Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2012

Edgar Alexandre Silva Antunes

Plataforma Tele-ECG para dispositivos móveis
Tele-ECG platform for mobile devices

Universidade de Aveiro Departamento de
2012 Electrónica, Telecomunicações e Informática

Edgar Alexandre Silva Antunes

Plataforma Tele-ECG para dispositivos móveis
Tele-ECG platform for mobile devices

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor Carlos Manuel Azevedo Costa (professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro).

agradecimentos

Gostaria de agradecer a todos os meus amigos e familia pelo apoio incondicional.
Um agradecimento especial ao Professor Carlos Costa por me guiar neste trabalho e ao Doutor José Ribeiro por partilhar o seu conhecimento médico no tema.

acknowledgements

I would like to show my gratitude to all my friends and family for the unconditional support.
A special thanks to Professor Carlos Costa for guiding me in this work and to Doctor José Ribeiro for sharing medical insights on the subject.

palavras-chave ECG, Cardiologia, Telemedicina, Mobilidade, Privacidade, Segurança, HTML5

resumo

Nos dias de hoje, os sistemas Tele-ECG têm ganho cada vez mais importância, permitindo melhorar principalmente a qualidade de serviço prestado aos utentes de saúde que sofrem de doenças cardiovasculares. Estes sistemas permitem reduzir o tempo de interpretação de um ECG ao fornecer os requisitos necessários para que um cardiologista efetue a sua análise de forma remota, melhorando a capacidade de resposta em pequenas cidades, áreas remotas ou países em desenvolvimento; locais estes onde não existe, frequentemente, pessoal qualificado disponível para realizar a tarefa.

Esta dissertação foca-se no estudo da área emergente da computação móvel, mais especificamente na análise de diferentes tecnologias de desenvolvimento multi-plataforma capazes de atingir eficazmente as várias frentes do atualmente segmentado mercado móvel. Numa fase posterior é apresentada e discutida a implementação de uma estação de visualização de ECG para dispositivos móveis através da utilização de uma das tecnologias previamente discutidas, assegurando sempre os aspetos de segurança e confidencialidade inerentes ao manuseamento de dados clínicos. Esta estação permitirá reduzir substancialmente o tempo de resposta em situações de emergência médica ao tirar partido da disponibilidade de cardiologistas prontos a realizar uma interpretação clínica 24 horas por dia, independentemente da sua localização, através de um dispositivo móvel (smartphones, tablets, etc.).

keywords ECG, Cardiology, Telemedicine, Mobility, Privacy, Security, HTML5

abstract

Tele-ECG systems have gained an extreme importance in the last decade, making it possible to increase the quality of service provided to health care patients with cardiovascular diseases. These systems outperform a regular and traditional ECG analysis by reducing the response time in small villages, remote locations or in Third World countries; places that frequently lack of qualified professionals to accomplish such tasks.

This dissertation focuses on the study of the ever-emerging mobile computing field, where a thorough analysis is presented regarding various cross-platform development technologies, capable of targeting effectively all the major platforms currently available in the today's highly segmented mobile market. Furthermore, this work presents an implementation of a mobile station with the capability of visualizing and analyzing an ECG, yet always assuring all the confidentiality and security aspects that should be taken into account while handling clinical data. This station will lead to a significant improvement of the response time in medical emergencies, mainly as a consequence of the constant availability of a group of cardiologists, regardless of their location, at a distance of a mobile device (smartphones, tablets, etc.).

Contents

List of Figures

List of Tables

Acronyms

aECG	Annotated Electrocardiogram. 17, 18
AES	Advanced Encryption Standard. 38, 55
API	Application Programming Interface. 2, 11–13, 30, 33, 36, 38, 43, 44
APK	Android Package File. 12
CEN	Comité Européen de Normalisation. 17
CSS3	Cascading Style Sheets 3. 2, 7, 11, 13, 40, 41, 50, 51, 55, 56
ECG	Electrocardiogram. 1–3, 15–23, 25, 26, 28–31, 33, 35–41, 43–46, 50, 53, 55, 56
GPS	Global Positioning System. 25, 55
HTML	Hypertext Markup Language. 13
HTML5	Hypertext Markup Language version 5. 2, 3, 7, 11, 13, 14, 33, 35, 40–42, 44, 47, 50–52, 55, 56
HTTP	Hypertext Transfer Protocol. 38
HTTPS	Hypertext Transfer Protocol Secure. 27, 34, 35, 55
IDE	Integrated Development Environment. 35
IEETA	Instituto de Engenharia Electrónica e Telemática de Aveiro. 19
IIS	Internet Information Services. 35
IMAP	Internet Message Access Protocol. 27, 28, 35, 37, 38, 41, 49, 55, 56
JSON	JavaScript Object Notation. 42
NDK	Native Development Kit. 7
PDF	Portable Document Format. 39, 47
POP3	Post Office Protocol Version 3. 35

QR Code Quick Response Code. 37, 38, 41

R & D Research and Development. 17

SCP-ECG Standard Communications Protocol of Computerized Electrocardiography. 17

SDK Software Development Kit. 8, 12

SMTP Simple Mail Transfer Protocol. 27, 28, 35, 37, 41, 42, 49, 56

SOAP Simple Object Access Protocol. 36, 49, 56

SSL Secure Socket Layer. 28, 37, 56

UI User Interface. 7, 9, 14, 45

UID Unique Identifier. 38, 39

URL Uniform Resource Locator. 35, 37, 38, 52

VPN Virtual Private Network. 35

XAML Extensible Application Markup Language. 10

XML Extensible Markup Language. 7, 9, 10, 18, 35

Chapter 1

Introduction

The mobile market has suffered severe changes in the last years, and mobile devices are inevitably becoming the choice of common users as personal computers. The number of functions handled by these devices is an ever-increasing value, and every day different areas take more advantage of the mobile computing field. One example is the growing integration of health care systems with Internet platforms, and, more recently, mobile devices. However, the current state of the mobile market is unideal for mobile development, because the number of mobile operating systems competing with each other is significant and the market is fragmented.

For a mobile solution to aim at the major community of mobile users, several native mobile applications would have to be implemented, so that all the relevant mobile operating solutions would be contemplated, such as Apple's iOS running on iPhones and iPads, Google's Android running on many vendor specific like Samsung, LG, HTC, Sony Ericsson; Windows phone, etc.. For each implementation, developers would have to use platform specific tools and different API calls in order to achieve a similar application [5]. This can lead to the increase of development costs and bigger development cycles. Moreover, the user experience from device to device will vary significantly since some specific mobile operating devices will receive a bigger focus. This demand has lead to the appearance of cross-platform mobile solutions. These solutions focus on the development and maintenance of a single codebase that will adaptively work on most mobile devices, and aims to explore already known web languages without making pressure on developers to learn new languages like Objective-C and JAVA.

This work presents the creation of a platform for mobile devices that will give the ability to visualize Electrocardiograms (ECGs) and improve the health care system worldwide, more specifically in the analysis and treatment of data related to Electrocardiograms in the Cardiology field. This integration between Telemedicine and mobile computers impacts into the creation of a Tele-ECG system, that has proven to be a relevant and positive asset for patients with cardiovascular diseases. These systems allow the rapid communication and remote interpretation of Electrocardiograms which can be specially important in isolated regions or medical emergencies. Nowadays, common acquisition equipments for these medical examinations that output the results in a digital format are relatively cheap, which opens relevant opportunities to improve the quality of service in small villages, remote locations or in Third World countries; places that frequently lack of qualified professionals to accomplish such tasks. All the aspects described above provide a significant reason for the creation of systems that collect ECGs in digital format and give the possibility to analyze these exams on any mobile device anywhere in the world, by qualified personnel.

Although the mentioned systems bring a great amount of advantages both to patients and health care professionals, some concerns should be taken when using these platforms on real use cases. Firstly, the platform will interact with sensitive medical data that should be properly secured and protected. Secondly, the veracity of the medical reports built by the platform should present in a distinctive way the identity of the professional that performed the analysis. These and other difficulties will be address during this thesis.

1.1 Goals

This dissertation presents a study of the current state of mobile frameworks, where the main advantages and disadvantages are presented for each platform. Platforms such as PhoneGap [6], web solutions based on Hypertext Markup Language version 5 (HTML5), Cascading Style Sheets 3 (CSS3) and Javascript, and even native iOS and Android solutions; will be thoroughly discussed, compared and evaluated. Also, this case study serves as a key point for other research works that intend to develop a mobile solution with very specific requirements. Depending on what's desired, some frameworks may present major advantages over others and vice-versa.

In this context, the main goal of this dissertation is the development of a mobile platform, named Cardiobox mobile, which enables cardiologists around the world to access, view and analyze Electrocardiograms, also known as ECGs. Moreover, the performed study will be confronted with Cardiobox's requirements and one of the technological solutions presented will be chosen to aim the implementation of a mobile platform for performing medical analyses and capable of generating reports of ECGs. This work is motivated by the lack of existing applications that facilitate the visualization of ECGs on devices other than common computers. The developed platform should tackle and overcome the current interoperability issues between different mobile operating systems.

At the end of this thesis, all mobile devices with Internet and an HTML5 capable browser will have access to a publicly deployed platform for visualizing ECGs, through the usage of a secure web service that should provide an Application Programming Interface (API) for:

- Performing the configuration and importation of the required settings to connect to an ECG central;
- Executing authentication operations to successfully confirm the authenticity of all the users of the platform;
- Listing pending medical analyses available on the server;
- Querying the server for information about a specific exam, such as patient's information and the Electrocardiogram curves in question;
- Building a pdf document that will be seen as a medical report, made by cardiologists with the help of the information retrieved from the analysis made on mobile devices.

1.2 Thesis outline

This dissertation is organized in 5 chapters. The following is a brief description each of those chapters:

- **Chapter 2:** Sums up the current state of the mobile computing field and contextualizes the medical scenario that this thesis is inserted. Moreover, a critical analysis is made to similar platforms available for Android and iOS mobile operating systems.
- **Chapter 3:** Details this dissertation's requirements. Firstly, user requirements are tackled, followed by the analysis of functional and non-functional requirements are discussed, such as performance necessities, mobility issues and the platform independency factor.
- **Chapter 4:** Describes the architecture that is proposed to address the referred problems and proceeds to the implementation details. An architecture extension is proposed through the implementation of both a web service and a web application. The web service creates a way to securely interact with ECG centrals and the web client developed in HTML5 is described with core features of Cardiobox's platform. Finally, a global perspective over the implemented system is displayed.
- **Chapter 5:** Presents the key point of the developed platform, summarizing the main contributions and pointing out further work.

Chapter 2

State of the Art

2.1 Mobile Computing

The mobile computing field has gained a drastic relevance over the last decade. Not so long ago, mobile phones were heavy cases that would perform calls by roughly being able to connect to the cellular network infrastructure. Nowadays, after the explosion of the Internet, our smartphones bring us access to media content, significant services like email, banking, etc. However, the propagation of these devices has led to the competition for the mobile market between all the most relevant companies in the technology field, such as Microsoft, Google and Apple. This eager movement to control a new market inflicted major limitations for developers, since most companies built their own ecosystem and platform specific development tools [9]. Developers find it extremely hard to develop an application that could aim all mobile devices without having the limitations of implementing the same application as many times as the number of mobile operating systems they want to support. Fortunately, most mobile devices have a variable in common, which is the Internet. These devices usually come with a browser for accessing resources on the Internet, making it somehow a universal application for developers to write once and run anywhere. Thus, multi-platform frameworks had to be build in order to ease the development process for all devices.

In section 2.1.1 an overview of the mobile market share will be presented, comparing all the major mobile solutions nowadays. Moreover, in section 2.1.2, native mobile operating solutions will be analyzed and discussed, and its advantages and disadvantages will be compared with multi-platform solutions detailed in section 2.1.3.

2.1.1 Mobile market

The last decade has led to the growth of several mobile platforms, being the main ones iOS, Android and Windows Phone (previously called Windows Mobile). However, the competition for the market share inflicted the creation of other platforms such as Samsung's Bada or RIM's Blackberry OS. Also, since 2007 with Apple's iPad, tablets started to defend a position in the market, in spite of presenting a great amount of similarities with smartphones, being the screen size the only major difference, since the same mobile operating systems are used.

In table 2.1 it is possible to perform a comparison between the most relevant operating systems between the first quarter of 2011 and the first quarter of 2012 [4]:

Operating System	1Q12 (Units)	Market Share 1Q12 (%)	1Q11 (Units)	Market Share 1Q11 (%)
Android	81,067.4	56.1	36,350.1	36.4
iOS	33,120.5	22.9	16,883.2	16.9
Symbian	12,466.9	8.6	27,598.5	27.7
Research in Motion	9,939.3	6.9	13,004.0	13.0
Bada	3,842.2	2.7	1,862.2	1.9
Microsoft	2,712.5	1.9	2,582.1	2.6
Others	1,242.9	0.9	1,495.0	1.5
Total	**144,391.7**	**100.0**	**99,775.0**	**100.0**

Table 2.1: Worldwide Smartphone Sales to End Users by Operating System in 1Q12 (Thousands of Units). [4]

Google's Android market share increase a total of 19.7%, making it the leader of mobile operating systems. Moreover, Symbian had a drastic decrease, from 27.7% to, approximately, 8.6%. Overall, table 2.1 illustrates that the number of mobile systems is increasing every year, thus increasing the necessity for common desktop frameworks and applications to become available on any device. Unfortunately and in spite of Android having more than half the market share, there are still traces that point that the market remains fragmented. With this rises the questions about which platform to choose when deciding upon building a certain system. Choosing a single platform leaves users from the other platforms limited, but choosing all has significant costs and increased the delay in the development process. Summarizing, Android currently holds the most significant mobile market share and presents a significant set of different low, mid and high end devices, as opposed to iOS which runs on a very limited list of devices. In section 2.1.2 all the major native and multi-platform frameworks will be discussed in order to weight its advantages and disadvantages.

2.1.2 Native mobile application platforms

The number of mobile application platforms is continuously growing but it is important to make a thorough analysis to each of these platforms in order to make a decision regarding the platform of choice for the development of a specific application. The first and most relevant types of mobile application platforms that should be address are the native platforms. These platforms represent most of the development today and present many advantages when compared to others. Unfortunately, the number of native application platforms has been increasing since new companies have been building their own system for the past years. In table 2.2 we can see the major native application platforms, along with their programming languages, as seen in [10]:

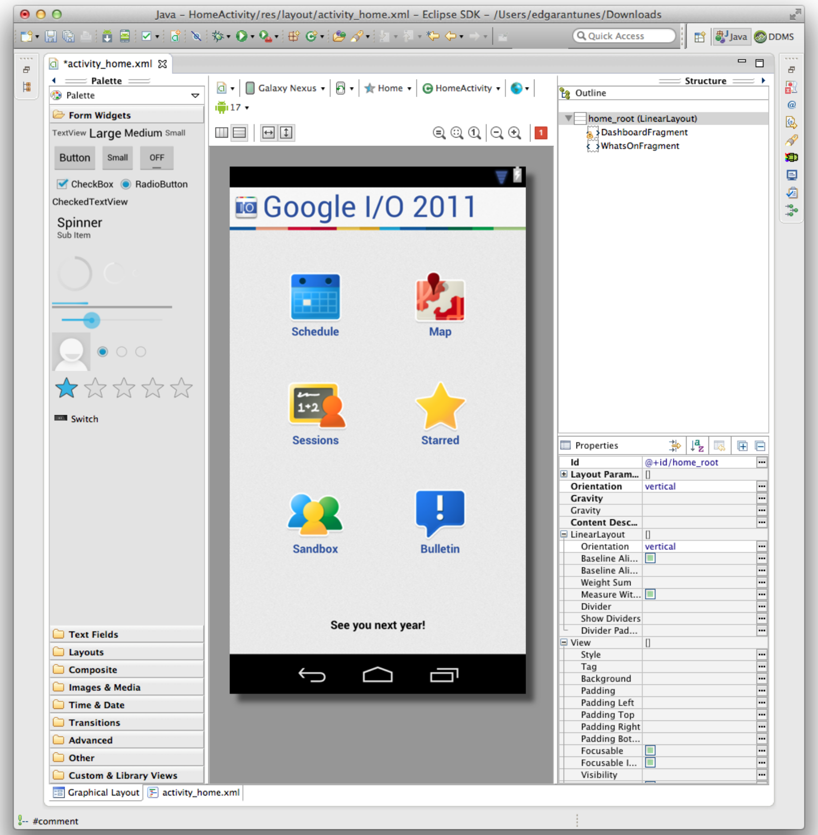
Platform	Programming Language(s)	Remarks
Android	Java, C, C++.	Open Source OS [11].
iOS	Objective-C, C	Introduced the mobile apps abstraction in the market. [12].
Windows 8	C#/VB.NET, C++, JavaScript	Desktop and mobile ecosystems integrated [13].
Windows Phone	C#/VB.NET	Silverlight, XNA frameworks [14].
Bada	C, C++.	Samsung's mobile platform running on Linux or RealTimeOS [15].
Blackberry	Java, Web Apps.	Java ME compatible, extensions enable tighter integration [16].
BlackBerry Tablet/-PlayBook OS (QNX)	ActionScript, C++, HTML5, CSS3, JavaScript	Blackberry 10 OS based on QNX [16,17].
Symbian	C, C++, Java, Qt, Web Apps, Others	Currently the longest running of all smartphone OSs [18].
webOS	HTML5, CSS3, JavaScript, C	Supports native HTML5 programming [19].

Table 2.2: Main native application platforms today.

Because of the mobile market segmentation, some native application platforms are more rich and have more content than others. For instance, Symbian was used around the world for a long time in mobile devices, making it the mobile operating system of choice for mid range and high range devices, such as Nokia mobile devices. However, the evolution and appearance of smartphones and tablets led to the need for a more integrated system, with Cloud services and with a better user experience.

Android

Android is an open source operating system developed by Google and presented in 2005. This operating system can be run on several smartphones, build from different manufacturers. Android applications are usually written in Java and works on top of an application framework called Dalvik. However, it is also possible to write applications using the Native Development Kit (NDK). The User Interface (UI) side of Android applications is usually built through the usage of Extensible Markup Language (XML). Recent versions of the eclipse plugin and the Android operating system improved the way user interfaces can be done and it is now possible to use a graphical user interface for managing objects on the screens, as illustrated in figure 2.1:



Id	@+id/home_root
Layout Param...	[]
Orientation	vertical
Gravity	
Gravity	
Content Desc...	
LinearLayout	[]
Orientation	vertical
Baseline Ali...	☐
Baseline Ali...	
Weight Sum	
Measure Wit...	☐
Divider	
Show Dividers	
Divider Pad...	
View	[]
Style	
Tag	
Background	
Padding	
Padding Left	
Padding Top	
Padding Right	
Padding Bot...	
Focusable	☐
Focusable I...	☐
Visibility	

Figure 2.1: Eclipse with the ADT plugin for managing UI screens in Android.

Moreover, with the explosion of the tablet market, Google improved its operating system and merged two distinct versions which led to the generation 4.0 (Ice Cream Sandwich). This version gives developers the ability to create both smartphone and tablet applications. The development environment relies on the Android Software Development Kit (SDK) [20] which gives the necessary tools for conducting the development of an application. Moreover, the Android SDK is available on all the major operating systems such as Windows, Linux or Mac OS X.

In order to distribute a native Android application, the developer should pay a fee of 25$ and perform the publish steps for making the application widely available through Google Play (former Android Market). There is also the possibility to share applications from outside Google Play, however the targeted user should accept the installation of applications from unknown sources. All these details build a good, easy to use, powerful and cheap development environment, thus being implicitly responsible for Android being the current market share leader.

Nevertheless, one of the main disadvantages of Android is known as fragmentation. Since Android is capable of being run on significantly different smartphones, from different manufacturers, with distinct hardware capabilities like different screen sizes, it hardens the implementation cycle since developers have to test its application on different mobile devices.

iOS

iOS is Apple's mobile operating system which was presented in 2007 with the first iPhone. This mobile operating system is based on Mac OS X. One of the disadvantages of developing native applications for iOS might be the initial learning curve. The programming language used for writing applications is the Objective-C, which is seen as a superset of the C language. This superset enables access for object oriented programming, thus approaching it to another object oriented languages for developing native mobile applications like Java for Android. However, Objective-C is an overall less known programming language while Java is used in a broader perspective for developing software, thus increasing the learning curve. Nevertheless, it is also possible to use C and C++ as programming languages in iOS.

Regarding the development environment, Apple provides an IDE called Xcode, along with an integrated set of tools such as iPhone and iPad emulators, making a consistent environment for developing both Mac OS X and iOS applications. This IDE has a great relevance while developing applications using Objective-C, since this programming language possesses a rather specific syntax. With this IDE, developers which are beginning to write applications in Objective-C take a great advantage of it through code completion and other aids. Unfortunately, Xcode is solely available on Mac OS X.

Former versions of Xcode had a standalone application called Interface Builder which was where the developer could build a UI through the usage of a specific syntax similar to XML. Currently Apple has integrated both applications and now Xcode comes with Interface Builder built into it. Also, since iOS 5 it is now possible to build Storyboards where the developer can create different screens and connect them, thus generating the flow of the application. Moreover, most UI tasks can now be done via simple drag-and-drop operations. An example of the Storyboard in iOS 5 along with Xcode is illustrated below, in figure 2.2:

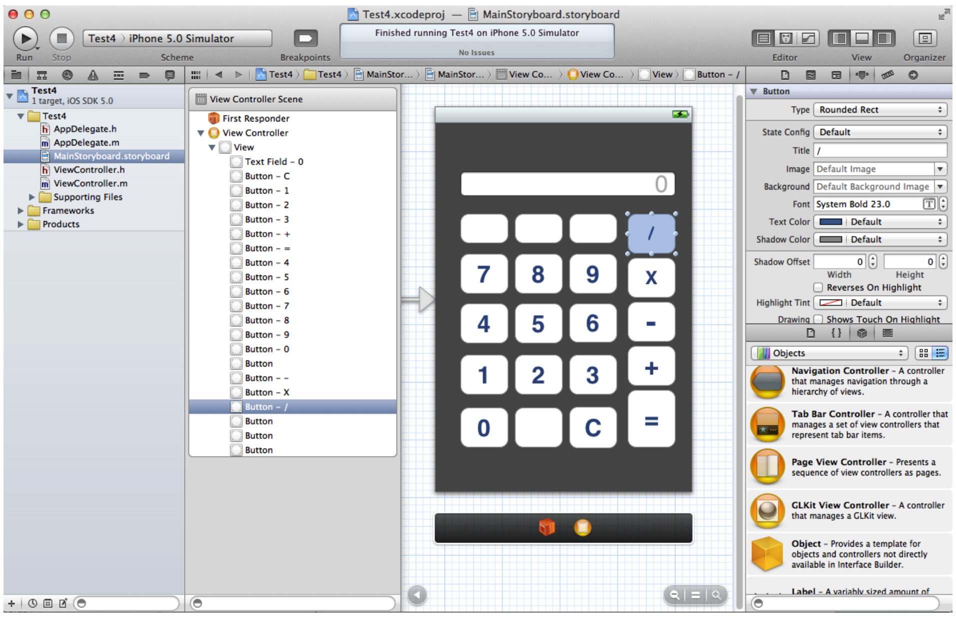

Test4.xcodeproj — MainStoryboard.storyboard
Run
Stop
Test4 › iPhone 5.0 Simulator
Scheme
Breakpoints
Finished running Test4 on iPhone 5.0 Simulator
No Issues
Editor
View
Organizer
Test4
1 target, iOS SDK 5.0
AppDelegate.h
AppDelegate.m
MainStoryboard.storyboard
ViewController.h
ViewController.m
Supporting Files
Frameworks
Products
View Controller Scene
First Responder
View Controller
View
Text Field - 0
Button - C
Button - 1
Button - 2
Button - 3
Button - +
Button - =
Button - 4
Button - 5
Button - 6
Button - 7
Button - 8
Button - 9
Button - 0
Button
Button - -
Button - X
Button - /
Button
Button
Button
0
/
7
8
9
x
4
5
6
-
1
2
3
+
0
C
=
Button
Type Rounded Rect
State Config Default
Title /
Image Default Image
Background Default Background Image
Font System Bold 23.0
Text Color Default
Shadow Color Default
Shadow Offset 0 0
Width Height
Reverses On Highlight
Highlight Tint Default
Drawing Shows Touch On Highlight
Objects
Navigation Controller - A controller that manages navigation through a hierarchy of views.
Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Object - Provides a template for objects and controllers not directly available in Interface Builder.
Label - A variably sized amount of

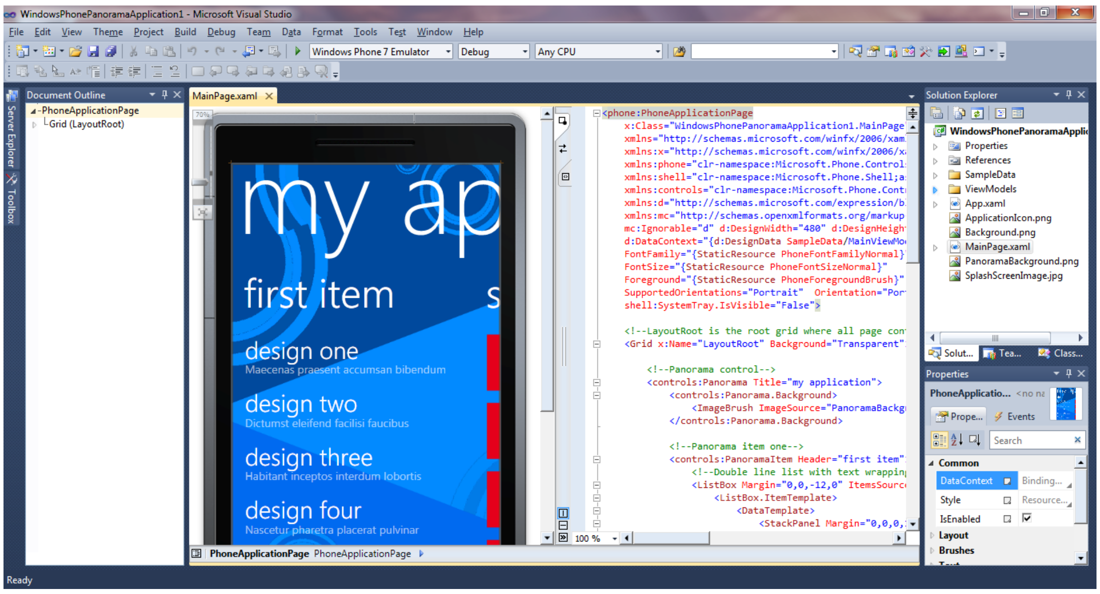
Figure 2.2: Apple's IDE for Mac OS X and iOS, called Xcode.

iOS applications are published on Apple's App Store, which is the only legal way to install applications on iOS devices such as iPhones, iPods or iPads. In order to access these features, the developer has an annual fee of 99$ to join and maintain a valid iPhone Developer Standard Program license.

Windows Phone

After Windows Mobile started loosing market share, Microsoft decided to improve its platform in order to compete with the major mobile operating systems, so Windows Phone was built. Windows Phone appeared with a simple an intuitive User Interface, formerly called Metro UI (more recently changed to Modern UI), which aim to give a great usability to the platform. Moreover, in 2011 Nokia selected Windows Phone as its primary platform, in order to increase Windows Phone's market share and stop the decreasing in the number of units sold per year with Symbian, as seen in section 2.1.1. However, Windows Phone does not currently old a significant amount of today's market share and previously stated.

Regarding the development process, Windows Phone applications are written in C# or VB.NET, using Microsoft's IDE called Visual Studio. Developers with previous knowledge in developing with Windows tools benefit a great advantage when writing Windows Phone applications since the environment is almost identical. For creating user interfaces, Microsoft uses Extensible Application Markup Language (XAML), a XML based language. Nevertheless, Visual Studio also lets developers build user interfaces the same way Android and iOS do, as illustrated in figure 2.3:



```
<phone:PhoneApplicationPage
    x:Class="WindowsPhonePanoramaApplication1.MainPage
    xmlns="http://schemas.microsoft.com/winfx/2006/xam
    xmlns:x="http://schemas.microsoft.com/winfx/2006/x
    xmlns:phone="clr-namespace:Microsoft.Phone.Control
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;a
    xmlns:controls="clr-namespace:Microsoft.Phone.Cont
    xmlns:d="http://schemas.microsoft.com/expression/b
    xmlns:mc="http://schemas.openxmlformats.org/markup
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeigh
    d:DataContext="{d:DesignData SampleData/MainViewMo
    FontFamily="{StaticResource PhoneFontFamilyNormal}
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"  Orientation="Por
    shell:SystemTray.IsVisible="False">

    <!--LayoutRoot is the root grid where all page con
    <Grid x:Name="LayoutRoot" Background="Transparent"

        <!--Panorama control-->
        <controls:Panorama Title="my application">
            <controls:Panorama.Background>
                <ImageBrush ImageSource="PanoramaBackg
            </controls:Panorama.Background>

            <!--Panorama item one-->
            <controls:PanoramaItem Header="first item"
                <!--Double line list with text wrappin
                <ListBox Margin="0,0,-12,0" ItemsSourc
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel Margin="0,0,0,
```

Figure 2.3: Microsoft's IDE for Windows, called Visual Studio.

A disadvantage of the Windows Phone operating system is its limited multitasking which stops all the applications that are not on the foreground. The only types of possible background processes are music playback and file transfer. The distribution is through Microsoft's ecosystem service, called Marketplace. Also, Windows Phone and iOS share a common limitation which is the fact that the development environment is solely available on its own major operating system, i.e., Microsoft's Visual Studio is exclusive for Windows while Apple's Xcode solely works on Mac OS X.

Native platforms discussion

Three of the main native mobile application frameworks were presented: Google's Android, Apple's iOS and Microsoft's Windows Phone. Each of these has a specific programming language and development environment. Moreover, while iOS and Android run on ARM architectures, meaning that a written application is converted to ARM instructions, Windows Phone was initially available only for x86 CPU architectures. However, Windows Phone 8 will support both x86 and ARM based architectures.

Regarding the initial learning curve, Objective-C's programming language seems to present a steeper step for initial developers since its syntax is different from most common object oriented programming languages. On the other side, Java and C# are commonly known among the software development community and presents less challenges when starting the mobile development process.

As seen in section 2.1.1, Android is the major mobile market share holder, making it the best decision if the intention is to write a native application and target more than half of the mobile market. Nevertheless, iOS also represents a popular solution because of Apple's ecosystem. Unfortunately, Microsoft's Windows Phone has yet to perform a significant market penetration.

Overall, the main advantages of native solutions are the full access to all software API and hardware capabilities and the creation of native user interfaces which increases the usability for its users. As main disadvantages, there is the initial learning curve, the limitations regarding the development environment and the deployment through each ecosystem.

Unfortunately, all of the above solutions require specific knowledge in each platform and in different programming languages. Moreover, it is not possible to develop an application to aim multiple operating systems with any of the presented solutions. This means that creating a mobile application with the desire to target all or most of the mobile market means that multiple applications should be developed, thus increasing the development cycle and the costs.

2.1.3 Cross-platform mobile application platforms

Due to the current mobile market fragmentation, several cross-platform mobile application platforms have been created in the past years. All of these solutions commonly target purposes such as reducing the development time during the implementation of a system, and the implementation of multiple native applications while using distinct development environments and programming languages. With these tools the developer writes once the application and its aim is to be run everywhere. Also, several cross-platform solutions try to take advantage of the developer Some cross-platform frameworks will be described regarding its platform support, hardware access limitations, etc. The details of Phonegap and HTML5 will be described and discussed.

Phonegap

Phonegap is a cross-platform framework for developing native applications through the usage of standard web technologies such as HTML5, CSS3 and Javascript [6]. Phonegap makes it possible for developers do build once and target all the major mobile operating systems such as Android, iOS, Windows Phone 7, Blackberry OS, Symbian, Bada, etc. Phonegap provided a Javascript API to the users that extends regular functionalities available in standard web applications. The final result is a native application, capable of being published on each ecosystem's application market, with a full screen web view to a mobile web application developed in web technologies.

The major advantages of this solution is the fact that web developers are capable of targeting mobile devices without the initial learning curve associated with all the native frameworks previously described. This means that the delay and costs associated with the development of an application are significantly smaller than with native solutions. Also, maintaining a single codebase is simpler and less error prone. Finally, depending on the type of the application, it could be useful to be able to deploy the application as a native application in the market, since it is easier to target a certain audience or even all mobile users, while with a mobile application users would have to find out through the internet.

Phonegap has developed an automated process for creating mobile applications, with integration with different IDEs and Javascript APIs, until the part of generating native applications (e.g., an Android Package File (APK) for Android). Recent changes in the build process led Phonegap to improve the usage of its platform by easing the developer to install native SDKs and just sending the code to Phonegap Build [1] and retrieving the final native applications, as illustrated in figure 2.4:

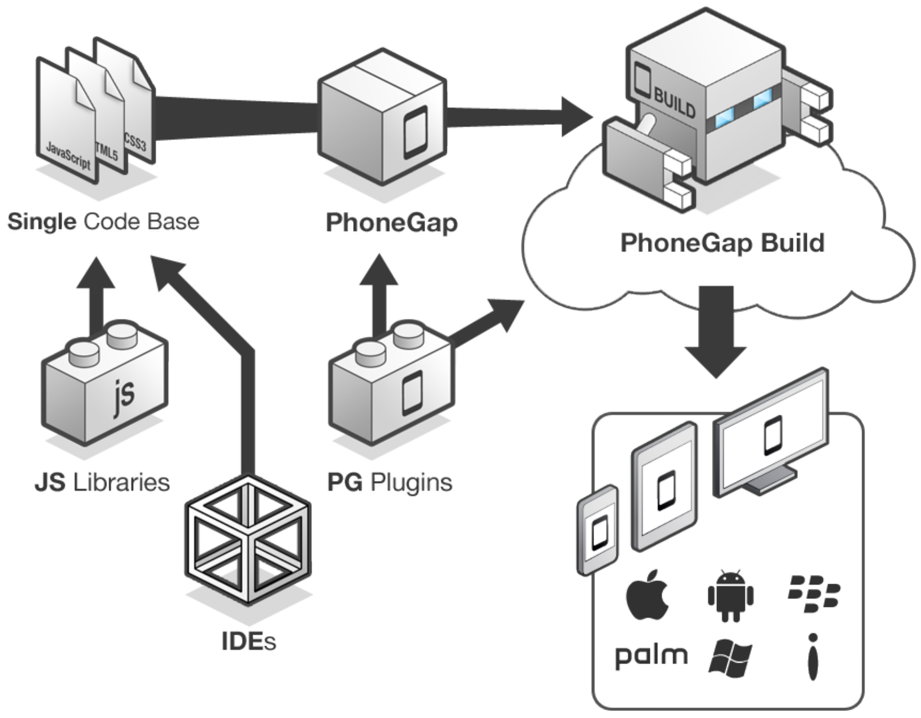


Figure 2.4: Phonegap's diagram explaining the process [1].

HTML5

Hypertext Markup Language (HTML) is a markup language used on the Internet for displaying information in a structured way. When the expression HTML5 appears, it represents the fifth version of HTML which is currently in development [21]. Among several new features, this new version supports tags for media content, such as <*audio*>, <*video*> and <*canvas*>. These new features have to be implemented by the browser in order for a website to be rendered correctly. One of the major advantages is the fact that these multimedia tags will eventually be standardized and become the solution of choice for most of the World Wide Web pages, because proprietary plugins can be replaced. Also, new elements such as the <*header*> or <*article*> were created in order to improve the semantic relevance in websites, as presented in figure 2.5

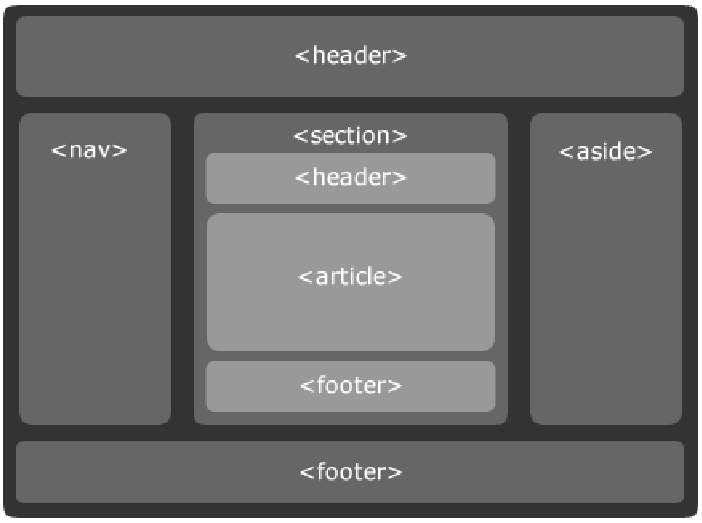


Figure 2.5: HTML5's new elements [2].

In spite of not being fully defined yet, HTML5 received a great amount of attention by some of the most important browsers and web developers started to take advantage of this. The <*canvas*> element became one of the most relevant new features since its API makes it possible do draw anything, from pictures to mouse or touch events.

Along with CSS3 media queries it became possible to create web applications with adaptive interfaces which would adjust itself to the screen size [22]. This represents one of the most relevant features regarding mobile web development due to the quantity of different screen sizes in today's market. Regarding the learning curve, it is possible that the developer already has knowledge about web technologies, making it easy to develop a mobile web application without having the need to switch to a different development environment or learning a new programming language.

Also, HTML5 when combined with other web programming languages such as CSS3 and Javascript, benefit from the same advantages as previous cross-platform solutions, like the single codebase. Moreover, while implementing a new feature or fix a problem on a native application implies build everything once again and passing through the publish process, changes to the mobile web application are seen immediately to its users.

As a downside, requirements that required accessing hardware capabilities are often limited but recent development made it possible for HTML5 mobile web applications to ask the user for its geolocation.

Finally, there are no fees regarding the deployment process, and an HTML5 mobile web application targets a great amount of the mobile market, since only mobile devices without an HTML5 capable browser will be prevented from utilizing the application.

2.1.4 Comparison

Depending on the required functionalities for a certain application, it might be useful to follow a cross-platform path and develop and maintain a single codebase while affecting most of today's mobile users. However, this decision is seen as a trade-off between user experience on the native side or deployment, development costs and time on the cross-platform side. If the application is expected to provide a native user interface, take advantage of specific UI patterns from each native platform or access hardware capabilities without any limitation, a native approach might be the best. On the other side, if the application's requirements are target as many mobile users as possible, decrease the deployment delays with a single codebase, etc., than cross-platform solutions are the answer.

2.2 Electrocardiograms: A brief explanation

An Electrocardiogram, usually abbreviated to ECG, is a non-invasive medical exam that analyzes the activity of the cardiac muscle. This exam has a graphical representation of the electric activity of the heart, thoroughly registered within a certain time interval. In normal conditions, ECG strokes have normal and predictable directions, durations and amplitudes. Due to this fact, these medical exams can be categorized as normal or abnormal [23]. Whenever the cardiac muscle contracts, electric pulses are generated and propagated through the patient's body until the body surface is reached. At the surface, it is possible to reproduce the rhythm of various contraction times, forming different but recognizable signals, that can be captured using an Electrocardiograph [24].

The standard ECG system comprises twelve curves, where each one captures the electrical activity of the heart in a different position of the body. Six of these standard derivations are captured in the patient's members forming a triangle (Einthoven's triangle) while the other leads, named precordial leads, are placed on the thoracic cage [3,25]. Figure 2.6 presents the standard ECG examination where the electrocardiograph is in contact with a patient's body in order to capture the twelve electrical pulses.

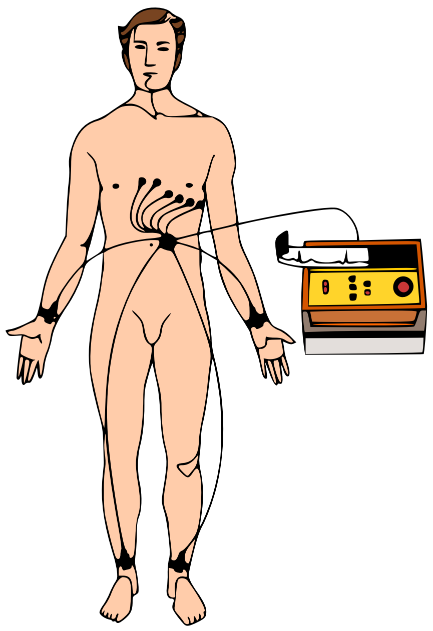
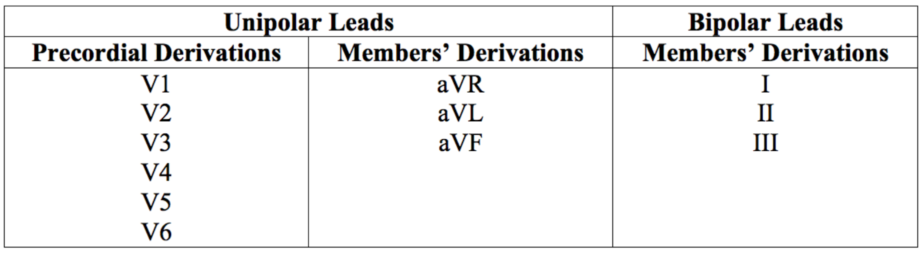


Figure 2.6: Diagram showing the connection of ECG leads, as seen in [3].

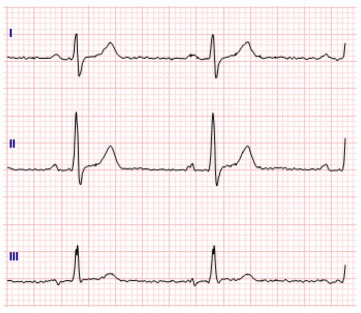
These twelve curves can be classified in bipolar leads when two electrodes are used to monitor the heart, or in unipolar leads which monitor the electrical activity between the lead's electrode and the center of the heart, as stated in figure 2.7.



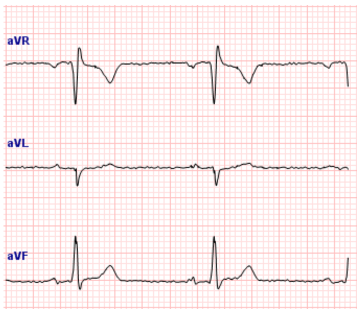
Unipolar Leads		Bipolar Leads
Precordial Derivations	**Members' Derivations**	**Members' Derivations**
V1	aVR	I
V2	aVL	II
V3	aVF	III
V4		
V5		
V6		

Figure 2.7: The twelve leads that constitute a standard ECG.

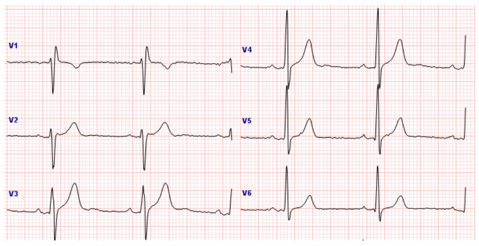
In figure 2.8 a graphical representation of the various ECG leads is presented like the digital format of a standard ECG [26, 27]. Moreover, this exam is also printed in m paper that is outputted from Electrocardiographs. Distances calculated in the horizontal axis of the millimetric paper represent the time, usually in seconds. On the other hand, the vertical axis is used to represent the signal's magnitude. In a usual calibration, one small block of paper represents a height of 0.1 mV and a weight of 40ms. Also, one main block of 5mm x 5mm represents five times more, which is 0.5 mV of height and 200ms of length.



(a) Bipolar leads by Einthoven.



(b) Bipolar leads by Goldberger.



(c) Unipolar leads by Wilson.

Figure 2.8: The twelve leads of an ECG.

After the exam is performed, Electrocardiographs with remote capabilities send the ECG to a central and compute some information regarding the medical exam, such as:

- Heart rate;
- Heart rhythm;

- Automatic ECG analysis that may conclude that certain abnormalities have been encountered in some of the electrical pulses captured.

In this context and because of the digital evolution, most electrocardiographs nowadays already capture the signals described above in a digital format. However, the explosion of this field led to the creation of different proprietary formats that may vary from machine to machine. Moreover, although some standard formats were created in order to ease the fragmentation of different formats, certain vendor specific implementations still persist and are presented in today's ECGs, as it will be seen on section 2.3.

2.3 Supported Electronic Formats

As previously described, most electrocardiographs nowadays already capture a digital abstraction of the ECG signals in digital format. Nevertheless, the specific rules associated with the format and storage of this data are proprietary and may vary from manufacturer. However, some standards were outlined in order to improve he interoperability between each system. The next subsections will present the most relevant characteristics of the main formats currently in use.

2.3.1 SCP-ECG

The electronic format Standard Communications Protocol of Computerized Electrocardiography (SCP-ECG) (Standard Communications Protocol of Computerized Electrocardiography) is based on a project called European AIM Research and Development (R & D) developed between 1989-1991. During this project, a new method for wave compression was invented, according to an extensive analysis of all the already existent ECG compression wave algorithms. This standard was approved in 1993 by Comité Européen de Normalisation (CEN) (European Committee for Standardisation) as a standard and implemented by a set of relevant manufacturers. Its usage proved the fact that this standard was usable for telemetry applications and for ECG data storage. However, the aimed flexibility for this standard with optional proprietary partial implementations could not keep the interoperability between different devices. In structural terms, SCP-ECG has a simple structure, destined to storing multi-segments of an ECG from a certain patient. Each ECG has several data sections where the first group is obligatory and the remaining optional.

2.3.2 aECG

The electronic format Annotated Electrocardiogram (aECG), which means annotated ECG, was the first attempt to create a digital standard for ECGs, and was introduced in 2001 [28]. In 2002, a group of advisors was gathered in order to specify, not only the codification details but also the electronic formats that were used at that time, so that the requirements for the new format could be traced. This group focused mainly on the way the data about each of the ECG curves would be codified and that most formats used would present this information along side with demographic details about the patient. It was concluded that one of the requirements would be that the new digital format should be flexible and abstracted for the notion of visualization of an ECG curve, and that there should exist a solid mechanism for annotations. The information related to a certain annotation is stored in one AnnotationSet which corresponds to a set of AnnotationActs. Each AnnotationAct has a group of annotations that are generated by a certain entity. This annotation mechanism would define five possible types [29]:

- Beat;
- Wave;
- Rhythm;
- Pacemaker;
- Noise.

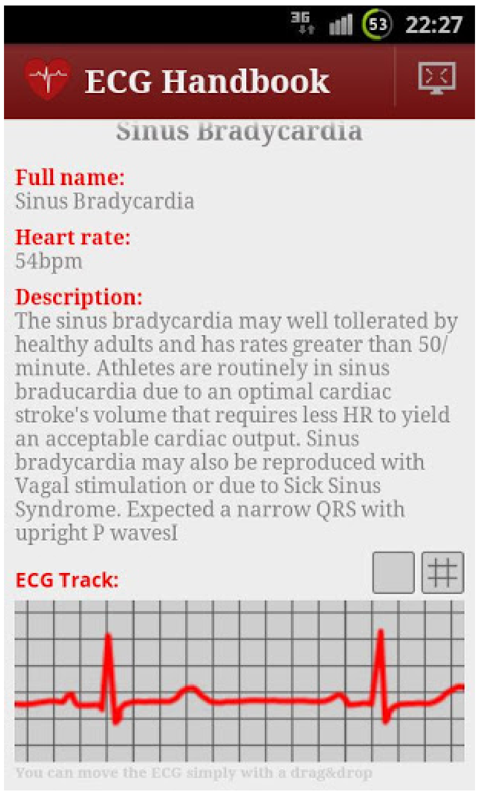
The annotation type is defined in the code property of the Annotation act.

2.3.3 Other formats

In spite of all the efforts previously stated regarding the development of standards that would improve the interoperability between manufacturers, a normalization of the ECG digital format was not achieved yet and many other formats are currently in use. For instance, another electronic format that takes advantage of XML encapsulation techniques is the ecgML format. This format presents the same type of encapsulation of aECG but possesses numerous structural differences. [30] The previously stated formats consist of the most recognized electronic formats for the representation of ECGs and make part of the main targets for a possible normalization of the format in the future. However there is an enormous quantity of other proprietary formats developed by specific manufacturers that are currently in use, such as the XML format from Mortara, one of the most important manufacturers, which will be used in this dissertation.

2.4 Related studies and applications

After a small research regarding the existence of possibly similar applications to what is intended to be this dissertation's final product, it became clear that other user agents have been developed in the past. After building a small set of keywords and performing different search on them in various marketplaces, some applications were found, mainly on the major markets such as Apple's ecosystem App Store and Google Play, formerly known as Android Market. A great amount of these applications would only work as a source of information about Electrocardiograms, such as *Electrocardiogram ECG Types* and *Heart ECG Handbook - Lite* [31,32] seen in Google Play, or *ECG* [33] as in App Store. These applications focus on providing information so that any regular reader can gain knowledge about ECGs, as it can been seen in Figure 2.9.



Sinus Bradycardia

Full name:
Sinus Bradycardia

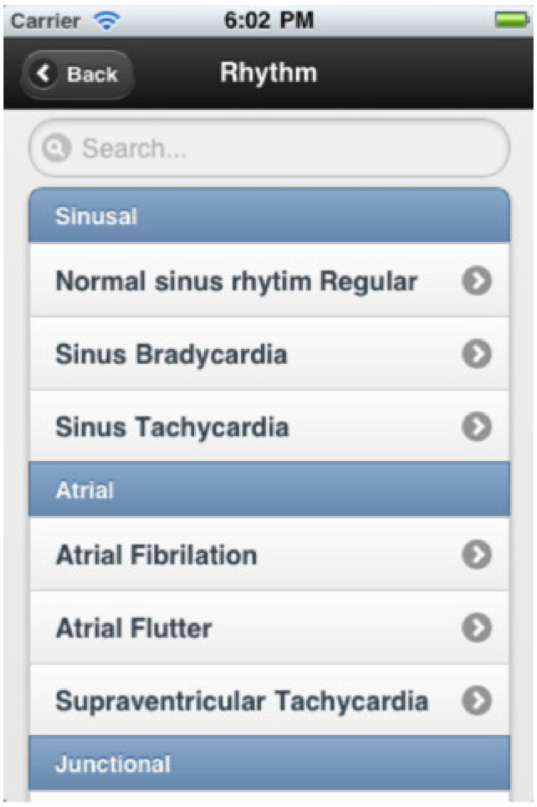
Heart rate:
54bpm

Description:
The sinus bradycardia may well tollerated by healthy adults and has rates greater than 50/ minute. Athletes are routinely in sinus braducardia due to an optimal cardiac stroke's volume that requires less HR to yield an acceptable cardiac output. Sinus bradycardia may also be reproduced with Vagal stimulation or due to Sick Sinus Syndrome. Expected a narrow QRS with upright P wavesI

ECG Track:

You can move the ECG simply with a drag&drop

(a) Heart ECG Handbook - Lite from Google Play.



Sinusal
- Normal sinus rhytim Regular
- Sinus Bradycardia
- Sinus Tachycardia

Atrial
- Atrial Fibrilation
- Atrial Flutter
- Supraventricular Tachycardia

Junctional

(b) ECG from App Store.

Figure 2.9: Two examples of applications related to ECG.

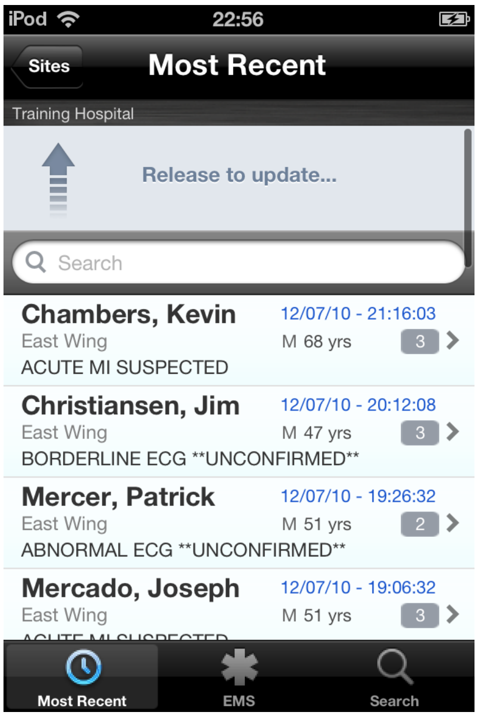
Some of them even develop quizzes so that readers can test their skills, but the number of applications focused on implementing standard ECG parsers and giving the user the ability to visualize a real ECG were almost nonexistent. One of those solutions, found on App Store, is capable of doing exactly what is intended. This solution comes from a company called Airstrip Technologies from Texas. [34] Unfortunately, their solution's business model relies on the fact that their application works exclusively with their hardware device, thus selling them together. Nevertheless, on subsection 2.4.1 a thorough analysis to the application will be presented so that some model decisions can be weighted before starting to implement Cardiobox mobile. Also, another relevant application will be presented and further analyzed in subsection 2.4.2, this time for Android, that was developed in the past by another student from the group of bioinformatics of the Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) department. [35]

2.4.1 AirStrip Cardiology

Airstrip Cardiology is an iOS native application available at the App Store that is part of an integrated solution for analyzing ECGs in mobile devices. It is said to be an integrated solution because this application works solely with an exclusive hardware ECG reader, and both are sold as a package. Fortunately the application is free and has a demo version, making it possible to explore and make a critical overview.

Regarding what platforms are compatible, one can clearly state that only iOS devices will be able to run it, and they need to have the version 4.2 or higher, which represents a small percentage of the overall mobile market.

Initially, the application shows a login form in order for the cardiologist to authenticate. After the login process, cardiologists are presented with a scrollable list of pendent ECGs, giving the possibility to search, refresh or select one of the ECGs, as seen on figure 2.10 part (a). The search option, although useful on many cases, won't make much sense on Cardiobox's paradigm since cardiologists won't know any of the patients' personal information such as their names. However the refresh feature is welcomed since new ECG can arrive to the central from time to time. When the user makes a selection, a new list is presented, but this time the cardiologist has access to an overview of all the analyses for that specific patient, making it possible to perform a comparison between two ECGs performed within a certain time window. Nevertheless, when the cardiologist selects one of the ECGs, it is finally presented with a perspective where the signals can be visualized. This perspective is presented in figure 2.10 part (b).



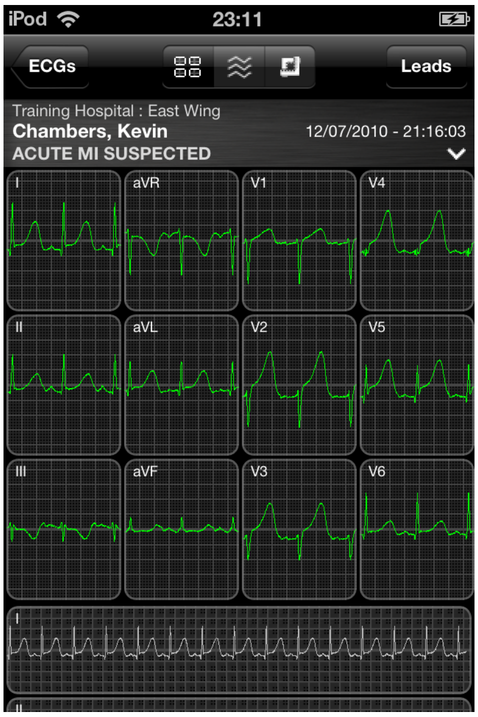
Chambers, Kevin 12/07/10 - 21:16:03
East Wing M 68 yrs 3
ACUTE MI SUSPECTED

Christiansen, Jim 12/07/10 - 20:12:08
East Wing M 47 yrs 3
BORDERLINE ECG **UNCONFIRMED**

Mercer, Patrick 12/07/10 - 19:26:32
East Wing M 51 yrs 2
ABNORMAL ECG **UNCONFIRMED**

Mercado, Joseph 12/07/10 - 19:06:32
East Wing M 51 yrs 3

(a) List of pending reports in CardioStrip.



(b) ECG Visualization tool in CardioStrip.

Figure 2.10: CardioStrip running on iOS.

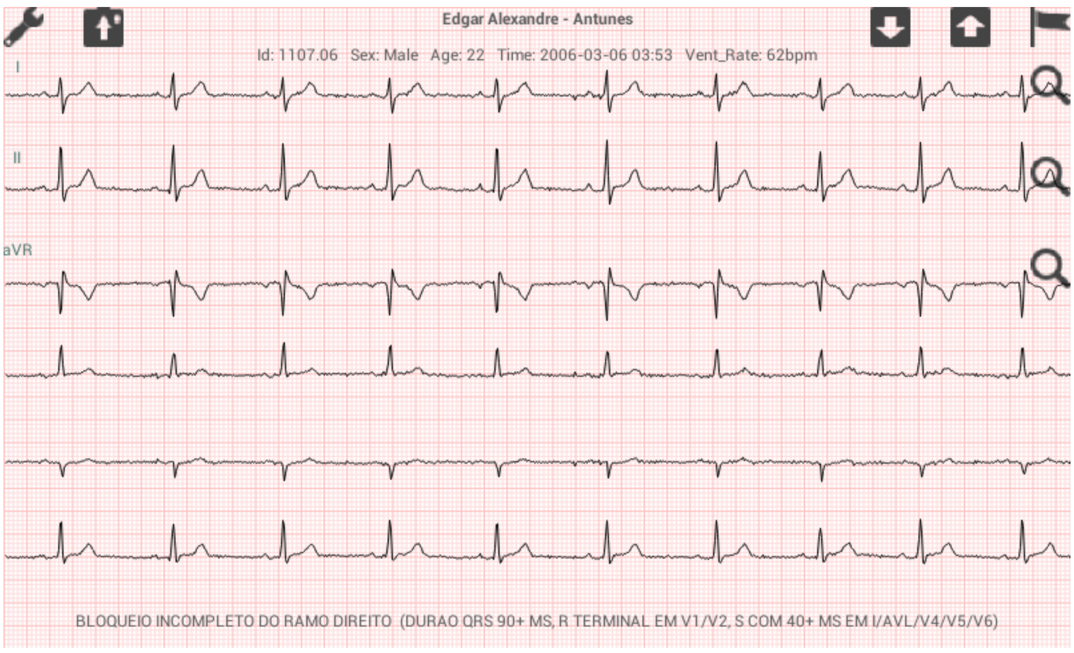
After a discussion with Dr José Ribeiro, co-supervisor of this dissertation, it was pointed out that the ability to see multiple curves simultaneously is a necessity for our application, as seen in AirStrip Cardiology. Also, a more detailed analysis of a specific curve is frequently needed; so it must be possible to perform synchronized zoom in/out operations on all curves simultaneously. It is required to show only one curve at a time at full screen as well. Moreover, although it is understandable to use dark colors to get better contrast underneath direct sunlight, natural ECG colors are more desired and ease the task of analyzing an ECG (cold colors, e.g., whites and creams). Finally, current iOS devices like iPod touch or iPhone present very limited screen sizes, thus increasing the importance of what information is displayed at all times, so the description bar with the patient's information should be avoided since it is not needed during the time of the analysis. However the cardiologist should always have the possibility to access said information.

Overall, this represents a professional solution for analyzing ECGs on mobile devices and all its features will be taken into consideration when developing Cardiobox mobile. It should be stated, however, that one of the greatest limitations of AirStrip Cardiology is the fact that it represents a native iOS only application, thus leaving a major part of the mobile market without a solution. With this, Cardiobox mobile will follow a slightly different path by being developed in a cross-platform solution.

2.4.2 ECG Viewer

The ECG Viewer, as opposed to AirStrip cardiology, is an Android application that lets the user analyze ECGs on Android capable devices. It shares one of the key points pointed out on the previous sections, and it is the fact that this is also a native application. Nonetheless, ECG Viewer represents a different way of interaction, by being a completely offline application. This means that cardiologists will be able to utilize the application without having any type of internet connection, which is a major advantage for users.

However, the application neither interacts with any central nor handles the hassle of retrieving ECGs for further analysis. So it is up to the cardiologist to manually download a set of ECGs for later analysis. Also, as shown in Figure 2.11, does not have a way to manage different ECGs, it is needed to crawl under the mobile device's files and select a proper file that would correspond to an ECG. Finally, there doesn't seem to exist any way to build and generate a medical report. In spite of all this, the application adjusts itself to the size of the screen and takes fully advantage of all the area available. Also, the decision on following ECG standards, regarding the colors, seems to support what was previously suggested.



Edgar Alexandre - Antunes

Id: 1107.06 Sex: Male Age: 22 Time: 2006-03-06 03:53 Vent_Rate: 62bpm

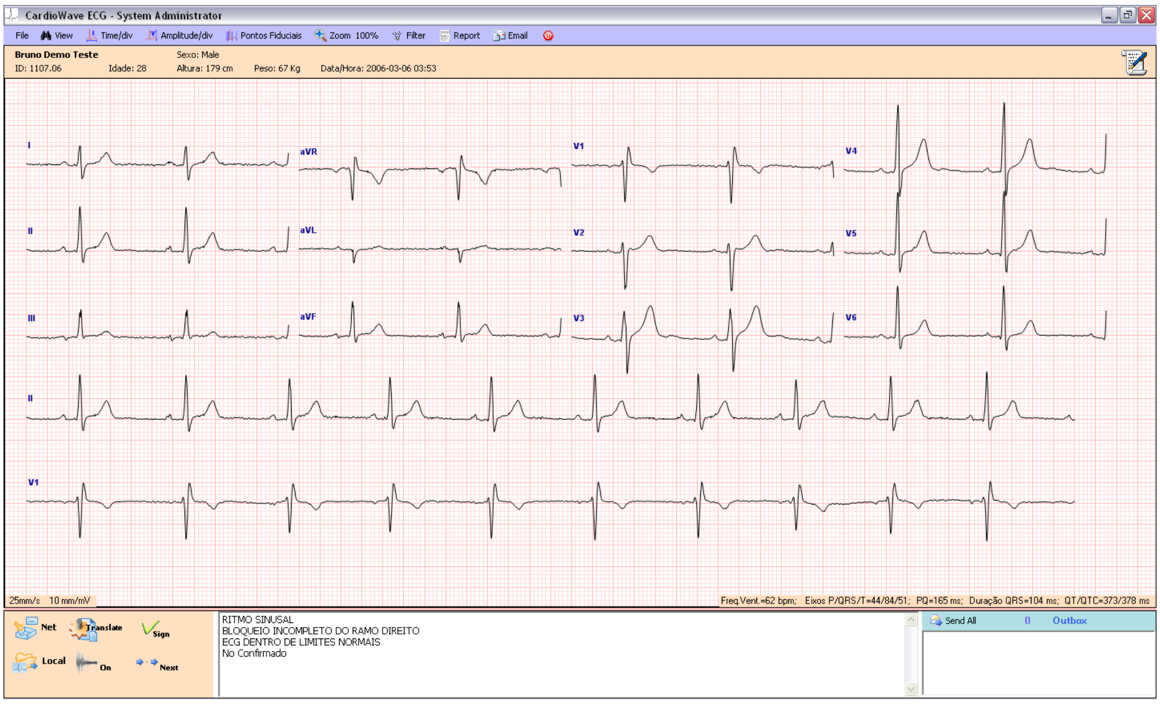
BLOQUEIO INCOMPLETO DO RAMO DIREITO (DURAO QRS 90+ MS, R TERMINAL EM V1/V2, S COM 40+ MS EM I/AVL/V4/V5/V6)

Figure 2.11: ECG Viewer application for Android.

2.5 Cardiobox

Cardiobox is a windows-based application capable of interacting, showing and analyzing Electrocardiograms on computer devices, and corresponds to the beginning point of this dissertation. This application lets cardiologists interact with different ECG centrals so that Electrocardiograms can be analyzed and a medical report generated. Cardiobox tries to fulfill a necessity in the Telemedicine field where certain countries might show a lack of qualified professionals capable of performing a thorough analysis to a cardiac exam such as the Electrocardiogram. Thus, Cardiobox can be seen as a medical utensil in the bioinformatics area where cardiologists can improve the quality of health care systems without being physically near the patient where the ECG is performed. The data sent by the manufacturers' devices is gathered in ECG centrals, where cardiologists are given the tool to see all the curves of each ECG and take a conclusion regarding a possible medical issue with the patient.

Cardiobox works as a decoupled solution where each ECG central is represented by a generic email account with pending ECG that are sent to be analyzed. In figure 2.12 the main perspective of the application can be seen, where the curves of a certain ECG are presented and the user has access to the inbox and outbox. It is also possible to generate a medical report and handle some adjustments in the visualization tool like changing the precision.



Bruno Demo Teste Sexo: Male
ID: 1107.06 Idade: 28 Altura: 179 cm Peso: 67 Kg Data/Hora: 2006-03-06 03:53

25mm/s 10 mm/mV

Freq.Vent.=62 bpm; Eixos P/QRS/T=44/84/51; PQ=165 ms; Duração QRS=104 ms; QT/QTC=373/378 ms

RITMO SINUSAL
BLOQUEIO INCOMPLETO DO RAMO DIREITO
ECG DENTRO DE LIMITES NORMAIS
No Confirmado

Figure 2.12: Main perspective of the original Cardiobox desktop application.

This type of architecture gives the overall system the possibility of interacting with different centrals seamlessly, and can be seen as a maintain-less structure since the persistence responsibility is delegated to the email service provided. However, with the recent explosion of mobile devices in our day to day life, there was a need to extend this architecture in order to obtain a solution that would both still work the same way for older users with older computers and attract and improve the usability issues related with the operating system and device characteristics limitations. This way Cardiobox mobile, as it will be seen later in this dissertation, will not try to replicate every functionality existent in the desktop version but will try to focus on a touch oriented, mobile device capable solution with the core functionalities that will give users the ability to interact with ECG centrals from its smartphones, tablets, etc.

Chapter 3

Requirements

The primary goal of this thesis was to create a platform that allows cardiologists to analyze ECGs and enhance health care systems by exploring a new approach of performing medical reports of an Electrocardiogram remotely, from Global Positioning System (GPS) a mobile device. In this section, the application requirements will be presented, more specifically to implement a platform that gives cardiologists the ability to visualize Electrocardiograms on mobiles devices.

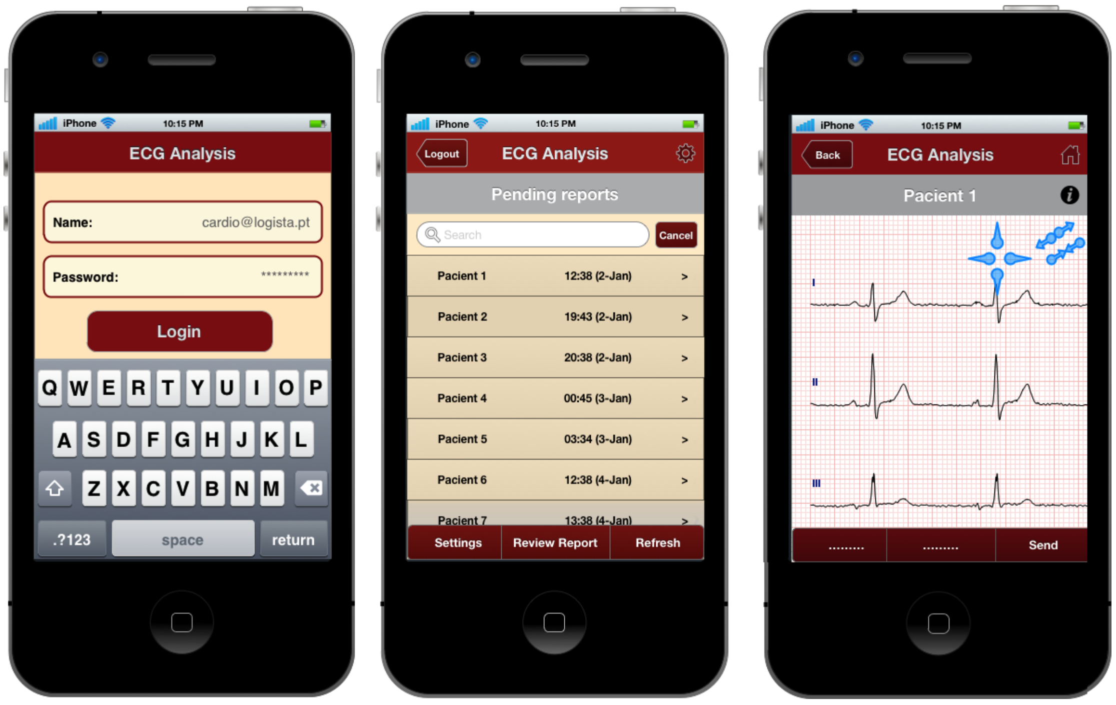
3.1 User requirements

The need for a software platform capable of adapting in various mobile devices was motivated by the fact that, despite existing tools for analyzing ECGs on a computer, exist a lack of solutions to perform those tasks on more limited situations, such as when the user is abroad, without a personal computer or even with a computer without the required operating system. Also, a better cohesion between the user experience across operating systems was required, along with the need of mobility and platform independency.

The most important requirement was the existence of an independent visualization tool capable of graphically representing an ECG examination. This implies being able to draw complex curves of a certain Electrocardiogram on a technology supported and adaptive on most mobile devices. Thus, the application must present a way for cardiologists to see and analyze ECGs, but since most mobiles devices nowadays present limited screen sizes, when compared to personal computers, some solutions will have to be found regarding the navigation within the exam, such as fast switching between curves of an exam, navigating in the X or Y axes or performing zoom operations. In subsection 3.1.1 some interface guidelines will be presented regarding what should the developed platform look like at the end of the development.

3.1.1 Interface

In order to develop a web platform to analyze and represent ECGs on mobile devices, there is a need to primarily define the user interface that will be the first point of interaction between the system and the users of the platform. Initial mockups were developed with the main purpose of defining guidelines for the user interface to be implemented. These mockups were subject to suggestions and critics afterwards, so that unseen features would be included and irrelevant functionalities dropped from the initial list of requirements. Figure 3.1 presents the first version of the mockups designed for what was intended for the mobile web client. As it will be stated later on section 4.3 some features like the search operation or de medical report generation were removed from Cardiobox's purposed functionalities, since it was said that they would be irrelevant for cardiologists:



(a) Mockup for the initial login screen.

(b) Initial mockup for accessing a list of exams.

(c) Mockup for the visualization tool in Cardiobox's platform.

Figure 3.1: Initial mockups for the Cardiobox mobile platform.

As seen in figure 3.1(c), some touch gestures are presented, which represent available operations over the visualization tool. This is a relevant requirement of the visualization tool since limited screen sizes will not be able to draw all twelve ECG curves within the screen's dimensions. So, the user should be able to perform single touch operations such as moving vertically or horizontally for navigating through ECG curves. It can also perform multiple touch operations like, for instance pinch and zoom for increasing or decreasing the ECG detail.

3.2 Functional requirements

In order to achieve the objectives and surpass the obstacles associated to this system, all the main functional requirements were identified and analyzed. Each of these requirements represents a certain functionality that is expected to be implemented on the platform. Moreover, the described requirements will be divided in two parts: requirements related to the web-service and requirements related to the mobile web application. Tables 3.1 and 3.2 show a list of the main requirements to be implemented, followed by a brief explanation:

Requirement	Description
Web-Service	
Configure access to a certain central	The web-service should be able to obtain central's credentials, test both Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP) connections and act upon the success or failure of such operations.
Maintain a session state for authenticated users	The server should maintain the session state of all the users currently authenticated on the platform.
Maintain connections to a certain central	The web-service should maintain IMAP and SMTP connections for authenticated users in order to improve the response time for further operations performed by users.
Secure communications between mobile user agents and the server	The server should accept exclusively connections made through Hypertext Transfer Protocol Secure (HTTPS) on port 443 and reject all the others to mobile user agents, in order to protect all the data exchanged between endpoints and the web-service.
Blinded authentication	The authentication system provided by the web-service should let cardiologists login to the platform without knowing the central credentials, but having personal credentials for being identified and performing medical analyses. Also, the server should not have access to the central credentials, making it unable to access sensitive data.

Table 3.1: Functional requirements of the Cardiobox's web-service.

Requirement	Description
Mobile Web application	
Manually configure an ECG central	The administrator of a certain ECG central should be able to manually configure its central by providing the email credentials, IMAP configurations such as the IMAP host, port and if the connection is made through Secure Socket Layer (SSL) or not. The same should be provided for an smtp connection so that the platform is able to send emails.
Import the configurations of a certain central	Users should be able to import the configurations of a certain central previously generated by an administrator. Upon success, Users should be able to perform normal operation on the platform, given that they are able to perform authentication on the server.
Perform authentication operations	It should be possible to perform both login and logout operation on the platform and validate the authenticity of the user interacting with the platform.
List new medical examinations on the platform	Users should be able to perform listing operations over the pending medical reports on the platform, and pagination is required so that the user is able to navigate through a big set of elements.
Get medical information about a certain examination	Authenticated users should have the possibility to query for medical information about a certain examination, so that when the user of the system selects a specific exam, detailed information is presented.
Get the ECG data about a certain exam	Users should be able retrieve ECG data related to a certain examination, so that a visualization tool can draw graphically all the curves and perform an analysis.
Build a medical report	It should be possible to build a pdf document based on data from a certain examination, such as a sample of the graphical visualization of the ECG curves, medical information regarding the patient in question, the cardiologist's analysis, among others. Moreover, the medical report should be sent back to the cardiologist and to the central through SMTP

Table 3.2: Functional requirements of Cardiobox's mobile web application.

As stated in tables 3.1 and 3.2, some of the most important functional requirements rely on the integration and interaction of both the mobile web application and the web-service. The server will be a web-service with the possibility of scaling in a near feature. Moreover, the server was required to be able to parse and decode ECGs' main data structure, and to build pdf documents that will be used as medical examinations for the analyzed ECGs.

The platform was developed based on the requirements previously identified. Considering all the implemented features and the influence between all the entities responsible for interaction with the system, figure 3.2 presents the system's use cases in a Use Case diagram.

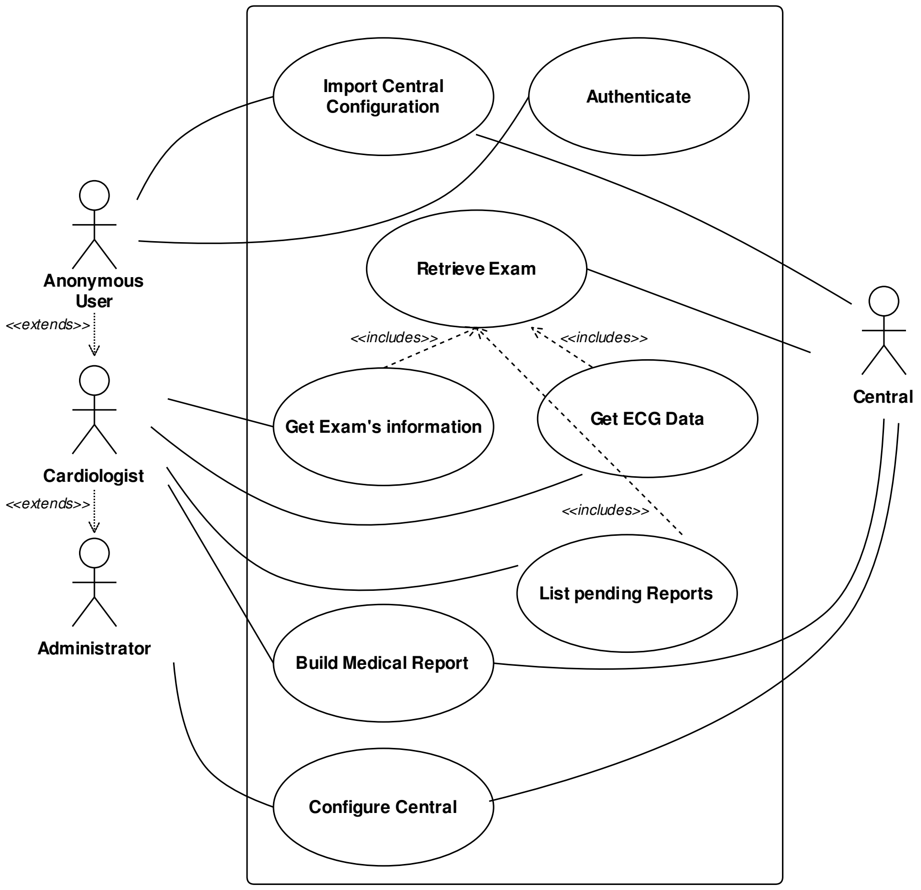


Figure 3.2: System use cases diagram.

As stated in the diagram, there are three types of users of the platform: anonymous users, cardiologists (authenticated users) and administrators. Moreover, there is an external system, as seen on the right of the diagram, that represents the central where all the medical reports are received and stored. Anonymous users are only allowed to access non-authenticated functionalities like the importing the central's configuration feature and the authentication process itself. On the other side, once the user becomes authenticated, it becomes a cardiologist, where features like listing pending reports, getting information about a certain examination or build a report are present. For instance, the use case *Build Medical Report* is dependent of the central because it will be required to be sent an email using the central's configuration.

Moreover, administrators should be the only ones with access to the central's credentials, thus the use case *Configure Central* should only be possible to be used by these users, aside from the fact that they can be seen as authenticated users by the platform as well and perform the same operations that cardiologists do. This use case has a very relevant importance in the platform since it represents the entry point where the administrator configures a certain central and gives access to a team of cardiologists. With this, cardiologists will be able to perform the login with its personal credentials and will blindly use the central's credentials without ever having access to it.

3.3 Non-Functional requirements

3.3.1 Accessibility and Availability

According to Cardiobox mobile's main purposes, it was established that the developed platform should be accessible from anywhere, without any restrictions whatsoever regarding the connection that the user is using to perform the interaction. Moreover, this server's availability should be high, meaning that anytime a cardiologist tries to access the platform, it should be possible to use it. This becomes extremely important in scenarios on the health care field, where the given service should never be interrupted and always available, in order to meet the needs.

3.3.2 Security

Regarding security, the deployed Tele-ECG system will have to inevitably interact with medical data, which implies extra security in order to protect sensitive data and avoid the loss or theft of medical information. Moreover, the platform implemented should present solutions for users to authenticate, should assure data integrity and finally, confidentiality.

3.3.3 Scalability and Performance

In addition to the previously described non-functional requirements, Cardiobox mobile should be designed taking in account its future scalability, easing the improvement of its computation and API extension, to handle more relevant demands that may possibly exist in a near future. This means that performed tasks should be done as quickly as possible, which implies that any processing that can be done before or after user interactions should be performed on the background, thus avoiding the users' wait time to increase. For instance, when an ECG is initially parsed in order to obtain patient's information like its sex, gender, height, weight, etc., characteristics used to list all the pending reports, a background task should start and perform the full parse and filter of the ECG curves. Since there is a high probability that the examination in question will be queried later for visualization purposes, this background task can start and be accomplished meanwhile. Figure 3.3 illustrates the strategy of placing a partial non-essential task in the background, os that the result is available when requested.

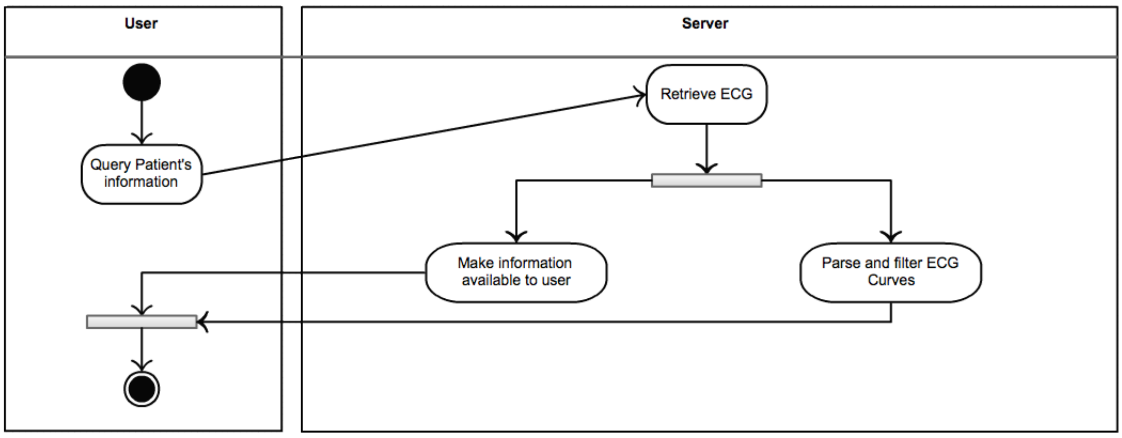

User
Server
Query Patient's information
Retrieve ECG
Make information available to user
Parse and filter ECG Curves

Figure 3.3: Diagram that illustrates parsing ECGs in a background task.

3.3.4 Mobility

The proposed platform should be integrated with nowadays mobile solutions, so that its users can access the platform from mobile devices, without any restrictions or relevant dependencies to other limited frameworks being made, such as Flash.

3.3.5 Platform independency

Finally, because the final product should be a platform running on most mobile devices, and since the mobile market is currently fragmented without a major operating system running on mobile devices, there is a necessity to implement a platform independent system, aiming to improve the user experience for all the major mobile operating system currently in use. This solution can resort to a cross-platform solution like what was discussed during section 2.1.3 Occasionally, some platform-specific specializations can be performed in order to take advantage of each operating system's functionalities.

3.4 Summary

This chapter presented the requirements of both the client side (mobile web application) and the server side (web-service) of Cardiobox. Also both functional and non-functional requirements were discussed. Taking into consideration that the platform target is the cardiology community and the fact that the platform should simplify and make it possible to analyze ECGs on mobile devices, usability becomes a relevant variable during the implementation, which should not be ignored when developing the system. In the next chapter, a proposal and all the implementation details will be described and explained for a better understanding of this dissertation's final product.

Chapter 4

Cardiobox Mobile: architecture and implementation

The requirements described in chapter 3 were faced through the creation of an appropriate extension to the current architecture of Cardiobox's platform. In order to achieve a scalable solution that would let this work be further explored, a web service was created, giving the possibility for other user agents to be developed in the future, such as a desktop client for Linux/Mac OS X or even an unified web client for all platforms. In more detail, some of the most important operations published by the API will be described with more attention and all the security steps and enhancements taken to guarantee the system's integrity will be explained. On the other side, a web app was developed to interact with and consume the former web service, making it possible for cardiologists to successfully analyze ECGs. As it will be shown, this web application takes advantage of media queries, HTML5 rules to make the ECG visualizer take full advantage of different screen dimensions, making it possible to load different view according to the size of the device's screen accessing the web application.

This chapter explores some of the details of both the server side and the client side implementations, the difficulties encountered while developing them and the solutions used to successfully conquer the objectives outlined.

4.1 Architecture extension

Since Cardiobox mobile will be an alternative for cardiologists to analyze ECGs when they can't use their computers, the set of functionalities required for mobile devices should be well thought. According to this, it was decided that the server implementation should be a well designed web service so that different user agents can interact and consume it. This can be seen as an extension to the former Cardiobox architecture, and while the original desktop application lets administrators manage users in the central, perform actions such as creating, editing or deleting them; Cardiobox mobile focuses on delivering information to the user, wherever he may be. It is suffice to say that this decision is a trade-off between having a more solid architecture but limited in terms of the platforms available and having an extension and new user agents working on an already deployed and in use architecture. One of the main advantages of this decision is that all the current users of Cardiobox will be able to use their mobile devices to continue performing tasks that were only available on their computer before. On the other side, this may be seen as a more error prone solution since it depends on the email servers' availability.

Nevertheless, in figure 4.1 the original architecture is described, and all the interactions to the proposed extension are shown. The light gray box marked by I represents the extension to the original Cardiobox architecture. The scalability of this solution is noticeable since the new web service represents a point of access to the information sent by different institutions, making it possible for all devices with Internet connectivity to consume and interact with as many centrals as wished. One might realize that nothing was changed or lost with these alterations, and users can still use Cardiobox on their desktops because the interaction between the web service and the centrals is seamless to the users. Also, all the connections made by endpoints through the new web service rely solely on HTTPS through port 443.

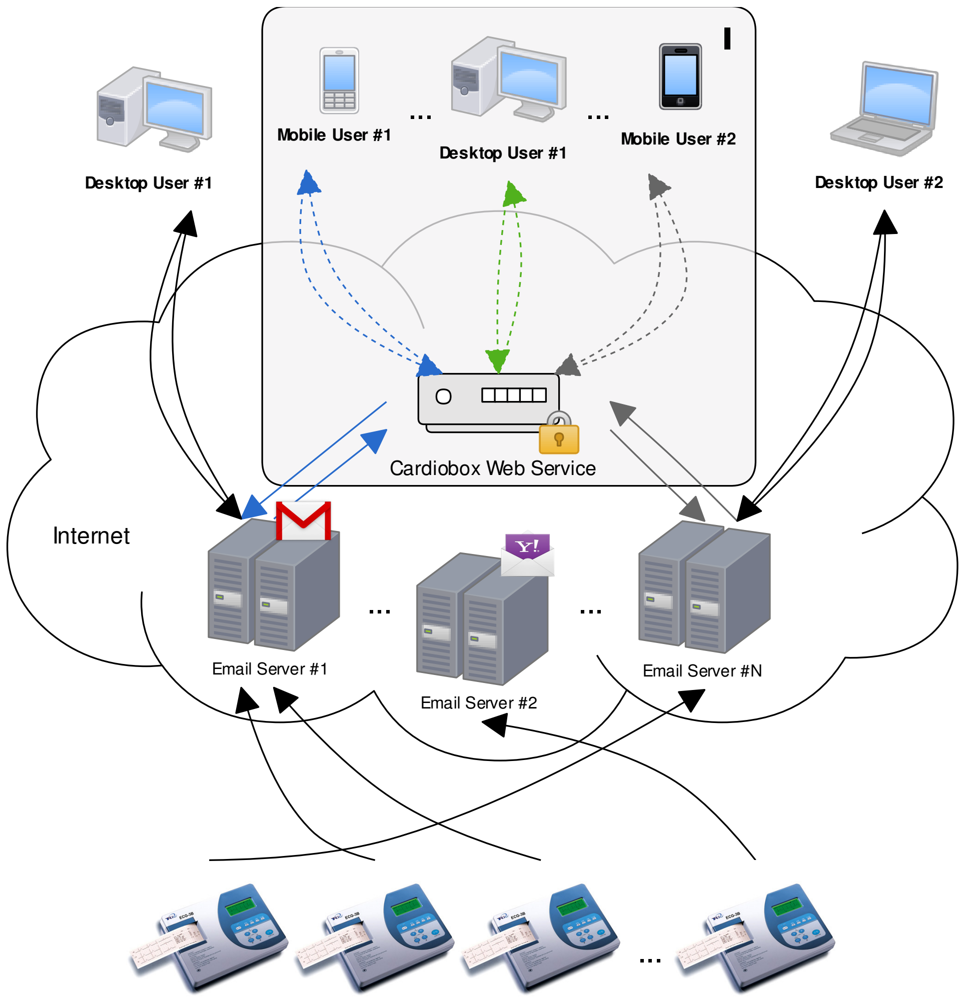


Figure 4.1: Cardiobox's architecture and the proposed extension.

Overall, the proposed architecture takes advantage of what has already been developed and enhances both viewing and reporting functionalities for all mobile devices, thus creating a portable review station. Also, since it represents a generic solution regarding the central structure, any kind of email servers can be used, even public Internet services like Yahoo and Gmail.

4.2 Server side implementation

As described in the previous section, the proposed architecture relies on the development of a web service that will be consumed through different mobile devices [36]. In this section the implementation process will be explained, along with the different engineered solutions to successfully secure both the data and the integrity of this service. Also, the most important available operations will be explained in detail and by the end the reader will be ready to fully understand the integration with the HTML5 web client.

4.2.1 Mission

Although the desktop version of Cardiobox may have many features, only a restricted set of them makes sense to be implemented for mobile devices. Therefore, the main objectives of this web service are to give mobile users the ability to view, analyze and perform a medical report on ECGs available at their central. In order to do this, the built server will have to develop an authentication process, so that cardiologists are able to perform simple login and logout operations. Also, all the methods that either consume or affect data at the central should only be effective upon a successful authentication.

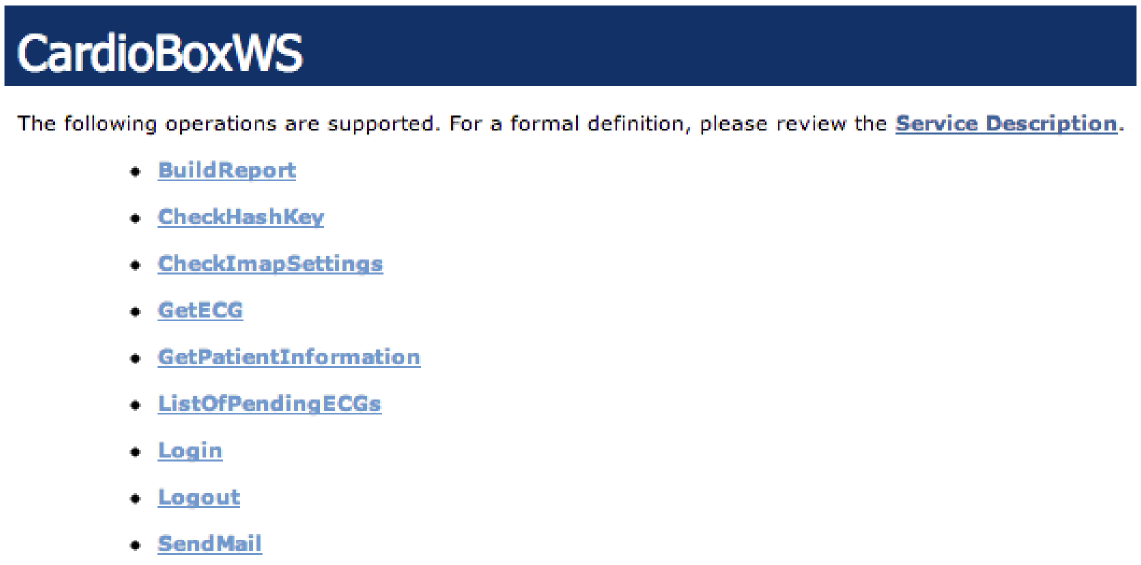
Due to Cardiobox's architecture regarding the way each ECG central is represented by an email account, there will have to be connections through one of the many email protocols like Post Office Protocol Version 3 (POP3), IMAP, SMTP, etc. Because of the big mobility factor attached to mobile devices, where each cardiologist may be connecting to the central from different places such as its home, while roaming, using mobile data, at the hospital, on a public network, etc.; there is a bigger chance that an implementation where the mobile application would connect itself through IMAP/SMTP would fail. As an example, University of Aveiro's Internet connection, named Eduroam, does not let a user connect to outside SMTP servers. This would mean that all the features on Cardiobox mobile that perform SMTP connections wouldn't work while the mobile device is using this connection (sending a medical report uses an SMTP connection to send an email). With the proposed solution, only the web service will be using Email protocols, while all the mobile user agents consume the web service through a simple and secure HTTPS connection (through port 443).

All the server side development was made in Microsoft Windows 7 x64, using Microsoft's official Integrated Development Environment (IDE), Visual Studio 2010. All messages exchanged between the server and the various web service consumers are in XML format and the communication is established exclusively over HTTPS. For development purposes, initial phases of this implementation took place solely on a virtual machine, where it was only possible to use emulators/simulators such as the iOS' simulator, Opera Mini Simulator [37, 38], Android emulator, etc.. On a more advanced phase, the web service was deployed using Internet Information Services (IIS) 7.0 on a machine inside University's Campus. During this transition, the client was required to either be connected to the University's network (Eduroam) or access it through Virtual Private Network (VPN). Finally, at its final and more mature stage, Cardiobox mobile became live and available worldwide through the following Uniform Resource Locator (URL): `https://bioinformatics.ua.pt/cardioboxmobile/`.

Nevertheless, the primary mission is to let mobile user agents use core features such as listing pending reports, viewing different curves of an ECG and building and sending a medical report, and for this the server implementation will have to be able to manage sessions for each user logged in, verify email credentials, connect to the email server and search for ECGs, parse the raw ECG to a simpler and filtered one and, finally, elaborate a pdf document with an image of the ECG, the analysis made by the cardiologist, the cardiologist's identification and other patient's information.

4.2.2 Available methods

After deciding upon which features were considered essential in a mobile device for the workflow of a cardiologist to analyze ECGs, the following methods were developed in the Simple Object Access Protocol (SOAP) API:



CardioBoxWS

The following operations are supported. For a formal definition, please review the Service Description.

- BuildReport
- CheckHashKey
- CheckImapSettings
- GetECG
- GetPatientInformation
- ListOfPendingECGs
- Login
- Logout
- SendMail

Figure 4.2: SOAP API available operations.

Apart for the login operation, there are only two methods that do not require the user of the system to be authenticated, those being CheckHashKey and CheckImapSettings. As it will be seen on the client side implementation, when the user first visits the web application, it is confronted with a two option configuration form, either **Insert key** if the user was given a key to automatically configure its central, or **Manual configuration** for a more powerful alternative that lets the user fill in a detailed form regarding the configurations needed to connect to the central. In spite of the simple API provided to configure an ECG central, it is worth to understand what really takes place in each of theses methods [39]. Not only the CheckImapSettings method lets the administrator of a certain central fully configure the details of the connection, but it also gives the ability to create a hash key that will be used and sent to the team of cardiologists of the given central. This method's parameters are:

- **name:**
  A basic configuration name to help users memorize and associate a certain configuration to a central. As it will be seen on the client side implementation, this will make it possible for the web client to have multiple configurations for different centrals set up and ready to be used.

- **email:**
  The email address of the central in question. This email address represents the entity

where all the pending ECGs will arrive, all the medical reports will be sent, etc..

- **password:**
  The password associated with the email address formerly described. Only the central's administrator should have access to this token, however, if the administrator decides to give away the password, the cardiologists will also be able to use this operation in order to perform a manual configuration.

- **imapHost:**
  The URL of the email address' IMAP server (e.g., imap.gmail.com for google accounts). This, along with the next 2 items, will be completely necessary to successfully login to the email account and retrieve and view ECGs.

- **imapPort:**
  The port number of the email address' IMAP server (e.g., 993 for google accounts).

- **imapSSL:**
  True if the IMAP connection should be made through SSL, false otherwise (e.g., true for google accounts).

- **smtpHost:**
  The URL of the email address' smtp server (e.g., smtp.gmail.com for google accounts). On the other side, this and the next 2 items will be solely used on features that require the platform to send an email through SMTP using the given email account. Emails are sent on two occasions: when the administrator finishes manual configuring a central and wishes to send an email to a set of cardiologists to give them access to the central, or when a certain cardiologist finishes analyzing an ECG and writes and builds a medical report, thus sending the generated document to the central and to himself.

- **smtpPort:**
  The port number of the email address' SMTP server (e.g., 465 or 587 for google accounts).

- **smtpSSL:**
  True if the SMTP connection should be made through SSL, false otherwise (e.g., true for google accounts).

- **nDevices:**
  This parameter lets the administrator decide upon how many activated devices can actively use this configuration. If, for example, he chooses 10, the generated configuration will only be available to be imported 10 times. After the tenth time, this configuration expires and only users who proceeded to import them will still be able to use it, other users with the URL / Hash / Quick Response Code (QR Code) will not be able to consume this configuration as the hash will expire.

This generic solution fulfills Cardiobox's requirements where any email address could be used as a central, as long as it concedes imap and smtp protocols. This operation tries to connect through IMAP first. If it connects successfully, it then tries to connect to smtp, otherwise sends an error message so that the form in the client side implementation will be able to give correct feedback to the user (e.g., incorrect credentials, invalid SMTP configurations, etc.). Finally and in case of success, this method returns 3 objects, those being a unique URL to the web application that lets users who access it import the configuration settings, a QR Code encoded image with the former URL so that cardiologists reading the email sent from their administrator will be able to easily decode the QR Code and be redirected to the web application on their mobile devices, and finally a token that corresponds to the configuration.

CheckHashKey is a simpler way to import a configuration, and is called when a user accesses the unique URL, decodes the QR Code or puts the key on the option **Insert key**. In figure 4.3 a sample QR Code along with the unique URL can be seen. Its content is: https://bioinformatics.ua.pt/cardioboxmobile/index.html#bikLTL59ZACoyb8282LA.

Figure 4.3: QR Code with a unique key to access the web application.

If the hash key is valid, the method returns the configuration token. This configuration token should be preserved and used during every login operation. This token corresponds to a base64 string of the serialized entity class MailSettings, consequently ciphered using Advanced Encryption Standard (AES) 256. This way, when a cardiologist logs in, in spite of its credentials being sent to the web service for validation, the configuration token is sent as well, letting the web service properly connect through IMAP to the central and perform the login operation. Nevertheless, the remaining operations require authentication and will return Hypertext Transfer Protocol (HTTP) 401 Unauthorized [40]. The login system is dealt on the server side through login session and the cookie expires after 1 hour or if the operation **logout** is performed.

Another important method in the API is the **ListOfPendingECGs**. Since many mobile devices show limitations on many hardware factors, such as the screen size, unstable or limited Internet connection, etc., this method should perform pagination so that the web application is able to show just a subset of the complete list of pending ECGs, and giving the ability to load more, thus extending the size of the list that is seen on the smartphone. When performing this operation a start and end values can be sent as parameters, which will let the client side control which part of the full list is displayed on the mobile device. In case the values outbound the available list, or if there are currently no pending ECGs, an error is returned to the client informing about it. Otherwise, a list of pending ECGs is retrieved with a unique identifier (Unique Identifier (UID)) for each ECG, the name of the patient and the date of the ECG.

With this UID the client application is able to call both **GetPatientInformation** and **GetECG**. **GetPatientInformation** gives an automatic interpretation of the ECG, information about the ECG curves like the ventilation rate, and the PR and QRS durations, and information about the patient such as its age, sex, height, weight, etc.. **GetECG**, on the other hand, lets the web client ask for the fully parsed and filtered set of 12 curves that consist the patient's ECG. Table 4.1 gives part of an ECG and its structure:

ECG parsed data
<string> [["I","II","III","aVR","aVL","aVF", "V1","V2","V3","V4","V5","V6"], [[29,32,36,40,40,40,34,27,32,37, 36,35,32,30,27,32,30,37,...],...] </string>

Table 4.1: Sample response of a partial ECG being retrieved from the SOAP API.

Showing the full response would be unreadable since each request returns the full ECG which is consisted by ten thousand points for each curve, making a total of one hundred and twenty thousand values for all the 12 curves of the ECG. Finally, the method **BuildReport** receives the same UID for identifying the patient's ECG, the cardiologist's name and the image of its signature and its personally written analysis. The server uses an open source library for the .NET language that easily creates Portable Document Format (PDF) documents [41]. A sample medical report, generated using the web client and the web service developed can be seen in the Appendix A.

In order to better understand the flow of events previously described, a sequence diagram is shown below which shows the details of interaction between one web client, the built web service and an email server. In this diagram (Figure 4.4) a normal flow is presented, where the user first logs in on the platform, upon which the list of pending ECG is retrieved and the user chooses one. After this selection, both the patient's information and the parsed values of the ECG curves are retrieved. In this case, the web client has all the necessary data to present to the cardiologist the drawn curves and all the relevant information expected in order to the medic to perform an analysis.

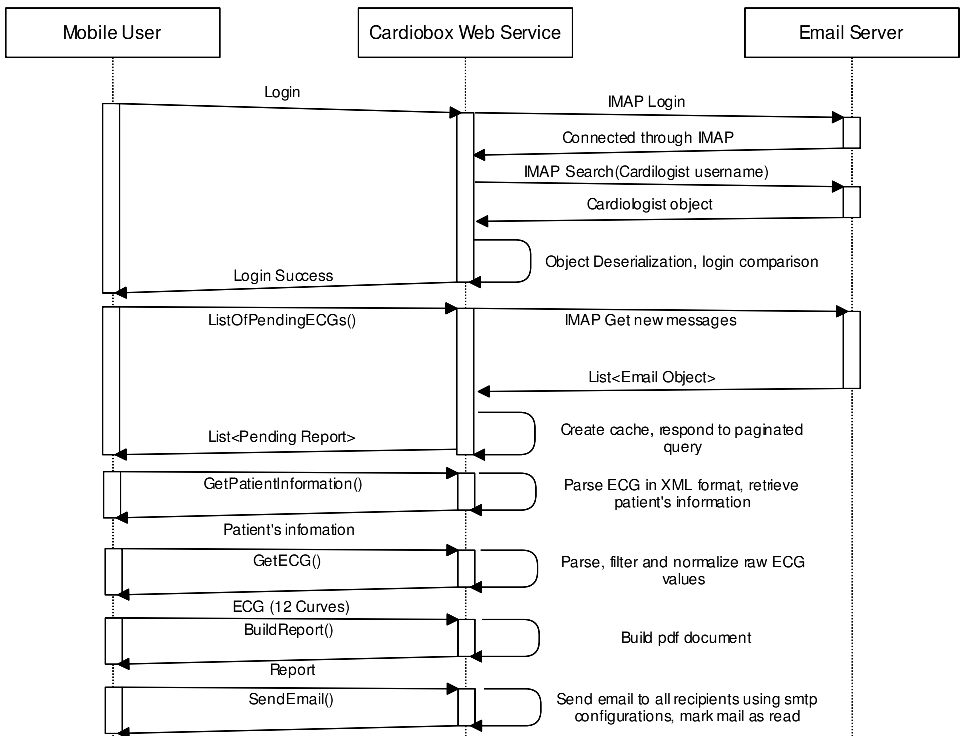


Figure 4.4: Sequence diagram of a normal interaction in Cardiobox's platform.

To summarize, this core set of functionalities will concede a secure and effective way to interact with any central and perform main tasks, such as listing pending ECGs, viewing its curves, generating a medical reports, etc.. On the next section the web application will be presented, along with all the steps taken to reach such solution.

4.3 Client side implementation

After describing the server side implementation, it is time to present the challenges occurred during the development of the web application. In this section, the HTML5 application will be fully explained, along with all the functionalities and HTML5 features used, such as media queries, local storage and others. Also, this section serves as a guide for the future users of the platform to familiarize with its characteristics. Finally, it will be showed just some of the many fallbacks that are used nowadays in order to tackle a situation where a certain HTML5 feature is unavailable.

4.3.1 Mission

The desired web application was developed using HTML5, CSS3 and Javascript and every feature was thoroughly analyzed and thought as a mobile user with touch capabilities. The development was initially done in Espresso [42], but as the project evolved the need for a more powerful tool became present and Espresso was replaced by Aptana Studio 3 [43]. Although the main focus of this dissertation was the development of a web client capable of running on most mobile devices, some effort was put on optimizing the user agent for a specific mobile operating system - iOS. In order to accomplish this, the iOS Simulator that comes with Xcode was intensively used, which replicates seamlessly the behavior of a real iPhone. However, in a more mature phase of the project, mainly after the deployment of the server to the Internet, a test device was requested to the Electronics department and the process of testing/bug fixing took place. This was an important decision because some minor implications were found when accessing Cardiobox mobile from a mobile connection, in this case through Vodafone's 3G connection.

Nevertheless, other tests were performed on both smartphone and tablet Android emulators, and in Opera mini. Obviously, since this represents a web solution, any browser is able to access the web application. However, the only desktop browsers that should be used to test Cardiobox mobile are Safari and Google Chrome. Both browsers share a renderer engine in common, named WebKit [44]. This engine is one of the most advanced nowadays in rendering Javascript and in supporting HTML5 & CSS3 features, and since both Android's Google Chrome and iOS' Safari share the same engine, rendering Cardiobox mobile in either the desktop or on a smartphone will have the same result (except for hardware limitations).

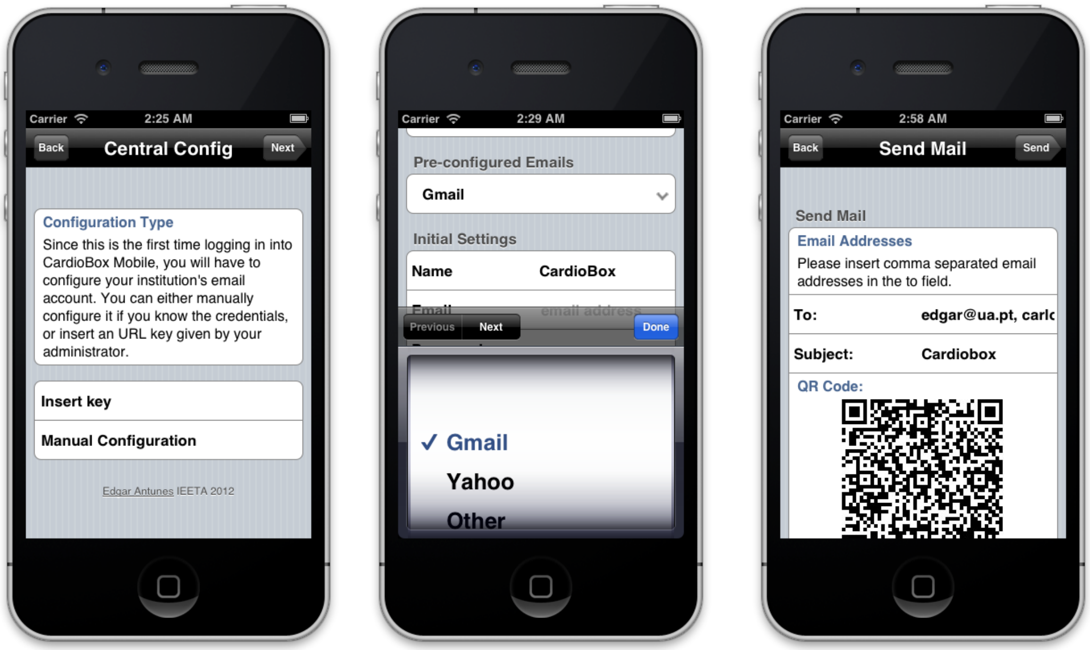
The remaining sections will follow the workflow of a user of the platform, where each section in show and explain in detail a certain feature and will present ways to navigate within the application, between different screens. Finally, this chapter will end with a set of improvements made after the first real tests performed in mobile devices.

4.3.2 Generating/Importing settings

Following the order used in the server implementation description, the first real limitation that the user finds when he first accesses the web application is how to be able to properly configure its central. Cardiobox mobile presents three ways to accomplish this, so that users find it easy to get their mobile devices correctly configured and ready to visualize ECGs:

- Through the link sent by the administrator
- By decoding a QR Code in the email body sent to cardiologists
- Using the hash key in the import option available on the web application

First, the central's administrator navigates to Cardiobox's web page in any browser and proceeds to use the manual configuration option for its central, as seen in Figure 4.5(a). This involves a more detailed form about the central's configurations such as the login credentials, IMAP and SMTP settings, etc.. In order to ease the administrator to insert information that may be known by the server, a drop-down is populated by the server with settings for centrals of publicly available email services, such as Gmail, Yahoo (Figure 4.5(b)). This lets the administrator avoid to enter a total of 6 fields for both imap and smtp configurations that may already be known. At the end of the configuration, the administrator will be given the possibility to define the number of activated accounts that can use this configuration. Upon validation of the filled in configuration, a unique link is generated that will give cardiologists access to the platform. It is important to state that the link will expire when the maximum number of devices is reached. For commodity to the administrator, after generating a schema of settings, it is presented with an email form, so that the created link is sent to the team (Figure 4.5(c)). In this body's email, the web service has put a QR Code that will let cardiologists quickly decode it with a smartphone's camera and gain access to Cardiobox



Configuration Type

Since this is the first time logging in into CardioBox Mobile, you will have to configure your institution's email account. You can either manually configure it if you know the credentials, or insert an URL key given by your administrator.

Insert key

Manual Configuration

Edgar Antunes IEETA 2012

Pre-configured Emails

Gmail

Initial Settings

Name	CardioBox
Email	email address

Send Mail

Email Addresses

Please insert comma separated email addresses in the to field.

To:	edgar@ua.pt, carlc
Subject:	Cardiobox

QR Code:

(a) Initial import screen. (b) Drop-down with email services. (c) Email form to send settings.

Figure 4.5: The import screen on the left, populated drop-down with email services in the middle and email form on the right.

mobile 4.5(c). The administrator can put multiple email addresses, separated by comma, and the server will be responsible to use the newly SMTP configurations to send emails from the central to all the cardiologists.

As seen in Figure 4.5(a), the third possibility to import said settings is to use the option **Insert key** and input the key given by the administrator. Importing this configuration is relatively simple thanks to HTML5 local storage API. In Javascript, if the server responds to the URL request with a status code of 200, its JavaScript Object Notation (JSON) response concedes the token to be imported. Storing it with local storage is just performing a *setItem* as showed in Listing 4.1.

Listing 4.1: Javascript's interaction with HTML5 local storage capabilities

```
localStorage.setItem("configToken", token);
```

After importing the settings, further accesses to the web application will always check if the stored token is present first, and if true the web application will redirect itself to a login form, as we will see in the next section.

4.3.3 Logging in and management of multiple centrals

The login window also happens to be the main web page of Cardiobox. In this page the user is able to login, change central and perform a new configuration through the import button in Figure 4.6.

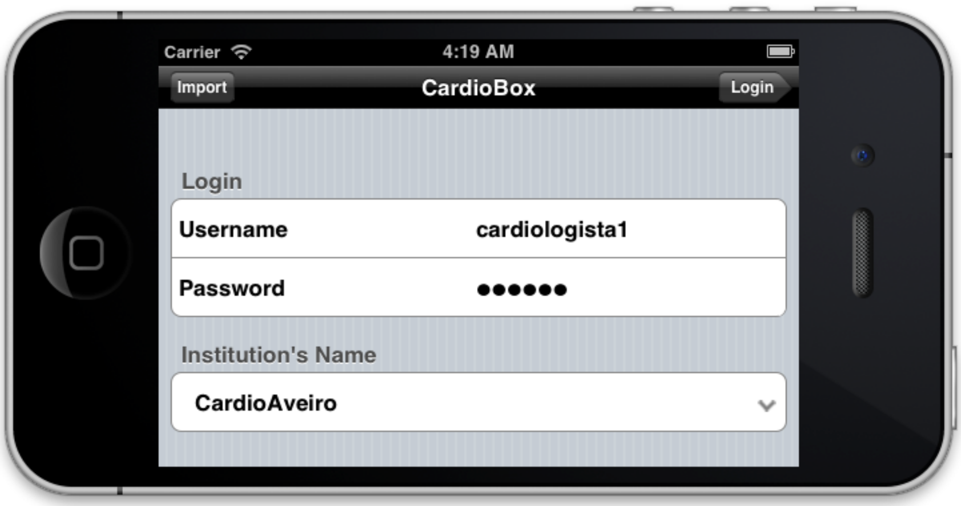
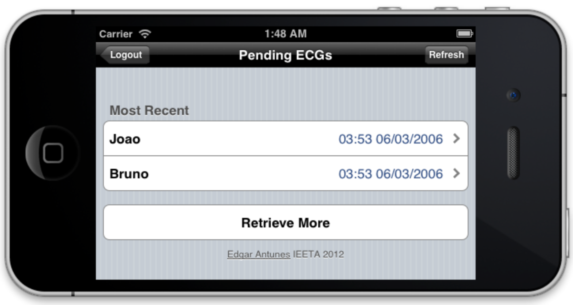


Figure 4.6: Cardiobox's login page.

Some complexity was added to the login process in order for the user to be able to use multiple centrals. When loading the page, the drop-down is populated with the list of settings available, retrieving each central's name so that the user selects the correct one without any hassle. This was a requested feature by the cardiology group since, without this capability the user would have to delete one configuration and fetch another, which would be a lot more troublesome.

After the login process, which is unique for each cardiologist, if succeeded the cardiologist is prompted with a list of pending ECGs. This window can be seen as an adaptive list of pending analyses, where the user can easily see the patient's name and the date of the exam in question. In an initial phase of the development, this feature suffered from a relevant performance issue. Since, at that time, the web service would return the number of pending reports and some information about each one, in case the central in question had a significant number of exams pending, the response's time would increase drastically. This could even lead to a worse case scenario where simultaneous requests for centrals with a great amount of pending reports could eventually provoke the server's failure. In order to tackle this issue, the API was changed and a pagination system was implemented, with a cache. This system, had inevitably to change the API so this design decision forced some changes on the client as well. However, after its implementation, the client is able to smoothly get a subset of the list of pendent ECGs and query for more if need, as seen in Figure 4.7(a).



(a) Cardiobox's pending ECGs.

Figure 4.7: List of pending ECGs ordered from the most recent to the oldest.

4.3.4 Visualizing the ECG

In this section the core feature of this dissertation will be presented - the ECG visualization station capable of displaying an Electrocardiogram. A cardiologist can reach this window with two simple steps:

- Log in
- Choose a pending report from the list.

Because of the limitations found in various mobile devices regarding the screen size, two perspectives are available. The default perspective, shows all the twelve curves with the screen size available, where each curve is presented in a distinct part of the screen. On the other size, the other visualization perspective available lets the user switch to a full-screen view with just one of the curves in detail. This perspective tries to tackle the problem of not having enough detail on small screens when all the curves are displayed at once. With this perspective, the cardiologist is able to click on the button showed at the top right bar of the screen named waves, that lets it switch between curves, as shown in figure 4.8. Regarding the perspective itself, each curve is drawn in an HTML5 canvas element, and these canvas are displayed in an adaptive way so that scroll is not needed and the window adjusts to the screen.

In order to replicate a real ECG's aspect, two elements should be drawn on each canvas, the paper grid and the ECG itself. For the paper grid, a Javascript module was implemented that uses the HTML5 canvas API to draw both vertical and horizontal lines. Listing 4.2 shows the API call on an initial approach of this module, that would generate an inaccurate line for the grid:

Listing 4.2: HTML5 Canvas API

```
var ctx = canvas.getContext('2d');

ctx.beginPath();
ctx.moveTo(0,0);
ctx.lineTo(canvas.width, canvas.height);
ctx.fill();
```

The sample showed above writes a line from the position (0,0) to the end of the canvas element. This approach performed poorly and showed some bad results since the drawn line would always be placed with an odd thickness, as seen in Figure 4.8(a). This proved to be insufficient because the grid would not look like the precise medical sheets used for printing ECGs. It was found out that instead of using the *lineTo* API calls, it was possible to use the *fillRect* instead, and the problem was solved (Listing 4.3)

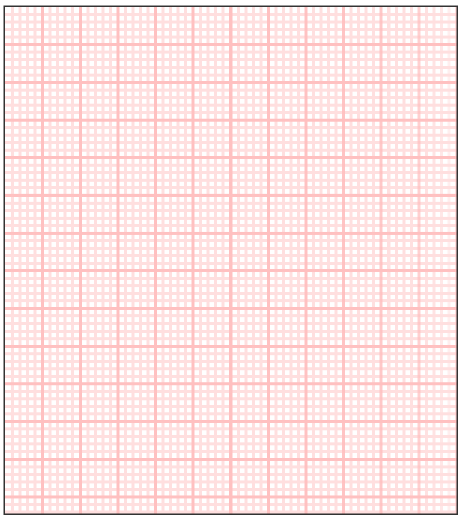
Listing 4.3: HTML5 Draw Rect Canvas API

```
var ctx = canvas.getContext('2d');
ctx.fillStyle = "#ffdfdf";
var y = 0;

while(y < canvas.height) {
        ctx.fillRect(0, y, canvas.width, 1);
        y += step;
}
```

This weird solution draws a rectangle from the point (0,0) until the end of the canvas' width with the height of one pixel. y iterates between a step thus making all the horizontal lines in the canvas. The same applies for the vertical lines. Also, two colors were used, one for the 1mm lines, and a darker color for every 5mm lines, as seen below. In Figure 4.8(b) the final version of the algorithm of the ECG grid generation is presented, where a more precise set of lines is used to replicate ECGs' real medical sheets.

The other element drawn on the canvas is the ECG. This was done in a complex Javascript module that interacts with the web service consumption part, in order to realize if the ECG has been retrieved or not. When the ECG is finally obtained, it is temporarily stored on the mobile device and the DrawCanvas module gets notified of such event. With this, this module will, this time, interact with the UI module, so that it can find out which perspective is active at the moment. It then reacts to the current perspective and draws either the select ECG curve or all the curves in different canvas. This algorithm will then iterate through a subset of the ECG values and draw them depending on the current zooming value and the position of the ECG.

It is important to notice that the user is able to zoom in and out by using a two finger gesture on the screen and pinching in or out. This will adapt the current perspective in order to see more or less detail about the selected curve. Also, to avoid performance issues, everything that is drawn on canvas is limited by the canvas's size and the methods *ontouchstart*, *ontouchmove* and *ontouchend* are overridden so that both the ECG grid and the ECG itself are displayed correctly upon user interaction. In Figure 4.9(b) only the first curve of the ECG


(a) A first version of the ECG grid.

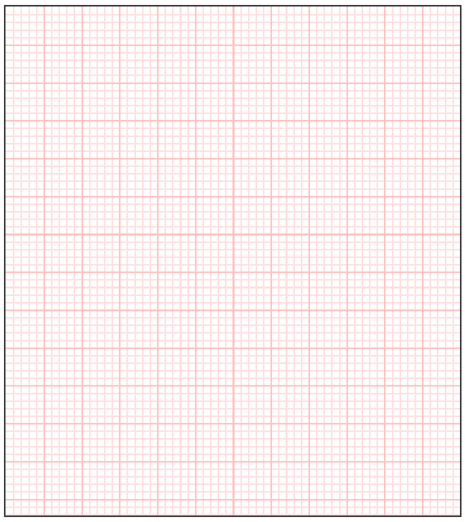
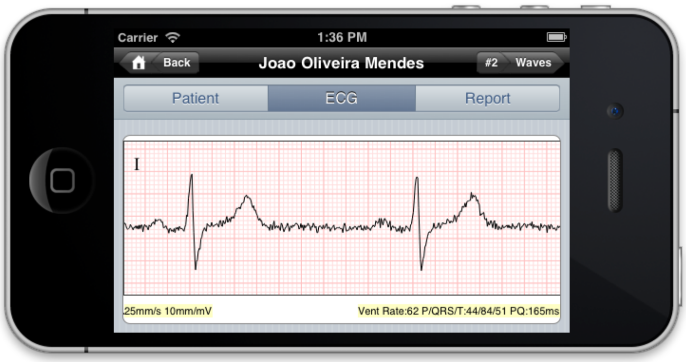

(b) The final version of the ECG grid.

Figure 4.8: Two distinct approaches at drawing the ECG grid.

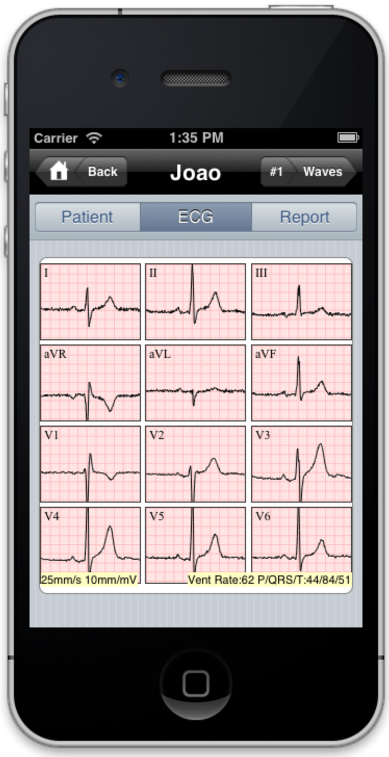
is displayed and the user performed some zoom in, while Figure 4.9(a) represents the default perspective where all ECG curves are shown.

Notice the small detail where the navigation bar is also adaptive. Not only it adjusts to the smartphone's width but the name of the patient is fully written, if there is enough space. Notice also that some effort was taken into making the web application work seamlessly on both landscape (Figure 4.9(a)) and portrait(Figure 4.9(b)).

To summarize, in Appendix B Cardiobox's mobile ECG visualizer is presented as seen on an iPad. The current perspective during the taken snapshot was the multiple curve view with all the Curves presented in separate canvas.



(a) Cardiobox's ECG visualizer with only one curve.



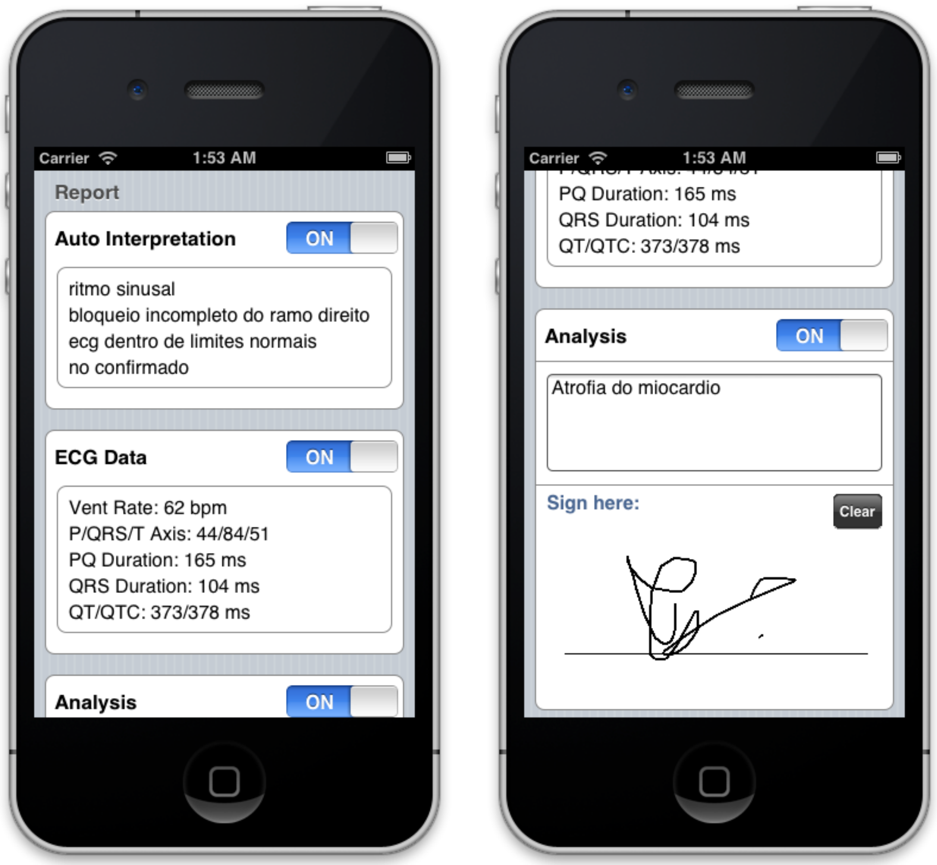
(b) Cardiobox's ECG visualizer with multiple curves.

Figure 4.9: Cardiobox's ECG visualizer tool in both available perspectives.

4.3.5 Writing and building the medical report

As seen in the previous subsection, the user can switch easily between the visualization window, the patient information and the medical report form by using the navigation bar at the top. The patient information, as seen in Figure 4.10(a), concedes several data about personal information regarding the patient, like the age, gender, height, weight,etc.. Moreover, the generation of the medical report comprises a significant set of information about the patient and the exam. This information, showed on the medical report window, can be turned on or off by the user by clicking the button of the desired information group. In order to fill the form and validate the generation of the medical report, the cardiologist is required to write its medical analysis and perform its signature. The signature also relies on the HTML5 canvas so create a signature and convert it to an image before sending the data to the server for generation of the report. Upon a successful send of the medical report, the server will mark this pending report as read and send an email to the central and to the cardiologist with the newly created report as an attachment, for history purposes (Figure 4.10(b)).

Finally, these features lead the workflow again to the pending reports list, where cardiologists can continue working and analyzing different ECGs. At the same but on the server side, a PDF document is generated and sent both to the central and to the cardiologist that performed the analysis. The generated report can be seen illustrated in figure 4.11. A fully


Carrier 1:53 AM

Report

Auto Interpretation ON

ritmo sinusal
bloqueio incompleto do ramo direito
ecg dentro de limites normais
no confirmado

ECG Data ON

Vent Rate: 62 bpm
P/QRS/T Axis: 44/84/51
PQ Duration: 165 ms
QRS Duration: 104 ms
QT/QTC: 373/378 ms

Analysis ON

(a) Cardiobox's medical report.

Carrier 1:53 AM

PQ Duration: 165 ms
QRS Duration: 104 ms
QT/QTC: 373/378 ms

Analysis ON

Atrofia do miocardio

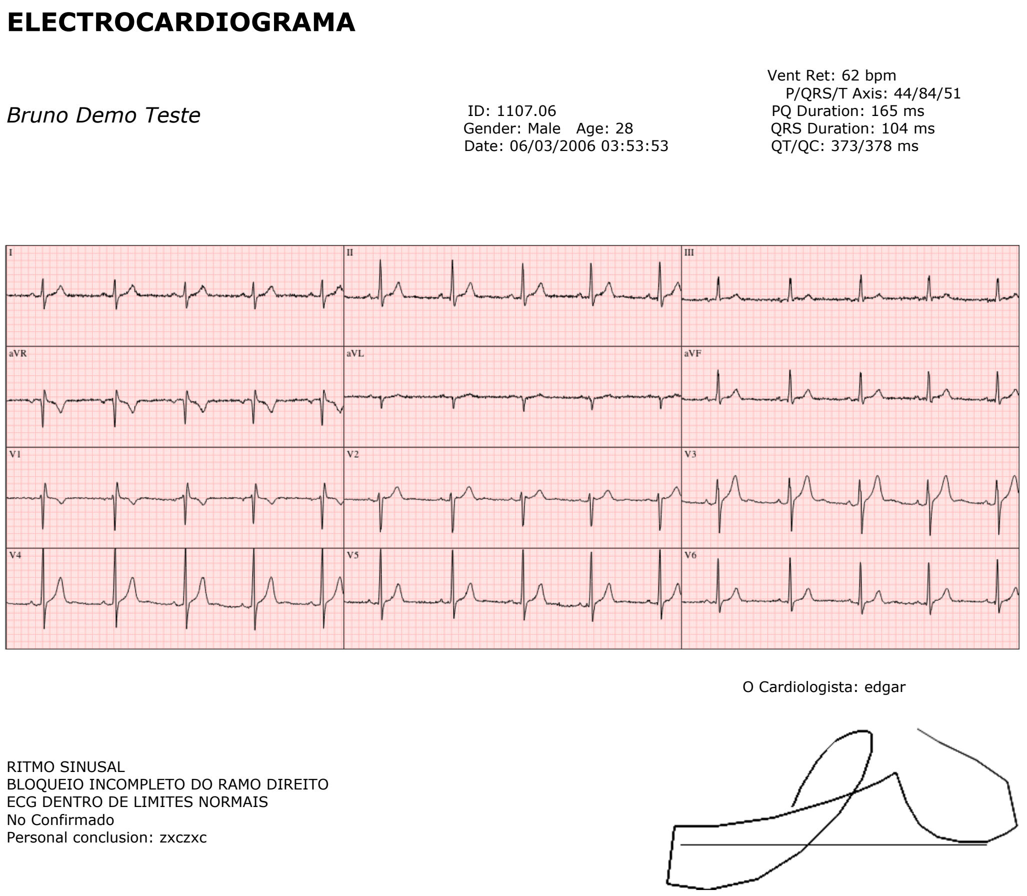
Sign here: Clear

[illegible]

(b) Cardiobox's signature capability.

Figure 4.10: Medical report form before sending the data to the server for validation.

detailed report can be seen in the Appendix A.



ELECTROCARDIOGRAMA

Bruno Demo Teste

ID: 1107.06
Gender: Male Age: 28
Date: 06/03/2006 03:53:53

Vent Ret: 62 bpm
P/QRS/T Axis: 44/84/51
PQ Duration: 165 ms
QRS Duration: 104 ms
QT/QC: 373/378 ms

O Cardiologista: edgar

RITMO SINUSAL
BLOQUEIO INCOMPLETO DO RAMO DIREITO
ECG DENTRO DE LIMITES NORMAIS
No Confirmado
Personal conclusion: zxczxc

[illegible]

Figure 4.11: Sample medical report generated by the server.

4.3.6 Usability improvements

After performing real tests, some significant concerns showed up regarding the platform's performance. Some research and some stress tests on the server showed that biggest delay in a response to the web service was the time to connect and login through IMAP/SMTP. Fortunately, the solution taken took advantage of IMAP's session capabilities to perform a series of customizations to improve performance, such as increasing IMAP's session timeout to 20 minutes, the same as the login session in the SOAP web service. Also, a noticeable cause of performance degradation was the fact that at the beginning of every request the server would create a new connection through IMAP/SMTP, and after building the response, the connection would be closed. This is a bad design decision since the probability that the same user performs consequent requests is high. So an integrated session manager was implemented that would let an imap connection open until it timed out or until the user would logout/suffer from a session timeout (20 minutes). This configuration improved response times greatly and led to a more responsive platform.

Another usability problem was found when testing Cardiobox mobile on different mobile devices. Although the visualization tool was tested on multiple devices with different operating systems and several screen sizes, some HTML5 capabilities were assumed to be existent where it may not always be true. In fact, as seen in [45,46], not all browsers support multi-touch:

> *"**ANDROID 2.3.3 (NEXUS)** - On the Android Gingerbread Browser (tested on Nexus One and Nexus S), there is no multi-touch support. This is a known issue."*

This implication conflicts with one of the core functionalities of the developed ECG visualizer, which is the usage of multi-touch support for performing zoom-in and zoom-out operations. Since this is an essential feature for the web application, some effort was put into solving this problem and finding an alternative solution that would perform well on all the devices. The first approach taken consisted in assigning a new gesture that wouldn't require multi-touch for performing the scale operations. Unfortunately, more touch system nowadays perform these actions with a two finger pinch or zoom. This would eventually lead users to try to perform such gesture with not avail. Another possible solution would be to use the user agent string to detect this very specific set of devices without multi-touch capabilities and give a different gesture from this situations. Once again, this solution would make the platform diverge in its behaviour across different devices. Meaning that a cardiologist used to interact with Cardiobox on a certain device (e.g., using multi-touch gestures) would have to inevitably change its way of interacting with it if, for instant, it had to change to another mobile device. So, none of the solutions were optimal.

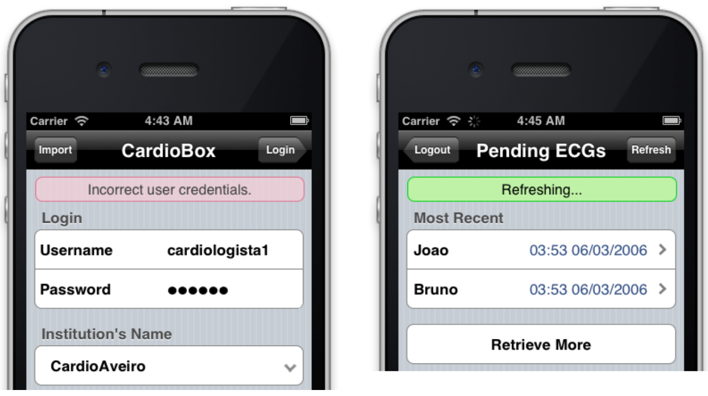
Fortunately, an open-source Javascript library called Modernizr [47] was found and is able to aid web application to figure out if certain HTML5 & CSS3 features are present or not in the client side, giving the possibility to inflict another behaviour depending on each situation. Modernizr uses an extensive set of features, both from HTML5 & CSS3, and creates an object called Modernizr with a list of boolean variables telling whether or not a certain feature is available. If the desired feature is inexistent, a fallback can be put in order to fix the problem. Modernizr was integrated to the developed web application and the following features are the relevant ones for Cardiobox [48]:

Listing 4.4: Modernizr touch detection

```
if (Modernizr.touch){
// bind to touchstart, touchmove, etc and watch `event.streamId`
} else {
// bind to normal click, mousemove, etc
}
```

Also, as a fallback for those specific devices that do not present multi-touch support, a simple overlay layout with a plus and a minus buttons, corresponding to the zoom-in and zoom-out operations. This solves the inconsistency problem of having the platform with different behaviours for the same features, since users that will find the limited zoom operations will see these buttons an easily perform such tasks. Modernizr proved to be a great addition to the web application an further improvements took into account its integration and tried to improved feature detection, like with HTML5 local storage.

Regarding the design, Dr. José Ribeiro suggested that there should be a notification system, that could be used to inform the user about either a confirmation message or an error. These notifications were implemented and propagated through the whole web application, thus making a good enhancement to the web client since that there was a way to say if the login was performed correctly or if the credentials were invalid, if the medical report was sent successfully or not, etc.. Two sample notifications can be seen in Figure 4.12. Moreover, with



Incorrect user credentials.

Login

Username cardiologista1

Password ●●●●●●

Institution's Name

CardioAveiro

Refreshing...

Most Recent

Joao	03:53 06/03/2006
Bruno	03:53 06/03/2006

(a) Notification regarding an error logging in.

(b) Notification informing a successful operation.

Figure 4.12: The notification system in Cardiobox mobile with two different types of notifications.

HTML5 media queries, is it possible to make a web application zoomable or not. For instance, in order to avoid the user to zoom in at Cardiobox mobile page, the following queries were inserted in every HTML5 page:

Listing 4.5: HTML5 media query to avoid zoom

```
            <meta name="viewport" charset="utf-8"
content="width=device-width,
initial-scale=1.0,maximum-scale=1.0,user-scalable=no"/>
```

In Listing 4.5 we make it impossible for the user to use touch gestures like pinch or zoom, which is the desired behaviour since Cardiobox relies on a adaptive layout that adjusts itself to any screen size. This can be seen in Listing 4.6 where different CSS3 layouts are inflated, depending on the screen size [49,50]:

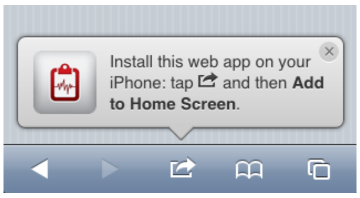
Listing 4.6: HTML5 media query to avoid zoom

```
<!--Diferent CSSs for different devices-->
<link href="_css/mobile.css" rel="stylesheet" type="text/css"
media="screen,projection" />
```

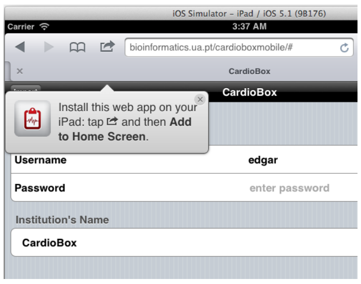
```
<link  href="␣css/tablet.css"  rel="stylesheet"  type="text/css"
media="all␣and␣(min-width:␣481px)␣and␣(max-width:␣800px)"  />

<link  href="␣css/desktop.css"  rel="stylesheet"  type="text/css"
media="all␣and␣(min-width:␣0px)␣and␣(max-width:␣480px)"/>
```

In the future, it is possible to scale this solution for a even better approach where, in an initial phase, the server would find the operating system of a certain client request, through the usage of User Agent Strings. According to the User Agent String, the css file load could replicate the user interface of the given platform. For instance, Cardiobox's layout currently adapts to the screen size and always loads a user interface that inherits the look and feel of iOS capable devices. With the proposed solution, it would be possible to use a different layout specifically for Android devices.

Another improvement done after the first implementation was the development of a specific popup information to the user, in order to know that the web application can be installed on the smartphone. This popup, shows only when the user accesses the web application from an iOS device and from the web browser. The popup stays for 15 seconds and informs the user that user about how to add the application to the home screen. Depending on the device, the popup may point a different position on the screen, show the button that should be touched in order to perform the addition, as shown in Figures 4.13(a) and 4.13(b). For iOS devices, if


Install this web app on your iPhone: tap and then **Add to Home Screen**.

(a) Popup balloon on iPhone.


iOS Simulator – iPad / iOS 5.1 (9B176)

Carrier 3:37 AM

bioinformatics.ua.pt/cardioboxmobile/#

CardioBox

CardioBox

Install this web app on your iPad: tap and then **Add to Home Screen**.

Username edgar

Password enter password

Institution's Name

CardioBox

(b) Popup balloon on iPad.

Figure 4.13: The popup balloon in an iPhone on the left and its correspondent in an iPad on the right.

a web application defines specific HTML5 variables, the mobile device can take advantage of the whole screen and give the user the possibility to *install*. This installation is achieved by the addition of a bookmark to the list of applications of the mobile device, in the home screen. However, there is a significant capability with this feature that can be pretty important in devices with limited screen size. By defining the following lines, Apple's web browser Safari will be able to start the web application in fullscreen, thus removing the lost screen space previously used by Safari's URL bar and bottom bar. Moreover, Apple is able to retrieve the web application's favicon and splash-screen and create an excellent user experience by replicating native applications' behaviour on a web application. Defining these variables has no effect on other devices, so only iOS powered devices will have these specific features:

Listing 4.7: Apple specific HTML5 queries

```
<!--This will try to make our web app in full screen mode-->
<meta name="apple-mobile-web-app-capable"
                    content="yes" />

<!--Setup an icon for the web app (Can be used by iOS
          to make bookmarks to the web app at the home screen)-->
<link rel="apple-touch-icon"
                    href="_images/cbm_icon.png" />

<!--Setup a startup screen for the web app (Can be used by iOS
          to make a splash-screen on loading)-->
<link rel="apple-touch-startup-image"
                    href="_images/cbm_startup.png" />
```

Overall, all these small details that were overlooked at first, made Cardiobox mobile a platform for ECG analysis and enhanced greatly the user experience, by supporting a great amount of devices and, at the same time, optimizing its utilization to one of today's major mobile operating system - iOS.

Chapter 5

Conclusions

The primary focuses of this dissertation were, firstly, the study of the ever-emerging mobile computing field, more particularly the accomplishment of a survey weighting the advantages and disadvantages of all the main multi-platform solutions currently available; and, secondly, the development of a mobile station for visualizing and analyzing ECGs. These goals were fully achieved through the implementation of both a web-service that connects and extends the Cardiobox architecture, and an HTML5 web app capable of interacting, consuming and showing an ECG.

The decision to follow a cross-platform route over a native solution or vice-versa proved to be highly related to the application's core goals, as currently there is no solution that fits all the use cases. However, a cross-platform solution based on HTML5, CSS3 and Javascript, proved to be able to satisfy Cardiobox's requirements. This decision proved to be successful since Cardiobox doesn't require any special access to hardware features like a camera, the GPS, etc.. Also, the time that would be needed in order to learn and implement native versions of Cardiobox for all the major platforms would make it impossible to either finish all the native clients or impact the same market share percentage as the implemented HTML5 version. Moreover, the adaptive capabilities inherent with HTML5 make this solution possible to adjust of different and new platforms, which represents a great advantage in the future.

In spite of focusing on the web client, with this dissertation a scalable architecture was implemented so that some feature work can be integrated with Cardiobox. The web service can be extended to support more ECG formats or implement new features. Also, the login system makes it possible for cardiologists to access pending reports from the central without having the email credentials in its possession. Also, the server is not able to access sensitive information without the interaction of the mobile web application, thus creating a secure and blinded authenticated system. Since Cardiobox's platform deals with sensitive medical information, all the security concerns required to handle medical data were taken into account, such as the communication with the server occurring solely through HTTPS, the IMAP configuration token being ciphered with AES, etc.

Overall, the implemented system represents a simple, viable and financially feasible solution to cardiologists, by supporting the diagnosis of cardiovascular diseases anywhere in the world, making it a major contribution to the construction of a worldwide health care system.

5.1 Final Product

At the end of this dissertation, an architecture is presented through the implementation of a secure SOAP web service capable of interacting with any generic email central and creating a new layer of abstraction for mobile clients to handle ECGs and medical reports. Also, a mobile client is shown and fully described, solely developed in web languages like HTML5, CSS3 and Javascript. This web client works on all the major mobile operating systems such as Android, iOS, and any other operating system equipped with an HTML5 capable browser. This product proved to override all the detected limitations that would be found with a native solution without a web service like, for instance, IMAP and SMTP connections being rejected by firewalls on certain restricted networks, security issues when accessing email servers without SSL, etc..

5.2 Future Work

When developing a real application, some assumptions are made regarding usability. However, these assumptions correspond frequently to logical interpretations made by a developer's perspective, not taking into account what may be users' most relevant needs. So, as a future remark, the main target of attention should be directed to the small usability and stability issues encountered by cardiologists during real medical analyses.

Some other features could be implemented on top of Cardiobox architecture, like for instance taking advantage of different IMAP tags in order to prioritize ECGs with certain keywords like 'urgent', 'important', 'serious', etc. This would make it possible for more relevant ECGs to be picked up and analyzed before hand.

Moreover, in order to improve the user experience, a new layout should be developed replicating Android's look and feel. Through the usage of user agent strings, the proper layout could be loaded, thus avoiding the current iOS layout being displayed on all operating systems.

Finally, HTML5's local storage could be even more explored, where more ECGs would be downloaded in the background, making it possible for cardiologists to work in an offline mode when a connection becomes unavailable. This feature would obviously require special, security-wise concerns, so that sensitive data could not be captured from the device. Also, it would force cardiologists to re-login as soon as the connection becomes back online so that currently pending medical reports could be successfully sent to the central.

Bibliography

[1] Phonegap's build process. https://build.phonegap.com/. Accessed: 20/12/2011.

[2] Html5's new elements. http://coding.smashingmagazine.com/2009/07/16/html5-and-the-future-of-the-web/. Accessed: 12/12/2011.

[3] Electrocardiography. http://en.wikipedia.org/wiki/Electrocardiography. Accessed: 10/01/2012.

[4] Worldwide smartphone sales to end users by operating system. http://www.gartner.com/it/page.jsp?id=2017015. Accessed: 20/06/2012.

[5] Allen S. Graupera V. Lundrigan L. *Pro Smartphone Cross-Platform Development: iPhone, BlackBerry, Windows Mobile and Android Development and Distribution.* New York, Apress., 2010.

[6] Phonegap. http://phonegap.com. Accessed: 14/12/2011.

[7] Appcelerator. http://www.appcelerator.com. Accessed: 15/12/2011.

[8] Adobe flex. http://www.adobe.com/products/flex.html. Accessed: 14/12/2011.

[9] Puputti K. *Mobile HTML5:Implementing a Responsive Cross-Platform Application.* 2012.

[10] Kromp J. Balaz A. *Don't Panic: Mobile Developer's Guide to the Galaxy, 10th Edition.* Enough Software., 2012.

[11] Android developer's platform. http://developer.android.com/. Accessed: 10/12/2011.

[12] Apple developer's platform. https://developer.apple.com/iphone. Accessed: 10/12/2011.

[13] Windows 8 developer's platform. https://msdn.microsoft.com/en-us/windows/apps. Accessed: 10/12/2011.

[14] Windows phone developer's platform. https://create.msdn.com. Accessed: 10/12/2011.

[15] Bada developer's platform. http://developer.bada.com/. Accessed: 10/12/2011.

[16] Blackberry developer's platform. http://blackberry.com/developers. Accessed: 10/12/2011.

[17] Blackberry 10 os based on qnx. https://en.wikipedia.org/wiki/BlackBerry_10. Accessed: 10/12/2011.

[18] Symbian developer's platform. https://www.forum.nokia.com/symbian. Accessed: 10/12/2011.

[19] webos developer's platform. https://developer.palm.com. Accessed: 10/12/2011.

[20] Android sdk for developers. http://developer.android.com/sdk. Accessed: 11/2011.

[21] Google's project named html5 rocks. http://www.html5rocks.com/. Accessed: 10/12/2011.

[22] Css3 media queries. http://www.w3.org/TR/css3-mediaqueries/. Accessed: 16/12/2011.

[23] D. Davis. *Quick and Accurate 12-lead ECG Interpretation.* 2005.

[24] T.D. Jardins. *Cardiopulmonary Anatomy Physiology, Fourth Edition.* 2002.

[25] Einthoven's triangle. http://medical-dictionary.thefreedictionary.com/Einthoven's+triangle. Accessed: 10/01/2012.

[26] N. Menche. *Medicina Interna e Cuidados de Enfermagem.* Loures, Lusociência., 2004.

[27] L. Thelan. *Enfermagem em Cuidados Intensivos: Diagnóstico e Intervenção. 2a Edição.* Lisboa, Lusodidacta., 1996.

[28] Gussak I. *Cardiac Repolarization: Bridging Basic and Clinical Science.* 2003.

[29] Freeware ecg viewer for the xml fda format. http://www.openecg.net/WS2_proceedings/Session05/S5.5_PA.pdf. Accessed: 15/01/2012.

[30] Joel M. *Cardiac Safety Of Noncardiac Drugs.* 2005.

[31] Electrocardiogram ecg types. https://play.google.com/store/apps/details?id=DOCECG2.doctor. Accessed: 17/01/2012.

[32] Heart ecg handbook - lite. https://play.google.com/store/apps/details?id=it.parisi.ecgguide_lite. Accessed: 17/01/2012.

[33] Ecg. https://itunes.apple.com/pt/artist/vandfald.net/id455605324. Accessed: 18/01/2012.

[34] Airstrip technologies. http://www.airstriptech.com/. Accessed: 29/12/2011.

[35] Ieeta. http://www.ieeta.pt/. Accessed: 15/09/2011.

[36] Wan L. Andry F. and Nicholson D. *A mobile application accessing patients' health records through a REST API.* 2011.

[37] Opera mini simulator. http://www.opera.com/developer/tools/mini/. Accessed: 23/11/2011.

[38] Saad M. Hamad H. and Abed R. *Performance Evaluation of RESTful Web Services for Mobile Devices.* 2010.

[39] Iskandar A. Hermawan K. and Hartono R. *Development of ECG Signal Interpretation Software on Android 2.2.* 2011.

[40] Http status codes. http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html. Accessed: 4/07/2012.

[41] Pdfsharp. http://www.pdfsharp.com/PDFsharp/. Accessed: 15/06/2012.

[42] Espresso. http://macrabbit.com/espresso/. Accessed: 11/2011.

[43] Aptana studio 3. http://www.aptana.com/products/studio3.

[44] Webkit information page. https://en.wikipedia.org/wiki/WebKit. Accessed: 15/05/2012.

[45] Multi-touch support on browsers. http://www.html5rocks.com/en/mobile/touch/. Accessed: 15/05/2012.

[46] Multi-touch lacking support for certain android devices. http://code.google.com/p/android/issues/detail?id=11909. Accessed: 29/06/2012.

[47] Modernizr - javascript library for html5 & css3 feature detection. http://modernizr.com/. Accessed: 29/06/2012.

[48] Modernizr - list of html5 & css3 features detected by modernizr. https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills. Accessed: 29/06/2012.

[49] Schmitt C. and Simpson K. *HTML5 Cookbook.* O'Reilly Media, Inc., 2012.

[50] Pilgrim M. *HTML5: Up and running.* O'Reilly Media, Inc., 2010.

[51] Oliveira JL. Costa C. *Telecardiology over Internet ubiquitous services.* International Journal of Medical Informatics, 2011.

[52] Fang Q. Sufi F. and Cosic I. *A Mobile Device Based ECG Analysis System.* I-Tech, Vienna, Austria, 2008.

[53] Fulton J. Fulton S. *HTML5 Canvas.* I-Tech, Vienna, Austria, 2011.

[54] Leblon R. *Building advanced, offline web applications with HTML 5.* 2010.

[55] Html5 or native for mobile development? http://www.google.com/events/developerday/2010/prague/sessions/html5-or-native-mobile-dev.html. Accessed: 29/02/2012.

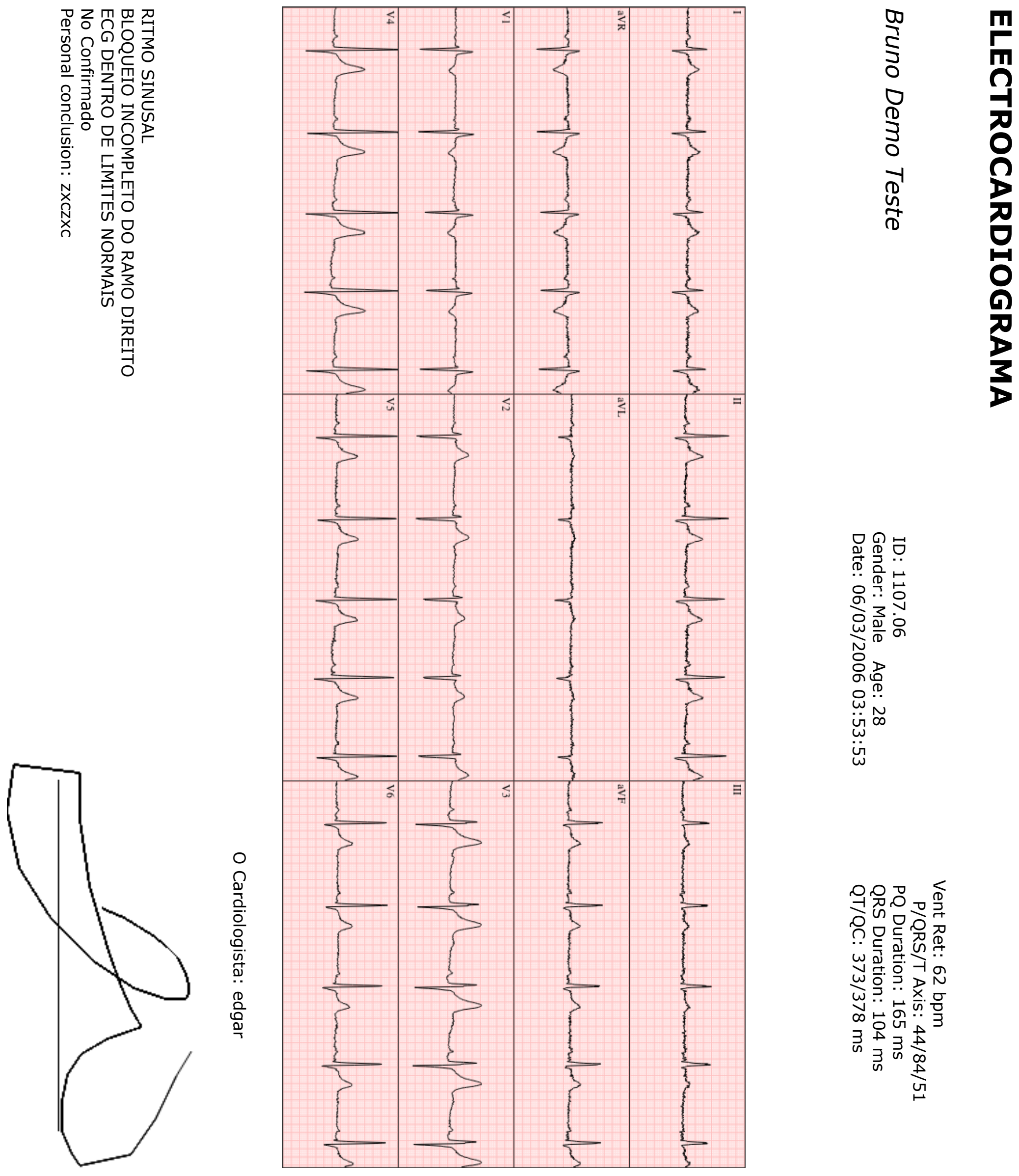
[56] Allen S. Graupera V. and Lundrigan L. *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution.* Apress, Berkely, CA, USA, 2010.

[57] Titanium's developer platform. `http://www.appcelerator.com/platform/titanium-sdk/`. Accessed: 10/12/2011.

[58] Statistics regarding ios versions. `http://www.marco.org/2011/08/13/instapaper-ios-device-and-version-stats-update`. Accessed: 10/04/2012.

Appendix A

Sample Medical Report



ELECTROCARDIOGRAMA

Bruno Demo Teste

ID: 1107.06
Gender: Male   Age: 28
Date: 06/03/2006 03:53:53

Vent Ret: 62 bpm
P/QRS/T Axis: 44/84/51
PQ Duration: 165 ms
QRS Duration: 104 ms
QT/QC: 373/378 ms

O Cardiologista: edgar

RITMO SINUSAL
BLOQUEIO INCOMPLETO DO RAMO DIREITO
ECG DENTRO DE LIMITES NORMAIS
No Confirmado
Personal conclusion: zxczxc

Figure A.1: Sample medical report generated by the server.

Appendix B

ECG Visualizer in iPad

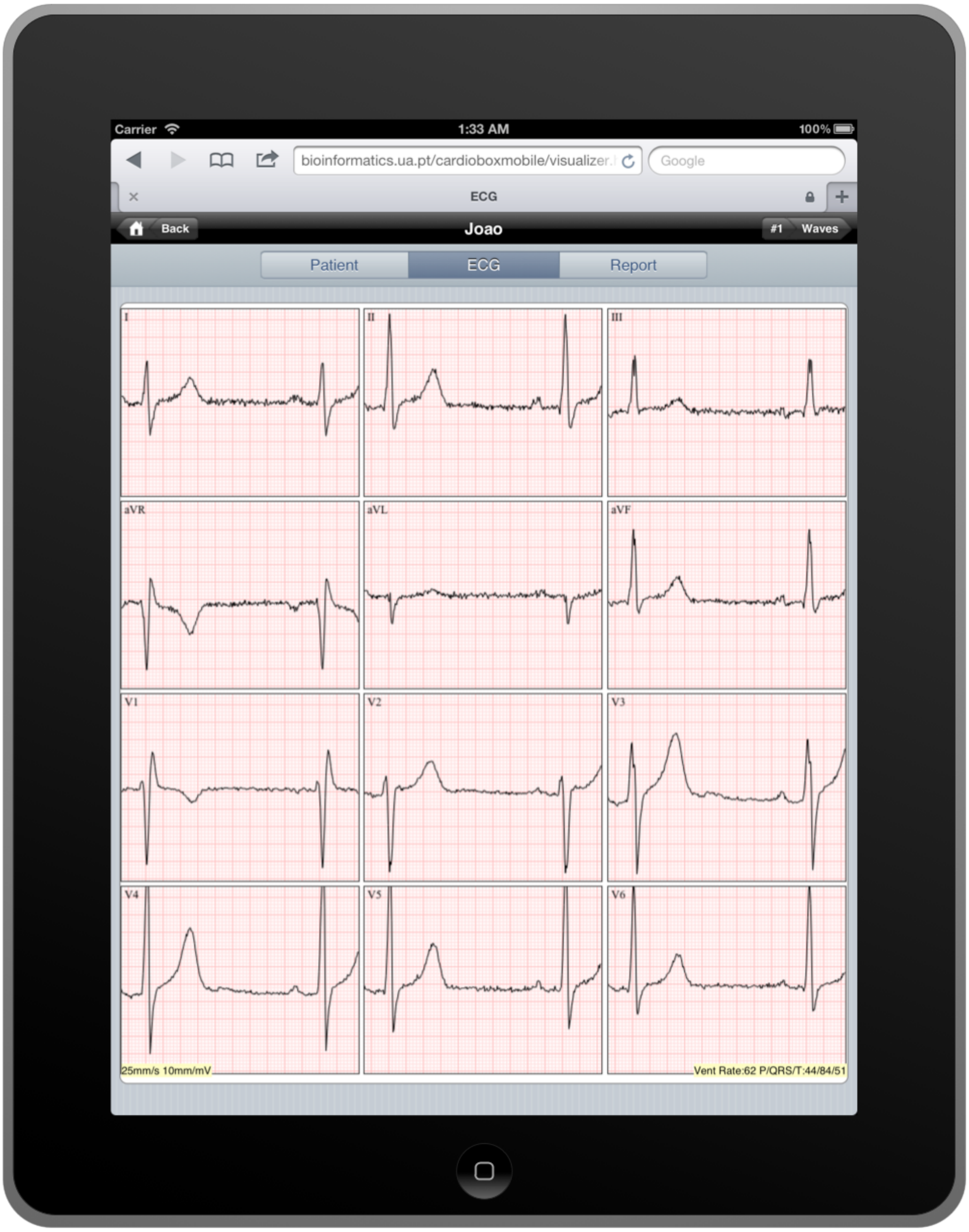


Figure B.1: ECG visualizer as seen on the iPad.