL 2020

Note: the fields are read-only, to change go from New Visit

General History

Arterial hypertension, cigarette smoke (stop 2 months ago), dyslipidemia, allergic diathesis (in particular iodine), previous ablation of trans accessory catheter of the kent type, jatal hernia.

Cardiovascular History

Infero-lateral IM of 10/10/2017 (primary PTCA + medicated stent on Cx) followed by myocardial revascularization surgery (10/12/2017)

Recent History

ECG: sinus rhythm 75 / min. PR within limits: QRS low voltage tendency in peripheral branches.

Systolic Pressure [mmHg] - range: 60/270

125

Diastolic Pressure [mmHg] - range: 30/120

130

Attention please: Value out of range

Conclusions

Revascularized post- infarct ischemic heart disease
Surgically, undergoing clinical stability. For reduced hydrate tolerance
Carbon, it is recommended to eliminate semplece sugars from the diet

Therapy

It is recommend:
- pantorc 40mg 1 cp in the morning (fasting)
- bisoprolol 2,5mg 1 cp at 8, 1 cp at 20
- plavix 75mg 1 cp at dinner to be taken until 1/12/2020

Send the prescription to the patient

SEND

Back

Save and go on

Figure 2.6: Proposed Screen of Anamnesi

Graphical improvements have been proposed in Figure 2.6 and options have also been added: the user can always save or delete what he has written in the various anamnesis, in addition, the doctor can send the recommended therapy to the patient in pdf format, so that the patient has a medical prescription to be displayed in the pharmacy if he intends to buy the recommended drugs . The doctor will also be able to print the final part of the anamnesis with the aim perhaps to expose the diagnosis quickly to other doctors who at that moment do not have a tablet / personal computer.

New Visit / Therapy
Exit without saving

Step 1: Anticoagulant Therapy

Atrial Fibrillation (AF)
Valvular AF
Implanted Mechanical Valve

WARFARIN DRUGS

Start Therapy
dd/mm/yyyy
End Therapy
dd/mm/yyyy
NEXT CHECK
dd/mm/yyyy

TREATMENT
3-4
CURRENT INR
4.5
Weekly dose (mg)
5mg
Time
10:00 am

Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

COUMADIN (Bing)
None
Half
A quarter
None

Add to therapy plan

Clicking on Add to therapy the drug will be shown in the left column

Step 2: Anti-platelet therapy

☐ TICAGRELO
☐ PRASUGREL
☐ CLOPIDOGREL
☐ CARBOCIPHEIN

Days of therapy
☒ Every day
☐ Monday
☐ Tuesday
☐ Wednesday
☐ Thursday
☐ Friday
☐ Saturday
☐ Sunday

Timetable
Time
Time
Time
Time
Dose/gg
Tada

☒ After meals
☐ Empty stomach
☐ As different

Add to therapy plan

Clicking on Add to therapy the drug will be shown in the left column

Step 3: Other Drugs

Please Type Drug name

Days of therapy
☒ Every day
☐ Monday
☐ Tuesday
☐ Wednesday
☐ Thursday
☐ Friday
☐ Saturday
☐ Sunday

Timetable
Time
Time
Time
Time
Dose/gg
Tada

☒ After meals
☐ Empty stomach
☐ As different

Add to therapy plan

Clicking on Add to therapy the drug will be shown in the left column

Back
Save and go on

Figure 2.7: Original Screen of Therapy

STEP 1:
ANTICOAGULANT
THERAPY

STEP 2:
ANTIPLATELET
THERAPY

STEP 3:
OTHER DRUGS

APRIL 2020

Exit without saving

Atrial fibrillation (AF):

Non Valvular AF

DOAC Drugs (Direct-acting oral anticoagulant)

Serum Creatinine (mg/dl)

30-200

Verapamil Therapy

NAO

DABIGATRAN

☒ APIXABAN

RIVAROXABAN

EDOXABAN

☒ Every day

☒ Monday
 ☐ Tuesday
 ☐ Wednesday
 ☐ Thursday
 ☐ Friday

☒ Saturday
 ☐ Sunday

Timetable

Time

Time

Time

Time

☐ After meals
 ☐ Empty stomach
 ☐ Indifferent

Add to therapy plan

Clicking on Add to therapy, the drugs will be shown in the left column

Prescribed drugs (List)

Nome farmaco

Nome farmaco

Nome farmaco

Figure 2.8: Proposed Screen of Therapy

STEP 1:
ANTICOAGULANT
THERAPY

STEP 2:
ANTIPLATELET
THERAPY

STEP 3:
OTHER DRUGS

APRIL 2020

Exit without saving

TICAGRELOR

PRASSUGREL

CLOPIDOGREL

CARDIOASPIRIN

Days of therapy

☒ Every day
 ☐ Monday
 ☐ Tuesday
 ☐ Wednesday
 ☐ Thursday
 ☐ Friday
 ☐ Saturday
 ☐ Sunday

Timetable

Time

Time

Time

Time

☒ After meals
 ☐ Empty stomach
 ☐ Indifferent

Dosaggio

Testo

Add to therapy plan

Clicking on Add to therapy, the drugs will be shown in the left column

Figure 2.9: Proposed Screen of Therapy

25

New Visit / Therapy
Exit without saving

STEP 1:
ANTICOAGULANT
THERAPY
STEP 2:
ANTIPLATELET
THERAPY
STEP 3:
OTHER DRUGS
APRIL 2020

Please Type Drug name

Days of therapy
☒ Every day
☐ Monday
☐ Tuesday
☐ Wednesday
☐ Thursday
☐ Friday
☐ Saturday
☐ Sunday

Timetable
Time
Time
Time
Time

Dosaggio
Testo

☒ After meals
☐ Empty stomach
☐ Indifferent

Add to therapy plan

Clicking on Add to therapy, the drugs will be shown in the left column

Back
Save and go on

Figure 2.10: Proposed Screen of Therapy

The changes proposed in the interface relating to the insertion of the therapeutic plan (Figures 2.8 - 2.9 - 2.10) by the cardiologist, are intended to make the interface more logical and intuitive.

The goal of the change is not to compress all three phases of the therapeutic plan into a single page, but to have three pages, one for each therapeutic phase. The pages follow one after the other, always leaving the doctor the opportunity to get an idea of his current position within the therapeutic plan.

The proposed improvements are intended to make the app easier to use for cardiologists and nurses.

Chapter 3

Frontend Development of Cardiofilo

After the creation of the mockups, illustrative and exhibition models of the application, the next step is the conversion from design mockup to HTML / CSS / JAVASCRIPT code: thus obtaining with the Front-End development of Cardiofilo web app.

The work of a Front-End developer (or Front-End, in the initials FE) consists in the design and creation of layouts, style sheets, interactions, animations, and optimizations on the navigation of the application through the creation of user interfaces. At the same time the FE work is coordinated with the Back-End work, from which it acquires the input data of the user interfaces and at the same time processes the data in ways that comply with predefined specifications, such as to make them usable later by the Back-End.

In the case of a web app, the term Front-End therefore means public part of the application (the part accessible to the user), while the Back-End refers to the part of the application not accessible by the users but only app administrators can have access and through which it is possible to enter new data (new sections, new posts, etc.) and at the same time acquire new data entered in the interfaces. Through the Front-End work, the doctor who visits Cardiofilo web app will view the so-called "user interfaces" (UI), designed and built to be compatible with the needs and possible interactions of any average user with the app pages. It is therefore essential to analyze the needs and actions of users in order to create the correct user interfaces.

The front-end development is therefore essential in the entire world of the web, as it is necessary for the creation of a well-organized structure and to make the use of the UI by the user all in a fluid, easy and intuitive way.

It is possible to observe the following advantages related to front-end services such as:

- **Rapid development and user-driven results**

The use of some modern frameworks like React will allow a faster development of

the components of the UI;

- **Protected**

The framework in the front-end development contains a fully protected encryption. This is an advantage why never have to worry about site working on any existing browser;

- **Features and apps that react quickly**

The latest frameworks and technologies used by developers allow for fast response features and structures that finally power the app to respond, react and work fast;

- **Technologies that are easy to learn, use and scale**

Most frameworks and technologies feature very intuitive building levels that are very simple to both follow and learn;

- **Powerful features and layout**

The frameworks used by the front-end developers will have the advantage of developing a powerful website thanks to its predefined styles ensuring the robustness of its functionality;

- **Real-time programming**

The developer can observe the changes in the browser without any fear of losing the app state and reloading the browser pages.

However, there are limitations in the front-end development to which the developer is subjected such as:

- **Not on par with compute heavy back-end systems**

Apps often involve heavy calculations with tons of data transits happening every minute and therefore few front-end / back-end technologies fail to match. The solution to this problem can be to dismantle complex situations into small organized self-dependent modules, which contain within them more suitable technologies that interact quickly with the back-end in case of heavy calculations;

- **Custom development**

Every app has at least some elements that need customization if not most of them. Consequently, for the creation of customized features, it is necessary to rely on certain frameworks and therefore choose the most appropriate framework for the front-end development;

- **New releases**

Whenever there is any new update in the app, it must always evaluate if it is possible

to get the latest update, if not, it will lack compliance with modern standards. Experienced developers can therefore overcome this limitation by staying in constant contact with changes and updates.

3.1 Front-End development tools

To create web pages, a Front-End developer essentially uses three programming languages:

- **HTML:** to create the structure and content.

In computer science, HyperText Markup Language (HTML) is a markup language. It was born for the formatting and pagination of hypertext documents available on the web and is a public domain language, whose syntax is established by the World Wide Web Consortium (W3C). It derives from SGML, a metalanguage aimed at defining languages that can be used for drafting documents intended for transmission in electronic format.

An HTML element in a document is distinguished from the rest of the text by "tags", which consists of the element name surrounded by "<" and ">". The name of an element within a tag is case insensitive.

HTML uses "markup" to annotate text, images, and other content for display in a web browser. HTML markup includes special "elements" such as <head>, <title>, <body>, <header>, <section>, <p>, <div>, and many more.[7]

Example for creating a simple web page in HTML:

Code written with a text editor (the file has an HTML extension '.html'):

```
<!doctype html>
<html lang="it">
<head><title>Cardiofilo Web App</title></head>
<body>
<h1>Cardiofilo Web App</h1>
<img src='logo.svg'>
<p>This is the first HTML page!</p>
</body>
</html>
```


Clicking from the file manager on a file with an HTML extension will automatically open the system default browser, which shows a screen similar to that in Figure 3.1:



Figure 3.1: Example of a simple HTML page

From this HTML file it is possible to extrapolate a classic example of a DOM (Document Object Model) tree, generated by the interpretation of the file by the web browser (see Figure 3.2).

The DOM, as the word itself says (Document Object Model) is a model, or a way of representing a document or an interface, or rather an API (Application Programming Interface) designed by the W3C (World Wide Web Consortium) consortium, which allows you to access the elements of a page.

It should be noted that in this example also the so-called blank characters (spaces, tabs, etc.) are considered elements of the tree, although these are not displayed by the browser.

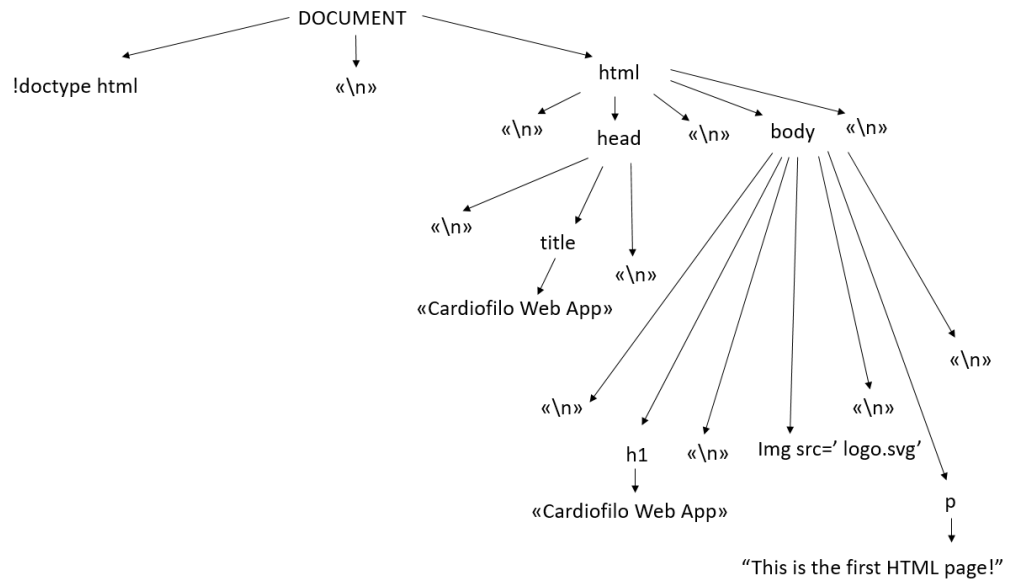


Figure 3.2: Example of DOM tree generated by the web browser in the interpretation of the HTML file

- **CSS:** to equip the page with colors, style, fonts and background images.

CSS (abbreviation of Cascading Style Sheets), in computer science, is a language used to define the formatting of HTML and XML documents. The introduction of CSS has become necessary to separate the contents of HTML pages from their layout or style. [8]

Example for creating a simple web page in HTML with CSS style:

Code written with a text editor (the file has an HTML extension '.html'):

```
<!doctype html>
<html lang="it">
<head><title>Cardiofilo Web App</title>
<style type="text/css">
body
color: #48ccbf;
background-color: #ffffff ;
text-align: center;
font-family: Arial, Helvetica, sans-serif
</style>
</head>
<body>
<h1>Cardiofilo Web App</h1>
<img src='logo.svg'>
<p>This is the first HTML page!</p>
</body>
</html>
```

Clicking on a file with an HTML extension will automatically open the system default browser shows a screen similar to that in Figure 3.3:

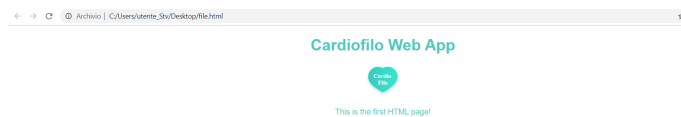


Figure 3.3: Example of a simple HTML page with CSS style

- **JAVASCRIPT:** to give the page greater dynamism and interaction possibilities.

JavaScript was first standardized in 1997 by ECMA under the name ECMAScript. The latest standard, as of June 2017, is ECMA-262 Edition.

In computing, JavaScript is an object and event-oriented programming language, commonly used in client-side Web programming for the creation, in websites and web applications, of interactive dynamic effects through invoked script functions by events triggered in turn in various ways by the user on the web page in use (mouse, keyboard, page load). These script functions, therefore used in the presentation logic, can be suitably inserted in HTML files or in special separate files with the ".js" extension.

The files however may have both '.js' and '.jsx' extension, the difference between the two is the transpiler / bundler, which may not be configured to work with JSX files, but with JS! So very often you are forced to use JS files instead of JSX. In particular, it is already possible to anticipate that for the React framework it makes no difference to choose between JSX or JS. They are completely interchangeable.

In some cases, users / developers might also choose JSX over JS, due to code highlighting, but most newer editors also correctly display reaction syntax in JS files.

JavaScript is a simple programming language, particularly suitable for use by the browsers.

Complications arise when using an interface for the site, namely the DOM (Document Object Model) which relies on the frameworks and JavaScript libraries, aimed at facilitating the task of the developers. JavaScript libraries are reusable codes. Specific properties and functions are designed for the website. The most famous JavaScript libraries are jQuery, React, Zepto etc. [9]

Example for creating a simple web page in HTML with CSS style and Javascript syntax:

In this example, the page contains a text type input field (which is generally used by the user to write data).

To know the length of the text contained in the input form, the object model and the properties used by JavaScript are used.

Code written with a text editor (the file has an HTML extension '.html'):

```
<!doctype html>
<html lang="it">
<head><title>Cardiofilo Web App</title>
<style type="text/css">
body
color: #48ccbf;
background-color: #ffffff ;
text-align: center;
font-family: Arial, Helvetica, sans-serif
</style>
</head>
<h1>Cardiofilo Web App</h1>
<img src='logo.svg'>
<p>This is the first HTML page!</p>
<body onLoad="alert(document.Form.test.value.length)">
<form name="Form">
<input type="text" name="test" value="Cardiofilo">
</form>
</body>
</html>
```

Clicking on a file with an HTML extension will automatically open the system default browser and show a screen similar to in that figures 3.4-3.5:

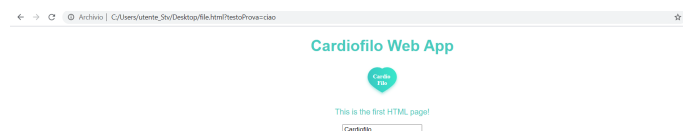


Figure 3.4: Example of a simple HTML page with CSS style and Javascript syntax



Figure 3.5: Example of a simple HTML page with CSS style and Javascript syntax

3.2 Framework for user interface creation

React is the first JavaScript library born with a specific objective: to become the definitive solution for frontend and mobile app developers based on HTML5. It is used to build dynamic and increasingly complex user interfaces while remaining simple and intuitive to use.

The creation of Web applications, regardless of the framework chosen for development, necessarily involves the three fundamental languages of the platform: HTML for the structure, CSS for the stylization and JavaScript for the application logic.

For most existing Javascript libraries including React, the HTML language specifically is used almost exclusively to create reusable "web components". The power of React compared to other libraries is to allow the use of a declarative approach similar to HTML, to define the components that represent significant and logical parts of the user interface.

React was created to manage the presentation and interception of user input events, without forcing the adoption of specific functions and avoiding interfacing with other libraries as regards any communication with a backend server, or specific data binding architectures.

There are however many other libraries dedicated to the development of web applications. The comparison between React and other popular libraries such as JQuery and AngularJS will allow us to understand the decision to develop the user interfaces of the Cardiofilo web app using React.

- **JQuery:** it is a very convenient tool in the development of applications by directly manipulating the page structure. Thanks to functions, it is possible to access the DOM (in computer science the Document Object Model) and to modify elements, to create new ones, move existing ones. JQuery works mainly using the syntax of CSS selection expressions: this assumes that in most cases it is necessary to assign an ID, or a specific class, to the elements of the page, in order to reference them later within the script that contains the client-side logic of the application. This procedure is very simple to use, since it is sufficient to know the CSS expressions

and the basic functions of JQuery, in addition to the basic syntax of JavaScript, to write the logic.

Compared to JQuery, React does not require the mandatory assignment of unique IDs or classes, nor does it require the developer to intervene directly on the DOM, but it is the library itself that takes on this task based on the structure of the component declared in the first place and, secondly, based on the internal state of the Web component and the variation in the values of the properties assigned to it. In short, React excludes direct intervention on the DOM, leaving us the task of defining the components that make up the interface of the application and the data to be used for generating the markup. The library will intervene on the DOM accordingly, using the method that guarantees the best possible performance.[10]

- **AngularJS:** is the framework created by Google for the development of Web applications. AngularJS takes full advantage of the Model View Controller (MVC) paradigm: unlike JQuery, the user interface is totally separate from the model (the object that contains the data to represent) and also from the business logic (the application logic, implemented through services) while the Controller acts as a "glue" between all these elements by managing user input and modifying the data model, through business logic, which involves dynamic updating of the HTML view.

The problems related to the architecture of the application that are encountered with JQuery thus disappear with AngularJS, which allows you to divide all the components of the application (views, controllers, services, etc.) into separate modules. However, even complex functions can be easily obtained on small projects, but as complexity increases, the entire development team working on the project must be equipped with the knowledge necessary to implement what is needed, following precise dictates that allow the integration of the elements of the application with the infrastructure provided by AngularJS, under penalty of renouncing the advantages that the framework is able to guarantee. The goal that React has is completely identical to that of AngularJS: to untie the developer from the need to manage the DOM directly.

However, React focuses on the visual part of the application by dividing it into blocks (components) that contain its own state and its management logic, while communication with the server or the structuring of the modules are delegated to external libraries.

React is more focused and accessible to learning, with the possibility of resorting to known libraries (e.g. also the same JQuery) for the parts that are not strictly related to the user interface.[11]

The comparison ends by favoring the use of the React library since it manages to create insulated components and components to create complex UIs (User Interfaces). Furthermore, considering that the interactions and the logic between the components are implemented in JavaScript, it is easy to pass and access complex data structures in various points of the web application without having to save information on the DOM.

React, therefore, creates interactive UI, designs interfaces for each state of the application and at each change of state React efficiently updates only the parts of the UI that depend on such data.

The declarative nature of the UI makes the application code more predictable and easier to debug.

3.2.1 React

React (also known as React.js or ReactJS) is a JavaScript library for creating user interfaces. It can be used as a basis in the development of single page or mobile applications.

In an MVC context (Model-View-Controller, an architectural pattern capable of separating the presentation logic of the data from the business logic) React therefore acts as a support for the creation of views, possibly generated from a data model. The data model consists of React components that are used to create the DOM elements, which constitute the interface, based on the data contained in JavaScript objects.

The components are self-sufficient, independent micro-entities that describe a part of the user interface. An application's user interface can be divided into smaller components, where each component has its own code, structure and API (Application Programming Interface). The component architecture allows you to consider each piece in isolation. Each component can update everything within its reach without being concerned about how it affects the other components. The components are also reusable and need data to work on. There are two different ways to combine components and data (see 3.6): either props or state. Props and state determine what it is the behaviour of the component renders.[12]

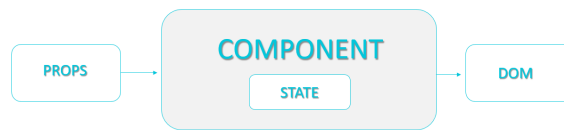


Figure 3.6: General scheme relating to the operation of React

Props:

Props (short of properties): constitute immutable and top-down values for which no alteration is foreseen, useful for example to configure the component or to indicate default values for the status. This means that a parent component can pass any type of data it wants to its children as props, but the child component cannot modify its props.

A virtually infinite number of properties can be passed to a React component; in the component code it will be possible to read their values at any time by accessing the props object, and it will not be possible to modify them.

State:

It is a set of data that, if modified, results in an update of the component itself. Its scope is limited to the current component. A component can initialize its state and update it whenever necessary. The state of the parent component usually ends up being the props of the child component. When the state is passed outside the current scope, we refer to it as props.

The choice of passing information to the components in the form of properties or placing them in the state depends first of all on the variability expected for the information and, more generally, on the conceptual design on which one relies on to create one's own Web application.

A React component can be of two types: a class component or a functional component.

Functional components:

Functional components are just JavaScript functions. They take an optional input which is a props.

Class components:

Class components offer multiple functions. The main reason for choosing class components over functional components is that they can have states.

There are two ways in which you can create a class component. The traditional way is to use `React.createClass()`.

React event management:

Event management allows to make the React component interactive, responding to user actions and updating the graphic interface accordingly, acting on the state of the component itself.

The state is represented by an object accessible in the component code through the state property which can be initialized as desired; unlike what happens with the properties, provided by the props object, the status can be changed by calling the `setState ()` method. Following a change in status, generally through the management of an event that occurs with a user action, React invokes the `render ()` function and verifies the need to make changes to the part of the DOM that constitutes the representation of the component within the web page.

The `getInitialState ()` function is responsible for returning a JavaScript object that contains the initial state of the component; in the object it is possible to include all the properties we want according to the requirements of the desired components.

Virtual DOM:

Using React it is not necessary to use the DOM functions to modify the elements of the page: it is sufficient to change the information contained within the object representing the data model, represented by the state (`state`) in React. React uses a different system, from the other Javascript libraries, to track the changes, using the so-called Virtual DOM, that is to say a virtual representation of the structure of the page stored in memory and very similar to the original DOM, of which it can be seen as an abstraction. When an event occurs and it is necessary to "react" to it by changing the elements of the page, React first applies these actions to the Virtual DOM. By analyzing the differences between the status of the Virtual DOM prior to the occurrence of the event and the new one obtained by applying the changes, React determines the actual changes to be made to the real DOM. The calculation of the differences between the two states of the virtual DOM is extremely fast, and thanks to it the interventions on the real DOM, tendentially slower, are limited to the bare minimum, thus guaranteeing excellent performance.

In the Virtual DOM there are both complex components like the classic elements of the HTML standard, such as a simple `<div>`. The management of the Virtual DOM and its transformations is completely the responsibility of React: the developer remains completely transparent and no use is required API (Application Programming Interface) for its manipulation, as happens in the direct use of the real DOM, which should not be used. The abstraction represented by the Virtual DOM also allows to take advantage of a language independent of the browser, in which it is possible to express the characteristics of the component related to its appearance and behavior regardless of the browser in which the application will be hosted, leaving React the task of translating the developer's

intentions into calls to JavaScript functions - both global and DOM functions - according to their actual availability (feature detection) within the browser of the user.

Finally, there are also server-side implementations of React's Virtual DOM which, by using the same functions and thus writing code completely analogous to the client version, allow to return HTML pages generated directly by the server. To conclude, Virtual DOM is therefore of fundamental importance in the functioning of React.

Use of Node.JS with React:

Node.js is a runtime for the execution of JavaScript code based on the V8 engine, the same on which the Google Chrome JavaScript interpreter is based, famous in particular for its performance. It is an open source product available for all popular operating systems and can be downloaded for free. In this case, Node.js is used locally, to enhance the development environment with the aim of producing and optimizing HTML pages, style sheets and JavaScript scripts which can then be hosted on any type of server (or hosting). The installation of Node.js aims to create a development environment consisting of different tools and services written in JavaScript and designed to work thanks to the Node runtime.

Node.js, therefore, becomes a fundamental component to support the development phase of the web application. Thanks to the support of Node and its package manager npm, it is possible to install additional packages and import the Babel transpiler to read files with JS syntax and create standard JavaScript files, interpretable by any browser, taking advantage of the features of the most modern versions of the language such as ES6.

3.3 Front-End development of Cardiofilo

React has therefore been used in the Cardiofilo project as a JavaScript library for the front-end development of dynamic user interfaces.

Scripts were edited using Visual Studio Code source code editor, developed by Microsoft for Windows, Linux and macOS.

In this code editor you can open an integrated terminal (see Fig.3.7).

The terminal is a command line tool that is generally available on most operating systems and is used to execute commands entered by the user. With the use of these commands it is possible to automate tasks through scripts and batch files, perform advanced administrative functions and solve certain types of problems or bring a package manager able to download and install in complete autonomy a series of packages that help us in work and front-end development. For example after installing Node.js from the terminal, you can fully use all the tools offered.

The development of the front-end starts from the initialization and construction of a new React project and to do so the "create-react-app" function provided by the npm package provided by node.js is used.

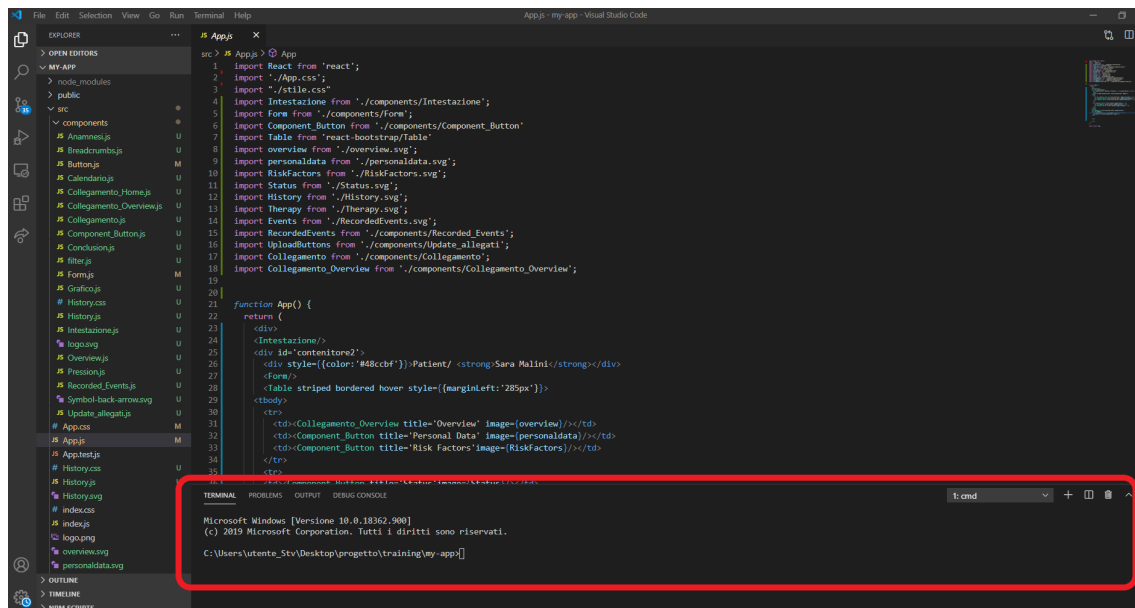


Figure 3.7: Script and the terminal integrated into Visual Studio Code

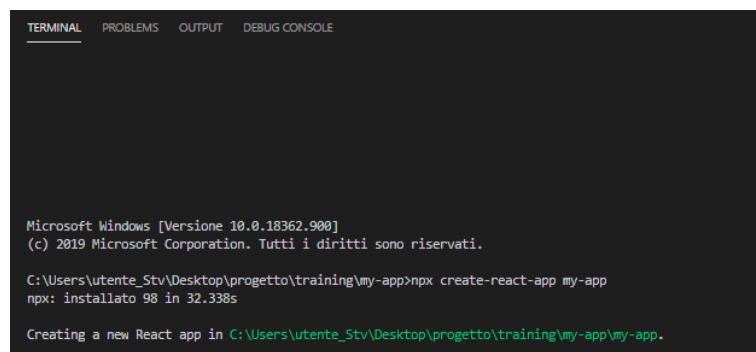
3.3.1 Create a new project with function "Create React App"

"Create React App" is the best way to start building a new single-page application in React and includes all the JavaScript packages needed to run a React project, including code transpilation, basic linting, testing and building systems.

It also includes a hot reloading server that will update the page as you make code changes. Finally, it will create a structure for the directories and components so that you can start coding in minutes.

However, it will be necessary to have Node.js installed in the development environment used, in order to create a new app using the npm package manager which will install the JavaScript packages in the project.

Visual Studio Code npm terminal executable package allows you to install and manage packages for use in Node.js applications, in this case the command (*npx create-react-app my-app*) will install create-react-app in the specified directory (see Fig.3.8).



```
Microsoft Windows [Versione 10.0.18362.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\utente_Stv\Desktop\progetto\training\my-app>npx create-react-app my-app
npx: installato 98 in 32.338s

Creating a new React app in C:\Users\utente_Stv\Desktop\progetto\training\my-app\my-app.
```

Figure 3.8: Installation through node.js npm package and creation of a new React project folder

So the command *npx create-react-app* creates a new project in a directory and it's the best way to start building a new single-page application in React. The following modules are present within the new project created (see Fig.3.9):

- `node_modules` contains all external JavaScript libraries used by the application. You will rarely need to open this form.
- The `public` directory contains some basic HTML, JSON and image files. These are the roots of the project.
- The `src` directory contains the project's React JavaScript code.
- The `.gitignore` file contains some default directories and files that git (source control) will ignore, such as the `node_modules` directory. Ignored items tend to be larger

directories or log files that would not be needed in source control.

- README.md is a markdown file that contains a lot of useful information about Create React App, such as a summary of commands and links to advanced configuration.
- The last two files are used by the package manager. When the initial npm command was run, the base project was created, but the additional dependencies were also installed with the associated creation of a package-lock.json file. This file is used by npm to ensure that packages match exact versions.
- The last file is package.json. This contains project metadata, such as title, version number and dependencies.

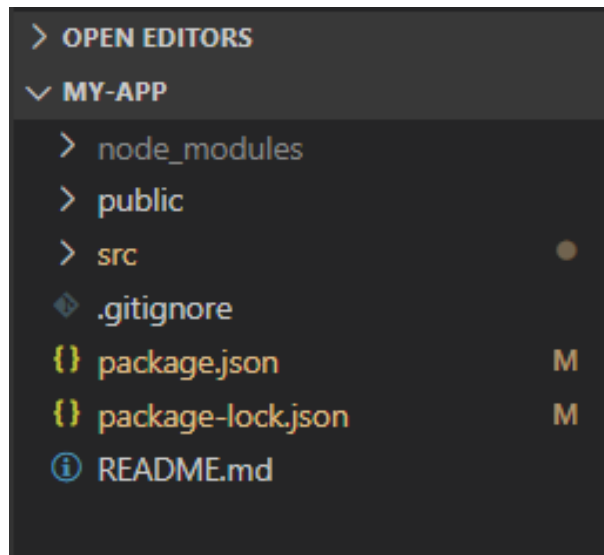


Figure 3.9: Folders present in the created "my-app" project

The structure of the project is in principle except for the content of src which contains the main files of the application that will be created.

The src folder contains within it (see Figure 3.10):

- The components folder: contains the components relating to the application views. For example, the "Intestazione.js" component represents the header present in each Cardiofilo Web App interface. The creation of a component has the purpose of creating an element present in the view which, if it is common with other UI, can be reused in the various interfaces , thus avoiding rewriting the same lines of code in

the various interfaces in the app. Each component has its own state and props. Of course, application UIs are dynamic and change over time. The state allows React components to change their output over time as a result of user actions, responses from the network (API) and anything else that can cause a different output to be rendered from time to time.

- The various files with the extension '.png' or '.svg' representing the icons on Cardiofilo Web App, these have been imported from the mockups previously built by the company Abinsula S.r.l.
- The style.css file: The CSS file contains the layouts and styles associated with the elements of each individual interface to which the IDs are assigned, so that they can be referred to later in the script.
- The App.js file: contains the declaration of the class that represents the main component of the application: in this file there is the construction of the home page of the various UI, there is the page with the dashboard related to a specific patient. This screen is the result of the doctor's log-in to Cardiofilo Web App with the subsequent selection of a specific patient in the app. The file is linked to its relative css file with the description of the style of the file.
- App.test: Contains a test that makes sure that a component was not launched during rendering. Tests like this provide a lot of value with minimal effort, so they're great as a starting point.
- Index.js: contains the ReactDOM.render() function which has the purpose of rendering a React element in the 'root' DOM node. React elements are simple objects and therefore faster to create. The React DOM takes care to update the browser DOM to be consistent with React elements. It is called the "root" DOM node as everything inside will be managed by the React DOM. Applications built only with React usually have only one root DOM node. If, on the other hand, you integrate React into existing apps, you could have multiple isolated DOM 'root' elements. The file has its associated css style file.
- ServiceWorker: This code is optional and is used to register a service worker. The Service Worker is a method that allows Web Applications to take advantage of persistent background processes. The Service Worker through an event-driven system responds to events sent to him from the web page (DOM) or from other resources.

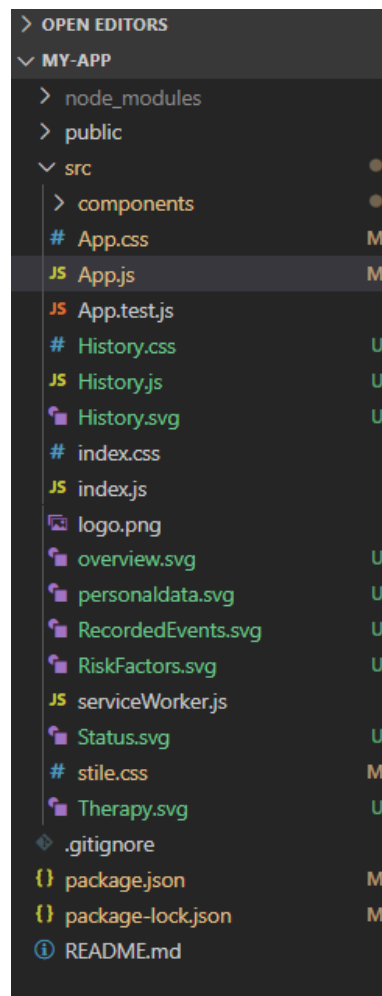


Figure 3.10: Contents of src folder

3.3.2 Components present in Cardiofilo Web App

As already mentioned, the 'App.js' file contains the main component of the built application, namely the dashboard connected to the individual patient selected by the doctor in the Cardiofilo Web app.

The project previously started by Abinsula, considered a log-in screen to which the doctor had to connect with his email and password and then the doctor found a list of the patients he is responsible for monitoring. By clicking on the individual patient, the doctor has access to the dashboard which will provide all the features designed to better monitor a patient suffering from ongoing or previously treated cardiovascular diseases.

The Front-End development, made entirely with React, starts from the individual patient's dashboard which will therefore identify the home screen in 'App.js'. (see Figure 3.11)

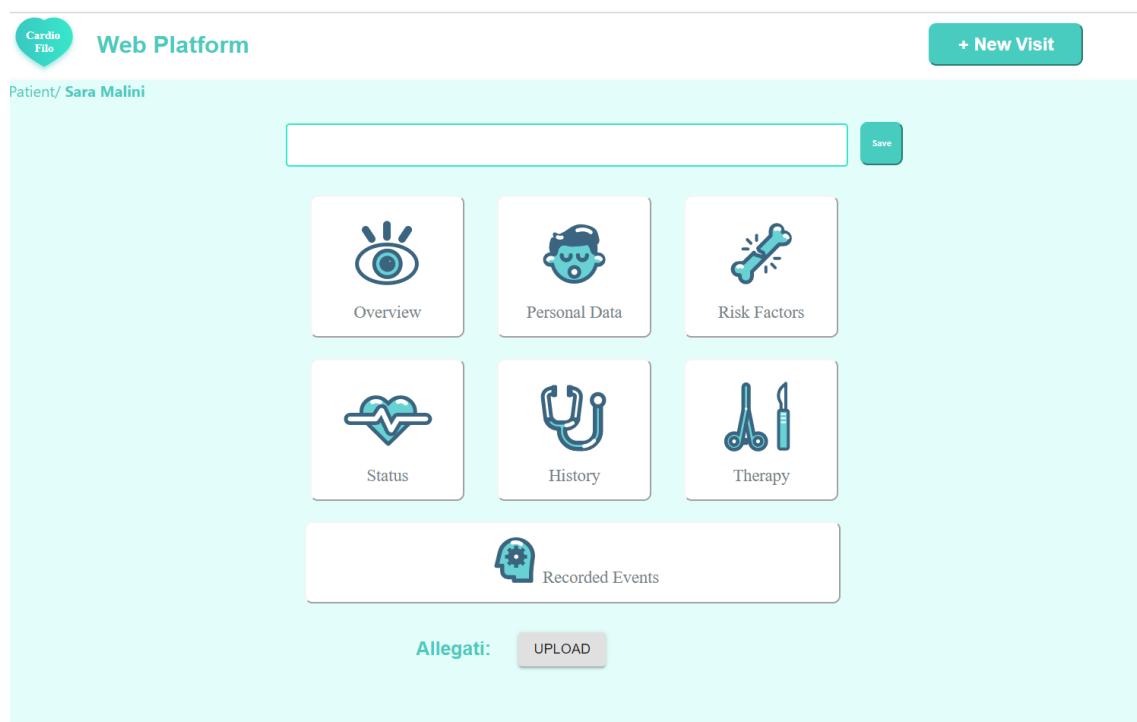


Figure 3.11: Contents present in App.js

The first part of 'App.js' script file is dedicated to imports (see in Fig.3.13 lines of the script from 1 to 18). Using this keyword, it is possible to import external elements within the current script, such as external resources, CSS style sheets or images and the insertion of instructions that allow us to import the fundamental types of React such as:

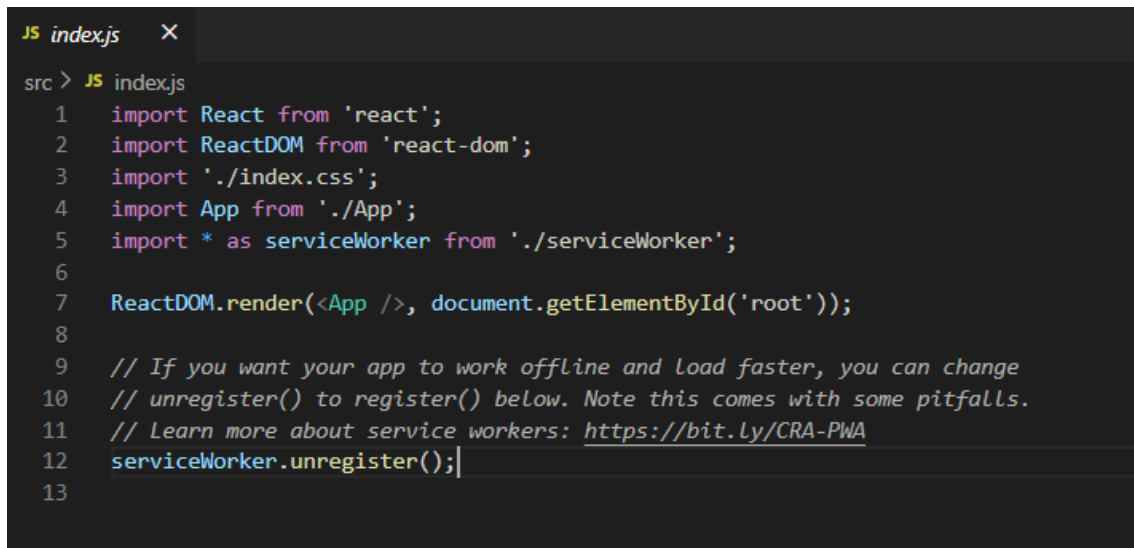
import React, Component from 'react'

The central part of the 'App.js' script file contains the definition of the main screen. Therefore a general functional component will be created (with the *function* method) that is called *App* and it will contain various components inside it (for example Intestazione, Form, Table, RecordedEvents, etc...).

Each component in *App* can be seen in green in the Figure 3.13 and each of them can be found in a <div> tag which has the task of defining the section of a specific element within the functional component.

The series of components that make up the are conceptually Javascript functions and allow to divide the UI of the project into smaller and smaller pieces. They take arbitrary data as input (under the name of "props") and return React elements that describe what should appear on the screen. The main reason a user interface is created through small pieces is the concept of re-usability and independence. In fact, the aim is precisely to create reusable components from larger components and so on.

Finally the functional component *App* will be exported outside the module so that it can be recalled in the 'index.js' file which implements the *render()* method, which renders a React element in the DOM (with the function *ReactDOM.render()*) within the container provided as input and returns a reference to the component (see Fig.3.12).

A screenshot of a code editor window with a dark theme. The title bar at the top shows 'JS index.js' and a close button. The editor content shows a JavaScript file named 'index.js' with the following code: Line 1: 'import React from 'react';', Line 2: 'import ReactDOM from 'react-dom';', Line 3: 'import './index.css';', Line 4: 'import App from './App';', Line 5: 'import * as serviceWorker from './serviceWorker';', Line 6: (empty), Line 7: 'ReactDOM.render(<App />, document.getElementById('root'));', Line 8: (empty), Line 9: '// If you want your app to work offline and load faster, you can change', Line 10: '// unregister() to register() below. Note this comes with some pitfalls.', Line 11: '// Learn more about service workers: <https://bit.ly/CRA-PWA>', Line 12: 'serviceWorker.unregister();|', Line 13: (empty). The cursor is at the end of line 12.

```
JS index.js X
src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import * as serviceWorker from './serviceWorker';
6
7  ReactDOM.render(<App />, document.getElementById('root'));
8
9  // If you want your app to work offline and load faster, you can change
10 // unregister() to register() below. Note this comes with some pitfalls.
11 // Learn more about service workers: https://bit.ly/CRA-PWA
12 serviceWorker.unregister();|
13
```

Figure 3.12: Index.js script

If the React element was previously rendered in the container, an update of the element will be performed and only the necessary DOM will be modified to reflect the last React element.

The main screen has been built in such a way as to be user-friendly or easy to understand by the end user who will use the app and at the same time the creation of the page was done by assigning a specific priority to each content present, determining how much space to allocate to a particular element and where to place that element on the screen.

```

JS App.js X
src > JS App.js > App
1  import React from 'react';
2  import './App.css';
3  import './stile.css'
4  import Intestazione from './components/Intestazione';
5  import Form from './components/Form';
6  import Component_Button from './components/Component_Button'
7  import Table from 'react-bootstrap/Table'
8  import overview from './overview.svg';
9  import personaldata from './personaldata.svg';
10 import RiskFactors from './RiskFactors.svg';
11 import Status from './Status.svg';
12 import History from './History.svg';
13 import Therapy from './Therapy.svg';
14 import Events from './RecordedEvents.svg';
15 import RecordedEvents from './components/Recorded_Events';
16 import UploadButtons from './components/Update_allegati';
17 import Collegamento from './components/Collegamento';
18 import Collegamento_Overview from './components/Collegamento_Overview';
19
20
21 function App() {
22   return (
23     <div>
24       <Intestazione/>
25       <div id='contenitore2'>
26         <div style={{color:'#48ccbf'}}>Patient/ <strong>Sara Malini</strong></div>
27         <Form/>
28         <Table striped bordered hover style={{marginLeft:'285px'}}>
29           <tbody>
30             <tr>
31               <td><Collegamento_Overview title='Overview' image={overview}/></td>
32               <td><Component_Button title='Personal Data' image={personaldata}/></td>
33               <td><Component_Button title='Risk Factors' image={RiskFactors}/></td>
34             </tr>
35             <tr>
36               <td><Component_Button title='Status' image={Status}/></td>
37               <td><Collegamento title='History' image={History}/></td>
38               <td><Component_Button title='Therapy' image={Therapy}/></td>
39             </tr>
40           </tbody>
41         </Table>
42         <RecordedEvents title='Recorded Events' image={Events}/>
43         <div id='Update'>
44           <div id='Allegati'><strong>Allegati:</strong></div>
45           <UploadButtons style={{marginLeft:'316px'}}/>
46         </div>
47
48       </div>
49
50     </div>
51   );
52 };
53
54
55 export default App;

```

Figure 3.13: Creating a function component in App.js

This function is a valid React component in that it accepts a parameter object containing data in the form of a single "props" (named after "properties") which is a parameter object with data in it and returns a React element. These types of components are referred to as "function components" because they are literally JavaScript functions. But it is also possible to use a *class* method to define a component (can see an example of a component created with the *class* method in figure 3.14).



```
JS Intestazione.js X
src > components > JS Intestazione.js > ...
1  import React from 'react';
2  import "../stile.css"
3  import logo from './logo.svg';
4  import Button from './Button'
5
6  class Intestazione extends React.Component {
7    constructor(props) {
8      super(props);
9      this.state = {
10        stato: 'Benvenuto!'
11      };
12    }
13    render() {
14      return(
15        <div id="intestazione">
16          <div id="logo">
17            <img src={logo} alt="logo" />
18          </div>
19          <div id="text">Web Platform</div>
20          <Button id="NewVisit" title='+ New Visit' />
21        </div>
22      );
23    }
24  }
25
26  export default Intestazione;
27
28
29
30
```

Figure 3.14: Creating a class component in file Intestazione.js

The construction of two components just seen are equivalent from React's point of view.

The *App* component was built with the function method and turns out to be a functional component while the *Intestazione* component was built with the class method and turns out to be a class component.

The dashboard is therefore composed of a series of components, each of which has been designed to be reused according to the property declared at the time of use of the particular component.

The following components are included in the construction of the *App* functional component with the respective functions and reasons for which they were created:

- The class component *Intestazione* present in each screen of Cardiofilo web app (see Fig.3.15), in which there is the logo that identifies the application and another class component calls *Button* that allows the doctor to start a new visit to the given patient. As soon as the doctor clicks on the "New Visit" button, an alert message will be displayed that will notify the doctor that a new visit is about to be entered (see the script in figure 3.14 of *Intestazione.js* and in Figure 3.16 of *Button.js*)



Figure 3.15: Intestazione component

```

JS Button.js  X
src > components > JS Button.js > Button > message_to_users
1  import React from 'react';
2  import "../style.css"
3
4  //Sto usando una class component//
5  class Button extends React.Component {
6    constructor(props) {
7      super(props);
8      this.message_to_users = this.message_to_users.bind(this);
9    }
10   message_to_users() {
11     alert("Doctor inserts new visit");
12   }
13
14   render() {
15     return(
16       <div>
17         <button id='NewVisit' onClick={this.message_to_users}>{this.props.title}</button>
18       </div>
19     );
20   }
21 }
22
23
24 export default Button;
25

```

Figure 3.16: Creating a class component in file Button.js

- After entering the path describing what folder we are in and in correspondence with which patient, the *Form* component (see Fig.3.17) is inserted in which the doctor has the possibility to insert any notes to remember (see the script in Fig.3.18)

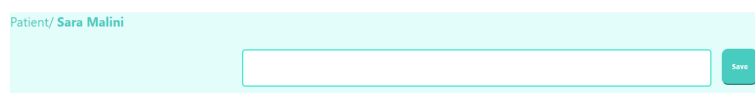


Figure 3.17: Path of Patient and Form component

```

JS Form.js x
src > components > JS Form.js > ...
1  import React from 'react';
2  import '../style.css'
3
4  class FormNome extends React.Component {
5    constructor(props) {
6      super(props);
7      this.state = {value: ''};
8
9      this.handleChange = this.handleChange.bind(this);
10     this.handleSubmit = this.handleSubmit.bind(this);
11   }
12
13   handleChange(event) {
14     this.setState({value: event.target.value});
15   }
16
17   handleSubmit(event) {
18     alert('E' stata salvata la seguente nota: ' + this.state.value);
19     event.preventDefault();
20   }
21
22   render() {
23     return (
24       <form onSubmit={this.handleSubmit}>
25         <input id="Form" type="text" value={this.state.value} onChange={this.handleChange} />
26         <input id="submit" type="submit" value="Save" />
27       </form>
28     );
29   }
30 }
31
32 export default FormNome;

```

Figure 3.18: Creating a class component in file Form.js

- Subsequently, with the *Table* component (imported from React-bootstrap, Bootstrap is a collection of free tools for creating Web sites and applications) a table was created in which there are the various buttons, each of which related to the particular functionality in Cardiofilo web app, like (see Fig.3.28):

– OVERVIEW (with summary graphs that describe the trends of certain values measured on the patient over time).

See script in Fig.3.24.

By clicking on the OVERVIEW button (see script of the link between the Home screen and the Overview screen in Fig.3.20, a new interface opens (see Fig 3.19)

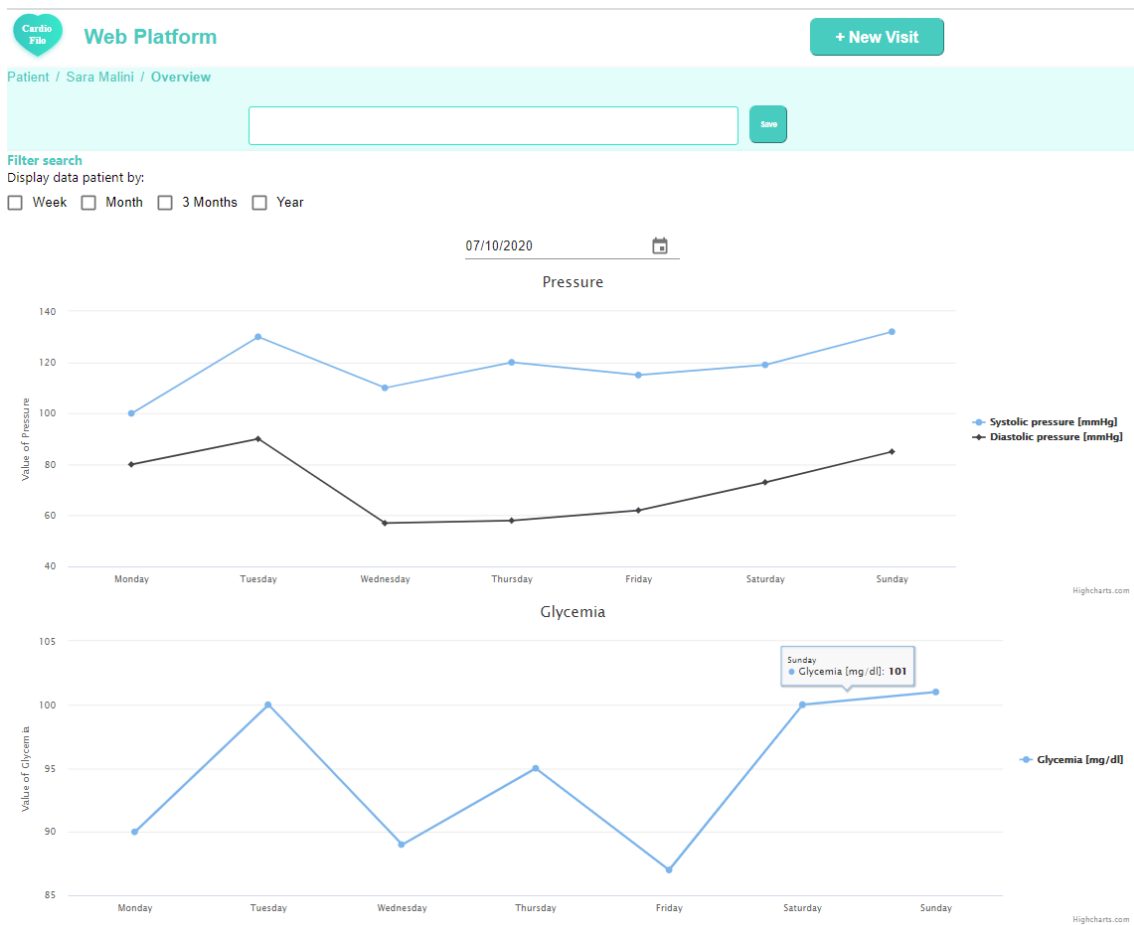


Figure 3.19: Overview interface


```

JS Collegamento_Overview.js
src > components > JS Collegamento_Overview.js > Collegamento_Overview > constructor
1  import React from 'react';
2  import "../style.css"
3  import Overview from '../Overview'
4  import ReactDOM from 'react-dom';
5
6
7  //Sto usando una class component//
8  class Collegamento_Overview extends React.Component {
9      constructor(props) {
10         this.open_new_window = this.open_new_window.bind(this);
11     }
12     open_new_window() {
13         ReactDOM.render(<Overview />, document.getElementById('root'))
14     }
15
16     render() {
17         return(
18             <div>
19                 <button id='Overview'> <img src={this.props.image} onClick={this.open_new_window}/>{this.props.title}</button>
20             </div>
21         );
22     }
23 }
24
25
26 export default Collegamento_Overview;

```

Figure 3.20: Link to overview screen's script

In this interface several components are observed including:

- * the common header in each interface;
- * the path of the screen from which it is possible to navigate to the previous page by clicking on the patient's name through a component called "Breadcrumb" (part of the library taken from material UI see Fig.3.21);

```
JS Breadcrumbs.js •
src > components > JS Breadcrumbs.js > Breadcrumb > constructor
1  import React from 'react';
2  import Typography from '@material-ui/core/Typography';
3  import Breadcrumbs from '@material-ui/core/Breadcrumbs';
4  import Link from '@material-ui/core/Link';
5  import App from '../App';
6  import ReactDOM from 'react-dom';
7  import './style.css'
8
9  class Breadcrumb extends React.Component {
10   constructor(props) {
11     this.handleClick = this.handleClick.bind(this);
12   }
13
14   handleClick() {
15     ReactDOM.render(<App />, document.getElementById('root'))
16   }
17
18   render(){
19     return(
20       <Breadcrumbs id='breadcrumbs' aria-label="breadcrumb">{this.props.title}
21       <Link color="inherit">
22         Patient
23       </Link>
24       <Link color="inherit" onClick={this.handleClick}>
25         Sara Malini
26       </Link>
27       <Typography><strong>{this.props.title}</strong></Typography>
28     </Breadcrumbs>
29   );
30 }
31 }
32 export default Breadcrumb;
```

Figure 3.21: Breadcrumbs component script

- * a form in which the doctor can enter notes and then save them;
- * a filter component from which it is possible to select a specific time frame from which to search for the graphs of the parameters to be observed (part of the library taken from material UI see Fig.3.22);

```

filterjs X
rc > components > JS filterjs > ...
1 import React from 'react';
2 import FormGroup from '@material-ui/core/FormGroup';
3 import FormControlLabel from '@material-ui/core/FormControlLabel';
4 import Checkbox from '@material-ui/core/Checkbox';
5
6
7
8
9 export default function CheckboxLabels() {
10   const [state, setState] = React.useState({
11     checkedA: false,
12     checkedB: false,
13     checkedF: false,
14     checkedG: false,
15   });
16
17   const handleChange = (event) => {
18     setState({ ...state, [event.target.name]: event.target.checked });
19   };
20
21   return (
22     <FormGroup row>
23       <FormControlLabel
24         control=<Checkbox checked={state.checkedA} onChange={handleChange} name="checkedA" color="#48ccbf"/>
25         label="Week"
26       />
27       <FormControlLabel
28         control={
29           <Checkbox
30             checked={state.checkedB}
31             onChange={handleChange}
32             name="checkedB"
33             color="#48ccbf"
34           />
35         }
36         label="Month"
37       />
38       <FormControlLabel control={<Checkbox name="checkedC" color="#48ccbf"/>} label="3 Months" />
39
40       <FormControlLabel
41         control={<Checkbox checked={state.checkedG} onChange={handleChange} name="checkedG" color="#48ccbf" />}
42         label="Year"
43       />
44     </FormGroup>
45   );
46 }
47

```

Figure 3.22: Filter component script

- * a calendar from which you can select a date from which to retrieve the relative medical history (part of the library taken from material UI);

```

JS Calendario.js X
src > components > JS Calendario.js > ...
1 import 'date-fns';
2 import React from 'react';
3 import Grid from '@material-ui/core/Grid';
4 import DateFnsUtils from '@date-io/date-fns';
5 import {
6   MuiPickersUtilsProvider,
7   KeyboardDatePicker,
8 } from '@material-ui/pickers';
9
10 export default function MaterialUIPickers() {
11   // The first commit of Material-UI
12   const [selectedDate, setSelectedDate] = React.useState(new Date('2020-07-10T21:11:54'));
13
14   const handleDateChange = (date) => {
15     setSelectedDate(date);
16   };
17
18   return (
19     <MuiPickersUtilsProvider utils={DateFnsUtils}>
20       <Grid container justify="space-around">
21         <KeyboardDatePicker
22           disableToolbar
23           variant="inline"
24           format="MM/dd/yyyy"
25           margin="normal"
26           id="date-picker-inline"
27           value={selectedDate}
28           onChange={handleDateChange}
29           KeyboardButtonProps={{
30             'aria-label': 'change date',
31           }}
32         />
33       </Grid>
34     </MuiPickersUtilsProvider>
35   );
36 }
37
38

```

Figure 3.23: Calendar component script

- * two graphs that respectively represent the weekly trend of the systolic and diastolic pressure and the patient's blood glucose values.

The graphs were built with the use of HighCharts (Javascript Charting Library), a library for the representation of graphs written in pure JavaScript language, which provides an easy way to add interactive graphs to your websites or web applications. Currently it supports line, spline, area, areapline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange and polar chart types.

```
JS Overview.js X
src > components > JS Overview.js > ...
1  import React from 'react';
2  import Intestazione from './Intestazione';
3  import Breadcrumbs from './Breadcrumbs';
4  import Form from './Form';
5  import Filter from './filter.js'
6  import Calendario from './Calendario.js'
7  import '../stile.css'
8  import Grafico from './Grafico'
9
10
11
12 function Overview() {
13   return (
14     <div>
15       <Intestazione button='SAVE' />
16       <div id='contenitore2'>
17         <Breadcrumbs title='Overview' />
18         <Form />
19         <div id='filter'>
20           <div style={{color: '#48ccbf'}}><strong>Filter search </strong></div>
21           Display data patient by:
22           <Filter />
23           <Calendario />
24         </div>
25         <div>
26           <Grafico />
27         </div>
28       </div>
29     </div>
30   )
31 }
32
33
34 export default Overview;
35
```

Figure 3.24: Overview script

- PERSONAL DATA (with the description of the patient's personal and personal data);
- RISK FACTORS (with the description of the various risk factors connected to the patient, plus the possibility for the doctor to send a message to the patient if there were any significant changes in the clinical data observed);
- STATUS (describes the information relating to the cardiovascular interventions to which the patient has undergone);

- HISTORY (with the general and cardiovascular anamnesis prescribed by the doctor, it is also possible in this interface to prescribe a recommended therapy that can be sent to the patient or printed).

See script in Fig.3.27.

By clicking on the HISTORY button (see script of the link between the Home screen and the History screen in Fig.3.26), a new interface opens (see Fig 3.25):

Cardio Web Platform + New Visit

Patient / Sara Malini / History

07/10/2020

General History:

Cardiovascular History:

Recent History:

Systolic pressure mmHg: Diastolic pressure mmHg:

mmHg mmHg

Pression Pression

Conclusion

Therapy

Back

SAVE SAVE SAVE SAVE SEND PRINT

Figure 3.25: History interface

```

JS Collegamento.js
src > components > JS Collegamento.js > Collegamento > constructor
1 import React from 'react';
2 import "../style.css"
3 import History from './History.js'
4 import ReactDOM from 'react-dom';
5
6
7 //Sto usando una class component//
8 class Collegamento extends React.Component {
9   constructor(props) {
10     this.open_new_window = this.open_new_window.bind(this);
11   }
12   open_new_window() {
13     ReactDOM.render(<History />, document.getElementById('root'))
14   }
15
16   render() {
17     return(
18       <div>
19         <button id='Overview'> <img src={this.props.image} onClick={this.open_new_window}/>{this.props.title}</button>
20       </div>
21     );
22   }
23 }
24
25
26 export default Collegamento;

```

Figure 3.26: Link to History screen's script

In this interface several components are observed including:

- * the common header in each interface;
- * the path of the screen from which it is possible to navigate to the previous page by clicking on the patient's name through a component called "Breadcrumb" (part of the library taken from material UI see Fig.3.21);
- * a form in which the doctor can enter notes and then save them;
- * a calendar from which you can select a date from which to retrieve the relative medical history (part of the library taken from material UI see Fig.3.23);
- * then various forms in which the doctor will have the opportunity to enter the general, cardiovascular, recent anamnesis, systolic and diastolic pressure, then subsequent conclusions from the doctor and the related recommended therapy. The therapy can be printed or sent to the patient;
- * finally a button back from which the doctor can return to the main page or the patient's initial dashboard.

```

JS History.js X
src > components > JS History.js > ...
1  import React from 'react';
2  import Intestazione from '../Intestazione';
3  import Form from '../Form';
4  import Anamnesi from '../Anamnesi';
5  import PrintIcon from '@material-ui/icons/Print';
6  import Calendario from '../Calendario.js';
7  import Pression from '../Pression';
8  import Collegamento from '../Collegamento_Home';
9  import '../stile.css';
10 import Breadcrumbs from '../Breadcrumbs';
11
12 class History extends React.Component {
13   constructor(props) {
14     super(props);
15     this.state = {
16       stato: 'Benvenuto!'
17     };
18   };
19   render() {
20     return(
21       <div>
22         <Intestazione button='SAVE' />
23         <div id='contenitore2'>
24           <Breadcrumbs title='History' />
25           <Form />
26           <Calendario />
27           <Anamnesi label='General History:' button='SAVE' />
28           <Anamnesi label='Cardiovascular History:' button='SAVE' />
29           <Anamnesi label='Recent History:' button='SAVE' />
30
31           <div style={{color: '#48ccbf'}}><strong>Sistolic pressure mmHg:</strong>
32           <Pression />
33         </div>
34         <div id='layout'>
35           <strong>Diastolic pressure mmHg:</strong>
36           <Pression id='layout' />
37         </div>
38
39         <div>
40           <Anamnesi label='Conclusion' button='SAVE' />
41           <button id='print'><PrintIcon /></button>
42
43         </div>
44         <div>
45           <Anamnesi label='Therapy' button='SEND' />
46           <button id='print'><PrintIcon /></button>
47
48         </div>
49         <Collegamento title='Back' />
50
51       </div>
52     );
53   };
54 }
55
56 export default History;

```

Figure 3.27: History script

- THERAPY (with the therapeutic indication prescribed by the doctor);
- RECORDED EVENTS (which describes the events recorded by the patient via the mobile version of the Cardiofilo app).

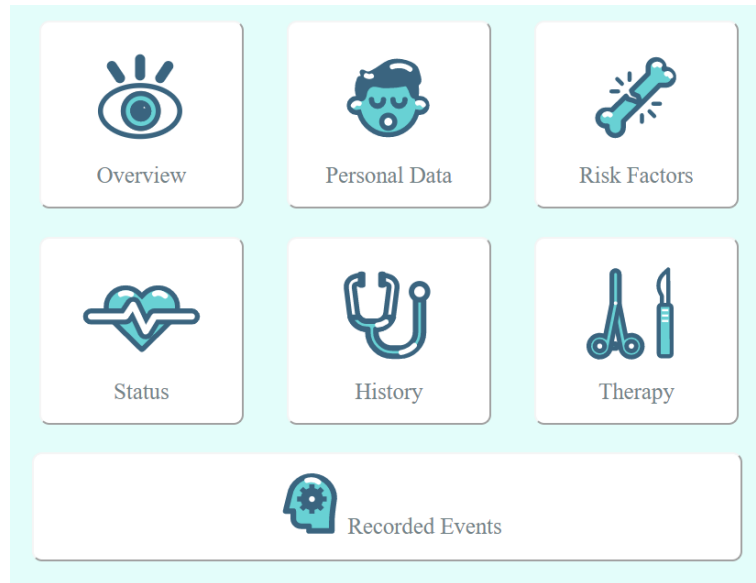


Figure 3.28: Dashboard components

- Finally, the *UploadButtons* component (see Fig.3.29) is inserted in the UI, which will provide the user with the functionality to insert attachments relating to patient data, selecting them from the folders of the PC used by the doctor (see figure 3.29, part of the figure 3.11).

This feature has been designed to attach, for example, paper documents filled in by the doctor to the web app in order to track them over time. The doctor will then have the possibility to attach, for example, the visit reports, the test report recorded by the records that the patient makes with the mobile app (recording his current weight, blood pressure, heart rate, blood sugar, number of smoked cigarettes) or medical certificates. That component comes from a component library: Material-UI.

This library offers the typical building blocks of a library using Google's Material Design language as a guideline [13].

Another feature is mobile first, which means it's easy to produce a user interface that adapts very easily to large screens, smartphones, and tablets (see script in Fig. 3.30).

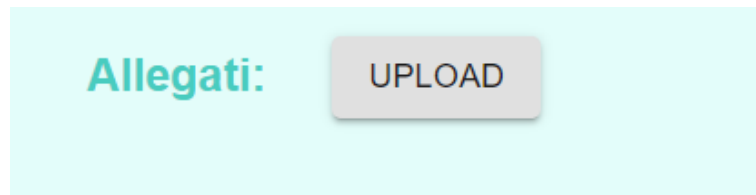


Figure 3.29: UploadButtons component

```

JS Update_allegatijs X
src > components > JS Update_allegatijs > ...
 1  import React from 'react';
 2  import { makeStyles } from '@material-ui/core/styles';
 3  import Button from '@material-ui/core/Button';
 4
 5
 6  const useStyles = makeStyles((theme) => ({
 7    root: {
 8      '& > *': {
 9        marginLeft: '305px'
10      },
11    },
12    input: {
13      display: 'none',
14    },
15  }));
16
17  export default function UploadButtons() {
18    const classes = useStyles();
19
20    return (
21      <div className={classes.root}>
22        <input
23          accept="image/*"
24          className={classes.input}
25          id="contained-button-file"
26          multiple
27          type="file"
28        />
29        <label htmlFor="contained-button-file">
30          <Button variant="contained" color="#48ccbf" component="span">
31            Upload
32          </Button>
33        </label>
34      </div>
35    );
36  }
37

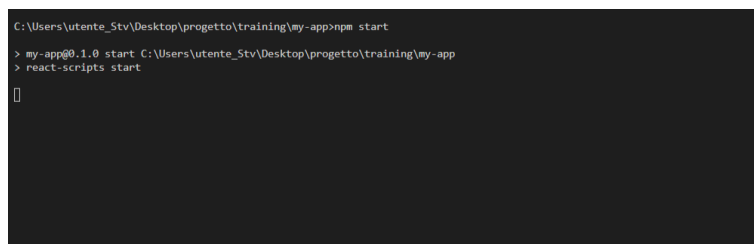
```

Figure 3.30: Component from Material UI

The execution of the entire script takes place from the terminal integrated in Visual Studio Code with the *npm start* command (see fig.3.31), part of the package downloaded from Node.js, by positioning in the current folder where the entire project is present, in the folder 'my-app' created.

The *npm start* command will run the application in development mode and you can preview the application on any browser page that will open by navigating to [http: localhost: 3000](http://localhost:3000).

The page will automatically reload whenever it detects a code change in the source files. Warnings and errors can also be viewed in the console.

A screenshot of a terminal window with a dark background. The text in the terminal is as follows:

```
C:\Users\utente_Stv\Desktop\progetto\training\my-app>npm start
> my-app@0.1.0 start C:\Users\utente_Stv\Desktop\progetto\training\my-app
> react-scripts start
█
```

Figure 3.31: Terminal execution of the application

Chapter 4

General Data Protection Regulation and Cardiofilo

4.1 Processing of personal data including biomedical data

General Data Protection Regulation or GDPR is a regulation of the European Union relating to the protection of personal data and privacy.

It was published, for the first time, in the European Official Journal on May 4, 2016 and entered into force on May 24, 2016.[14]

However, its implementation took place after two years, starting from May 25 2018.

After its entry into force, the GDPR has replaced the contents of the data protection directive (Directive 95/46 / EC) and, in Italy, has repealed the articles of the code dedicated to the protection of personal data (Legislative Decree no. 196/2003) incompatible with it. Being a regulation, it is implemented in the same way in all the States of the European Union without margins of freedom in the adaptation and the European Commission proposes as an objective to strengthen the protection of personal data of citizens of the European Union, of residents in the European Union: both inside and outside the borders of the European Union (EU).

All this because after the Lisbon Treaty (signed on 13 December 2007) the protection of personal data has become a fundamental right of people, and therefore it must be ensured in the same way throughout the EU. The regulation is functional to digital development, following the regulations in the European Union, and to the protection of the freedom of circulation of the personal data of the individual. In particular, with the introduction of the new regulation, greater attention was placed on the use of personal data including health data.

In Cardiofilo, health data such as blood pressure data, cardiovascular data and data related to current therapies will be managed. So it is necessary to understand how to treat this type of data in the application.

4.2 Contents of the GDPR regulation

The GDPR regulation is more explicit than directive 95/46, proclaiming the protection of the right to the protection of personal data understood as the fundamental right of individuals.

The main principle of the GDPR regulation is information self-determination: the individual's right to decide, outside external pressure, whether and within what limits to disclose facts related to his personal life.

It is a necessary condition for the free development of the citizen's personality and also an essential element in a democratic society.

In particular, the regulation on the processing of personal data highlights the need to implement protection and guarantee measures for the data processed, following the:

- privacy by design principle, which emphasizes that products and services must be designed from the outset in order to protect users' privacy, that is, processing must be foreseen and configured from the outset, providing guarantees to protect the rights of concerned;
- privacy by default with which the privacy measures must be implemented by default;
- principle of the personal nature of the IP address.

Compared to directive 95/46, in the GDPR there is also an approach based on risk assessment (risk based), where the extent of responsibility of the owner or manager of the treatment is determined, taking into account the nature, scope, context and purpose of the processing, as well as the probability and seriousness of the risks to the rights and freedoms of users.

A risk-based approach has the obvious advantage of demanding obligations that can go beyond mere compliance with the law, it is certainly more flexible and adaptable to changing needs and technological tools.

A risk based approach, however, has the disadvantage of delegating risk assessment to the company, making disputes more difficult in the event of violations.

The regulation applies to the processing of personal data, and to the non-automated processing of data stored in a defined "archive".

The Regulation therefore applies to any fully or partially automated processing of personal data and to the non-automated processing of personal data contained in an archive or intended to appear in it. In this perspective, the GDPR regulates only the processing of personal data concerning a natural person, with the exception of legal persons (except for a few exceptions).

So only natural persons can be interested in the treatment, not also legal persons. The

GDPR, however, does not apply to deceased persons, but here the Decree of adaptation of the Privacy Code intervenes which extends the GDPR's protections to the processing of data of the deceased (art.2).

GDPR, therefore emphasizes the principle of transparency, with a view to respecting the purpose. Consequently, it is necessary to carefully evaluate the purposes of the treatment, in order to establish exactly which data can be processed and which cannot. Furthermore, the data controllers will have to identify the legal basis of the processing (for example the consent of the interested party) and document it.

4.3 Data

The data processing explained by GDPR includes various types of data. In addition to personal data, we also find genetic, biometric and health-related data, which is all information that allows the univocal identification of a natural person.

- Personal data (art.4 paragraph 1)[15]:
information related to an identified or identifiable natural person. The person can be identified directly or indirectly, with particular reference to an identifier such as the name, an identification number, location data, an online identifier or to one or more characteristic elements of his physical, physiological, genetic, psychic identity, economic, cultural or social;
- Sensitive data (art.9 paragraph 1)[15] :
considering personal data such as racial or ethnic origin, political opinions, religious or philosophical beliefs, or union membership, data relating to the sexual life or sexual orientation of the person, as well as:
 - Genetic data: inherited or acquired, obtained through DNA and RNA analysis from a biological sample of the physical person in question;
 - Biometric data: such as the facial image, thanks to which it is possible to identify one and only one natural person;
 - Health data: both physical and mental, past, present or future, but also information on health care services, where present, regardless of the source, such as, for example, a doctor.
 - Personal data relating to criminal convictions or crimes (art.10): the processing of personal data relating to crimes or convictions must take place under the control of the public authority or if it is authorized by Union law.

4.4 GDPR application in Europe

After three months the entry of the new regulation on the protection of personal data, CNIL (Commission nationale de l'Informatique et des libertes), the French authority, equal to the Italian Guarantor for Privacy, said that companies did not seem to be aligned in the application of the directive.

Following an analysis based on the provisions of GDPR and its own resolutions, CNIL underlined that there is a general lack of awareness of the companies on the risks of non-compliance and any penalties that could result.

The supervisory body said that many believe that they will not be subject to an inspection, others that it will be sufficient to show proactivity and collaboration, still others have the belief that a policy is sufficient to protect themselves from the risks of violations.

The situation in Europe after the entry of the GDPR (May 25, 2018) is this:

- in Luxembourg Parliament the directive was introduced on 26 July;
- in Spain national legislation was approved on 27 July 2018;
- in France, the legislation was adopted just 2 weeks before the entry of the GDPR;
- in Italy, the Guarantor expressed a favourable opinion on the outline of the legislative decree; approved by the House and Senate commissions on June 20, arrived at the Council of Ministers on August 8 2018. The Guarantor for the Protection of Personal Data, National Supervisory Authority, published a resolution on the inspection activity planned between July and December 2018, which was addressed:
 - to check on data processing carried out by companies / entities that manage databases of significant size;
 - to ascertain personal data processing carried out at credit institutions in relation to the legitimacy of the consultation and subsequent use of data, the tracing of accesses and related protection measures;
 - to check on personal data processing carried out by companies for telemarketing activities;
 - to check on subjects, public or private, belonging to homogeneous categories on the assumptions of lawfulness of the treatment and on the conditions for consent if the treatment is based on this assumption, on compliance with the disclosure obligation and on the duration of the conservation some data.

The European Personal Data Protection Authorities (DPA) have therefore faced a huge volume of work (there is talk of managing 50,000 data breaches notified since May 25,

2018 across Europe). In fact, these are time-consuming activities and the results (even of a high profile) could be verified only in the future.

4.5 Health data treatment

One of the sectors on which the EU Regulation 679/2016 (GDPR) has had a greater impact is the health: a new approach on the management of the processing of personal data in the health sector. [12]First of all, the Regulation introduces for the first time a definition of "health data", according to which these data consist of personal data relating to the physical or mental health of a natural person. Greater attention was subsequently paid to the meaning of consent that it is considered as a manifestation of free and informed will of the interested party. It should be noted that consent cannot be considered freely expressed, and therefore valid, if the interested people cannot make a truly free choice or if he cannot refuse to give his consent without suffering prejudice. With reference to the consent to the processing of personal data for the execution of a health treatment, it is clear that such consent is not necessary as a health professional could never cure a patient without also processing his personal data. Doctors will therefore be able to process patient data, for treatment purposes, without having to request their consent, but will still have to provide them with complete information on the use of the data. All public bodies, as well as private operators who carry out large scale processing of health data, such as nursing homes, are required to appoint the DPO (Data Protection Officer). Instead, consent is required, or a different legal basis, when such health treatments are not strictly necessary for the purposes of treatment, even when they are carried out by health professionals.

Finally, GDPR clarifies that it is mandatory for all healthcare professionals to keep a register listing the treatment activities carried out on patient data. This document represents, in any case, an essential element for the management of the treatments and for the effective identification of those at greatest risk, also to demonstrate compliance with the principle of accountability envisaged by the GDPR. It was therefore clarified that the data can be processed only for the purpose of protecting the health and physical safety of the interested person or of third person or of the community, pursuant to Article 9, paragraphs 2, letters h) and i).

It follows that in the context of scientific research in the medical, biomedical and epidemiological fields, particular attention must be paid to the reuse of personal data, previously collected for diagnostic purposes.

The reference to art. 89 clarifies that a fundamental prerequisite for proceeding with further processing is the adoption of adequate guarantees for the data subject's fundamental rights and freedoms, or technical and organizational measures suitable for pursuing the principle of data minimization, which may include pseudonymisation. Therefore, the reuse

of data previously collected for prevention, diagnosis and treatment purposes can be considered legitimate, and the condition of lawfulness, as an alternative to consent, is to be identified in art. 9, par. 2, lett. j) of the GDPR. So the methods of data processing for research purposes are these: the investigation must be conducted, if possible, on anonymous data. If the search results cannot be pursued without the identification of the interested person, data encryption or pseudonymisation solutions must be adopted, or other measures that do not make the data directly attributable to the interested party.

The subjects involved in the research activity who act as data controllers can communicate with each other the personal data, necessary for the conduct of the study if they cover the ground as promoter, coordinating or participating role. The personal data and biological samples used for research must be kept by means of encryption techniques or the use of identification codes, for a period not exceeding that necessary to achieve the purposes for which they were collected or subsequently processed.

Furthermore, in order to increase the level of security of the treatment, other particular measures are indicated such as: the protection of the study data from the risks of abusive access or theft (through file system or database encryption), the use of protected transmission channels, the labelling of samples for storage and transmission of the same.

4.6 How to treat Cardiofilo data

The most appropriate and safe way for the management and storage of patients' personal data is to rely on a digitalization of health data with a medical software, as a Cardiofilo, suitable for the GDPR.

Cardiofilo is a medical application that will also allow to manage patients' personal and sensitive data in accordance with the GDPR in compliance with the law for the management of health data.

Patient privacy will be protected thanks to various internal and external provisions of the company that will market Cardiofilo. Possible proposals for the application to be adequate for the processing of personal and health data for example are:

- Define the data collector that will allow to collect different sets of data in a completely secure way. Data collection can be performed constantly or on a user-defined schedule by means of a dedicated server. The use of a dedicated server compared to the use of a cloud guarantees power, top-level performance and maximum security in data management precisely because its operation is confidential and is not shared with anyone else.
However, in order to have a dedicated server, the company will have to face the costs and long maintenance times of its.
- Management of data consent: patients could choose between using the application with their own data by signing a consent to the processing of the data. Or patients could use the application by pseudonymising his personal data, so the patient will be provided with a unique code useful for his identification.
- Encrypted technology: all communications and notifications, of personal and health data, and documents sent with Cardiofilo app to patients are controlled through an https and password encryption system ensuring greater security of privacy;
- The same access to the information present in the medical software could be profiled thanks to a two-factor personal and nominative authentication. Furthermore, all the actions performed by users could be traceable by geo-localizing access to the software via IP.

Chapter 5

Classification of Cardiofilo as a medical device

5.1 Introduction of medical devices

A medical device is an instrument used in medicine and can have diagnostic and / or therapeutic purposes.

They are often subject to a specific regulatory discipline, which may vary from country to country. This category includes active medical devices, in vitro devices but also software and medical applications.

The medical device sector is of great importance in European healthcare, contributing to the improvement of the level of health protection through the development of innovative solutions for diagnosis, prevention, treatment and rehabilitation. [16]

The EEC Directive 93/42 on medical devices (abbreviated as DDM 93/42), published on the GUCE (Official Journal of the European Union) in June 1993 [17], is a document that reports the general criteria to be used in the design and construction of some categories of medical devices, in force in the states of European Union.

In order to become legal, a medical device must comply with the reference regulations, it must therefore be linked to the scientific literature and clinical studies consistent with the relevant health provisions, it must be able to exhibit certifications in accordance with the law and ensure the possibility of transparent verification of the certifications issued by notified bodies. In general, a medical device in use in a member country of the European Union must always exhibit the CE marking, with a clear indication of the number of the notified body that issued it.

This allows the notified body to verify the actual characteristics for which the device has received authorization in the health sector, and whether the marking is authentic and consistent with the manufacturer's advertised declaration of use.

Cardiofilo project consists of a web interface for cardiologists and a mobile interface for

patients and provides an additional tool for secondary prevention of cardiovascular diseases, therefore it provides support for therapeutic and diagnostic purposes. It is clear that the application can be classified as a medical device.

In particular, it can be classified as a "software as medical device" (SaMD).

5.2 Directive CEE 93/42

The EEC Directive 93/42 relating to medical devices (abbreviated as DDM 93/42), reports the general criteria to be used in the proposal and structure of categories of medical devices.

In the first article of the implementation of Directive 93/42 concerning medical devices (decree lgs. 24 febbraio 1997, n. 46 emended with D. lgs. 25.01.2010, n.37 - Transposition of Directive 2007/47/CE), it says that: "This decree is applied to medical devices and related accessories. For the purposes of this decree accessories are considered full-fledged medical devices. In the present decree and its annexes, medical devices and their accessories are indicated with term devices" [18]. It imposes the obligation of CE marking for the sale of these medical devices (DM); to obtain the CE mark, the essential requirements of the DDM 93/42 must be respected.

The requirements mentioned in the document must be interpreted and applied in order to take into account the technology and practices existing in the design phase.

DDM 93/42 is a supranational validity document and was implemented in Italy in February 1997 with Legislative Decree 24 February 1997, n. 46 ("Implementation of Directive 93/42 / EEC, concerning medical devices ").

The directive proposes provisions for the certification of medical devices that are common to all States of the European Community. In previous years, the legislative and administrative provisions concerning the safety and functioning characteristics of the DMs, the certification and control procedures had generally different content and scope in each State, therefore they represented an impediment to intra-Community trade.

The directive is not a list of requirements, that is, it does not list all the rules for each type of device, since:

- the prescriptive rules require periodic review; due to the rapid technological progress in this sector, the standards would have to be revised very frequently (every 2 or 3 years), which would entail high costs;
- Furthermore, thanks to progress, we can have a very wide instrumentation and in certain cases it is possible to obtain a safe device by traveling on different roads with the same validity: it would be almost impossible to consider them all.

This decree does not apply:

- devices intended for in vitro diagnosis;
- active implantable devices governed by the legislative decree of 14 December 1992, n. 507, and subsequent modifications;
- to medicinal products subject to Legislative Decree 24 April 2006, no. 219, which implements the Community code on medicinal products for human use. In establishing whether a particular product falls within the scope of this decree or this decree particular consideration should be given to the main mechanism of action of the product same;
- to cosmetic products governed by decree no. 713, and later modifications;
- human blood, human blood products, human plasma or blood cells of human origin, or to devices that, at the time of entry into trade, contain similar products derived from blood, plasma or cells blood, with the exception of the devices referred to in paragraph 2-bis;
- to organs, tissues or cells of human origin, or to products including or derived from tissues or cells of human origin, with the exception of the devices referred to in paragraph 2-bis;
- to organs, tissues or cells of animal origin, unless the device is manufactured using non - viable animal tissue or non - viable products derived from animal tissue.

Definition of medical devices

According to what established by the legislative decree n. 46 of 24 February 1997, or in implementation of Directive 93/42 / EEC, relating to DMs, a medical device is defined as a any instrument used alone or in combination intended for use in humans for the purpose of diagnosis, prevention or therapy. [18]

Medical devices, are a generic element of medicine. They are medical devices such as simple tongue depressors, laser surgical devices, complex programmable pacemakers with microchip technology, laboratory equipment for general use, and other electronic devices. These also include devices that may include monoclonal antibody technology.

Another category of medical devices that should be mentioned is that of active medical devices. They are defined in this way because, for their correct functioning, they must be connected to a source of electricity or to any other source of energy other than that

produced directly by the human body or by gravity. Some examples of this type are computerized axial tomography (CT), the defibrillator, the infusion pump, but also the laser. Subcategory of active medical devices is that of active implantable medical devices, i.e. any active medical device intended to be implanted entirely or partially by surgery or medical intervention in the human body or by medical intervention in a natural orifice and intended to remain there after the intervention.

Classification

A categorization of medical devices into classes is designed to group them according to their complexity and potential risk to the patient. They are essentially divided into four classes: I, IIa, IIb, III. This classification always depends on the intended use indicated by the manufacturer and is attributed by consulting Annex IX of Legislative Decree 46/1997. The classification is carried out according to three parameters judged as fundamental:

- invasiveness;
- the duration of the contact time required with the human body: if the expected continuous duration is less than 60 minutes we speak of short-term use, if the expected continuous duration does not exceed 30 days we speak of long-term use, if the duration is continuous then the contact time is greater than 30 days;
- dependence or not on energy sources.

However, when we talk about medical devices we can group them into two macro categories: invasive and non-invasive.

- Invasive medical devices are those intended to penetrate, even partially, into the body;

In turn, invasive devices are divided into:

- devices that penetrate the body through orifices;
 - surgical devices, which penetrate through the body surface in the context of or outside surgical operations.
- Non-invasive devices therefore do not penetrate any part of the body: either through orifices, or through the skin.

Some categories of medical devices are subject to special classification rules, which is why they must be subject to rigorous controls: to guarantee the patient the correct efficacy

with respect to their field of application.

In the large category of medical devices, it can be found various products designed for diagnostic and / or therapeutic purposes, including also "software as medical device" and medical apps.

The European Union has drawn up specific rules to differentiate medical devices from computer programs in the field of software and apps dedicated to health.

Software as a medical device

Before an incessant technological development regarding new healthcare software suitable for example to monitor and process physiological parameters, physical activity, calories burned, heart rate, etc., the boundary line between softwares that qualify as medical devices or that do not have this attribution appear increasingly blurred and difficult to identify.

Qualifying any softwares as a medical device, means that it will be subject to strict sector regulations aimed at ensuring compliance with the safety, efficacy and quality requirements, differentiated according to the risk classes into which the medical devices are divided. However, even in cases where a software cannot be qualified as a device, it must be able to meet safety and reliability requirements, with the aim of increasing consumer confidence in the Health-app, with positive results for collective well-being.

Software has become an important part of all products, integrated into digital platforms that serve both medical and non-medical aim. "Software as medical device" is one of three different kinds of software related to medical devices.

Two types of software are:

- software that is integral to a medical device (Software in a medical device)
- software used in the manufacture or maintenance of a medical device

Software as a Medical Device is defined as a software intended to be used for one or more medical purposes. Use of Software as a Medical Device is continuing to be used more and more. It can be used in a large range of technology platforms.

Directive 93/42 / EEC regulates the software intended by the manufacturer to be a medical device in a marginal way. The explicit prescription constituting an essential Requirement is given by point 12.1bis of Annex I.

It subordinates the validation of the software in the reference sector, taking into account the principles of the development life cycle, risk management, validation and verification. With this legislation, however, manufacturers have found it difficult to understand if their software was actually classified as a medical device.

In fact, the topic has been the subject of a specific guide, MEDDEV 2.1 / 6, from which

a practical algorithm to solve the problem has come down.

In July 2016, the MEDDEV 2.1 / 6 guidelines were updated with the introduction of some definitions, including that of "software" and "Software as medical device" (such as "software intended to be used for one or more medical purposes that perform these purposes without being part of a hardware medical device").

The fulfilment of the Essential Requirements imposed by the Directive was almost entirely based on the prescriptions concerning active medical devices, largely improper if applied to a software. No classification rule names software, so their classification was based on that of active medical devices (Rules 9 to 12, Annex IX).

Subsequently there is the problem of having many software classified in class I according to Rule 12, because they do not fall within the previous Rules with higher classification, but are not suitable or too stringent for the intended destination of a software. Vice versa, the fulfilment of the Essential Requirements, which can be considered as a problem following the legal framework, was resolved by virtue of point 12.1bis on the basis of the application of a quality management system (based on ISO 13485), constituting in itself one of the basic harmonized standards in the medical field. In addition, recourse was made to the specific harmonized standard for software based on IEC 62304. Both standards helped balance the general aspects with respect to the specific ones, also managing to complement each other on aspects not directly covered (above all, the exclusion of the final validation and release of the software, even when constituting a medical device in its own right, from the scope of application of standard 62304).

The fate of the software that is an integral part of active medical devices has been easier, thanks to the harmonized standard that concerns them, which incorporates IEC 60601-1, as well as point 12.1 of Annex I. In this case, by virtue of the point 2.3 of Annex IX, the software can be classified directly on the basis of the active medical device on which it operates and the detailed prescriptions of 60601-1 did the rest.

The guidelines also recognize the existence of a "grey area", as the distinction between software that is or is not a medical device is not always clear. However, it is up to the manufacturer to decide whether his product falls within the definition of a medical device (Legislative Decree 46/97 art. 1 paragraph 2). In case of doubt, the competent authorities will decide on the placement of the limit products. To this end, a working group has been set up at the European Commission Group of experts on borderline and classification medical devices.

The new European regulation on medical devices: Regulation (UE) 2017/745

The new regulation on medical devices (EU) 2017/745, published on May 5, 2019 (which will be applicable from May 2020), introduces significant changes compared to the previous legislation.

In summary, the changes are:

- establishment of a European database for medical devices (Eudamed);
- definition and detailed obligations of all economic operators;
- new figure of the Head of compliance with the law (PR);
- supervision of notified bodies;
- clinical evaluation, post-marketing surveillance and post-market clinical follow-up plan;
- transparency and traceability of medical devices (UDI system);
- deepening on the devices that contain / consist of nanomaterials;
- classification rules for borderline products.

The new regulation emphasizes the criterion of destination impressed on the software by the manufacturer, establishes that "a software in itself constitutes a medical device when it is specifically intended by the manufacturer to be used for one or more of the medical purposes established in the definition of medical device "and that" a generic software, even if used in a healthcare context, or a software for lifestyle-related applications and well-being is not a medical device "(cons. 18a). Then, to ensure a consistent classification in all Member States, with particular regard to borderline cases, the regulation provides that the Commission can decide on a case by case basis, on its own initiative or at the request of a Member State, whether a product or group of products falls within the definition of a medical device or not.

The regulation also clarifies that software classified as medical devices are in class I (except those intended to monitor physiological processes that are in class IIa or IIb when monitored are vital parameters whose variation can create an immediate danger for the patient). This means that the assessment of the suitability for marketing of a software-device should, in principle, take place under the sole responsibility of the manufacturers (except for those falling within Classes IIa and IIb which require the intervention of a notified body).

In conclusion, the regulatory process testifies to the particular attention paid by the European institutions for this theme, establishing essential criteria to sanction the distinction between pure and simple software and device.

5.3 CE Marking

The stages of the CE marking procedure for medical devices, according to Legislative Decree 46/97 [19], are:

1. device classification:

Classification is the first fundamental step that the manufacturer must take in order to determine the class of the device, according to the rules contained in Annex IX of Legislative Decree 46/97, and to use the consequent marking procedures.

2. verification of compliance with the essential requirements:

In order to be CE marked, any medical device must correspond to the so-called "essential requirements" presented in Annex I of Legislative Decree 46/97. These are safety and efficacy requirements that both the devices and their production system must have. As Annex I states, in fact, "the devices must be designed and manufactured in such a way that their use does not compromise the clinical status and safety of patients, nor the safety and health of users and possibly third parties, when used under the conditions and for the purposes envisaged, it being understood that any risks must be of an acceptable level, taking into account the benefit provided to the patient, and compatible with a high level of protection of health and safety". This means that - to create a medical device compliant with Legislative Decree 46/97 - the manufacturer must prove that not only his product, but also the production process in its various aspects (design, manufacture, controls, etc.) respect the essential requirements. The list of essential requirements is divided into two parts: the first dedicated to general requirements, completely aimed at the intrinsic safety of the devices, the second - further divided into 7 groups - covers all the design and construction aspects of the device. The higher the riskiness of the device, the greater the safety guarantees that the manufacturer must provide for the production of the device.

3. CE marking of products:

For the

- class I, the manufacturer may CE mark the product and place it on the market after drawing up a declaration of CE conformity with the essential requirements, based on Annex VII of Legislative Decree 46/97. With this document,

the manufacturer guarantees and declares that its products meet the provisions of the directive. However, the company must have all the technical documentation to demonstrate the safety of the device produced. The CE declaration of conformity is the simplest CE marking procedure. It is a simple declaration of assumption of responsibility, without the intervention of a Notified Body.

- For class IIa, the manufacturer will ask the Notified Body for approval of its production facilities and / or its product. The manufacturer can choose between two different applicable procedures:
 - Annex II (complete quality assurance system) with exclusion of point 4;
 - Annex VII (EC declaration of conformity) + Annex IV (EC verification) or V (production quality assurance) or VI (product quality assurance).
- 4. For class IIb, the manufacturer will ask the Notified Body for approval of its production facilities and / or its product, but with further guarantees and controls. The manufacturer can choose between two different applicable procedures:
 - Annex II (complete quality assurance system) with exclusion of point 4;
 - Annex III (CE certification) + Annex IV (CE verification) or V (production quality guarantee) or VI (product quality guarantee).
- 5. For class III, the manufacturer will ask the Notified Body for approval of its production facilities and / or product, but with even stricter guarantees and controls that involve, in particular, the aspect of design. The manufacturer can choose between two different applicable procedures:
 - Annex II (complete quality assurance system) including point 4 (examination of the product design);
 - Annex III (CE certification) + Annex IV (CE verification) or V (production quality assurance)

All documentation must be kept available to the health authority for a period of at least five years from the last manufacturing date.

It can therefore be deduced that the CE marking is a mandatory process for all medical devices covered by the EEC directive 93/42. The procedure must be performed by the manufacturer of a product regulated in the European Union, who declares by means of the declaration of conformity, that the product complies with the safety and health requirements set out in the directive.

5.4 Classification of Cardiofilo as a SaMD

In the last few years, the so-called eHealth was born, or mobile health understood as medicine and public health practice supported by mobile devices, such as cell phones, devices for monitoring patients, hand-held computers and other wireless devices.

Among the applications dedicated to health, Cardiofilo project was born which includes various technological solutions that allow to measure vital parameters such as body weight, blood sugar level, blood pressure, body temperature and sports activities carried out on a daily basis.

After the design and construction phase, it is legitimate to think about the marketing of the application and therefore it is necessary to understand if Cardiofilo can be understood as a medical device, how to classify it and how to obtain the CE marking accordingly.

The Medical Devices Directive 93/42 / EC has been implemented in Italy with Legislative Decree 37/2010 recognizing the importance of software in the medical device sector, as well as the existence of the software as a medical device also independently, that is, the so-called standalone software.

In addition, the Court of Justice, taking up MEDDEV 2.1 / 2, has declared that the medical purpose for therapeutic purposes is essential for the qualification of the product. According to guidelines "MEDDEV 2.1 / 6 on the qualification and classification of standalone software used in healthcare within the regulatory framework of medical devices ", Cardiofilo can be considered as a "Software as medical devices", because it is a software destined to a medical and therapeutic use. It is not a software that has the single goal of storing data or being a project of clinical research only. It can therefore be said that the application is considered a medical device.

Analysing the functions that the application provides, it can be established that Cardiofilo can be classified as a class IIa medical device since it is intended to allow control of vital physiological processes, where the nature of the variations is such as to be able to create an immediate danger for the patient.

For class IIa, the company that wants to market Cardiofilo will have to ask the notified body for approval of its product.

The company can choose between two different applicable procedures for device conformity assessment. (see section 4.3 CE marking).

In accordance with the new Regulation (EU) 2017/745 for the conformity assessment, the company must guarantee and document a quality management system, including the PostMarket Clinical Follow-up (PMCF).

In addition, the company must have a figure responsible for compliance with the regulation (PR) that must have the necessary skills in the field of medical devices. This figure can be internal or external, in relation to the size of the company.

Chapter 6

Future improvements

The interfaces thus developed of the Cardiofilo Web application contain scalable and reusable components and a single CSS style file, in this developed project React takes only the concept of 'View'. It is a library that renders the Views.

Subsequently it is possible to implement a UI in React following both the MVC (Model-view-controller) pattern and that with Redux, the problem that will have to be overcome is the management of the state in the applications.

From a purely technical point of view, the state of a system (for example of an application) is the set of internal conditions in a specific instant that determine the result of interactions with the outside.

In other words, the status of an application is the set of information that determines the output at a given input in a specific instant.

In application development there is always something to do with state management and this management is one of the most critical points responsible for much of the complexity of the software.

In the case of the Cardiofilo project, problems related to the management of the state at the local level could arise, as the number of developers increases, as it may happen that not everyone in the team will correctly apply the state control at the local level, therefore, have a well-defined architecture with the help of Redux, it helps to maintain good code quality. So local state management is very good only in case:

- the application is not large;
- the team is not large;
- people are developing component libraries

The management of the state besides being difficult due to the increase of developers who manage the app, If changes can occur asynchronously, the scenario can be quite complex and it is easy to lose control over the evolution of the state, with the result that, in

the presence of malfunctions, it is really difficult to recreate such a situation to allow an analysis of the problem and its resolution.

So one possible solution to the problem is the use of Redux, [20] an open source JavaScript library to manage the state of the application. It is commonly used with libraries such as React or Angular to build user interfaces. Dan Abramov, the creator of Redux, identifies the cause of the complexity in managing the state of an application in two elements: modifiability and asynchronicity.

The solution proposed by Redux aims precisely to remove these two aspects, based on three fundamental principles:

1. there is a single source of truth=the status of the entire application is stored in a single object;
2. the status is read-only=it is not possible to directly modify the information stored in the state of the application; the only way to do this is through explicit actions;
3. changes to the state must be made with pure functions=the current state is replaced by a new state generated by functions exclusively based on an action and the previous state

Respect of these principles allows to obtain benefits such as:

- the predictability of state transitions: the changes in the state of an application are analysable and reproducible, and it is therefore easier to understand how a specific situation was arrived at;
- the organization of the code: Redux forces you to organize the code following a specific pattern, which indirectly defines a coding standard within a development team;
- the rendering server: you can define an initial state at will and start the application from that state, in this way + you can facilitate the rendering of JavaScript applications from the server;
- status tracking: state transitions can be tracked for debugging purposes;
- the testability of the code: since state changes are made through pure functions, it is easier to write unit tests for application.

However, it should be said that Redux is not the solution to all ills. The benefits it introduces in the management of the state of an application have a price in terms of complexity, or rather, of the direction in which the code is written.

Therefore for the future development of the Cardiofilo application the use of the Redux library is proposed, since it would allow managing the complexity of the various user and developer actions with the addition at the same time of a high degree of programming complexity which can however guarantee a truly usable and interoperable application .

Chapter 7

Conclusions

The Cardiofilo project in its entirety (Cardiofilo web app and Cardiofilo mobile app) aims to provide an auxiliary monitoring to the doctor, who will always be able to keep his patients suffering from cardiovascular diseases under control.

The next purpose of the front-end development is to provide an excellent user experience, therefore try to understand everything that revolves around the interaction of a user with the app and at the same time provide the end user who will use the app aspects such as the utility, ease of use and efficiency of the system.

Designing an application following all aspects related to usability is not the only step that a company must take when it decides to develop an application compliant with the world of telemedicine in the healthcare field.

In fact, in order to access foreign markets, the CE mark represents a respect and a guarantee of quality and conformity.

Applying the CE mark on its products therefore means that a company wants to be attentive and in compliance with all legal obligations for safety and compliance, both legislative and environmental.

Designing an application following all aspects related to usability is not the only step that a company must take when it decides to develop an application compliant with the world of telemedicine in the healthcare field.

In fact, in order to access foreign markets, the CE mark represents a respect and a guarantee of quality and conformity. Applying the CE mark to its products therefore means that a company wants to be careful and in compliance with all legal obligations for safety and compliance, both legislative and environmental.

In order to have access to the world of telemedicine, therefore, an efficient, useful, simple application must be presented, which can assist the work of health personnel with the use of information and telecommunication technologies for the benefit of human health.

With the Cardiofilo application, therefore, we want to respond to a very concrete and

rapidly growing technological sector for the benefit of human health, designed and then developed to generate numerous important benefits in terms of personalized aids to significantly improve the quality of life of people (improving health, fitness, mental well-being, safety).

Part I

Appendix

App.js

```
import React from 'react';
import './App.css';
import './stile.css'
import Intestazione from './components/Intestazione';
import Form from './components/Form';
import Component_Button from './components/Component_Button'
import Table from 'react-bootstrap/Table'
import overview from './overview.svg';
import personaldata from './personaldata.svg';
import RiskFactors from './RiskFactors.svg';
import Status from './Status.svg';
import History from './History.svg';
import Therapy from './Therapy.svg';
import Events from './RecordedEvents.svg';
import RecordedEvents from './components/Recorded_Events';
import UploadButtons from './components/Update_allegati';
import Collegamento from './components/Collegamento';
import Collegamento_Overview from './components/Collegamento_Overview';
```

```
function App()
return (
  <div>
    <Intestazione/>
    <div id='contenitore2'>
      <div style=color:'48ccbf'>Patient/ <strong>Sara Malini</strong></div>
      <Form/>
      <Table striped bordered hover style=marginLeft:'285px'>
        <tbody>
          <tr>
            <td><Collegamento_Overview title='Overview' image=overview/></td>
            <td><Component_Button title='Personal Data' image=personaldata/></td>
            <td><Component_Button title='Risk Factors' image=RiskFactors/></td>
          </tr>
          <tr>
            <td><Component_Button title='Status' image=Status/></td>
            <td><Collegamento title='History' image=History/></td>
```

```

<td><Component_Button title='Therapy' image=Therapy/></td>
</tr>
</tbody>
</Table>
<RecordedEvents title='Recorded Events' image=Events/>
<div id="Update">
<div id="Allegati"><strong>Allegati:</strong></div>
<UploadButtons style=marginLeft:'316px'/>
</div>

</div>

</div>
)

export default App;

```

stile.css

#Form

width: 591.6px;
height: 46.2px;
padding: 12px 20px;
margin: 8px 0;
box-sizing: border-box;
border: 2px solid 38e6ca;
border-radius: 4px;
margin-top: 24px;
margin-left: 292px;

#intestazione

width: 1366px;
height: 70px;
object-fit: contain;
box-shadow: 0 3px 6px 0 rgba(0, 0, 0, 0.16);
background-color: #ffffff;
position: relative;

#logo

position: relative;
width: 60px;
height: 52.4px;
object-fit: contain;
float: left;

#text

position: absolute;
width: 168px;
height: 30px;
object-fit: contain;
font-family: 'Montserrat', sans-serif;
font-size: 25px;
font-weight: bold;
line-height: 1.2;
letter-spacing: normal;

```
text-align: left;
color: #48ccbf;
margin-left: 93px;
margin-top: 18px;
```

```
#NewVisit
position: absolute;
width: 162px;
height: 45px;
object-fit: contain;
-moz-border-radius: 8px;
border-color:#48ccbf;
-webkit-border-radius: 8px;
border-radius: 8px;
color:#fff;
background:#48ccbf;
font-family: 'Montserrat', sans-serif;
font-size: 19px;
font-weight: bold;
font-stretch: normal;
font-style: normal;
line-height: 1.21;
letter-spacing: normal;
text-align:center;
color: #ffffff;
margin-left: 909px;
margin-top: 10px;
```

```
#Back
position: absolute;
width: 162px;
height: 45px;
object-fit: contain;
-moz-border-radius: 8px;
border-color:48ccbf;
-webkit-border-radius: 8px;
```



```
border-radius: 8px;
color: #fff;
background:#48ccbf;
font-family: 'Montserrat', sans-serif;
font-size: 19px;
font-weight: bold;
font-stretch: normal;
font-style: normal;
line-height: 1.21;
letter-spacing: normal;
text-align:center;
color: fffff;
margin-left: 15px;
margin-top: -20px;
```

```
#contenitore2
width: 1366px;
height: 900px;
object-fit: contain;
box-shadow: 0 3px 6px 0 rgba(0, 0, 0, 0.16);
background-color: #e3fdfa;
position: relative;
```

```
#submit
position: absolute;
width: 45px;
height: 45px;
object-fit: contain;
-moz-border-radius: 8px;
border-color:#48ccbf;
-webkit-border-radius: 8px;
border-radius: 8px;
color: #fff;
background:#48ccbf;
font-family: 'Montserrat', sans-serif;
font-size: 9px;
```

```
font-weight: bold;
font-stretch: normal;
font-style: normal;
line-height: 1.21;
letter-spacing: normal;
text-align:center;
color: #ffffff;
margin-top: 23px;
margin-left: 13px
```

```
#Overview
width: 162.6px;
height: 149px;
object-fit: contain;
position: relative;
font-family: FaktPro;
font-size: 18px;
font-weight: normal;
font-stretch: normal;
font-style: normal;
line-height: 1.22;
letter-spacing: normal;
text-align: center;
background-color: white;
color: #728388;
-moz-border-radius: 8px;
-webkit-border-radius: 8px;
border-radius: 8px;
border-color: whitesmoke;
margin-top: 20px;
margin-left: 30px;
```

```
#RecordedEvents
width: 563.6px;
height: 86.2px;
object-fit: contain;
```

```
position: relative;
font-family: FaktPro;
font-size: 18px;
font-weight: normal;
font-stretch: normal;
font-style: normal;
line-height: 1.22;
letter-spacing: normal;
text-align: center;
background-color: white;
color: #728388;
z-index: 1;
-moz-border-radius: 8px;
-webkit-border-radius: 8px;
border-radius: 8px;
border-color: whitesmoke;
margin-left: 312px;
margin-top: 20px;
```

```
#Update
margin-left: 232px;
margin-top: 30px;
```

```
#Allegati
position: absolute;
width: 168px;
height: 30px;
object-fit: contain;
font-family: 'Montserrat', sans-serif;
font-size: 20px;
font-weight: bold;
line-height: 1.2;
letter-spacing: normal;
text-align: left;
color: #48ccbf;
margin-left: 197px;
margin-top: 5px;
```

```
#Anamnesi
width: 1245px;
height: 50px;
border: solid 1px #cbf0ef;
padding: 12px 20px;
margin: 8px 0;
box-sizing: border-box;
border-radius: 4px;
margin-top: 22px;
margin-left: 3px;
```

```
#print
```

```
width: 45px;
height: 45px;
object-fit: contain;
-moz-border-radius: 8px;
border-color: #48ccbf;
-webkit-border-radius: 8px;
border-radius: 8px;
color: #fff;
background: #48ccbf;
margin-left: 1262px;
margin-top: -5px
```

```
#layout
margin-top: -116px;
margin-left: 322px;
color: #48ccbf;
```

```
#breadcrumbs
color: #48ccbf;
```

```
#filter
width: 1366px;
```

```
height: 140px;
object-fit: contain;
box-shadow: 0 3px 6px 0 rgba(0, 0, 0, 0.16);
background-color: white;
```

App.test.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

it('renders without crashing', () =>
const div = document.createElement('div');
ReactDOM.render(<App />, div);
ReactDOM.unmountComponentAtNode(div);
);
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
```

```
ReactDOM.render(<App />, document.getElementById('root'));
```

```
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

index.css

```
#body
margin: 0;
font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
"Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
```

```
#code
font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
monospace;
```

Anamnesi.js

```
import React from 'react';
import "../stile.css"
```

```
class Anamnesi extends React.Component
constructor(props)
super(props);
this.state = value: "";
```

```
this.handleChange = this.handleChange.bind(this);
this.handleSubmit = this.handleSubmit.bind(this);
```

```
handleChange(event)
this.setState(value: event.target.value);
```

```
handleSubmit(event)
alert('E' stata salvata la seguente nota: ' + this.state.value);
event.preventDefault();
```

```
render()
return (
<form onSubmit=this.handleSubmit>
<label style=color:'48ccbf'>
<strong>this.props.label</strong>
<div>
<input id="Anamnesi" type="text" value=this.state.value
onChange=this.handleChange />
```

```
<input id='submit' type="submit" value=this.props.button />
</div>
```

```
    </label>
  </form>
);
```

```
export default Anamnesi;
```

```
Breadcrumbs.js import React from 'react';
import Typography from '@material-ui/core/Typography';
import Breadcrumbs from '@material-ui/core/Breadcrumbs';
import Link from '@material-ui/core/Link';
import App from '../App'
import ReactDOM from 'react-dom';
import "../style.css"
```

```
    class Breadcrumb extends React.Component
    constructor(props)
    super(props);
    this.handleClick = this.handleClick.bind(this);
```

```
    handleClick()
    ReactDOM.render(<App />, document.getElementById('root'))
```

```
    render()
    return(
    <Breadcrumbs id='breadcrumbs' aria-label="breadcrumb">this.props.title
    <Link color="inherit">
    Patient
    </Link>
    <Link color="inherit" onClick=this.handleClick>
    Sara Malini
    </Link>
```

```

<Typography><strong>this.props.title</strong></Typography>
</Breadcrumbs>
);

```

```

export default Breadcrumb;

```

Button.js

```

import React from 'react';
import "../style.css"

```

```

    //Sto usando una class component//
class Button extends React.Component
constructor(props)
super(props);
this.message_to_users = this.message_to_users.bind(this);

```

```

message_to_users()
alert('Doctor inserts new visit');

```

```

    render()
return(
<div>
<button id='NewVisit'
onClick=this.message_to_users>this.props.title</button>
</div>
);

```

```

export default Button;

```

Calendario.js

```

import 'date-fns';
import React from 'react';
import Grid from '@material-ui/core/Grid';
import DateFnsUtils from '@date-io/date-fns';
import
MuiPickersUtilsProvider,

```



```

KeyboardDatePicker,
from '@material-ui/pickers';

export default function MaterialUIPickers()
// The first commit of Material-UI
const [selectedDate, setSelectedDate] = React.useState(new
Date('2020-07-10T21:11:54'));
const handleDateChange = (date) =>
setSelectedDate(date);
;
return (
<MuiPickersUtilsProvider utils=DateFnsUtils>
<Grid container justify="space-around">
<KeyboardDatePicker
disableToolbar
variant="inline"
format="MM/dd/yyyy"
margin="normal"
id="date-picker-inline"
value=selectedDate
onChange=handleDateChange
KeyboardButtonProps=
'aria-label': 'change date',

/>

</Grid>
</MuiPickersUtilsProvider>
);

```

Collegamento_Home.js

```

import React from 'react';
import "../stile.css"
import ReactDOM from 'react-dom';
import App from '../App'

//Sto usando una class component//

```

```

class Collegamento extends React.Component
constructor(props)
super(props);
this.open_new_window = this.open_new_window.bind(this);
open_new_window()
ReactDOM.render(<App />, document.getElementById('root'))

```

```

    render()
return(
  <div>
    <button id='Back'
onClick=this.open_new_window>this.props.title</button>
  </div>
);

```

```

export default Collegamento;

```

Collegamento_Overview.js

```

import React from 'react';
import "../stile.css"
import Overview from './Overview'
import ReactDOM from 'react-dom';

//Sto usando una class component//
class Collegamento_Overview extends React.Component
constructor(props)
super(props);
this.open_new_window = this.open_new_window.bind(this);

open_new_window()
ReactDOM.render(<Overview />, document.getElementById('root'))

    render()
return(

```

```

<div>
<button id='Overview'> <img src=this.props.image
onClick=this.open_new_window/ > this.props.title < /button >
< /div >
);

```

```

export default Collegamento_Overview;

```

Collegamento.js

```

import React from 'react';
import "../stile.css"
import History from './History.js'
import ReactDOM from 'react-dom';

//Sto usando una class component//
class Collegamento extends React.Component
constructor(props)
this.open_new_window = this.open_new_window.bind(this);

open_new_window()
ReactDOM.render(<History />, document.getElementById('root'))

render()
return(
<div>
<button id='Overview'> <img src=this.props.image
onClick=this.open_new_window/>this.props.title</button>
</div>
);

```

```

export default Collegamento;

```

Component_Button.js

```

import React from 'react';
import "../stile.css"

```

```

    //Sto usando una class component//
class Component_Button extends React.Component
constructor(props)
super(props);

    render()
return(
<div>
<button id='Overview'> <img src=this.props.image
/>this.props.title</button>
</div>
);

```

```

export default Component_Button;
Conclusion.js

```

```

import React from 'react';
import "../stile.css"

```

```

class Conclusion extends React.Component
constructor(props)
super(props);
this.state = value: "";

```

```

this.handleChange = this.handleChange.bind(this);
this.handleSubmit = this.handleSubmit.bind(this);

```

```

    handleChange(event)
this.setState(value: event.target.value);

```

```

    handleSubmit(event)
alert('E' stata salvata la seguente nota: ' + this.state.value);
event.preventDefault();

```

```

    render()
  return (
    <form onSubmit=this.handleSubmit>
    <label style=color:'#48ccbf'>
    <strong>this.props.label</strong>
    <div>
    <input id="Conclusion" type="text" value=this.state.value
    onChange=this.handleChange />
    <input id='submit' type="submit" value=this.props.button />
    </div>

    </label>
  </form>
);

```

```

    export default Conclusion;
filter.js
import React from 'react';
import FormGroup from '@material-ui/core/FormGroup';
import FormControlLabel from '@material-ui/core/FormControlLabel';
import Checkbox from '@material-ui/core/Checkbox';

```

```

    export default function CheckboxLabels()
  const [state, setState] = React.useState(
    checkedA: false,
    checkedB: false,
    checkedF: false,
    checkedG: false,
  );

```

```

    const handleChange = (event) =>
  setState( ...state, [event.target.name]: event.target.checked );
;

```

```

    return (
    <FormGroup row>
    <FormControlLabel
    control=<Checkbox checked=state.checkedA onChange=handleChange
    name="checkedA" color='#48ccbf' />
    label="Week"
    />
    <FormControlLabel
    control=
    <Checkbox
    checked=state.checkedB
    onChange=handleChange
    name="checkedB"
    color='#48ccbf'
    />

    label="Month"
    />
    <FormControlLabel control=<Checkbox name="checkedC" color='#48ccbf' />
    label="3 Months" />

    <FormControlLabel
    control=<Checkbox checked=state.checkedG onChange=handleChange
    name="checkedG" color='#48ccbf' />
    label="Year"
    />

    </FormGroup>
    );

```

Form.js

```

import React from 'react';
import "../style.css"

class FormNome extends React.Component
constructor(props)

```

```

super(props);
this.state = value: ";

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);

    handleChange(event)
    this.setState(value: event.target.value);

    handleSubmit(event)
    alert('Éstata salvata la seguente nota: ' + this.state.value);
    event.preventDefault();

render()
return (
    <form onSubmit=this.handleSubmit>
    <input id="Form" type="text" value=this.state.value
    onChange=this.handleChange />
    <input id='submit' type="submit" value="Save" />
    </form>
);

export default FormNome;
Grafico.js
import React from 'react';
import Highcharts from 'highcharts/highstock';
import HighchartsReact from 'highcharts-react-official';
const options2=
title:
text: 'Glycemia'
,

yAxis:

```

```

title:
text: 'Value of Glycemia'

,

  xAxis:
categories: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
,

  legend:
layout: 'vertical',
align: 'right',
verticalAlign: 'middle'
,

  series: [
name: 'Glycemia [mg/dl]',
data: [90,100,89,95,87,100,101]
],

  responsive:
rules: [
condition:
maxWidth: 500
,
chartOptions:
legend:
layout: 'horizontal',
align: 'center',
verticalAlign: 'bottom'

]

const options =

```



```

    title:
    text: 'Pressure'
  ,

  yAxis:
    title:
    text: 'Value of Pressure'

  ,

  xAxis:
    categories: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
  ,

  legend: { layout: 'vertical', align: 'right', verticalAlign: 'middle' },

  series: [ { name: 'Systolic pressure [mmHg]', data: [100,130,110,120,115,119,132] },
    { name: 'Diastolic pressure [mmHg]', data: [80,90,57,58,62,73,85] } ],

  responsive: { rules: [ { condition: 'maxWidth: 500', chartOptions: { legend: { layout:
    'horizontal', align: 'center', verticalAlign: 'bottom' } } } ] }

  class Charts extends React.Component
  constructor(props)
  super(props);

  render()
  return(
    <div>
    <HighchartsReact
    highcharts=Highcharts

    options=options
  />
    <HighchartsReact
    highcharts=Highcharts

```

```

        options=options2
    />
</div>
);

```

```
export default Charts;
```

History.js

```

import React from 'react';
import Intestazione from './Intestazione';
import Form from './Form';
import Anamnesi from './Anamnesi';
import PrintIcon from '@material-ui/icons/Print';
import Calendario from './Calendario.js'
import Pression from './Pression'
import Collegamento from './Collegamento_Home'
import './stile.css'
import Breadcrumbs from './Breadcrumbs'

```

```

    class History extends React.Component
    constructor(props)
    super(props);
    this.state =
    stato: 'Benvenuto!'

;
render()
return(
<div>
<Intestazione button='SAVE'/>
<div id='contenitore2'>
<Breadcrumbs title='History'/>
<Form/>
<Calendario/>
<Anamnesi label='General History:'button='SAVE'/>
<Anamnesi label='Cardiovascular History:'button='SAVE'/>
<Anamnesi label='Recent History:' button='SAVE'/>

```

```

    <div style=color:'48ccbf'><strong>Sistolic pressure mmHg:</strong>
    <Pression/>
  </div>
  <div id='layout'>
    <strong>Diastolic pressure mmHg:</strong>
    <Pression id='layout' />
  </div>

```

```

    <div>
    <Anamnesi label='Conclusion' button='SAVE' />
    <button id="print"><PrintIcon/></button>

```

```

    </div>
  <div>
    <Anamnesi label='Therapy' button='SEND' />
    <button id="print"><PrintIcon/></button>
  </div>
  <Collegamento title="Back" />

```

```

  </div>

```

```

  </div>

```

```

);

```

```

  export default History;

```

Intestazione.js

```

import React from 'react';
import "../stile.css"
import logo from './logo.svg';
import Button from './Button'

```

```

  class Intestazione extends React.Component
  constructor(props)
  super(props);
  this.state =

```

```

    stato: 'Benvenuto!'
    ;

    render()
    return(
      <div id="intestazione">
        <div id="logo">
          <img src=logo alt="logo" />
        </div>
        <div id='text'>Web Platform</div>
        <Button id="NewVisit" title='+ New Visit' />
      </div>

    );

    export default Intestazione;

```

Overview.js

```

import React from 'react';
import Intestazione from './Intestazione';
import Breadcrumbs from './Breadcrumbs';
import Form from './Form';
import Filter from './filter.js';
import Calendario from './Calendario.js';
import './stile.css';
import Grafico from './Grafico';

function Overview()
return (
  <div>
    <Intestazione button='SAVE' />
    <div id='contenitore2'>
      <Breadcrumbs title='Overview' />
      <Form />
      <div id='filter'>
        <div style=color:'#48ccbf'><strong>Filter search </strong></div>
        Display data patient by:

```

```

<Filter/>
<Calendario/>
</div>
<div>
<Grafico/>
</div>

</div>
</div>

```

```
export default Overview;
```

Pression.js

```

import React from 'react';
import clsx from 'clsx';
import makeStyles from '@material-ui/core/styles';
import OutlinedInput from '@material-ui/core/OutlinedInput';
import InputAdornment from '@material-ui/core/InputAdornment';
import FormHelperText from '@material-ui/core/FormHelperText';
import FormControl from '@material-ui/core/FormControl';

```

```

const useStyles = makeStyles((theme) => (
root:
display: 'flex',
flexWrap: 'wrap',
, margin:
margin: theme.spacing(1),
,
withoutLabel:
marginTop: theme.spacing(3),
, textField:
width: '25ch', ,
));

```

```

export default function InputAdornments()
const classes = useStyles();
const [values, setValues] = React.useState(

```

```

    amount: "",
    password: "",
    weight: "",
    weightRange: "",
    showPassword: false,
  );
  const handleChange = (prop) => (event) =>
    setValues( ...values, [prop]: event.target.value );
  ;
  return (
    <div className=classes.root>

      <FormControl className=clsx(classes.margin, classes.textField)
        variant="outlined">
        <OutlinedInput
          id="outlined-adornment-weight"
          value=values.weight
          onChange=handleChange('weight')
          endAdornment=<InputAdornment position="end">mmHg</InputAdornment>
          aria-describedby="outlined-weight-helper-text"
          inputProps=
            'aria-label': 'weight',
          labelWidth=0
        /> <FormHelperText
          id="outlined-weight-helper-text">Pression</FormHelperText>
        </FormControl>

      </div>
    );
Recorded_Events.js
import React from 'react';
import "../style.css"

//Sto usando una class component//
class RecordedEvents extends React.Component
  constructor(props)
  super(props);

```

```

    render()
return(
<div>
<button id='RecordedEvents'><img src=this.props.image
/>this.props.title</button>
</div>
);

```

```

    export default RecordedEvents;
Update__allegati.js
import React from 'react';
import makeStyles from '@material-ui/core/styles';
import Button from '@material-ui/core/Button';

```

```

    const useStyles = makeStyles((theme) => (
root:
' > *':
marginLeft:'305px'
,
,
input:
display: 'none',
, ));

```

```

    export default function UploadButtons()
const classes = useStyles();
return (
<div className=classes.root>
<input
accept="image/*"
className=classes.input
id="contained-button-file"
multiple
type="file"
/>
<label htmlFor="contained-button-file">

```

```
<Button variant="contained" color="48ccbf" component="span">
Upload
</Button>
</label>

    </div>
);
```


Bibliography

- [1] S. Prete, “Design of a web application for patient monitoring after coronary angioplasty,” 2019.
- [2] V. S. Silvia Selvaggi, *TELEMEDICINA - approccio multidisciplinare alla gestione dei dati sanitari*. Springer Verlag Italia.
- [3] Ricerca. [Online]. Available: <https://www.pphc.it/telemedicina-in-italia-non-e-mai-partita-davvero-e-oggi-il-covid-19-svela-la-debolezza-del-nostro-paese/>
- [4] Ischemic. [Online]. Available: <https://www.humanitas.it/malattie/cardiopatia-ischemica>
- [5] Atrial. [Online]. Available: <https://www.my-personaltrainer.it/salute/fibrillazione-atriale.html>
- [6] Flutter. [Online]. Available: <https://www.my-personaltrainer.it/salute/flutter-atriale.html>
- [7] HTML. [Online]. Available: <https://www.html.it/guide/react-la-guida/>
- [8] CSS. [Online]. Available: <https://www.html.it/guide/guida-css-di-base>
- [9] JAVASCRIPT. [Online]. Available: <https://www.html.it/guide/guida-javascript-di-base>
- [10] JQuery. [Online]. Available: <https://www.html.it/guide/guida-jquery/>
- [11] AngularJS. [Online]. Available: <https://www.html.it/guide/guida-angularjs/>
- [12] React. [Online]. Available: <https://it.reactjs.org/>
- [13] material-ui. [Online]. Available: <https://material-ui.com/>
- [14] regulation. [Online]. Available: https://it.wikipedia.org/wiki/Regolamento_generale_sulla_protezione_dei_dati

- [15] article. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>
- [16] dispositivi. [Online]. Available: http://www.salute.gov.it/portale/temi/p2_4.jsp?area=dispositivi-medici,
- [17] ddm. [Online]. Available: https://it.wikipedia.org/wiki/Direttiva_CEE_3/42_sui_dispositivi_medici,
- [18] art2. [Online]. Available: http://www.salute.gov.it/imgs/C_17_pagineAree_1636_listaFile_itemName_1_
- [19] ce. [Online]. Available: <http://www.marcatura-ce.com/2364-2/>,
- [20] redux. [Online]. Available: <https://www.html.it/guide/redux-la-guida/>,
- [21] Disease. [Online]. Available: <http://www.salute.gov.it/portale/donna>
- [22] data. [Online]. Available: <https://www.cybersecurity360.it/legal/gdpr-e-ricerca-scientifica-come-e-quando-trattare-i-dati-sanitari>,