Technical Report - Project specifications

Beeg Brain

universidade de aveiro

Course:	Projeto em Informática
Date:	Aveiro, 16 May 2022
Students:	98474: João Reis 98679: João Farias 98388: Ricardo Rodriguez 98390: Mariana Rosa
Project description:	This project will be a multi-tenant information system capable of receiving and visualizing EEG files from distinct proveniences, independently of the manufactures. It will provide tools to manage, visualize and create reports in a web environment. Also, this project will improve the communications between medical institutions.

Index

1. Introduction

Nowadays, many medical institutions have problems sharing electroencephalogram exams, because files can get lost very easily and different equipment creates compatibility issues. From time to time, these interoperability problems may lead to a second examination of the patient, which is time-consuming and costs money.

In addition, medical institutions often have their own kind of software to visualize EEG exams and there is a lack of a web platform capable of visualizing these exams.

Our main goal for the *Projeto em Informática* course is to develop a web platform where it is possible to visualize and manage EEG (electroencephalogram) exams, anytime, anywhere, in the cloud.

This project aims to solve these problems which affect the daily routine of medical institutions' employees who work in the EEG area field, by providing a simple and intuitive web platform that provides a monitoring page for analytic purposes, a workspace area to manage different exams and an EEG viewer which features a series of tools to help doctors having a better grasp of the exam. Besides that, doctors can write a report about the selected exam and generate a PDF file automatically.

2. Our team

BeegBrain is composed of 6 elements: two of them are the guiding teachers, the other ones are the developers. The following table (Table 1) will specify better the role and function of each one.

Name	Role	Function
Carlos Costa	Guiding Teacher	Advise and guide in all stages of the software development
Luís Bastião	Guiding Teacher	Advise and guide in all stages of the software development
João Reis	Team Manager/ Developer	Responsible for setting targets, defining weekly tasks, and assisting with any issues the team may have. Keep the team motivated and focused. Also, he has the role of a developer whose function is researching, designing and implementing the product software program.
João Farias	Software Architect/ Developer	His main function is to design software applications, plan the different features of the software program and integrate them into a functional system. Also, he has the role of a developer whose function is researching, designing and implementing the product software program.
Mariana Rosa	Product Owner/ Developer	Define the product backlog - a prioritized set of customer requirements. Also, she has the role of developer whose function is researching, designing and implementing the product software program.
Ricardo Rodriguez	DevOps Master/ Developer	Responsible for merging all the development and operations (deployment and integration) into a single and continuous set of processes. Also, he has the role of a developer whose function is researching, designing and implementing the product software program.

Table 1 - Team roles

3. Inception Phase

a. Product concept

i. Vision statement

An EEG is an exam that detects electrical activity in the brain using electrodes (small metal discs) with the main purpose of analysing brain activity. This activity shows up as wavy lines on an EEG recording (Fig. 1).

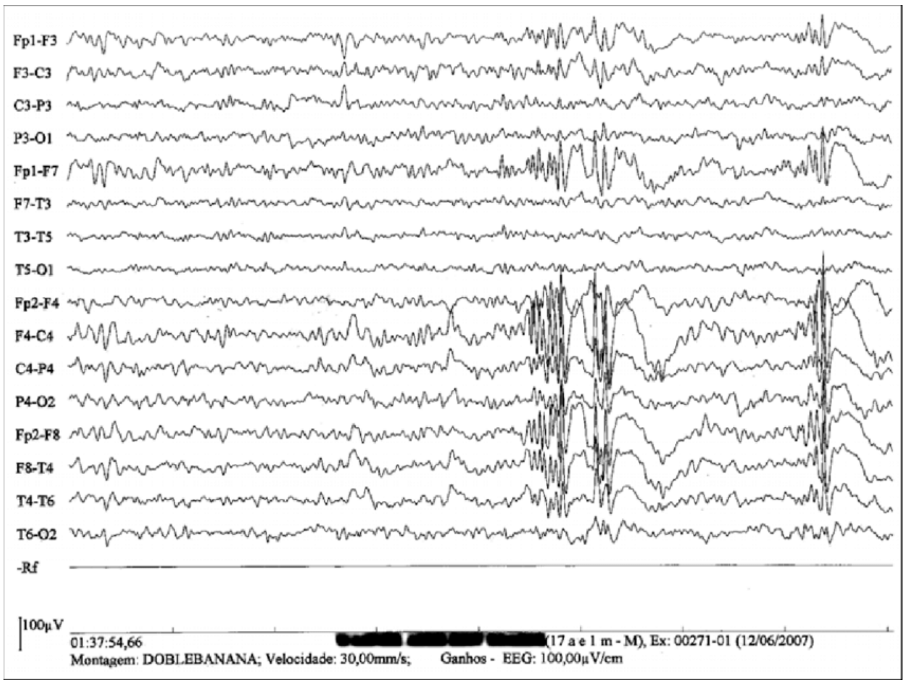

Fp1-F3
F3-C3
C3-P3
P3-O1
Fp1-F7
F7-T3
T3-T5
T5-O1
Fp2-F4
F4-C4
C4-P4
P4-O2
Fp2-F8
F8-T4
T4-T6
T6-O2
-Rf
100µV
01:37:54,66
[redacted](17 a e 1 m - M), Ex: 00271-01 (12/06/2007)
Montagem: DOBLEBANANA; Velocidade: 30,00mm/s; Ganhos - EEG: 100,00µV/cm

Fig. 1 - EEG example

However, these exams could have different formats, the main ones are **EDF**, **BDF**, **GDF** and several others. For this project, we will focus on the usage of EDF (European Data Format).

ii. Problem

Sometimes it is necessary to quickly visualize an exam (ECG, EEG, x-ray…), the process from loading the exam in the system until the exam is ready for visualization, takes too much time. Also, online platforms capable of doing this work are slow, because these files have a lot of data, the visualization is not efficient, and these platforms don't have too many tools to help in signal visualization.

Another existing problem in the real world is the precarious communication between medical institutions, since, in some cases, the institution where the EEG exams are taken are different from the institutions which analyze them. Additionally, data-sharing and remote diagnosis is a hard and time-consuming process where the exams can be lost.

iii. Goals

BeegBrain is a platform where all the data is read and written in the same format, so there is not any discrepancy between them. Our goal is to develop a multi-tenant system to analyse and visualize EEG from different patients. The idea is to have a continuous vision of the exam with the possibility of analysing it with intuitive tools and writing a report about them.

This project also aims to improve collaboration between medical institutions through contracts between them to make communication between them faster and more efficient.

We will develop a distributed system, capable of supporting multiple institutions and users, where they can not only upload or visualize exams depending on their role, but also write reports about the exam, and so on.

iv. Related Work and State-of-art

To understand a bit more about EEGs and what we were working with, a lot of research was done, and we found a website (https://www.learningeeg.com/) with a lot of information to rely on. We found the **channels'** concept, which we didn't know, as well as **EDF** files. We think these are the two main concepts in this area, and we are going to explain them:

- **Channel** corresponds to an electrode capturing brainwave activity. Typical EEG systems can have as few as a single channel to as many as 256 channels.

- **EDF (European Data Format)** is a simple and flexible format for exchange and storage of multichannel biological and physical signals. EDF files are composed by 2 headers:

1. The first one, contains 256 bytes long, and the following entries:

Bytes	Description
8	version of this data format, usually 0 for original EDF
80	patient identification
80	local recording identification
8	startdate of recording (dd.mm.yy)
8	starttime of recording (hh.mm.ss)
8	number of bytes in header record
44	not used in original EDF specification
8	number of data records
8	duration of a single data record in seconds
4	number of signals in data record

Table 2 - First header entries

2. The second header has 256 bytes for each signal, and the following structure:

Bytes	Description
16	label for the signal
80	transducer type
8	units
8	minimum possible value in units
8	maximum possible value in units
8	minimum value numerically
8	maximum value numerically
80	type of any prefiltering
8	number of samples in each data record
32	reserved

Table 3 - Second header entries

After understanding these two concepts, we carried on to the visualization module and we found an online viewer: EDF Viewer, that was our first “1st hands-on” in visualization of the signals.

For EEG file parsing and interpretation, there were some available Python libraries to work with EDF files, such as MNE and PyEdfLib, the latter being our preference for EEG reading.

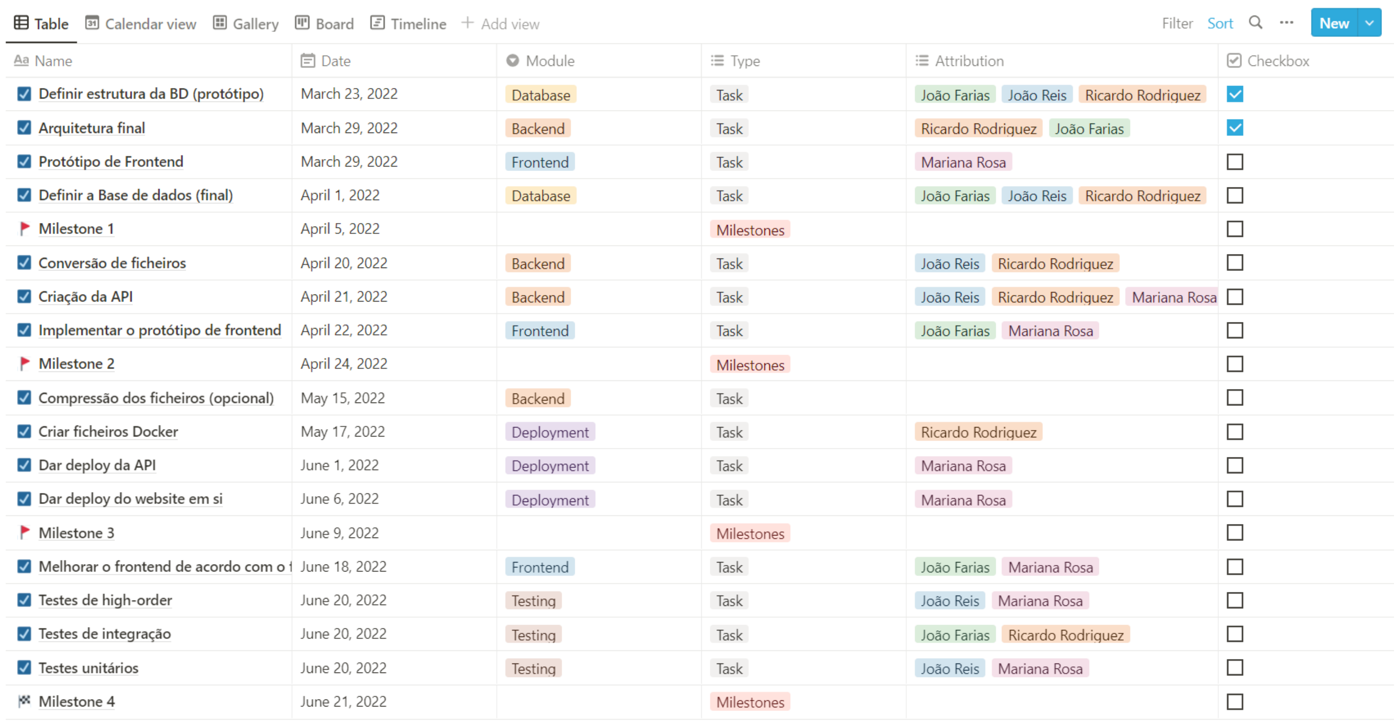
Additionally, there are many different pre-existing modules that can represent time series in Angular, like NgxCharts and ApexCharts, but ECharts was the chosen library.

Not in the same area, but in a close approach, our guiding teacher, Luís Bastião, developed a similar project to ours: Cardiobox, instead of EEG, it works with Echocardiograms (ECG) with the main functionality similar to ours: analysing and reporting those exams.

b. Workflow

i. Tasks and Project Calendar

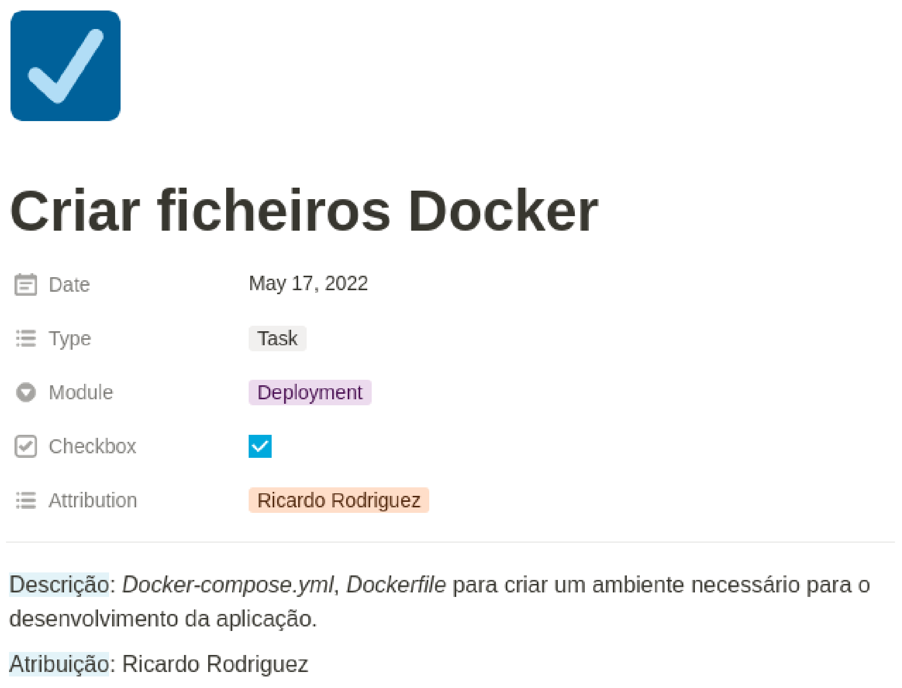
To build our Calendar (Fig 2) we used the platform Notion which allowed describing the tasks, the person/people responsible for them and also gives a clear vision of the dates:



Name	Date	Module	Type	Attribution	Checkbox
Definir estrutura da BD (protótipo)	March 23, 2022	Database	Task	João Farias, João Reis, Ricardo Rodriguez	✓
Arquitetura final	March 29, 2022	Backend	Task	Ricardo Rodriguez, João Farias	✓
Protótipo de Frontend	March 29, 2022	Frontend	Task	Mariana Rosa	☐
Definir a Base de dados (final)	April 1, 2022	Database	Task	João Farias, João Reis, Ricardo Rodriguez	☐
Milestone 1	April 5, 2022		Milestones		☐
Conversão de ficheiros	April 20, 2022	Backend	Task	João Reis, Ricardo Rodriguez	☐
Criação da API	April 21, 2022	Backend	Task	João Reis, Ricardo Rodriguez, Mariana Rosa	☐
Implementar o protótipo de frontend	April 22, 2022	Frontend	Task	João Farias, Mariana Rosa	☐
Milestone 2	April 24, 2022		Milestones		☐
Compressão dos ficheiros (opcional)	May 15, 2022	Backend	Task		☐
Criar ficheiros Docker	May 17, 2022	Deployment	Task	Ricardo Rodriguez	☐
Dar deploy da API	June 1, 2022	Deployment	Task	Mariana Rosa	☐
Dar deploy do website em si	June 6, 2022	Deployment	Task	Mariana Rosa	☐
Milestone 3	June 9, 2022		Milestones		☐
Melhorar o frontend de acordo com o f	June 18, 2022	Frontend	Task	João Farias, Mariana Rosa	☐
Testes de high-order	June 20, 2022	Testing	Task	João Reis, Mariana Rosa	☐
Testes de integração	June 20, 2022	Testing	Task	João Farias, Ricardo Rodriguez	☐
Testes unitários	June 20, 2022	Testing	Task	João Reis, Mariana Rosa	☐
Milestone 4	June 21, 2022		Milestones		☐

Fig. 2 - Project calendar

The project calendar was a difficult task to do, not because of the difficulty of "building" the calendar, but because we didn't know the task's duration, and some of them take more time than what we expected. We are going to explain this in **Changes in the initial plan section**. An example of a task description is available on Figure 3.



Criar ficheiros Docker

Date	May 17, 2022
Type	Task
Module	Deployment
Checkbox	✓
Attribution	Ricardo Rodriguez

Descrição: *Docker-compose.yml*, *Dockerfile* para criar um ambiente necessário para o desenvolvimento da aplicação.

Atribuição: Ricardo Rodriguez

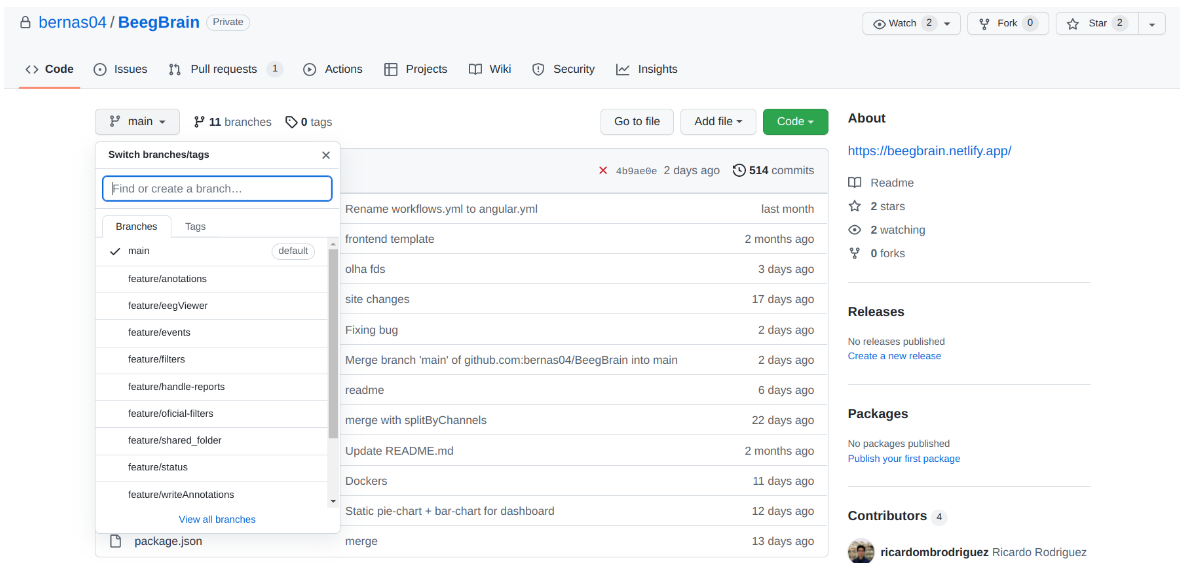
Fig. 3 - Task description

ii. Communication Plan and workflow

We chose the **GitHub** platform to store our code in a repository and we developed a *website* to store all the information related to our work, including reports, presentations and the project calendar.

Regarding the workflow, we used feature-branching and pull-requests. For every feature, a branch was created. We usually had sprints of one week, therefore at the end of that week, we merged the code into the main branch (Fig. 4), deleted the branches and created new ones for that sprint's tasks.

To keep in touch with our guiding teachers and what was happening we scheduled a weekly meeting, so we could ask for a review or advice. The group itself reunited twice a week to catch up on how the work was evolving and to merge the code.



bernas04 / BeegBrain Private

Watch 2 · Fork 0 · Star 2

Code · Issues · Pull requests 1 · Actions · Projects · Wiki · Security · Insights

main · 11 branches · 0 tags · Go to file · Add file · Code

Switch branches/tags

Find or create a branch…

Branches · Tags

- ✓ main (default)
- feature/anotations
- feature/eegViewer
- feature/events
- feature/filters
- feature/handle-reports
- feature/oficial-filters
- feature/shared_folder
- feature/status
- feature/writeAnnotations

View all branches

4b9ae0e 2 days ago · 514 commits

	Rename workflows.yml to angular.yml	last month
	frontend template	2 months ago
	olha fds	3 days ago
	site changes	17 days ago
	Fixing bug	2 days ago
	Merge branch 'main' of github.com:bernas04/BeegBrain into main	2 days ago
	readme	6 days ago
	merge with splitByChannels	22 days ago
	Update README.md	2 months ago
	Dockers	11 days ago
	Static pie-chart + bar-chart for dashboard	12 days ago
package.json	merge	13 days ago

About

https://beegbrain.netlify.app/

Readme · 2 stars · 2 watching · 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 4

ricardombrodriguez Ricardo Rodriguez

Fig. 4 - Github - Branches example

4. Elaboration Phase

a. Key requirements and constraints

For this project we did a lot of research and brainstorming between our team and we gathered up these requirements:

i. Functional Requirements

- The system should be a centralized cloud-based platform;
- The system must have a RESTful API, in order to upload files and do other important tasks;
- The application should have different registration pages for doctors and operators and a universal login page;
- A user should be able to monitorize all EEG exams he has access to, though a dashboard page;
- The dashboard page should be able to present important metrics, such as charts and a logging system that can identify, for example, how many EEG exams need reviewal;
- EEG exams can only be uploaded by operators;
- The user should also be able to upload files inside the application, with the help of an upload button or a drag and drop area, alongside with the identification of the patient and the priority given to that exam;
- Uploaded EEG exams can only be accessed by employees of the institutions which have an agreement between themselves;
- The file processing area will be divided in two workspaces: Transferred area (with the files that have been successfully transfered) and the Error area (with the ones that weren't able to parse or sent);
- Doctors can only access the transferred area, whereas operators can have access to both transferred and error area;
- In the Error area, it will be displayed not only the exams with parsing errors but also with other error types. The user should be notified of these errors to solve these conflicts manually.
- In the transferred area, users can have access to their institution's shared folder, containing all the EEG exams they have permission to open;
- The user should be able to filter the EEG exams in their shared folder through important parameters, such as the date, institution, patient and others;

- The files in the shared folder will be kept in a temporary cache for 7 days;
- The user should be able to visualize EEG exams, selecting the channels he wants to analyse, moving back and forth in time, extending the time interval of the chart, playing the exam in different speeds, pausing the exam, zooming in the signals, reading annotations and other important tools that help the interpretation and analysis of the EEG exam in an easy and intuitive manner;
- A doctor should be able to write, or edit, a report of the EEG exam and format it the way he wants it to be presented;
- The platform should be able to generate a PDF document with the content written by the doctor;
- The application should have a patients page, where institution employees can search information about a specific patient, including its exam history;
- Users can delete EEG exams included in the shared folder of their institution(s);

ii. Non-functional Requirements

- The application EEG viewer should have high performance;
- The system must be scalable, allowing the use of all application functionalities for various users at the same time, without sacrificing performance;
- EEG visualizer tools must be easy for the user to use and understand its functionalities;
- EEG files should be disassembled into different subparts and stored in the database, each one representing an EEG channel;
- EEG channel files should be compressed when uploaded to the database, in order to be space efficient;
- When the system fails during an EEG upload, the operator should be notified of that event;
- The system should have a token authentication and a permission mechanism, allowing authenticated users to only do the tasks they're allowed to do;
- The system must track events related to the life cycle of an EEG (e.g. upload, delete, PDF generation…) and associated it with an owner and a timestamp;
- EEG exams that reside more than 1 week in the database should be removed from the transferred area, although still persisting in the database;
- When generating a PDF, the user should be able to add/remove sections in the text editor and format them the way he likes;
- EEG information of the channels selected by the user shouldn't be sent in its entirety, but rather through a buffering technique;

b. Fears

While developing this project some fears come to us, some technical and some non-technical. In the non-technical field, at the beginning we were a little overwhelmed because we were immersed into an unknown field, the medicine world. So, at first understanding the minimum about EEGs and exam formats required some research. We contacted some neurologists to understand the basic requisites and important aspects associated to EEG's. We also would like to show our final version to neurologists and get some feedback about the viewer component and the report feature, if they think that would improve their work.

Regarding technical aspects, the EEG Visualizer worried us. For the viewer to be useful and helpful to doctors to analyse, it needs to plot a few or a lot of signals at the same time. Therefore, we had to be careful on how we were going to process the huge amount of data coming from the EEG exam. To better understand what we are talking about, one EEG with approximately 1000 seconds has more than 300000 values, so plotting all this data in a single graphic is very inefficient. We did that in an early version of our work, but we aborted this idea quickly. We wanted to have an efficient and quick viewer, able to plot a lot of different channels of the EEG exam at the same time in an intuitive way. At the same time, we wanted our application to be the most intuitive for doctors and operators to use in their daily tasks.

c. Risks

When working with a lot of private data (information about patients, exams) it's really important to have good security policies. Not just in the application itself, to prevent attacks from hackers, but also in the information that is shared between doctors and operators, they only have permission to access data from the doctor revision center or providence that they have a contract with.

d. Actors

In our project we have two types of actors: doctors and operators. Doctors are health professionals who need to analyze, study patients' cases and write reports regarding an EEG exam of one of his/hers patients. Operators are professionals who perform the exams on patients and only they can upload exams into the system.

e. Use Cases

i. Personas

Fig. 5 - Alice Torres

Alice Torres is a 33 year-old Portuguese neurologist working at the Infante D. Pedro hospital, in Aveiro. She's been married for five years with Vasco Regal and has a 2-year-old son. She's been fascinated with the study of the brain since she was a little child, interested in the diagnostician of brain diseases like multiple sclerosis, epilepsy and Alzheimer's, which affects her grandmother who doesn't recognize her.

Alice has a Ph.D. in neuroscience at the *Faculdade de Medicina de Lisboa*.

Fig. 6 - Rui Moniz

Rui Moniz is a 52 years-old man, living in Coimbra, United Kingdom, a historic county near London. He's been married since 2000 to Rebecca Jimenez, a very famous cardiologist, and together they had 3 children: Otis, Yasmin and Sienna. The medical world always was a thing that fascinated him, however, he didn't want to be a doctor because of the highly demanding work, therefore he chose to follow the career of technical operator. He has been working for 8 years at Coimbra's Hospital as a technical operator that has a contract with Infante D. Pedro Hospital.

Fig. 7 - Duarte Pereira

Duarte Pereira is a 61-year-old man, born in Coimbra. He's been married to Helena Ramos for 36 years and both of them have 2 kids: Hugo and Jéssica. He studied at *Faculdade de Medicina da Universidade de Coimbra*, where he graduated with a Doctorate Degree in Medicine.

Since he started working at *CHUC*, Coimbra's main hospital, he's been observing the evolution of neuroscience and started diverging into that area, where he currently works as a Lead Neuroscientist. He and his team are studying how the brain constructs dreams.

ii. Main scenarios and their tasks

- Alice wants to check an **EEG** exam while he is writing a report. So, she goes to the dashboard, chooses the exam in the application, sees it and she can start writing important information about what she saw.
    - Task 0: Alice wants to analyze and examine the exams that an operator did today;
    - Task 1: The exam is on the *Transferred Area;*
    - Task 2: She clicks on the Analysis button to see the exam;
    - Task 3: She writes the report while viewing the exam;

- Duarte has intentions of generating a PDF directly on the platform with all the important information gathered in that exam.
    - Task 0: The exam is on the *Transferred Area;*
    - Task 1: He clicks on the Analysis button to see the exam;
    - Task 2: He generates a report by clicking on the *Generate PDF* button.

- Rui wants to upload the exams that he made today so that the doctors can review them.
    - Task 0: Osmand wants to upload a file;
    - Task 1: He goes to the workspace and clicks on the upload button;
    - Task 2: He inserts the operator and patient's ID;
    - Task 3: He drags and drops the file;
    - Task 4: The exam will be in the Transferred Area;

- Duarte is a very experienced doctor and he has done thousands of exams in his whole life. Imagining that he has access to all of the exams of his institution, Duarte wants to search for a certain patient to study his case and see how he's going.
    - Task 0: Duarte wants to check information about one of his patients;
    - Task 1: Duarte goes to the *Patient*'s area;
    - Task 2: He searches for the patient's name;
    - Task 3: He sees his patient's exam history and study his case;

- Rui has 20 exams to upload for today's work. However, after uploading them all, he sees that one exam is on the "Error Area". Therefore, he needs to contact the BeegBrain support team, in order for the doctors in Aveiro's hospital to analyze it. He has to make sure that her exam is not in the error zone.
    - Task 0: Rui uploads the exams for the shared folder;
    - Task 1: Rui checks that the exam was not sent to the shared folder;
    - Task 2: Rui sees that has a notification on the error zone;
    - Task 3: Rui opens the error zone and see which exams have problems;
    - Task 4: He contact's BeegBrain support team for helping with that exam and waits for feedback;

iii. Use Case Diagram

With those scenarios, we gather which use cases exist for each one of the actors in our application presented in figure 8.

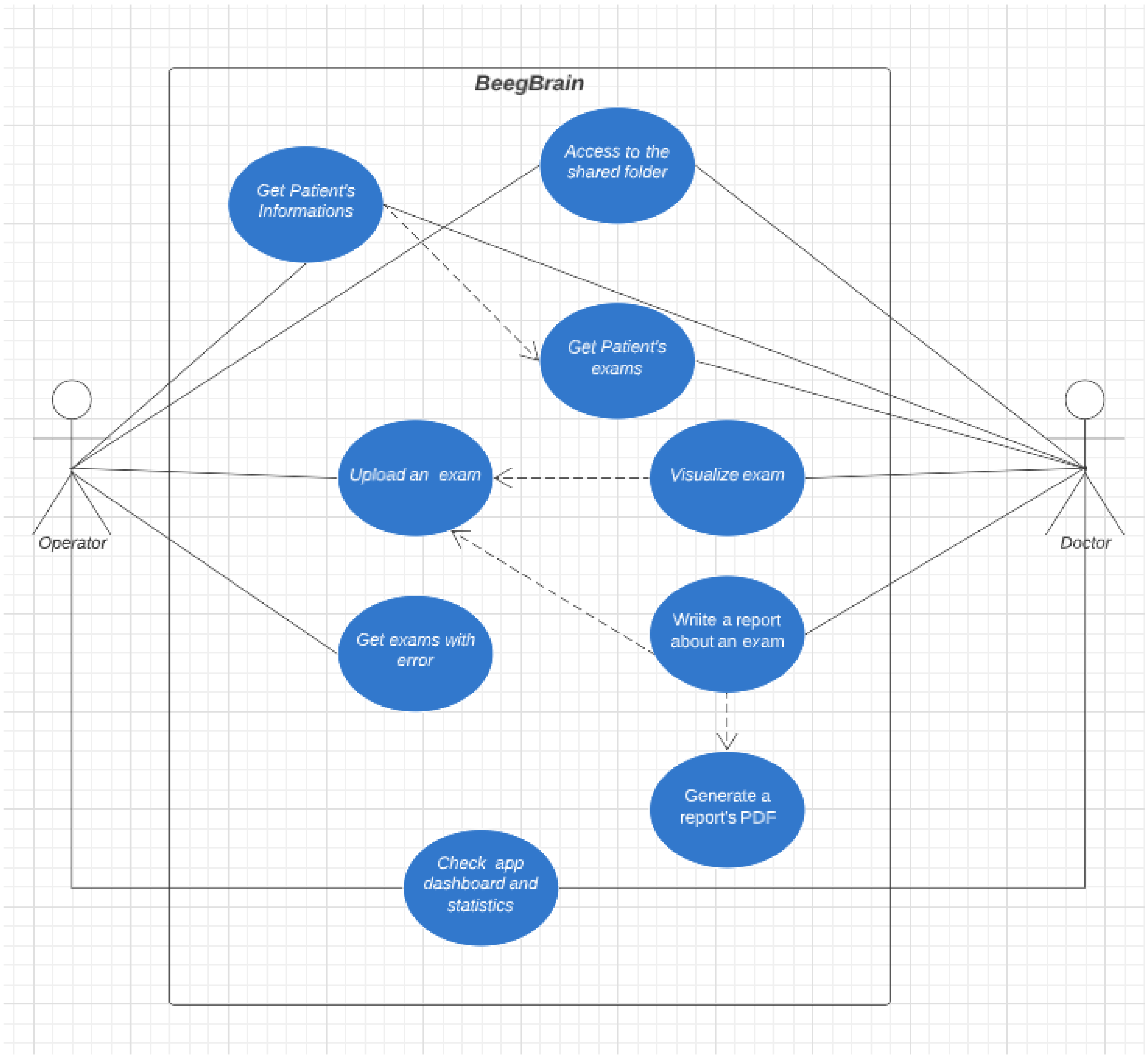

BeegBrain
Get Patient's Informations
Access to the shared folder
Get Patient's exams
Upload an exam
Visualize exam
Get exams with error
Write a report about an exam
Generate a report's PDF
Check app dashboard and statistics
Operator
Doctor

Fig. 8 - Use case diagram

f. Architecture notebook

i. Architectural view

Figure 9 describes our project's architecture. It will be developed with Angular Framework for the presentation, Django for the backend (including the construction of the API) and our database will be created with MySQL, which is supported by the Django framework.

The frontend module should be able to interact with the users by displaying the EEG data in the browser, allowing users to upload files in several ways and many other functionalities that require user input. These events should be registered in a logging system and sent to the backend module.

The backend module should receive API requests triggered by user events, process them and store/retrieve information from the database.

The database should persist all structured information critical for the system to work (e.g. entities and relations between them). This module is explained in detail in chapter 4.

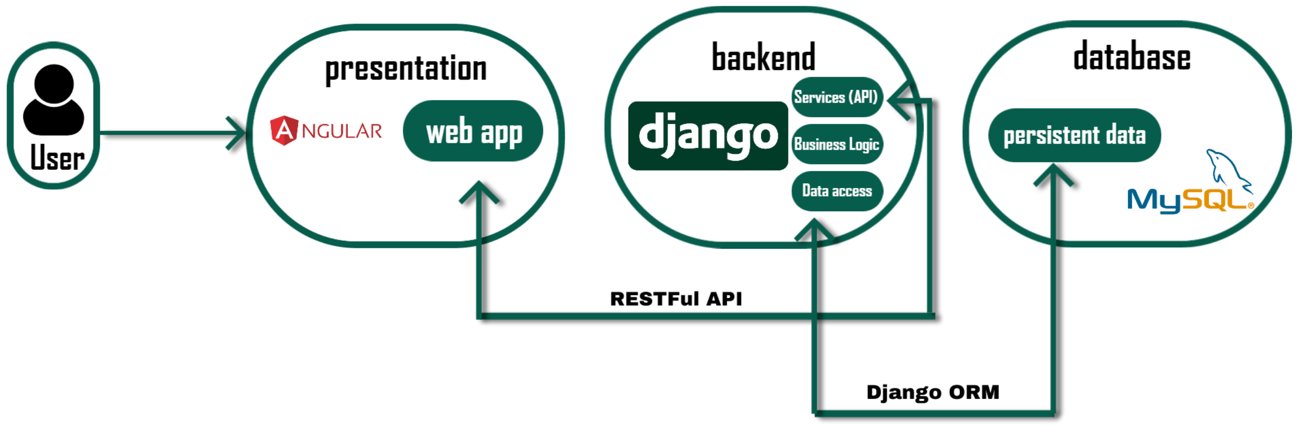


Fig. 9 - Project's architecture

ii. Deployment view

As we explained in the previous chapter, the doctor is going to access our application through a web page, developed in **Angular**, which is going to communicate with the backend, developed in **Django** and this one is going to store data in **MySQL** database. These two components will be deployed separately using **docker containers**. The backend module is where the system will communicate with the EEG files shared folder.

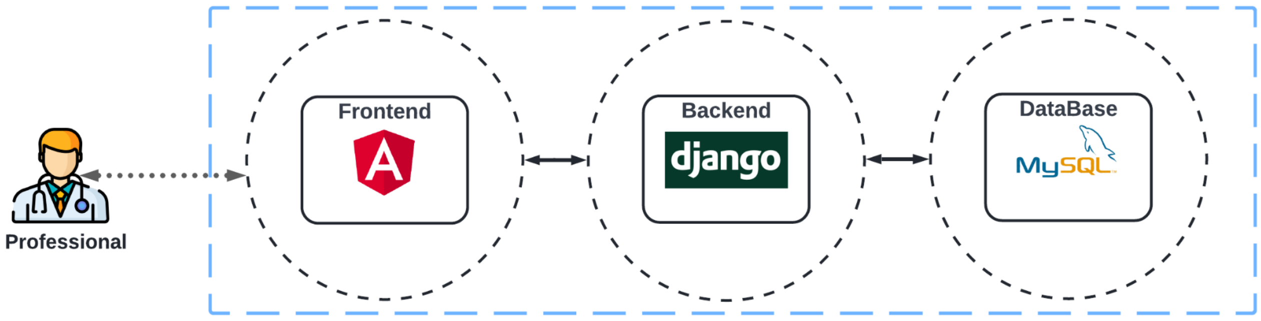


Fig.10 - Deployment diagram

g. Information model

This diagram represents the database of our application. On it we show our entities, the relationships between them and their fields.

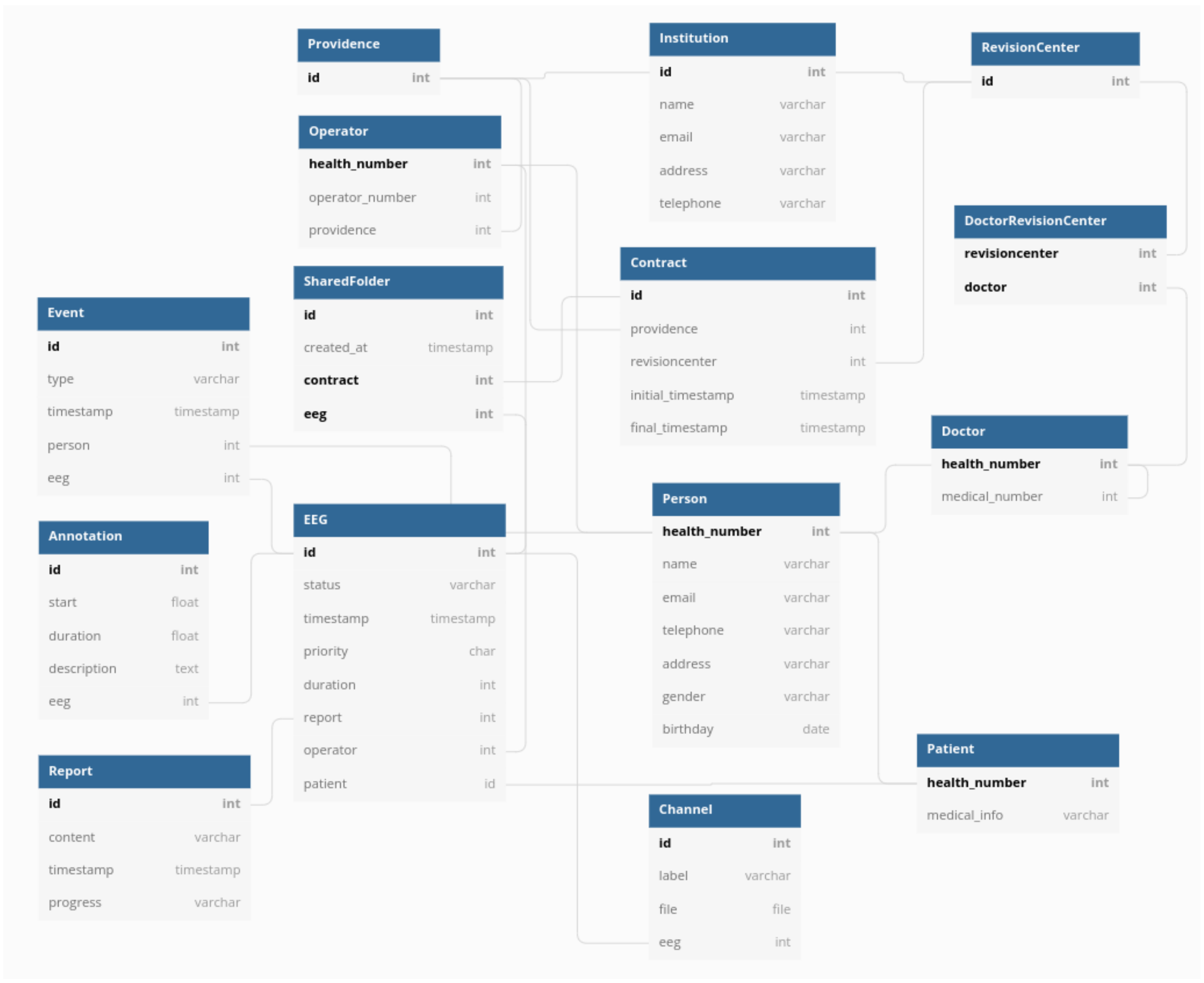


Fig. 11 - EER diagram

Institution

The institution entity represents a medical institution. It has basic attributes that identify them, such as the name, email, address and telephone. It can be divided into two different sub-entities: a Providence or a Revision Center.

Providence

The providence is the medical institution responsible for recording and uploading the electroencephalogram exams.

Revision Center

The revision center is the medical institution responsible for the visualization, analysis and report generation for an electroencephalogram exam.

Contract

The contract entity is responsible for representing a unique agreement between the providence and the revision center, for the sole purpose of redirecting electroencephalogram exams. When a providence uploads an EEG exam, it will be accessible to the employees of the revision center linked by the contract established by both institutions. The contracts can exclusively be done by the system administrators, once a person could make other contracts and access other institution EEGs.

Person

A person, as the name indicates, represents an individual who participates in the business model. It has some related attributes, such as a unique health number, name, email, telephone, address, gender and birthday.

Just like the institution, the person entity can be divided into three different entities: an operator, a doctor and a patient.

Operator

An operator is a person who works for a providence. He is responsible for performing an electroencephalogram on a patient and for the upload of the EEG exam on the platform.

Doctor

A doctor works for one or more revision centers. He has access to all the EEG exams of the revision centers he works with. A doctor is responsible for the monitorization, management, analysis and report generation of EEG exams.

DoctorRevisionCenter

Since a doctor can work in more than one revision center and each revision center can have more than one doctor, this entity is created. It maps doctors to its revision centers and vice versa.

Patient

A patient is a non-employee individual who gets the electroencephalogram exam. He has all the basic information required for a person with the addition of his medical information. He has a history of EEG exams carried out on him, which can be accessed on the platform by professionals.

EEG

EEG is the entity that represents an electroencephalogram exam. It has a status associated with it (it is valid or with has some error), a timestamp which denotes the date when the EEG was performed, a priority (from very low to very high), a duration (in seconds), a report, an operator (who performed the EEG) and a patient (person who was examined).

SharedFolder

Both institutions which participate in a contract share the same access to the EEG exams. The entity responsible for mapping EEG exams to a contract, which includes a providence and a revision center, is the SharedFolder one.

An institution only has access to the EEGs belonging to its shared folder, whether it's providence or a revision center.

Channel

Channel is an entity responsible for storing the information about an electroencephalogram channel, which has information about the activity of the brain in a specific area, captured by an electrode placed on the scalp.

It's associated with an EEG and has a label, which identifies the area of the brain where it is recorded. Alongside these fields, it also stores a file that contains the values of the channel.

Annotation

An annotation describes an event that happened during the exam and which is important for the doctor to interpret the obtained results. It starts at some point in time, lasts some time and has a description. An example of an annotation can be the recording of eye movement or muscle contraction.

Report

EEG exams, when interpreted and analysed by a doctor, should have a report with the taken conclusion. Other doctors can access reports made by previous doctors to understand the problem or other important information about the exam.

Event

The Event entity is made for monitorization purposes and denotes some action made by a professional related to an EEG exam. There are many types of events, such as the upload, visualization and review of the electroencephalogram.

h. Results

In this section we are going to explain our system in a detailed way.

Designing the website

A prototype was developed using Balsamiq Studios to help us visualize how the final product would turn out. We also did, with the help of our friends and family, some UI tests. We established 4 tasks and each user rated on a scale of 0 easy to 5, the level of difficulty to achieve the task's goal (being 0 easy and 5 harder).

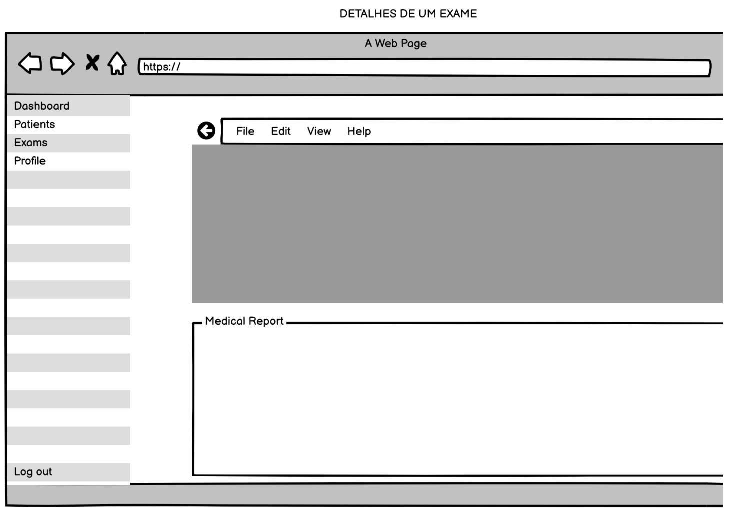
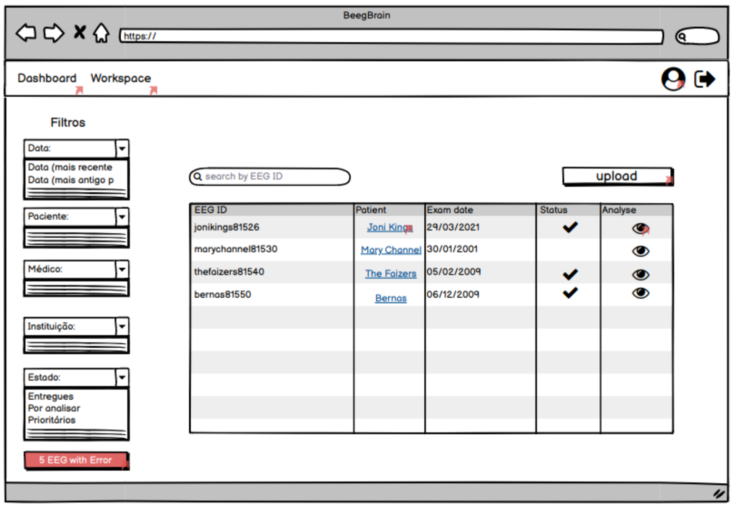


Fig. 12 - EEG Viewer prototype



EEG ID	Patient	Exam date	Status	Analyse
jonikings81526	Joni Kings	29/03/2021	✓	👁
marychannel81530	Mary Channel	30/01/2001		👁
thefaizers81540	The Faizers	05/02/2009	✓	👁
bernas81550	Bernas	06/12/2009	✓	👁

Fig. 13 - Workspace prototype

Tasks:

1. Upload of an EEG and visualize it;
2. Searching for a patient's exam and writing a report about it;
3. Search for a patient and view his clinical history;
4. See the exams that have some errors and contact the support team.

User / Task	Task 1	Task 2	Task 3	Task 4
Ana Sobreira	2	3	5	2
Beatriz Santana	2	3	4	0
Alexandre Serras	1	2	5	0
Maria Almeida	3	5	3	0
Mafalda Reis	3	1	0	0

Table 4 - Users evaluation

With this study we concluded that we must refactor the patient's area and how a user would search for a patient's information and clinical history.

Uploading files

Files' upload are done by drag and drop or select, as you can see in the picture below:

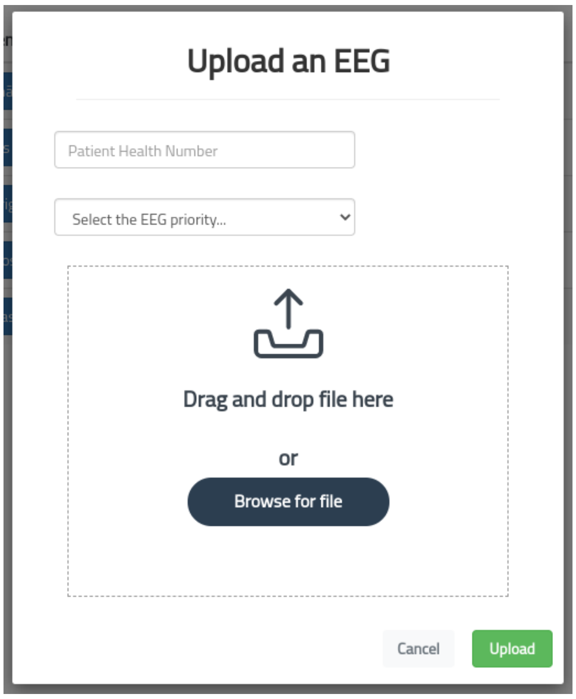


Fig. 14 - Uploading a file

After being uploaded, EEG can take the error area or the transferred area. As you can imagine, the error area contains files that have errors, for example, read errors, when the patient is not defined, and other ones, on the other hand, the transferred area contains files that were uploaded successfully and can be visualized by both the operator and the doctor (we are going to explain the system roles and permissions after), during a number of days, working as a temporary cache. After that, the files will be deleted.
The files are uploaded by a **RESTful API**, developed in Django, as we already said before.

This upload section only detects **EDF** files, if we try to upload a file in a different format, it will not detect it.

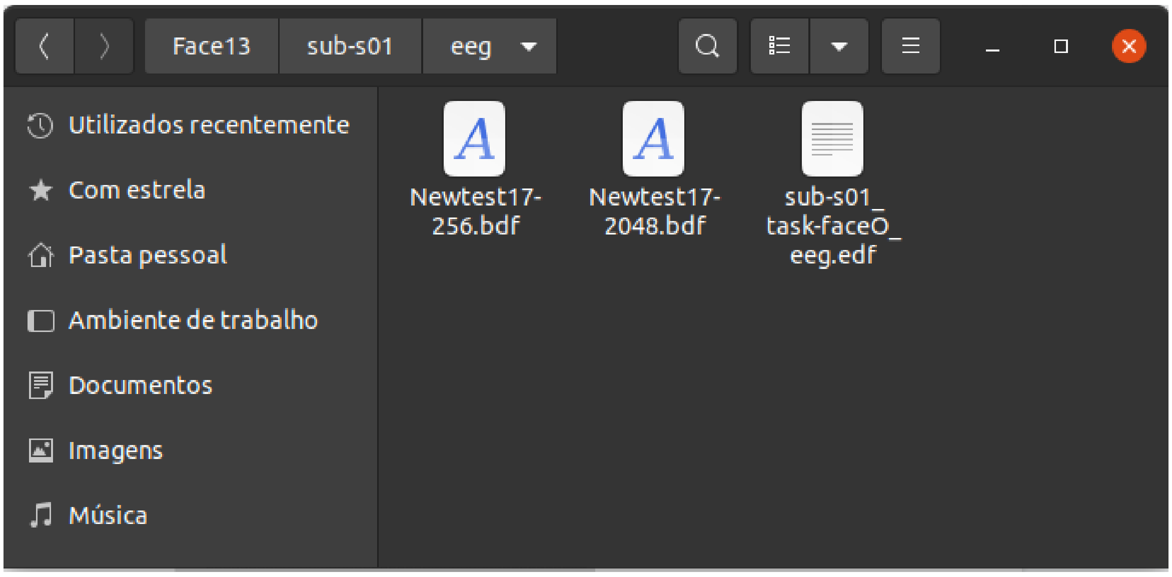
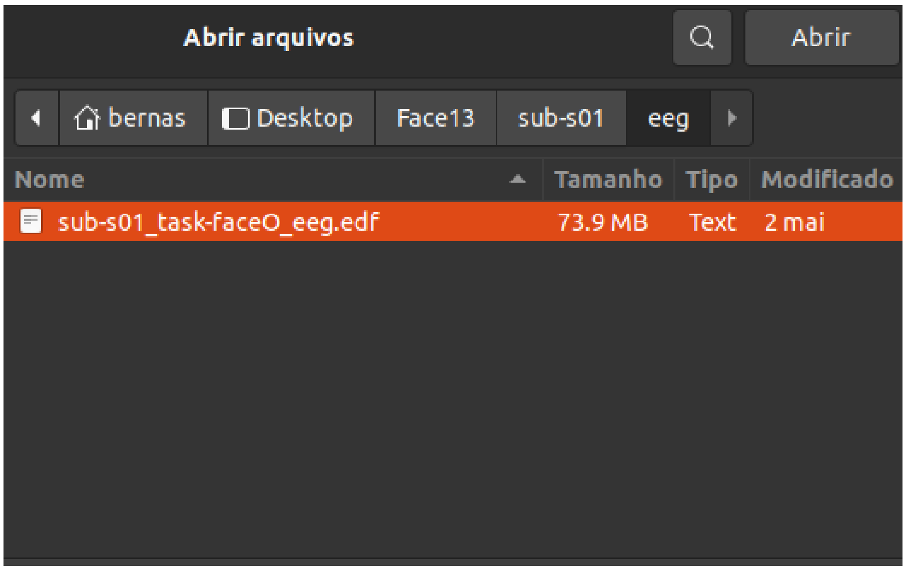


Fig. 15 - 3 types of EEG files



Nome	Tamanho	Tipo	Modificado
sub-s01_task-faceO_eeg.edf	73.9 MB	Text	2 mai

Fig. 16 - Modal of upload in BeegBrain

As you can see in these pictures (fig. 15 and fig. 16), in the same directory, there are 3 files: 1 in EDF format and the other ones in **BDF** formats, the app couldn't detect them.

Standardisation of EDF Service

An EDF, which stands for *European Data Format*, is a file made to universally represent medical time series. This being said, it can be used to store information about electrocardiography (ECG), electromyography (ECM) and electroencephalography (EEG) exams.

This type of file follows a simple structure, containing information about each captured channel, the patient, the used equipment, the duration of the exam and many other critical information.

It has an extension to itself, called EDF+, which was created in 2003 and it's more flexible than the previous format, since it can contain uninterrupted recordings, annotations, etc.

The medical community uses many different formats to store electroencephalogram exams, such as EDF, BDF, VHDR and dozens of others proprietary formats, which difficults the sharing and visualization of these files, since different medical entities may have different software or equipment to read the EEG exam.

The initial plan of the project had a feature to be implemented that would solve this issue by transforming every different proprietary format of an EEG file to the EDF extension. However, as we are going to explain later in this report, this feature wasn't implemented since our team is composed of 4 people, instead of the expected 6, and our project advisors wanted other features, with a higher priority, implemented such as an EEG viewer. You can find the complete description on changes in the initial plan.

Web EEG Viewer

Last, but not least, Web EEG Platform and Viewer. This feature was the one that took us most of the time and we were not expecting that. In the earliest versions of our work, our calendar doesn't contemplate the real number of hours we spent developing and altering this component, but we are going to explain this later in this report.

The viewer suffered several changes, as we already said, and now we are going to show the different stages.

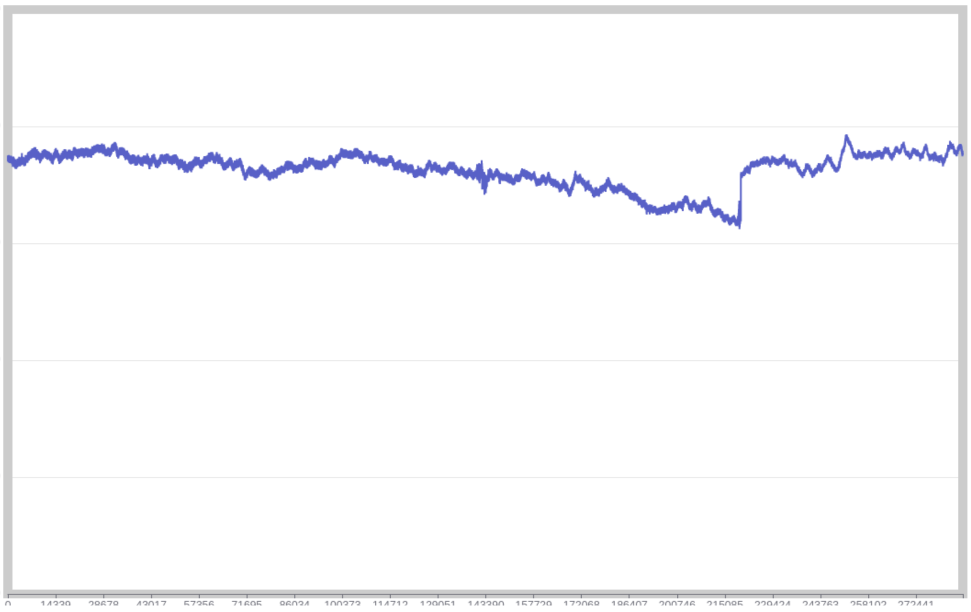


Figure 17: First viewer we implemented

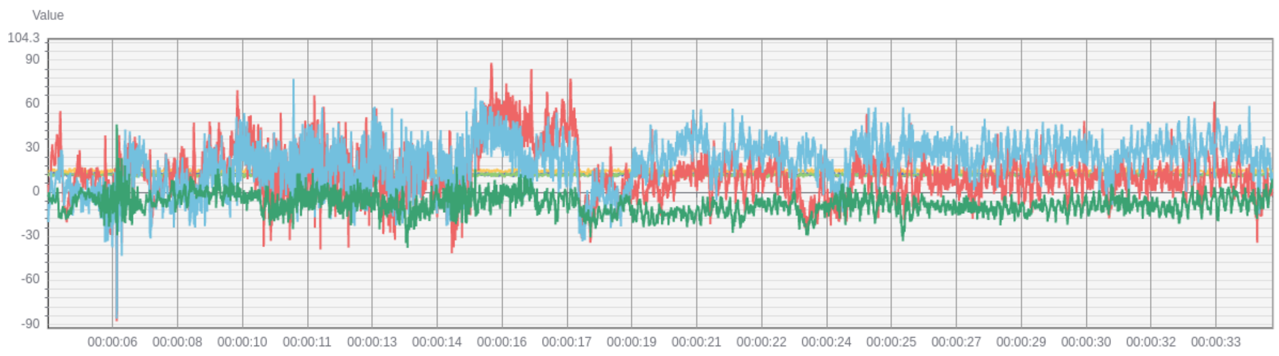


Fig. 18: Second viewer

The first graphic (fig. 18) shows one channel plot, but after showing this one to our guiding teachers, they said that the signals looked like straight lines, so the doctor couldn't achieve the correct results.

Now, the viewer looks like the second image (fig. 19), there are many channels and the signals are overlapped. This facilitates the view to doctors, once right now the signals are bigger and it is easier to check patterns.

Another difference between the two graphics is the number of signals in each sample. In the first image, all data is displayed, while in the second one, just a piece of the signal is used, you can imagine why: **performance**.

Performance was one of the main aspects presented in our work. To give you an idea, one EEG with 1281 seconds (more than 20 minutes) has more than 300000 values per channel, and each exam can have between **1** and **256** channels. With all these values it is hard to maintain the performance and in the earliest versions of the viewer component, we concluded that it was impossible to just get data from **API** and plot it, so we decided to adopt a buffering strategy, we are going to explain right now.

Instead of plotting all data from a channel, we get just a part from the API and we plot less data than what we ask for. Imagine an EEG with 1000 seconds, we ask for data for the first 200 seconds and plot just 100 seconds and we wait for the EEG translation to plot the other 100 seconds (or less) depending on the translation done. The quantity of data the system asks for and the number of channels chosen is inversely proportional.

In order to have an efficient plot, we have a **Map** structure in our code, responsible for storing data for each channel, like a cache. Imagine the user starts visualising the time interval [0,30] seconds and then he translates the graphic to visualize the [30,60] interval. When he gets back to the initial interval the system wouldn't ask for more data. It has stored this **Map** where it can go to and get the necessary information.

Still, in the viewer component, there are various tools to help in visualization, starting with zoom and viewer information when we pass the mouse on the image.

The zoom can be done in a specific area of the chart or through the **axis**, so you can have more or less signal time, if you move **x-axis** and increase/decrease the signal amplitude if you move **y-axis**.

Also, some EEGs may have annotations about important aspects, for example, one EEG that we used, has the following annotation "A1+A2 OFF", this can represent important information to the doctor, so we regard that this information is important and should be represented in our viewer, as you can see in the following figure:

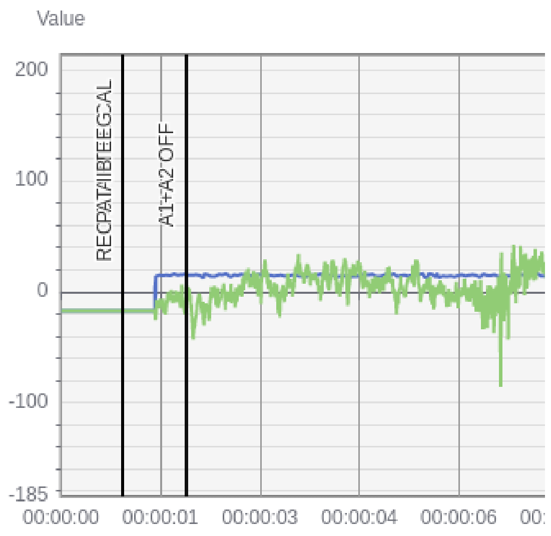


Fig. 19 - Annotations

i. Changes in the initial plan

During the production of a software product, not everything goes as planned. It is normal to have unforeseen events that do not follow the initial plan and, consequently, the final product.

So, in the initial plan, **Operators** and **Doctors** had the same permissions and access in the web application. However, after the beginning of the implementation phase and with some discussion in the group, we concluded that it didn’t make sense. Therefore, Operators don't have access to reports of an exam, while Doctors can’t upload files and also can’t access the error exam files (error area), only the exams where the upload occurred with success (shared folder). The use case diagram (fig. 8), shows these actors' permissions and accesses.

The big change in the origin plan is quitting the File Conversion task. We said in an early phase of our work that this system would be able to receive an EEG in any format and convert it to the standard format EDF. Still, our guiding teachers said to us that according to the number of people involved in the project, it wouldn’t be the priority thing to do, so we didn't implement it due to lack of time and we decided to center our focus on the signal viewer, as we are going to explain after.

The most difficult event was the construction of the EEG Viewer. This task took way more time than we expected, getting us behind our schedule with other tasks. While plotting the signal, first we thought to send the original EEG file, but after implementing it, we quickly realised that this would not be efficient since it would take a lot of time to send a lot of information. To resolve this problem we decided to split the information according to EEG channels (each exam has the information divided into different channels, different brain points). So, when a doctor wants to visualize any signal, he would choose which channel he would like to see (either could be one or all). With all this implemented, we weren’t satisfied with the speed of the plotting of the signal, therefore, we chose to stream the data (as we previously explained before).

A few changes were made to the domain model, for example, adding the attribute “progression” to the report table, so doctors could be more organized in their work: by seeing what is done, in progression and what’s left to do.

In the initial plan, we also intended to test our application and have more statistics about the use of the application in the dashboard, however with the lack of time this wasn’t possible.

j. Conclusions

With this work, we hope to improve health professionals' productivity towards contributing to the world of medicine. Adding this platform to their daily work, it’s expected to boost their work: the high-performance visualization, due to the partial signal rendering and the possibility to write the report while viewing the EEG, will facilitate their tasks. Also, having access to EDF files via the web, allows professionals to access them anywhere and anytime.

We believe that this project has a huge potential and with some improvements could be even better. It could even integrate some AI to detect patterns on anomalies and automatically identify some diseases in order to help analyze exams.

5. Useful links

Our GitHub Repository: Github project

Our Website: Project Website

Our Calendar: Project Calendar

Tasks for each team member: Each member tasks

6. References

EDF and EDF+ files | URL: http://paulbourke.net/dataformats/edf/

EDF viewer version 1 | URL: https://bilalzonjy.github.io/EDFViewer_V1/EDFViewer_V1.html