TÉCNICO LISBOA

Biony: A Web Application for Biosignal Processing

Rodrigo Domingues Oliveira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Sandra Pereira Gama

Examination Committee

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Sandra Pereira Gama
Member of the Committee: Prof. Daniel Simões Lopes

January 2021

Acknowledgments

First, I would like to dedicate this document to my family. You were the reason why I was able to accomplish what I did and for that I shall be forever grateful. Thank you so much for all of your sacrifices and for everything you did to ensure that I would be able to start and finish this chapter of my life. This is for you.

I also want to thank my dissertation supervisor Prof. Sandra Gama and Tomás Alves for their support and guidance. Without you I would neither be able to express and implement my ideas like I did nor finish this dissertation.

I would like to thank everyone who participated in this work, either during the tests or during the co-creation workshops.

Last but definitely not least, I would like to express my gratitude towards the friends that I met throughout these five years. Thank you for being there for me and for all of the memories that we created together. Thank you so much, David, Francisco, Hugo, João and Rita.

Abstract

Physiological data analysis has become a multidisciplinary area of research in conjunction with the growth of wearable technology while physiological sensing applications are being adopted in the health and sports fields to monitor physical conditions. Nevertheless, state-of-the-art applications render data analysis as a cumbersome and difficult task by requiring specialists to rely on their programming skills or to interact with limited platforms. This work presents the development of Biony, a web-based application for visualizing, analyzing, sharing and storing biological signals. It features innovative web technologies, such as Ruby On Rails and MongoDB, as well as BioSPPy toolkit for biosignal processing, thus supplying an easy-to-use platform for different levels of expertise. Weighting how co-creation techniques can increase the chances of long-term adoption, we included four different specialists in our prototype development. Additionally, we conducted a usability study (N=27) to evaluate the final prototype regarding perceived usability, ease-of-use, usefulness, and mental workload. Results indicate that Biony was perceived to be useful, very easy to use, and that participants experienced a small workload whenever interacting with it.

Keywords

Biosignals, visualization, signal processing, web application, physiological computing, co-design

Resumo

A análise de dados fisiológicos tornou-se numa área de pesquisa multidisciplinar em conjunto com o crescimento da tecnologia portátil, enquanto as aplicações de medição fisiológica variam desde aplicações em áreas da saúde como no desporto para monitorizar as condições físicas. A análise de tais dados é dificultada com base no estado-da-arte atual, que exige que os especialistas dependam das suas habilidades de programação ou interajam com plataformas limitadas. Este trabalho apresenta o desenvolvimento de Biony, uma aplicação web para visualização, análise, partilha e armazenamento de sinais biológicos. Biony faz uso de tecnologias inovadoras, como Ruby On Rails e MongoDB, em conjunto com o BioSPPy para processamento de biossinais, fornecendo assim uma plataforma fácil de utilizar para diferentes níveis de conhecimento. Tendo em conta a forma como as técnicas de cocriação aumentam a adoção de tecnologias a longo prazo, incluímos quatro especialistas diferentes no desenvolvimento do nosso protótipo. Além disso, conduzimos um estudo de usabilidade (N = 27) para avaliar o protótipo final em relação à usabilidade percebida, facilidade de utilização, utilidade e carga de trabalho mental. Os resultados indicam que Biony foi considerado útil, muito fácil de utilizar e que os participantes tiveram uma pequena carga de trabalho ao interagir com o mesmo.

Palavras Chave

Biossinais, visualização, processamento de sinais, aplicação web, computação fisiológica, co-design

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interfaces

Ajax Asynchronous JavaScript

D3 Data-Driven Documents

DOM Document Object Model

EBS Extensible Biosignal File Format

ECG Electrocardiogram

EDF+ European Data Format 'plus'

EEG Electroencephalogram

ERB Embedded Ruby

GSR Galvanic Skin Response

HCI Human-Computer Interaction

HDF5 Hierarchical Data Format

HR Heart Rate

IT Instituto de Telecomunicações

iOS iPhone OS

JS JavaScript

MEG Magnetoencephalography

MFER Medical Waveform Format Encoding Rules

MVC Model-View-Controller

NASA-TLX NASA-Task Load Index

ODM Object-Document Mapper

PEOU Perceived Ease Of Use

PU Perceived Usefulness

RTLX RAW-TLX

RoR Ruby-on-Rails

SUS System Usability Scale

SpO2 Blood Oxygen Saturation

TAM Technology Acceptance Model

UUID Universally Unique Identifier

1

Introduction

Contents

Biological signals, also known as biosignals, provide communication between biosystems and are our primary source of information on their behavior [12]. Its applicability is vast as biosignals can be used as medical monitoring tools, where they are able to identify potential pathological conditions that may happen, such as seizures [13] and heart attacks [14]. While said monitoring is of utter importance for healthcare, it can also be applied on sports for improving performance [15]. Real-time analysis and processing of biosignals can be used towards helping people with different disabilities enabling them to control prostheses [16] and electric wheelchairs [17] with an electroencephalogram-based control methodology. Biosignals are also used on gaming industry, either by introducing an adaptable in-game difficulty [18–20] or by predicting playing time and game preferences [21].

Biosignals can be acquired through different ways, but often contain noise, being one of the reasons they need to be processed [22]. Current state-of-the-art available solutions can either solely rely on the researcher programming experience to directly deal with programming languages such as C++ and Python [23–26] or rely on different platforms to process and visualize biosignals that do not depend on programming skills [4–9]. However, none of them offer a web-based application structure where a specialist is capable of visualizing, analyzing, sharing and storing biosignal data by only accessing a public domain. This work aims to develop such web-based application, interacting with an existing toolbox for biosignal processing written in Python called BioSPPy [26], which applicability extends to a vast multidisciplinary area, and providing an easy-to-use tool to any specialist. To increase the chances of long-term adoption by specialists, this web application was developed by following a co-creation methodology previously evaluated by Molina et al. [27]. Such methodology introduced the participation of four specialists into this work throughout two workshops, taking their feedback into consideration to create two different prototypes along the process. In order to valuate our prototype, we conducted a user study where it was also evaluated with the System Usability Scale, the Technology Acceptance Model and a modification of NASA-Task Load Index to verify if it was perceived to be useful, easy to use and not effort demanding.

1.1 Objectives

The main objective of this work is **to develop a biosignal web-based application**.

In order to accomplish the main objective, it is necessary to create intermediate goals taking both the current *state-of-the-art* of biosignals web-based applications and tasks that the platform should fulfil into account that will guide this work development, namely:

- Model and create a robust backend design;
- Create an interface capable of communicating with a existing biosignal processing toolbox, namely the BioSPPy toolbox;

- Model and create a frontend design;
- Elaborate a User Study to understand if Biony achieves its objective.

1.2 Document Structure

The remaining of the document is organised as follows: Section 2 provides a theoretical background knowledge regarding core concepts of this work by explaining how physiological computing and biosignals have been studied, exemplifying its usage and acquisition. Section 3 contains a run-through of previously presented work regarding biosignal processing, storage of biosignals and web-based applications, discussing about their achievements, which technologies were used, which features were included, structure, their connectivity, complexity and their presentation. Section 4 presents the co-creation process of this work, describing its structure that will follow a previously studied methodology together with an overview of developed prototypes and how they were conceived, while Section 5 provides different manners of evaluating the solution and introduces this work's limitations. Finally, Section 6 describes conclusions and relevant aspects that can be drawn concerning this document followed by a description of potential future work and directions that should be taken into consideration.

2

Fundamentals of Physiological Computing

Physiological computing is an innovation of Human-Computer Interaction (HCI) where system interaction is achieved by monitoring, analysing and responding to psychophysiological activity from the user in real-time [28]. While physical computing deals with the study and development of interactive systems that sense and react to the analog world [29], physiological computing focuses specifically on the human body with biosignals, which opens up a whole separate class of problems, such as the need for higher signal-to-noise ratios or greater accuracy in the sampling rate [30].

Biosignals are measurable signals from the body. They are used in healthcare and research for patient's monitoring, behavioural signal analysis and physical diagnosis [31]. There is a broad amount of signals that come into the criteria of biosignals, such as Electrocardiogram (ECG), Heart Rate (HR) and Blood Oxygen Saturation (SpO2) [6]. Some relevant biosignals in the context of HCI are [32]:

- *Bioelectrical signals*: signals that originate in nerves and muscles, such as the electrocardiogram and the electroencephalogram.
- *Electrical conductance*: Electrodermal activity refers to the variation of the electrical conductivity of the skin, in particular electrodermal resistance and electrodermal potential. Galvanic Skin Response (GSR) measures their combined values by measuring the resistance on the skin.
- *Bioimpedance signals*: the resistance measured when applying a small alternating current to tissue (typically μA and frequencies above 50kHz), such as the heart rate, blood flow and breathing.
- *Bioacoustic signals*: sounds created by changes in the body, such as blood flow, heart function, ventilation in the lungs, digestion, and movement, can be detected with microphones.
- *Bio-optical signals*: signals that capture the change in optical properties (even if not visible to the human eye) of an organism or a body part, for example, blood-oxygen saturation based on reflection or pulse rate by change in skin color.
- *Kinematic data*: two or three-dimensional measures that includes displacement and orientation of body segments, joint angles and spatio-temporal gait parameters.

Measurement of such biosignals can be acquired through different procedures, such as invasive (enters the body) or non-invasive (do not involve tools that break the skin or physically enter the body) and intrusive (significant impact on the mobility or comfort of the person involved as shown in Fig. 2.1) or non-intrusive (minimal impact on the person involved).

Taking into consideration that invasive and intrusive procedures will most likely cause modifications to the normal body state and prejudicate the overall outcome of measuring biosignals in view of the discomfort introduced by these procedures, non-invasive and non-intrusive procedures, which have little to no impact on someone's body state, are being studied and developed as technology continues to grow in order to obtain legitimate data since these procedures.

Baek et al. [3] presented a non-intrusive biosignal monitoring system for drivers by installing sensors on the steering wheel, driver's seat, and seat belt, illustrated in Fig. 2.2. Watanabe et al. [33] developed a non-invasive pneumatics-based system capable of measure heartbeat, respiration, snoring, and body movements of a subject in bed. Birjandtalab et al. [2] collected data using non-invasive wrist worn biosensors, shown in Fig. 2.1. Zhang et al. [34] went a step further and used non-invasive saliva nano-biosensors to monitor glucose. All of the aforementioned apparatuses try to minimize their impact on a person mobility and comfort in order to not interfere with any measurement. The device presented by Rantanen et al. [1] for measuring the intensity of facial activity is a good example of an apparatus that can cause discomfort, leading to an abnormal body state and therefore producing biosignals that would not be verified on a daily basis without wearing any device.

Figure 2.1: Apparatus for BCI. a) Head-mounted measurement device presented by Rantanen et al. [1]. b) The wrist worn Affectiva used by Birjandtalab et al. [2].

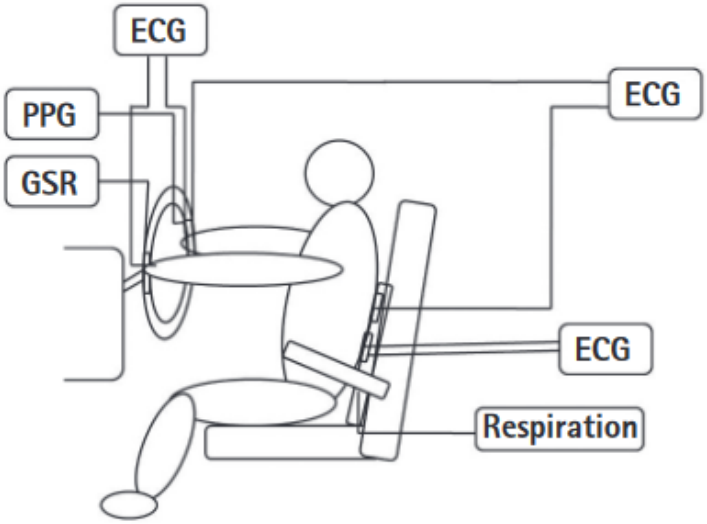

Figure 2.2: Setup of the physiological signal acquisition system in [3]. GSR, galvanic skin response; PPG, photoplethysmogram; ECG, electrocardiogram.

3

Related Work

Contents

This section covers pertinent work regarding biosignal processing and web-based applications that were developed to date for biosignal storage and visualization.

3.1 Biosignal Processing

Biosignals are primarily acquired for monitoring, detecting or estimating specific pathological/physiological states for purposes of diagnosis and evaluating therapy. Typical applications of biosignal processing are [22]:

- Noise removal.
- Accurate quantification of signal model and its components through analysis in order to identify the system for modeling and control purposes.
- Feature extraction for deciding function or dysfunction.
- Prediction of future pathological or functional events as in prosthetic devices for heart and brain.

Several toolkits were presented to date, e.g. the BioKIT [23] focused on biosignals such as speech, motion, muscle and brain activities. Kaldi toolkit [24], a flexible open-source toolkit written in C++, and RWTH Aachen toolkit [25], a software package of several libraries and tools written in C++, are mainly focused on speech recognition research. BioSPPy is a toolbox for biosignal processing written in Python [26] with a set of functions to enable students with engineering and non-engineering backgrounds to easily perform operations, such as filtering, onset detection, feature extraction, among other [11].

The main difference between BioSPPy and other toolkits is that BioSPPy was created in order to help users with any background to perform different biosignal processing activities by resorting to a simpler programming language, Python. Similarly to BioSPPy, BioKIT uses an extra layer with Python scripting on top of its C++ core to facilitate development of new applications. Even though Kaldi toolkit and RWTH Aachen toolkit are focused on producing flexible code, their main goal is not exactly to facilitate operations to users without any programming background. Based on this work goals, BioSPPy toolkit will be further discussed and investigated.

BioSPPy builds on the concepts of keywording function parameters and return values, using the parameter's name (keyword) instead of its specific position whenever calling a function, shown in Fig.3.1. All functions are prepared to receive the input arguments either as an ordered sequence according to the function definition, or as a set of key/value pairs in which the key provides the semantic context of each parameter. The outputs of the function are also returned as key/value pairs, allowing the user to access the results of the function computation using their semantic association [11].

```
from biosppy import storage
from biosppy.signals import ecg

# load raw ECG signal
signal, mdata = storage.load_txt('./examples/ecg.txt')

# process it and plot
out = ecg.ecg(signal=signal, sampling_rate=1000., show=True)
```

Figure 3.1: The code above loads an ECG signal, filters it, performs R-peak detection, computes the instantaneous heart rate and produces a plot for it.

Da Silva et al. [11] evaluated the BioSPPy toolbox using the System Usability Scale (SUS) approach, demonstrated in Table 3.1, to measure how users perceive the usability of a given system [35]. One hundred participants were equally divided into four different groups: (a) *Biomedical engineering* (BME): users familiar with electronics, computer programming and biosignals; (b) *Electrical engineering* (ECE): users mostly familiar with electronics and computer programming; (c) *Computer science* (CS): users mostly familiar with computer programming; and (d) *Health sciences* (HS): users familiar with biosignals but non-proficient in computer programming nor electronics [11].

As expected, *Biomedical engineering* users obtained the best SUS Score whilst the *Health sciences* users obtained the poorest. Even though BioSPPy was designed as a way of introducing a semantic nature into individual or groups of operations that are typically used with biosignals [11] in order to address the long learning curve that most students without computer programming background face, users without such a background (*Health sciences*) were not completely comfortable with BioSPPy itself. As shown in Table 3.1, their SUS score can be rated as OK or as a D using a school grading scale [36], thus it is evident that *Health sciences* users would appreciate being able to use the BioSPPy toolbox without interacting with Python and having an appropriate Graphical User Interface instead.

Table 3.1: Results of the usability study for BioSPPy showing the ranking obtained from SUS in terms of mean and standard deviation ($\mu \pm \sigma$). The rating is on a five-point scale from "strongly disagree" to "strongly agree", partial table from [11].

Group	BME	ECE	CS	HS
I think that I would need the support of a technical person to be able to use this system.	1.92 ± 0.91	2.44 ± 1.08	2.00 ± 1.19	2.92 ± 1.16
I would imagine that most people would learn to use this system very quickly.	4.20 ± 0.65	3.72 ± 1.44	3.80 ± 1.15	3.15 ± 1.08
I felt very confident while using the system.	4.20 ± 0.76	3.88 ± 0.88	4.16 ± 0.75	3.38 ± 0.80
SUS Score	83.40 ± 11.88	73.80 ± 15.88	79.30 ± 15.90	64.33 ± 20.34

3.2 Biomedical Web-Based Applications

Web technologies continues to grow in popularity as it only requires a compatible web browser. It is much easier for one to access a specific domain and make use of the application capabilities online than it is to install a specific software that has its minimum system requirements met. In some cases, such as in hospitals, their computer systems are restrictive to maintain integrity and users usually cannot install their own software since they work with sensitive data [37]. The need of an online connection will be further discussed in this subsection as it may be disadvantageous.

Alves et al. [4] presented SignalBIT, a web-based platform for real-time biosignal visualization and recording. The framework was designed under a Model-View-Controller (MVC) architecture, decoupling the presentation from the processing and persistency layers, thus dividing the system into three main modules:

- *View*: a front-end based on Web technologies that displays the user interface and allows all the interaction.
- *Controller*: where all the events triggered in the user interface are mapped into operations.
- *Model*: which coordinates the application logic by evaluating the messages received by the controller, executing the operations and producing results.

SignalBIT user interface, shown in Fig. 3.3, was designed recurring to HTML5 and CSS3 [4]. It offers several possible interactions with the displayed plot, such as changing the time scale, the scale amplitude by zooming or the offset of the graphic, adjusting the baseline of the signal. The controller, as described before, acts between the view and the model, and was implemented in JavaScript (JS) using universally recognized libraries such as jQuery [38] and Flot, a JS plotting Application Programming Interfaces (API) for jQuery. The communication is made via WebSocket exchanging JSON messages, a standard data-interchange format, used in a large variety of programming languages, and then the operation is processed by the model in Python. The system architecture can be better visualized in

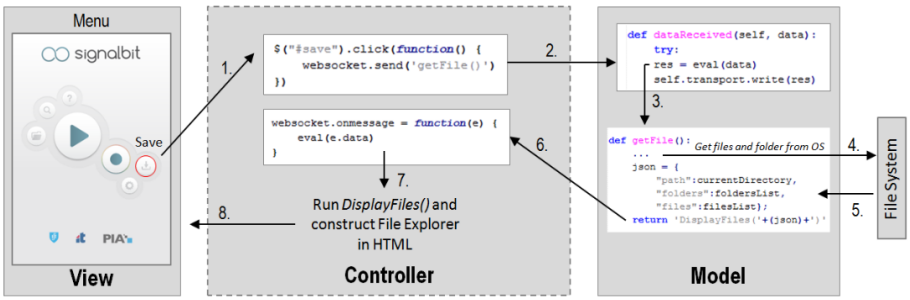

Figure 3.2: SignalBIT System Architecture [4] example. Pressing the save button on the view initiates an action on the controller that interacts with the model through a websocket. Afterwards, the model notifies the controller whenever it finishes through the same websocket and then the controller finally executes an action to interact with the view.

Fig. 3.2. Even though results show that the system was well accepted and easy-to-use, without requiring any specific knowledge or background in computer-based signal acquisition software [4], its backend component can be considered complex, since the programmer directly deals with WebSockets and communication protocols (*Twisted*).

Rosa et al. [39] presented a web-based framework to preparate gait or shoulder movement analysis reports using the benefits of the biomechanical modeling from OpenSim, an open-source software to create and analyze dynamic simulations of movement [40]. Designed under a MVC architecture, likewise SignalBIT, its backend is built upon Python programming language and its frontend uses Impress.js, a JS library that controls the style and layout of the web page. It has innumerous similarities to SignalBIT, using WebSockets and the same *Twisted* communication protocol.

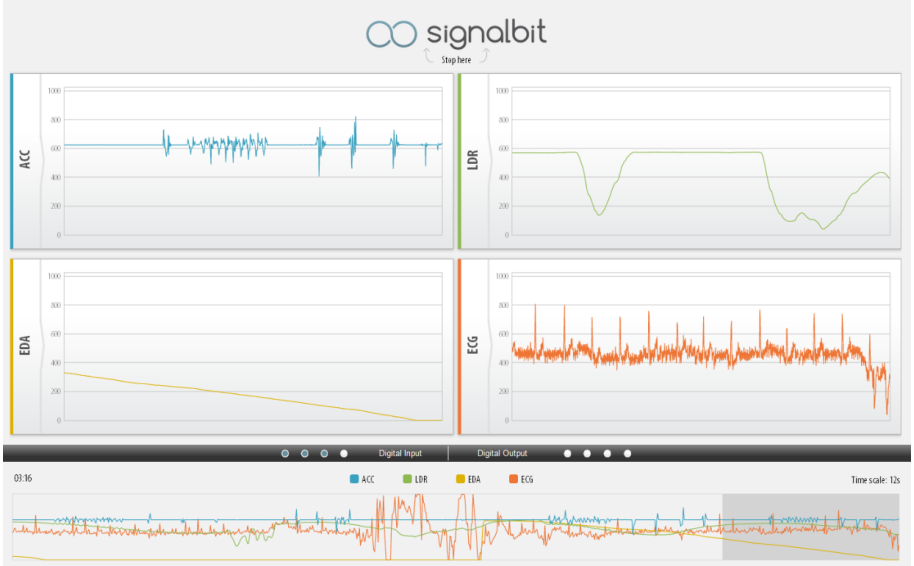

Figure 3.3: SignalBIT application User Interface [4]. Four different channels plotting individual biosignals and a single channel plotting all of them together.

Beier et al. [5] presented a web-based platform, shown in Fig. 3.4, for biosignals visualization that requires no server-side processing and was implemented in React, a JS library for building user interfaces, in order to facilitate its development in separate components that can be used independently or in composition. The chosen data format was JSON since one object contains an unlimited number of events as key-value pairs. The system can work offline in order to guarantee that there is no data-leakage. The Biosignal Igniter Toolkit (BIT) [11] also takes this availability aspect into account providing APIs for real time data acquisition.

Figure 3.4: User Interface presented by Beier et al. [5]. Multiple channels for different biosignals, while multiple detected features are shown on the left.

Overall, such systems have different availability requirements, some can be accessed with and without an internet connection [5, 11] while others can only be accessed with an internet connection [6, 7]. The platform described by Mata et al. [6], shown in Fig. 3.5, makes use of mobile computing, and needs both backend and frontend processing. While using mobile computing is a great way to innovate it can be really disadvantageous since the system described only works on Android and neither evaluate nor analyse the power consumption which is a vital component when using mobile computing.

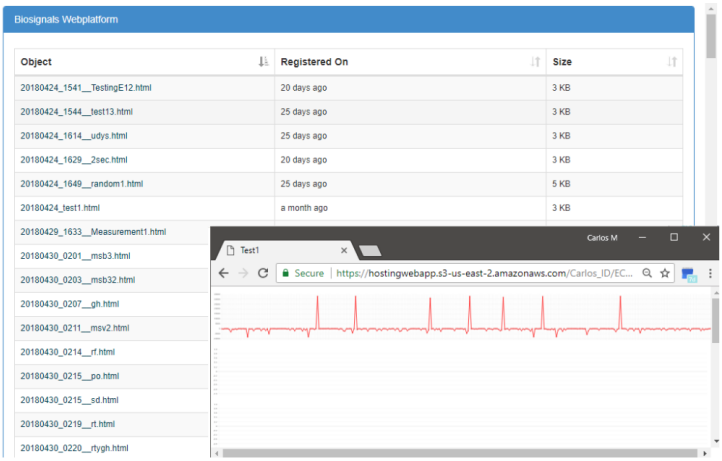

Figure 3.5: Screenshot of WebPlatform presented by Mata et al. [6]. A simple plot for an individual biosignal.

Garza et al. [7] presented a system mainly focused on Electroencephalogram (EEG) data visualization and analysis that works both on iPhone OS (iOS) and on a regular desktop version, as shown in Fig. 3.6. The power consumption component was not analysed for the mobile version. While using the

desktop version the user is capable of visualize EEG data in the database and perform general data analysis. Its mobile application is not yet totally integrated with the website, therefore the user is only capable of visualize and collect data. Since visualizing data on a mobile platform can be difficult due to its physical size, it is possible to display data in *text mode* by viewing individual data frames.

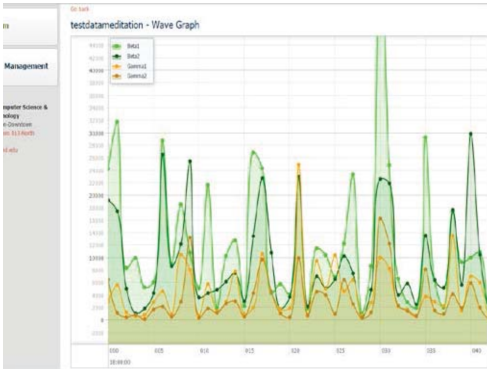

Figure 3.6: EEG Wave Forms Rendering in Web Interface presented by Garza et al. [7].

Gomes et al. [8] developed tools to visualize and process long-term biosignals while proposing a new data structure for storing them. Based on the HDF5 format, the proposed data structure gives more attention to the biosignal's metadata with signal acquisition parameters and information about its recording, achieving similar results to StorageBIT [41]. The visualization tool was developed using a client-server model, illustrated in Fig. 3.7, using Python on its backend and Javascript with HTML on its frontend, focusing on zooming and panning features to enable the users to explore and analyze long-term biosignals, though it is not limited to large sized signals.

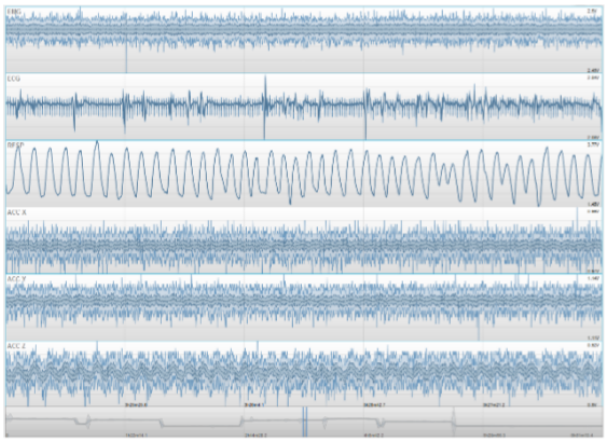

Figure 3.7: Developed biosignal visualization tool by Gomes et al. [8]. Multiple channels plotting individual biosignals.

Later on, Cavaco et al. [9] also developed a web-based platform, which can be better visualized in Fig. 3.8, following the client-server model using Python on its backend and Javascript on its frontend to visualize and process long-term biosignals and, similarly to Gomes et al. [8], proposed a new data

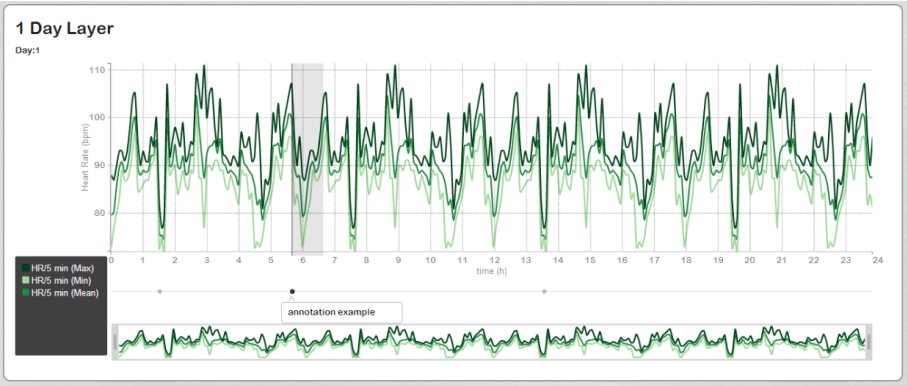

Figure 3.8: Developed biosignal visualization tool by Cavaco et al. [9] using annotations.

structure based on HDF5 while its visualization tool had a focus on zooming and panning features introducing annotations as well as the ability to export a snapshot of its current state to a PDF.

Finally, BioPortal is an open repository of biomedical ontologies [42] that stores ontologies[1], it adopts a layered architecture approach, which decouples the logic and domain object models between each layer. Its user interface uses Ruby-on-Rails (RoR) technology and the system provides API access to the ontologies and the resource index. Users can browse and search the ontologies, submit new versions of the ontologies in the repository, comment on any ontology (or portion of an ontology) in the repository, add a review of the ontology, describe their experience in using the ontology or make suggestions to ontology developers.

3.3 Storage of Biosignals

File format standardization for the storage and sharing of biosignals continues to be one of the most challenging tasks whenever dealing with biosignals [44]. Several file formats that are mainly focused on the core biosignal data were introduced up to this time.

Designed to support EEG and Magnetoencephalography (MEG) signals, the Extensible Biosignal File Format (EBS) [45] has a tag-length data architecture making it possible to include image or video data. The calibrated recording of dc types of signals such as blood pressure is not possible because of the format's main flaw, an omitted dc-offset field in the description of the biosignals.

The European Data Format 'plus' (EDF+) [46] was introduced in order to address the limitations of its predecessor EDF [47], such as only being able to handle uninterrupted recordings. An EDF file has an ASCII header that identifies the patient and specifies the technical characteristics of the recorded signals. Its data record is a series of 2-byte integer values, all with the same duration specified in the header.

[1] Controlled vocabularies that allow describing the meaning of data in a human and machine readable way to aid processing of information in biomedical research and in healthcare systems [43].

A format specialized in medical waveform data such as ECG and EEG was presented in 2002, the Medical Waveform Format Encoding Rules (MFER) [10]. The format, illustrated in Fig. 3.9, encodes waveform data values in frames which include a header to identify its content. Both header and waveform data are composed by: type (or tag), consists of one or more octets and indicates the attributes of the data value; length, is the length of data values indicated in one to four octets; and value, the content, waveform data or the attribute identified by the tag.

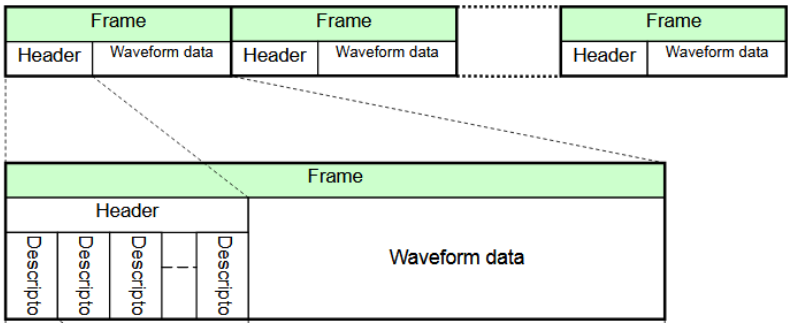

Figure 3.9: Medical Waveform Format Encoding Rules format description [10].

More recently, the general lack of structured metadata describing how all the data was gathered led to the development of new semantic approaches [48–50] where contextual information is included. The Hierarchical Data Format (HDF5) [51] allows for embedding of metadata making it self-describing. Its structure is divided into groups, datasets and associated metadata, as shown in Fig. 3.10. A group contains other groups or datasets within it, while the datasets are the actual data contained within the HDF5 file. Based on this, HDF5 is able to support heterogeneous data since different types of datasets can be contained within one HDF5 file.

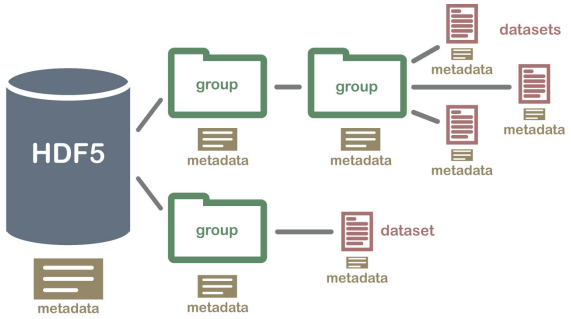

Figure 3.10: The Hierarchical Data Format (HDF5) file structure.

Carreiras et al. [41] proposed StorageBIT, a system that combines HDF5 and MongoDB, a NoSQL database, in order to solve many of the common problems faced by biosignal researchers, such as handling the metadata, no size limits to header and metadata information, and seamless distributed access to the data repositories.

3.4 Discussion

Taking into consideration the aforementioned studies, it is noticeable that every web-based application can be partitioned into technologies and features (Table 3.2), thus, further discussion will respectively bring attention to these two relevant aspects.

The MVC architecture is an effective popular [4, 39, 42] approach that offers multiple advantages to developers, since it enables logical grouping of related actions on a controller together, adds great scalability to the platform with its modularity making it easily modifiable and also allows the same data to be presented as many ways as desired. Thus, the MVC architecture offers more options to developers while creating a well-structured and modular web-based application than a simple client-server model as used in [8, 9].

Developing the entire backend of a web-based application on Python and dealing with communication protocols is a complex task that most of the studies had to overcome, especially when implementing all of the MVC architecture and not resorting on simple client-server models. The valuable time and effort spent on that could have been avoided by using different technologies such as RoR, which was already explored on BioPortal [42].

Previous studies regarding biomedical web-based applications did not explore many formats of storage for biosignals. Instead, most of those studies decided to utilize some variant of HDF5 since it supports large amounts of heterogeneous data while being a portable and extensible format, making the overall development of such web-based applications easier. Carreiras et al. [41] had special attention to different database formats, discussing and evaluating different implementations and concluded that NoSQL is the most indicated database format while working with biosignals because of its scalability and flexibility. It was also pointed out that MongoDB was a faster NoSQL alternative than CouchDB due to the fact that MongoDB uses memory-mapped files, keeping the more frequently accessed information in Random Access Memory (RAM).

Featurewise, being able to create and manipulate annotations within individual records or groups of individual records is an important feature to help experts in biosignals to contribute with their knowledge to annotate the signals with ground truth information [52]. There are two common solutions regarding signal visualization, these being multichannel visualization [4, 5, 8, 9] and single channel visualization [4, 6, 7]. In order to better understand each signal and its metadata, multichannel visualization comes out on top, and to directly compare similarities between signals, a single channel visualization approach is more adequate if it is possible to visualize more than one signal.

Regarding the User Interfaces that were presented on previous studies, SignalBIT [4], the platform presented by Beier et al. [5] and both visualization tools developed by Gomes et al. [8] and Cavaco et al. [9] are more focused towards user interaction with zooming and panning features as well as highlighting the biosignal's metadata. The rest of the presented platforms did not focus on presentation as their

frontends are simple visualizations of different biosignals. Beier et al. [5] brought special attention to different events, like unusual increase, abrupt transitions, saturation, cardiac interference, muscle movement and slow ondulations in the visualized biosignal while [4,8,9] focused more on signals themselves.

Mobile computing is an interesting topic that was already explored [6,7] but based on this work goals, this web application will not support mobile computing.

Table 3.2: Biosignals Web-Based Applications technologies and features.

	Technologies				**Features**			
	Python Backend	**ReactJS**	**Mobile**	**MongoDB**	**Multichannel Vis.**	**Single channel Vis.**	**Offline**	**Annotation**
SignalBIT [4]	✓			✓	✓	✓	✓	✓
Mata et al. [6]			✓			✓		
Beier et al. [5]		✓			✓		✓	
Garza et al. [7]			✓			✓		
Gomes et al. [8]	✓				✓		✓	
Cavaco et al. [9]	✓				✓		✓	✓

4

Co-Creation Process

Contents

This work aims to develop a web-based application capable of visualizing, analyzing, sharing and storing biosignal data to help biosignals specialists and researches who deal with such data routinely. Thus, adopting a co-creation design methodology would assist us to achieve these goals while working together with four different specialists, these will be referred as *co-designers*, throughout different development stages.

Co-creation is a design methodology that focus on collaborating with potential customers or end-users in order to ensure that they play a critical role in the design, and to increase the chances of long-term adoption [27]. It is based on the principles of mutual learning, empowerment, openness and diversity, involvement and ownership, transparency, and effectiveness [53].

Based on the evaluated methodology followed by Molina et al. [27] we conducted two co-creation workshops for collaborative design, which will be further discussed in depth along the next sections of this chapter. By dividing this work development into different releases, it was expected to take advantage from the Incremental Model [54], making it more flexible, easier to test and debug throughout every release. The entire co-creation process is better illustrated in Fig. 4.1.

Figure 4.1: Overview of the co-creation process and development phases in-between. Workshops are in blue, prototyping in grey and user testing in green.

4.1 Ideation Workshop

In order to better understand which tasks and functionalities should be supported by this work, semi-structured interviews with two *co-designers* were conducted before developing Prototype A. A feature wish-list, described below, was created based on current state of the art and on what was discussed on Section 3.4. It was asked to the *co-designers* to talk about and classify each item from the aforementioned feature wish-list from 1 to 5 in terms of relevancy, where 1 corresponds to a completely irrelevant feature and 5 corresponds to a completely relevant feature.

F1 Create a personal user with credentials.

The first *co-designer* (CD1) classified this feature as completely relevant (5) and also mentioned that it could be helpful to have a way to use the platform without the need of authentication. The second *co-designer* (CD2) classified this feature as relevant (4) and talked about the importance of the platform's accessibility, mentioning how it should be easy for any user to access his data from anywhere by just using his credentials. CD2's classification for this feature also took in con-

sideration that implementing such authentication system could demand way too much time and effort.

F2 Create or associate a user by a social network account from websites like *Facebook*.

CD1 classified this feature as completely relevant (5), mentioning how developing an entire authentication system of your own is not that interesting nowadays based on different technologies available. It was also suggested that, if it was possible to authenticate credentials using a external social network it could be interesting to add other options than *Facebook*, such as *Google*, *LinkedIn* and *ResearchGate*. CD2 classified this feature as neutral (3), even though it could be interesting it would also demand a lot of time and effort and because of this, the feature should not be of utter importance for this work.

F3 Create groups of users to share information.

Even though CD1 believes this feature is completely relevant (5), he was worried about adding new barriers for sharing biosignals and suggested that it should be possible to access biosignals from other users by only having access to their links. Once more, CD2 classified this feature as neutral (3) because he sees how it would be helpful to share information with groups but also how demanding it could be to develop such feature.

F4 Use the platform API with and without internet connection.

CD1 classified **F4** as completely relevant (5) without adding too much information but reinforced how using an API to add or obtain biosignal data could be helpful. CD2 classified **F4** as irrelevant (2) based on this work goals, such as creating easier ways to interact with biosignals data throughout a Graphical User Interface.

F5 Multichannel visualization and Single channel visualization.

CD1 classified this feature as completely relevant (5) but mentioned that it is often difficult to determine which biosignals should be grouped into the same temporal channel and so multichannel visualization should have more importance as it is more utilized. CD2 classified this feature as neutral (3) because Single channel visualization could lead specialists to be confused as there would be way too much information into the same channel.

F6 Create annotations for a biosignal.

Both CD1 and CD2 classified this feature as completely relevant (5) based on the importance that annotations could have for different biosignals providing additional and valuable information to them. CD1 mentioned that it could be interesting if the platform was able to recognize some patterns and generate annotations alone.

F7 Adjust the biosignal visualization timescale, zoom-in and zoom-out.

Once more, both CD1 and CD2 classified this feature as completely relevant (5) since being able to interact with biosignals' visualizations is a vital feature to better understand them.

F8 Create groups for biosignals.

CD1 mentioned how it is important to group biosignals that present similarities, citing as an example how grouping biosignals by their morphology and dividing normal biosignals from biosignals with pathological conditions could be useful, and thus classified this feature as completely relevant (5). CD2 said that the feature was relevant (4) and did not add more information.

F9 Customize color scheme for different types of biosignals.

Both CD1 and CD2 mentioned that the platform should have a default color scheme for different types of biosignals but the user should be able to change them if he wants to, and classified **F9** as completely relevant (5) and relevant (4) respectively.

F10 Save biosignal visualization snapshots for future investigation.

This feature was classified as completely relevant (5) by CD1 that stated that sometimes a specialist sets up all of the visualization settings with a specific zoom-in degree and timescale to better visualize something and losing that every time could be frustrating. CD2 said that even though he knows how it could be useful he does not think that **F10** is very important and classified this feature as irrelevant (2).

F11 Change between *dark theme* platform and *light theme*.

CD1 classified this feature as completely relevant (5) because different persons have different tastes and so having both options should be helpful. CD2 classified this feature as irrelevant (2) because he believes that **F11** more related to personal preferences and thus is not important for this work.

Both interview sessions took around thirty minutes and as they continued it is noticeable that both *co-designers* generated some fatigue from talking too much since their feedback was not as well-elaborated for the last features as it was for the first ones. By the end of the interview, CD1 suggested a new feature where users should be able to submit a video together with a biosignal in order to better understand what happened during the biosignal acquisition time.

Overall, the Ideation Workshop was a great way to understand the *co-designers*' desires and expectations for this work while making them comfortable. Since we presented them a feature wish-list from the beginning, the *co-designers* were more confident to talk about the web application itself since they did not have to elaborate it from scratch, which was one of them problems that Molina et al. [27] faced

when going through the Ideation Workshop, where participants seemed to have a hard time developing their own ideas from scratch.

4.2 Prototype A

Developing the entire backend on Python and dealing with communication protocols is a complex task. As presented on Section 3.2, BioPortal [42] was the only platform built with RoR but was not focused on biosignals and so RoR technologies applied to biosignals web-based applications are yet to be explored. Since this is a perfect scenario for a MVC application and RoR is mainly focused on building MVC software, this work explores its features while developing a web-based application for biosignals. The solution executes Python code, mainly related to the BioSPPy toolkit itself, through an interface developed on RoR. MongoDB was the chosen database format since it has been proved to be the most indicated database format while working with biosignals because of its scalability and flexibility [41]. Working with RoR also facilitates the implementation of an API that can be used on a localhost without recurring to user interfaces and without an internet connection.

We developed a simple RoR application capable of receiving a uploaded text file, an ECG sample, via browser and communicating with BioSPPy toolkit on its backend, in order to better understand the *pros and cons* of following the proposed architecture.

BioSPPy toolkit return values are mainly composed of a custom object class called *ReturnTuple* which purpose is, as defined on BioSPPy documentation [26], to strengthen the semantic relationship between function output variables, their names, and what is described in the documentation. In theory, such custom class would facilitate the communication between BioSPPy and other platforms since it would be really easy to convert any return value from BioSPPy to JSON, but that is not the case since BioSPPy uses numpy arrays on its internal logic and numpy arrays are not JSON serializable. Thus, before establishing any communication with BioSPPy, it is necessary to manually convert the entire *ReturnTuple* to a JSON object composed of standard arrays.

As to be expected, the greatest downside of not developing a backend on the same programming language as BioSPPy, Python, is the introduction of communication time between any RoR backend operation that includes BioSPPy as shown in Fig. 4.2. Fifty simulations produced the results demonstrated in Fig. 4.2, the average time execution for three different contexts: (a) normal BioSPPy execution; (b) executing a BioSPPy method and converting its result to a JSON object; and (c) communicating with BioSPPy and receiving its result via RoR.

Even though the communication time increased the total execution time of (c) when compared to contexts (a) and (b) by a great percentage, its total execution time should not affect the overall flow of the final application based on its actual execution that is only a second long and that the vast majority of

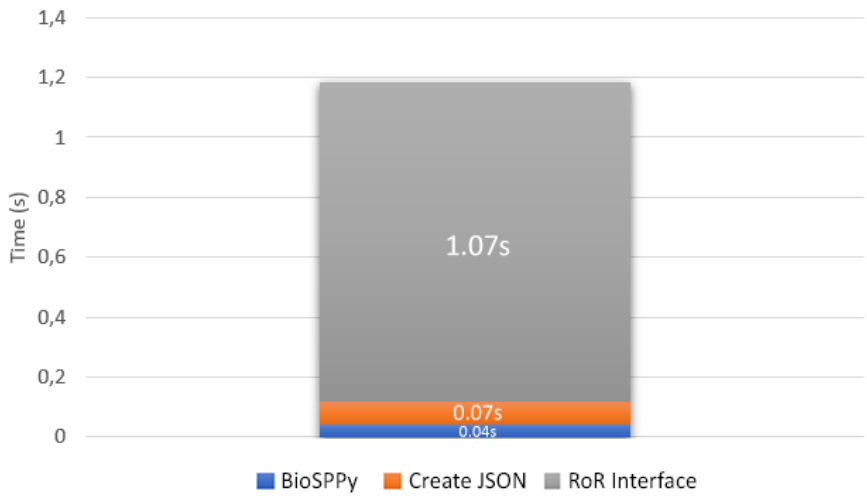

Figure 4.2: Average execution time for three different contexts.

the features described on 4.1 do not include any interaction with BioSPPy at all. Features that depend on BioSPPy will not be used that frequently, since the main goal of this work is to develop a web-based application capable of not only submiting and processing biosignal data but also visualizing, analyzing, sharing and storing.

Following the MVC pattern, a user can interact either with the API or with any View produced by the Web Application. Views interact with Models and Controllers to generate a HTML page using Embedded Ruby (ERB), while the API only interact with a specific Controller. Models interact with Controllers, Views and MongoDB using a famous Ruby Gem[1] known as Mongoid, an Object-Document Mapper (ODM) framework for MongoDB in Ruby[2]. A specific Controller then execute methods from the BioSPPy Interface in order to interact with the BioSPPy toolkit.

Recurring to Ruby Gems facilitates the entire development of the Web Application based on the fact that most of the common functionalities are already implemented. These are some of the core Ruby Gems that were used during the Prototype A:

- ***Mongoid***: an ODM framework for MongoDB in Ruby[2].
- ***Devise***: a flexible authentication solution for Rails[3].
- ***BSON***: an implementation of the BSON specification in Ruby[4].
- ***Puma***: a simple, fast, multi-threaded, and highly concurrent HTTP 1.1 server for Ruby/Rack applications[5].

[1] Ruby libraries with reusable code that offers particular functionalities.
[2] Mongoid: https://github.com/mongodb/mongoid (Accessed on November 12, 2019)
[3] Devise: https://github.com/plataformatec/devise (Accessed on November 12, 2019)
[4] BSON: https://github.com/mongodb/bson-ruby (Accessed on November 12, 2019)
[5] Puma: https://github.com/puma/puma (Accessed on November 12, 2019)

Since this was the first release, it was expected to feature core functionalities, such as an authentication system, biosignals visualization and storing, creating groups of users and groups of biosignals together with a simple and flexible User Interface. Based on the MVC architecture, this platform has three main components: models, views and controllers. Each one of these components will be further discussed in order to explain how Prototype A was conceived.

4.2.1 Models

Documents are the core objects in Mongoid and represent models. The representation of a Document in MongoDB is a BSON object that is very similar to a Ruby hash or JSON object. Documents can be stored in their own collections in the database, or can be embedded in other Documents n levels deep [6]. Such description perfectly fits what can be described as an HDF5 variant, thus, the data structure developed in this work was based on the HDF5 file format while using Mongoid in RoR.

A User document has three different array fields with *ObjectIds* for teams, groups of biosignals and biosignals tagged as favorites. It also contains a *"has many"* relation with Biosignals documents, which means that a User has n number of referenced Biosignals. Authentication related fields, such as email and password, were automatically created by Devise and thus will not be further discussed.

Biosignal documents have four different fields for describing its metadata, that includes name, date, notes (any other additional information about the biosignal) and signal type. The other two fields are related to the biosignal itself, one of them is the raw signal, which is the uploaded file before being processed by BioSPPy toolkit, and the other is the processed data, which contains a dynamic JSON object for the biosignal processing response from BioSPPy.

A BiosignalGroup document describes a group of biosignals created by a user and contains two fields for describing its metadata, which are the name and notes. It also contains two array fields with *ObjectIds* for teams that it was shared with and biosignals that are within the group itself.

Finally, a Team document has three different fields for its metadata, which are the name, notes and total number of elements that are part of the team. The Team document also contains three different array fields with *ObjectIds*, one for users that part of the team, and other two for groups of biosignals and biosignals that were shared with the team.

4.2.2 Views

Taking all of the aforementioned feedback from the interviews into account we designed a prototype for this platform's Graphical User Interface. Prototyping is an important technique to reduce the cost and risk involved in developing complex software systems [55]. It essentially involves building a small scale

[6] Mongoid: https://mongoid.github.io/old/en/mongoid/docs/documents.html (Accessed on October 26, 2020)

version of a complex system in order to acquire critical knowledge required to build the system [56].

As illustrated in Fig. 4.3, the profile page was initially designed to contain four main blocks: one with the user's overall information, such as name and email; one block that contains information about the user's linked accounts to different social networks; one with customization options regarding the color scheme for different types of biosignals; and finally, one block to change between dark and light theme.

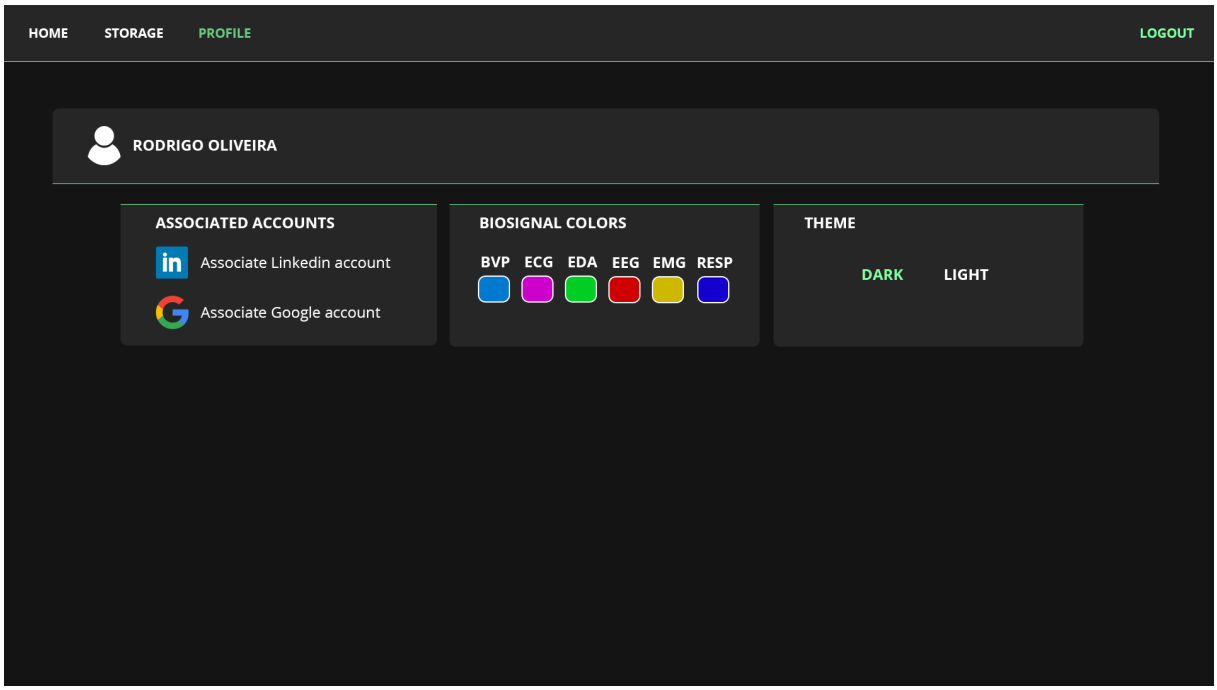

Figure 4.3: Profile page prototype for this web-application.

While it is not an exclusive part of the profile page itself, it is worth mentioning that there is a horizontal navigation bar on top of every page where the user should be able to navigate through all the three different pages and end his session if he desires.

The storage page, shown in Fig. 4.4, works as a dashboard for the user while using this platform. It is divided into three different blocks: a block for biosignals data; one block for snapshots of biosignals; and one block for groups that the user is part of. The user is capable of navigating through different entities in any block by scrolling while applying filters to facilitate the search for specific characteristics, such as being an ECG or being part of a determined group.

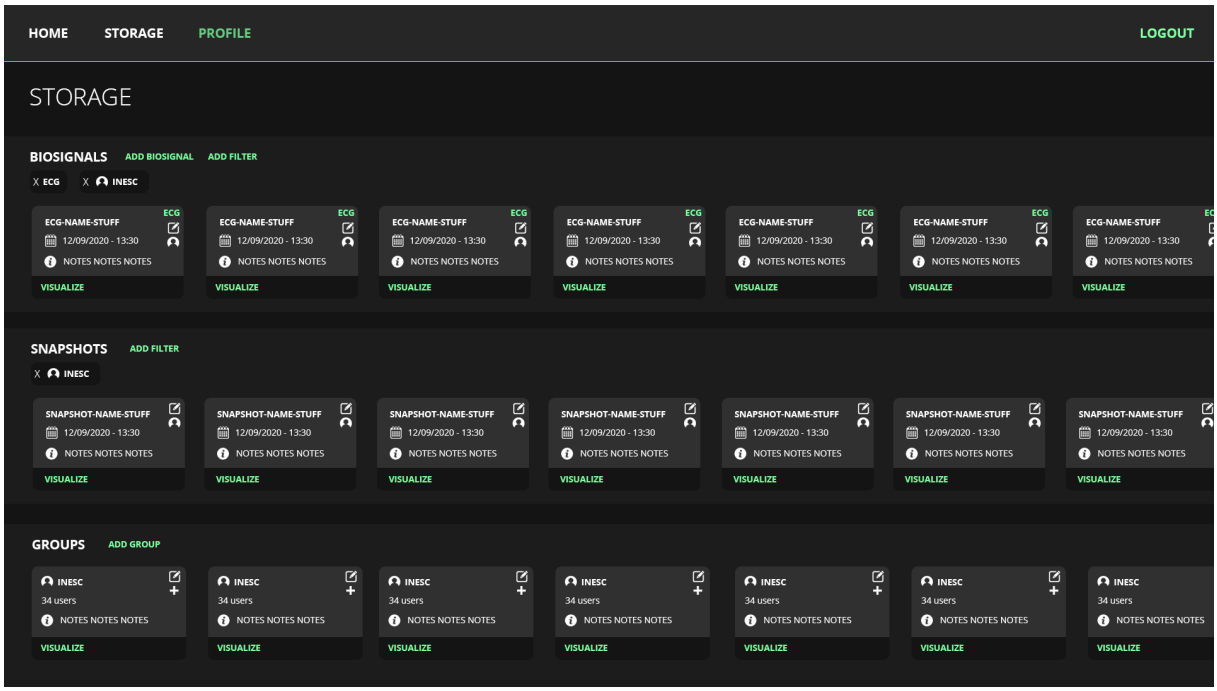

Figure 4.4: Storage page prototype for this web-application.

The third page, shown in Fig. 4.5 was called as the home page of this platform and it contains biosignals that the user tagged as favorites to better visualize them individually. It is divided into as many blocks as the total number of biosignals tagged as favorites. Each block contains the metadata for each biosignal on top of it, buttons to create annotations, share the biosignal or hide it and a visualization for the biosignal data itself where the user is capable of applying zoom-in, zoom-out and panning both vertically or horizontally. It should be also possible to visualize the annotations within each biosignal, their description and time interval.

Finally, the page demonstrated in Fig. 4.6 does not have a specific name but it is described as the page where the user should be able to visualize the acquisition video itself together with its biosignal data. It contains both a visualization of the biosignal data, as described on the home page, and a video for its acquisition. The user should be able to identify and follow the acquisition video throughout the biosignal data visualization by following the vertical red bar that indicates the video's time-frame compared to the biosignal itself.

Views are described as the frontend of this platform, they are responsible for showing data to the user while creating a possible way to interact with controllers. The views are static HTML pages generated by Embedded Ruby, so it was decided to utilize AngularJS together with jQuery to add dynamical elements to these pages and enhance the user's experience. AngularJS is a structural framework for dynamic

Figure 4.5: Home page prototype for this web-application.

web applications[7] and jQuery[8] is a fast, small, and feature-rich JavaScript library. Using both together facilitate the use of Asynchronous JavaScript (Ajax) to communicate with controllers and make HTTP requests.

As mentioned before on Section 4.2, the Prototype A was expected to feature core functionalities, such as an authentication system, biosignals visualization and storing, creating groups of users (teams) and groups of biosignals, and, because of that, only views necessary for this functionalities were develop.

Figure 4.6: Page prototype where the user can visualize the acquisition video itself together with its biosignal data.

[7] AngularJS: https://docs.angularjs.org/guide/introduction (Accessed on October 26, 2020)
[8] jQuery: https://jquery.com/ (Accessed on October 26, 2020)

It was decided to change the page that was once named as ”home page” in the prototype to ”favorites page” since it provides a more explicit idea of its content. The storage page itself was labeled as the new home page of the platform, which means that whenever a user opens the platform that is the first page he is going to see (if he is already authenticated). It was also decided that the user is only capable of utilizing the platform while authenticated.

The favorites page, illustrated on Fig. 4.7, only shows biosignals tagged as favorites together with their metadata and a button to remove it from the favorites. It is also possible to visualize each biosignal individual page by clicking on its name. Data-Driven Documents (D3) was used for plotting since it enables direct inspection and manipulation of a native representation on HTML pages: the Document Object Model (DOM) [57]. On each plot it is possible to observe the filtered signal and vertical bars with different colors that represent the features extracted from the raw signal.

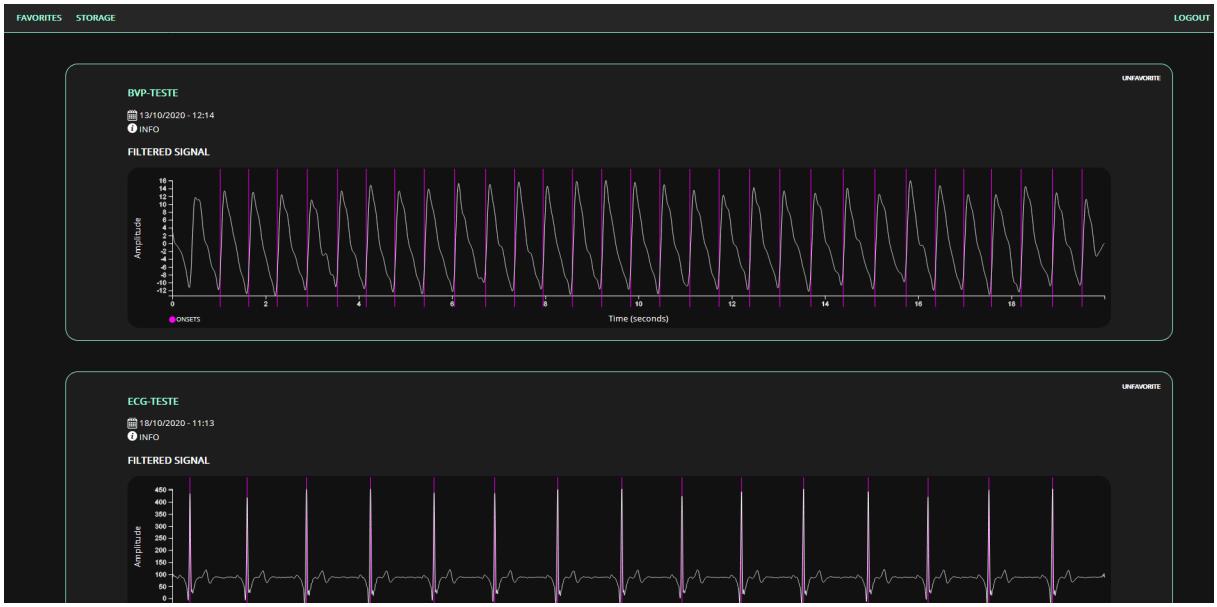

Figure 4.7: The home page from the prototype was renamed to Favorites on Prototype A.

The other developed page for this release was the storage page, shown in Fig. 4.8, very similar to the one presented on the prototype but instead of having a block for snapshots of biosignals, the page has a block for groups of biosignals. During Prototype A, it was only possible to add both biosignals and groups of biosignals, and also share them with teams (groups during Prototype A release).

Both aforementioned pages contain asynchronous loading, which means that entities are loaded separately from the rest of the page via Ajax calls using HTTP requests to the controllers which will then interact with MongoDB to retrieve these entities. By following this approach it was expected to obtain a better performance overall since asynchronous loading does not block page render and allows the browser to continue processing the page while retrieving other entities at the same time.

As shown in Fig. 4.9, a individual block for a biosignal contains its name, date, notes, signal type and icons that allow the user to perform different actions such as adding a biosignal to favorites, removing biosignals from the platform and sharing them with different teams. In all three blocks of biosignals, groups of biosignals and teams, each individual block chains with each other to form a horizontal scrollable block that can be moved by using the mouse wheel.

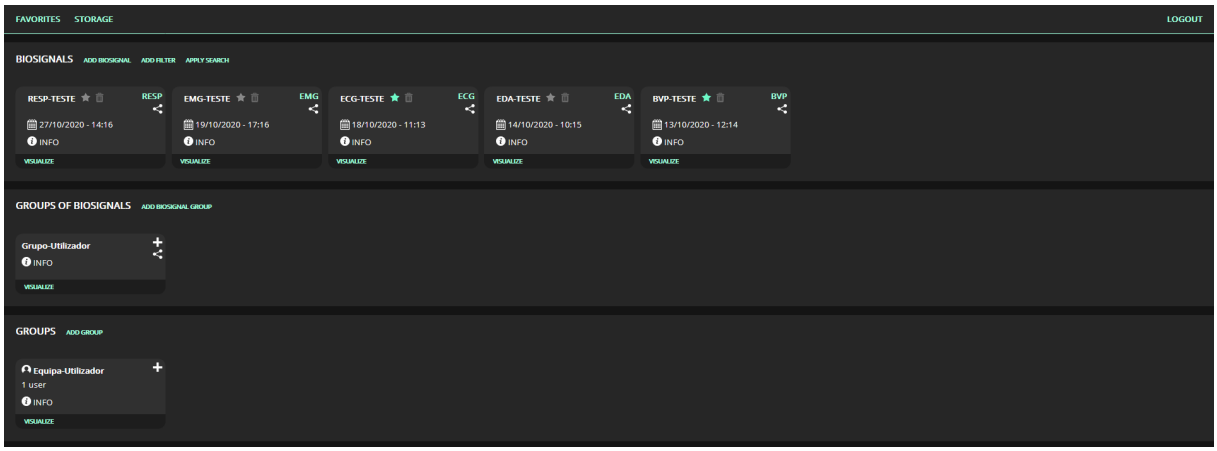

Figure 4.8: Storage page on Prototype A for this web-application.

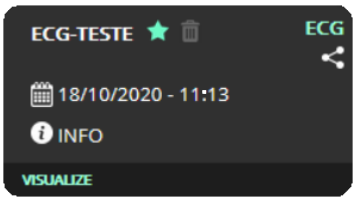

Figure 4.9: Individual block for every biosignal in the Storage page on Prototype A.

Finally, as an advantage of following the MVC pattern it is possible to present the same document, a Biosignal, as many different ways as many signal types there are. Which means, that based on the Biosignal field that identify its signal type, it is possible to render its information differently. This comes as great help for this platform since every biosignal type, such as EEG, ECG and Respiration, produces a different response from its BioSPPy toolkit processing. The dynamic field on the Biosignal document which contains the processed data from BioSPPy will then be presented accordingly to its signal type on different pages. As shown in Fig. 4.10, every signal type page contains its raw and filtered signal, and based on its type new plots are generated, such as the heart rate plot for a ECG.

Figure 4.10: Individual page for a ECG on Prototype A.

4.2.3 Controllers

Controllers are a core component of this platform backend as it acts as a coordinator between users, views and documents. Each controller is created to coordinate specific actions and does not interact with different controllers, but the same controller can interact with different views and models. On this release, there are a total of five controllers, one for each page (favorites and storage) and one for each entity (biosignals, groups of biosignals and teams). The goal of this subsection is to explain certain processes and why they were made in the first place.

As mentioned before, the controllers are responsible for interacting with the BioSPPy toolkit itself. This is only possible by creating a child process that executes a python script and returns its value to the RoR controller. Open3 [9] is a Ruby module that grants access to the stdin, stdout, stderr and a thread to wait for the child process when running another program. Whenever the user submits the form with a file containing the raw signal to create a biosignal in the platform, both the raw signal file path and the signal type specified by the user are sent from the BiosignalsController to a child process that runs a python script where it loads the accordingly file and process its information. After processing its information, the result from BioSPPy is converted to a JSON object and wrote to the same file replacing its old content completely. After the child process is done, the controller reads the file that got its content replaced to obtain the processed data from BioSPPy toolbox.

[9] Open3: https://ruby-doc.org/stdlib-2.6.1/libdoc/open3/rdoc/Open3.html

All of the controllers follow a specific simple protocol whenever responding to HTTP requests from the views. The controller will always reply with a JSON message with a field called *result* to identify if the action was actually performed or not. For example, whenever a user tries to add a Biosignal document to the favorites, the BiosignalsController will perform an action where it will try to add its specific *ObjectId* to the array field containing other biosignals tagged as favorites. If it fails for some reason, the Biosignal does not exist or if there is a database error, the controller will reply to the view with a JSON message containing the field *result* and the value *false*, otherwise it will reply with a JSON message where this field is set as *true*.

Whenever the user opens the storage page, a HTTP request is sent from the view to the BiosignalsController asking for biosignals that the user has access to. This process is made as it follows: first, a hash containing biosignals that belong to the user directly is created; next, the controller iterates over every team that the user is part of, adds biosignals that were previously shared with these teams to the hash and tags them with the team's name on a specific field, the biosignals added are neither within the initial created hash nor were added to the hash already, this guarantees that only one instance of the same biosignal was added to hash; finally, the controller iterates over the *ObjectIds* from the favorites array field from the user and tags every biosignal within the created hash that matches any *ObjectIds* from it as a favorite. The controller then replies with a JSON message containing this hash where each biosignal have additional fields identifying if they came from teams or if they are tagged as favorites.

Since many fields of different documents are array fields containing *ObjectIds* and not the documents themselves, we develop a way to remove *ObjectIds* from documents that no longer exist from these arrays, since they could cause database errors if a controller tries to iterate over a document that does not exist. For example, whenever a user deletes a Biosignal document, the controller does nothing but removing it from the database itself, assuming that it will be eventually deleted from every other field that refers to it based on a eventual consistency approach. This means that every action made by the controllers that need to iterate over *ObjectIds* will first check if the actual document exists. If the document exists, the controller adds the *ObjectId* to a temporary array with valid documents, while if it does not exist it will change a flag value to *true* informing the controller that it needs to update this array field with the new temporary array that is being created while iterating over the documents. This update is only made if such flag is set as *true* and after iterating over every document on the array, guaranteeing that the update is only made once and if needed.

4.3 Converging Workshop

The proposed co-creation process on this work follows a double diamond[10] design process model, where the co-creation process started with divergent thinking, described on Section 4.1 with the Ideation Workshop, followed by convergent thinking where *co-designers* discussed about Prototype A's features and design during the Converging Workshop.

To instigate the aforementioned discussion, the *co-designers* that were part of this workshop were instructed to perform a set of tasks, described below, through another set of semi-structured interviews while using Prototype A:

T1 Perform *Login* with a specific account.

T2 Add a biosignal to the platform.

T3 Visualize the biosignal's page.

T4 Create a team of users.

T5 Share the created biosignal with the team.

T6 Create a group of biosignals, add the biosignal to it and share it with the team.

T7 Visualize the page corresponding to the group of biosignals.

T8 Add the biosignal to favorites.

T9 Delete the biosignal from the platform.

Three *co-designers* were part of this workshop, one of which, CD1, participated on the Ideation Workshop and two new participants, the third *co-designer* (CD3) and the fourth *co-designer* (CD4). CD3 was completely passive when going through the tasks even when encouraged to talk more about his experience and, as a result, his only valuable feedback was related to the text font used, which he described as "not appropriate to achieve user interaction" even though he admitted that the platform was interesting. Both CD1 and CD4 were a lot more talkative and their feedback was mainly related to the prototype's user interface. They mentioned how there was a lot of blank and empty space in different pages, bringing special focus to the storage page. It was also suggested that it should be easier to differentiate biosignals that are marked as favorites from non-favorites. They also mentioned that it could be useful if the user could submit more than one signal at the same time, or if the user could submit an entire folder that would create a group of biosignals automatically with the submitted biosignals. Even though the later suggestion is very reasonable and has its value, uploading dozens

[10] Design Council: The 'double diamond' process model https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond

of large files that could have been acquired from hours of monitoring would cause several impact on the platform's performance with much needed work to improve it in order to create a pleasant user experience, thus, this should be explored as future work.

4.4 Prototype B

The Prototype B is the final prototype developed for this work and because of this it should have all of the features that were mostly classified as relevant by the *co-designers* during the Ideation Workshop. Based on this, Prototype B was developed with the following features in mind:

F1 Create a personal user with credentials.

F3 Create groups of users to share information.

F4 Use the platform API with and without internet connection.

F5 Multichannel visualization.

F6 Create annotations for a biosignal.

F7 Adjust the biosignal visualization timescale, zoom-in and zoom-out.

F8 Create groups for biosignals.

F9 Customize color scheme for different types of biosignals.

F12 Being able to play a video while visualizing a biosignal.

Features **F2**, **F10** and **F11** were not classified as essential for this work outcome and thus are to be considered as potential future work. **F5** was also modified to address only Multichannel visualization based on the feedback provided by CD1 and CD2. A ***CarrierWave***[11] (Ruby Gem) variant[12] was introduced to address **F12**, it provides a simple and extremely flexible way to upload files from Ruby applications and supports *Mongoid*.

Devise is built on top of a Ruby Gem called Warden[13], that provides a mechanism for authentication in Rack based Ruby applications. It uses the concept of cascading strategies to determine if a request should be authenticated. Warden will try every strategy until one succeeds, fails, or if none are found applicable for the particular request. Based on this, an API-Token Strategy to demonstrate a simple implementation of an API (**F4**) was developed. The strategy verifies if the request is valid by checking if a token was sent with the request, it then tries to find a corresponding user for the token sent in the *Authorization* header. If it succeeds, the platform will execute the corresponding controller action.

[11] CarrierWave: https://github.com/carrierwaveuploader/carrierwave
[12] CarrierWave-Mongoid: https://github.com/carrierwaveuploader/carrierwave-mongoid
[13] Warden: https://github.com/wardencommunity/warden

4.4.1 Models

An Annotation document was created (**F6**) and it contains two different fields to determine its minimum and maximum values on the time axis in seconds, a string field for its description, a string field based on HTML hexadecimal values to describe its color and a integer field called *innerIdx* to match annotations with the biosignal's sub-plots, where 0 always stands for an annotation on the raw signal and 1 stands for an annotation on the filtered signal. The Annotation documents are embedded in a Biosignal document, which means that they do not exist without a corresponding Biosignal document.

The User document now contains new string fields based on HTML hexadecimal values that describe how a user customized the color scheme for different types of biosignals (**F9**), which are six different fields for each signal type supported by the BioSPPy toolkit and another six different fields for features extracted from the signals using the toolkit. A string field to store the user's personal API token (**F4**) was added to the document and its value is a Universally Unique Identifier (UUID) randomly generated when the document is created.

Based on the feedback provided by CD1, a BiosignalGroup document now contains three different fields that describe an acquisition video (**F12**) for the biosignals within the group. Which are two different fields for its minimum and maximum values on the time axis in seconds and a string field that describes the uploaded file from *CarrierWave*.

4.4.2 Views

The profile page was now developed for Prototype B, shown in Fig. 4.11, and contains basic information about the user, such as email, personal api-token and customized color scheme. Here the user can change the color scheme for different biosignals by clicking on the different colors that he wants to customize.

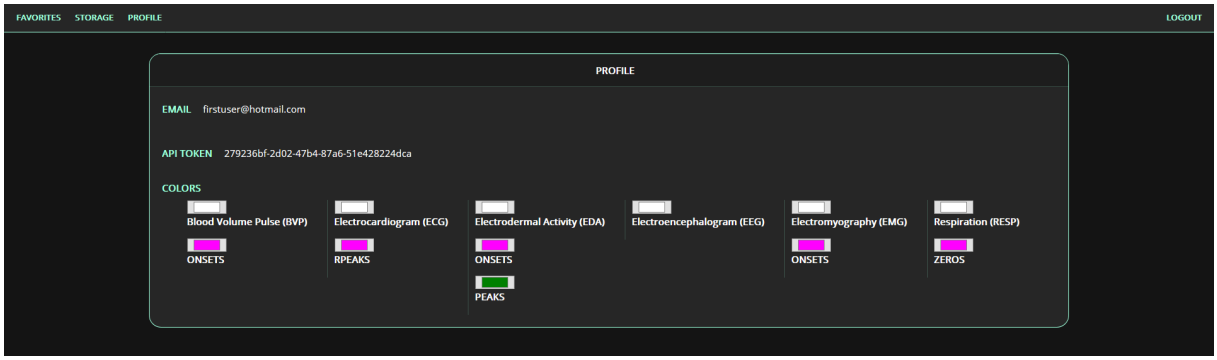

Figure 4.11: Profile page for Prototype B.

The storage page did not change much when compared to Prototype A, as shown in Fig. 4.12. Now the block containing biosignals is divided by favorites (upper block) and non-favorites (lower block). All of the *co-designers* mentioned how scrolling through a large number of biosignals could be bad and time-consuming, thus a search bar to search biosignals by name was added. A simple signal type filter was also added to demonstrated the usage of filters.

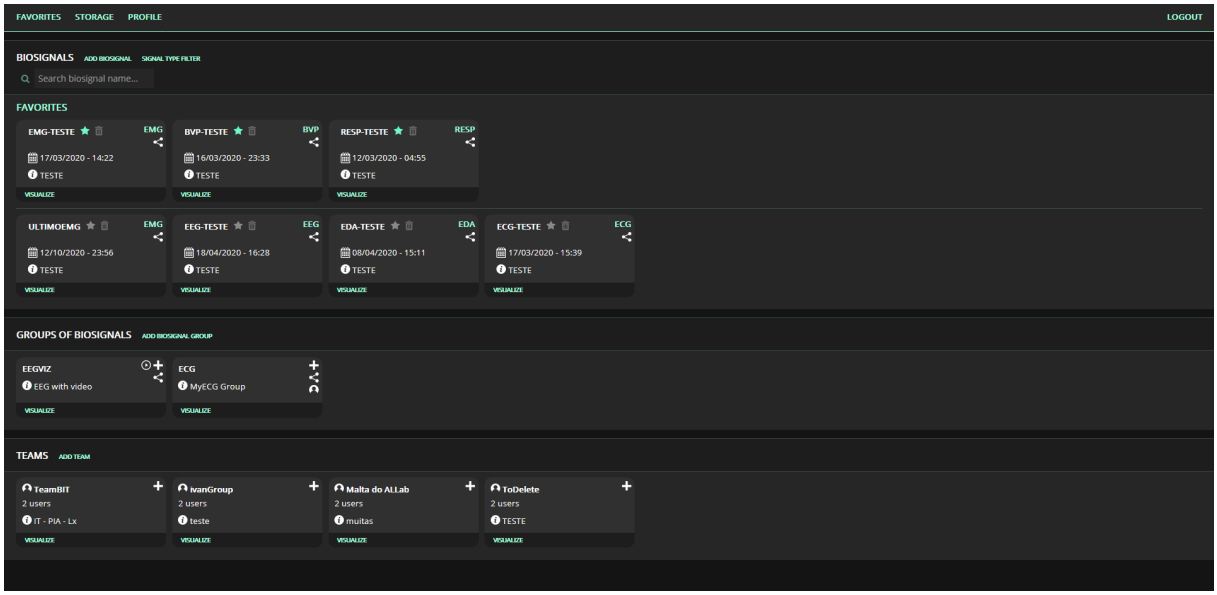

Figure 4.12: Storage page for Prototype B.

As mentioned on Section 4.4.1, Annotation documents have a field called *innerIdx* and its usage can be better visualized at Fig. 4.13. Each signal plot has the option to create annotations since a annotation could refer to a specific event on any signal, which means that whenever a user creates an annotation the platform automatically attributes its document the corresponding *innerIdx* field.

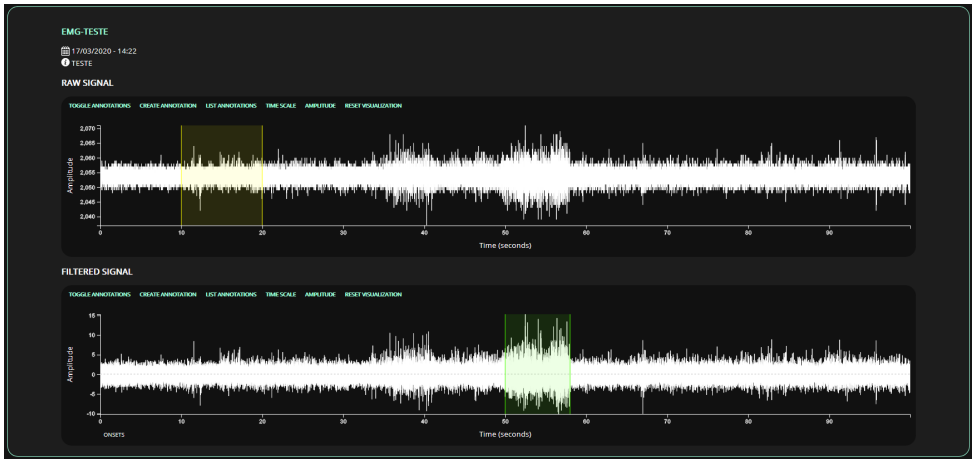

Figure 4.13: Individual Electromyography page for Prototype B.

Prototype B offers a way to visualize overlapping annotations as shown in Fig. 4.14 by mouse hover. Every page that presents a plot for biosignals will sort the annotations array field by its total time-interval, which is the equivalent of sorting the array by the widths of its annotation blocks. This ensures that the smallest overlapping annotations will always be visible since they will be rendered on top of the others. Whenever the mouse cursor moves within a signal plot, the platform computes the bounding rectangles of its annotations and verifies if the cursor position collides with these bounding rectangles, adding them to an array ordered by their starting positions on the time axis that is then shown to the user.

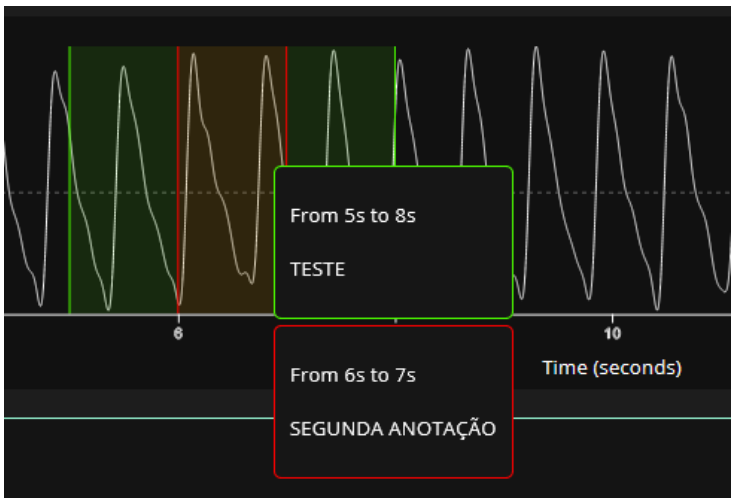

Figure 4.14: Overlapping annotations demonstration on Prototype B.

Pages for groups of biosignals, shown in Fig. 4.15, now also have a block for an acquisition video (**F12**) that can be submitted while creating the group of biosignals. The user can change its acquisition interval through a slider and its value is used to match the video's current timestamp with the actual biosignal plot, which means that the vertical red bar on each plot will take into account the video's current timestamp and subtract it by the minimum value on the slider whenever moving in the plot.

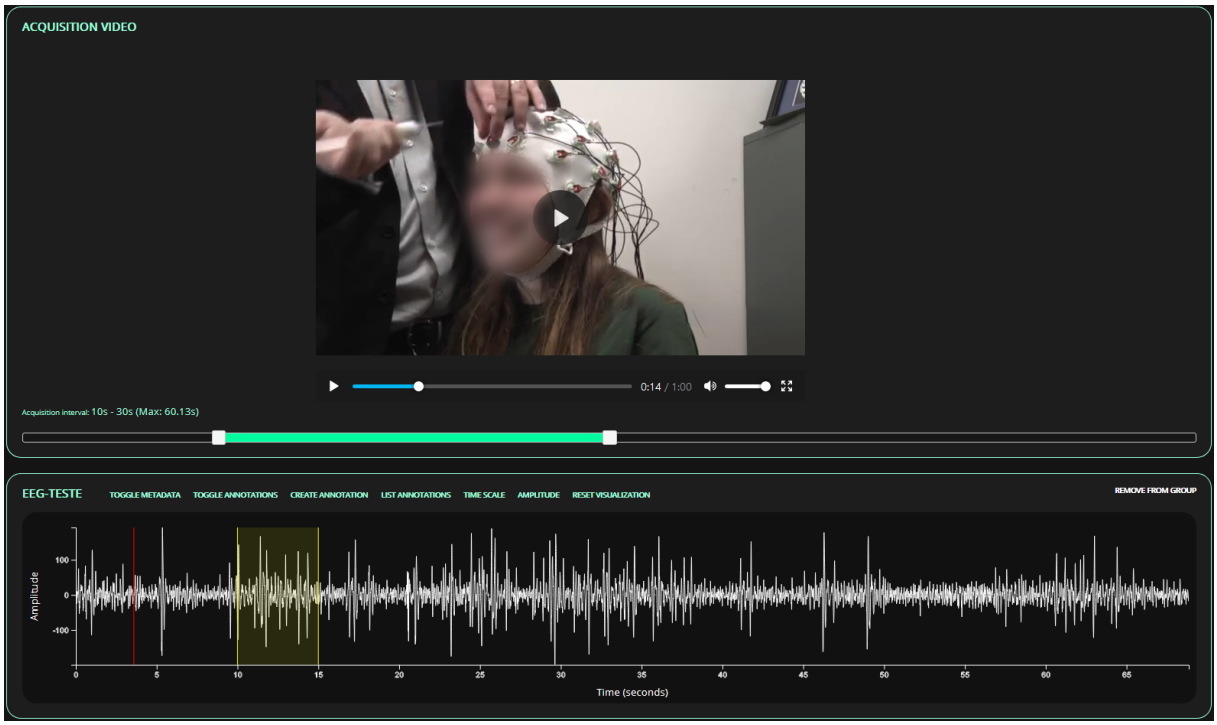

Figure 4.15: Page for a group of biosignals with an acquisition video for Prototype B.

Finally the favorites page, illustrated on Fig. 4.16, did not change much in terms of design, the main difference being that the plots are more compacted and its metadata is hidden by default in order to facilitate the visualization and comparison of multiple biosignals. In terms of features now it is possible to interact with any plot on any page, which means that this does not apply only to the favorites page. A user can pan the graph by dragging the plot viewing area around and zoom-in or zoom-out by using the mouse-wheel. The user can also change plot's axis one by one or change them all at once to the same value. It is also possible to reset every visualization to its default view before any interaction was made. If the user wants to hide all features extracted from the signals (vertical bars) he can do so by clicking on the features' circles below the plot. Both metadata and annotations can be hidden or shown too.

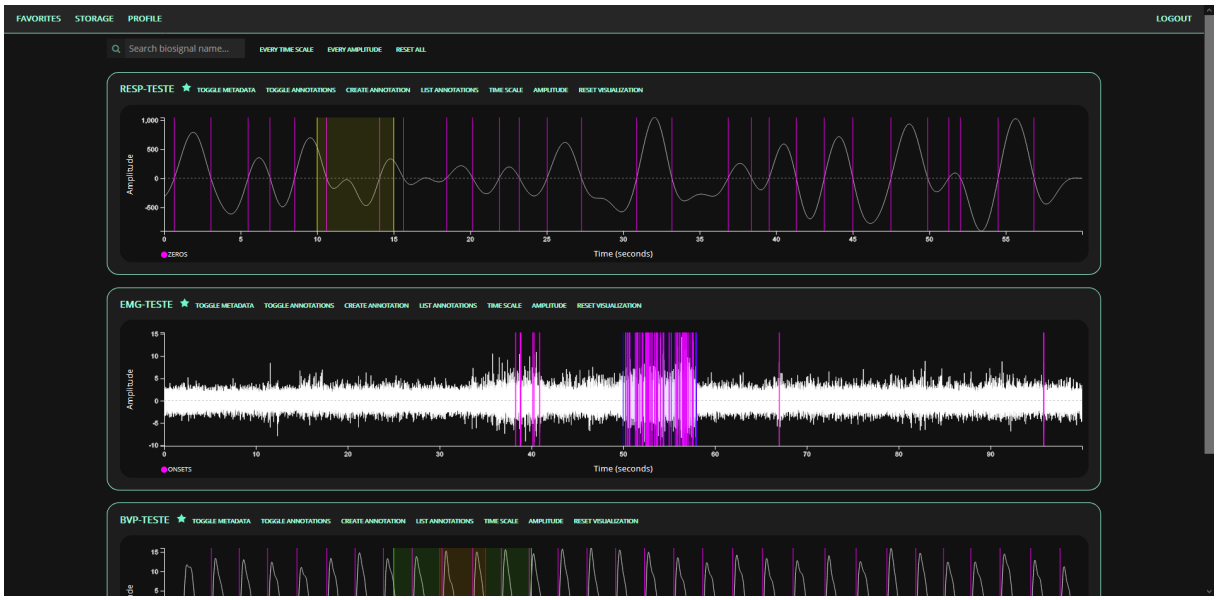

Figure 4.16: Favorites page for Prototype B.

4.4.3 Controllers

A new controller was created for the profile page called ProfileController. It has only three possible actions: rendering the profile page itself, checking for color related fields for users whenever updating the colors and finally updating the colors themselves while following the acknowledgement protocol specified on Section 4.2.3.

The BiosignalsController is now capable of performing advanced searches, which are used whenever looking for biosignals with specific filters. It starts by collecting all of the biosignals that the user has access to, following the procedure specified on Section 4.2.3. Next, the controller verifies if there are any parameters from the HTTP request that are filters, and if so, the specified value is added to a hash where the key is the corresponding document field's name and its value is the specified value from the HTTP request. It then iterates over the biosignals selecting the ones that have the same value as the filter for a specific document field's name using a filter function, illustrated on Fig. 4.17.

For better modularity a new controller called ApiController was created and its goal is to contain as many actions as the API is able to perform. Since the API was only developed to demonstrated how it was possible to create a simple API that could perform different actions, mainly related to obtaining information, the controller only has two possible actions: retrieving a JSON object about the user associated with the token used, illustrated on Fig. 4.18 and Fig. 4.19, and retrieving a JSON object about a biosignal with a specific *ObjectId*.

```ruby
def filterBiosignals(filters, biosignal)
    if(filters.length > 0)
        filters.each do |f|
            if(biosignal[f[0]] != f[1])
                return false
            end
        end
    end
    return true
end
```

Figure 4.17: The Ruby code above is the filter function that determines if a biosignal has fields with specific values, where *f[0]* is the actual field and *f[1]* is the testing value.

```
curl http://localhost:3000/api/user -H 'Authorization: Bearer
↪  279236bf-2d02-47b4-87a6-51e428224dca'
```

Figure 4.18: The curl request above tries to retrieve information about the user with the specified token.

```
{
    "id": {"$oid":"5e55e618c64c400397082c29"},
    "email": "user@domain.com",
    "groups": [ {"$oid":"5eb42456c64c40010ed4fa2c"}, ... ],
    "biosignalgroups": [ {"$oid":"5f74cdb2c64c400168272903"}, ... ],
    "biosignals": [
                    {
                        "id":{"$oid":"5e72fb4dc64c4001435645c4"},
                        "name":"ECG-TEST",
                        ...
                    }
                ],
    "respColor": "#ffffff",
    "respZerosColor": "#ff00ff",
    "edaColor": "#ffffff",
    "edaOnsetColor": "#ff00ff",
    "edaPeaksColor": "#008000",
    "emgColor": "#ffffff",
    "emgOnsetColor": "#ff00ff",
    "bvpColor": "#ffffff",
    "bvpOnsetColor": "#ff00ff",
    "eegColor": "#ffffff",
    "ecgColor": "#ffffff",
    "ecgRPeaksColor": "#ff00ff"
}
```

Figure 4.19: JSON Object reply example from the curl request on Fig. 4.18.

5

Evaluation

Contents

This chapter describes the process of evaluating the developed prototype (Prototype B) mentioned on Chapter 4. It includes the definition of hypotheses, evaluation instruments and protocols that were used to interact with different participants. By relying on both Usability and Utility testing we aimed to measure and understand how participants perceive the usability of our prototype and if our solution is actually useful.

5.1 Usability testing

As previously done by Da Silva et al. [11] when evaluating BioSPPy, this work outcome was evaluated using SUS in order to collect qualitative data assessed by means of individual interviews with users to determine how users perceive the usability of Prototype B [35].

To better understand whether using the developed biosignal web-based application is beneficial for potential users, this work was also evaluated using the Technology Acceptance Model (TAM) proposed by Davis et al. [58]. TAM is used to predict and explain use by focusing on two theoretical constructs, Perceived Usefulness (PU) and Perceived Ease Of Use (PEOU), which are theorized to be fundamental determinants of system use [58]. PU is defined as the degree to which a person believes that using a particular system would enhance his or her job performance, and PEOU is defined as the degree to which a person believes that using a particular system would be free of effort [58]. Both PU and PEOU were measured with a 7-point Likert Scale where higher scores indicate better results.

Finally, Prototype B was also evaluated with NASA-Task Load Index (NASA-TLX) to estimate the overall workload experienced by the participants while performing different tasks [59]. In this work we used RAW-TLX (RTLX), a common modification made to NASA-TLX, that eliminates the weighting process between the sub-scales which simplifies the entire evaluation method [59] and still allows for a high experimental validity [60]. Each RTLX sub-scale was measured with a 20-point Likert Scale.

5.1.1 Participants

A total of twenty-seven participants (sixteen male, eleven female) evaluated Prototype B after performing different tasks, described later on Section 5.1.4. With this sample size we can detect up to 99% problems while being able to have statistically relevant results to analyze [61]. Out of these twenty-seven participants, twenty-two were Computer Science (CS) students and the other five were Biomedical engineering (BME) students. The participants ranged in age from twenty-one to twenty-seven ($M = 22.96;\ SD = 1.37$).

5.1.2 Apparatus

Throughout the evaluation phase, two computers with both mouse and keyboard were needed to perform the task list mentioned on Section 5.1.4. One of the computers was used by a host that could run the server for Prototype B while the other was the participant's computer itself where he could interact with Prototype B via remote control using a video-conference software (e.g. Zoom[1]).

The host needed to have a stopwatch to measure the amount of time needed by the participant to finish a task and also run a screen recording software (e.g. Open Broadcaster Software[2]) while the participant was interacting with its computer.

The questionnaire used was a Google Form where each section had a specific theme: the first section, where the participant would specify his age and gender; the second section that contained the perceived usability questionnaire for SUS; the third section that contained four different questions for both PU and PEOU; and the final section, where the user would classify his experienced Cognitive Workload with six different questions for each RTLX sub-scale.

5.1.3 Procedure

Due to norms of social distancing in the face of the Coronavirus pandemic, the user testing sessions took place on Zoom. The host would start and run Prototype B on his own machine while the participant would interact with the prototype via remote control. The database was always reset before each session to a specific state.

The user testing session starts with the reading of a brief summary of this work to enlighten the participant about our goals with the session. The participant needs to give permission to record the whole conversation before continuing. Afterwards, the participant is encouraged to verbalize his thoughts, feelings and opinions while performing the tasks (Think aloud method [62]).

Before performing any task, the participants had two minutes to interact with the Prototype B by themselves without any help from the host. The participants had only one restriction, they could neither create nor delete any entities (e.g. biosignals, teams, groups of biosignals). Each participant was then asked to perform a randomly sorted list of tasks without any time restriction. Afterwards, the participants were asked to fill a questionnaire containing questions related to SUS, TAM and RTLX.

[1] Zoom: https://zoom.us/
[2] Open Broadcaster Software: https://obsproject.com/

5.1.4 Tasks

The following list of task was conceived in order to explore different features of Prototype B:

T1 Login.

T2 Create a biosignal and an annotation on its filtered signal.

T3 Create a biosignal, create a group of biosignals with an acquisition video, add the aforementioned biosignal to the group and change its acquisition interval.

T4 Create a team, add a user to the team, filter the biosignals by respiratory signals, share a specific biosignal with the aforementioned team, logout and login with the added user to verify if he has access to the shared biosignal.

T5 Search for a specific biosignal, add it to favorites, open the profile page, change the color of a specific signal type and its extracted feature, open the favorites page, verify if the aforementioned biosignal is there with different colors and remove the biosignal from favorites.

T6 Open the favorites page, change the timescale for every biosignal, identify a specific biosignal, show its metadata, describe the biosignal's annotations, delete an annotation, hide its extracted feature and finally hide its annotations.

T7 Search for a specific biosignal and delete it.

5.1.5 Hypotheses

Different hypotheses were formulated based on expected results for this work's goals and acceptable time needed to finish core tasks. Both **T1** and **T7** were relatively simple since they did not demand much from the participants when compared to other tasks, only login and deleting a biosignal. Based on this, these were the hypotheses formulated for this work:

H1 Participants should take less then 125 seconds to finish **T2**.

H2 Participants should take less then 210 seconds to finish **T3**.

H3 Participants should take less then 105 seconds to finish **T4**.

H4 Participants should take less then 90 seconds to finish **T5**.

H5 Participants should take less then 150 seconds to finish **T6**.

H6 Prototype B should have a SUS score above 70.

H7 Prototype B should have both PU and PEOU scores above 5.

H8 Participants should experience a Cognitive Workload smaller than 5.

5.1.6 Results

The time needed to concluded each task is illustrated on Fig 5.1. As to be expected, both **T1** and **T7** did not demand a lot of time of the participants based on their simplicity. All of the other tasks will be further discussed with their respective hypotheses as they were considered to be core tasks and presented more interesting results.

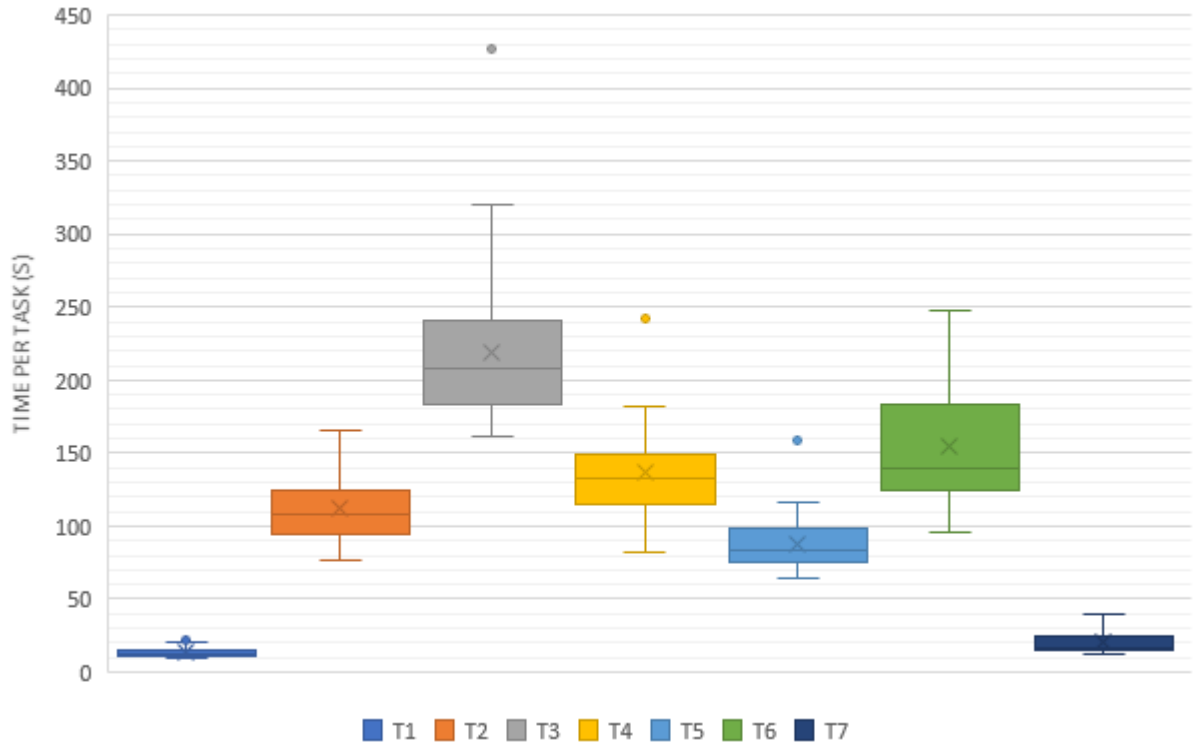

Figure 5.1: Boxplot for the time needed to concluded each task.

H1 Participants should take less then 125 seconds to finish **T2**.

Even though **T2** may look as a simple task it also contains core components of this work, which are the ability of adding a biosignal to the platform and also adding annotations to any sub-plot within the processed biosignal. Overall, the participants did not find any difficulty to perform this task, some were simply slower or faster than the others which can be confirmed by its results ($M = 111.85;\ SD = 21.76$).

At 95% confidence, the true mean could lie between 104 seconds to 120 seconds. As 125 seconds is out of this range, we can successfully accept **H1**, which means that the participants should, in fact, take less time then predicted to finish **T2**.

H2 Participants should take less then 210 seconds to finish **T3**.

During the user sessions, **T3** was by far the task that presented the most diverse results ($M = 218.89;\ SD = 57.30$), shown by its standard deviation, as some participants were able to finish it very quickly and others needed a lot more time to figure out how to perform one of the steps of this task. The step that caused more problems was the one were participants were asked to change the acquisition interval for the group of biosignals, and it was due to the fact that they were not able to recognize at first that there was a slider where you could change its value.

At 95% confidence, the true mean could lie between 197 seconds to 240 seconds. As 210 seconds is in this range, we fail to reject **H2**, thus **H2** is inconclusive.

H3 Participants should take less then 105 seconds to finish **T4**.

While not expected, **T4** appears as the third most time-consuming task ($M = 136.15;\ SD = 31.72$). This was due to the fact that a good part of the participants did not correctly understand how to apply a filter to respiratory signals. They mentioned that it was not difficult to apply the filter itself but the major problem was that they were looking at the team's section and not at the biosignals' sections on Prototype B's user interface.

At 95% confidence, the true mean could lie between 124 seconds to 148 seconds. As 105 seconds is below this range, we refute **H3**, and in fact, participants take more time then predicted to finish **T4**.

H4 Participants should take less then 90 seconds to finish **T5**.

T5 was one of the tasks that produced less problems for the participants ($M = 87.78;\ SD = 19.93$) as they were able to easily go through each step. Only a minority of participants did not understand at first how to change the color for a specific signal type in the profile page.

At 95% confidence, the true mean could lie between 80.3 seconds to 95.3 seconds. As 90 seconds is in this range, we fail to reject **H4**, thus **H4** is inconclusive.

H5 Participants should take less then 150 seconds to finish **T6**.

T6 ($M = 155.11;\ SD = 38.20$) faced the same problem as **T3** where most users had a hard time to progress through a step of the task. This would occur when the participant was asked to hide the biosignal's extracted feature, which are the vertical bars present on the plot. As shown on Fig. 5.2, in order to hide such vertical bars the participants needed to click on the legend from the features themselves. For them, it was not easy to understand that the legend was clickable and thus making it more noticeable or creating a button alongside the others on top of the plot would facilitate this step.

At 95% confidence, the true mean could lie between 141 seconds to 170 seconds. As 150 seconds is in this range, we fail to reject **H5**, thus **H5** is inconclusive.

Figure 5.2: Prototype B's plot for a signal that was added to the favorites.

H6 Prototype B should have a SUS score above 70.

The obtained SUS score for Prototype B ($M = 87.69;\ SD = 8.23;\ 95\%\ CI = [84.6,\ 90.8]$) is illustrated on Fig. 5.3, which could be rated as *Excellent* or as an A using a school grading scale [36], this being the highest possible grade. Since the SUS score is an indicator of how participants perceive the usability of a system, it is easy to conclude that the participants considered Prototype B to be usable. With the aforementioned results it is possible to accept **H6**.

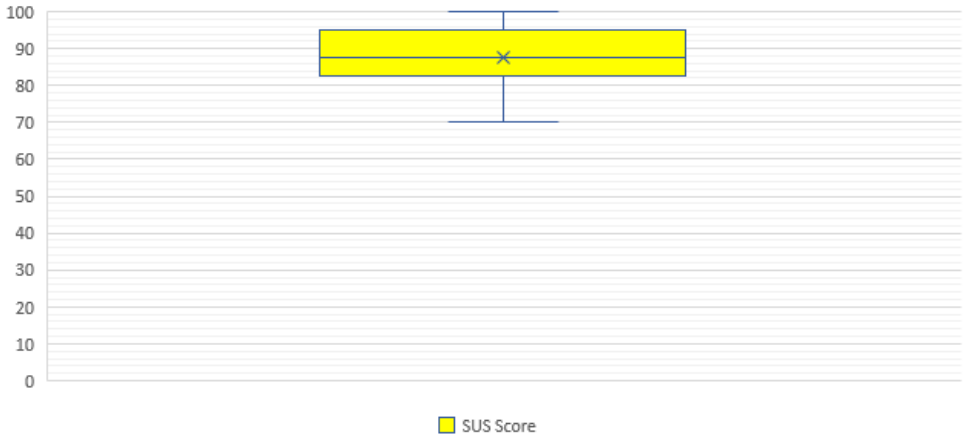

Figure 5.3: Boxplot for the obtained SUS Score.

H7 Prototype B should have both PU and PEOU scores above 5.

Another evaluation method used was TAM and the obtained scores for both Perceived Usefulness ($M = 6.04;\ SD = 0.94;\ 95\%\ CI = [5.69,\ 6.40]$) and Perceived Ease Of Use ($M = 6.21;\ SD = 0.54;\ 95\%\ CI = [6.01,\ 6.42]$) can be better visualized on Fig. 5.4. These results suggest that the consumers perceive Prototype B to be very useful, easy to use and not difficult to understand, learn or operate [58], and because of this we can accept **H7**.

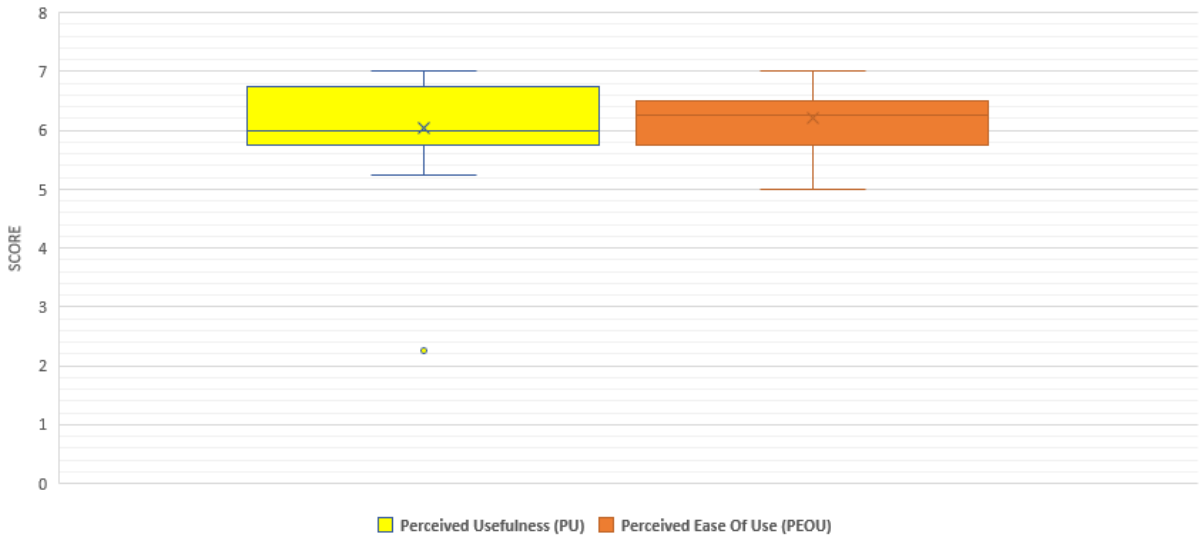

Figure 5.4: Boxplot for the obtained PU and PEOU.

H8 Participants should experience a Cognitive Workload smaller than 5.

Finally, Prototype B was also evaluated with RTLX and the results for each sub-scale are illustrated on Fig. 5.5. The participant's experienced Cognitive Workload (CW) ($M = 3.09;\ SD = 1.49;\ 95\%\ CI = [2.53,\ 3.65]$) can be obtained through an addition of the scores and then division on the six different dimensions to get the average. The higher it is, the higher the experienced cognitive workload [63]. The obtained results means that the overall cognitive loads for the participants were pretty low, thus we can accept **H8**.

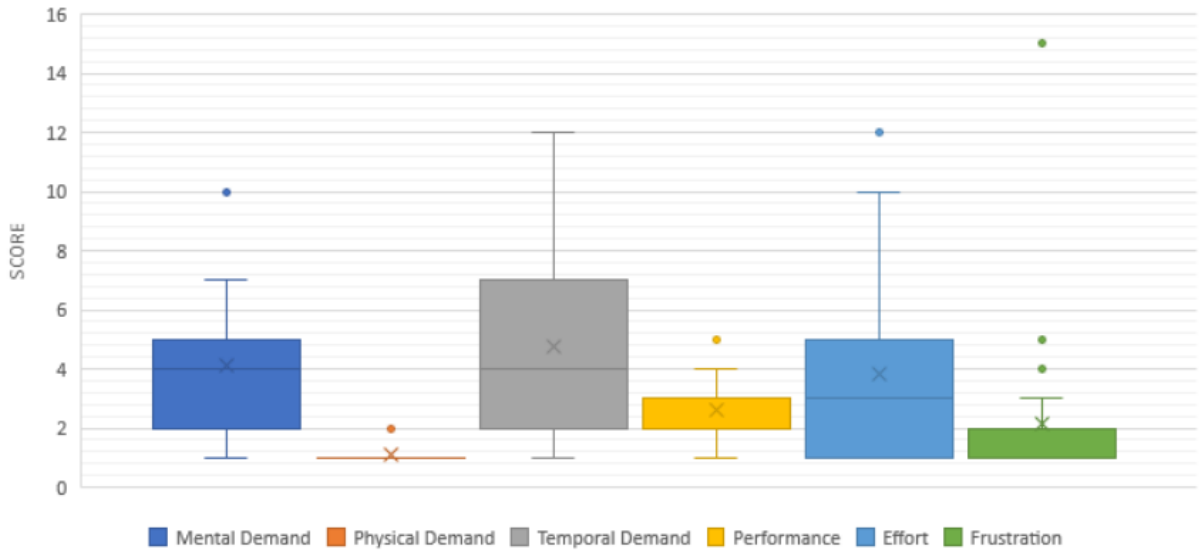

Figure 5.5: Boxplot for the different RTLX sub-scales' scores.

5.2 Utility testing

In order to better understand if Prototype B is useful towards its potential users we invited the first *co-designer* (CD1), a biosignal specialist and researcher from Instituto de Telecomunicações (IT), to a common dialogue where he could interact and test Prototype B, following the think aloud method [62]. The dialogue was conducted in a way where it was possible to cover and talk about all the features mentioned on Section 4.1, going through all the different pages available on Prototype B.

Starting with the storage page, CD1 talked about the interface in general and highlighted how the page looks like a dashboard that is really easy to work as you get more used to it. He also mentioned that both the Search Bar and the ability to filter by signal types were very good ideas to identify biosignals faster, while suggesting that it should be possible to add more filtering options in the future. As mentioned during the Converging Workshop on Section 4.3, CD1 reinforced that it could be useful if the user could submit more than one signal at the same time. Another feature that CD1 suggested was the ability to drag and drop biosignals to groups or teams in order to add or share those biosignals with such entities.

Regarding the page for a group of biosignals where it was possible to visualize an acquisition video together with biosignals' plots, CD1 mentioned that while he believes that the feature is very well-conceived and the ability to change the acquisition interval was a great idea, there should be a way to visualize the actual video timeline itself together with the biosignal's plots and not only follow the video by the red bar introduced on all plots.

CD1 was very pleased to see how interactive the interface was in general, especially where you could interact with plots by dragging, panning and applying zoom. CD1 mentioned that the annotations' related feature was also very well-conceived since it was possible to visualize overlapping annotations, but he suggested that it should be possible to create annotations by using a shortcut with the user's mouse or keyboard, for example, if the user clicks on a specific timestamp and holds it while dragging the cursor to another timestamp it should automatically create an annotation after releasing the click.

Overall, CD1 made it clear that Prototype B was very well-conceived and all of its features were very useful whenever visualizing and working with biosignals. He mentioned that the platform has a lot of potential and was glad to be part of design process.

5.3 Discussion

First, as aforementioned on Section 5.1.3, it is of utter importance to mention how operating and leading both usability and utility testing sessions through a video-conference software (Zoom) could have deteriorated the overall results. Every single session had a moment where the participant said that he miss-clicked because of the video-conference software that was not completely accurate. Based on this, measuring any other values such as miss-clicks was not reliable. Thus, only time needed to finish tasks was measured and it is not possible to conclude if the video-conference software introduced significant noise or not.

It is important to mention how following a co-creation design methodology may have impacted the obtained results. Hence, interacting and obtaining continuous feedback from different *co-designers* increased the scores themselves and the chances of obtaining better results for SUS, TAM and RTLX, considering that these metrics are mainly related to User Experience and the ultimate goal of the co-creation process was to create a product that would satisfy potential users both in terms of interface and utility.

Taking all of the formulated hypotheses into consideration, we can conclude that the only task that was surprisingly out of the expected time range to finish was **T4** leading to our only refuted hypothesis **H3**. In total, we had four accepted hypotheses (**H1, H6, H7** and **H8**), three inconclusive hypotheses (**H2, H4, H5**) and only one refuted hypothesis (**H3**).

During the usability testing, the participants never mentioned that they felt pressured to finish a task since there was no time restriction. Results suggest that all of the tasks, with the exception of **T4**, produced expected results. Thus, not having a time restriction was overall beneficial to the participants.

Participants brought into attention two different aspects many times, these being: (a) most of the time, it was hard to identify how to successfully hide the biosignal's extracted feature; and (b) it should be easier to understand how to change the acquisition interval for a group of biosignals. Even though these were very common issues, all of the participants mentioned that it would only happen once since the process itself was very easy and the problem was mainly related to identify how to do perform said steps.

Finally, taking both usability and utility testing results into consideration, it is fair to conclude that this work produced a web application for interacting with biosignals that was perceived to be useful, very easy to use, and that participants did not experience a great workload whenever interacting with Prototype B. Overall, both the focus user and the general participants were very happy and satisfied with the web application where the most common problems faced were related to a first time impression, and thus, using the platform afterwards should be even easier.

5.3.1 Limitations

As aforementioned on Section 4.3, a clear limitation of the produced prototype is that it is not possible to submit more than one biosignal at the same time and large scale biosignals were not taken into consideration. Both Gomes et al. [8] and Cavaco et al. [9] worked with large scale biosignals analysis using sophisticated multi-level visualization techniques into account, and thus, their work should be taken into consideration whenever moving forward with this work.

Based on this work's goals that are mainly related to Human Computer Interaction, the produced platform raw performance was not evaluated, thus, it is not clear to estimate the overall platform capabilities, how much requests it can handle at the same time, how its performance varies over time and how the platform works with a over populated database. Also, obtaining participants with different backgrounds to test our prototype was very hard due to the Coronavirus pandemic, which would be ideal along the lines of Human Computer Interaction.

Based on the sensitive nature related to the data that could be contained in this platform, this work evaluation could be improved with different security tests and protocols to indicate this platform's strengths and weaknesses. One of the main flaws associated with the MongoDB version used is the fact that, currently, there is no data encryption, thus, running this platform without a trustful environment is not ideal and future work integrating the Enterprise version of MongoDB is needed.

Finally, the platform is only able to deal with one-dimensional biosignals and supports the BioSPPy [3] version deployed at the 30th of April of 2019, thus, a parallel observation for the library progress is needed to keep this work up to date with new functionalities that may be introduced into the BioSPPy toolkit itself.

[3] BioSPPy: https://github.com/PIA-Group/BioSPPy

6

Conclusions and Future Work

Contents

This document provides a solution for the lack of biosignals web-based applications taking into account the current state-of-the-art and unexplored technologies, such as Ruby On Rails. It contains an easy-to-use Graphical User Interface and provides a web-based platform capable of visualizing, analyzing, sharing and storing biological signals to non-proficient users in computer programming. The developed application follows a Model-View-Controller architecture and its database was designed under MongoDB, a NoSQL document-oriented database. It interacts with BioSPPy toolkit, a toolbox for biosignal processing written in Python, to process biosignals internally through an interface capable of running Python code on Ruby On Rails.

Such web-based application followed a co-creation design methodology with two workshops for collaborative design while working together with four different specialists. A functional prototype was developed after the first workshop and improved based on the specialists' feedback after the second. Thus, the final prototype took into account important feedback from potential customers and end-users, increasing the chances of long-term adoption.

With all of the aforementioned technologies and methodologies used, it was possible to develop a web-based application capable of: (i) creating users with credentials; (ii) creating groups of users to share information; (iii) visualizing biosignals' plots and interact with them by dragging, panning and applying zoom; (iv) creating annotations for the biosignals; (v) creating groups of biosignals; (vi) customizing the color scheme for different types of biosignals; (vii) reproducing an acquisition video together with biosignal plots; and finally, (viii) interacting with an API to obtain information.

The prototype evaluation relied both in Usability and Utility testing. The usability evaluation used the System Usability Scale, the Technology Acceptance Model and a modification of NASA-Task Load Index. Taking these evaluation metrics into account, it is fair to conclude that such prototype was perceived to be useful, very easy to use, and that participants experienced a small workload whenever interacting with it. As expected after relying on a co-creation design methodology, it is possible to conclude from the utility evaluation that the focus user considered the prototype to be well designed and very useful for their future work.

With this work we contribute to different research areas on a multidisciplinary manner, while facilitating biosignal visualization, sharing and analysis for different levels of expertise that can used by either students who want to learn more about biosignals or by specialists who want to minimize their workload, opening new research horizons and prospects, providing an easy-to-use tool to any biosignal enthusiast.

6.1 Future Work

Future work should take the aforementioned limitations described on Section 5.3.1 into consideration. This application should be able to accept the submission of more than one biosignal at the same time, as an example, it should be possible to submit an entire folder containing biosignal data that would be converted to a group of biosignals automatically.

Long-term monitoring and acquisition of biological signals is a common problem faced nowadays towards cases that include sleep, movement monitoring, control of cardiac problems and falls detection [9]. Based on this, while the developed platform should work with biosignals of any size, it was neither tested nor evaluated. It is possible that in order to obtain better performance results, mainly related to visualization methods for long-term biosignals, previous studied approaches should be implemented [8, 9]. Alongside the performance evaluation, security evaluation should also be made in the future in order to recognize and address possible flaws of this work. In order to assure data encryption with MongoDB, future work should also focus on integrating the functionalities of the Enterprise version of MongoDB.

As some features described on Section 4.1 were discarded since they were not considered to be core concepts for this work, those should be now considered whenever moving forward. Saving biosignal visualization snapshots for future investigation can be done by simply implementing a famous javascript library named *html2canvas*[1] into this work, which allows HTML content to be exported as an image. Changing between dark and light theme should not be difficult as this platform uses AngularJS which introduces possible dynamism into every HTML page. Associating an account from different websites can be done with OmniAuth[2] gem, a library that standardizes multi-provider authentication for web applications.

While this platform only uses BioSPPy was a biosignal processing tool, nothing stops its potential to reach and interact with different biosignal processing tools. Since this platform follows a Model-View-Controller architecture, it should be easy to integrate new processing tools as different views can be created for the same model with different toolboxes, which can be determined by a simple attribute on the biosignal's model.

Finally, even though the results are promising and very positive, it should be noted that the entire evaluation process for the usability testing could be improved by testing participants with different backgrounds and comparing the results by grouping such participants, as previously done by Da Silva et al. [11] when evaluating BioSPPy.

[1] html2canvas: https://github.com/niklasvh/html2canvas
[2] OmniAuth: https://github.com/omniauth/omniauth

Bibliography

[1] V. Rantanen, H. Venesvirta, O. Spakov, J. Verho, A. Vetek, V. Surakka, and J. Lekkala, "Capacitive measurement of facial activity intensity," *IEEE Sensors journal*, vol. 13, no. 11, pp. 4329–4338, 2013.

[2] J. Birjandtalab, D. Cogan, M. B. Pouyan, and M. Nourani, "A non-eeg biosignals dataset for assessment and visualization of neurological status," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2016, pp. 110–114.

[3] H. J. Baek, H. B. Lee, J. S. Kim, J. M. Choi, K. K. Kim, and K. S. Park, "Nonintrusive biological signal monitoring in a car to evaluate a driver's stress and health state," *Telemedicine and e-Health*, vol. 15, no. 2, pp. 182–189, 2009.

[4] A. P. Alves, H. P. da Silva, A. Lourenço, and A. Fred, "A web-based platform for real-time biosignal visualization and recording."

[5] M. Beier, T. Penzel, and D. Krefting, "A performant web-based visualization, assessment and collaboration tool for multidimensional biosignals," *Frontiers in neuroinformatics*, vol. 13, p. 65, 2019.

[6] C. Mata, "Development and evaluation of an open platform for recording, storage, visualization and analysis of biosignals on the cloud," 2018.

[7] A. J. Garza, B. S. Subedi, C. Y. Zhang, and D. H. Lin, "A web-based system for eeg data visualization and analysis," in *Int'l Conf. Health Informatics and Medical Systems (HIMS2015)*, 2015, pp. 119–124.

[8] R. R. B. Gomes, "Long-term biosignals visualization and processing," Ph.D. dissertation, Faculdade de Ciências e Tecnologia, 2011.

[9] C. A. d. Q. S. Cavaco, "New visualization model for large scale biosignals analysis," Ph.D. dissertation, 2014.

[10] M. Committee *et al.*, "Medical waveform description format encoding rules," *MFER Part I, Version*, pp. 1–01.

[11] H. P. da Silva, A. Lourenço, A. Fred, and R. Martins, “Bit: biosignal igniter toolkit,” *Computer methods and programs in biomedicine*, vol. 115, no. 1, pp. 20–32, 2014.

[12] J. Semmlow, “Circuits, signals, and systems for bioengineers,” *Parasitic Element (Electrical networks)*, pp. 134–135, 2005.

[13] T. N. Alotaiby, S. A. Alshebeili, T. Alshawi, I. Ahmad, and F. E. Abd El-Samie, “Eeg seizure detection and prediction algorithms: a survey,” *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 183, 2014.

[14] J. Soni, U. Ansari, D. Sharma, and S. Soni, “Predictive data mining for medical diagnosis: An overview of heart disease prediction,” *International Journal of Computer Applications*, vol. 17, no. 8, pp. 43–48, 2011.

[15] T. Thompson, T. Steffert, T. Ros, J. Leach, and J. Gruzelier, “Eeg applications for sport and performance,” *Methods*, vol. 45, no. 4, pp. 279–288, 2008.

[16] G. R. Müller-Putz, R. Scherer, G. Pfurtscheller, and R. Rupp, “Eeg-based neuroprosthesis control: a step towards clinical practice,” *Neuroscience letters*, vol. 382, no. 1-2, pp. 169–174, 2005.

[17] K. Tanaka, K. Matsunaga, and H. O. Wang, “Electroencephalogram-based control of an electric wheelchair,” *IEEE transactions on robotics*, vol. 21, no. 4, pp. 762–766, 2005.

[18] F. A. C. S. de Melo, “Flow-z: A flow-based adaptable game to maintain optimal challenge,” 2017.

[19] K. M. Gilleade and A. Dix, “Using frustration in the design of adaptive videogames,” in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 2004, pp. 228–232.

[20] K. Gilleade, A. Dix, and J. Allanson, “Affective videogames and modes of affective gaming: assist me, challenge me, emote me,” *DiGRA 2005: Changing Views–Worlds in Play.*, 2005.

[21] K. Poels, W. v. d. Hoogen, W. Ijsselsteijn, and Y. de Kort, “Pleasure to play, arousal to stay: the effect of player emotions on digital game preferences and playing time,” *Cyberpsychology, behavior, and social networking*, vol. 15, no. 1, pp. 1–6, 2012.

[22] J. Muthuswamy, “Biomedical signal analysis,” *Standard handbook of biomedical engineering and design*, pp. 18–1, 2003.

[23] D. Telaar, M. Wand, D. Gehrig, F. Putze, C. Amma, D. Heger, N. T. Vu, M. Erhardt, T. Schlippe, M. Janke *et al.*, “Biokit—real-time decoder for biosignal processing,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[25] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney, "The rwth aachen university open source speech recognition system," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[26] C. Carreiras, A. P. Alves, A. Lourenço, F. Canento, H. Silva, A. Fred *et al.*, "BioSPPy: Biosignal processing in Python," https://github.com/PIA-Group/BioSPPy/, 2015–, (Accessed on December 28, 2019). [Online]. Available: https://github.com/PIA-Group/BioSPPy/

[27] G. Molina León and A. Breiter, "Co-creating visualizations: A first evaluation with social science researchers," 2020.

[28] S. H. Fairclough, "Fundamentals of physiological computing," *Interacting with computers*, vol. 21, no. 1-2, pp. 133–145, 2009.

[29] J. Guerreiro, "A biosignal embedded system for physiological computing," Ph.D. dissertation, Instituto Superior de Engenharia de Lisboa, 2013.

[30] H. P. da Silva, A. Fred, and R. Martins, "Biosignals for everyone," *IEEE Pervasive Computing*, vol. 13, no. 4, pp. 64–71, 2014.

[31] E. Kaniusas, *Biomedical signals and sensors I: Linking physiological phenomena and biosignals*. Springer Science & Business Media, 2012.

[32] A. Schmidt, "Biosignals in human-computer interaction." *interactions*, vol. 23, no. 1, pp. 76–79, 2016.

[33] K. Watanabe, T. Watanabe, H. Watanabe, H. Ando, T. Ishikawa, and K. Kobayashi, "Noninvasive measurement of heartbeat, respiration, snoring and body movements of a subject in bed via a pneumatic method," *IEEE transactions on biomedical engineering*, vol. 52, no. 12, pp. 2100–2107, 2005.

[34] W. Zhang, Y. Du, and M. L. Wang, "Noninvasive glucose monitoring using saliva nano-biosensor," *Sensing and Bio-Sensing Research*, vol. 4, pp. 23–29, 2015.

[35] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[36] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[37] R. J. Anderson, *Security in clinical information systems*. British Medical Association London, 1996.

[38] D. S. McFarland, *JavaScript & jQuery: the missing manual*. " O'Reilly Media, Inc.", 2011.

[39] J. Rosa, "A web-based framework devised using a model-view-controller architecture," Ph.D. dissertation, 2015.

[40] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: open-source software to create and analyze dynamic simulations of movement," *IEEE transactions on biomedical engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[41] C. Carreiras, H. Silva, A. Lourenço, and A. L. Fred, "Storagebit-a metadata-aware, extensible, semantic and hierarchical database for biosignals." in *HEALTHINF*, 2013, pp. 65–74.

[42] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute *et al.*, "Bioportal: ontologies and integrated data resources at the click of a mouse," *Nucleic acids research*, vol. 37, no. suppl_2, pp. W170–W173, 2009.

[43] B. M. Konopka, "Biomedical ontologies—a review," *Biocybernetics and Biomedical Engineering*, vol. 35, no. 2, pp. 75–86, 2015.

[44] A. Varri, B. Kemp, T. Penzel, and A. Schlogl, "Standards for biomedical signal databases," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 33–37, 2001.

[45] G. Hellmann, M. Kuhn, M. Prosch, and M. Spreng, "Extensible biosignal (ebs) file format: simple method for eeg data exchange," *Electroencephalography and clinical neurophysiology*, vol. 99, no. 5, pp. 426–431, 1996.

[46] B. Kemp and J. Olivan, "European data format 'plus'(edf+), an edf alike standard format for the exchange of physiological data," *Clinical Neurophysiology*, vol. 114, no. 9, pp. 1755–1761, 2003.

[47] B. Kemp, A. Värri, A. C. Rosa, K. D. Nielsen, and J. Gade, "A simple format for exchange of digitized polygraphic recordings," *Electroencephalography and clinical neurophysiology*, vol. 82, no. 5, pp. 391–393, 1992.

[48] D. Brooks, "Extensible biosignal metadata a model for physiological time-series data," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2009, pp. 3881–3884.

[49] D. J. Brooks, P. J. Hunter, B. H. Smaill, and M. R. Titchener, "Biosignalml—a meta-model for biosignals," in *2011 annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2011, pp. 5670–5673.

[50] A. Kokkinaki, I. Chouvarda, and N. Maglaveras, "An ontology-based approach facilitating unified querying of biosignals and patient records," in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2008, pp. 2861–2864.

[51] T. H. group, "Hierarchical data format version 5," 2000-2010. [Online]. Available: www.hdfgroup.org/HDF5

[52] A. Lourenço, H. P. Da Silva, C. Carreiras, A. P. Alves, and A. L. Fred, "A web-based platform for biosignal visualization and annotation," *Multimedia Tools and Applications*, vol. 70, no. 1, pp. 433–460, 2014.

[53] C. Bossen, C. Dindler, and O. S. Iversen, "Evaluation in participatory design: a literature survey," in *Proceedings of the 14th Participatory Design Conference: Full papers-Volume 1*, 2016, pp. 151–160.

[54] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[55] J. Rudd and S. Isensee, "Twenty-two tips for a happier, healthier prototype," *interactions*, vol. 1, no. 1, pp. 35–40, 1994.

[56] P. Szekely, "User interface prototyping: Tools and techniques," in *Workshop on Software Engineering and Human-Computer Interaction*. Springer, 1994, pp. 76–92.

[57] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[58] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS quarterly*, pp. 319–340, 1989.

[59] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[60] E. A. Bustamante and R. D. Spain, "Measurement invariance of the nasa tlx," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, no. 19. SAGE Publications Sage CA: Los Angeles, CA, 2008, pp. 1522–1526.

[61] M. J. Fonseca, P. Campos, and D. Gonçalves, "Introdução ao design de interfaces," *FCA-Editora de Informática*, 2012.

[62] M. W. Jaspers, T. Steen, C. Van Den Bos, and M. Geenen, “The think aloud method: a guide to user interface design,” *International journal of medical informatics*, vol. 73, no. 11-12, pp. 781–795, 2004.

[63] S. G. Hart and L. E. Staveland, “Development of nasa-tlx (task load index): Results of empirical and theoretical research,” in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.