Challenges in Designing Software Architectures for Web Based Biomedical Signal Analysis

Alan Jovic[1*], Kresimir Jozic[2], Davor Kukolja[1], Kresimir Friganovic[1], Mario Cifrek[1]
[1]University of Zagreb Faculty of Engineering and Computing, Unska 3, HR-10000 Zagreb, Croatia,
[2] INA - industrija nafte, d.d., Avenija Veceslava Holjevca 10, p.p. 555, HR-10002 Zagreb, Croatia
[*]Corresponding author: alan.jovic@fer.hr

Abstract. In contemporary explorations of biomedical data, there is a strong inclination towards software platforms that offer ubiquitous access and ease of use. Traditional biomedical signal analysis, such as detection of disorders in electrocardiogram recordings is considered as difficult task. There are many approaches to tackle such tasks, but the common assumption is that such an analysis should be performed both offline (after data collection) and particularly off the web, using MATLAB (or Octave) programs or other specialized software. There has been little effort involved in complete web solutions for biomedical signal analysis, which, compared to traditional analysis, have many advantages: access from afar, browser-only software requirements, reliance on fast server solutions for calculations, etc. This chapter considers web systems for biomedical signal analysis and describes their software architecture design aspects. It examines its applications: home care, medical education, signal repositories with visualization capabilities, signal analysis environments with data mining. The focus is on challenges involving these systems, like: data privacy, frontend workflow, frontend and backend interactions, database design, integration of data analysis and reporting libraries, programming language issues, etc. We emphasize that little work was published regarding architectural considerations in this setting and highlight the importance of overcoming these issues.

1. Introduction: Background and Driving Forces

There has been a significant progress made recently in scientific, computerized exploration of biomedical data, where novel software platforms are developed that offer remote access and data mining capability. Traditionally, biomedical time series (signal) analysis tasks include, among many others: detection of disorders by processing of electrocardiogram (ECG) recordings [1], analysis of sleep in electroencephalogram (EEG) recordings [2], analysis of gait and muscle dynamics using surface electromyogram (EMG) [3], calm/distress classification by skin conductivity [4], etc. All of these tasks are considered as difficult to accomplish, since mathematical methods for accurate analysis are computationally demanding and since the difference between a disorder and healthy state may not be totally reflected in the analyzed signal. There are many approaches developed that offer solutions to such analysis problems, but the mainstays are still considered to be offline analysis (after all the data was collected) and off-the-web analysis, using either MATLAB (or Octave) based scripts or frameworks [5], or other specialized software, e.g. for heart rate variability analysis [6].

The primary reason for such a mainstay, aside from biomedical engineering tradition, lies in complex algorithmic procedures for signal processing and analysis that take a lot of computer time to complete, visualize and interpret [7].

In Fig. 1, we depict one of the usual workflows for a complex biomedical signal analysis task. The primary steps included in such a workflow are:

1. the acquisition of signal data, either from an existing web repository, local database, file storage or directly from a measuring instrument [8,9],
2. signal preprocessing, including noise filtering, signal transformations, characteristic points detection [10],
3. feature extraction, usually recommended by medical experts or medical guidelines for a particular type of signal and analysis goal [11],
4. feature selection and/or dimensionality reduction, usually based on several statistical and machine learning techniques (e.g. principal component analysis, filter based selection, wrappers, etc.) [12-14],
5. model construction, usually involving multivariate inductive statistics, classification, regression, or unsupervised learning machine learning methods [15,16],
6. results visualization and interpretation, using a variety of statistical and machine learning evaluation tools and methods [17].

Although some steps may be skipped, depending on the analysis goal, input signal type and methods used, the entire process may be automated only partially and is cumbersome on the hardware and software resources of the platform on which it is performed. Also, special expertise in biomedical engineering research field is needed in order to intervene in the process.

In order to ameliorate the issues with the hardware and software resources, the majority of novel research focuses on remote healthcare, where services operate in the way to offer remote collection of data, transfer of data via network, and medical center based analysis on a server or cloud architecture using a variety of complex signal processing tools [18]. Still, the main issue with such a process is that both signal analysis and signal inspection are considered to be reserved only for medical and biomedical engineering experts, acting locally, in hospitals or research centers.

There has been little effort involved in complete web based solutions for biomedical signal analysis that would enable on-the-web analysis of patient's data. The advantages of web based solutions for biomedical signal analysis are plenty: remote and ubiquitous access, browser-only software installation requirements, reliance on fast server solutions for calculations, independent client and server side development, etc.

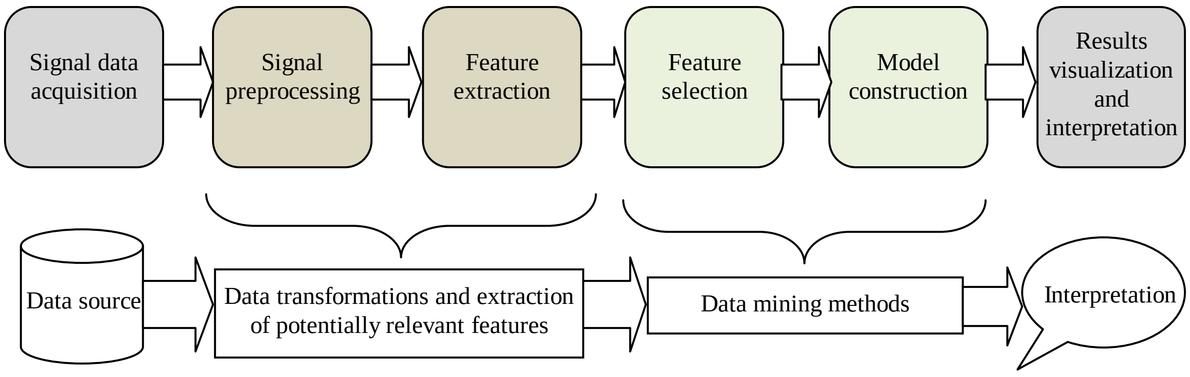


Figure 1. Complex biomedical signal analysis scenario

In this chapter, we mostly consider web based systems that enable biomedical signal analysis. We focus on software architecture aspects in designing such systems, from the perspective and experience of an ongoing research project, named MULTISAB, aimed at developing both research and application based web platform for parallel, heterogeneous biomedical time series analysis, intended for medical disorder diagnostics and classification [19-21] that would support complex analysis scenarios such as the one depicted in Fig. 1. For the sake of completeness, we also examine typical applications of web based systems for home healthcare that have very little or no options for the end user [22], medical educational software platforms that offer training to medical staff regarding biomedical signal analysis [23], biomedical signal repositories that offer both signal recordings and signal visualization [8], as well as electronic and personal health record systems that are related to the current topic [24].

The chapter does not consider offline and off-the-web biomedical time series analysis software developed in specific medical domains, although there are many examples of such software [5,6,25,26]. We also do not discuss numerous biomedical image applications and software in this work, given either as a web or as offline solution [27-30].

Challenges in developing web based biomedical signal analysis software are considered, such as: data privacy, security and user roles, frontend workflow organization, frontend and backend interactions, changes in implementation languages and libraries, database design, integration of existing data analysis and reporting libraries, and workload amelioration. We would like to emphasize that not much work was published regarding these practical aspects of biomedical time series analysis software design and implementation in a web setting. Therefore, through an in-depth exploration of these challenges, we highlight the importance of further development of web based solutions to end users.

In this chapter, we will intermittently use the terms "biomedical signal", "biomedical time series" and "physiological signal" as synonyms. In subchapter 2, an overview of related work is provided, and in subchapter 3, architectural challenges are presented, with some examples and discussion from our developing MULTISAB web platform. Subchapter 4 discusses specifics regarding design, hardware, software and other requirements needed to construct a biomedical signal analysis web platform. Subsection 5 provides a short discussion and conclusion of this chapter.

2. Overview of Web Based Systems for Biomedical Time Series Analysis

Web based systems for biomedical time series analysis have evolved significantly since the late 1990s (the advent of major internetization). The evolution of such systems has had several directions:

- web physiological data repositories, highlighted mostly by the well known PhysioNet website [8];
- remote healthcare systems for online patient monitoring, based on improvements in wireless sensor technologies [31-33], with different storage capabilities, such as cloud infrastructure [34], and governed today by a multitude of companies covering various aspects of online service, see e.g. [35];
- medical educational software, intended for better understanding of complex physiological processes, elucidated by biomedical time series visualization [36];
- electronic health record systems, primarily intended to replace traditional paper health records and to allow information sharing among medical specialists and primary care physicians for the purpose of improving patient care [37], but also intended for clinical decision support, usually taking into account multiple information sources available in the record [38], still having many architectural challenges [39];

- research and application oriented platforms/environments that enable decision support and data mining of time series data, with major focus in bioinformatics [39], and only with recent development in biomedical time series data analysis [19,40].

2.1. Web Physiological Data Repositories

Web physiological data repositories, such as the renowned openly available PhysioNet platform [8], or [41] were developed in the last two decades as an ongoing effort to promote better understanding of biomedical time series data through scientific exploration of their mechanisms and behavior. Although a web repository may provide a user with somewhat limited analytical capabilities, its primary use is to expose reliable and available anonymized patient data to interested researchers and other users. Contributors to databases in web repositories are usually hospitals and established medical research centers. The web repository allows better comparison of developed feature extraction and data mining algorithms among researchers, usually with standard physiological signal databases available as reference points (e.g. MIT-BIH Arrhythmia Database database on PhysioNet web portal).

Web based biomedical signal visualization and off-the-web use of biomedical signal analysis software (such as WFDB toolkit available from PhysioNet [8]) are also important aspects that contributed to widespread influence of the web signal repositories. Aside from open web physiological data repositories, there also exist commercial or membership oriented repositories, such as the THEW project [42] and Ann Arbor Electrogram Libraries [43] that offer similar or even improved (better annotated, higher sampling quality) data records, compared to the openly available repositories, but at a price.

2.2. Remote Healthcare Systems

In aging societies, the importance of remote healthcare systems cannot be overstated. Indeed, immediate assistance in the case of medical emergency is imperative, and organization of technological environment so that it best suits patient's needs and limitations has become a largely investigated issue. Specialized scientific conferences (e.g. HealthyIoT, AmIHEALTH) and journals (e.g. Journal of Healthcare Engineering, Journal of Biomedical and Health Informatics, Sensors) cover this research area. Scientific literature usually focuses on descriptions of wireless sensors and connectivity protocols [21,44], with the majority of applications in home environments [45,46]. Particular effort is usually attributed to server side data processing [47], where many different architectures and technologies may be used, e.g. cloud based architecture [27,48], standard server with relational database architecture [49], and standard server with online (script based) processing [47].

Data mining in remote healthcare systems may be related to simple outlier detection (whether sensor anomaly or patient emergency) [50], fuzzy rule based diagnostic systems that alert physicians in the case of emergency [51], prediction of disorder onset (as is the case with blood glucose in diabetes) [52], or a more complex expert system that allows multiple diagnoses based on a variety of measured parameters [47], to name just a few. Some of the analyses are more suitable for online processing, such as anomaly detection and rule based alerting, while others are more appropriate for offline analysis, such as prediction of disorders. There are multiple challenges involved in remote health monitoring systems, such as data acquisition and preprocessing (in particular, a lack of standardized wireless sensor solutions), medical equipment pricing, data privacy preserving, network coverage and bandwidth allocation, data modeling methodology, results evaluation options, etc. [27,53,54].

Web solutions in remote healthcare system usually offer unrestricted access only to qualified physicians and medical personnel [22], with home (patient) side of the web interface being either non-existent [55] or limited to simple actions [54,56,57]. For example, a user may see some measured health parameters through a web based interface and request a feedback from a medical expert [57].

Remote monitoring by smart phones and tablets may offer price and access related improvements [58], although mobile applications may present an obstacle for some patients if the usability is not in focus [59]. Web interface on the server side differs from one system to another, but usually supports: patient vital signs (emergency) monitoring, recorded signal visualization for inspection, and in some occasions, data mining [47,60].

2.3. Medical Educational Software

Although there are many instances of web based medical software intended for physicians' education, such as software for medical 2D [61] and 3D imaging [62,63], there are only a handful of online systems available for biomedical time series analysis education, most of them ECG related [23,36,64].

The purpose of online learning system is to provide medical students with easier access to a large number of interesting examples of physiological recordings, as well as to accelerate students marking. Also, automated computer based detection and annotation of morphological signal features (such as waves in ECG) and comparison with expert based annotations is another application of the on-the-web educational approach [23]. Aside from specialized educational software, medical personnel also have the option to use web physiological data repositories, as the ones described in section 2.1, for educational purposes.

2.4. Electronic and Personal Health Record Systems

Contemporary electronic health record (EHR) system may be considered as a big data system, particularly in the sense of large data volume and considerable data variety (structured text, unstructured text, signals, images, 3D data) [65]. This is clearly reflected in various categories of medical knowledge required to describe the state of a patient through time. The knowledge may include standard medical history, demographical information, medications taken, various laboratory tests results, undergone treatments, and many forms of recorded signals and images (e.g. ECGs, echocardiography, radiology, MRI [66]). To support this kind of information diversity and communication between medical institutions, and not only within institutions, as is the case with standard electronic medical records (EMR), a set of standards and regulations were developed, such as HL7, ISO 13606 (and its underlying standards), DICOM, and ISO/TC 215's set of health informatics standards.

The experience of using EHRs in hospitals is mostly positive, with reported improvements in hospitals efficiency [67]. However, focusing specifically on the information coming from biomedical signals in EHRs, there are very few proposed and implemented solutions for storing ECG, EEG, and other important biomedical signals directly in an EHR [68]. The main issue lies in integrating various heterogeneous formats of the signal recordings with the existing architectural standards in EHRs [69]. As a commonly used alternative, the signals may be stored (and also transformed to images) in a separate record system unavailable to the EHR. In such cases, usually, only a final report is summarized and presented in the EHR [70]. This certainly prohibits the interoperability between medical centers and therefore lowers the quality of service to patients.

There are some efforts to standardize various formats, such as the one in ECG analysis domain, by conversion of various ECG formats to DICOM-PACS image format [24] and integrating ECG-as-image in EHR stored in the cloud. Such images may be used later for learning disease models based on image mining techniques, involving automatic detection and classification [71]. Nevertheless, aside from individual uses in medical centers on the pay-per-use basis, the full service of EHR, especially in the context of biomedical signals analysis, has not yet been established [72]. The hospitals and medical centers are sometimes unwilling to use the available cloud based solutions for EHRs, and this is not without a cause. Some of the major challenges that influence hospitals in having such a stance, aside from the incompleteness of structure of EHRs due to lack of standards, are: data safety and security issues, limited storage capacity, network unreliability and low transfer capacity [73]. Also, in the context of use of EHR in mobile devices (mobile health), which is considered as especially convenient for medical personnel and patients in remote areas, short battery life, small storage capacity, and limited processing ability of the mobile devices represent additional challenges [69]. However, using big data architecture and resources can lead to many opportunities in healthcare: data quality improvement, improved population management and health, early detection of diseases, accessibility, improved decision making, and cost reduction [73].

Personal health records (PHR) are a more recent development in the field of electronic health records. Essentially, the main difference between EHR and PHR is in the way of access privileges and user roles. Whereas EHR may be looked upon and updated only by medical professionals, PHR may be read and, sometimes, modified by the patient, since it is considered as the patient's property. Alyami et al. [74] mention two types of personal health record systems: untethered and tethered. In untethered PHR, patient has full control over his PHR, where he can collect, manage and share data in his record. Tethered PHR is linked to specific healthcare provider's EHR (or hospital EHR), and can be used remotely by patients to view their data. Patients sharing untethered PHR with healthcare providers may be one venue to go in supporting interoperability between medical centers. In this respect, blockchain technology, developed for data privacy, secure access and scalability [75] may play an important role in developing applications for serving PHR to interested and authorized users [76]. We do note that we are unaware of any current research that investigates individualized web based biomedical signal analysis and processing based on personal health record information.

2.5. Research and Application Oriented Web Platforms/Environments for Biomedical Signal Analysis

Decision support systems (DSS) may be included as an important part of EHR systems, with the intent of helping the medical practitioner in reaching accurate diagnosis, treatment or prognosis. DSS are of particular importance to less experienced physicians and general physicians in rural areas, where absence of experts in a particular domain (e.g. cardiology) as well as medical emergency requires immediate and educated action [77]. Web based solutions in a form of web application that accesses EHR in order to provide decision support should allow easy access to relevant data and conclusions to doctors in a well defined clinical domain [78].

Research focused web platforms that offer some analysis and decision support based on biomedical signals are mostly developed for some specific domain, e.g. stress level in virtual reality environments [40] or ECG diagnostic interpretation [79]. As far as we are aware, there are currently no complete research and application oriented web platforms available that would allow biomedical signal analysis and data mining of multiple heterogeneous signals, whether in real time, or offline. Still, at least one such platform is currently in development [20,21].

3. Architectural challenges in web based biomedical signal analysis software

3.1. Data privacy, Security and User Roles

Data privacy is a very important topic in biomedical data analysis. In recent times, a number of software exploits are rising. Most reasons are related to bugs in software and to weak configuration of servers. Programmers don't keep security and possible consequences of software malfunctions in mind. Common errors are related to poor or absent verification of user input and to an absence of verification of sizes of data buffers. Possible consequences are crashes caused by invalid user input or leakage of confidential data [80]. The solution for this kind of problems is to use defensive programming techniques [81].

A weak server configuration allows an attacker to trick the server to use an older version of SSL or TLS security protocols that have many exploits and then to use one or more exploits to get confidential data. The solution is to pay more attention while writing server configuration files. Namely, all unneeded services should be disabled, all not user ports should be blocked by a firewall and only the latest version of TLS cryptographic protocol should be used [82].

The most common attacks fall in the following groups: 1) weak configuration of servers, 2) protocol design flaws, and 3) protocol implementation flaws. We provide examples for each group of attacks, as follows.

Group 1 example: Downgrade attack. A protocol downgrade attack tricks a server to use one of older versions of SSL/TLS protocols that have design flaws and then employ one of the many known exploits to get confidential data. This attack is performed by an attacker in the handshake phase of the connection. The client initiates a handshake by sending the list of supported TLS and SSL versions. An attacker intercepts the traffic, posing as a server (Man-in-The-Middle attack) and persuades the client to accept one of the older versions of TLS or SSL protocol. Now that the connection between the client and the server is established on an older protocol version, the attacker can perform one of many known attacks that exploit protocol design or implementation flaws.

Group 2 example: SSL 2.0 design flaws. SSL version 2.0 has many known flaws. One of the flaws is to use the same cryptographic keys both for message integrity and encryption. This is a problem if a chosen cryptographic algorithm is weak. If an attacker successfully breaks the encryption, he can change the content of a message. But he can also change the message integrity part that is used to verify its content. Another flaw is an unprotected handshake, which leads to Man-in-The-Middle attacks that can possibly go undetected.

Group 3 example: Heartbleed attack. Heartbleed attack exploits the bug in the OpenSSL library [83]. The client sends a "heartbeat" message to the server, which contains data and data size. The server responds with the same data and the size of the data that was received from the client. The problem is that if a client sent the false data size (bigger than the real size), the server responds with the data received from the client + random data from server RAM to fill the response to the required size. That random data can be e.g. password or encryption keys.

We can safely conclude that using only cryptographic protocols is not enough. For additional protection, authentication should be used. There are two major approaches: cookies based and token based [84]. Most websites use a strategy that stores a cookie in the browser. After a user logs in, he receives a cookie with the session identifier, which is used in a later request to the server. Cookie based authentication is stateful. This means that a session must be kept both on a server and on a client. Token based authentication is similar to cookies, but the major difference is that token based authentication is stateless. The server does not keep a record of which users are logged in. Every request from a client to the server contains a token, which the server uses to verify the request.

Arguably, the most popular token-based authentication technology nowadays are JWT - JSON Web Tokens [85]. JWT are used for representing claims securely between two parties. Representation is in the form HEADER.DATA.SIGNATURE. The header describes the token type and encryption algorithm. Data is user data that is protected by JWT. Signature contains header and user data signed by the encryption algorithm, for example:

Header:
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Data:
```
{
  "name": "Alan",
  "admin": true
}
```

Password:
"secret" - without quotes

JWT is constructed in the following form:
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
).
```

The JWT token looks like this:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiQWxhbiIsImFkbWluIjp0cnVlfQ.fTHiKsW8gH_Bp5AUzqoOTx7FVTL0PZZrnVBnto05He0

In a later communication, JWT token is used in this way:
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiQWxhbiIsImFkbWluIjp0cnVlfQ.fTHiKsW8gH_Bp5AUzqoOTx7FVTL0PZZrnVBnto05He0

JWT token is generated on the backend after the successful login of a user. It is used in the Authorization header of each HTTP request, which goes from frontend to backend afterwards. When backend receives JWT token, it verifies it, extracts the user credentials and generates a response to the HTTP request. If TLS encryption is compromised, an attacker can inject or modify JWT data, but the attack will be unsuccessful because JWT token will not pass verification. It is a recommended practice to limit the lifetime of a JWT token, which is done using the “exp” field.

One additional layer of security can be implemented on a server, even before transmission of data. That layer is the storage of files in an encrypted form. By implementing the layer, security is increased a lot, because the only point left to attack is RAM. Encryption can be achieved in two ways, depending on the file systems that are used. On older file systems (eg. FAT, ext2, ext3), which do not support encryption, a file is run through some encryption software or library and then stored as a regular file. On newer file systems, it is necessary only to enable encryption and then the files are encrypted on the fly before being stored. Most notable examples of the newest file systems are ZFS and APFS, which contain many other improvements in addition to file encryption [86]. The approach to use the newest file systems is the preferable method for encryption and it is very easy to implement, because server administrator just needs to create a file system that supports encryptions and enable it.

In the most recent times, exploit of hardware vulnerabilities are beginning to emerge. Most notable are Meltdown and Spectre [87]. These vulnerabilities are related to CPU parts that are responsible for speculative execution and caching (used for speeding up program execution), but they follow different routes. Meltdown is used to attack a kernel (the core of and operating systems) and Spectre is used to attack another user program. Meltdown is simpler and easier to protect from. The solution is to isolate the kernel and user program page tables. Spectre is more complex and more dangerous, because it can get confidential data from a program that doesn't have bugs. Spectre patches are complex and significantly slow down the performance of a CPU.

Data anonymization is important in medicine in order to avoid identification of patients to unauthorized users. As a part of data security for a biomedical signal analysis platform, the users should be made aware that, if possible, anonymized data only should be the one sent to the analysis process on the web platform. Such data, stripped of any possibility for patient identification, should be used to reach diagnosis or to model a disorder through the use of the platform. Since data anonymization procedures may be complex and are investigated in details in literature, they are considered beyond the scope of this work, but we nevertheless point the interested reader to some of the existing related work [88,89].

User roles are the usual mechanism used to prevent unauthorized user to access confidential data and to prevent damage to a database [90]. Administrator role does not have restrictions and a regular user is restricted to access data for which administrator grants access to him. It is also possible to define multiple levels of user roles, which grant access to more or less data. As an example, administrator of hospital EHR system can change everything: user data (user is a doctor), patient data, password, system information, etc. Superuser of hospital EHR system can change user data, patient data, and password, but cannot change system information. Regular user (doctor) can change only patient data and his own password.

3.2. Frontend Workflow Organization

Workflow organization of a frontend solution for complex signal analysis poses a problem because of many interconnected analysis steps and actions that are not directly related to signal analysis. There are several problems that must be solved: how to organize display, how to switch between steps of analysis, how to pause and continue analysis session, etc.

The MULTISAB frontend is built using Angular web framework [91]. The central concept of Angular is a component. A component consist of HTML, CSS and TypeScript (or JavaScript) code. The MULTISAB consists of a large number of components. Some components are unrelated to biomedical signal analysis (e.g. user login, main windows, change the password, used data editor), but most of them are. Navigation between components is done by using routes. An example of routes in MULTISAB frontend is shown in the following code segment:

```
export const routes: Routes = [
    {path: "", redirectTo: "/login", pathMatch: "full"},
    {path: "login", component: UserLogin},
    {
        path: "panel", component: PanelComponent,
         children: [
            {path: "", redirectTo: "/panel/multisab", pathMatch: "full"},
            {path: "multisab", component: MultisabComponent},
            {path: "type_selection", component: TypeSelectionComponent},
            {path: "scenario_selection", component: ScenarioSelectionComponent},
            {path: "file_upload", component: FileUploadComponent},
            {path: "record_inspection", component: RecordsInspectionComponent},
            {path: "records_preprocessing", component:
             RecordsPreprocessingComponent},
            {path: "features_extraction", component:
             FeaturesExtractionComponent},
            {path: "features_selection", component: FeaturesSelectionComponent},
            {path: "model_construction", component: ModelConstructionComponent},
            {path: "reporting", component: ReportingComponent},
            {path: "manage_users", component: ManageUsersComponent},
            {path: "change_password", component: ChangePasswordComponent},
            {path: "edit_self_data", component: EditSelfDataComponent}
        ]    }
];
```

Navigation is orchestrated by the main component, which is also responsible for the layout of components on the web page. Navigation between routes that are not related to biomedical signal analysis is not restricted, except for limitations imposed by user roles. For example, an ordinary user can only do signal analysis and change his password. Navigation between routes that are related to biomedical signal analysis is regulated by a finite state machine. This allows navigation to components that are related to current state and to action that a user wants to perform. For example, when a user creates new analysis, he is permitted only to close the analysis or to select the type of analysis. This is important in order to support the usual process of biomedical signal analysis, in which one usually moves forward, starting from a state in which the analysis goal is set and ending in the state where reporting the analysis results is performed. A transition between the states is resolved on the frontend and sent to the backend, which stores it in the database. The backend does not have any semantics regarding the states, except when it filters open and closed analysis sessions. We show all the possible state transitions in Table 1. For a selected state in a row, X represents the allowed state for transition (in a column).

3.3. Frontend and Backend Interactions

Because of data confidentiality, all communication in MULTISAB goes over a secure

Table 1. State transitions in MULTISAB frontend finite state machine. S0: Start state; S1: New analysis session; S2: Continue analysis session; S3: Close analysis session; S4: Select analysis type; S5: Scenario selection; S6: Input data selection; S7: Records inspection; S8: Records preprocessing; S9: Features extraction; S10: Features selection; S11: Model construction; S12: Reporting

State	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
S0	X	X										
S1			X	X								
S2												

S3	X											
S4			X	X	X							
S5			X	X	X	X						
S6			X	X	X	X						
S7			X			X	X	X	X			
S8			X			X	X	X	X			
S9			X			X	X	X	X			
S10			X							X	X	X
S11			X							X	X	X
S12			X									X

connection. Contemporary HTTP/2 protocol was chosen for communication, because it supports TLS and many other features [92]. Although HTTP/2 does not require encryption, the majority of web browsers only support HTTP/2 over TLS. HTTP/2 has several strong points, of which the most prominent are that:

- It is binary protocol, so it is much faster to parse than the previous versions of protocol, which were textual.
- It is multiplexed, so it significantly reduces round trip times without any additional optimizations. In previous versions of the protocol, every request was followed by the response. In version 2 of the protocol, many requests can be sent at once followed by one (or more) responses.

After a successful login, backend generates a random number, which is used as authorization token, and sends it to the frontend. The token represents a session that is maintained on the backend. Frontend sends the token to the backend in every request. Backend verifies the token and, according to session data, determines if the user is logged in and if the user is allowed to access a resource, with respect to his role. Frontend often sends the request to the backend to refresh the token. If the token expires, backend automatically logs out the user for safety reasons.

All communication between the backend and the frontend is done over RESTful API. However, contrary to common practice of using GET, PUT, POST and DELETE HTTP requests, we have chosen to use only POST request. The consequence of that decision is that we need to use one element or URL path more than the usual approach. For example, for deletion of user data:

Common practice:

```
DELETE /api/users
{
    userid: 123456
}
```

MULTISAB implementation:

```
POST /api/users/delete-user
{
    userid: 123456
}
```

We consider that our approach is better, because API structure looks more uniform (although a bit more complex) and reduces the number of errors during programming, since we can copy-paste functions that create HTTP request and change only the URL. Although it is generally believed that RESTful architectural style should mirror CRUD (create, read, update, delete) to HTTP methods POST, GET, PUT and DELETE, according to the author of the RESTful protocol, this is not mandated [93].

3.4. Changes in Implementation Languages and Libraries

Programming languages evolve over time. Some changes improve performance, other simplify or improve syntax and thus make writing programs easier. Changes in programming languages, no matter how small, often break working programs. Changes in languages also cause changes in its libraries [94]. If a program uses more libraries, then there is a higher probability that it will not work correctly when a programming language change occurs. It is often necessary to create wrappers or workarounds for the libraries that cause problems due to language changes. These generally work only temporarily. For some changes, there is no easy solution and parts of a program must be rewritten. From our experience, changes occur faster in frontend technologies compared to backend technologies and it requires more time to modify a program and to ensure that security issues are not introduced.

On MULTISAB backend, we had an occasion where we had to modify several dozen lines of code, because Java 9 changed legal identifier names [95]. Apparently, Java 9 removed the symbol "_" (underscore) from the set of legal identifier names. That problem was easy to solve, because compiler issued an error for each occurrence of "_" as an identifier name, so we only had to follow the reported errors and rename the variables.

Additionally, Java 9 introduced the module system. We soon recognized the importance of the module system, because it adds an additional layer of security to the usual private / protected / public visibility scope. By using module system, it is possible to completely hide the classes that programmer does not want to export. For the exported classes, programmer continues to use the usual visibility scope. When we tried to implement the module system in MULTISAB backend, we ran into a problem, because some of the external libraries that we use had issues with the module system and stopped working. We had to revert our code to the previous version and wait for the libraries to be converted to Java 9.

On MULTISAB frontend, we also had issues with RxJs library a few times. The Angular library changes often (rarely causing a problem), but the RxJs library, which we use for reactive programming, changes somewhat slower. On several occasions, we had a problem with the inclusion of the RxJs operators. The initial strategy was to wait several days for RxJs to resolve problems. The strategy was soon abandoned, because we wasted time waiting. Therefore, we decided to create a workaround for each problem as a temporary solution. When the next version of RxJs solved some problems, the workarounds were removed.

Not all changes are negative or require a lot of work. For instance, when Angular introduced HttpClient service as a replacement for the old Http service, we saw that as an opportunity to simplify our program. HttpClient response returns JSON object by default, so it became unnecessary to explicitly parse textual response. Changes were trivial to implement and the end result was a smaller code base.

On the other hand, the use of contemporary frontend development frameworks such as Angular for developing the platform also poses a challenge. For example, in order to allow the upload of multiple files in the platform in an elegant, drag-and-drop manner, as shown in Fig. 2, we needed to write our own component with a significant number of lines of code (as there was no readily available one), as is shown in the example:

```
import {Component} from "@angular/core";
import {Observable} from "rxjs";
import {AnalysisUrls} from "../../services/url/AnalysisUrls";
import {PrintMessages} from "../../services/PrintMessages";
import {Router} from "@angular/router";
import {HttpErrorResponse} from "@angular/common/http";
import {Language, LocaleService, TranslationService} from "angular-l10n";

@Component({
    templateUrl: "/app/components/analysis/FileUploadComponent.html"
      })
export class FileUploadComponent extends PrintMessages {
    @Language() lang: string;
    files: FileList;
    progress: Array<number> = [];

    constructor(private analysisUrls: AnalysisUrls, private router:
      Router,private locale: LocaleService, private translation:
      TranslationService) {
        super(translation);
    }
    fileChangeEvent(event: Event): void {
        this.files = (event.target.valueOf() as
              HTMLInputElement).files;
        if (this.files.length === 0) {
            this.progress = [];
        } else {
            this.progress = new Array<number>(this.files.length);
        }
    }
      dragHandler(event: DragEvent): void {
        event.stopPropagation();
        event.preventDefault();
    }
    dropHandler(event: DragEvent): void {
        event.stopPropagation();
        event.preventDefault();
        this.files = event.dataTransfer.files;
        if (this.files.length === 0) {
            this.progress = [];
        } else {
            this.progress = new Array<number>(this.files.length);
        }
    }
    submit(): void {
        for (let prog of this.progress) {
            prog = 0;
        }
        for (let i: number = 0; i < this.files.length; i++) {
            this.uploadFile(i);
        }
    }
...
}
```

We also need to write an HTML template for the component (`FileUploadComponent.html`). In this case, the code is not long and is given here in full:

```
<p-growl [value]="msgs"></p-growl>
<div style="position: absolute; width: 450px; top: 10px; left: 10px;">
    <form>
        <div id="drop_zone" (drop)="dropHandler($event)"
             (dragenter)="dragHandler($event)"
             (dragover)="dragHandler($event)" (click)="inputFiles.click()"
             style="border: dashed 2px lightgray; border-radius: 20px; height:
          125px; font-size: 20px; color: gray; width: 100%; display:
              table;">
            <span style="text-align: center; display: table-cell; vertical-
       align: middle;">
                {{'FileUpload.dragFiles' | translate:lang}} <br>
                {{'FileUpload.dragFilesOrClick' | translate:lang}}
            </span>
        </div>
        <div hidden>
            <input type="file" id="file" (change)="fileChangeEvent($event)"
            multiple #inputFiles>
        </div>
        <br><br>
        <div style="text-align: center;">
            <button type="submit" pButton (click)="submit()" label="
            {{'FileUpload.buttonSend'   |   translate:lang}}"   [disabled]="files   ==
          null"></button>
            <button type="submit" pButton (click)="nextPage()"
            label="{{'FileUpload.buttonNext' | translate:lang}}"></button>
        </div>
        <hr *ngIf="files != null">
        <div style="width: 100%" *ngFor="let file of files; let i = index;">
            <label>{{files[i].name}}</label>
            <p-progressBar [value]="progress[i]"></p-progressBar>
        </div>
    </form>
</div>
```

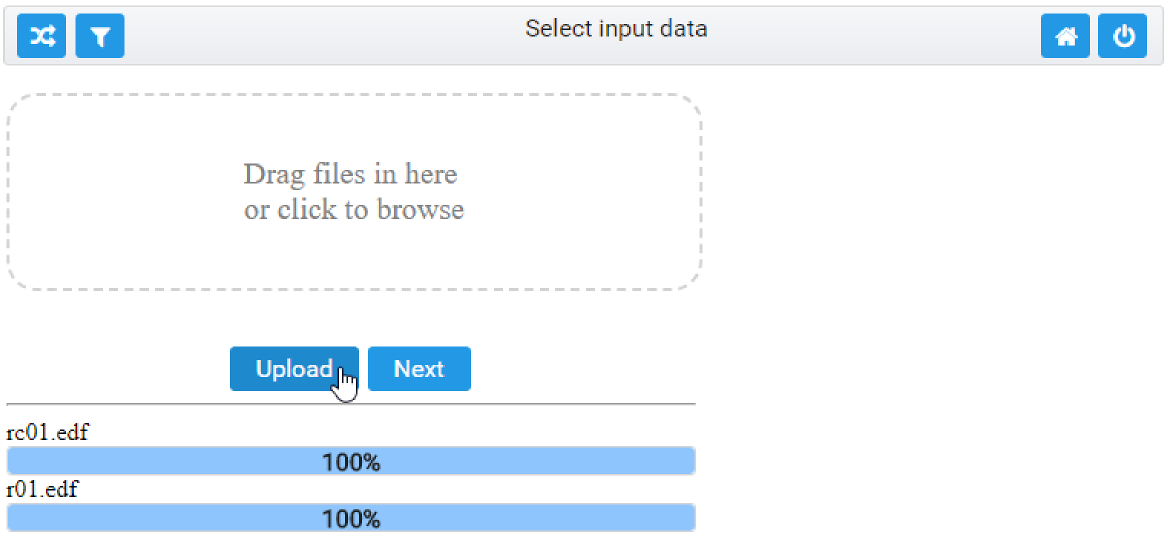


Figure 2. Multiple file upload page on frontend, as an example of a complicated Angular component, with some example files uploaded onto the MULTISAB platform

Hence, in our experience, there are significant challenges in keeping up with frontend development languages and technologies, which slows down the overall platform development. We suspect that fast language and libraries evolution, in particular related to frontend, may be one of the reasons for having such a small number of relevant online biomedical signal analysis platforms. Another reason may be because traditional biomedical signal analysis community needed not to concern much with frontend development technologies, and due to the complexity of its programming, the implementation of such a platform remained out of the community's reach.

3.5. Database Design

Changes in database inevitably lead to changes on the backend and likely to changes in the frontend. The opposite direction is true as well, changes in backend and frontend cause changes in database. In the period of planning and implementation of a database, it is hard to predict all the possible use cases, and therefore the database design should evolve as implementation progresses, much as any other program. An important advice is not to try avoiding changes in database, because then, backend and frontend suffer. Changes done in one of the three segments (database, backend, or frontend) should trigger changes in the other two segments. Hence, it is necessary to have good code organization in all three segments.

In MULTISAB, we have only one central place for data prototypes in each of the three segments. In the database, we have the file "Database.ddl" that holds data definitions of the database tables. All changes in the file are propagated to database implementation. In the example, we provide the definition of a class Phase that is a part of implementation of our database.

```
CREATE TABLE Phase (
  Id   BIGINT NOT NULL,
  Name VARCHAR(256)    NOT NULL,
  PRIMARY KEY (Id),
  CHECK (LENGTH(TRIM(Name)) > 0),
  UNIQUE (Name)
);
```

On backend, we use Java Persistence API - JPA. This is a mechanism that maps Java classes to database tables by using annotations. JPA defines Java Persistence Query Language, which is a simplified version of SQL language and is adapted to the object oriented way of programming. Changes in the database reflect easily to backend. One only needs to change the variable definitions and annotations. In the example, we provide the definition of the class Phase in backend.

```
@Entity
@Table(name = "PHASE")
public class Phase implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "ID")
    private Long id;
    @Basic(optional = false)
    @Column(name = "NAME")
    private String name;
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

On frontend, we keep the TypeScript class prototypes in services, which are responsible for their part of RESTful API (e.g. LoginService, SessionService). Changes in frontend are not 1:1 related to changes in databases. Instead, they are related to changes in the RESTful API.

To simplify documentation of MULTISAB RESTful API, we use OpenAPI. OpenAPI was originally known as the Swagger Specification. Description of a user RESTful API is contained in a YAML file [96]. For editing the YAML file, Swagger Editor tool can be used, while for viewing, Swagger UI tool is available [97]. Both Swagger Editor and Swagger UI translate YAML file to HTML and visually display users' RESTful API, see our example in Fig. 3.

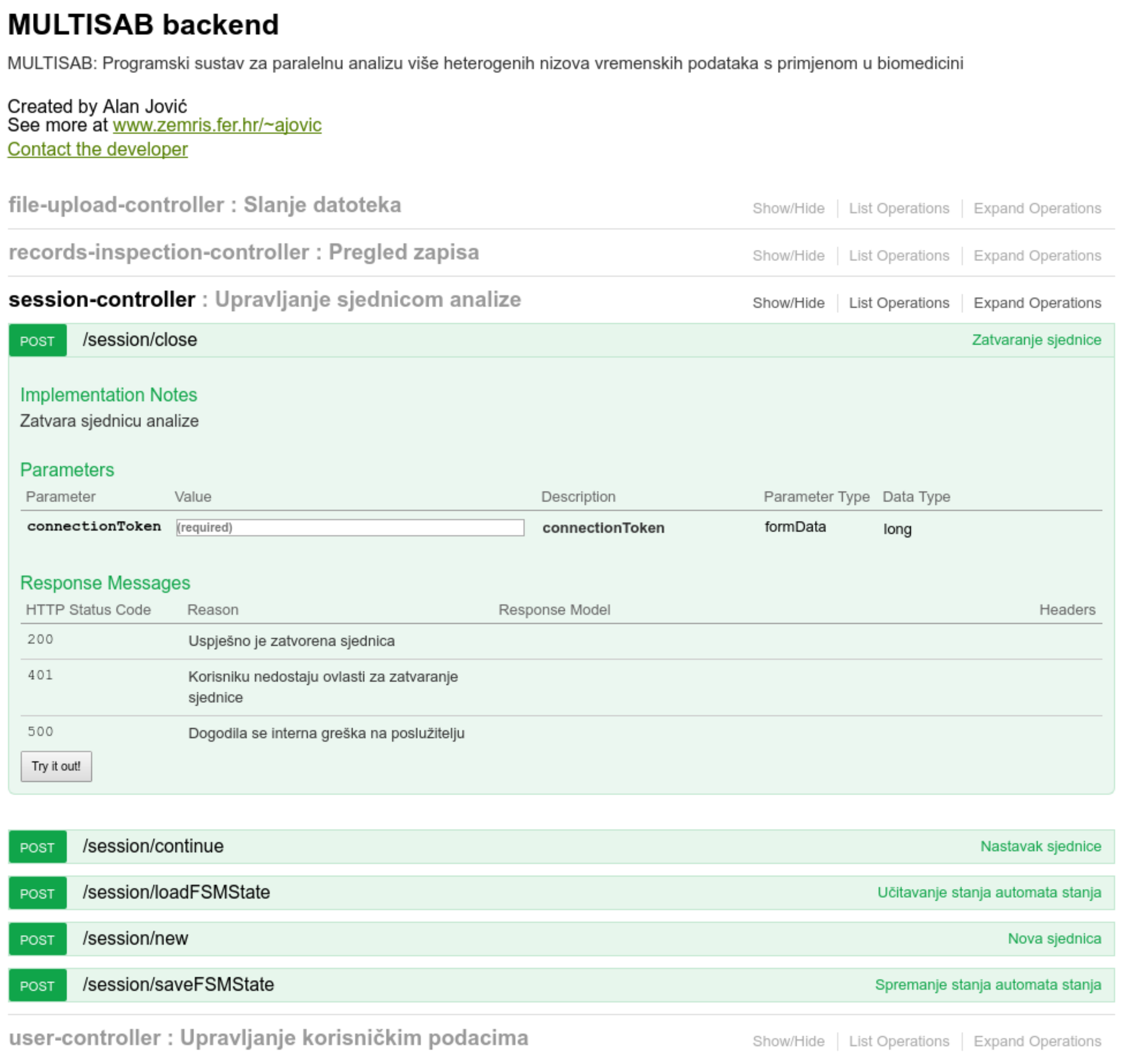
Our experience has shown that RESTful API should be edited first. Afterwards, backend and frontend can be programmed according to the changes, and database can also be changed if needed. If RESTful API needs to be changed due to changes in database, backend or frontend, we recommend to change the documentation first and then propagate the changes accordingly.

Thus, one can always have an up to date documentation and one can develop different parts of program independently, using documentation to synchronize.

In MULTISAB project development, we use a simplified approach, skipping the usageof the Swagger Editor. We first write backend code and annotate it with comments. In this case, we merge two steps into one - writing of documentation and implementation in the backend. Annotations in the code are transformed by the SpringFox library [98] into documentation. When backend is launched, documentation can be accessed over Swagger UI. The Swagger UI is packaged into the SpringFox so the end user just needs to visit the URL at http://localhost:8081/swagger-ui.html to view the documentation. As an example, we show the "closeSession()" function annotated with documentation comments. End result for the code can be seen in Fig. 3.

```
@RequestMapping(value = "/close",
      method = RequestMethod.POST,
      consumes = MediaType.APPLICATION_FORM_URLENCODED_VALUE,
      produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
 @ApiOperation(value = "Zatvaranje sjednice", notes = "Zatvara
   sjednicu analize")
 @ApiResponses(value = {
      @ApiResponse(code = 200, message = "Uspješno je zatvorena
        sjednica"),
      @ApiResponse(code = 401, message = "Korisniku nedostaju
        ovlasti za zatvaranje sjednice"),
      @ApiResponse(code = 500, message = "Dogodila se interna
        greška na poslužitelju")
 })
 synchronized ResponseEntity<Void> closeSession(@RequestParam
   long connectionToken) {
  Optional<UserData> data =
     loggedInUsers.findUserByToken(connectionToken);
     if (!data.isPresent()) {
         return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
     }
     if (closeSessionInternal(connectionToken, data.get())) {
         return new ResponseEntity<>(HttpStatus.OK);
     } else {
         return new
             ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
     }
 }
```

In the beginning of the MULTISAB backend project, we wanted a solution that would be easily portable between different computers and that could be copy-pasted from a portable medium without installation. Therefore, we chose Java as the programming language. For the main development framework on backend, we chose Spring Boot, due to its high efficiency in creating web applications [99]. Database candidates were Apache Derby and H2 [100].


MULTISAB backend

MULTISAB: Programski sustav za paralelnu analizu više heterogenih nizova vremenskih podataka s primjenom u biomedicini

Created by Alan Jović
See more at www.zemris.fer.hr/~ajovic
Contact the developer

file-upload-controller : Slanje datoteka — Show/Hide | List Operations | Expand Operations

records-inspection-controller : Pregled zapisa — Show/Hide | List Operations | Expand Operations

session-controller : Upravljanje sjednicom analize — Show/Hide | List Operations | Expand Operations

POST /session/close — Zatvaranje sjednice

Implementation Notes
Zatvara sjednicu analize

Parameters

Parameter	Value	Description	Parameter Type	Data Type
connectionToken	(required)	connectionToken	formData	long

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Uspješno je zatvorena sjednica		
401	Korisniku nedostaju ovlasti za zatvaranje sjednice		
500	Dogodila se interna greška na poslužitelju		

Try it out!

POST /session/continue — Nastavak sjednice

POST /session/loadFSMState — Učitavanje stanja automata stanja

POST /session/new — Nova sjednica

POST /session/saveFSMState — Spremanje stanja automata stanja

user-controller : Upravljanje korisničkim podacima — Show/Hide | List Operations | Expand Operations

Figure 3. Screenshot of Swagger UI displaying part of MULTISAB RESTful API

Apache Derby had an advantage of being a part of the Java distribution (Java DB). The main disadvantage was that it stores database files in many files and folders within the main folder. We wanted as few files as possible, so H2 database was chosen instead, because it stores all data in a single file. The disadvantage was that we had to pull the library from the separate repository. As we use Maven in our project, this disadvantage is insignificant, because Maven pulls H2 database library files as it also pulls other libraries that we use. H2 is a very compact DMBS (only 1.5 MB) and it lacks some features that larger DBMSs have. For example, it does not have a language for stored procedures and it does not support multi-threaded processing.

Our current database model may be found in Friganovic et al. [20]. The database is designed to support the workflow of the biomedical signal analysis platform. It is not designed to store signal records, but rather to contain user session data, including the scenario that is conducted through the platform, phase of the analysis in which the user is currently found (so that the continuation of the session that was last discontinued is possible), the preprocessing methods and the features that the user selected for the analysis. Also note that, aside from the analysis-related data, we only store user personal data. MULTISAB backend is not demanding regarding data storage, so H2 database is just enough for this purpose. This is because the files that contain biomedical signals, extracted feature vectors, models and reports are all stored in the server filesystem and only their names are stored in the database as a part of an analysis session. Although we may modify some parts of the database design in the future, it should be noted that the initial design that was proposed was found to be satisfactory thus far. We contend that such a database solution is good enough for a web based biomedical signal analysis platform.

3.6. Integration of Existing Data Analysis and Reporting Libraries

When one is developing an online biomedical signal analysis platform that also includes machine learning and data mining capabilities, one needs to consider two options: 1) development of machine learning algorithms from scratch, and 2) integration of already existing data analysis libraries. The decision between using the first or the second option is largely based on the intended platform licensing and commercial preferences of the platform.

In MULTISAB platform, we opted to either integrate the existing data libraries licensed under free and permissive Apache, MIT, BSD, and (somewhat less permissive) LGPL licences, or to write our own code. In this way, we keep the option to develop a commercial version of the platform if deemed needed. For some of the machine learning algorithms (e.g. neural network), we integrate the Encog framework [101], licensed under Apache 2.0 license. We use the libsvm library for support vector machines classifier [102]. This library is licensed under the modified BSD license.

We wrote some of the machine learning algorithms from scratch, particularly those pertaining to feature selection (e.g. symmetrical uncertainty, chi square, ReliefF), as we did not find any implementation under permissive licenses. For reporting the results of the analysis process (final statistics and evaluation measures results), we opted to use the JasperReports (JR) library [103]. JR is licensed under LGPL, briefly meaning that in other to keep the possibility of commercializing our platform, we can only use its API, not change its source code. The use of a separate reporting library should be considered suitable for a web platform in the case of biomedical signal analysis applications, as many different types of statistical results may be obtained from the analysis scenario (e.g. for classification: class distribution, confusion matrix, total classification accuracy, sensitivity, specificity, F1 measure, etc) [104]. Some of the results may be presented in a form of table or a list of evaluation measures, while others may use pie charts, histograms, etc.

Achieving a successful and uniform data connection between several different libraries and our implementations is done through classes that adapt the data for particular API requirements of certain libraries. As the platform is evolving, we expect to add more machine learning algorithms in time, either through our own implementations or by integrating permissive licensed implementations. We consider the approach we propose here reasonable and viable for similar purpose online analysis software.

3.7. Workload Amelioration

In the backend planning phase for the MULTISAB project, we decided to separate it into two parts. The first part is the core of the backend, which is responsible for database access, RESTful API, and user rights management. The second part is a subproject called *processing*. *Processing* implements all signal analysis algorithms, including machine learning, which are used in the MULTISAB project. We made that decision, because we wanted to have better control over intellectual property protection and to have the possibility to improve the performance of algorithms without touching the core backend functionality.

We started with the implementation of algorithms in a sequential way. After that, we conducted experiments with the OpenCL library. We did not use OpenCL directly, but over the Aparapi library [105]. Aparapi translates native Java bytecode in OpenCL kernels dynamically at runtime. We initially did synthetic benchmark tests on GPU and they looked promising. After that, we did the implementation of several concrete feature extraction algorithms in Aparapi. Results were far below the results achieved on synthetic tests. The conclusion was that the overhead of real-time compilation of Java bytecode to OpenCL kernels, as well as data transfer between CPU and GPU, took too much time. Another cause was that data analysis algorithms were either too simple, or if they were complex, too small amount of data was used to mask the latency of transfer to the GPU. We have also tried to use CPU as an OpenCL computing device, but the results were similar.

Finally, we made a decision to rely on the classical approach with Java threads. The parallelization procedure that we implemented for feature extraction step of the platform is, as follows:

- if there are multiple segments present in a signal, then these are resolved (all the features are extracted) in parallel,
- else if there are no multiple segments, but there are signals of the same type (e.g. EEG), then these are resolved (all the features are extracted) in parallel,
- if there are no multiple segments nor signals of the same type, but there are multiple patient files, then these are resolved (all the features are extracted) in parallel,
- lastly, if there is only a single file with a single signal of a particular type and a single segment, then it is treated as sequential and single threaded.

Currently, we use parallelization over primitive data arrays, however, in the future, we plan to use Java streams (introduced in Java 8). The concept of streams is simple; instead of viewing data as a collection (sets, maps), user sees data as a "stream of elements". A stream can be created from existing collections using functions "stream()" and "parallelStream()". By using "parallelStream()", elements are automatically parallelized. The user only needs to ensure that algorithms are suitable for parallel execution, which may be difficult in signal analysis.

Another optimization is possible and that is the usage of several computers. In the future, we will have to implement a kind of message passing interface to enable communication of the core of the backend with the *processing* instances running on several physical machines.

4. Requirements for Constructing a Biomedical Signal Analysis Web Platform

4.1. Hardware and Software Requirements

Architectural requirements for a web platform designed to perform biomedical signal analysis may vary depending on the amount of expected data processing and the analysis scenarios supported by the platform. For example, if one wants only to record some physiological signals in a home environment for a single (or a few) persons and provide a few health markers, which may be sent to a medical expert acting remotely for further (manual) evaluation, then such a system need not have significant hardware requirements. A usual personal computer or even a handheld device may suffice for the client side, provided that it works as a gateway to a remote server in a health institution [57,106]. In such a setting, a remote server is usually used for storing collected patient data for visual inspection by the medical professional and perhaps for a future, offline analysis.

However, supporting many users at the same time and performing complex analysis scenarios (as depicted in Fig. 1) may require more resources for a general and expandable solution. A typical minimum solution would include a single computer, acting as a server for data analysis with fast multi-core processor capabilities (4 or more logical cores), large hard drive and RAM capacities. The computer would need to have a web server installed to support the web application (e.g. Apache Tomcat), H2 or similar in-memory relational DBMS and would need to provide software support for the whole web development technological stack in order to accommodate for potential software improvements, hence: 1) frontend technologies, such as HTML, CSS, Bootstrap, JavaScript/TypeScript, Angular or similar frontend development framework, and 2) backend technologies, such as Java 9, Spring Boot, and JPA (or related backend Microsoft, PHP, or Python technologies). Additionally, permissive license libraries used to cover the various steps in biomedical signal analysis would be a welcome, but not a necessary requirement for construction of the web platform, as some of the required methods may be efficiently implemented from scratch.

Any expansion of the proposed minimal solution would be primarily concerned with workload improvements, as we already elaborated in subchapter 3.7. Although we focused mostly on single-computer parallelism in our work, distributed backend processing, through the use of OpenMP [107] bindings or through service oriented architecture based on remote method invocation [108] should be considered.

A step further might be the integration of the web based biomedical signal analysis platform within a cloud infrastructure. Although some cloud based solutions for medical big data analysis were already proposed in the literature [109,110], we are only aware of one solution that deals in any way with biomedical signal analysis in such a setting [111]. Namely, this solution, called Cloudware, focuses on ECG preprocessing and visualization through the use of Hadoop big data technology in a cloud. Nevertheless, the Cloudware platform still lacks the complete complex analysis scenarios for several different types of biomedical signals that we proposed in the MULTISAB platform and which we depicted in Fig. 1, as it is primarily intended for better visualization capabilities of ECG recordings.

4.2. Other requirements

Designing use cases for the requirements of the web platform proved to be a difficult task in our case, due to the complexity of possible analysis scenarios. Nevertheless, the use of UML use case diagrams to specify possible user behavior in communication with the system proved to be a valuable asset in development of the platform [112]. Therefore, we would recommend the use of UML tools, especially in specifying platform requirements through the use of scenarios.

Development of the web analytical platform requires that the team members are well-versed in a variety of frontend, backend and database technologies, aside from an expertise in signal analysis. This is something not often encountered or required in practice, and this may be one of the reasons such a platform has not been proposed earlier. As we have elaborated in section 3.4, numerous challenges with implementation languages and libraries exist, especially in frontend technologies. A well-versed, competent team requirement might seem as a trivial requirement at first, but in our experience, and this is also corroborated by other researchers [113], it is very important that all of the research team members are at least good acquainted with all the aspects of the platform. Still, we also agree with the conclusions of the Software Sustainability Institute that state that it may be counterproductive to instill all the software development details to all team members, as some of them are better in, e.g. signal analysis than in frontend development [114].

5. Discussion and Conclusion

Designing a web platform intended for biomedical signal analysis is challenging, as we showed in this work. Indeed, there are many contributing factors to design and implementation complexity. In subchapter 3, we presented some of the most pervasive challenges that we encountered in developing the MULTISAB platform. Not all the challenges were reported in detail here, though. For example, the issue of biomedical signal data itself is problematic, because analyzing multiple heterogeneous time series poses difficulties with respect to input data formats, feature extraction and parallelization. We reported progress on some of these issues in an earlier work [21] and we also plan additional publications to cover these issues in more detail.

We would like to stress out that one of the biggest problems in designing the web platform are the web development technologies, which are still mostly not standardized (apart from HTML) and are in continuous development. Thus, designing an offline biomedical signal analysis platform that would accommodate all the complex scenarios related to multiple heterogeneous signal analysis may be difficult, but designing the same platform for the web is even more difficult. We consider that the only other serious issue in web platform construction aside from the development technology is security. In biomedical applications, software safety and security need to be of the highest degree possible. As we have elaborated in this work, this issue has not been solved in a general sense. Although employing most recent security protocols, message encryption, and other advanced techniques should suffice in most cases [115], there are really no theoretical guarantees with the safety of web systems.

Integrating all of the technologies to have a working platform is something that we are currently working on. Despite the challenges explored in this chapter, we still consider that having such an integrated analysis platform would greatly benefit both researchers and medical professionals. Overcoming the challenges would allow distant access to platform with advanced diagnostical and analytical capabilities. In the future, we plan to investigate how the platform could be applied in medical practice, including:

- the modeling of body state, based on multiple patient records and signal types, e.g. for stress detection [116], pregnancy characterization [117], etc.,
- possible smooth integration with remote patient monitoring technologies to enable online analysis through its decision support capabilities [21],
- the integration with medical center computer infrastructure, in order to enable electronic health record information processing in the platform to achieve better medical diagnostics.

Acknowledgements

This work has been fully supported by the Croatian Science Foundation under the project number UIP-2014-09-6889.

References

[1] Clifford, G.D., Azuaje, F., McSharry, P.E. 2006. *Advanced Methods and Tools for ECG Data Analysis*. Norwood MA, USA: Artech House.

[2] Campbell, I.G. 2009. EEG Recording and Analysis for Sleep Research. *Curr Protoc Neurosci.* chapter: Unit 10.2.; doi:10.1002/0471142301.ns1002s49

[3] Cifrek, M., Medved, V., Tonkovic, S., Ostojic, S. 2009. Surface EMG based muscle fatigue evaluation in biomechanics. *Clinical Biomechanics* 24(4), pp. 327-340.

[4] Zangroniz, R., Martinez-Rodrigo, A., Pastor, J.M., Lopez, M.T., Fernandez-Caballero, A. 2017. Electrodermal Activity Sensor for Classification of Calm/Distress Condition. *Sensors* 17(10), p. 2324; doi:10.3390/s17102324

[5] Robbins, K.A. 2012. EEGVIS: A MATLAB Toolbox for Browsing, Exploring, and Viewing Large Datasets. *Front Neuroinform.* 6, p. 17; doi: 10.3389/fninf.2012.00017

[6] Tarvainen, M.P., Niskanen, J.P., Lipponen, J.K., Ranta-aho, P.O., Karjalainen, P.A. 2014. Kubios HRV – Heart rate variability analysis software. *Comput Methods Programs Biomed* 113(1), pp. 210-220.

[7] Srhoj-Egekher, V., Cifrek, M., Medved, V. 2011. The application of Hilbert-Huang transform in the analysis of muscle fatigue during cyclic dynamic contractions. *Med & Biol Eng & Comput* 49(6), pp. 659-669.
[8] Goldberger, A.L., Amaral, L.A.N., Glass, L., et al. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101(23), e215-e220, 2000.
[9] Baumert, M., Porta, A., Cichocki, A. 2016. Biomedical Signal Processing: From a Conceptual Framework to Clinical Applications [Scanning the Issue]. *Proceedings of the IEEE* 104(2), pp. 220-222. doi:10.1109/JPROC.2015.2511359
[10] Friganovic, K., Jovic, A., Kukolja, D., Cifrek, M., Krstacic, G. 2017. Optimizing the detection of characteristic waves in ECG based on exploration of processing steps combinations. *Joint Conf of the European Medical and Biological Engineering Conference (EMBEC'17) and the Nordic-Baltic Conf on Biomedical Engineering and Medical Physics (NBC'17)*, IFMBE Proceedings, vol. 65, Tampere : Springer Nature Singapore, pp. 928-931.
[11] Sassi, R., Cerutti, S., Lombardi, F., Malik, M., Huikuri, H.V., Peng, C.-K., Schmidt, D., Yamamoto, Y. 2015. Advances in heart rate variability signal analysis: joint position statement by the e-Cardiology ESC Working Group and the European Heart Rhythm Association co-endorsed by the Asia Pacific Heart Rhythm Society. *Europace* 17, pp. 1341-1353; doi:10.1093/europace/euv015
[12] Mladenic, D. 2006. Feature Selection for Dimensionality Reduction. In: *Subspace, Latent Structure and Feature Selection. Lecture Notes in Computer Science* 3940. Springer, Berlin, Heidelberg, pp. 84-102.
[13] Nandi, D., Ashour, A.S., Samanta, S., Chakraborty, S., Salem, M.A.M., Dey, N. 2015. Principal component analysis in medical image processing: a study. *International Journal of Image Mining* 1(1), pp. 65-86; doi:10.1504/IJIM.2015.070024
[14] Dey, N., Ashour, A.S., Borra, S. (Eds.) 2017. *Classification in BioApps: Automation of Decision Making*. Lecture Notes in Computational Vision and Biomechanics Vol. 26. Springer; doi: 10.1007/978-3-319-65981-7
[15] Kamal, M.S., Dey, N., Ashour, A.S. 2017. Large Scale Medical Data Mining for Accurate Diagnosis: A Blueprint. In: *Handbook of Large-Scale Distributed Computing in Smart Healthcare*, pp. 157-176. Springer, Cham; doi: 10.1007/978-3-319-58280-1_7
[16] Rangayyan, R.M. 2015. Pattern Classification and Diagnostic Decision. In: *Biomedical Signal Analysis*. John Wiley & Sons, Inc. pp. 571-632; doi:10.1002/9781119068129.ch9
[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.* 11(1), pp. 10-18; doi:10.1145/1656274.1656278
[18] Bezerra, V.L., Leal, L.B., Lemos, M.V., Carvalho, C.G., Filho, J.B., Agoulmine, N. 2013. A pervasive energy-efficient ECG monitoring approach for detecting abnormal cardiac situations. *IEEE 15th Int. Conf. on e-Health Networking, Applications and Services (Healthcom 2013)*, Lisbon, Portugal, pp. 340-345.
[19] Jovic, A., Kukolja, D., Jozic, K., Horvat, M. 2016. A Web Platform for Analysis of Multivariate Heterogeneous Biomedical Time-Series - a Preliminary Report. *Proc 23rd Int Conf on Systems, Signals and Image Processing (IWSSIP 2016)*, Bratislava, Slovakia, p. 70.
[20] Friganovic, K., Jovic, A., Jozic, K., Kukolja, D., Cifrek, M. 2017. MULTISAB project: a web platform based on specialized frameworks for heterogeneous biomedical time series analysis - an architectural overview. *Proc. Int. Conf. on Med & Biol Eng (CMBEBiH 2017)*, Sarajevo, Bosnia and Herzegovina, Springer Nature, pp. 9-15.
[21] Jovic, A., Kukolja, D., Friganovic, K., Jozic, K., Car, S. 2017. Biomedical Time Series Preprocessing and Expert-System Based Feature Extraction in MULTISAB Platform. *Proc. Int. Conf. MIPRO 2017*, Opatija, Croatia, pp. 349-354.
[22] Chen, C.M. 2011. Web-based remote human pulse monitoring system with intelligent data analysis for home health care. *Expert Syst Appl* 38(3), pp. 2011-2019.
[23] Zunic, E., Djedovic, A., Boskovic, D. 2016. Web-based and mobile system for training and improving in the field of electrocardiogram (ECG). *5th Mediterranean Conference on Embedded Computing (MECO 2016)*, Bar, Montenegro, pp. 441-445.
[24] Hsieh, J.-C., Hsu, M.-W. 2012. A cloud computing based 12-lead ECG telemedicine service. *BMC Med Inform Decis Mak.* 12, p. 77; doi:10.1186/1472-6947-12-77
[25] Jovic, A., Bogunovic, N., Cupic, M. 2013. Extension and Detailed Overview of the HRVFrame Framework for Heart Rate Variability Analysis. *Proc. Int. Conf. Eurocon 2013*, IEEE Press, Zagreb, Croatia, pp. 1757-1763.
[26] Bao, F.S., Liu, X., Zhang, C. 2011. PyEEG: An open source Python module for EEG/MEG feature extraction. *Comput. Intell. Neurosci.* 2011, p. 406391; doi:10.1155/2011/406391
[27] Moraru, L., Moldovanu, S., Dimitrievici, L.T., Shi, F., Ashour, A.S., Dey, N. 2017. Quantitative Diffusion Tensor Magnetic Resonance Imaging Signal Characteristics in the Human Brain: A Hemispheres Analysis. *IEEE Sensors Journal* 17(15), pp. 4886-4893; doi: 10.1109/JSEN.2017.2714701
[28] Wang, D., Li, Z., Cao, L., et al. 2017. Image fusion incorporating parameter estimation optimized Gaussian mixture model and fuzzy weighted evaluation system: A case study in time-series plantar pressure data set. *IEEE Sensors Journal* 17(5), pp. 1407-1420; doi: 10.1109/JSEN.2016.2641501

[29] Chakraborty, S., Chatterjee, S., Ashour, A.S., Mali, K., Dey, N. 2017. Intelligent Computing in Medical Imaging: A Study. In: *Advancements in Applied Metaheuristic Computing*, p. 143; doi:10.4018/978-1-5225-4151-6.ch006
[30] Dey, N., Ashour, A.S. 2018. Computing in Medical Image Analysis. In: Soft Computing Based Medical Image Analysis, pp. 3-11.
[31] Elhayatmy, G., Dey, N., Ashour, A.S. 2018. Internet of Things Based Wireless Body Area Network in Healthcare. In: *Internet of Things and Big Data Analytics Toward Next-Generation Intelligence*, pp. 3-20. Springer, Cham.
[32] Dey, N., Hassanien, A.E., Bhatt, C., Ashour, A., Satapathy, S.C. (Eds.). 2018. Internet of Things and Big Data Analytics Toward Next-Generation Intelligence. Studies in Big Data Vol. 30. Springer. doi: 10.1007/978-3-319-60435-0
[33] Banaee, H., Ahmed, M.U., Loutfi, A. 2013. Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges. *Sensors* 13(12), pp. 17472-17500; doi:10.3390/s131217472
[34] Kaur, P.D., Chana, I. 2014. Cloud based intelligent system for delivering health care as a service. *Comput Methods Programs Biomed* 113(1), pp. 346-359; 10.1016/j.cmpb.2013.09.013
[35] Xtelligent Media, LLC. 2018. *Top 10 Remote Patient Monitoring Companies for Hospitals*. https://mhealthintelligence.com/news/top-10-remote-patient-monitoring-solutions-for-hospitals (accessed: 2018-01-07)
[36] Lessard, Y., Sinteff, J.P., Siregar, P., et al. 2009. An ECG analysis interactive training system for understanding arrhythmias. *Stud Health Technol Inform.* 150, pp. 931-935.
[37] Ross, M.K., Wei, W., Ohno-Machado, L. 2014. “Big Data” and the Electronic Health Record. *Yearb Med Inform.* 9(1), pp. 97-104; doi:10.15265/IY-2014-0003
[38] Murdoch, T.B., Detsky, A.S. 2013. The Inevitable Application of Big Data to Health Care. *JAMA* 309(13), pp. 1351-1352; doi:10.1001/jama.2013.393
[39] Hsu, C.-L., Wang, J.-K., Lu, P.-C., Huang, H.-C., Juan, H.-F. 2017. DynaPho: a web platform for inferring the dynamics of time-series phosphoproteomics. *Bioinformatics* 33:12, pp. 3664-3666; doi:10.1093/bioinformatics/btx443
[40] Gaggioli, A., Cipresso, P., Serino, S., et al. 2014. A decision support system for real-time stress detection during virtual reality exposure. *Stud Health Technol Inform* 196, pp. 114-120.
[41] National Sleep Research Resource. 2018. https://sleepdata.org/ (accessed: 2018-01-07)
[42] University of Rochester Medical Center. 2018. *Telemetric and Holter ECG Warehouse (THEW)*. http://thew-project.org/ (accessed 2018-01-07)
[43] University of Michigan in Ann Arbor. 2018. *Ann Arbor Electrogram Libraries*. http://electrogram.com/ (accessed 2018-01-07)
[44] Gardasevic, G., Fotouhi, H., Tomasic, I., Vahabi, M., Björkman, M., Linden, M. 2017. A Heterogeneous IoT-based Architecture for Remote Monitoring of Physiological and Environmental Parameters. *4th EAI Int. Conf. on IoT Technologies for HealthCare (HealthyIoT 2017)*, Angers, France, p. 4869.
[45] Guan, K., Shao, M., Wu, S. 2017. A Remote Health Monitoring System for the Elderly Based on Smart Home Gateway. *Journal of Healthcare Engineering* 2017, p. 5843504; doi:10.1155/2017/5843504
[46] Dey, N., Ashour, A.S., Shi, F., Fong, S.J., Sherratt, R.S. 2017. Developing residential wireless sensor networks for ECG healthcare monitoring. *IEEE Transactions on Consumer Electronics* 63(4), pp. 442-449; 10.1109/TCE.2017.015063
[47] Granulo, E., Becar, L., Gurbeta, L., Badnjevic, A. 2016. Telemetry System for Diagnosis of Asthma and Chronical Obstructive Pulmonary Disease (COPD). *3rd EAI Int. Conf. IoT Technologies for HealthCare (HealthyIoT 2016)*, Vasteras, Sweden, pp 113-118.
[48] Matallah, H., Belalem, G., Bouamrane, K. 2017. Towards a New Model of Storage and Access to Data in Big Data and Cloud Computing. *International Journal of Ambient Computing and Intelligence (IJACI)*, 8(4), pp.1-14; doi:10.4018/IJACI.2017100103
[49] Tomasic, I., Petrovic, N., Fotouhi, H., Linden, M., Bjorkman, M. 2017. Relational Database to a Web interface in Real Time. *Eur Med Biol Eng Conf & Nordic-Baltic Conf on Biomed Eng & Med Physics (EMBEC & NBC 2017)*, Tampere, Finnland, pp. 89-92; doi:10.1007/978-981-10-5122-7_23
[50] Salem, O., Guerassimov, A., Mehaoua, A., Marcus, A., Furht, B. 2013. Sensor fault and patient anomaly detection and classification in medical wireless sensor networks. *IEEE Int Conf on Communications (ICC 2013)*, Budapest, Hungary, pp. 4373-4378; doi: 10.1109/ICC.2013.6655254
[51] Ivascu, T., Aritoni, O. 2015. Real-time health status monitoring system based on a fuzzy agent model. *E-Health and Bioengineering Conference (EHB 2015)*, Iasi, Romania, pp. 1-4; doi: 10.1109/EHB.2015.7391502
[52] Chatterjee, S., Dutta, K., Xie, H.Q., Byun, J., Pottathil, A., Moore, M. 2013. Persuasive and Pervasive Sensing: A New Frontier to Monitor, Track and Assist Older Adults Suffering from Type-2 Diabetes. *Proc. 46th Hawaii Int. Conf. on System Sciences*, Grand Wailea, HW, USA, pp. 2636-2645, 2013.

[53] Nangalia, V., Prytherch, D.R., Smith, G.B. 2010. Health technology assessment review: remote monitoring of vital signs--current status and future challenges. *Crit Care* 14(5), p. 233; doi:10.1186/cc9208
[54] Villarrubia, G., Bajo, J., De Paz, J.F., Corchado, J.M. 2014. Monitoring and Detection Platform to Prevent Anomalous Situations in Home Care. *Sensors* 14(6), pp. 9900-9921; doi:10.3390/s140609900
[55] Mainetti, L., Patrono, L., Secco, A., Sergi, I. 2016. An IoT-aware AAL system for elderly people. *Int. Multidisc. Conf. on Computer and Energy Science (SpliTech 2016),* Split, Croatia, pp. 1-6; doi: 10.1109/SpliTech.2016.7555929
[56] Lin, S.-S., Hung, M.-H., Tsai, C.-L., Chou, L.-P. 2012. Development of an Ease-of-Use Remote Healthcare System Architecture Using RFID and Networking Technologies. *J Med Syst* 36(6), pp. 3605-3619; doi:10.1007/s10916-012-9836-0
[57] Tsujimura, S., Shiraishi, N., Saito, et al. 2009. Design and Implementation of Web-Based Healthcare Management System for Home Healthcare. *13th Int Conf Biomed Eng*, IFMBE Proceedings, vol 23. Springer, Berlin, Heidelberg, pp. 1098-1101; doi: 10.1007/978-3-540-92841-6_270
[58] Secerbegovic, A., Suljanovic, N., Nurkic, M., Mujcic, A. 2015. The Usage of Smartphones in Remote ECG Monitoring Scenarios. *6th Eur. Conf. Int. Federation for Med & Biol Eng (MBEC 2014),* IFMBE vol. 45, Dubrovnik, Croatia, Springer, pp. 666-669.
[59] Zapata, B.C., Fernandez-Aleman, J.L., Idri, A., Toval, A. 2015. Empirical Studies on Usability of mHealth Apps: A Systematic Literature Review. *J Med Systems* 39(2), p. 1; doi: 10.1007/s10916-014-0182-2
[60] Fong, E.-M., Chung, W.-Y. 2013. Mobile Cloud-Computing-Based Healthcare Service by Noncontact ECG Monitoring. *Sensors*, 13(12), pp. 16451-16473; doi:10.3390/s131216451
[61] Huang, Q., Huang, X., Liu, L., Lin, Y., Long, X., Li, X. 2018. A case-oriented web-based training system for breast cancer diagnosis. *Comput Methods Programs Biomed* 156, pp. 73-83; doi:10.1016/j.cmpb.2017.12.028
[62] Qualter, J., Sculli, F., Oliker, A., et al. 2012. The biodigital human: a web-based 3D platform for medical visualization and education. *Stud Health Technol Inform* 173, pp. 359-361.
[63] Hackett, M., Proctor, M. 2016. Three-Dimensional Display Technologies for Anatomical Education: A Literature Review. *Journal of Science Education and Technology* 25(4), pp. 641-654.
[64] Porras, L., Drezner, J., Dotson, A., et al. 2016. Novice interpretation of screening electrocardiograms and impact of online training. *Journal of Electrocardiology* 49(3), pp. 462-466; doi:10.1016/j.jelectrocard.2016.02.004
[65] Hilbert, M. 2016. Big Data for Development: A Review of Promises and Challenges. *Development Policy Review* 34(1), pp. 135-174; doi:10.1111/dpr.12142
[66] Dey, N., Ashour, A.S., Shi, F., Balas, V.E. 2018. *Soft Computing Based Medical Image Analysis*. Academic Press. ISBN:978-0-12-813087-2
[67] van Poelgeest, R., van Groningen, J.T., Daniels, J.H., et al. 2017. Level of Digitization in Dutch Hospitals and the Lengths of Stay of Patients with Colorectal Cancer. *J Med Syst* 41(5), p. 84, 2017; doi: 10.1007/s10916-017-0734-3
[68] Rubio, O.J., Alesanco, A., Garcia, J. 2013. Secure information embedding into 1D biomedical signals based on SPIHT. *J Biomed Inform* 46(4), pp. 653-664; doi:10.1016/j.jbi.2013.05.002
[69] Hsieh, J.-C., Li, A.-H., Yang, C.-C. 2013. Mobile, Cloud, and Big Data Computing: Contributions, Challenges, and New Directions in Telecardiology. *Int. J. Environ. Res. Public Health* 10(11), pp. 6131-6153; doi:10.3390/ijerph10116131
[70] McCarthy, L.H., Longhurst, C.A., Hahn, J.S. 2015. Special requirements for electronic medical records in neurology. *Neurol Clin Pract.* 5(1), pp. 67-73; doi:10.1212/CPJ.0000000000000093
[71] Karaa, W.B.A., Dey, N. 2015. *Biomedical Image Analysis and Mining Techniques for Improved Health Outcomes*. IGI Global, Hershey, PA, USA, ISBN:1466688114.
[72] Griebel, L., Prokosch, H.-U., Kopcke F., et al. 2015. A scoping review of cloud computing in healthcare. *BMC Med Inform Decis Mak* 15, p. 17; doi:10.1186/s12911-015-0145-7
[73] Kruse, C.S., Goswamy, R., Raval, Y., Marawi, S. 2016. Challenges and Opportunities of Big Data in Health Care: A Systematic Review. *JMIR Med Inform* 4(4), p. e38; doi:10.2196/medinform.5359
[74] Alyami, M.A., Almotairi, M., Aikins, L., Yataco, A.R., Song, Y.-T. 2017. Managing personal health records using meta-data and cloud storage. *IEEE/ACIS 16th Int. Conf. on Computer and Information Science (ICIS 2017),* Wuhan, China, pp. 265-271; doi:10.1109/ICIS.2017.7960004
[75] Rifi, N., Rachkidi, E., Agoulmine, N., Taher, N.C. 2017. Towards using blockchain technology for eHealth data access management. *4th Int. Conf. on Advances in Biomed Eng (ICABME 2017),* Beirut, Lebanon, pp. 1-4; doi:10.1109/ICABME.2017.8167555
[76] Azaria, A., Ekblaw, A., Vieira, T., Lippman, A. 2016. MedRec: Using Blockchain for Medical Data Access and Permission Management. *2nd Int. Conf. on Open and Big Data (OBD 2016),* Vienna, Austria, pp. 25-30; doi: 10.1109/OBD.2016.11

[77] Mitchell, J., Probst, J., Brock-Martin, A., Bennett, K., Glover, S., Hardin, J. 2014. Association between clinical decision support system use and rural quality disparities in the treatment of pneumonia. *J Rural Health* 30(2), pp. 186-195. doi: 10.1111/jrh.12043
[78] Miller, P., Phipps, M., Chatterjee, S., et al. 2014. Exploring a Clinically Friendly Web-Based Approach to Clinical Decision Support Linked to the Electronic Health Record: Design Philosophy, Prototype Implementation, and Framework for Assessment. *JMIR Med Inform* 2(2), e20; doi:10.2196/medinform.3586
[79] Cloughley, R.G., Bond, R.R., Finlay, D.D., Guldenring, D., McLaughlin, J. 2016. An interactive clinician-friendly query builder for decision support during ECG interpretation. *Comput Cardiol Conf (CinC 2016)*, Vancouver, BC, Canada, pp. 381-384; doi:10.23919/CIC.2016.7868759
[80] Kell, S. 2016. Dynamically Diagnosing Type Errors in Unsafe Code, *Splash OOPSLA 2016 Conf.*, Amsterdam, Netherlands, pp. 800-819; doi: 10.1145/2983990.2983998
[81] McConnell, S. 2004. *Code Complete*. 2nd ed. Microsoft Press. ISBN: 0735619670
[82] Dierks, T., Rescorla, E. 2008. The Transport Layer Security (TLS) Protocol, Version 1.2. Network Working Group Request for Comments 5246. https://tools.ietf.org/html/rfc5246 (accessed: 2018-02-20)
[83] Prokhorenko, V., Choo, K.-K.R., Ashman, H. 2016. Web application protection techniques: A taxonomy. *Journal of Network and Computer Applications* 60, pp. 95-112, doi:10.1016/j.jnca.2015.11.017
[84] Huang, X.-W., Hsieh, C.-Y., Wu, C.-H., Cheng, Y.-C. 2015. A Token-Based User Authentication Mechanism for Data Exchange in RESTful API, *18th Int. Conf. on Network-Based Information Systems (NBiS 2015)*, Taipei, Taiwan, pp. 601-606; 10.1109/NBiS.2015.89
[85] Jones, M., Bradley, J., Sakimura, N. 2015. *JSON Web Token (JWT)*. Internet Engineering Task Force. Request for Comments 7519. https://tools.ietf.org/html/rfc7519
[86] Shen, Q., Yang, Y., Wu, Z., Wang, D., Long, M. 2013. Securing data services: a security architecture design for private storage cloud based on HDFS. *International Journal of Grid and Utility Computing* 4(4), pp. 242-254; doi:10.1504/IJGUC.2013.057118
[87] Graz University of Technology. 2018. *Meltdown and Spectre*. https://meltdownattack.com (accessed: 2018-01-21)
[88] Gkoulalas-Divanis, A., Loukides, G. 2013. *Anonymization of Electronic Medical Records to Support Clinical Analysis*. SpringerBriefs in Electrical and Computer Engineering. doi:10.1007/978-1-4614-5668-1_2
[89] Emam, K.E. Rodgers, S., Malin, B. 2015. Anonymising and sharing individual patient data. *BMJ* 350, h1139; doi:10.1136/bmj.h1139
[90] Fernandez-Aleman, J.L, Senor, I.C., Lozoya, P.A.O., Toval, A. 2013. Security and privacy in electronic health records: A systematic literature review, *J Biomed Inform* 46(3), pp. 541-562; doi:10.1016/j.jbi.2012.12.003
[91] Google. 2018. *Angular*. https://angular.io/ (accessed: 2018-01-21)
[92] Belshe, M., Peon, R., Thomson, M. 2015. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Internet Engineering Task Force (IETF). RFC 7540. http://httpwg.org/specs/rfc7540.html (accessed: 2018-01-19)
[93] Fielding, R.T. 2009. *It is okay to use POST*. Untangled, Blog. http://roy.gbiv.com/untangled/2009/it-is-okay-to-use-post (accessed 2018-01-19)
[94] Orchard, D. 2016. *Programming language evolution and sustainable software*. Blog. The Software Sustainability Institute, UK, https://www.software.ac.uk/blog/2016-09-12-programming-language-evolution-and-sustainable-software (accessed: 2018-02-20)
[95] Oracle. *Java Platform, Standard Edition What's New in Oracle JDK 9*. https://docs.oracle.com/javase/9/whatsnew/toc.htm (accessed: 2018-01-19)
[96] Ben-Kiki, O., Evans, C., Net, I.D. 2009. *YAML Ain't Markup Language (YAML™) Version 1.2*. http://www.yaml.org/spec/1.2/spec.html (accessed: 2018-01-20)
[97] SmartBear Software. 2018. *Swagger Editor and Swagger UI*. https://swagger.io/ (accessed: 2018-01-20)
[98] Krishnan, D., Kelly, A. 2018. *Springfox Reference Documentation*. Available at: https://springfox.github.io/springfox/docs/current/ (accessed: 2018-01-20)
[99] Pivotal Software, Inc. 2018. *Spring Boot*. https://projects.spring.io/spring-boot/ (accessed: 2018-01-20)
[100] H2 database. 2018. http://www.h2database.com/html/features.html (accessed: 2018-01-20)
[101] Heaton, J. 2015. Encog: Library of Interchangeable Machine Learning Models for Java and C#. *JMLR* 16, pp. 1243-1247.
[102] Chang, C.-C., Lin, C.-J. 2011. LIBSVM : a library for support vector machines. *ACM TIST* 2(27), pp. 1-27.
[103] Tibco Software, Inc. 2018. *JasperReports® Library, Open Source Java Reporting Library*. https://community.jaspersoft.com/project/jasperreports-library (accessed: 2018-01-21).
[104] Witten, I.H., Frank, E., Hall, M.A. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd ed. Morgan Kaufmann. ISBN:0123814790
[105] Syncleus. 2018. *Aparapi*. http://aparapi.com, (accessed: 2018-01-21)
[106] Majumder, S., Mondal, T., Deen, M.J. 2017. Wearable Sensors for Remote Health Monitoring. *Sensors* 17(1), p. 130; doi:10.3390/s17010130

[107] Chapman, B., Jost, G., van der Pas, R. 2007. *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press; Scientific and Engineering edition, Boston, MA, USA, ISBN:0262533022.
[108] Kalin, M. 2013. *Java Web Services: Up and Running: A Quick, Practical, and Thorough Introduction*. 2nd ed., O'Reilly Media, Sebastopol, CA, USA, ISBN:1449365116
[109] Li, W.-S., Yan, J., Yan, Y., Zhang, J. 2010. Xbase: cloud-enabled information appliance for healthcare. *Proc 13th Int Conf on Extending Database Technology (EDBT '10),* ACM, New York, NY, USA, pp. 675-680. doi:10.1145/1739041.1739125
[110] Vukicevic, M., Radovanovic, S., Milovanovic, M., Minovic, M. 2014. Cloud Based Metalearning System for Predictive Modeling of Biomedical Data. *The Scientific World Journal* vol. 2014, p. 859279; doi:10.1155/2014/859279.
[111] Sahoo, S.S., Jayapandian, C., Garg, G., Kaffashi, F., Chung, S., Bozorgi, A., Chen, C.-H., Loparo, K., Lhatoo, S.-D., Zhang G.-Q. 2014. Heart beats in the cloud: distributed analysis of electrophysiological 'Big Data' using cloud computing for epilepsy clinical research. *JAMIA* 21(2), pp. 263-271, 2014; doi:10.1136/amiajnl-2013-002156
[112] Jovic A., Kukolja, D., Jozic, K., Cifrek, M. 2016. Use Case Diagram Based Scenarios Design for a Biomedical Time-Series Analysis Web Platform *Proc. Int. Conf. MIPRO 2016*. Opatija, Croatia, pp. 326-331.
[113] Brett, A., Croucher, M., Haines, R., Hettrick, S., Hetherington, J., Stillwell M., Wyatt, C. 2017. *Research Software Engineers: State of the Nation Report 2017*. University of Southampton, UK: The Research Software Engineer Network (RSEN); doi:10.5281/zenodo.495360
[114] Grieve, S., Mueller, E., Morley, A., Upson, M., Adams, R., Clerx, M. 2018. *Bridging the gap: Convincing researchers with different backgrounds to adopt good (enough) software development practices*. Blog. The Software Sustainability Institute, UK, https://software.ac.uk/blog/2018-02-09-bridging-gap-convincing-researchers-different-backgrounds-adopt-good-enough (accessed: 2018-02-20)
[115] Cairns, C., Somerfield, D. 2017. *The Basics of Web Application Security*, MartinFowler.com blog, https://martinfowler.com/articles/web-security-basics.html (accessed: 2018-02-20)
[116] Healey, J.A., Picard, R.W. 2005. Detecting stress during real-world driving tasks using physiological sensors. *IEEE TITS* 6(2), pp. 156-166; doi:10.1109/TITS.2005.848368.
[117] Chudacek, V., Spilka, J., Bursa, M., Janku, P., Hruban, L., Huptych, M., Lhotska, L. 2014. Open access intrapartum CTG database. *BMC Pregnancy and Childbirth* 14, p. 16.

Relating the chapter title with the scope and title of the book

Here, we consider the relation between the title of the book: "Medical Big Data and Internet of Medical Things: Advances, Challenges, and Applications" and its scope with the title of our proposed chapter: "Challenges in Designing Software Architectures for Web Based Biomedical Signal Analysis".

The first word of our chapter involves architectural design "Challenges", which is also mentioned in the title of the book. Next, the scope of the book in "Part A: IoT in Life Sciences" involves the topic "Medical Big Data Management Systems and Infrastructures". This is reflected in the title of our chapter, because we focus on "designing software architectures" in biomedical engineering, including big data applications.

Our chapter deals with telemedicine and health care based on web solutions in signal analyis that includes recommendations to researchers, and the scope of the book mentions under "Part B: Telemedicine and Health care" the topic "Recommender Systems and Decision Support Systems".

Although not explicitly mentioned in the title, the scope "Part C: Medical Big Data Mining and Processing" with the topic "Pattern Recognition, Features Extraction, Feature Reduction and Selection Techniques in Biomedical Applications" is greatly reflected in our title, as the whole topic forms part of the complex biomedical signal analysis scenario, namely, the extraction of features, its reduction and selection are all part of the signal analysis process. This is also shown and discussed in the text of the chapter.

In the scope "Part D: Case studies for Classification in Medical Problems", the topic "Privacy and Security Issues in Big Data" is mentioned, which is reflected in the design of software architecture that we discuss in detail in the manuscript and mention in the title of the chapter.

To conclude, apparently, the title of our chapter encompasses several parts of the scope of the book and thus we consider it suitable for publication in the current form.