UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

JOÃO LUDOVICO MAXIMIANO BARBOSA

CADASTRO, PROCESSAMENTO E VISUALIZAÇÃO DE ELETROENCEFALOGRAMA

UBERLÂNDIA – MG
2019

JOÃO LUDOVICO MAXIMIANO BARBOSA

CADASTRO, PROCESSAMENTO E VISUALIZAÇÃO DE ELETROENCEFALOGRAMA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Biomédica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciências.

Área de Concentração: Engenharia Biomédica

Linha de Pesquisa: Sistemas Computacionais e Dispositivos Aplicados à Saúde

Orientador: Dr. João Batista Destro Filho

UBERLÂNDIA – MG

2019

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

B238 2019

Barbosa, João Ludovico Maximiano, 1991-
Cadastro, Processamento e Visualização de Eletroencefalograma [recurso eletrônico] / João Ludovico Maximiano Barbosa. - 2019.

Orientador: João Batista Destro Filho.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Biomédica.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.2018
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia biomédica. I. Destro Filho, João Batista, 1970-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Biomédica. III. Título. CDU: 62:61

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

JOÃO LUDOVICO MAXIMIANO BARBOSA

CADASTRO, PROCESSAMENTO E VISUALIZAÇÃO DE ELETROENCEFALOGRAMA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Biomédica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciências.

Área de Concentração: Engenharia Biomédica

Linha de Pesquisa: Sistemas Computacionais e Dispositivos Aplicados à Saúde

Aprovado em 14 de maio de 2019.

Banca Examinadora:

Prof. João Batista Destro Filho, Dr. – Orientador (UFU)

Prof. Pedro Frosi Rosa, Dr. – Banca Interna (UFU)

Prof. Luiz Otavio Murta Junior, Dr. – Banca Externa (USP)

UBERLÂNDIA – MG

2019

Dedico este trabalho à minha mãe, Maria Barbosa da Silva e a todos os meus amigos que, com muito amor, carinho e apoio não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

Agradeço a João Batista Destro Filho, por aceitar o desafio de me orientar, me escutar, nesta jornada. Quero expressar o meu reconhecimento e admiração pela sua competência profissional, por ser um profissional extremamente qualificado e pela forma humana que conduziu minha orientação.

Aos amigos e colegas do Mestrado que sempre me apoiaram e auxiliaram em todas as etapas do mestrado. Em especial à Flávia Gonçalves, Gaspar Eugênio e Camila Davi.

Agradeço à minha mãe, Maria Barbosa da Silva, heroína que me deu apoio e incentivo nas horas difíceis, de desânimo e cansaço.

Ao Programa de Pós-Graduação em Engenharia Biomédica (PPGEB) e ao corpo docente pelos conhecimentos a mim transmitidos e pelas orientações fornecidas, contribuindo para a minha formação profissional e pessoal. Também, aos funcionários da Universidade Federal de Uberlândia pelo atendimento imediato, serviço eficiente e respeito para com os estudantes, em especial a Marli e Edson, secretários do PPGEB, que sempre me auxiliaram em todos os momentos.

Um agradecimento ao Ronaldo, que fez a diferença na finalização deste trabalho, me auxiliando na correção ortográfica do mesmo. Deixo aqui meu Muito obrigado.

A todos que contribuíram direta ou indiretamente para a realização e conclusão deste trabalho.

"Nada pode ser obtido sem uma espécie de sacrifício. Para se obter algo é preciso oferecer algo em troca de valor equivalente. Esse é o princípio básico da alquimia, a Lei da Troca Equivalente."

FullMetal Alchemist

RESUMO

BARBOSA, João Ludovico Maximiano. **Cadastro, Processamento e Visualização de Eletroencefalograma**. 2019. 143 p. Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Biomédica da Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia-MG, 2019.

O eletroencefalograma (EEG) faz a leitura elétrica do escalpo cerebral, mas os valores coletados por ele são apenas dados que, isoladamente, não transmite uma mensagem ou significado. No entanto, quando estes dados são processados e novos dados são associados a ele, o EEG começa a trazer algumas informações. Podendo ser utilizado para diagnóstico de patologias neurológicas, acompanhamento de pacientes críticos, assim como, para aplicações de interfaces cérebro-máquina entre outras aplicações. Nesta perspectiva, este trabalho apresenta um software para processamento de EEG que gera vários quantificadores, e que foi aprimorado de tal forma que ele necessita de apenas um arquivo de entrada em formato pré-determinado, com um conjunto de informações necessárias para gerar os quantificadores, e assim ele pode processar vários EEG's sem a intervenção continua do pesquisador, o qual pode desempenhar outras atividades enquanto o processamento é realizado. O presente trabalho também contempla uma ferramenta para geração de visualização topográfica dos quantificadores extraídos no processamento, o qual permite fazer comparações entre vários EEG's podendo ser de etiologias diferentes, e também mostra a evolução temporal do quantificador, visualizações estas que são importantes pois facilita análise de resultados, uma vez que os EEG's possuem inúmeros dados e compreende-los apenas na forma numérica não é trivial. Além disso, foi construído também uma plataforma para cadastro dos EEG's coletados pelo grupo de pesquisa, utilizando spring-boot e react, o qual tem por objetivo facilitar a consulta e gerencia dos EEG's coletados, permitindo uma maior destreza na seleção de exames para a realização de estudos clínicos.

Palavras-chave: EEG, Processamento, Visualização, Cadastro.

ABSTRACT

BARBOSA, João Ludovico Maximiano. **Registration, Processing and Visualization of Electroencephalogram**. 2019. 143 p. Essay (Master) - Programa de Pós-Graduação em Engenharia Biomédica da Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia-MG, 2019.

The electroencephalogram (EEG) make the electric reading of the cerebral scalp, but the values collected by it are only data that alone convey a message or meaning. However, when this data is processed and new data is associated with it, the EEG begins to bring some information. It can be used for diagnosis of neurological disorders, critical patient follow-up, as well as for applications of brain-machine interfaces among other applications. In this perspective, this work presents software for EEG processing that generates several quantifiers, and which has been improved in such a way that it needs only one input file in predetermined format, with a set of information needed to generate the quantifiers, and so it can process multiple EEG's without the continuous intervention of the researcher, who can perform other activities while processing is performed. The present work also contemplates a tool for generating topographic visualization of the quantifiers extracted in the processing, which allows comparisons between several EEGs and may be of different etiologies, and also shows the temporal evolution of the quantifier, these views are important because it facilitates analysis of results, since the EEG's have numerous data and understand them only in numerical form is not trivial. In addition, a platform was also built to register the EEG's collected by the research group, using spring-boot and react, which aims to facilitate the consultation and management of collected EEG, allowing greater dexterity in the selection of exams for conducting clinical studies.

Keywords: EEG, Processing, Visualization, Registration.

PUBLICAÇÕES

Publicações Resultantes deste Trabalho em Anais de Congressos:

RAMOS, G. E. O., **BARBOSA, J L M**, et al.. COMPARAÇÃO DO ELETROENCEFALOGRAMA DE PACIENTE COMATOSO COM DIFERENTES GRUPOS.. **In: Anais do V Congresso Brasileiro de Eletromiografia e Cinesiologia e X Simpósio de Engenharia Biomédica.** Anais...Uberlândia(MG) Center Convention Uberlândia, 2018. Disponível em: <https//www.even3.com.br/anais/cobecseb/78921-COMPARACAO-DO-ELETROENCEFALOGRAMA-DE-PACIENTE-COMATOSO-COM-DIFERENTES-GRUPOS>. Acesso em: 12 fev. 2019.

L. M. João Barbosa; E. O. Gaspar Ramos; B. João Destro-Filho. COMPARAÇÃO ENTRE DOIS MÉTODOS DE VISUALIZAÇÃO TOPOGRÁFICA DE DADOS DE ELETROENCEFALOGRAFIA (EEG). **In: ANAIS DO CONFERÊNCIA DE ESTUDOS EM ENGENHARIA ELÉTRICA**, 2017. Anais eletrônicos... Campinas, GALOÁ, 2018. Disponível em: <https://proceedings.science/ceel/papers/comparacao-entre-dois-metodos-de-visualizacao-topografica-de-dados-de-eletroencefalografia-%28eeg%29?lang=pt-br> Acesso em: 12 fev. 2019.

E. O. Gaspar Ramos; **L. M. João Barbosa**; D. Camila Ramos; C. Cláudio Oliveira et al. AVALIAÇÃO DO RUÍDO EM SINAIS ELETROENCEFÁLICOS EM UM AMBIENTE DE TRATAMENTO INTENSIVO. **In: ANAIS DO CONFERÊNCIA DE ESTUDOS EM ENGENHARIA ELÉTRICA**, 2017. Anais eletrônicos... Campinas, GALOÁ, 2018. Disponível em: <https://proceedings.science/ceel/papers/avaliacao-do-ruido-em-sinais-eletroencefalicos-em-um-ambiente-de-tratamento-intensivo?lang=pt-br> Acesso em: 12 fev. 2019.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AVC - Acidente Vascular Cerebral

EEG - Eletroencefalograma

MEG - Magnetoencefalografia

HTML - *HyperText Markup Language* (Linguagem de Marcação de Hipertexto)

XML - *eXtensible Markup Language* (Linguagem de Marcação Extensível)

PCP - Porcentagem da contribuição de potencia

FM - Frequência Mediana

BCI - *brain-computer interface* (Interface Cérebro-Computador)

JSON - JavaScript Object Notation (Notação de objeto javaScript)

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O cérebro é composto por milhões de células, conhecidas como neurônios, que desempenham atividades relacionadas ao sistema nervoso. Estes neurônios são interconectados através de sinapses, que são áreas de proximidade dos neurônios com outras células, onde os impulsos nervosos são transmitidos, podendo atuar de forma inibitória ou excitatória.

Um potencial de ação (impulso nervoso) é uma corrente elétrica bastante sutil e de difícil detecção/leitura. No entanto, quando milhares de neurônios, em sincronia, disparam estes potenciais de ação no cérebro cria-se um campo elétrico forte o suficiente para se espalhar através do tecido, osso e crânio, facilitando a detecção/leitura destes campos na superfície da cabeça através de pequenos discos metálicos chamados de eletrodos/sensores.

Em meados de 1875 e 1877, Richard Caton realizou os primeiros experimentos para medir os potenciais elétricos do cérebro de animais como ratos, coelhos e macacos (NIEDERMEYER; SILVA, 2004). Para medir tais potenciais elétricos Caton utilizou um Galvanômetro, colocando um dos sensores na substancia cinzenta do cérebro e o outro no crânio. Desta forma, percebeu-se que a tensão variava com o passar do tempo. Para registrar essas alterações de tensão Caton acoplou uma agulha ao Galvanômetro que vibrava conforme a tensão mudava, registrando as oscilações de tensão em uma folha de papel que se deslocava constantemente. Sendo assim, viu-se surgir o registro da atividade elétrica cerebral que atualmente é conhecido como Eletroencefalograma (EEG) (LAUCSEN DA ROSA, 2009).

A palavra Eletroencefalograma (EEG) formou-se pela junção de três palavras de origem grega: eletro (do grego *élektron*), que remete a eletricidade; encéfalo (do grego *egképhalos*), caracterizando o cérebro; e grama (do grego *grámma*),

representando letra, escrita, figura, desenho, etc (LAUCSEN DA ROSA, 2009). Portanto o EEG significa o registro da atividade elétrica cerebral.

Com a evolução dos equipamentos e dos sensores que fazem a captura/registro do EEG várias aplicações foram surgindo, tanto na área da saúde, da neurociência, das interfaces cérebro maquina, entre outras.

1.1.1 EEG na saúde

O EEG é um exame de grande relevância no SUS, pois, além de econômico, é bastante conhecido, utilizado e não-invasivo. O EEG desempenha um papel fundamental tanto em nível ambulatorial, onde os pacientes são examinados com o objetivo de se fazer um diagnóstico, quanto do ponto de vista de pacientes em estado crítico sob terapia intensiva, sendo utilizado para identificar o grau/nível de consciência, lesão ou até mesmo a morte encefálica (LAUCSEN DA ROSA, 2009; NIEDERMEYER; SILVA, 2004).

O EEG pode ser realizado em pacientes de qualquer idade e visa diagnosticar eventuais anormalidades da atividade elétrico cerebral como, por exemplo: distúrbio convulsivo, distúrbios do sono, confusão, traumatismos cranianos, tumores cerebrais, infecções, doenças degenerativas, epilepsia, alterações vasculares, derrames, distúrbios metabólicos que afetem o cérebro, entre outras aplicações (CENTRO DE SAÚDE DO TRABALHO – CST MED, [s.d.]).

1.1.2 EEG e interface cérebro maquina

Há algumas décadas muitos especulavam se a atividade eletroencefalográfica ou outras medidas eletrofisiológicas da função cerebral poderiam permitir uma nova maneira, não muscular, de interação com o mundo externo. Tal interação foi denominada como interface cérebro-computador (BCI *brain-computer interface*) (WOLPAW et al., 2002).

Com o grande avanço da ciência, tanto na compreensão da função cerebral, como no advento de equipamentos computadorizados poderosos e de custos mais acessíveis, juntamente com o crescente reconhecimento das necessidades e potencialidades das pessoas com deficiência, várias pesquisas de BCI se

concentraram no desenvolvimento de novas tecnologias de comunicação aumentativa e de controle (WOLPAW et al., 2002). Tendo como objetivo disponibilizar um método de acesso para a comunicação ou reabilitação apropriada a indivíduos que não conseguem atender às suas necessidades linguísticas expressivas por meio da fala natural, caligrafia, etc, ou operar neuropróteses, assim, estas BCI's interpretam diretamente a atividade cerebral, evitando o movimento físico e confiando em sinais neurofisiológicos como um método de acesso (M. et al., 2014).

As BCI's podem utilizar uma grande variedade de sensores para monitorar a atividade cerebral, como, por exemplo: eletroencefalograma (EEG), eletrocorticografia (ECoG), magnetoencefalografia (MEG), tomografia por emissão de positrões (PET - *positron emission tomography*), ressonância magnética funcional (fMRI - *functional magnetic resonance imaging*), entre outras. No entanto, alguns destes sensores ainda são tecnicamente trabalhosos de se realizar e bastante caros (que é o caso do MEG, PET e fMRI) impedindo o seu uso generalizado. Além disso, PET e fMRI dependem de processos metabólicos que possuem constantes de tempo demoradas, fazendo com que eles não sejam indicados para sistemas que necessitem de respostas rápidas. Por outro lado, o EEG e ECoG são métodos que utilizam equipamentos relativamente simples e com o custo mais acessível, além de possuírem alta resolução temporal (SCHALK; MELLINGER, 2010).

O ECoG é um método invasivo que faz o registro a partir da superfície cortical; já o EEG é um método não invasivo que faz a leitura a partir do escalpo cerebral. Pelo fato do EEG não ser invasivo e de possuir um custo mais acessível ele acaba sendo uma das ferramentas mais escolhidas para criação das BCI's.

1.1.3 EEG e neurociência

Comumente os sinais EEG / MEG são descritos em termos de bandas de frequência (lentas [< 0.2 Hz], delta [0.2 a 3.5 Hz], teta [4 a 7.5 Hz], alfa [8 a 13 Hz], beta [14 a 30 Hz], gama [30 a 90 Hz] e oscilações de altas frequências [> 90 Hz]), cujos limites foram artificialmente definidos sem o conhecimento de mecanismos neurofisiológicos (LOPES DA SILVA, 2013).

Existem muitos estudos voltados para a neurocognição, dos quais, muitos que focam em potenciais relacionados a eventos (ERP - *Event-Related Potential*), campos

magnéticos (ERFs - *Event-Related Magnetic Fields*) e oscilações neuronais (LOPES DA SILVA, 2013).

Boa parte destes estudos tentam associar as oscilações registradas no EEG com as atividades neuronais que estão ocorrendo, com o intuito de compreender melhor com qual intensidade e região do cérebro estão sendo ativadas em experimentos bem delineados.

1.1.4 EEG e monitorização contínua em unidade de tratamento intensivo (UTI)

A monitorização de pacientes em uma unidade de tratamento intensivo (UTI) é bastante útil para: a detecção de estado epiléptico não convulsivo (NCSE - *nonconvulsive status epilepticus*) em pacientes com transtorno de consciência inexplicável ou deterioração mental; avaliação do estado sedativo ou anestésico; detecção precoce de isquemia cerebral retardada associado à hemorragia subaracnóidea; avaliação de resultado de pacientes com encefalopatia pós-ressuscitação ou distúrbios neurológicos graves subsequentes (KUBOTA et al., 2018). Para pacientes com NCSE, a monitorização contínua pode auxiliar na detecção de alterações, permitindo, desta forma, um tratamento precoce e avaliação de resposta ao tratamento (KUBOTA et al., 2018).

No Brasil há falta de investimentos e o custo para treinar e manter uma pessoa dedicada a avaliar os sinais que estão sendo coletados continuamente é dispendioso. Algumas estratégias vêm sendo utilizadas para contornar essas dificuldades, dentre elas está o uso de algoritmos que avaliam os sinais continuamente e separam épocas que possuem características patológicas ou disparam algum aceno para que os médicos possam avaliar o estado do paciente.

1.2 JUSTIFICATIVA

O eletroencefalograma (EEG) faz a leitura elétrica do escalpo cerebral, mas os valores coletados por ele são apenas dados que, isoladamente, não transmitem uma mensagem ou significado. No entanto, quando estes dados são processados e novos dados são associados a ele, o EEG começa a trazer algumas informações. Estas informações, associadas com outras informações do contexto biológico,

conduzem à compreensão de alguns fenômenos e, destarte, é possível ser assertivo em diagnosticar ou prever determinadas circunstâncias.

O EGG oferece enormes possibilidades como uma janela para a função cerebral, disponibilizando um potencial para compreender em grande escala as atividades da rede neural, servindo como uma ponte entre os neurônios e o comportamento (LEVIN et al., 2018). O EEG pode ser realizado em populações típicas e clínicas de qualquer faixa etária e sem a necessidade de sedação, facilitando, desta maneira, a aquisição de dados. A sua portabilidade e acessibilidade relativa permitem que seja facilmente utilizado para estudos multicêntrico, o que é particularmente útil quando se estudam doenças raras ou se adquirem grandes conjuntos de dados (LEVIN et al., 2018).

Apesar dos grandes avanços do processamento e da compreensão do EEG, a acessibilidade de análises integradas permanece limitada em alguns casos. Para usuários sem experiência em codificação a criação de um script ou pipeline de análise de um EEG pode ser um processo intimidador e assustador (LEVIN et al., 2018). Para os usuários que criam tais scripts, manter o controle das entradas, saídas e configurações específicas em várias etapas de uma análise pode ser, também, um grande desafio. Estas são algumas das preocupações dos usuários que avaliam grandes conjuntos de dados, nos quais o formato nativo de EEG, taxas de amostragem, *layouts* de eletrodos durante a aquisição, nomes de variáveis e até mesmo as frequências de ruído de linha (para estudos internacionais) podem diferir entre os arquivos analisados (LEVIN et al., 2018).

Do mesmo modo, outra dificuldade encontrada com os dados do EEG é a sua validação e análise imediata, sendo necessário o apoio de ferramentas de ciência de dados e especialistas qualificados (RAMACHANDRAN et al., 2018). Quando os dados são registrados em ambientes não clínicos e/ou com escassez de recursos, a validação dos dados é difícil devido à indisponibilidade de sistemas computacionais e de profissionais especializados, que geralmente se concentram mais em laboratórios. Durante a análise de dados muitos registros/coletas são frequentemente excluídos devido à sua alta taxa de ruído e algumas outras razões. Se os dados puderem ser validados usando plataformas acessíveis e de fácil manuseio, o treinamento de profissionais de saúde ou de ciência de dados será mais eficiente, evitando-se grandes perdas (RAMACHANDRAN et al., 2018).

O processamento do EEG pode ser realizado de inúmeras maneiras, gerando inúmeros outros dados, e este trabalho apresenta uma maneira mais automatizada de processar o EEG, de tal forma que os profissionais da saúde possam manuseá-lo sem possuírem conhecimentos avançados em programação. Ademais, a exibição dos dados processados em forma numérica nem sempre é de fácil compreensão. Por vezes, torna-se necessário haver outras ferramentas que exibam tais informações de maneira mais ilustrativa, de modo que possam apresentar um significado mais claro. Por isso, este trabalho também aborda a visualização de quantificadores do EEG.

À medida que os EEG's vão sendo coletados e a necessidade de guarda/armazenar existe, aparece uma nova demanda que é a gestão destes dados. Quando a quantidade dos dados é pequena, um controle manual e pequenas anotações sobre os dados são suficientes. No entanto, à medida que estes dados aumentam, quantitativamente, um controle automatizado se torna necessário. Imagine que se tenha mil registros de EEG, dentre os quais se esteja procurando um registro específico. Todas as informações disponíveis estão no papel, o que exige uma organização bem delineada sobre estes dados, caso contrário, seria difícil vistoriar registro por registro até encontrar o que se procura, o que demandaria muito tempo. Já no controle das informações dos registros de forma computadorizada, com campos de consulta bem definidos, esta procura por um registro em específico levaria poucos segundos. Logo, este trabalho contempla, também, uma ferramenta computadorizada para o cadastro de registros de EEG's coletados.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Este trabalho tem por objetivo principal aprimorar os procedimentos de cadastro, processamento e visualização de EEG, de tal forma a otimizar a análise quantitativa e o armazenamento dos registros, facilitando a realização de estudos clínicos.

1.3.2 Objetivos Específicos

Com o propósito de alcançar o objetivo principal desta dissertação, foram estipulados objetivos específicos, listados a seguir:

- Versionar o código fonte do software de processamento do EEG;
- Reestruturar o *software* já existente de processamento utilizado pelo grupo de pesquisa do Prof. Dr. João Batista Destro Filho;
- Automatizar as etapas do software, evitando ao máximo interações com o usuário;
- Criar um teste de regressão para comparar os resultados encontrados na versão nova com os resultados da versão anterior e, assim, validar a nova versão;
- Verificar maneiras de visualizar os resultados do processamento do EEG;
- Desenvolver uma plataforma *WEB* (*World Wide Web* - Rede mundial de computadores) para o cadastro dos EEG coletados;

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação encontra-se estruturada e organizada da seguinte forma:

No Capítulo 1, foi apresentada uma introdução e demonstrada a relevância do EEG para a ciência e a sociedade.

O Capítulo 2 apresenta o Estado da Arte, mostrando software utilizados para o processamento do EEG e uma comparação sucinta destes software do ponto de vista do desenvolvimento/implementação.

O Capítulo 3 apresenta as ferramentas e tecnologias utilizadas para o desenvolvimento deste trabalho.

O Capítulo 4 apresenta o desenvolvimento e os resultados da nova versão do software de processamento de EEG do grupo de pesquisa.

O Capítulo 5 apresenta o desenvolvimento e os resultados de maneiras de visualizar os quantificadores extraídos do processamento do EEG.

O Capítulo 6 apresenta o desenvolvimento e os resultado de um sistema para cadastro dos EEG coletados.

O Capítulo 7 apresenta as conclusões e os trabalhos futuros.

CAPÍTULO 2 - ESTADO DA ARTE

Um registro de EEG consiste em X_t amostras de tempo para Y_{ch} canais ou localizações de eletrodos; cada amostra é um vetor de Y_{ch} números reais, cada número representando o potencial elétrico em um local/canal/eletrodo específico e o vetor, como um todo, fornece uma amostragem discreta do campo elétrico contínuo (VON WEGNER; LAUFS, 2018). Um conjunto de dados de EEG pode, portanto, ser visualizado como uma série temporal de mudanças nos padrões espaciais, frequentemente chamados de mapas (VON WEGNER; LAUFS, 2018).

Os software de processamentos de EEG trabalham estas X_t amostras para extraírem outras informações, e cada software pode ter finalidades diferentes e trabalhar estas amostras também de maneiras diferentes. Eles podem compreender desde a coleta do sinal, pré-processamento, processamento, visualização entre outras funcionalidades. Desta maneira, abaixo é descrito brevemente alguns destes software.

2.1 EEGLAB

EEGLAB é uma ferramenta interativa de script baseado em menus desenvolvido em Matlab® para processar dados eletrofisiológicos incorporando análise de componentes independentes (ICA), análise de tempo / frequência, rejeição de artefatos, estatísticas relacionadas a eventos e vários modos úteis de visualização. O EEGLAG é, ainda, multiplataforma, podendo ser executado em Linux, Unix, Windows, e Mac OS X (DELORME et al., 2011) (SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE, 2018).

O EEGLAB compreende mais de 400 funções do Matlab®, totalizando mais de 50.000 linhas de programação. O tutorial on-line do EEGLAB inclui mais de 300 páginas de documentação, e ainda cada uma das 400 funções modulares EEGLAB independentes contém sua própria documentação (DELORME et al., 2011). Possui também uma função de histórico de comandos que facilita a transição dos usuários da exploração de dados baseada em GUI (Interface gráfica do usuário) para a

construção e execução de scripts de análise de dados em lote ou personalizados (SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE, 2018).

Ele oferece, também, uma plataforma extensível de código aberto por meio da qual os usuários podem compartilhar novos métodos com a comunidade de pesquisa mundial, publicando funções *'plug-in'* do EEGLAB que aparecem automaticamente no menu EEGLAB para que eles possam baixar e utilizar (SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE, 2018). Já existem pelo menos 28 *plug-ins* que já foram desenvolvidos e disponibilizados por grupos de usuários, que estão acessíveis através do EEGLAB (DELORME et al., 2011).

O código fonte está disponível em: https://github.com/eeglabdevelopers/eeglab.

2.2 ERPLAB

ERPLAB é uma caixa de ferramentas de código aberto disponível gratuitamente para processar e analisar dados de potencial relacionado a eventos (ERP - *Event-Related Potential*) em ambiente MATLAB®. O ERPLAB está intimamente integrado ao EEGLAB, e adiciona funções de processamento de EEG ao EEGLAB, fornecendo ferramentas adicionais para filtragem, detecção de artefatos e classificação de eventos, entre outros (LOPEZ-CALDERON; LUCK, 2014).

O ERPLAB também fornece ferramentas robustas para o cálculo da média de segmentos EEG juntos para criar média de ERP, formando ondas de diferenças e outras recombinações de formas de onda ERP através de expressões algébricas, para filtrar e re-referenciar os ERPs médios, para traçar gráfico das formas de ondas e do escalpo cerebral, para quantificar os vários tipos de amplitudes e latências (LOPEZ-CALDERON; LUCK, 2014).

Possui, do mesmo modo, várias formas de documentação incluindo um guia detalhado do usuário, um tutorial passo a passo, um guia de scripts e um conjunto de demonstrações baseadas em vídeo (LOPEZ-CALDERON; LUCK, 2014).

O código fonte está disponível em: https://github.com/lucklab/erplab.

2.3 FIELDTRIP

FieldTrip é um toolbox do matlab® para análise de MEG, EEG e outro dados eletrofisiológicos. Distribuído gratuitamente sobre a licença pública GNU, o seu desenvolvimento se iniciou em 2003 pelo *F.C. Donders Centre for Cognitive Neuroimaging* e até hoje continua sendo desenvolvido ativamente pelo *Donders Institute for Brain, Cognition and Behaviour* da Radboud University Nijmegen, na Holanda, junto com a colaboração de institutos de pesquisas. Ele é composto por aproximadamente 108 funções de alto-nível e 858 funções de baixo nível. Seu principal foco é a análise invasiva e não-invasiva de dados eletrofisiológicos. Suporta a leitura de um grande número diferente de formatos de entrada de dados e não possui uma interface gráfica de usuário (OOSTENVELD et al., 2011).

O código fonte está disponível em: https://github.com/fieldtrip/fieldtrip.

2.4 MNE-PYTHON

MNE-Python é um subprojeto de um pacote de software acadêmico mais geral, denominado MNE, que possui o objetivo de implementar e fornecer um conjunto de algoritmos que permita aos usuários criar *pipelines* completos de análise de dados que abrangem a maioria das fases do processamento de dados M / EEG. Ele permite combinar um conjunto de técnicas avançadas e padrões para a análise e processamento do sinal. Projetado para se integrar com pacotes da comunidade Python e, também, com outros componentes da suíte MNE, porque utilizam o mesmo formato de arquivo Neuromag FIF. O MNE-Python e o subpacote relacionado do MNE-Matlab® que acompanham o MNE são ambos de código aberto e distribuídos sob a nova licença BSD, permitindo seu uso em software livre e comercial. Fornece vários recursos, como: análise de tempo-frequência, estatística não-paramétrica, estimativa de conectividade, análise de componente independente (ICA) e decodificação, também conhecido como análise multivariada de padrões (MVPA) ou simplesmente aprendizado supervisionado. Cada um dos quais é prontamente integrado no pipeline de análise padrão da MNE (GRAMFORT et al., 2013).

Para alcançar essa tarefa, o MNE-Python é construído sobre a base das principais bibliotecas fornecidas pelo ambiente científico do Python, suporta a leitura de dados brutos de vários formatos de arquivo como, por exemplo, BTI / 4D, KIT, EDF, Biosemi BDF e BrainVision EEG (GRAMFORT et al., 2013).

O código fonte está disponível em: https://github.com/mne-tools/mne-python

2.5 PYEEG

PyEEG é um módulo Python focado somente em extrair características a partir de segmentos de EEG/MEG. Portanto, ele não contém funções para importar dados de vários formatos ou exportar recursos para um classificador. Isso se deve aos princípios de modularidade e composição da criação de software de código aberto que indicam que os pequenos programas que podem funcionar bem, juntos por meio de interfaces simples, são melhores que os grandes programas monolíticos, uma vez que ferramentas de código aberto, como importadores de dados EEG / MEG (por exemplo, EEGLab, Biosig, etc.) e front ends de classificadores já estão disponíveis. Os usuários podem facilmente conectar o PyEEG a vários software de código aberto existentes para construir cadeia de ferramentas para suas pesquisas de EEG/MEG. O PyEEG utiliza apenas funções da biblioteca padrão do Python e no SciPy, o módulo Python para os cálculos científicos (BAO; LIU; ZHANG, 2011). Detalhes sobre as funções estão disponíveis no guia de referência do PyEEG em http://pyeeg.sourceforge.net/.

O código fonte está disponível em: https://github.com/forrestbao/pyeeg

2.6 BRAIN VISION ANALYZER

O BrainVision Analyzer iniciou-se em 1997 e não é de código aberto, sendo necessário a compra de uma licença para utilizá-lo. Em sua última versão Analyzer 2, possui uma interface amigável, suporte científico gratuito, baseado em métodos científicos confiáveis, recursos facilmente expansíveis (MATLAB®) e compatíveis com outros software de análise (BESA), suporte para mais de 50 formatos de arquivo (incl. coordinates, triggers, etc.) (BRAIN PRODUCTS GMBH, 2018).

Contém uma representação visual das etapas de análise em uma estrutura chamada “*History Trees*”. Tais etapas de análise no “*History Trees*” permitem explorar e comparar várias estratégias de processamento de dados. Cada nó do “*History Tree”* contém uma descrição completa dos parâmetros usados no processamento de dados (BRAIN PRODUCTS GMBH, 2018).

Possibilita ainda a aplicação de filtros, inspeção de dados brutos, ICA interativo, FFT, Wavelets, coerência, LORETA. Transformações para correção de artefatos MR e CB; integração de dados Eye-Tracking, podendo se comunicar em tempo real com o MATLAB® (BRAIN PRODUCTS GMBH, 2018).

2.7 BRAINSTORM

A primeira versão do Brainstorm foi liberada em 2000 com a colaboração entre University of Southern California em Los Angeles, o Hospital Salpêtrière em Paris e o Laboratorio Los Alamos no Novo México. O projeto foi apoiado pelos Institutos Nacionais de Saúde (NIH - National Institutes of Health) nos EUA e pelo Centro Nacional de Pesquisa Científica (CNRS - Centre National de la Recherche Scientifique) na França. Com o objetivo de disponibilizar uma ampla gama de técnicas de visualização e geração de imagens de origem eletromagnética para usuários não técnicos, com ênfase na interação dos usuários com seus dados em múltiplos estágios da análise (TADEL et al., 2011).

Em 2004 foi adicionado uma completa interface gráfica de usuário (GUI) e o número de usuários começou a crescer, sendo a interface gráfica completamente redesenhada e aprimorada. Muitas outras ferramentas foram integradas no Brainstorm para cobrir todo o processamento e visualização das gravações MEG / EEG, desde a importação de arquivos de dados, de uma grande seleção de formatos até a análise estatística dos mapas de geração de imagens de origem. Assim, em 2009, foi disponibilizado o Brainstorm 3 (TADEL et al., 2011).

Brainstorm é um software de código aberto escrito quase inteiramente em scripts Matlab® e distribuído sob os termos do Licença Pública Geral (GPL - General Public License). Sua interface está escrita em Java/Swing incorporado em scripts Matlab®, usando a capacidade do Matlab® de trabalhar como um console Java. A

combinação destas duas ferramentas fez com que ele se tornasse totalmente portátil e multi-plataforma (TADEL et al., 2011).

Todas as funções do software são acessíveis através da GUI, sem qualquer interação direta com o ambiente Matlab®, podendo ser usado sem Matlab® ou sem experiência em programação. Para usuários mais avançados também é possível executar todos os processos e exibições dos scripts do Matlab®, bem como, todas as estruturas de dados manipuladas pelo Brainstorm podem ser facilmente acessadas a partir da janela de comando do Matlab®. O código-fonte é acessível para desenvolvedores em um servidor SVN. A documentação do usuário é organizada principalmente em tutoriais on-line ilustrados com várias capturas de tela que orientam o usuário passo a passo através de todos os recursos do software (TADEL et al., 2011).

O código fonte está disponível em: https://github.com/brainstorm-tools/brainstorm3.

2.8 EMEGS (ELECTROMAGNETIC ENCEPHALOGRAPHY SOFTWARE)

EMEGS (*electromagnetic encephalography software*) é um software de código aberto escrito em MATLAB® e desenvolvido para a análise de dados coletados com alta densidade, eletroencefalografia (EEG) e magnetoencefalografia (MEG) (PEYK; DE CESAREI; JUNGHÖFER, 2011).

Completamente baseado em interface gráfica de usuário, seu desenvolvimento iniciou-se em 1997 por Markus Junghofer, que na época estava na *Konstanz University* (Alemanha). Em 2003 Peter Peyk (*Saarland University*, Alemanha) se juntou à equipe de desenvolvedores e o EMEGS foi publicado sob os termos do GNU (*General Public License*) e, desde então, continuou a ser aprimorado por um conjunto pequeno, mas crescente, de usuários (PEYK; DE CESAREI; JUNGHÖFER, 2011).

Este software compreende um conjunto de funções para segmentar e filtrar dados contínuos, excluir artefatos estatisticamente, corrigir artefatos de movimento ocular, média entre os ensaios, interpolação para sensores ausentes

ou ruidosos e média entre os participantes (PEYK; DE CESAREI; JUNGHÖFER, 2011). Ele está disponível para download gratuitamente em http://www.emegs.org/ e é gerenciado como um repositório CVS (*Concurrent Version System*) por um servidor localizado na Universidade de Saarland (PEYK; DE CESAREI; JUNGHÖFER, 2011).

EMEGS oferece uma variedade de módulos de visualização para plotagem de curva de campos relacionados a eventos / ERP / ERF (event-related potentials), espectrogramas de wavelets, projeção de dados em modelos 3D (esfera, realista ou cerebral) e coloração estatística de P, t, ou valores de F. Permite executar diretamente uma variedade de análises e testes estatísticos em dados ERP / ERF, tais como estimativas de fonte inversa, funções de densidade de fonte de corrente, *co-registration* de posição de sensor MEG, transformação rápida de Fourier, análise de componentes principais, testes t, análise de medidas repetidas de variância (ANOVA - *analysis of variance*), testes de permutação, regressão e correlação. Oferece integração com o ambiente R para computação estatística, permitindo o cálculo de experimentos fatoriais avançados diretamente em dados EEG e MEG (PEYK; DE CESAREI; JUNGHÖFER, 2011).

O código fonte está disponível em: https://sourceforge.net/p/emegs/code/.

2.9 BRAINVISA

É uma plataforma de software que permite integrar várias ferramentas para visualização e processamento de dados cerebrais (MRI, fMRI, MEG / EEG, etc.) em um ambiente comum. Procurando seguir algumas diretrizes, como: compatibilidade permitindo a integração com outros plugins já existentes, abertura proporcionando que novos algoritmos sejam incorporados facilmente, e ergonômico possuindo uma interface gráfica amigável tanto para clínicos como para cientistas (COINTEPAS et al., 2001). A documentação do software pode ser encontrada em: http://brainvisa.info/web/documentation.html.

O código fonte está disponível em: https://github.com/brainvisa.

2.10 ELAN

ELAN é um pacote de software que possui muitas ferramentas de pré-processamento e análise de sinais para inúmeros tipos de dados eletrofisiológicos. Suas principais características são: mapeamento topográfico, análise de tempo-frequência em conjunto com rotinas estatísticas adaptadas e ferramentas de visualização interativa e com uma interface amigável para navegar nos conjuntos de dados. Sua primeira versão foi lançada em 2001, sendo ele um software livre para download. Guia do usuário, tutoriais e recursos adicionais sobre o ELAN estão disponíveis no seguinte site: http://elan.lyon.inserm.fr/ (AGUERA et al., 2011).

2.11 BIOSIG

BioSig é um software de código aberto para processamento de sinais biomédicos. Pode ser utilizado tanto no Matlab® quanto no Octave, que é uma linguagem de programação de alto nível muito semelhante ao Matlab®, mas de código aberto e gratuito, possuindo compatibilidade entre estas duas plataformas (SCHÖLGL; VIDAURRE; SANDER, 2011). Ele possui, também, uma ferramenta intitulada rtsBCI, que é uma interface cérebro computador para a criação de sistemas on-line.

O código fonte está disponível em: https://sourceforge.net/p/biosig/code/ci/master/tree/.

2.12 ASA

ASA (*Advanced Source Anlaysis*) distribuído por ANT B.V. (Advanced Neuro Technology), foi desenvolvido em 1996 com o objetivo de disponibilizar novos métodos de visualização e análise MEG em um projeto de pesquisa MEG na *Twente University of Enschede* na Holanda (ZANOW; KNÖSCHE, 2004).

Este software permite visualizações de modelos de cabeça individual e padronizado, além de métodos de análise de sinal como: filtragem, remoção de

tendência linear, cálculo de médias em condições múltiplas e médias grandes de eventos relacionados a dessincronizarão (ERD - *Event-related desynchronization*) (ZANOW; KNÖSCHE, 2004).

2.13 CARTOOL

Cartool não é um projeto de código aberto, mas é disponibilizado gratuitamente para qualquer grupo de pesquisa sem fins lucrativos, sendo originalmente desenvolvido para técnicas de mapeamentos de EEG sem referência a 20 anos atrás. Com o passar dos anos o Cartool foi incorporando novas funcionalidades além de visualizações dinâmicas de mapas de EEG e medidas topográficas quantitativas, como: pré-processamento dos dados, interpolação de eletrodos, filtragem, média, re-referência, métodos de análise espacial para EEG espontâneo e baseado em eventos, segmentação espacial microstática, análise de cluster, métodos de ajuste, métodos estatísticos de análise topográfica (BRUNET; MURRAY; MICHEL, 2011).

2.14 BEAPP

A plataforma automatizada de processamento de lotes de eletroencefalograma (BEAPP – Batch Electroencephalography Automated Processing Platform) tem por objetivo auxiliar na acessibilidade e na reprodutibilidade de processamento automatizado de conjunto de dados de EEG, baseando-se em ferramentas de análise de EEG preexistentes, fornecendo uma estrutura flexível para este processamento (LEVIN et al., 2018).

Oferece uma série de etapas automatizadas para gerenciar EEGs coletados em várias configurações de aquisições; minimiza artefatos, realiza vários tipos de re-referência, segmenta EEG contínuo e/ou relacionados a eventos e realiza análises básicas de tempo-frequência (LEVIN et al., 2018).

O código fonte está disponível em: https://github.com/lcnbeapp/beapp.

2.15 CONSIDERAÇÕES FINAIS

Abaixo é exibido uma tabela resumindo as características básica dos software.

Tabela 1 - Características básicas dos software de processamento de EEG pesquisados

Software	Lingua-gem	Interface Gráfica	Licença de Uso	Tamanho para download *	Versiona-mento	Dados Versionamento	Características	Vantagens/Virtudes	Deficiências
EEGLAB	Matlab	Sim	GNU General Public License	30.2 MB	GIT	Commits: 12214 Releases: 0 Contribuidores:14 Ultimo commit: 06/12/2018	Funções: 400 Linhas de código: 50000 Plataforma: Multi	Permite adição de novos plugins que podem ser compartilhados com todo o mundo.	
ERPLAB	Matlab	Sim	Creative Commons Attribution License (CC BY)	4.2 MB	GIT	Commits: 418 Releases: 10 Contribuidores: 4 Ultimo commit: 06/12/2018	Número de downloads do software: 7000	Teste de resultados com: Neuroscan2, brain electrical source analysis (BESA3), e BrainVision Analyzer4.	
FIELDTRIP	Matlab	Não	GNU General Public License	76.1 MB	GIT	Commits: 15417 Releases: 1 Contribuidores:75 Ultimo commit: 06/12/2018	Plataforma: Multi Funções alto nível: 108 Funções baixo nível: 858 Linhas de código: 103227	Possibilidade de aplicação de análise em tempo real, devido sua velocidade de processamento	
MNE-Python	Python	Não	BSD licenced	58.5 MB	GIT	Commits: 13824 Releases: 64 Contribuidores: 150 Ultimo commit: 07/12/2018	Linhas de Código em Python: 44000 Linhas de comentário: 22000 Documentação: 35% Está entre os mais bem documentado projetos Python	Suíte de teste com cobertura de 86% das linhas de Código. Leitura de dados em disco sobre demanda, poupando gastos de processamento.	
PyEEG	Python	Não	Open Source	22.8 KB	GIT	Commits: 58 Releases: 0 Contribuidores: 5 Ultimo commit: 13/09/2016		Capaz de exportar recursos para o formato *svmlight* para chamar ferramentas de aprendizado de máquina. Criado inicialmente para reconhecimento de EEG	Não possui um importador de arquivos de EEG, sendo necessário utilizar ferramentas de

Software	Lingua-gem	Interface Gráfica	Licença de Uso	Tamanho para download *	Versiona-mento	Dados Versionamento	Características	Vantagens/Virtudes	Deficiências
								epiléptico contra o EEG normal	terceiros (ex: ,EDFBrowser, EEGLAB, etc)
Brain Vision Analyser	-	Sim	100 euros por 2 meses de teste	175,8 MB	-		Recursos facilmente expansíveis (MATLAB®) e compatíveis com outros software de análise (BESA)	Auto documentado Boa usabilidade e interface amigável Suporte Científico Gratuito Baseado em métodos científicos confiáveis	
Brainstorm	Matlab	Sim	GNU General Public License	62 MB	GIT	Commits: 1337 Releases: 0 Contribuidores: 15 Ultimo commit: 07/12/2018	Usuários registrados: > 20000	Ferramenta de fácil utilização, através de interface gráfica rica e intuitiva, não requerendo nenhum conhecimento em programação	
EMEGS	Matlab	Sim	GNU General Public License	312 MB	CVS (Concurrent Version System)		Memória RAM necessária: > 2GB Aceleradores gráficos são bem-vindos para montagem de visões 3D.	Adverte o usuário através de mensagens de aviso e sugestões para parâmetros de configurações ou alertas de combinação atípica de métodos, prevenindo erros. No modo “EXPERT MODE”, estes avisos são desconsiderados.	
BRAINVISA	Python	Sim		23 MB	GIT	Ultimo commit: 17/12/2018	Plataforma: Multi Memória RAM: 2Gb Espaço em disco: 1.5Gb	Permite a integração com outros plugins.	
ELAN	C	Sim		5.5 MB			Plataforma: Linux Memória RAM necessária: > 128MB Memória RAM recomendada: 4 GB	Analise virtual de qualquer tipo de sinal contínuo eletrofisiológico (EEG, MEG, iEEG, ECoG, LFP, etc).	

Software	Lingua-gem	Interface Gráfica	Licença de Uso	Tamanho para download *	Versiona-mento	Dados Versionamento	Características	Vantagens/Virtudes	Deficiências
								Alta velocidade de processamento através de algoritmos otimizados e código C compilado	
BIOSIG	Matlab/C	Sim	GNU General Public License	11.2 MB	GIT	Downloads: 140.013 Ultimo commit: 03/12/2018		Estratégia de uso memória para EEG e MEG em cálculos de alta densidade; Compatibilidade com o Octave.	
ASA		Sim	Paga				Plataforma: Windows Processador (CPU): > 2.2 GHz Memória RAM: > 1024 MB Internet Explorer: > 6.0.26 DirectX: > 9.0 c		
CARTOOL	C++	Sim	Gratuito para pesquisas sem fins lucrativos	19.6 MB			Plataforma: Windows Usuários cadastrados: 650 Memória RAM recomendada: 8 GB Tela de resolução recomendada: 1600x1200 pixels	Necessita de uma placa de aceleração gráfica OpenGL. Escrito em C ++ para atingir a maior velocidade e compacidade em uso de memória	
BEAPP	Matlab	Sim	GNU General Public License	394 MB	GIT	Commits: 34 Releases: 0 Contribuidores: 2 Ultimo commit: 28/09/2018	Rastreia a saída e os parâmetros de cada etapa da análise, permitindo a revisão de etapas anteriores ou a execução novamente de uma parte da análise com	As entradas através do GUI podem ser salvas como um modelo, para futuros usuários, permitindo que os usuários determinem suas análises e parâmetros sem escrever seu próprio código.	

Software	Linguagem	Interface Gráfica	Licença de Uso	Tamanho para download *	Versionamento	Dados Versionamento	Características	Vantagens/Virtudes	Deficiências
							novos parâmetros quando necessário.		
Analise Geral	50% Matlab, 22% Python, 28% Outros	78% Sim, 22% Não	71% Gratuito, 15% Pago, 14% Não especificado	Artefatos de tamanho pequeno. Media: 60.85 MB	64% GIT, 7% CVS, 29% Não especificado	A maioria dos repositórios contém commit's recentes. Commit's em 2018: 57% Total Contribuidores: 265	Apenas 36% especifica os requerimentos de sistema para utilização. 21% Multiplataforma, 15% Windows, 7% Linux, 57% Não especificado	Extensibilidade, testes, documentação, desempenho, reprodutibilidade.	

- **Tamanho para download –** esta informação foi adquirida acessando os websites dos próprios software e realizando o download dos respectivos.

Como pode-se observar, grande parte dos software para processamento de EEG são codificados em Matlab® (50%). A média do tamanho do artefato do software para download é de 60,85 Mb (megabytes de memória), mas vale ressaltar que o tamanho para download se refere apenas ao código fonte, por vezes sendo necessário instalar o executor que pode demandar alguns gigas de memória do computador no caso do Matlab®.

Boa parte dos software (78%) disponibilizam uma interface gráfica para que o usuário interaja com a ferramenta. Alguns ainda permitem a realização das atividades tanto via interface como por linha de comandos. Além do mais, 71% dos software analisados são gratuitos e de código aberto, o que permite que a comunidade contribua mais facilmente para sua extensão e desenvolvimento, tornando-os mais acessíveis para os estudantes (COMBRISSON et al., 2017).

Apenas uma pequena parcela dos software (36%) se preocupou em fornecer informações sobre os requisitos computacionais para rodarem suas aplicações. Por vezes, esta falta de informação pode estar relacionada ao fato dos software serem de código aberto e utilizarem o Matlab®, no qual as configurações computacionais necessárias seriam as mesmas requeridas pela versão do Matlab® em questão. Como os dados em um registro de EEG diferem em tamanho e dependem do comprimento do evento ou tarefa em estudo, as análises/processamentos podem necessitar de sistemas com poder computacional acima da média (RAMACHANDRAN et al., 2018).

Percebe-se, também, que os software se preocuparam com as seguintes virtudes: extensibilidade, testes, documentação, desempenho e reprodutibilidade.

A reprodutibilidade dos processamentos e das análises realizadas pelos pesquisadores ainda permanece limitada. Entre as análises automatizadas, o código de computador que os pesquisadores usam para gerar dados pode vincular vários pacotes de software, definir parâmetros que são relatados apenas parcialmente e não necessariamente serem salvos ou compartilhados (LEVIN et al., 2018). Por esta razão, alguns dos software pesquisados registram os parâmetros e as operações realizadas para que os experimentos possam ser reproduzidos por outros grupos.

Comumente, os pacotes de software que se preocupam com o desempenho são os que fornecem processamento e análise de dados em tempo real, podendo ser

encontrados, predominantemente, nas áreas de *neurofeedback* e pesquisas de interface cérebro-computador (BCI) (ESCH et al., 2018). Esta preocupação com o desempenho vem devido à latência, que é um dos fatores mais importantes para todos os sistemas de tempo real, sendo o tempo de resposta do processamento desde o recebimento dos sinais biológicos, a extração de características para classificação e a produção de alguma saída, fornecendo assim um feedback para o usuário (KNOPP et al., 2017).

Percebe-se, por fim, uma tendência na criação/utilização de dispositivos portáteis, operados por pequenos notebooks ou *smartphones,* para monitorar e cuidar da saúde de pacientes, permitindo, assim, uma alta usabilidade como, por exemplo, na aquisição de dados de EEG durante o movimento natural ao ar livre, como caminhada ou ciclismo (BLUM et al., 2017). Com este intuito, várias aplicações são criadas para que os *smartphones* possam processar as informações recebidas dos dispositivos vestíveis, como é o caso da SCALA (*Signal ProCessing and CLassification on Android*) (BLUM et al., 2017), que é uma aplicação desenvolvida para a plataforma Android (Sistema operacional de *smartphone*), sendo uma solução modular de BCI para processamento de séries temporais de dados fisiológicos.

Existem vários software com propósitos distintos que perfazem algumas atividades semelhantes, executando por vezes algoritmos diferentes em plataformas ou linguagens distintas para alcançar um determinado objetivo em comum. Escolher dentre tantas opções de software para analisar o EEG não é uma tarefa fácil. Normalmente busca-se um software que seja de fácil utilização e que tenha uma boa confiabilidade. Nesta linha de raciocínio, surge o presente o trabalho que realiza a analise do sinal de EEG, extraindo alguns quantificadores de forma simplificada para o usuário, no qual ele necessita apenas preencher uma planilha com campos bem definidos no formato CSV e o software se encarregara de extrair os quantificadores que serão armazenados em um arquivo .MAT, e se o usuário quiser eles também poderão ser salvos em uma planilha do Excel.

CAPÍTULO 3 - MATERIAIS E MÉTODOS

Este capítulo é dedicado a mostrar e descrever, brevemente, as tecnologias e metodologias utilizadas durante o desenvolvimento do trabalho.

3.1 GIT

GIT é um sistema de controle de versão de arquivos que facilita o trabalho e a gestão de alterações dos mesmos, registrando as modificações que ocorrem, permitindo que desenvolvedores trabalhem no mesmo conjunto de arquivos sem o risco de suas alterações serem sobrescritas (SCHMITZ, 2015).

Um de seus grandes destaques é a criação de *branch,* que é uma cópia do projeto (arquivos) em um determinado momento. Esta *branch* pode seguir um desenvolvimento de novas funcionalidades em paralelo com a evolução do projeto em si e, apenas quando estas funcionalidades estiverem bem desenvolvidas e testadas, são integradas novamente no projeto (SCHMITZ, 2015).

O GIT define um conjunto enorme de regras para versionar os arquivos existindo várias empresas que seguem estas regras e disponibilizam estes serviços, como o *bitbucket* e o *github*.

3.1.1 Bitbucket

Mantido pela *Atlassian*, é gratuito para um número ilimitado de repositórios privados e para equipes pequenas de até 5 desenvolvedores. Há, ainda, os planos Padrão (US $ 2 / usuário / mês) ou Premium (US $ 5 / usuário / mês), para equipes grandes ou em crescimento (ATLASSIAN, 2018).

3.1.2 GitHub

O GitHub é gratuito para uso em projetos públicos e de código aberto, podendo, porém, trabalhar em repositórios privados com um plano pago (GITHUB, 2018).

3.2 PROCESSAMENTO DO EEG

3.2.1 Tecnologias utilizadas

3.2.1.1 Matlab®

Matlab® é um software interativo voltado para o cálculo numérico, processamento de sinais e construção de gráficos, sendo bastante utilizado devido a sua facilidade para manipular matrizes e vetores.

3.2.1.2 CSV

É uma extensão de arquivo que significa *Comma-separated values* - CSV (Valores Separados por Vírgula). É um tipo de arquivo que pode ser criado ou editado pelo Excel ou em qualquer outro editor de texto, por simplesmente ser um arquivo de texto no qual se separa os valores por virgula (ou ponto e vírgula). Nele cada virgula representa uma coluna, ou seja, em vez de armazenar informações em colunas, os arquivos CSV armazenam informações separadas por vírgulas (MICROSOFT, [s.d.]).

3.2.1.3 StarUML

É um software para modelagem ágil e concisa, o que possibilita a criação e edição de diagramas, que é uma representação visual geométrica simbólica de um modelo de software. Assim, o modelo de software pode ser representado em um ou mais diagramas com diferentes aspectos, como: estrutura hierárquica de classes, interação entre objetos, entre outros. Portanto, os diagramas consistem em elementos de visualização, que são representações visuais de um elemento de modelo (STARUML, 2018).

3.2.2 Metodologia

O primeiro passo foi compreender como o software de processamento do EEG existente funcionava, através de reuniões com o grupo de pesquisa, no qual foi

apresentado todos os passos e a sequência necessária para processar o EEG. O software existente foi construído inteiramente sobre a plataforma MATLAB®.

Logo em sequência, em uma análise mais aprofundada sobre o código fonte do software, foi identificado alguns pontos de melhorias, onde alguns trechos de códigos que se repetiam em várias funções, foram transformados em novas funções havendo reaproveitamento de código. Durante esta etapa de melhorias, algumas diretrizes foram definidas pelo grupo de pesquisa: o código fonte deve ser compatível/executável na versão do MATLAB® 2007; usar uma ferramenta de versionamento de código; e a cada melhoria e/ou reestruturação do código fonte deve ser repassada com os integrantes do grupo de interesse em questão e, se aprovada, o mesmo deverá ser submetido a testes de regressão.

Como ferramenta de versionamento de código fonte, foi escolhido o bitbucket pois ele permite criar um repositório privado de forma gratuita.

Identificou-se que o software necessitava de várias interações com o usuário para inserir os dados de entrada para processar o EEG, que por vezes, o usuário poderia digitar algum dado errado, tendo que refazer o processamento do EEG em questão novamente. Diante disso, surgiu a necessidade de automatizar algumas atividades, evitando-se ao máximo a interação com o usuário.

Para evitar que o usuário fica-se inserindo os dados um por um e mitigar as possibilidades de erro, criou-se um *template* de um arquivo com extensão .CSV (ANEXO B), para que o usuário pudesse preenche-lo uma única vez com todas as informações necessárias para o processamento dos EEG's, e assim o usuário precisaria inserir apenas um arquivo como entrada, e o software fica encarregado de processar todos os EEG's contidos no arquivo .CSV. Essa forma automatizada foi considerada como a **NOVA** versão do software.

Inicialmente os testes de regressão eram feitos de forma manual. Porém, com o aumento contínuo das melhorias e da necessidade de realizar os testes, fazê-los manualmente acabou se tornando dispendioso e bastante suscetível a erros. Assim, o grupo optou em desenvolver códigos para que estes testes fossem feitos de forma mais automática, desta maneira, criou-se dois testes de regressão, um utilizando o teste estatístico de Friedman e o outro utilizando o erro relativo. Testes estes que tem

o intuito de verificar se as variáveis de saída do código atual são iguais às variáveis de saída da versão antiga.

E por fim, para documentar e deixar mais fácil a compreensão e a manutenção do software de processamento, foi criado diagramas de execuções das principais funções. Estes diagramas foram construídos utilizando o starUML, e no **ANEXO A** pode-se visualizar estes diagramas.

3.3 VISUALIZAÇÃO DOS QUANTIFICADORES DO EEG

3.3.1 Tecnologias utilizadas

3.3.1.1 Matlab®

Matlab®, como já dito, é um software interativo voltado para o cálculo numérico, processamento de sinais e construção de gráficos, sendo bastante utilizado devido a sua facilidade para manipular matrizes e vetores.

3.3.1.2 Sloreta

O sLoreta – *Low Resolution Electromagnetic Tomography* (Tomografia eletromagnética de baixa resolução) teve suas primeiras publicações datadas de 1998 e 1999 (STRIK et al., 1998) (PASCUAL-MARQUI et al., 1999). Tais publicações versavam sobre o mapeamento paramétrico estatístico de alta resolução para imagens tomográficas de atividade neuronal elétrica (LORETA). O objetivo desta plataforma é adotar métodos de inferência estatística para a localização da função cerebral, como utilizado nos estudos de PET e fMRI *(functional magnetic resonance imaging*), aplicando esta metodologia em alta resolução de tempo, variando as imagens LORETA. Ele é um pacote acadêmico gratuito, destinado ao uso em pesquisa. O seu uso clínico e comercial são estritamente proibidos (SLORETA, 2017).

3.3.2 Metodologia

Incialmente realizou-se uma pesquisa bibliográfica para verificar os tipos de visualizações de EEG, em um segundo momento verificou-se que a visualização topográfica do EEG seria mais pertinente ao trabalho, e então iniciou-se uma pesquisa sobre softwares que geravam estas visualizações. Dentre todos os softwares pesquisados encontrou-se o sLoreta que possui uma interface amigável e é relativamente fácil de se utilizar. Mas, como o software de processamento do EEG foi escrito em Matlab®, a integração com sLoreta ficou muito manual exigindo a execução de vários passos. Para minimizar a utilização destes passos manuais e ter uma integração mais rápida, o grupo de pesquisa desenvolveu uma função em Matlab® para gerar as visualizações topográficas.

Logo em sequência, criou-se um tutorial de como instalar e gerar as visualizações no sLoreta e no Matlab®, e realizou-se a comparação entre estas ferramentas para verificar qual seria mais adequada para se utilizar no trabalho.

A partir da função criada no Matlab®, o grupo de pesquisa gerou outras visualizações para facilitar a comparação dos resultados entre os EEG's, assim como outras visualizações para ilustrar a evolução temporal dos quantificadores extraídos.

3.4 SISTEMA PARA CADASTRO DOS EEG COLETADOS

3.4.1 Tecnologias utilizadas

3.4.1.1 Java

Java é uma linguagem de programação e plataforma de computação lançada pela Sun Microsystems em 1995. Devido à sua confiabilidade, agilidade e segurança, várias empresas o escolhem para desenvolver seus aplicativos e sites. Por esta razão o java está presente em quase toda parte como, por exemplo, em: laptops, datacenters, videogames, supercomputadores científicos, celulares e Internet (ORACLE, [s.d.]).

3.4.1.2 Maven

O Apache Maven é uma ferramenta de gerenciamento e compreensão de projetos de software. Baseado no conceito de um modelo de objeto de projeto (*POM - Project Object Model*). Ele gerencia a construção de projetos, desde as suas dependências, até a geração do artefato final do projeto. Podendo ser utilizado para criar e gerenciar qualquer projeto baseado em Java. Tendo como objetivo, facilitar o trabalho cotidiano dos desenvolvedores e ajudar na compreensão de qualquer projeto baseado em Java (APACHE MAVEN PROJECT, 2019).

3.4.1.3 JavaScript

JavaScript é uma linguagem de programação voltada para a WEB que pode atualizar e mudar tanto o HTML como o CSS. Não se restringindo apenas a isto, pode, também, calcular, manipular e validar dados. Ele programa o comportamento de páginas da web (W3SCHOOLS [A], [s.d.]).

3.4.1.4 node.js

Node.js é um ambiente de servidor de código aberto, gratuito e de multiplataforma (podendo ser executado tanto em: Windows, Linux, Unix, Mac OS X etc.). Ele utiliza javascript em seu servidor, podendo gerar páginas com conteúdo dinâmico, manipular arquivos no servidor, coletar dados de formulário, além de adicionar, excluir e modificar dados em banco de dados (W3SCHOOLS [B], [s.d.]).

3.4.1.5 Webpack

O webpack é um empacotador de módulos estáticos para aplicações JavaScript. Quando o webpack processa a aplicação, ele cria internamente um gráfico de dependências que mapeia todos os módulos de que o projeto precisa e gera um ou mais pacotes configuráveis (WEBPACK, [s.d.]).

3.4.1.6 React

O React é uma biblioteca JavaScript declarativa, eficiente e flexível para criar interfaces com o usuário. Ele permite compor interfaces de usuário complexas a partir de pequenos e isolados códigos chamados “componentes” (REACT, [s.d.]).

3.4.1.7 CSS

CSS significa *Cascading Style Sheets* (folhas de estilo em cascata). Ele descreve como os elementos HTML devem ser exibidos na tela, no papel ou em outras mídias, permitindo controlar o layout de várias páginas da web de uma só vez através de folhas de estilo externas armazenadas em arquivos CSS (W3SCHOOLS [C], [s.d.]).

3.4.1.8 HTML

Hyper Text Markup Language – HTML (Linguagem de marcação de hipertexto) é a linguagem de marcação padrão para criação de páginas da Web. Ela descreve a estrutura das páginas da Web usando marcações (*tags*), que são blocos de construção de páginas HTML. As *tags* marcam partes de conteúdo como "título", "parágrafo", "tabela" e assim por diante. Os navegadores não exibem as *tags* HTML, mas as usam para renderizar o conteúdo da página (W3SCHOOLS [D], [s.d.]) .

3.4.1.9 Spring-boot

O Spring-Boot facilita a criação independente de aplicações baseadas em *Spring* a nível de produção que se deseja executar. Ele adota uma visão opinativa da plataforma *Spring* e de bibliotecas de terceiros, para que se possa criar novas aplicações com o mínimo de esforço possível. A maioria dos aplicativos *Spring Boot* precisa de uma configuração de *Spring* muito pequena (WEBB et al., 2018).

Pode-se utilizar o *Spring Boot* para criar aplicações Java, que podem ser iniciados usando o java -jar ou mais implementações tradicionais usando o *war*. Fornece, também, uma ferramenta de linha de comando que executa “*scripts spring*” (WEBB et al., 2018).

Seus objetivos são (WEBB et al., 2018):

- Fornecer uma experiência de se criar aplicações de maneira mais rápida e amplamente acessível para todo o desenvolvimento do *Spring*.
- Ser opinativo, mas ao mesmo tempo flexível, com relação aos padrões, pois os requisitos das aplicações começam a divergir dos padrões.
- Fornecer vários recursos não funcionais que são comuns a grandes classes de projetos (como servidores incorporados, segurança, métricas, verificações de integridade e configuração externalizada).
- Não realizar qualquer geração de código, nem exigir qualquer requisito para a configuração XML.

3.4.1.10 Banco de Dados

Todos possuem várias informações que devem ser guardadas e gerenciadas para os requisitos de sua vida cotidiana. Estas várias informações devem, também, permanecer disponíveis para aqueles que delas necessitarem, podendo serem guardadas em caixas de papelão, armários, computadores, etc. No entanto, é importante que sejam definidas regras para se armazenar e recuperar essas informações, de forma que seu acesso e manuseio sejam fácil e ágil (ORACLE, 2017).

Portanto, banco de dados é uma coleção organizada de informações tratadas como uma unidade. A finalidade de um banco de dados é coletar, armazenar e recuperar informações relacionadas para uso por aplicativos de banco de dados. Para gerir estes dados é necessário um sistema de gerenciamento de banco de dados (DBMS - *Database Management System*), que é um software que controla o armazenamento, a organização e a recuperação de dados (ORACLE, 2017).

3.4.1.11 SQL

SQL (*Structured Query Language*) significa Linguagem de Consulta Estruturada. Utilizada para acessar e manipular bancos de dados, tornou-se um padrão do *American National Standards Institute* (ANSI) em 1986 e da *International Organization for Standardization* (ISO) em 1987 (W3SCHOOLS [E], [s.d.]).

3.4.1.12 H2

O H2 é um banco de dados em memória, de código aberto e rápido, aplicação de console baseado em browser, desenvolvido em java (H2, [s.d.]).

3.4.1.13 PostgreSql

O PostgreSQL é um sistema de banco de dados relacional, de código fonte aberto que usa e estende a linguagem SQL, combinando vários outros inúmeros recursos que permitem armazenar os dados com segurança e dimensionar as cargas de trabalho. Ele foi criado em 1986 como parte do projeto POSTGRES da Universidade da Califórnia, em Berkeley, e têm mais de 30 anos de desenvolvimento ativo na plataforma central (POSTGRESQL, [s.d.]).

3.4.1.14 Hibernate

O Hibernate é uma ferramenta que implementa o conceito de mapeamento objeto-relacional (ORM - *Object / Relational Mapping*), permitindo aos desenvolvedores escrever mais facilmente aplicações, cujos dados necessitam existir após o processo de aplicação. Ele está preocupado com a persistência dos dados, uma vez que se aplica a bancos de dados relacionais (via JDBC - *Java Database Connectivity*) (HIBERNATE, [s.d.]).

3.4.1.15 Heroku

Heroku® é uma plataforma em nuvem baseada em um sistema de gerenciamento de containers. Disponibiliza serviços de computação em nuvem, popularmente chamado de PaaS (*platform-as-a-service*), plataforma como serviço, fornecendo um ambiente pronto para receber aplicações. Aplicações estas que podem ser desenvolvidas em Ruby, Node.js, Java, Python, Clojure, Scala, Go e PHP (HEROKU, [s.d.]).

Quando se cria uma aplicação no Heroku® ele associa um novo GIT remoto tipicamente chamado heroku com o repositório local GIT para a aplicação criada. Assim, o container disponibilizado consegue construir (compilar e gerar o artefato) para executar e disponibilizar a aplicação (HEROKU, [s.d.]).

3.4.2 Metodologia

Inicialmente, realizou-se uma pesquisa bibliográfica para pesquisar sistemas/plataformas de EEG disponíveis online, no qual se levantou alguns aspectos positivos e negativos destas plataformas. Em um segundo momento, foi realizado um levantamento de requisitos com base nas informações obtidas nas plataformas online, e também pegou-se os formulários de requerimento de exames e os formulários preenchidos nas coletas de EEG realizadas pelo grupo de pesquisa, e a partir destas informações foi estabelecido quais os dados que deveriam ser armazenados no banco de dados. Com isso foi montado o diagrama de entidades e relacionamentos do banco da aplicação.

Prosseguindo, foi realizado uma pesquisa bibliográfica para verificar quais seriam as melhores ferramentas para se desenvolver esta plataforma web para cadastro dos exames coletados. Diante da grande gama de ferramentas e da experiência dos integrantes do grupo de pesquisa escolheu-se utilizar o Spring-Boot devido a sua facilidade e agilidade para construir novas aplicações, disponibilizando vários recursos pré configurados, e como linguagem de programação para estruturar a aplicação utilizou-se o JAVA.

Como ferramenta de versionamento de código fonte, escolheu-se o github pela sua popularidade e facilidade de uso. Para gerenciar as dependências e gerar o artefato de *deploy* do projeto, utilizou-se o Maven no qual todas as configurações da construção do projeto foram definidas no seu arquivo de controle chamado *POM.xml*.

Como arquitetura de software adotou-se o MVC (*MODEL, VIEW e CONTROLLER*), que é projetado para separar a lógica de negócio da lógica de apresentação/visualização. Composto por três componentes: *Model, View e Controller*. No qual o *MODEL* é responsável pela manipulação dos dados como: leitura, escrita e validação. Já a *VIEW* é a camada de interação com o usuário fazendo a exibição dos dados. E por fim o *CONTROLLER* recebe todas as requisições do usuário, controlando qual *MODEL* deve ser acessado e qual *VIEW* deve ser exibida (WINK, 2016).

A camada do *MODEL* e *CONTROLLER* foi desenvolvido com Spring-Boot e JAVA, e na camada de persistência com o banco de dados adotou-se o hibernate, que

seguindo os seus padrões, caso haja a necessidade da troca do banco de dados por outro, não é necessário fazer uma refatoração na camada do *MODEL.*

Quando uma aplicação está sendo desenvolvida comumente é necessário mais de um ambiente para testa-la, ou seja, um ambiente no qual o desenvolvedor utiliza para testar os seus desenvolvimentos, e outro ambiente que pode ser de homologação e/ou produção, onde os usuários finais estarão acessando a ferramenta. Desta maneira o hibernate auxiliou bastante, pois no ambiente de desenvolvimento utilizou-se o banco de dados em memória H2 e em produção utilizou-se o postgreSql.

Já para a construção da camada *VIEW* da arquitetura MVC, utilizou-se o REACT que permitiu desenvolver essa camada em componentes, proporcionando o reaproveitamento de vários trechos de código. Para montagem e controle dos formulários utilizou-se o ReactRedux.

Como *template/layout* da aplicação da camada *VIEW* utilizou-se o AdminLTE2 que já possui alguns padrões de HTML e CSS bem definidos para deixar a aplicação com um visual mais amigável. Para visualizar este *template* basta acessar o seguinte link https://adminlte.io/themes/AdminLTE/index2.html.

Para gerenciar as dependências desta camada, utilizou-se o node.js através do arquivo package.json que defini todas as dependências necessárias à aplicação. E para gerar e controlar os arquivos de saída desta camada utilizou-se o webpack.

Por fim, para expor a aplicação para o ambiente de homologação/produção utilizou-se o Heroku que disponibiliza o serviço de plataforma, hospedando a aplicação e disponibilizando um domínio para acessa-la através da internet de forma gratuita, mas com algumas limitações.

3.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as ferramentas/tecnologias utilizadas, assim como realizou-se uma descrição rápida da metodologia usada para o desenvolvimento deste trabalho.

CAPÍTULO 4 - PROCESSAMENTO DO EEG

Este capítulo visa expor, brevemente, como o software de processamento funcionava e como atualmente ele trabalha.

4.1 MODELO ANTIGO

Composto de 6 (seis) programas principais que perfazem o processamento do EEG, desde a conversão do arquivo .PLG até a extração de características do sinal. Além de mais 4 funções que auxiliam no processamento dos sinais.

Abaixo é descrito, ligeiramente, a função de cada programa.

4.1.1 Programas

4.1.1.1 p1_conversor

Converte o arquivo gerado pelo aparelho que registra o EEG (*BrainNet BNT-EEG* - taxa de amostragem de 240 Hz), que está com a extensão PLG, para um arquivo de variáveis legíveis pelo Matlab®, que possui a extensão .MAT. Esta conversão gera dados de um exame de Eletroencefalografia de N canais (comumente N canais do padrão 10-20 sendo o 21° canal foto).

4.1.1.2 p2_epocas

Através do arquivo .MAT, gerado no programa anterior (p1_conversor), ele solicita uma entrada manual de duração das épocas (comumente 2 segundos) e o tempo em que se inicia cada época (comumente 10 épocas). Separa as épocas dos exames conforme os parâmetros informados e adiciona as épocas separadas no mesmo arquivo .MAT.

4.1.1.3 p3_validaeeg

Este programa objetiva validar o sinal EEG dividido em épocas, gerado pelo programa anterior (p2_epocas), conforme o nível de ruído encontrado nos eletrodos. Dessa forma, o programa calcula, baseado na densidade espectral de potência, o nível do sinal em porcentagem na faixa 1-40Hz e o nível de ruído na faixa 58-62Hz. Para um sinal ser completamente aprovado todos os 20 eletrodos devem seguir a seguinte condição: que o valor máximo de potência da faixa do ruído seja metade ou menor que o valor máximo de potência da faixa do sinal. Caso essa condição não seja válida para até no máximo três eletrodos o sinal é aceito, porém, deve-se verificar quais eletrodos não foram aprovados. Caso a condição inicial não seja válida para mais de quatro eletrodos, o sinal é reprovado

4.1.1.4 p4_pcp

Através do arquivo .MAT, alterado após separar as épocas (p2_epocas), este código calcula a **potência** de cada faixa e adiciona no arquivo selecionado. A potência é calculada para as seguintes faixas de frequência: delta, teta, alfa, beta, gama, supergama, ruído. Nesse programa, o sinal EEG em épocas é carregado e, em seguida, para cada época definida, são realizados os seguintes cálculos:

- Criação de um vetor de frequência a partir da taxa de amostragem (Hz) do sinal;
- Remoção do nível DC do sinal, colocando-o em média zero;
- Cálculo da autocorrelação do sinal com Taumax (indicado pelo usuário ou o valor de 1.67 x frequência de amostragem) usando a função denominada 'autocorr_matriz';
- Calcula a Transformada de Fourier (TF) do sinal usando a função denominada 'Calcula_DFT_Matriz_V3';
- Zera-se os valores da amplitude do módulo da resultante da TF nas faixas de frequências 0-1Hz e acima de 100 Hz;
- Realização da normalização do sinal resultante usando a função 'integralMod';
- Cálculo da potência do sinal normalizado usando a função 'integralMod';
- Divisão do sinal conforme os ritmos do sinal EEG. Cada ritmo tem uma contribuição predominante em uma faixa de frequência. Ritmos analisados:

Delta (1-3.5Hz), Teta (3.5 - 7.5Hz), Alfa (7.5-12.5Hz), Beta (12.5-30Hz), Gama (30-58Hz e 62-80Hz), Supergama (80-100Hz) e Ruído (58-62Hz);

- Cálculo da potência relativa de cada faixa/ritmo.

4.1.1.5 p5_fm

Através do arquivo .MAT, alterado após separar as épocas (p2_epocas), este código calcula a **frequência mediana** e adiciona no arquivo selecionado. São calculadas as FREQUÊNCIAS MEDIANAS para as faixas de frequência: delta, teta, alfa, beta, gama, supergama e ruído. Nesse programa o sinal EEG em épocas é carregado e, em seguida, para cada época definida, são realizados os seguintes cálculos:

- Criação de um vetor de frequência a partir da taxa de amostragem (Hz) do sinal;
- Remoção do nível DC do sinal colocando-o em média zero
- Cálculo da autocorrelação do sinal com Taumax (indicado pelo usuário ou o valor de 1.67 x frequência de amostragem) usando a função denominada 'autocorr_matriz'.
- Cálculo da Transformada de Fourier (TF) do sinal, usando a função denominada 'Calcula_DFT_Matriz_V3'.
- Zera-se os valores da amplitude do módulo da resultante da TF nas faixas de frequência: 0-1Hz e acima de 100 Hz.
- Realiza-se a normalização do sinal resultante usando a função 'integralMod'.
- Calcula-se a potência do sinal normalizado usando a função 'integralMod'.
- Divide-se o sinal conforme os ritmos do sinal EEG. Cada ritmo tem uma contribuição predominante em uma faixa de frequência. Ritmos analisados: Delta (1-3.5Hz), Teta (3.5 - 7.5Hz), Alfa (7.5-12.5Hz), Beta (12.5-30Hz), Gama (30-58Hz e 62-80Hz), Supergama (80-100Hz) e Ruído (58-62Hz).
- Calcula-se a frequência mediana relativa de cada faixa/ ritmo.

4.1.1.6 p6_novo_cor

Através do arquivo .MAT, alterado após separar as épocas (p2_epocas), este código calcula a **coerência** entre os eletrodos do hemisfério esquerdo com o direito e adiciona no arquivo selecionado. Nesse programa o sinal EEG em épocas é submetido à análise da coerência usando uma função do Matlab® denominada 'mscohere'. Para isso foram criadas duas matrizes, uma do lado esquerdo e outra do lado direito. A matriz do lado esquerdo corresponde aos eletrodos do lado esquerdo do escalpo cerebral (FP1, F7, F3, T3, C3, T5, P3 e O1) enquanto a matriz do lado direito corresponde aos eletrodos do lado direito do escalpo cerebral (FP2, F8, F4, T4, C4, T6, P4 e O2). A partir dessas matrizes calcula-se as respectivas coerências de cada par de eletrodos simétricos: FP1-FP2, F7-F8,F3-F4,T3-T4,C3-C4,T5-T6,P3-P4,O1-O2.

4.1.1.7 Calcula_DFT_Matriz_V3

Calcula a transformada de Fourier usando a teoria.

4.1.1.8 autocorr_matriz

Calcula a autocorrelção normalizada de uma matriz.

4.1.1.9 integralMod

Calcula o resultado numérico da integral de um vetor pelo método dos trapézios.

4.1.2 Versionamento

O versionamento do código era feito através da centralização do mesmo em uma única pessoa, que ficava responsável por adicionar as novas funcionalidades, tomando o cuidado para não perder códigos quando outro desenvolvedor fazia alterações.

Quando as alterações eram concluídas os desenvolvedores se reuniam para fazer a junção dos códigos que eles alteravam, registrando estas alterações através de comentários no próprio código fonte.

4.2 MODELO NOVO

Para se desenvolver o novo modelo a primeira premissa era de que os códigos já existentes continuassem funcionando da mesma forma, sem nenhuma alteração nos resultados encontrados por eles. Assim, a primeira etapa do desenvolvimento do novo modelo foi quebrar os códigos em funções que pudessem ser reutilizadas.

No **ANEXO A** é ilustrado os diagramas de execução dos principais programas da nova versão do software de processamento do EEG, no qual fica evidente que a interação do usuário com o *software,* utilizando o modelo antigo (P1 a P6) é muito maior do que com o modelo automatizado (processRecords).

4.2.1 Versionamento

Durante o desenvolvimento dessa nova versão existiam alguns desenvolvimentos concomitantes, além do mais, a reestruturação do código seria grande e para registrar essas alterações de forma mais eficiente e facilitar a volta do código para versões anteriores, optou-se em utilizar o controle de versão do GIT através do bitbucket com um repositório privado, contendo apenas os desenvolvedores da equipe.

Outra vantagem de se utilizar esta abordagem é que geramos **TAG (uma etiqueta/marca),** a qual marcamos as versões estáveis do código para que a equipe de pesquisa pudesse ir utilizando até que as novas funcionalidades ficassem prontas e uma nova TAG fosse gerada. Desta forma, os pesquisadores baixavam a TAG de forma mais independente, sem a necessidade de que uma pessoa sozinha ficasse encarregada de cuidar do código estável.

4.2.2 Programas

A premissa é que o funcionamento dos códigos anteriores continuasse a mesma. Desta forma, os códigos anteriores não mudaram seu objetivo, apenas a forma interna como eles processavam é que foi alterada. Destarte, abaixo será mostrado apenas os novos programas criados.

4.2.2.1 readFilePLG

Realiza a leitura de um arquivo de extensão plg (*.plg), referente a um exame de eletroencefalograma (EEG), para que os dados do exame possam ser processados pelo matlab®.

4.2.2.2 customizeRecordEEG

Personaliza o registro do EEG, traduzindo os nomes dos canais conforme a variável de tradução e colocando os canais ordenados de acordo com uma lista de ordem passada como parâmetro.

4.2.2.3 translateChannelNamesByPrefixComparator

Traduz os nomes dos canais conforme mapeamento passado como parâmetro, onde a primeira coluna é o prefixo a ser traduzido e a segunda coluna é a tradução.

Observação: o mapeamento não precisa ser um para um, ou seja, a tradução pode ser para apenas um único valor, mesmo se a matriz de entrada contiver n linhas.

4.2.2.4 sortChannelsByListNames

Ordena os canais pelo nome (xn, nameChannels, typeChannels), de acordo com a lista/ordem passada como parâmetro.

4.2.2.5 splitChannels

Divide os canais de acordo com a lista passada como parâmetro - retornando uma lista com os canais definidos no parâmetro e outra lista com os canais restantes.

4.2.2.6 findTypeChannel

Retorna a posição da primeira correspondência encontrada para o tipo de canal passado como argumento, caso não exista retorna -1.

4.2.2.7 findNameChannel

Retorna a posição da primeira correspondência encontrada para o nome de canal passado como argumento, caso não exista retorna -1

4.2.2.8 verifyAllValuesInInterval

Verifica se todos os valores da matriz, passada como argumento, estão no intervalo informado como parâmetro.

4.2.2.9 epochSeparator

Função responsável pela separação das épocas do EEG. Recebe como argumento o tamanho da época e uma string no formato: MM:SS|MM:SS|MM:SS onde cada MM:SS representa o início de cada época. Muito semelhante ao que o programa p2_epocas do modelo anterior fazia.

4.2.2.10 validateChannels

Este programa objetiva validar o sinal EEG conforme o nível de ruído verificado nos eletrodos. Dessa forma, o programa calcula, baseado na densidade espectral de potência, o nível do sinal em porcentagem na faixa 1-40Hz e o nível de ruído na faixa 58-62Hz. Para um sinal ser completamente aprovado os eletrodos devem seguir a condição que o valor máximo de potência da faixa do ruído seja

metade ou menos do valor máximo de potência da faixa do sinal. Muito semelhante ao que o programa p3_validaeeg fazia.

4.2.2.11 computeMeanZeroPowerSignalRXTXFtestXMXFxmFilteredAndNormAndFreq

Este programa objetiva calcular a Media Zero, Potência do Sinal, Rx, Tx, XM, XF, fTest, XM filtrado, M Normalizado.

4.2.2.12 computePCPsaveInMatAndExcel

Este programa objetiva calcular a porcentagem da contribuição de potência - PCP para cada faixa de frequência definida (Delta, Teta, Alfa, Beta, Gama, Super Gama e Ruído), e salvar os valores calculados no arquivo .MAT passado como referência. Caso o FLAG de gerar excel estiver marcado, gera-se um arquivo do excel com os valores calculados.

4.2.2.13 computeFMsaveInMatAndExcel

Este programa objetiva calcular a Frequência Mediana para cada faixa de frequência definida (Delta, Teta, Alfa, Beta, Gama, Super Gama e Ruído) e salvar os valores calculados no arquivo .MAT, passado como referência. Caso o FLAG de gerar excel estiver marcado, gera-se um arquivo do excel com os valores calculados.

4.2.2.14 computeCoherenceLeftWithRightHeadSaveMatAndExcel

Nesse programa o sinal EEG em épocas é submetido à análise da coerência usando uma função do matlab® denominada 'mscohere'. Para isso foram criadas duas matrizes, leftHEAD e rightHEAD. A matriz leftHEAD corresponde aos eletrodos do lado esquerdo do escalpo cerebral (FP1,F7,F3,T3,C3,T5,P3 e O1) enquanto a matriz rightHEAD corresponde aos eletrodos do lado direito do escalpo cerebral (FP2,F8,F4,T4,C4,T6,P4 e O2). A partir dessas matrizes foram calculadas as respectivas coerências de cada par de eletrodos simétricos: FP1-FP2, F7-F8,F3-F4,T3-T4,C3-C4,T5-T6,P3-P4,O1-O2. Salva-se os valores calculados no arquivo

.MAT, passado como referência e, se o FLAG de gerar excel estiver marcado, gera-se um arquivo do excel com os valores calculados.

4.2.2.15 processRecords

Faz a leitura de um arquivo de CSV no seguinte formato (o exemplo deste arquivo pode ser visualizado no **ANEXO B**):

- Coluna A (Nome Arquivo) = deve-se inserir nessa coluna o nome do arquivo .PLG, o qual será processado (Atenção: "SEM" a extensão, apenas o nome);
- Coluna B (Duração Épocas) = duração das épocas "em segundos";
- Coluna C (Qtd Épocas) = insira o número de épocas a ser separadas;
- Coluna D a M (EpX) = o tempo onde se inicia a época, deve seguir o formato mm:ss (minutos:segundos);

Nota: a quantidade de épocas aqui é apenas ilustrativa. O que determinará a quantidade de colunas de épocas será o valor do campo "Qtd Épocas";

- Coluna N (Qtd Ruidosos) = Quantidade de canais ruidosos levantados pelo médico. Mais que 3 canais o exame deve ser descartado;

Nota: a quantidade de canais ruidosos aqui é apenas ilustrativa. O que determinará a quantidade de colunas de canais ruidosos será o valor do campo "Qtd Ruidosos";

- Coluna O a Q (Canal Ruidoso X) = Deve ser informado o nome do canal ruidoso que pode ser: FP1, FP2, F7, F3, FZ, F4, F8, T3, C3, CZ, C4, T4, T5, P3, PZ, P4, T6, O1 ,OZ, O2;
- Coluna R (Normal/Coma) = deve-se informar se o exame é Normal (100Hz) ou Coma (30Hz). Qualquer palavra digitada diferente de "COMA" será considerada como NORMAL;
- Coluna S (Filtro) = informa qual a frequência máxima para processamento do exame;
- Coluna T (Nome Saída) = informa qual o nome do arquivo .MAT que deve ser gerado ao se processar a linha;

- Coluna U (Gera Excel) = palavra (SIM/NAO) informando se deve gerar ou não o arquivo Excel para o PCP, FM e COERENCIA. Utilizar NAO para que o processamento seja mais rápido.

Cada linha deste arquivo .CSV representa um exame a ser processado, no qual será gravado um arquivo .MAT para cada exame com os cálculos realizados de acordo com os parâmetros informados.

Os cálculos realizados são:

- Conversão do arquivo .PLG em .MAT;
- Separação de Épocas;
- Validação do Exame, verificando se o ruído [58~62] é maior do que o valor do sinal;
- Cálculo do PCP;
- Cálculo da FM;
- Cálculo da Coerência hemisfério esquerdo com o hemisfério direito;
- Cálculo da Variação da Potência Cerebral (VPC);
- Cálculo da porcentagem de segmentos não gaussianos (PSNG);
- Cálculo da porcentagem de segmentos não estacionários (PSNE).

Ao final da execução do código é gerado um arquivo com o prefixo Result_X, onde X é o nome do arquivo de entrada. Este arquivo será salvo no mesmo diretório do arquivo de entrada, assim como todos os .MAT gerados para cada .PLG.

Nota: todos os arquivos .PLG a serem processados devem estar no mesmo diretório do arquivo .CSV de entrada.

O arquivo Result_X.CSV tem o mesmo conteúdo que o arquivo .CSV passado como parâmetro, com adição das seguintes colunas (o exemplo deste arquivo pode ser visualizado no **ANEXO C**):

- Coluna V (Conversão?) = conterá a palavra (SIM/NAO) informando se a conversão do .PLG para .MAT ocorreu com sucesso;

- Coluna W (Canais Ruidosos) = Conterá os nomes de canais que apresentarão ruído na validação por software;

- Coluna X [Exame Válido (POUCO RUIDO?)] = conterá a palavra (SIM/NAO) informando se o exame contém pouco ruído, ou seja, se possui "até" 3 canais ruidosos;
- Coluna Y (PCP) = conterá a palavra (SIM/NAO) informando se o Cálculo do PCP foi realizado com sucesso;
- Coluna Z (Frequência Mediana) = conterá a palavra (SIM/NAO) informando se o Cálculo da Frequência Mediana foi realizada com sucesso;
- Coluna AA (Coerência) = conterá a palavra (SIM/NAO) informando se o Cálculo da Coerência foi realizada com sucesso;
- Coluna AB (FreqMax Adotada) = conterá o valor da freqMax (frequência máxima) adotada nos cálculos.

4.2.2.16 readFileCSVForProcessRecords

Faz a leitura de um arquivo CSV considerando a mesma estrutura informada para o programa processRecords.

4.2.2.17 saveFileMAT

Faz a conversão do arquivo .PLG e salva um arquivo .MAT com os cálculos de P1 a P6.

4.2.2.18 updateFreqMaxAccordingWithBands

Retorna as novas bandas/faixas de acordo com a frequência máxima do exame, assim como o nome dessas bandas e a nova frequência máxima.

Elimina as bandas/faixas de frequências que não se enquadram completamente dentro da frequência máxima. Assim, se a frequência máxima está no meio de uma faixa de frequência, o novo valor da frequência máxima será igual ao valor inicial dessa faixa de frequência.

A frequência máxima não pode ser maior do que metade da frequência de amostragem e, também, não pode ser maior do que 100. Caso for configurado em

100, as faixas de frequências que irão permanecer serão as que tiverem o valor final da faixa, menor do que a frequência máxima nova.

4.2.2.19 groupEpochsByFrequences

Este programa pega os valores de cada frequência de cada época e monta um *cell* (tipo de dado no Matlab®) final de todas as frequências juntando as épocas.

4.2.2.20 brainPowerVariation

Calcula a Variação da Potência Cerebral.

4.2.2.21 MRT

Realiza o Teste Razão Média (MRT *Mean Ratio Test*). Este teste compara as funções de distribuição espectral dos subconjuntos de amostras a fim de observar se a função é ou não estacionária (DESTRO-FILHO; FARIA, 2018).

4.2.2.22 myfunc

Realiza teste não paramétrico Jarque-Bera (JB), que possui ótimas propriedades assintóticas de poder e bom desempenho para amostras finitas. O procedimento para a construção do teste de gaussianidade usa o princípio do método multiplicador de Lagrange (LM - *Lagrange Multiplier*) (DESTRO-FILHO; FARIA, 2018).

4.2.2.23 psngAndPsne

Calcula a porcentagem de segmentos não gaussianos – PSNG, que é o resultado retornado pelo teste JB. Calcula também a porcentagem de segmentos não estacionários – PSNE. que é o resultado retornado pelo teste MRT.

4.2.2.24 isEqualsErrorRelative

Verifica se as duas variáveis passadas como parâmetro são iguais, aceitando um limiar de erro relativo que também é passado como parâmetro.

4.2.2.25 isEqualsExamsProcessedErrorRelative

Verifica se dois exames processados são iguais através de seus arquivos .MAT gerados, considerando um determinado erro relativo aceitável.

4.2.2.26 isEqualsFriedman

Verifica se as duas variáveis passadas como parâmetro são iguais, comparando linha a linha através do teste Friedman.

4.2.2.27 validateFriedmanCellVector

Verifica se o vector de *cell (*Tipo de dado do Matlab®*)* A e B são iguais, através do teste estatístico de Friedman.

4.2.2.28 isEqualsExamsProcessedFriedman

Verifica se dois exames processados são iguais, através de seus arquivos .MAT gerados, considerando um determinado P valor aceitável.

4.2.2.29 testCompareExameByCsv

Faz a leitura de um arquivo CSV, onde a primeira coluna representa o arquivo de referência e a segunda coluna representa o arquivo a ser comparado. No final gera-se dois arquivos de resultado mostrando, linha a linha, o resultado da comparação pelo erro relativo e pelo teste de Friedman.

4.3 DESCRIÇÃO DAS VARIAVEIS IMPORTANTES

Abaixo é realizado a descrição das variáveis que são salvas após rodar os programas de P1 a P6 (modelo antigo) ou a execução de processRecords (modelo automatizado).

4.3.1 nameChannels (vetor de cellstr)

Vetor de tamanho L (L = quantidade de canais EEG do registro [comumente L=numberChannels=20]). Os nomes dos canais têm correspondência direta com "xn" e "typeChannels". A ordem dos canais é:

Fp1 Fp2 F7 F3 Fz F4 F8 T3 C3 Cz C4 T4 T5 P3 Pz P4 T6 O1 Oz O2

4.3.2 typeChannels (vetor cellstr)

Vetor de tamanho L (L = quantidade de canais EEG do registro [comumente L=numberChannels=20]). Este vetor contém o tipo de cada canal que pode assumir, por exemplo, os valores: EEG, FOTO, EKG, etc. Os tipos dos canais têm correspondência direta com "xn" e "nameChannels".

4.3.3 xn (matriz de escalares)

Matriz de tamanho LxN que contém o registro EEG convertido no formato .mat, onde as L linhas representam os canais que são identificados no vetor "nameChannels"; e as N colunas representam o tempo discreto.

Cada valor está em uV e representa a amplitude daquela amostra. Dimensão: L=numberChannels linhas e N colunas.

4.3.4 numberChannels (escalar)

Quantidade de canais na matriz "xn", numberChannels é igual a L.

4.3.5 N (escalar)

Quantidade de amostras por canal.

4.3.6 T (escalar)

Período de amostragem (baseado na taxa de amostragem) em segundos.

4.3.7 Fa (escalar)

Frequência de amostragem da coleta em Hz.

4.3.8 t (vetor de escalares)

Vetor tempo para plotagem (em segundos) tamanho N.

4.3.9 totalTime (escalar)

Tempo total do exame (em segundos).

4.3.10 freqMax (escalar)

Frequência de Corte [Hz] que salva o filtro passa baixa que deve ser utilizado (100 Normal, 30 Coma).

4.3.11 variablesInformation (matriz de cellString)

Contém uma breve descrição das variáveis que são salvas no arquivo .MAT.

4.3.12 dayOfConversion (string)

Salva a data em que o arquivo PLG foi convertido.

4.3.13 remainingChannels (matriz de escalares)

Contém os registros de canais do EEG convertido no formato .MAT, mas que não fazem parte da lista descrita em listChannels. Matriz de LRxN, onde:

- LR = quantidade de canais EEG diferente de listChannels [comumente LR=1], o qual representa o canal FOTO;
- N = número de amostras (valores aqui estão em uV e representam a amplitude daquela amostra.).

4.3.14 remainingChannelsName (vetor cellstr)

Nome dos canais que estavam presentes no exame .PLG, mas que não fazem parte da lista descrita em listChannels. Tamanho LR (quantidade de canais EEG restante), cada linha/coluna deste vetor tem correspondência direta com "remainingChannelsName" e "remainingChannels".

4.3.15 remainingChannelsType (vetor cellstr)

Tipo dos canais que estavam presentes no exame .PLG, mas que não fazem parte da lista descrita em listChannels. Tamanho LR (quantidade de canais EEG restante), cada linha/coluna deste vetor tem correspondência direta com "remainingChannelsType" e "remainingChannels

4.3.16 epochsValues (vetor de cell)

Tamanho nEpochs (número de épocas definidas pelo usuário [comumente 10 épocas]) cada posição desse vetor é uma célula que possui uma Matriz de LxNE, onde:

- L = quantidade de canais EEG do registro [comumente L=20];
- NAE = número de amostras da EPOCA (fa * duração da época em segundos).

4.3.17 epochsTimes (vetor de cell)

Vetor com a representação dos instantes de tempo associados a cada época, possui tamanho de nEpochs (número de épocas definidas pelo usuário). Cada célula contém o conjunto de instantes de tempo associados aos dados presentes na respectiva época.

Cada linha/coluna é uma célula que contém um vetor de tamanho NAE = número de amostras da epoca (fa * duração da época em segundos).

4.3.18 epochsValuesZerosInNoiseChannel (vetor de cell)

Igual a epochsValues, mas com zeros nos canais que foram considerados ruidosos, tanto no cálculo como pela avaliação do médico.

4.3.19 PCP (vetor de cell)

Possui tamanho de nFrequences (nFrequences comumente tem valor 7 - Delta, Teta, Alfa, Beta, Gama, Super Gama, Ruido). Cada elemento desse vetor é uma matriz de escalares de tamanho L x nEpochs que corresponde a valores da contribuição de potência de cada ritmo cerebral para aquela Época, onde:

- L = quantidade de canais EEG do registro [comumente L=20];
- nEpochs = número de épocas do exame [comumente nEpochs=10].

4.3.20 powerSignal (matriz de escalares)

Possui tamanho L x nEpochs, potência total de cada época.

4.3.21 frequencesNamePCP (vetor de cellstr)

Possui tamanho nFrequences. Contém os nomes das frequências associadas a percPCPBandFrequences (Delta, Teta, Alfa, Beta, Gama, Supergama e Ruído) valor igual à variável "frequencesName".

4.3.22 frequencesPCP (matriz de escalares)

De tamanho nFrequences x 2. Contém o intervalo das frequências [Hz] associadas a percPCPBandFrequences, possui relação direta com as linhas de "frequencesNamePCP".

4.3.23 FM (vetor de cell)

De tamanho nFrequences (nFrequences comumente tem valor 7 - Delta, Teta, Alfa, Beta, Gama, Supergama e Ruído). Cada elemento desse vetor é uma matriz de

escalares de tamanho L x nEpochs que corresponde a valores da Frequência Mediana de cada ritmo cerebral, onde:

- L = quantidade de canais EEG do registro (comumente L=20);
- nEpochs = número de épocas do exame (comumente nEpochs=10).

4.3.24 frequencesNameFM (vetor de cellstr)

Possui tamanho igual nFrequences. Contém os nomes das frequências [Hz] associadas a FMbandFrequences (Delta, Teta, Alfa, Beta, Gama, Supergama e Ruído) valor igual a "frequencesName".

4.3.25 frequencesFM (matriz de escalares)

De tamanho nFrequences x 2. Contém o intervalo das frequências associadas a FMbandFrequences. Possui relação direta com as linhas de "frequencesNameFM".

4.3.26 COR_PairsOfElectrodes (vetor de cell)

Possui o tamanho de nPairsOfElectrodes (comumente nPairsOfElectrodes=8 devido ao cálculo da coerência). Cada elemento do vetor é uma matriz de escalares de tamanho nEpochs x Y, onde:

- nEpochs = quantidade de épocas selecionadas no exame (comumente nEpochs=10);
- Y = quantidade de frequências geradas pela função do Matlab® 'mscohere'.

Essas matrizes contém os valores de coerência calculados, sendo que cada matriz é referente a um par de eletrodo. Ou seja, 1=FP1-FP2, 2=F7-F8, 3=F3-F4, 4=T3-T4, 5=C3-C4, 6=T5-T6, 7=P3-P4 e 8=O1-O2.

4.3.27 CORRELATION (vetor de cell)

De tamanho nFrequences (nFrequences comumente tem valor 7 - Delta, Teta, Alfa, Beta, Gama, Super Gama e Ruído). Cada elemento desse vetor é um vetor de cells de tamanho nPairsOfElectrodes (comumente nPairsOfElectrodes=8, que é o número de pares formados pelos eletrodos da esquerda combinados com a direita).

Cada elemento do vetor de pares é uma matriz de escalares de tamanho nEpochs x NY, onde:

- nEpochs = quantidade de épocas selecionadas no exame (comumente nEpochs=10);
- NY = representa a quantidade de frequências analisadas por cada banda/faixa de frequência de acordo com a variável Y de "COR_PairsOfElectrodes".

4.3.28 frequencesNameCORR (vetor de cellstr)

Possui tamanho igual a nFrequences. Contém os nomes das frequências associadas a bandFrequencesCORR (Delta, Teta, Alfa, Beta, Gama, Supergama e Ruído) valor igual a "frequencesName".

4.3.29 frequencesCORR (matriz de escalares)

De tamanho nFrequences x 2. Contém o intervalo das frequências [Hz] associadas a bandFrequencesCORR. Possui relação direta com as linhas de "frequencesNameCORR".

4.3.30 F_cor (vetor de escalares)

Vetor de frequências (em Hz) [tamanho Y], gerado pela função do Matlab® 'mscohere'.

4.4 TESTES DE REGRESSÃO PARA COMPARAÇÃO DA NOVA VERSÃO COM A ANTIGA

Desenvolveu-se dois tipos de testes: um usando erro relativo e outro usando o teste de Friedman.

4.4.1 Teste erro relativo

O erro relativo é calculado pela diferença do valor esperado com o valor encontrado, que é conhecido como erro absoluto, dividido pelo valor esperado. Assim temos que:

- Erro Absoluto = Valor Encontrado – Valor Esperado;
- Erro Relativo = Erro Absoluto / Valor Esperado.

O programa de teste erro relativo - que foi intitulado isEqualsExamsProcessedFriedman, o qual verifica se dois exames processados são iguais através de seus arquivos .MAT gerados em sua primeira parte - verifica se as duas variáveis são idênticas, através da própria função existente do matlab® *isequal*. Caso sejam diferentes, realiza-se, então, o teste do erro relativo para cada valor. Ele faz a comparação das seguintes variáveis, conforme a Tabela 2.

Tabela 2 – Variáveis testadas no erro relativo

1	**fa**
2	**t**
3	**xn**
4	**typeChannels**
5	**nameChannels**
6	**remainingChannels**
7	**remainingChannelsName**
8	**remainingChannelsType**
9	**epochsTimes**
10	**epochsValues**
11	**epochsValuesZerosInNoiseChannel**
12	**nameChannelsNoise**
13	**PCP**
14	**powerSignal**
15	**FM**
16	**F_cor**
17	**COR_PairsOfElectrodes**
18	**COERENCIA**

Recebe como parâmetros de entrada:

- pathName (string) = diretório onde os arquivos a serem comparados se encontram;
- fileNameReference (string) = nome do arquivo a ser lido que será referência;
- fileNameOther (string) = nome do arquivo a ser lido que será comparado;
- errorMax (float) = máximo erro aceitável para o erro relativo (padrão = 0.005 caso nenhum seja informado).

E como parâmetros de saída:

- isEqualExame (escalar):
  - 0 - se são diferentes;
  - 1 - se são idênticos.
- lineCSVCompare (string) - representa uma linha do CSV de comparação referente a todas as variáveis comparadas, onde cada coluna representa uma variável comparada de acordo com a lista mostrada na Tabela 2. Se na coluna aparecer:
  - NAO – é porque a comparação encontrou diferenças;
  - SIM - são idênticos (de acordo com erro aceitável). Ele ainda mostra a quantidade de erros aceitáveis encontrados e o valor do maior erro encontrado.

4.4.2 Teste de Friedman

O teste de Friedman é um teste estatístico, não-paramétrico, utilizado para comparar experimentos em blocos randomizados para dados amostrais vinculados, ou seja, quando o bloco em questão é avaliado, testado ou experimentado mais de uma vez. Ele não utiliza os dados numéricos diretamente, mas, sim, os postos ocupados por eles após a ordenação feita para cada grupo separadamente. Após a ordenação é testada a hipótese de igualdade da soma dos postos de cada grupo (ESTRELA, 2018).

O teste de Friedman trabalha com duas hipóteses:

- Ho: as distribuições dos k experimentos são idênticos;
- H1: as distribuições se diferem na localização.

Para o programa de teste erro Friedman, que foi intitulado isEqualsExamsProcessedFriedman, verifica-se se dois exames processados são iguais, através de seus arquivos .MAT gerados. Esta comparação é realizada através da função do Matlab® *Friedman,* no qual, se o valor retornado por esta função for menor que o parâmetro minPValue passado como argumento, aceita-se a hipótese H1 informando, assim, que os exames são diferentes. Caso contrário, aceita-se H0 considerando que os exames são iguais. Ele faz a comparação das seguintes variáveis, conforme a Tabela 3.

Tabela 3 - Variáveis testadas no erro de friedman

1	**fa**
2	**t**
3	**xn**
4	**remainingChannels**
5	**epochsTimes**
6	**epochsValues**
7	**epochsValuesZerosInNoiseChannel**
8	**PCP**
9	**powerSignal**
10	**FM**
11	**F_cor**
12	**COR_PairsOfElectrodes**
13	**COERENCIA**

Os parâmetros de entrada são:

- pathName (string) = diretório onde os arquivos a serem comparados se encontram;
- fileNameReference (string) = nome do arquivo a ser lido que será referência;
- fileNameOther(string) = nome do arquivo a ser lido que será comparado;
- minPValue (escalar) = mínimo P valor aceito (padrão = 0.8 caso não seja informado nenhum);
- validateVarsBase (boolean) = validar variáveis de base (t, xn ve remanaingChannels). Como essas variáveis são grandes ele deixa o processamento lento, por default ele é zero;

Parâmetros de saída:

- isEqualExame (escalar)
    - 0 - se são diferentes;
    - 1 - se são idênticos.
- lineCSVCompare (string) - representa uma linha do CSV de comparação referente a todas as variáveis comparadas, onde cada coluna representa uma variável comparada de acordo com a lista mostrada na Tabela 3. Onde, se a coluna aparecer:
    - NAO - são diferentes
    - SIM - são idênticos (de acordo com erro aceitável). Ele ainda mostra a quantidade de erros aceitáveis encontrados e o valor do maior erro encontrado.

4.4.3 Função testCompareExameByCsv

Criou-se, também, uma função (testCompareExameByCsv) que recebe como entrada um arquivo .CSV, onde a primeira coluna representa os nomes de arquivos de referência e a outra coluna representa os nomes de arquivos de comparação, no qual o programa realiza a comparação linha por linha contrapondo os exames, retornando no final dois arquivos .CSV no qual um representa a comparação pelo erro Relativo e outro a comparação pelo erro de Friedman.

Estes arquivos retornados são idêntico ao arquivo de entrada com a adição das colunas referentes às variáveis comparadas: Tabela 2 para o erro relativo e Tabela 3 para Friedman, nas quais se aparecer a palavra **SIM** na coluna significa que a variável é relativamente igual entre os arquivos comparado e, se aparecer a palavra **NÃO**, representa que a variável comparada se difere entre os arquivos. No **ANEXO E** tem-se um exemplo do arquivo de saída do erro relativo e no **ANEXO F** tem-se um exemplo do arquivo de saída para o erro de Friedman.

4.5 RESULTADOS

4.5.1 Processamento dos PLG's

Para realizar o processamento dos EEG's através da nova versão (mais automatizada) é necessário montar o arquivo .CSV com todas as informações relacionadas ao EEG e aos parâmetros de entrada para o processamento como, por exemplo: nome do arquivo .PLG com a gravação do exame, o tamanho e o início das épocas, os canais ruidosos, o tipo do exame (COMA/NORMAL) para se aplicar determinado filtro passa baixa, nome do arquivo de saída .MAT e se deve ou não gerar arquivos excel com as principais variáveis de retorno. No **ANEXO B** pode-se visualizar um exemplo deste arquivo .CSV e os detalhes das colunas deste arquivo são descritos no item **4.2.2.15 processRecords**.

Para executar o programa é necessário que o usuário tenha o MATLAB® instalado com versão igual ou superior a 2007, uma vez que todo o código de processamento foi desenvolvido na versão 2007 e testado também na versão 2016. Além do mais, torna-se necessário configurar a área de trabalho do MATLAB® para o diretório onde se encontram todas as funções mencionadas neste trabalho para que o mesmo possa ser executado.

Uma vez realizada todas estas configurações, o usuário deve digitar o nome da rotina principal para processamento (porcessRecords) na janela de comandos do MATLAB® e, desta forma, será mostrada uma outra janela para a seleção do arquivo CSV a fim de processamento, conforme Figura 1.

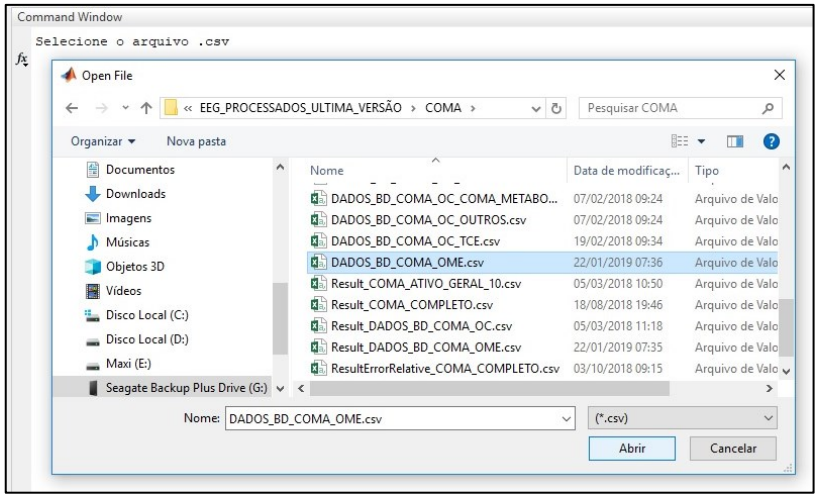


Figura 1 - Janela para seleção do CSV a ser processado

É importante ressaltar que o diretório onde o arquivo CSV (planilha) foi selecionado deve conter todos os arquivos PLG's referentes aos dados preenchidos na planilha para serem processados. Caso não esteja, o programa irá mostrar uma mensagem na janela de comandos, informando que não foi possível encontrar determinado PLG.

Após a seleção do CSV o programa inicia o processamento mostrando na janela de comandos os passos que ele está realizando para que o usuário tenha a percepção do que o programa está rodando e em qual parte ele está. Além do mais, são exibidas mensagens de tempo gasto para cada fase do processamento para que se possa identificar os pontos mais demorados, informações estas que podem ser visualizada na Figura 2. Este programa demora em média 20 segundos para processar cada PLG contendo 10 (dez) épocas de 2 segundos, com um exame de PLG de 20 minutos de gravação.

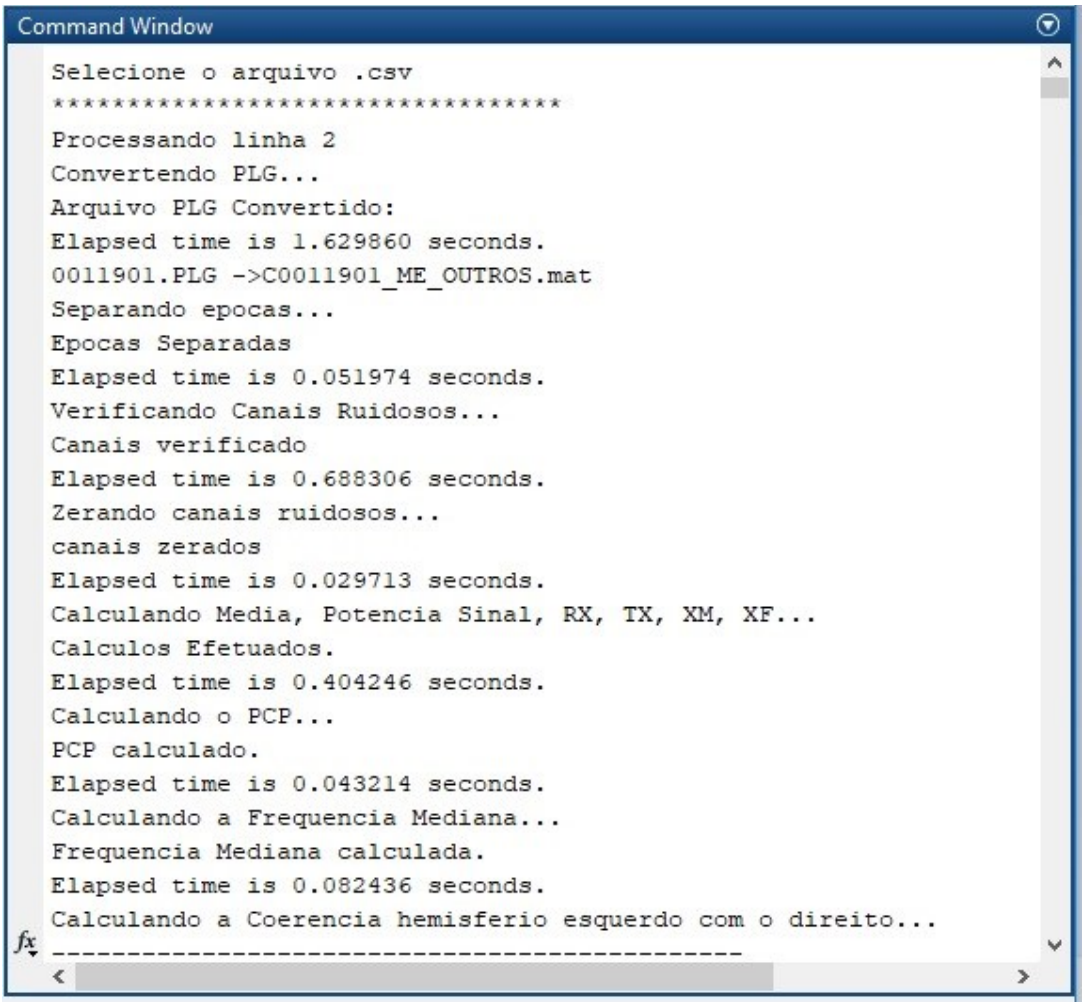


Figura 2 - Janela de comandos durante a execução do processRecords

Após a execução do processRecords é gerado um outro arquivo CSV com o prefixo “Result_” mais o nome do arquivo CSV de entrada, no qual contém as mesmas informações do arquivo CSV de entrada mais a adição de algumas colunas com status do processamento de cada linha, colunas estas que são: o status da conversão (se foi sucesso ou não), os canais ruidosos identificado pelo software, a validade do exame (se contém no máximo 3 canais ruidosos), a frequência máxima adotada (valor do filtro passa baixa) e se o processamento de PCP, FM e Coerência foi realizado com sucesso. No **ANEXO C** pode-se visualizar estas colunas adicionais referentes ao processamento do **ANEXO B** e os detalhes das colunas do **ANEXO C** são descritos no item **4.2.2.15 processRecords**.

4.5.2 Execução dos testes de regressão

A execução dos testes comparativos é semelhante à execução do processRecords, devendo-se criar um arquivo CSV com duas colunas. A primeira coluna deve conter o nome do arquivo .MAT gerado através da versão antiga de processamento, que é tomada como base. A segunda coluna, por sua vez, deve conter o nome do arquivo MAT gerado através da versão nova (automatizada) referente ao processamento do mesmo PLG do MAT informado na primeira coluna. No **ANEXO D** tem-se um exemplo deste arquivo CSV. Na janela de comandos do MATLAB® o usuário deve chamar a função testCompareExameByCsv, logo em sequência será mostrada a janela da Figura 3 para que o usuário possa selecionar o CSV.

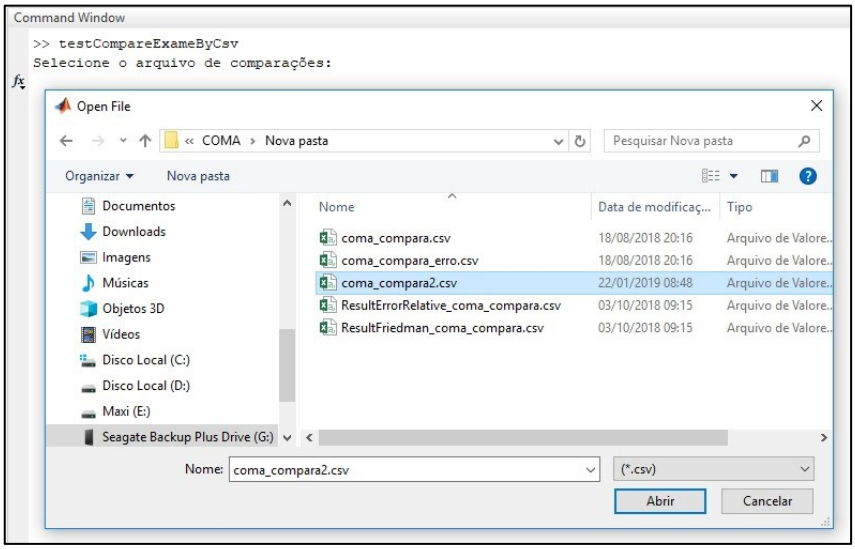


Figura 3 - Seleção do CSV para execução do teste de comparação de versão do software

Após selecionar o arquivo CSV o software começa a realizar as comparações de variáveis com o erro relativo e com o teste estatístico de Friedman para cada linha do arquivo. As variáveis que estão sendo comparadas são mostradas na janela de comandos (Figura 4) para que o usuário tenha a percepção do que o programa está executando e em qual ponto ele está.

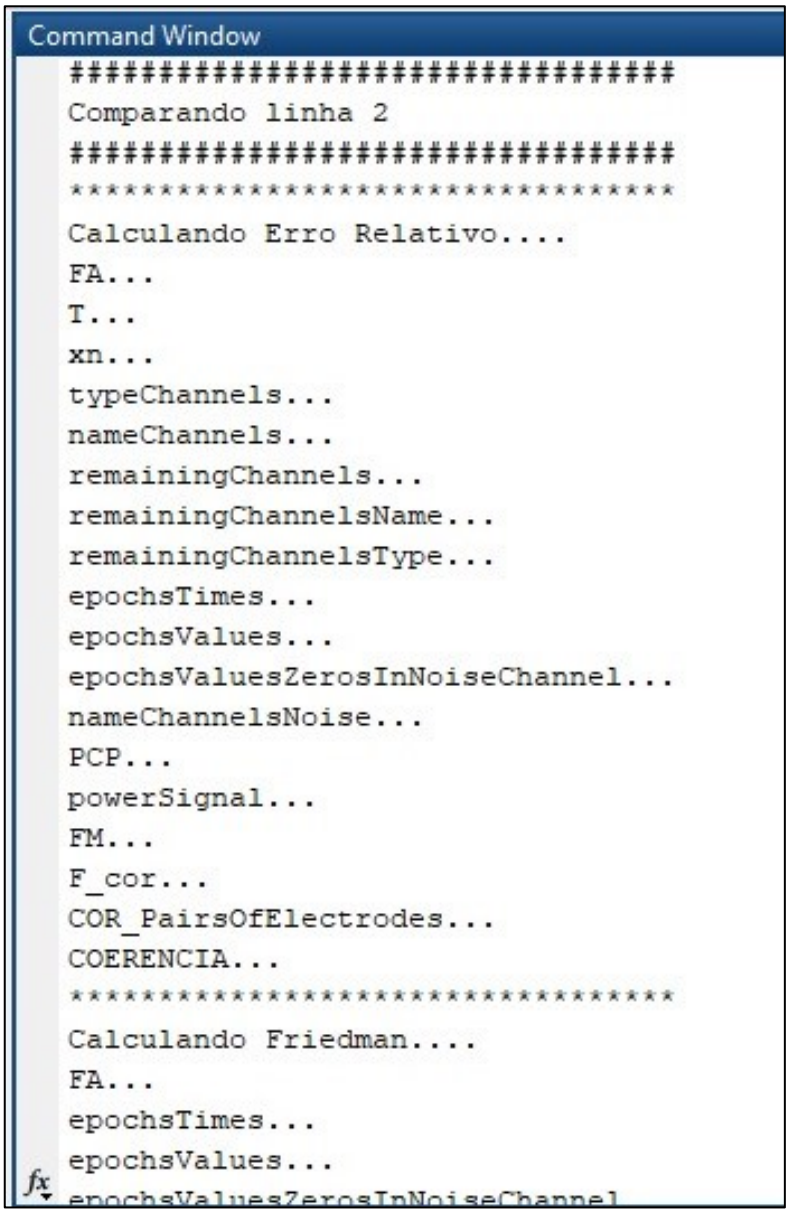


Figura 4 - Janela de comandos durante a execução do testCompareExameByCsv

Posteriormente, na execução testCompareExameByCsv são gerados dois arquivos CSV com o prefixo “ResultErrorRelative_” (**ANEXO D**) e “ResultFriedman_” (**ANEXO E**) seguido do nome do arquivo CSV de entrada, no qual contém as mesmas informações do arquivo CSV de entrada mais o resultado da comparação para cada variável testada, no qual se houver a palavra **SIM** significa que as variáveis são iguais,

havendo a palavra **NAO** as variáveis são diferentes. Na última coluna é apresentado o tempo gasto para a realização da comparação de variáveis para um determinado exame (.MAT).

4.6 CONSIDERAÇÕES FINAIS

A grande vantagem dessa reestruturação de código foi pelo fato de não ser mais necessário, como na primeira versão do código, que o usuário insira manualmente, toda vez que fosse processar um exame, as informações inerentes, como pode ser observado pelos diagramas do **ANEXO A** (Diagrama Execução: P1_CONVERSOR, P2_EPOCAS, P3_VALIDAEEG, P4_PCP, P5_FM e P6_NOVO_COR). A dificuldade era ainda maior se houvessem muitos exames a serem processados fazendo um trabalho manual que por vezes poderia ter um erro de digitação, pois para cada época se inseria um valor e para cada exame se selecionava um arquivo. No novo formato o usuário preenche uma única vez a planilha CSV que é utilizada como entrada (podendo ser até um arquivo de documentação de processamento) e o programa faz todo o cálculo, devolvendo como saída uma planilha com o resultado do processamento e os arquivos .mat gerados através dos PLG's, fato que pode ser observado facilmente pelo diagrama de processRecords (**ANEXO A** - Diagrama Execução processRecords), o qual mostra apenas uma interação com o usuário, que é a solicitação da planilha de entrada.

O desenvolvimento dos programas de testes foi de grande utilidade, pois ajudou, na refatoração do código, a encontrar muitos erros de desenvolvimento, visto que se comparava a versão antiga com a nova. Assim, pode-se assegurar que a nova versão, com as novas melhorias, não trouxe erros.

CAPÍTULO 5 - VISUALIZAÇÃO DOS QUANTIFICADORES DO EEG

No caso particular do EEG, seguindo o padrão internacional 10-20 de montagem dos eletrodos, existem pelo menos 20 sinais e cinco ritmos neurológicos distintos básicos (delta, teta, alfa, beta e gama). Alguns estudiosos ainda subdividem estes ritmos aumentando mais ainda o número de informações. Vide na Figura 5 o exemplo de um exame. Neste caso, um conjunto total de 20 x 5 = 100 valores, cada qual referente a um eletrodo e a um ritmo cerebral, tornando-se, assim, um complexo desafio de se extrair informações. Todavia, a partir do momento em que tais valores são visualizados através de um gráfico é possível melhorar a apresentação da informação. De fato, o diagnóstico neurológico exige este esforço (NIEDERMEYER; SILVA, 2004) e, desta forma, os neurologistas, ao analisarem visualmente o eletroencefalograma pelo seu traçado no tempo, conseguem identificar patologias como, por exemplo, a epilepsia.

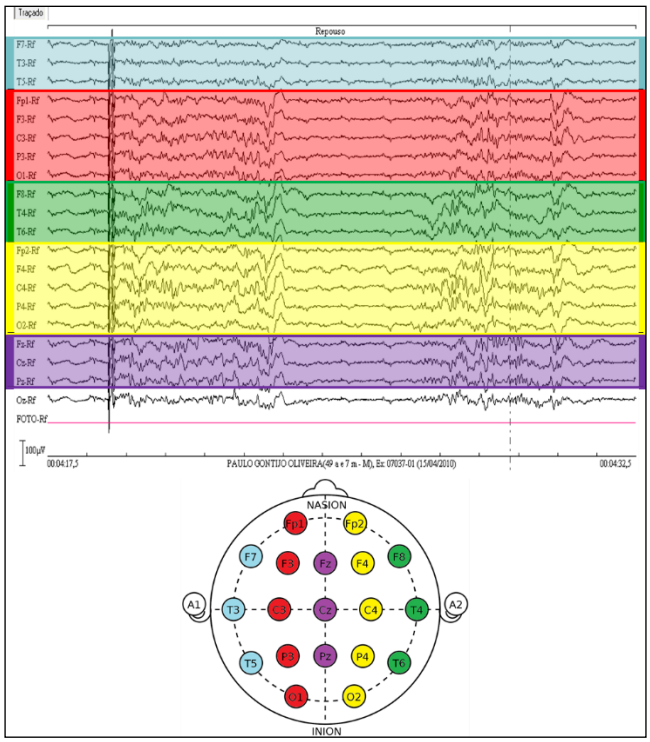


Figura 5 - Visualização de um exame de EEG representado no tempo, destacando-se o posicionamento dos eletrodos no padrão 10-20

Como não existe uma diretriz sucinta para indicar todas as circunstâncias sob as quais uma determinada regra de visualização pode ser aplicada (WARE, 2012),

projetar visualizações da informação é uma demanda complexa. Isso faz com que os projetistas gráficos levem em conta as interações entre símbolos pequenos e grandes áreas de cor e textura, bem como efeitos de sombreamento, efeitos de forma, o agrupamento de símbolos e assim por diante, de forma que pequenos detalhes possam ditar mudanças no que deve ser destacado e o que deve ser enfatizado (WARE, 2012).

O processo de visualização de dados envolve quatro estágios básicos, conforme ilustrado na Figura 6. Segundo (WARE, 2012), estes quatro estágios consistem de:

- Coleta e armazenamento de dados (**aquisição de dados**);
- Pré-processamento para transformar os dados em algo que é mais fácil de manipular (**transformação dos dados**). Normalmente, existe alguma forma de redução destes para revelar aspectos selecionados. Assim, a **exploração de dados** é o processo de alteração do subconjunto que está sendo visualizado;
- Mapeamento dos dados selecionados para uma representação visual, realizado através de algoritmos de computador que produzem uma imagem na tela (**Gerador gráfico do mapeamento visual**). As entradas e ações do usuário podem transformar os mapeamentos, destacar subconjuntos ou transformar a exibição (**manipulação da visualização**). Isto é comumente feito no próprio computador do usuário;
- Percepção do observador, que consiste no sistema perceptivo e cognitivo humano (**Analise da informação, Processamento visual e cognitivo**).

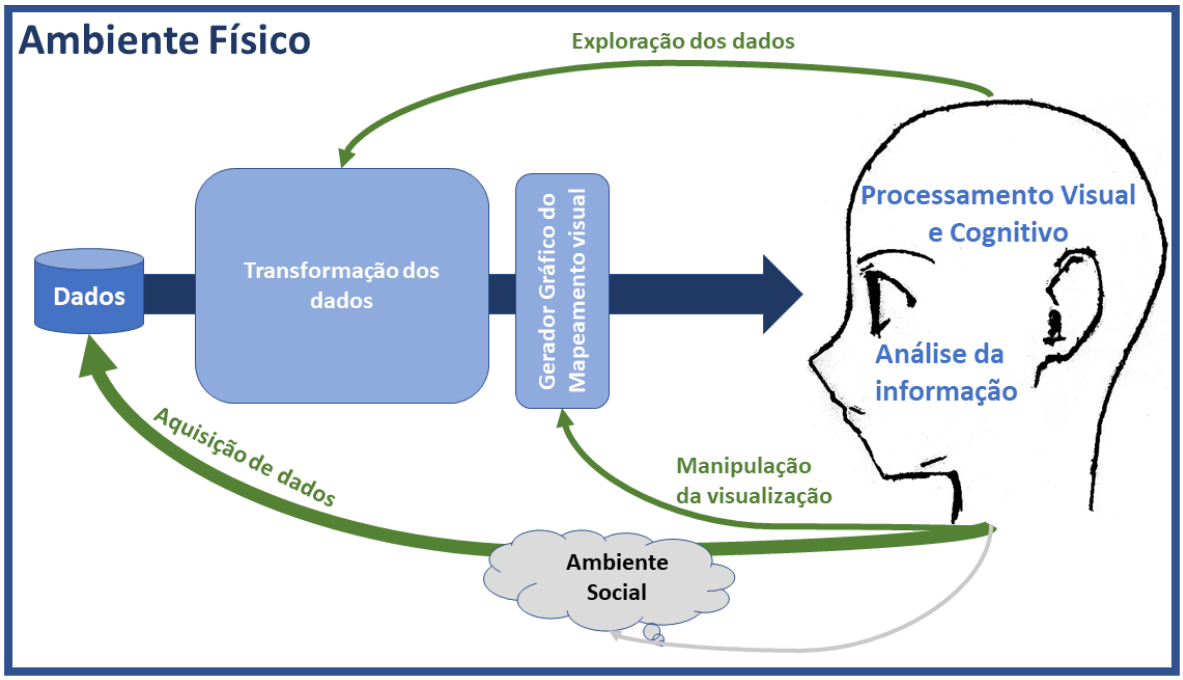


Figura 6 - Estágios da visualização da informação, adaptado de (WARE, 2012).

5.1 SLORETA

A geração topográfica no software (*sLoreta*) é um pouco trabalhosa, mas também é bem configurável. Nele deve-se criar um arquivo com o conteúdo em forma de vetor (cada linha um elemento), com a ordem dos eletrodos seguindo a nomenclatura padrão na literatura para o nome dos eletrodos. De posse do arquivo com os nomes e a ordem dos eletrodos, no próprio programa gera-se outro arquivo com as coordenadas de cada eletrodo no escalpo. Com o arquivo de coordenadas faz-se a geração dos pontos de cada eletrodo na imagem 3D e, em seguida, cria-se mais um arquivo com os valores a serem exibidos para cada eletrodo, conforme a ordem definida no primeiro arquivo. Este último arquivo consisti apenas na definição de um vetor coluna ou uma matriz e cada coluna dessa matriz representará um tempo para ser exibido. Em seguida, irá aparecer uma linha do tempo com os valores do arquivo de dados. Ao clicar em qualquer um dos tempos é gerada uma visualização topográfica daquele instante. Na Figura 7 pode-se visualizar um exemplo desta plotagem.

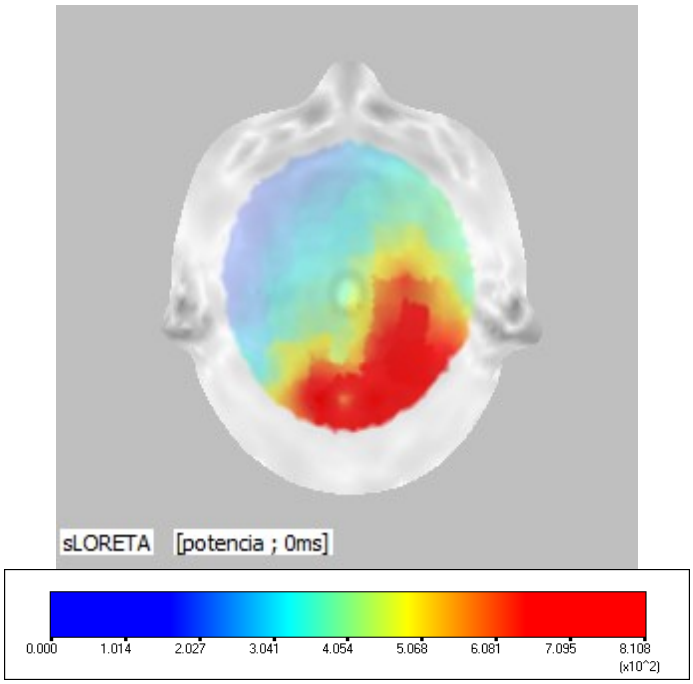


Figura 7 – Plotagem topográfica no *SLoreta*

No entanto, pode-se perceber que a imagem gerada pelo *sLoreta* (Figura 7) é uma vista superior deformada, projetada a partir de uma visualização 3D de uma cabeça. Mas, como o objetivo do *sLoreta* é fazer visualizações 3D, essa marcação é mostrada apenas nas visualizações 3D, conforme pode ser visto na Figura 8, através de uma imagem gerada com seis vistas. Note que nesta figura aparece sempre um ponto para marcar a posição de cada eletrodo.

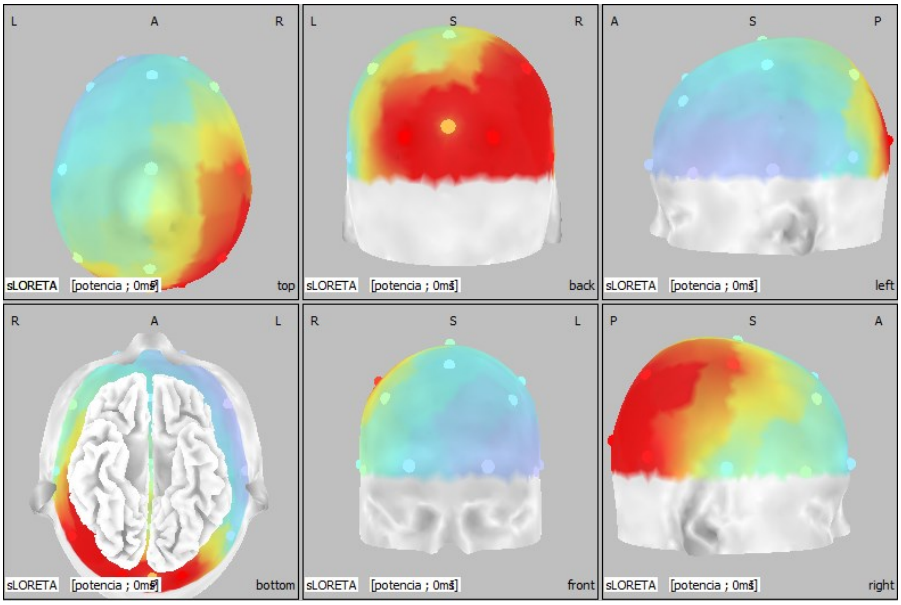


Figura 8 - Vistas geradas no *sLoreta*

O *sLoreta* ainda permite girar essa ilustração e gerar uma visualização da perspectiva 3D da imagem conforme escolha do usuário. Na Figura 9 há um exemplo dessa perspectiva.

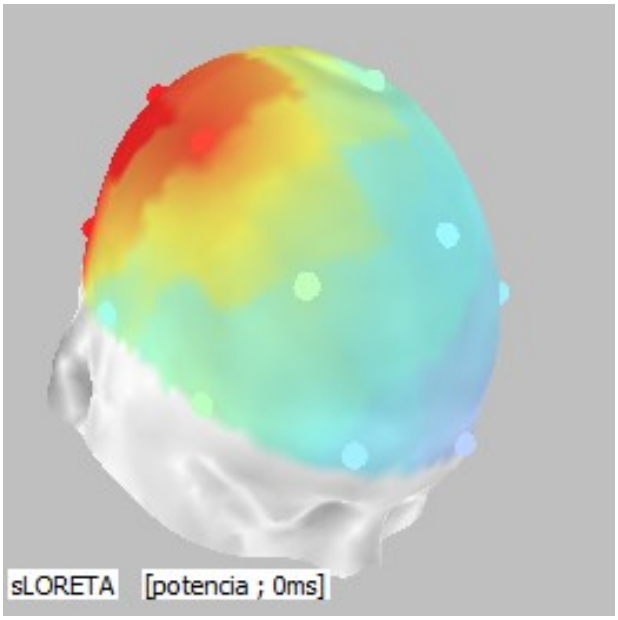


Figura 9 - Perspectiva *sLoreta*

O *sLoreta* possui, ainda, outro recurso bastante interessante: exibe a localização da função cerebral através de um dado momento do EEG, conforme ilustrado na Figura 10.

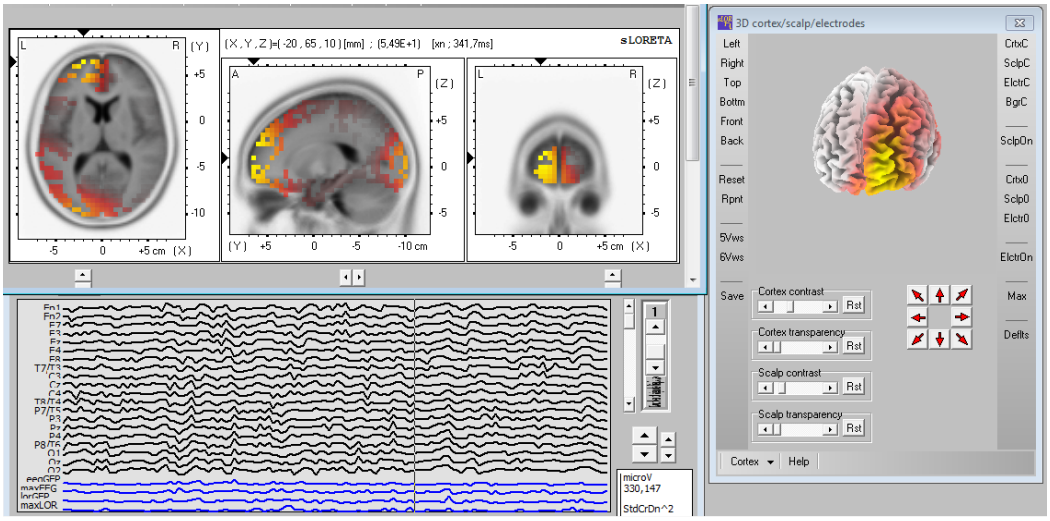


Figura 10 - Visualização da localização da função cerebral de um dado momento do EEG (*sLoreta*).

5.2 MATLAB

O programa para a visualização dos resultados do EEG no Matlab® foi desenvolvido pelo Gaspar, um dos integrantes da equipe de pesquisa. Para criar-se a imagem topográfica basta chamar a rotina desenvolvida, que recebe como parâmetro os valores de cada eletrodo, e o valor mínimo e máximo da escala. Vale ressaltar que o vetor de entrada dos eletrodos deve seguir uma ordem específica, ou seja, a aplicação espera que seja inserido um vetor com 20 valores, sendo que o primeiro valor represente o eletrodo FP1, o segundo FP2, o terceiro F7, e assim sucessivamente até O2. A sequência é a seguinte: Fp1, Fp2, F7, F3, Fz, F4, F8, T3, C3, Cz, C4, T4, T5, P3, Pz, P4, T6, O1, Oz, O2. Na Figura 11 pode-se visualizar um exemplo desta plotagem

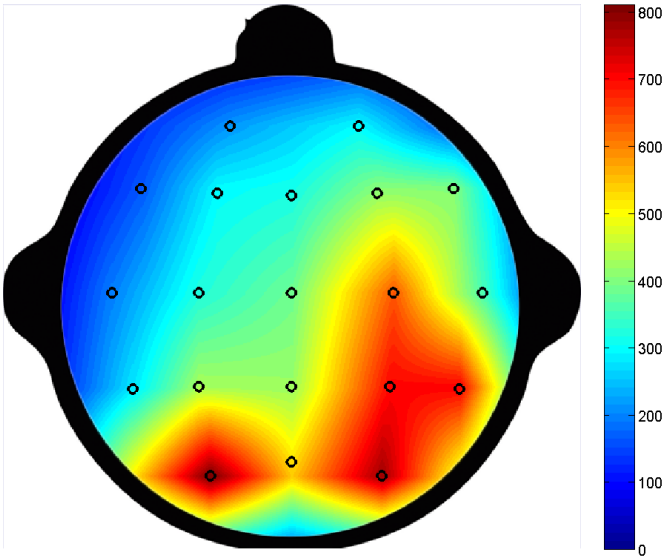


Figura 11 - Plotagem topográfica no Matlab®

5.3 VIDEO

Para o desenvolvimento do software de criação de vídeo gerou-se uma *branch* (uma cópia do código que também é versionada no mesmo repositório) com base na máster, pois o mesmo envolve uma alteração no software de processamento.

Esta alteração consiste em adicionar um parâmetro no arquivo de entrada (.CSV), onde, se a primeira coluna de épocas estiver com a palavra “VIDEO”, o sistema irá se comportar de maneira diferente no momento em que for separar as

épocas. Assim, ele irá dividir o exame em 60 épocas distintas de 1 segundo, pegando os primeiros 60 segundos.

Para a geração dos frames na montagem do vídeo, adotou-se a estratégia de pegar os resultados da variável PCP (porcentagem de contribuição de potência de cada ritmo) e a potência total para todas as épocas mencionadas. Depois disso, pegou-se o valor das primeiras 10 épocas, realizando a média desses valores, e, logo em sequência, gerou-se a visualização topográfica do primeiro *frame*. Em seguida, pegou-se o valor das outras primeiras 10 épocas, excluindo a primeira época, gerando outro *frame*. Assim procedeu-se sucessivamente até chegar às últimas 10 épocas (51 a 60), totalizando um total de 50 *frames*.

O vídeo foi configurado para rodar a uma taxa de 20 *frames* por segundo. Cada frame gerado no vídeo foi repetido 5 vezes, para deixar o vídeo com uma transição mais suave entre os *frames*. Desta forma, o vídeo gerado ficou com duração de 12,5 segundos no formato AVI (*Audio Video Interleave*). Na Figura 12, Figura 13, Figura 14 e Figura 15 pode-se visualizar cada tipo de vídeo gerado.

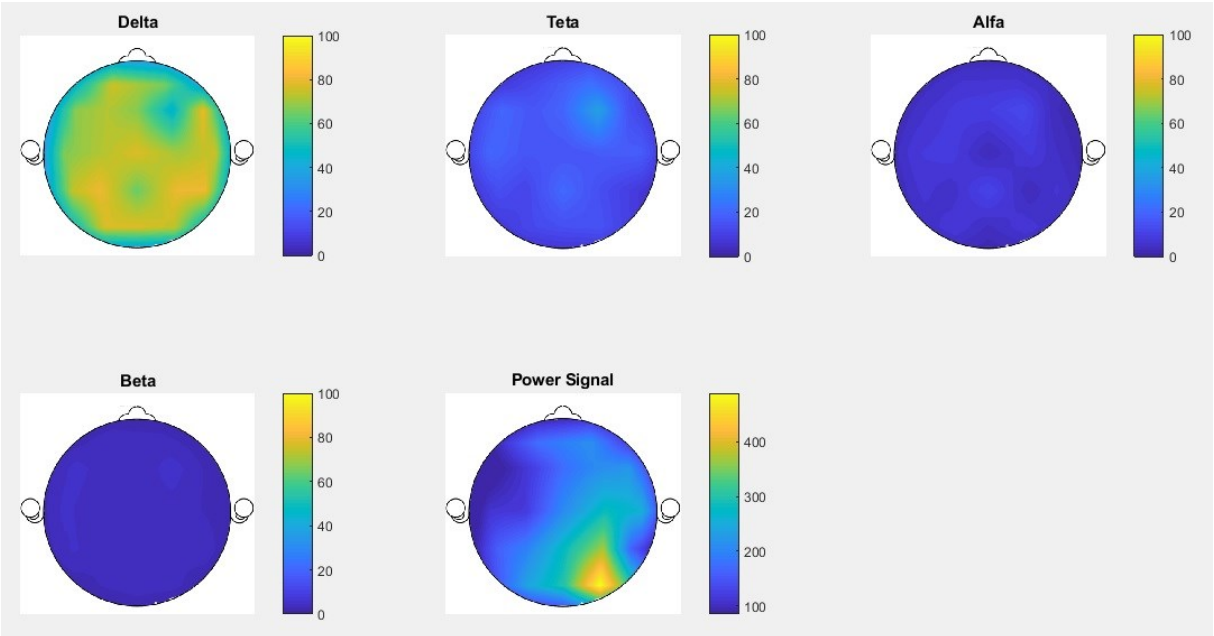


Figura 12 - Representação Gráfica 1– escala comum de 0 a 100% para as bandas de frequência e escala particularizada para a potência total, instante do vídeo 0 segundos.

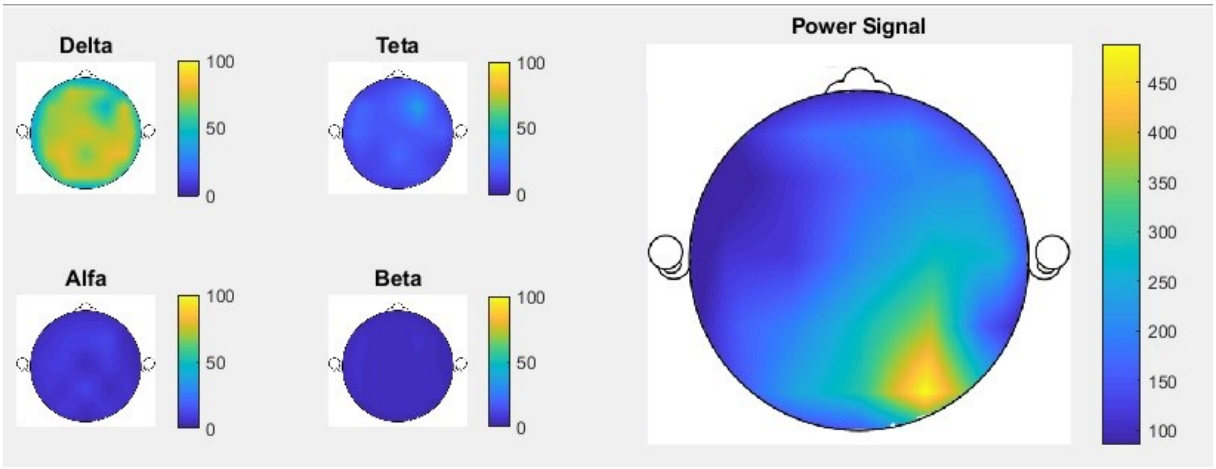


Figura 13 - Representação Gráfica 2– escala comum de 0 a 100% para as bandas de frequência e escala particularizada para a potência total, destacando-se a potência total, instante do vídeo 0 segundos.

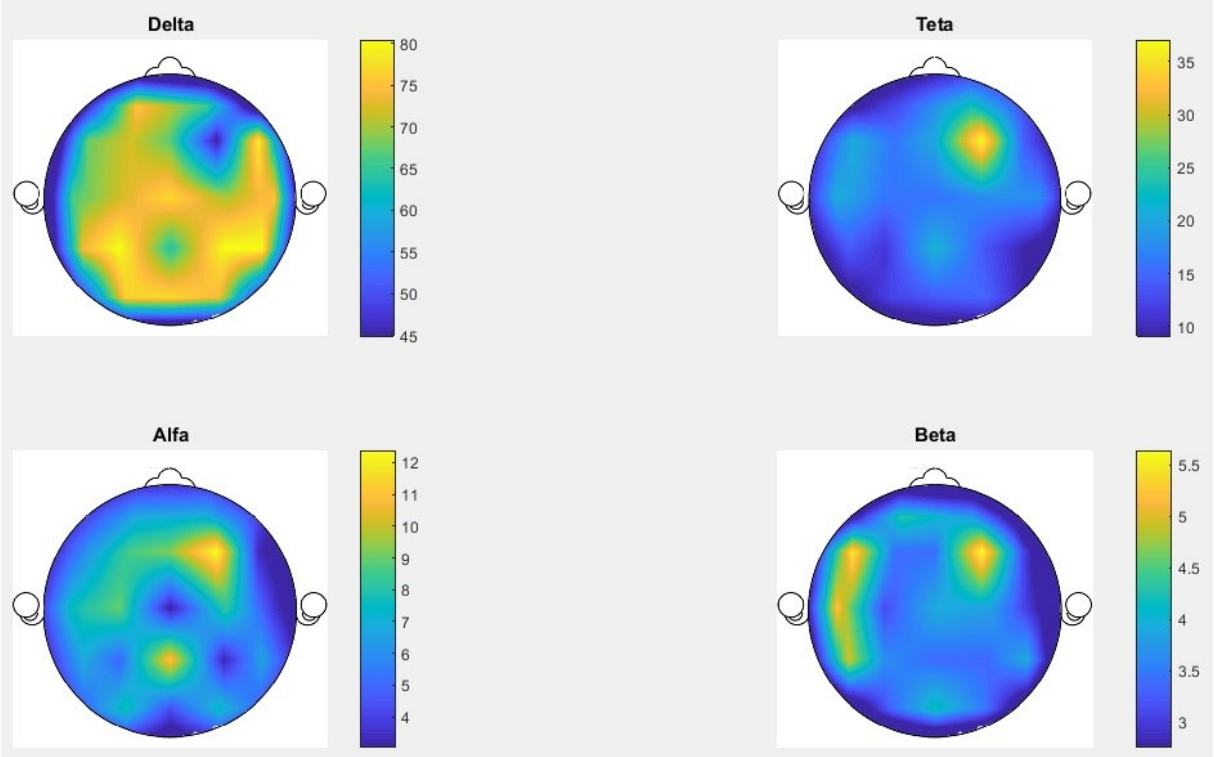


Figura 14 - Representação Gráfica 3 – escala particularizada para cada ritmo, sem a representação da potência total, instante do vídeo 0 segundos.

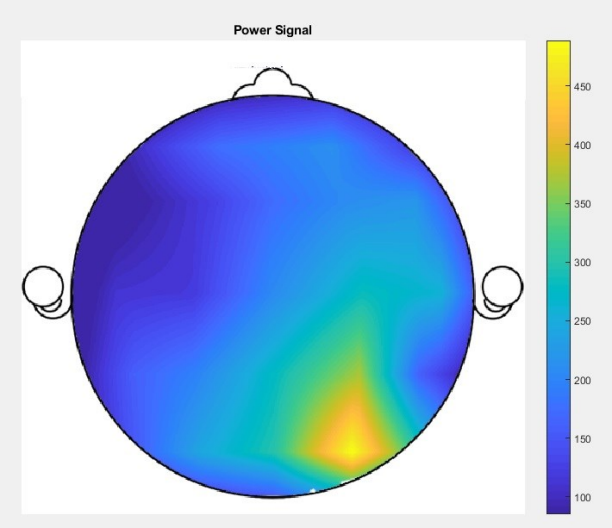


Figura 15 Representação Gráfica 4– Destaque da potência total, escala particularizada, instante do vídeo 0 segundos.

5.4 FRAMES

Com o intuito de criar uma visualização dos resultados para comparar os vídeos e mostrar a visualização do mesmo em uma folha de papel, fez-se um programa que gera uma imagem com os N primeiros *frames* de uma banda de frequência. Para ilustrar sua evolução temporal no vídeo, na Figura 16 e Figura 17 pode-se visualizar os N primeiros *frames* para 5 (cinco) etiologias diferentes, sendo elas: Coma Ativo por Traumatismo Cranioencefálico (TCE), Coma Ativo por Acidente Vascular Encefálico (AVE), Coma Ativo por Coma Metabólico (CM), Morte Encefálica com etiologia de Traumatismo Cranioencefálico (ME_TCE) e paciente normal.

Na Figura 16 pode-se visualizar a comparação, mencionada acima, seguindo a escala de cores de 0 a 100 para a banda de frequência alfa com os 6 primeiros frames.

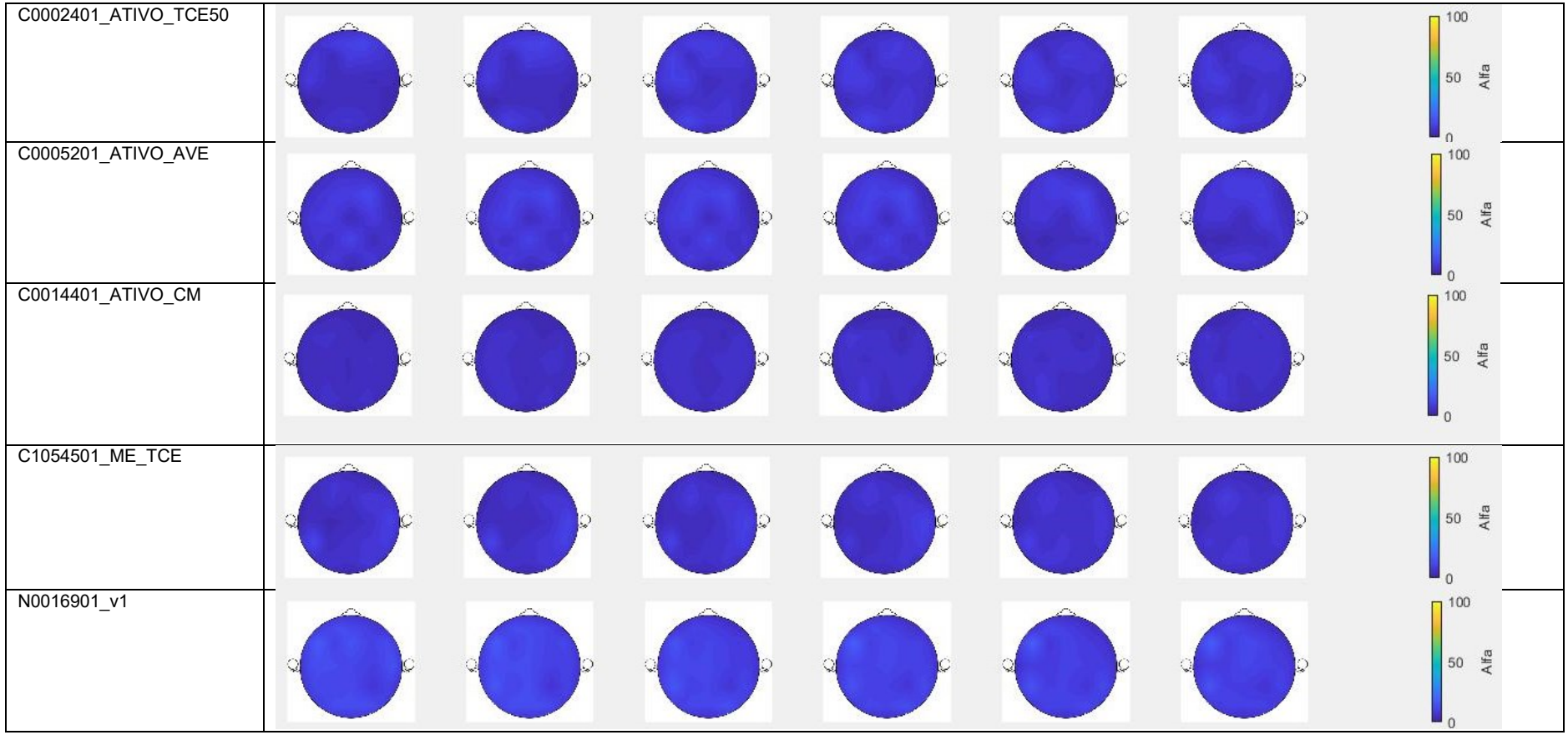


Figura 16 – Visualização criada a partir dos frames do Vídeo, com escala comum 0-100%. Cada linha representa uma etiologia diferente, considera-se apenas a banda alfa

A outra visualização gerada para comparar as etiologias (Figura 17), utilizou uma escala particularizada para cada frame destacando-se, desta maneira, a evolução temporal para a banda de frequência em questão, que neste caso é a banda Alfa.

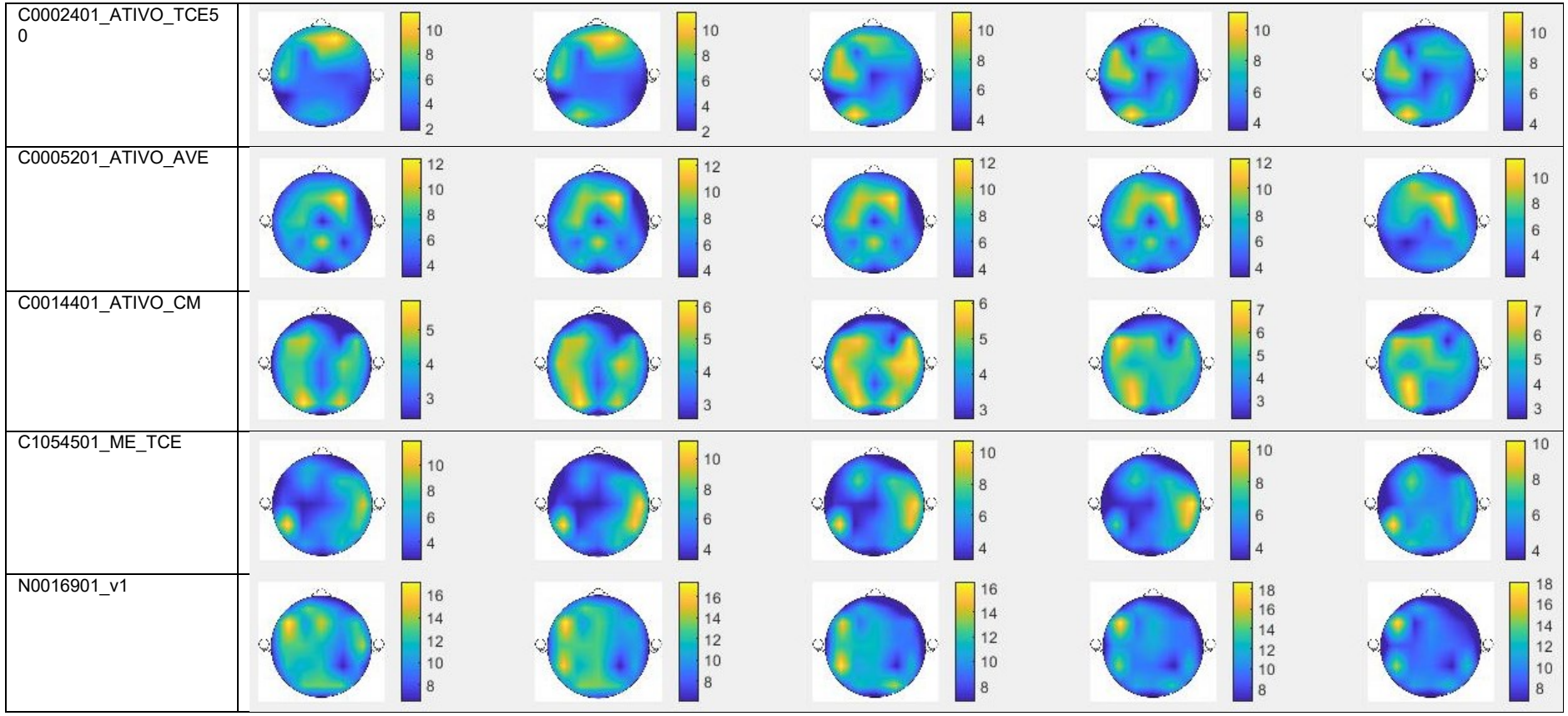


Figura 17 – Visualização criada a partir dos frames do Vídeo, considerando apenas a banda alfa, a escala de cada cabecinha foi obtida considerando-se a máxima e a mínima potência calculada

Criou-se, também, outro programa para montar uma visualização de comparação das quatro bandas de frequência (Delta, Teta, Alfa e Beta) de um exame utilizando os N primeiros frames, com uma diferença, onde a escala particularizada foi definida como

sendo o valor mínimo e máximo do PCP da banda em questão entre os N primeiros frames. Dessa forma, a escala ficou ilustrada no canto extremo à direita de cada linha que representa a evolução temporal dos frames para cada banda de frequência, conforme pode ser visualizado na Figura 18.

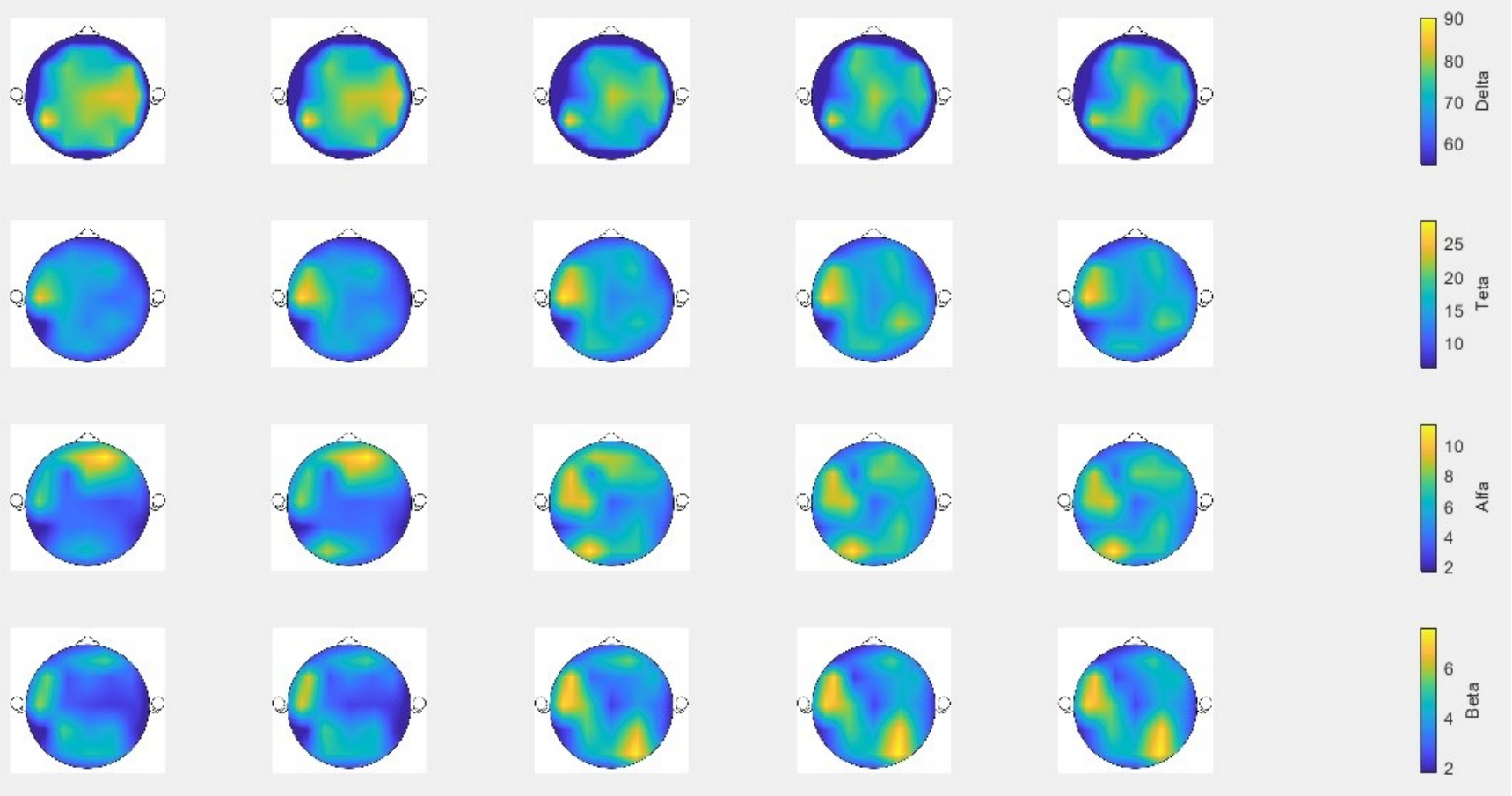


Figura 18 – Visualização criada a partir dos frames do Vídeo, de um mesmo indivíduo. Cada linha representa a evolução temporal da potência de uma determinada banda, conforme escala de cores particularizadas, mostrado na extremidade a direita de cada linha.

5.5 CONSIDERAÇÕES FINAIS

Comparando-se a Figura 7 com a Figura 11 pode-se perceber que não há diferenças significativas na visualização bidimensional da imagem topográfica para os dois software (*sLoreta* e Matlab®). No entanto, outros aspectos devem ser considerados.

O primeiro aspecto é quanto à licença de uso, sendo paga no caso do Matlab®, enquanto o *sLoreta* é de uso gratuito para pesquisa.

Caso se tenha a licença do Matlab® e o algoritmo para o processamento do sinal de EEG também tenha sido desenvolvido nele, é interessante utilizar a visualização através da aplicação em Matlab®, visto que a integração é mais simples e rápida. No entanto, caso o algoritmo de processamento tenha sido desenvolvido em outra ferramenta e se tenha os valores numéricos a serem exibidos, usar o *sLoreta* é mais vantajoso, pois sua instalação é simples e o espaço em memória ocupado é de aproximadamente 200 Megabytes, diferentemente das versões do Matlab® que exigem até alguns Gigabytes.

Outro fator refere-se à visualização. A ferramenta desenvolvida em Matlab® permite apenas a apresentação em 2D, enquanto o *sLoreta* permite visualização em 2D e 3D. Diante do exposto, a escolha do software para gerar a visualização topográfica do escalpo dependerá das necessidades da aplicação.

A geração de vídeos a partir do EEG foi desenvolvida no Matlab® com o intuito de se ter novas formas de visualização do EEG, visto que visualizações diferentes podem trazer novos *insights,* além do mais, elas permitem fazer um acompanhamento temporal da evolução do EEG. Para efeito de comparações gerou-se as várias formas de visualização.

Na Figura 12 montou-se o vídeo para quatro bandas de frequência mais a potência total, totalizando 5 visualizações distintas. No entanto, esta visualização dá a impressão de que há uma figura faltando no lado direito inferior. Como a potência total segue uma escala diferente das outras bandas de frequência, optou-se em gerar uma nova visualização tampando o espaço que está vazio e aumentando/destacando a potência total do exame, como pode ser visto na Figura 13.

Como a potência total na Figura 13 chamou muita atenção, por vezes diminuindo o foco do observador para as bandas de frequências, gerou-se, ainda, outra visualização do vídeo mostrando apenas as bandas de frequência, conforme a Figura 14, de forma compacta, ocupando a janela de visualização como um todo, sendo as 4 figuras do mesmo tamanho para não desviar o foco apenas para uma determinada banda de frequência.

Para demonstrar a evolução temporal do vídeo em um papel, comparando etiologias diferentes, gerou-se as imagens da Figura 16 e Figura 17. A Figura 16 utilizou uma escala fixa de 0 a 100%, mas, como os valores encontrados para cada eletrodo são relativamente baixos, a escala de cores na imagem quase não oscila, dificultando assim uma percepção melhor de qual área está sendo mais ativa. Já para a Figura 17, para melhorar a percepção de qual área está sendo mais ativada, montou-se a visualização com uma escala particularizada para cada instante mostrado, e, assim, a oscilação de cores entre as regiões ficou bem mais distinta, melhorando a percepção do observador. Antes, na Figura 16, quase não se percebia as variações; já na Figura 17 estas variações ficaram bem mais nítidas.

CAPÍTULO 6 - SISTEMA PARA CADASTRO DOS EEG COLETADOS

Este capitulo é dedicado a apresentar o sistema de cadastro de exames de EEG coletados pelo grupo de pesquisa. O código criado para a montagem desta aplicação está disponível em:

https://github.com/ludovicoMaxi/ufu-ppgeb-eeg

6.1 BANCO DE DADOS

Para a montagem dessa aplicação para o ambiente de teste/desenvolvimento foi utilizado o banco de dados em memória H2. Nele, a cada inicialização do servidor, através de um parâmetro de configuração do Spring e Hibernante, as tabelas do banco de dados são criadas e, logo em sequência, existe uma configuração do Spring para ele coletar um script para inserção de registros nas tabelas.

Já para o ambiente de produção, que é montado através do Heroku, utiliza-se o banco de dados PostgreSQL. Inicialmente pensou-se em utilizar o banco de dados mySQL, por ser ele gratuito, no entanto, pelo fato dele ser pago na plataforma Heroku optou-se pelo PostgreSQL.

O relacionamento do banco de dados tem-se como início a entidade *PATIENT* (Paciente) que representa o paciente/voluntário, e que pode ter vários *EXAM* (exames) e vários *EXAM_REQUEST* (requisições de exames), que é ilustrado na Figura 19 através do relacionamento um para muitos onde tem-se a ligação entre a entidade *PATIENT* com apenas um traço conectando a ela (representando um) e nas demais entidades relacionadas a ela tem-se três traços (representando muitos). A entidade *PATIENT* possui os seguintes atributos:

- ID – identificador, sendo a chave primaria;
- NAME – representa o nome do paciente/voluntário;
- DOCUMENT_NUMBER – representa um identificador único do paciente, como o CPF;

- SEX – o sexo, podendo ser masculino ou feminino, representado pelas letras M para masculino e F para feminino;
- BIRTHDATE – data de nascimento;
- NACIONALITY – nacionalidade, país de origem;
- CIVIL_SATUS – estado civil: solteiro, casado, etc;
- JOB – profissão;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

A entidade EXAM_REQUEST se relaciona com a entidade *PATIENT*, podendo ter um e apenas um *PATIENT* vinculado a ela (relacionamento muitos para um), podendo ter nenhum ou vários *EXAM* vinculados também (relacionamento um para muitos) (Figura 19). Os atributos desta entidade são mostrados logo abaixo:

- ID – Identificador, chave primária;
- MEDICAL_RECORD –
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

Já a entidade *EXAM* é a que possui mais relacionamentos, podendo ter: um e apenas um *PATIENT* vinculado (relacionamento muitos para um), nenhum ou apenas um *EXAM_REQUEST* (relacionamento muitos para um), nenhum ou vários *EPOCH* (épocas) (relacionamento um para muitos), nenhum ou vários EXAM_EQUIPMENT (equipamentos usados no exame) (relacionamento um para muitos), e nenhum ou vários EXAM_MEDICAMENT (medicamentos usados pelo paciente próximo a data de realização do exame) (relacionamento um para muitos) (Figura 19). A seguir são exibidos os atributos desta entidade:

- ID – Identificador, chave primária;
- MEDICAL_RECORD –
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;

- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

EPOCH é a entidade que mapeia as épocas selecionadas pelo neurologista ao se avaliar o *EXAM* para se realizar o processamento do mesmo. Ela se relaciona com a entidade *EXAM* tendo um e apenas um (relacionamento muitos para um) (Figura 19). Os atributos de *EPOCH* são:

- ID – Identificador, chave primária;
- EXAM_ID – chave estrangeira que vincula a época ao exame, através do ID do exame;
- START_TIME – tempo em segundos referente ao exame, no qual a época se inicia;
- DURATION – tempo em segundos referente a duração da época;
- DESCRIPTION – descrição da época;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

A entidade *EXAM_EQUIPMENT* representa o relacionamento de *EXAM* com *EQUIPMENT,* no qual ela pode ter um e apenas um *EXAM,* assim como, *EQUIPAMENT* (equipamento) e *UNIT* (unidade) (relacionamento muitos para um).(Figura 19). Abaixo temos os seus atributos:

- ID – Identificador, chave primária;
- EXAM_ID – chave estrangeira que vincula o equipamento ao exame, através do ID do exame;
- EQUIPMENT_ID – chave estrangeira que vincula o exame ao equipamento, através do ID do equipamento;
- AMOUNT – quantidade do equipamento utilizado no exame;
- UNIT_ID – chave estrangeira que vincula a unidade (gramas, litros, unidades, etc) da quantidade do equipamento, através do ID da unidade;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;

- UPDATED_BY - nome do usuário que atualizou o registro.

Já a entidade *EXAM_MEDICAMENT* representa o relacionamento de *EXAM* com *MEDICAMENT,* no qual ela pode ter um e apenas um *EXAM,* assim como, *MEDICAMENT* (medicamento) e *UNIT* (relacionamento muitos para um).(Figura 19). Logo em sequência é exibido seus atributos:

- ID – Identificador, chave primária;
- EXAM_ID – chave estrangeira que vincula o equipamento ao exame, através do ID do exame;
- MEDICAMENT_ID – chave estrangeira que vincula o exame ao medicamento, através do ID do medicamento;
- AMOUNT – quantidade do medicamento que o paciente utiliza/utilizou próximo a data de realização do exame;
- UNIT_ID – chave estrangeira que vincula a unidade (miligramas, comprimidos, mililitros, etc) da quantidade de medicamento consumido, através do ID da unidade;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

EQUIPMENT é a entidade que mapeia os equipamentos utilizados para a realização do exame. Ela se relaciona com a entidade *EXAM_EQUIPMENT* tendo nenhum ou vários (relacionamento um para muitos) (Figura 19). Os atributos de *EQUIPMENT* são:

- ID – Identificador, chave primária;
- NAME – Nome do equipamento;
- DESCRIPTION – descrição do equipamento;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

A entidade *MEDICAMENT* mapeia os medicamentos que os pacientes utilizam ou utilizaram próximo a data de realização do exame. Podendo ter relacionado

a ela nenhum ou vários *EXAM_MEDICAMENT* (relacionamento um para muitos) (Figura 19). Logo abaixo é mostrado seus atributos:

- ID – Identificador, chave primária;
- NAME – nome do medicamento;
- DESCRIPTION – descrição do medicamento;
- CREATED_AT – data de criação;
- CREATED_BY – nome do usuário que criou o registro;
- UPDATED_AT – data de atualização do registro;
- UPDATED_BY - nome do usuário que atualizou o registro.

Na Figura 19 mostra-se o diagrama de entidades e relacionamentos do banco de dados relacional da aplicação de cadastro.

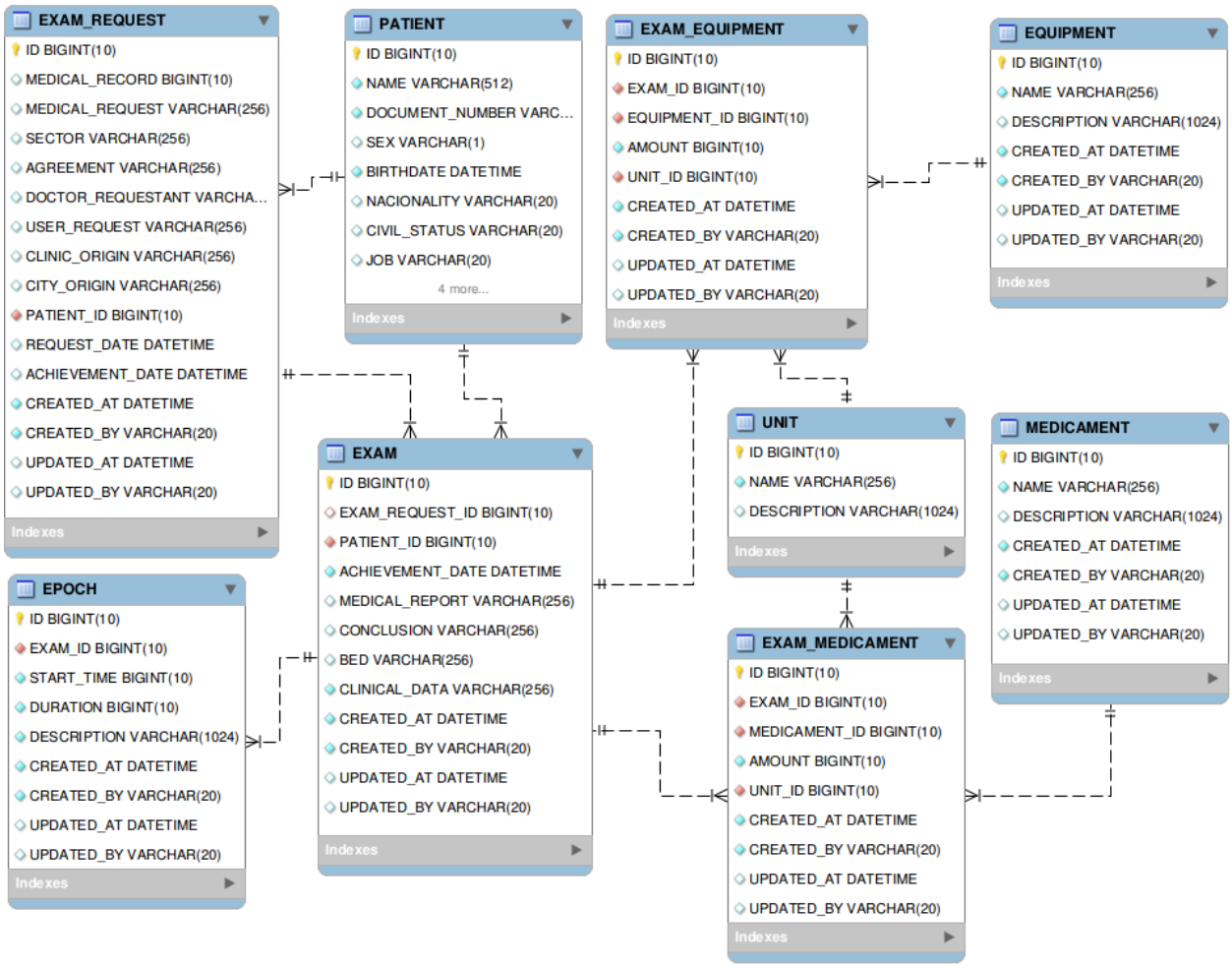


Figura 19 - Diagrama de Entidades e Relacionamento do Banco de Dados da Aplicação

6.2 BACKEND

BackEnd é o termo utilizado para expressar a parte da aplicação na qual o usuário não possui uma percepção direta, ou seja, é a parte da aplicação que faz o processo no lado servidor, para responder às requisições do usuário.

Para a montagem do backEnd utilizou-se o Spring-boot, onde novas aplicações são facilmente criadas, além de uma série de recursos já configurados, tornando o desenvolvimento mais ágil, focando mais nas regras de negócio do que na arquitetura da aplicação. Além disso, optou-se em utilizar o Hibernate para fazer a integração com o banco de dados.

O backEnd ficou encarregado de retornar as informações armazenadas no banco de dados, assim como adicionar e atualizar estas informações. Estes serviços foram disponibilizados através de URL's (*Uniform Resource Locator* - Localizador Padrão de Recursos) que é o endereço de onde está o recurso na rede. Abaixo é mostrado as URL's disponibilizadas e suas ações.

- **/api/equipment**
  - GET: retorna uma lista com todos os equipamentos cadastrados no banco de dados.
- **/api/unit**
  - GET: retorna uma lista com todas as unidades cadastradas no banco de dados.
- **/api/medicament**
  - GET: retorna uma lista com todos os medicamentos cadastrados no banco de dados.
- **/api/epoch**
  - GET: query parâmetro **examId** (identificador do exame) sendo obrigatório, este método retorna as épocas vinculadas ao exame passado como argumento;
  - PUT: Recebe um objeto do tipo exame no formato de json com uma lista de épocas vinculadas ao exame, para que esta lista possa ser inserida/atualizada no banco de dados para o exame em questão.
- **/api/epoch/{id}**
  - GET: retorna a época de id igual ao informado.

- **/api/patient**
  - GET: query parâmetros: **name** (Nome do paciente) e **documentNumber** (CPF do paciente). Pelo menos um destes parâmetros deve ser informado. Este método retorna todos os pacientes cadastrados no banco de dados de acordo com o filtro informado;
  - POST: recebe um objeto do tipo paciente no formato json para ser inserido no banco de dados;
  - PUT: recebe um objeto do tipo paciente no formato json para ser atualizado no banco de dados.
- **/api/patient/{id}**
  - GET: retorna o paciente de id igual ao informado.
- **/api/exam-request**
  - GET: query parâmetros: **medicalRecord** (Identificador do Prontuário), **medicalRequest** (Identificador da Requisição), **patientId** (Identificador do paciente) e **doctorRequestant** (Médico Solicitante). Pelo menos um destes parâmetros deve ser informado. Este método retorna os requerimentos de exame cadastrados no banco de dados de acordo com os filtros informados;
  - POST: recebe um objeto do tipo requerimento de exame no formato json para ser inserido no banco de dados;
  - PUT: recebe um objeto de requerimento de exame no formato json para ser atualizado no banco de dados.
- **/api/exam-request/{id}**
  - GET: retorna o requerimento de exame de id igual ao informado.
- **/api/exam**
  - GET: query parâmetros: **id** (Identificador do exame), **bed** (leito onde o exame foi realizado), **patientId** (Identificador do paciente) e **examRequestId** (Identificador do requerimento do exame). Pelo menos um destes parâmetros deve ser informado. Este método retorna os exames cadastrados no banco de dados de acordo com os filtros informados;
  - POST: recebe um objeto de exame no formato json para ser inserido no banco de dados;

    - PUT: recebe um objeto de exame no formato json para ser atualizado no banco de dados.
- **/api/exam/{id}**
    - GET: retorna o exame de id igual ao informado.
- **/api/exam/medicament**
    - PUT: Recebe um objeto do tipo exame no formato de json com uma lista de medicamentos que o paciente utiliza no período próximo à realização do exame para que esta lista de medicamentos possa ser inserida/atualizada no banco de dados para o exame em questão.
- **/api/exam/equipment**
    - PUT: Recebe um objeto do tipo exame no formato de json com uma lista de equipamentos utilizados para realizar o exame, onde esta lista de equipamentos possa ser inserida/atualizada no banco de dados para o exame em questão.

Para poder utilizar a aplicação, foi criada uma autenticação com usuário e senha em memória. Nela os usuários que podem acessá-la estão cadastrados diretamente no código fonte.

6.3 FRONTEND

FrontEnd é o termo utilizado para expressar a parte da aplicação na qual o usuário possui uma percepção direta, ou seja, é a parte da aplicação em que usuário interage diretamente, comumente de forma visual, através do browser que irá renderizar as páginas WEB.

Para a montagem do frontEnd escolheu-se o framework de javaScript React, que permite componentizar a aplicação e, desta forma, reaproveitar alguns componentes. À vista disso, as páginas e o comportamento que ela deve apresentar são montados em arquivos javaScript, utilizando as funcionalidades do react, bem como HTML, CSS e javaScript. A aplicação foi montada de tal maneira que se tornou responsiva, ou seja, para dispositivos móveis sua visualização é adaptada. No **ANEXO G** pode-se visualizar as páginas criadas, assim como, o seu proposito e a transição entre elas.

6.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a aplicação desenvolvida para o cadastro de exames de eletroencefalograma (EEG), mostrando os relacionamentos das tabelas de banco de dados criada, assim como a apresentação e navegação entre as telas da aplicação.

CAPÍTULO 7 - CONCLUSÕES E TRABALHOS FUTUROS

Destaca-se neste capítulo os aspectos da pesquisa e conclusões do trabalho apresentado nesta dissertação. Ademais, são apresentadas algumas sugestões para trabalhos futuros e a contribuição científica do presente trabalho.

7.1 CONCLUSÕES

Uma das premissas deste trabalho era a de obter melhor controle das versões do programa de processamento do EEG e, também, que este controle fosse mais automático, retirando, assim, uma sobrecarga em apenas uma pessoa responsável pela gestão de versão do código. Para tal tarefa, utilizou-se o Bitbucket, que é um software que perfaz o controle da versão de arquivos. Desta maneira, o código fonte do programa de processamento de EEG está disponível online e o grupo de pesquisa pode acessá-lo facilmente, bastando ter acesso no repositório.

Através do Bitbucket as alterações no código fonte ficam registradas por meio dos *commit's*, nos quais uma mensagem com a descrição dessa alteração é informada pelo usuário e o próprio Bitbucket mostra o que realmente foi alterado no código, apresentando as divergências da versão antiga com a versão atual. Esta funcionalidade facilita a identificação e resolução de possíveis erros que possam ser introduzidos em versões mais recentes. Além do mais, esta ferramenta permite desenvolvimentos paralelos à versão estável do código fonte, por meio da criação de *branchs,* que depois podem ser reintegradas à versão estável do código, fato este que ocorreu durante o desenvolvimento da geração de vídeos para visualização dos quantificadores do EEG.

A reestruturação do software de processamento do EEG possibilitou o reaproveitamento de alguns trechos de código que se repetiam em alguns programas, fato que mitiga o aparecimento ou a correção de erros, uma vez que é necessário corrigi-los apenas em um local ao invés de procurá-lo em vários códigos, nos quais um pode ser facilmente esquecido, sendo de difícil identificação.

Outra vantagem desta reestruturação do código foi a automatização das etapas de processamento do EEG, o que minimizou consideravelmente a interação do usuário com a ferramenta. Inicialmente, o usuário devia rodar cada um dos 6 (seis) programas individualmente, imputando os dados de entrada um por um, e, caso houvesse uma digitação errada de um parâmetro de entrada, o usuário tinha que iniciar o procedimento de processamento do registro em questão novamente, desde o início. Agora, imagine-se realizar esta tarefa para um conjunto grande de EEG's, por exemplo 100 registros; esta tarefa de processar este conjunto grande de dados era bastante árdua e bem suscetível a erros de digitação, devido ao grande número de interações com o usuário.

Já na nova versão, o usuário necessita preencher apenas uma vez uma planilha no formato .CSV com todas as informações de entrada para processar o registro de EEG, podendo colocar nesta planilha todos os registros que ele deseja processar, por exemplo, os 100 registros mencionados anteriormente. Após isto, o usuário precisa chamar apenas uma função do software, selecionar este arquivo .CSV e o programa se encarregara de realizar todo o processamento que os 6 (seis) programas mencionados anteriormente realizavam. O usuário não necessita acompanhar de perto o processamento, podendo realizar outras atividades enquanto o processamento é realizado, diferentemente da primeira versão do software no qual o usuário devia ficar imputando os dados a todo instante. Além do mais, este arquivo .CSV, criado como entrada para o processamento, pode ser utilizado como documentação e reprodução do estudo por outros grupos de pesquisa.

Foi desenvolvido, também, um teste de regressão para validar as alterações que eram realizadas no software de processamento, que consiste em testar se as variáveis geradas na nova versão são relativamente iguais aos valores gerados na versão anterior. Validação esta que tem por objetivo evitar a introdução de erros na nova versão, assim como, dar uma maior confiabilidade na realização de alterações no código, trazendo uma maior qualidade ao software de processamento.

A visualização dos quantificadores do EEG extraídos pelo software de processamento é importante, pois facilita o acompanhamento dos mesmos, visto que comumente tem-se um valor para cada eletrodo, ou seja, tem-se em torno de 20 eletrodos/valores e visualizá-los apenas em forma numérica não é bastante intuitivo, uma vez que cada eletrodo representa uma região do escalpo cerebral. Assim,

visualizações destes valores de forma topográfica, representando o escalpo cerebral, ficam mais intuitivas, pois se percebe com mais clareza qual a região do escalpo está sendo mais ativa e/ou menos ativa. Consequentemente, a montagem do vídeo da visualização topográfica das bandas de frequência traz a percepção da evolução temporal dos quantificadores do EEG. Portanto, a visualização dos quantificadores deixa a intelecção dos mesmos mais fácil e, também, facilita a extração de novas hipóteses/informações destes dados.

O desenvolvimento da plataforma de cadastro de EEG veio com o intuito de gerir melhor o armazenamento dos registros, uma vez que a base de dados do grupo de pesquisa está ficando enorme, com mais de 200 coletas. Uma gestão manual do mesmo, sem o apoio de ferramentas computacionais acaba se tornando dispendioso, podendo levar à perda de registros, bem como à dificuldade em pesquisar registros de interesse.

7.2 TRABALHOS FUTUROS

Como trabalhos futuros para a ferramenta de processamento de dados do EEG, têm-se:

- Otimizar a performance de processamento do EEG para que se possa utilizá-lo em um monitoramento de tempo real;
- Construir uma interface gráfica para utilização do mesmo;
- Adicionar novos quantificadores de EEG;
- Adicionar algoritmos de atenuação de ruído;
- Fazer integrações com outros software de processamento.

Para a visualização do EEG, possíveis trabalhos futuros são:

- Aprimorar a geração das imagens topográficas construídas pelo grupo de pesquisa para que o mesmo possa performar de forma mais rápida e, quem sabe, assim, utilizá-la em um sistema de monitoramento de tempo real;
- Pesquisar novas maneiras de visualizar os quantificadores do EEG.

Já para a plataforma de cadastro de registros de EEG, como trabalhos futuros têm-se:

- Criar uma tela de entrada para o usuário digitar seu usuário e senha;
- Adicionar algumas páginas WEB para descrição da plataforma, do grupo de pesquisa e do objetivo da plataforma;
- Mudar o cadastro de usuários para acessar a aplicação de memória para o banco de dados;
- Criar políticas de segurança para o acesso aos dados, como, por exemplo, visualização do nome e CPF do paciente;
- Armazenar os arquivos PLG's referentes aos registros de EEG, para fácil acesso e gestão dos dados;
- Disponibilizar a aplicação/plataforma para a comunidade, para que outros grupos de estudos possam utilizar a base de dados, assim como compartilhar informações;
- Adicionar na aplicação o cadastro dos resultados realizados durante o processamento do EEG;
- Montagem de um *BigData* com os resultados do processamento;
- Aplicar técnicas de *Busines Inteligence* no *bigData*;
- Construir uma tela de busca de exames mais personalizada, permitindo buscar grupos de interesse de forma mais fácil e intuitiva, para novos estudos de caso;

7.3 CONSIDERAÇÕES FINAIS

Este trabalho contemplou todo o decurso de gerenciamento dos registros de EEG coletados até o seu processamento, bem como, a visualização dos quantificadores extraídos dos EEG's.

Além disso, as aplicações desenvolvidas visam facilitar o dia a dia dos trabalhos envolvendo EEG, automatizando algumas tarefas e facilitando a visualização dos resultados.

Do ponto de vista científico, as principais contribuições deste trabalho são:

a) Otimização do tempo do pesquisador, pois ele pode deixar processando os registros de EEG e, em paralelo, pode realizar outras atividades, uma vez que o processo ficou bastante automatizado;

b) Gerenciamento dos registros de EEG coletados, facilitando os seus cadastros e a recuperação dos mesmos através de campos de buscas intuitivos;

c) Várias formas de se visualizar os quantificadores extraídos do software de processamento do EEG.

REFERÊNCIAS

AGUERA, P. E. et al. ELAN: A software package for analysis and visualization of MEG, EEG, and LFP signals. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–11, Janeiro, 2011.
https://doi.org/10.1155/2011/158970
PMid:21687568 PMCid:PMC3113286

APACHE MAVEN PROJECT. **Maven**. Disponível em: <https://maven.apache.org/>. Acesso em: 20 jan. 2019.

ATLASSIAN. **Bitbucket. Site Oficial Atlassian**. Disponível em: <https://bitbucket.org/product>. Acesso em: 20 out. 2018.

BAO, F. S.; LIU, X.; ZHANG, C. PyEEG: An open source python module for EEG/MEG feature extraction. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–7, Dezembro, 2011.
https://doi.org/10.1155/2011/406391
PMid:21512582 PMCid:PMC3070217

BLUM, S. et al. EEG Recording and Online Signal Processing on Android: A Multiapp Framework for Brain-Computer Interfaces on Smartphone. **BioMed Research International**, v. 2017, p. 1–12, Novembro, 2017.
https://doi.org/10.1155/2017/3072870
PMid:29349070 PMCid:PMC5733949

BRAIN PRODUCTS GMBH. **Analyzer 2. Site Oficial Brain Products**. Disponível em: <https://www.brainproducts.com/productdetails.php?id=17>. Acesso em: 20 out. 2018.

BRUNET, D.; MURRAY, M. M.; MICHEL, C. M. Spatiotemporal analysis of multichannel EEG: CARTOOL. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–15, Novembro, 2011.
https://doi.org/10.1155/2011/813870
PMid:21253358 PMCid:PMC3022183

CENTRO DE SAÚDE DO TRABALHO – CST MED. **Eletroencefalograma. Site Oficial CST Med**. Disponível em: <http://www.cst-eng.com.br/encefalo.htm>. Acesso em: 20 nov. 2018.

COINTEPAS, Y. et al. BrainVISA: Software platform for visualization and analysis of multi-modality brain data. **NeuroImage**, n. 6, p. 98, Junho, 2001.
https://doi.org/10.1016/S1053-8119(01)91441-7

COMBRISSON, E. et al. Sleep: An Open-Source Python Software for Visualization, Analysis, and Staging of Sleep Data. **Frontiers in Neuroinformatics**, v. 11, p. 1–11, September, 2017.
https://doi.org/10.3389/fninf.2017.00060
PMid:28983246 PMCid:PMC5613192

DELORME, A. et al. EEGLAB, SIFT, NFT, BCILAB, and ERICA: New tools for advanced EEG processing. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–12, Fevereiro, 2011.
https://doi.org/10.1155/2011/130714
PMid:21687590 PMCid:PMC3114412

DESTRO-FILHO, J. B.; FARIA, V. N. R. ANÁLISE DE SINAL MEA BASEADA EM TESTES ESTOCÁSTICOS DESCREVENDO A ESTRUTURA DE DENSIDADE DE PROBABILIDADE.

XI Simpósio de Engenharia Biomédica – SEB, p. 20–23, 2018.
https://doi.org/10.29327/xiseb.128303

ESCH, L. et al. MNE Scan: Software for real-time processing of electrophysiological data. **Journal of Neuroscience Methods**, v. 303, p. 55–67, Abril, 2018.
https://doi.org/10.1016/j.jneumeth.2018.03.020
PMid:29621570

ESTRELA, C. **Metodologia Científica: Ciência, Ensino, Pesquisa**. 3. ed. Porto Alegre: Artes Medicas, 2018.

GITHUB. **github. Site Oficial Github**. Disponível em: <https://github.com/pricing>. Acesso em: 20 out. 2018.

GRAMFORT, A. et al. MEG and EEG data analysis with MNE-Python. **Frontiers in Neuroscience**, v. 7, n. 7 DEC, p. 1–13, Dezembro, 2013.
https://doi.org/10.3389/fnins.2013.00267
PMid:24431986 PMCid:PMC3872725

H2. **H2 Database Engine. Site Oficial H2**. Disponível em: <http://www.h2database.com/html/main.html>. Acesso em: 20 out. 2018.

HEROKU. **How Heroku Works. Site oficial Heroku**. Disponível em: <https://devcenter.heroku.com/articles/how-heroku-works>. Acesso em: 25 out. 2018.

HIBERNATE. **Hibernate ORM. Site Oficial Hibernate**. Disponível em: <http://hibernate.org/orm/>. Acesso em: 20 out. 2018.

KNOPP, S. J. et al. A software framework for real-time multi-modal detection of microsleeps. **Australasian Physical and Engineering Sciences in Medicine**, v. 40, n. 3, p. 739–749, Junho, 2017.
https://doi.org/10.1007/s13246-017-0559-x
PMid:28573545

KUBOTA, Y. et al. Continuous EEG monitoring in ICU. **Journal of Intensive Care**, v. 6, n. 1, p. 1–8, Julho, 2018.
https://doi.org/10.1186/s40560-018-0310-z
PMid:30026951 PMCid:PMC6050674

LAUCSEN DA ROSA, D. **Sistema de Processamento de Sinais Biomédicos: Filtragem de Sinais de Eletroencefalograma**. Florianópolis: Dissertação de Mestrado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina, 2009.

LEVIN, A. R. et al. BEAPP: The batch electroencephalography automated processing platform. **Frontiers in Neuroscience**, v. 12, n. AUG, p. 1–13, Agosto, 2018.
https://doi.org/10.3389/fnins.2018.00513
PMid:30131667 PMCid:PMC6090769

LOPES DA SILVA, F. EEG and MEG: Relevance to neuroscience. **Neuron**, v. 80, n. 5, p. 1112–1128, Dezembro, 2013.
https://doi.org/10.1016/j.neuron.2013.10.017
PMid:24314724

LOPEZ-CALDERON, J.; LUCK, S. J. ERPLAB: an open-source toolbox for the analysis of event-related potentials. **Frontiers in Human Neuroscience**, v. 8, n. April, p. 1–14, Abril,

2014.
https://doi.org/10.3389/fnhum.2014.00213
PMid:24782741 PMCid:PMC3995046

M., A. et al. Noninvasive brain-computer interfaces for augmentative and alternative communication. **IEEE Reviews in Biomedical Engineering**, v. 7, p. 31–49, Dezembro, 2014.
https://doi.org/10.1109/RBME.2013.2295097
PMid:24802700 PMCid:PMC6525622

MICROSOFT. **Criar ou editar arquivos .csv para importação para o Outlook. Site ofical Microsoft para Suporte ao Word**. Disponível em: <https://support.office.com/pt-br/article/criar-ou-editar-arquivos-csv-para-importação-para-o-outlook-4518d70d-8fe9-46ad-94fa-1494247193c7>. Acesso em: 20 out. 2018.

NIEDERMEYER, E.; SILVA, F. L. DA. **Electroencephalography: Basic Principles, Clinical Applications, and Related Fields**. 5. ed. Philadelphia: Lippincott Williams & Wilkins, 2004.

OOSTENVELD, R. et al. FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–9, Outubro, 2011.
https://doi.org/10.1155/2011/156869
PMid:21253357 PMCid:PMC3021840

ORACLE. **What is Java technology and why do I need it?. Site Oficial Oracle**. Disponível em: <https://java.com/en/download/faq/whatis_java.xml>. Acesso em: 20 out. 2018.

ORACLE. **Database Concepts. Site Oficial Oracle**. Disponível em: <https://docs.oracle.com/database/121/CNCPT/intro.htm#CNCPT001>. Acesso em: 22 out. 2018.

PASCUAL-MARQUI, R. D. et al. Low resolution brain electromagnetic tomography (LORETA) functional imaging in acute, neuroleptic-naive, first-episode, productive schizophrenia. **Psychiatry Research: Neuroimaging**, v. 90, p. 169–179, Junho, 1999.
https://doi.org/10.1016/S0925-4927(99)00013-X

PEYK, P.; DE CESAREI, A.; JUNGHÖFER, M. Electromagnetic encephalography software: Overview and integration with other EEG/MEG toolboxes. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–10, Janeiro, 2011.
https://doi.org/10.1155/2011/861705
PMid:21577273 PMCid:PMC3090751

POSTGRESQL. **About. Site Oficial PostgreSQL**. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 22 out. 2018.

RAMACHANDRAN, N. P. et al. Design and Implementation of an Open-Source Browser-based Laboratory Platform for EEG Data Analysis. **2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**, p. 2313–2317, 2018.
https://doi.org/10.1109/ICACCI.2018.8554786

REACT. **Tutorial: Intro to React. Site Oficial REACT**. Disponível em: <https://reactjs.org/tutorial/tutorial.html>. Acesso em: 20 out. 2018.

SCHALK, G.; MELLINGER, J. **A Practical Guide to Brain–Computer Interfacing with BCI2000: General-Purpose Software for Brain-Computer Interface Research, Data Acquisition, Stimulus Presentation, and Brain Monitoring**. 2010. ed. New York: Springer,

2010.
https://doi.org/10.1007/978-1-84996-092-2

SCHMITZ, D. **Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar**. Disponível em: <https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>. Acesso em: 7 out. 2018.

SCHÖLGL, A.; VIDAURRE, C.; SANDER, T. H. BioSig: The free and open source software library for biomedical signal processing. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–12, Dezembro, 2011.
https://doi.org/10.1155/2011/935364
PMid:21437227 PMCid:PMC3061298

SLORETA. **Limitation of use. Site Oficial SLORETA**. Disponível em: <http://www.uzh.ch/keyinst/NewLORETA/sLORETA/04Slor.html>. Acesso em: 16 jul. 2018.

STARUML. **StarUML documentation. Site Oficial StarUML**. Disponível em: <https://docs.staruml.io/>. Acesso em: 20 out. 2018.

STRIK, W. K. et al. Three-dimensional tomography of event-related potentials during response inhibition: Evidence for phasic frontal lobe activation. **Electroencephalography and Clinical Neurophysiology - Evoked Potentials**, v. 108, n. 4, p. 406–413, Junho, 1998.
https://doi.org/10.1016/S0168-5597(98)00021-5

SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE. **What is EEGLAB? Site Oficial SCCN**. Disponível em: <https://sccn.ucsd.edu/eeglab/index.php>. Acesso em: 10 out. 2018.

TADEL, F. et al. Brainstorm: A user-friendly application for MEG/EEG analysis. **Computational Intelligence and Neuroscience**, v. 2011, p. 1–13, Janeiro, 2011.
https://doi.org/10.1155/2011/879716
PMid:21584256 PMCid:PMC3090754

VON WEGNER, F.; LAUFS, H. Information-Theoretical Analysis of EEG Microstate Sequences in Python. **Frontiers in Neuroinformatics**, v. 12, p. 1–10, Junho, 2018.
https://doi.org/10.3389/fninf.2018.00030
PMid:29910723 PMCid:PMC5992993

W3SCHOOLS [A]. **What is JavaScript?. Site Oficial W3SCHOOLS**. Disponível em: <https://www.w3schools.com/whatis/whatis_js.asp>. Acesso em: 20 out. 2018.

W3SCHOOLS [B]. **Node.js Introduction. Site Oficial W3SCHOOLS**. Disponível em: <https://www.w3schools.com/nodejs/nodejs_intro.asp>. Acesso em: 20 out. 2018.

W3SCHOOLS [C]. **CSS Introduction. Site Oficial W3SCHOOLS**. Disponível em: <https://www.w3schools.com/css/css_intro.asp>. Acesso em: 20 out. 2018.

W3SCHOOLS [D]. **HTML Introduction Site Oficial W3SCHOOLS**. Disponível em: <https://www.w3schools.com/html/html_intro.asp>. Acesso em: 21 out. 2018.

W3SCHOOLS [E]. **Introduction to SQL. Site Oficial W3SCHOOLS**. Disponível em: <https://www.w3schools.com/sql/sql_intro.asp>. Acesso em: 21 out. 2018.

WARE, C. **Information Visualization: Perception for Design**. 3. ed. Waltham: Morgan Kaufmann, 2012.

WEBB, P. et al. **Spring Boot Reference Guide**. Disponível em: <https://docs.spring.io/spring-boot/docs/2.1.0.RELEASE/reference/htmlsingle/#boot-documentation>. Acesso em: 21 out. 2018.

WEBPACK. **Concepts**. Disponível em: <https://webpack.js.org/concepts>. Acesso em: 20 jan. 2019.

WINK, A. S. **Android Platform Analysis from the Software Engineering perspective**. Porto Alegre: Trabalho de Conclusão de Curso em Engenharia da Computação da Universidade Federal do Rio Grande do Sul, 2016.

WOLPAW, J. R. et al. Brain–computer interfaces for communication and control. **Clinical Neurophysiology**, v. 113, p. 767–791, Junho, 2002. https://doi.org/10.1016/S1388-2457(02)00057-3

ZANOW, F.; KNÖSCHE, T. R. ASA - Advanced Source Analysis of continuous and event-related EEG/MEG signals. **Brain Topography**, v. 16, n. 4, p. 287–290, Junho, 2004. https://doi.org/10.1023/B:BRAT.0000032867.41555.d0

ANEXO A - Diagramas de execução da nova versão do software de processamento do EEG

Diagrama Execução P1_CONVERSOR

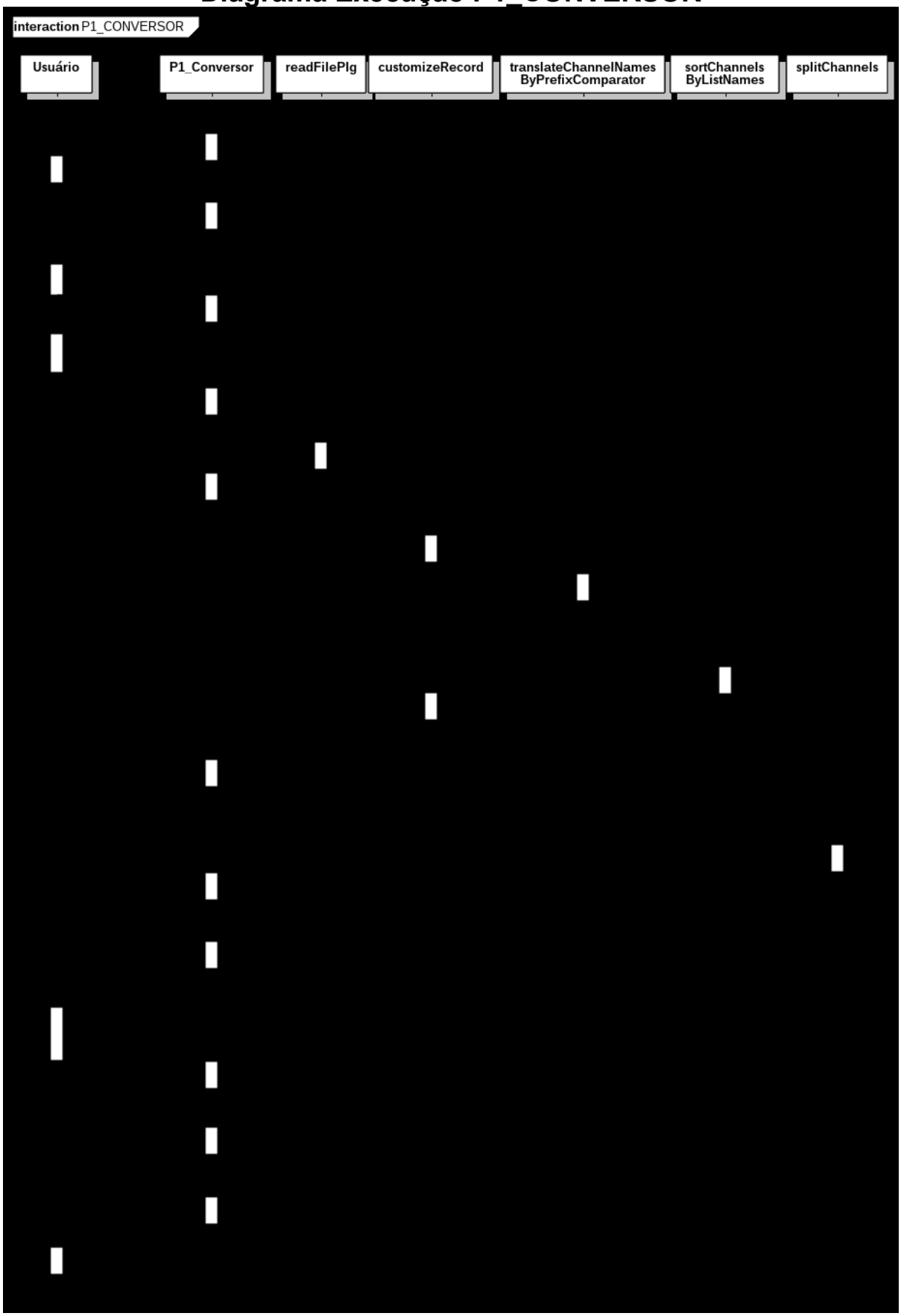


Diagrama Execução P2_EPOCAS

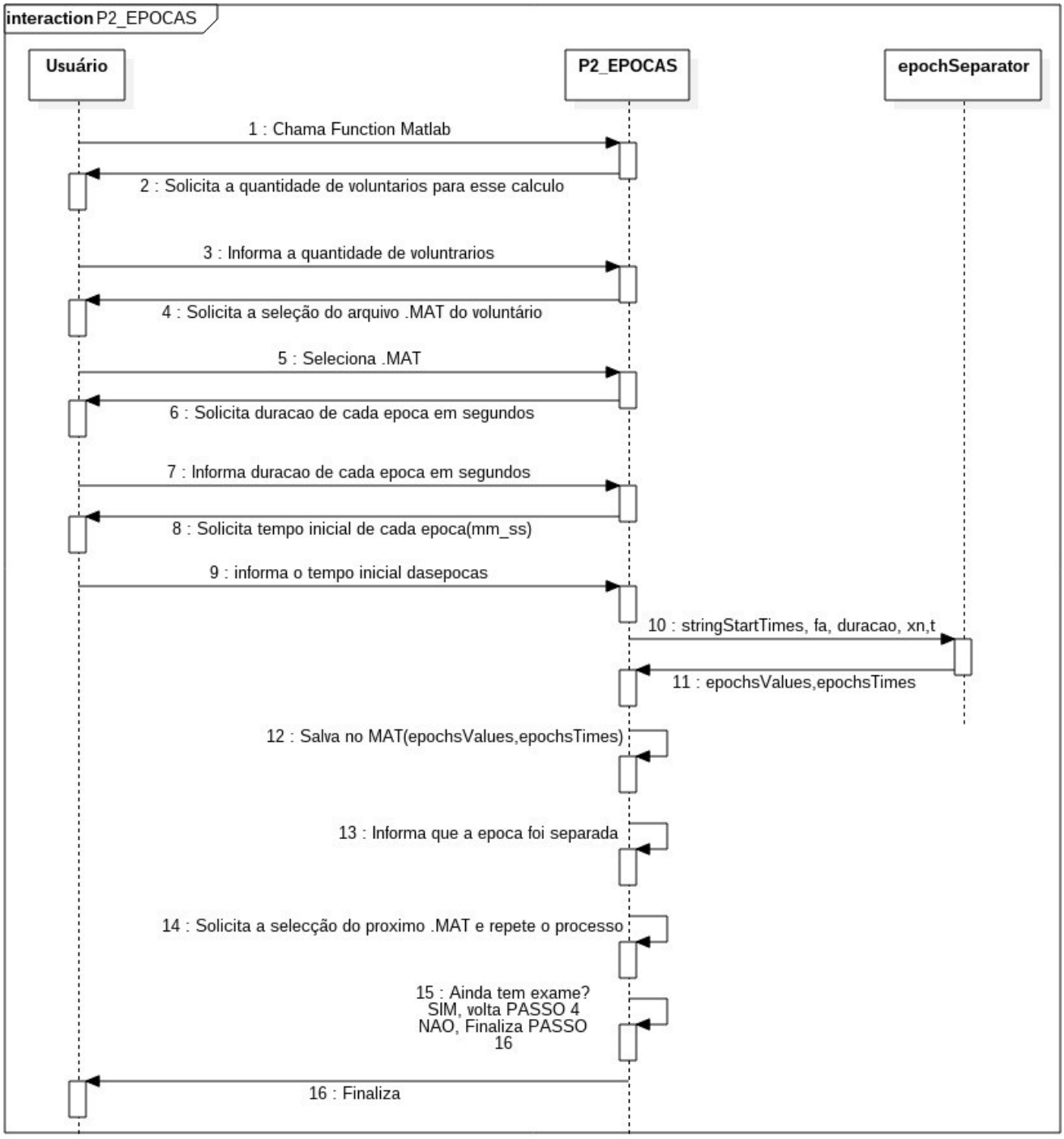


Diagrama Execução P3_VALIDAEEG

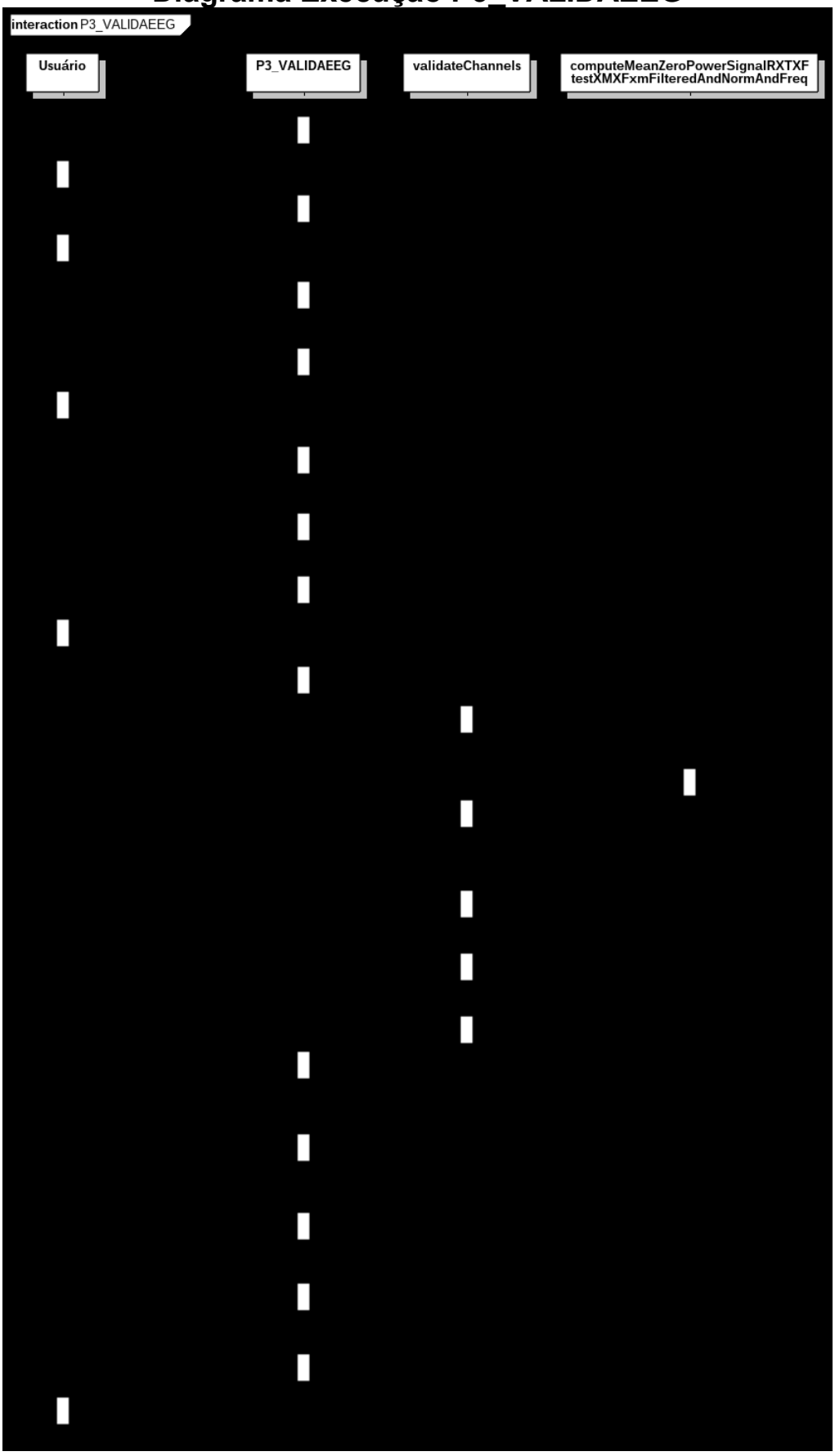


Diagrama Execução P4_PCP

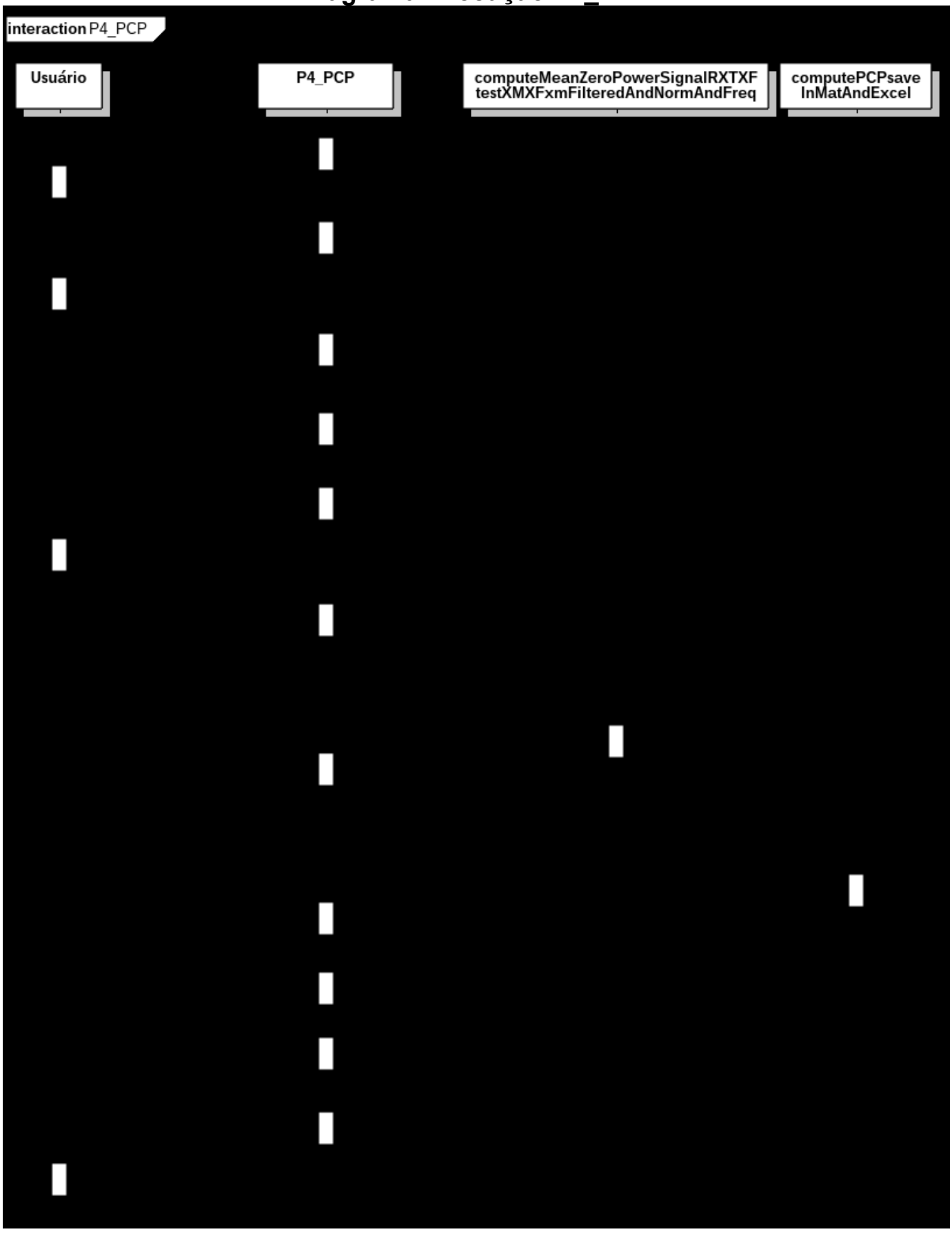


Diagrama Execução P5_FM

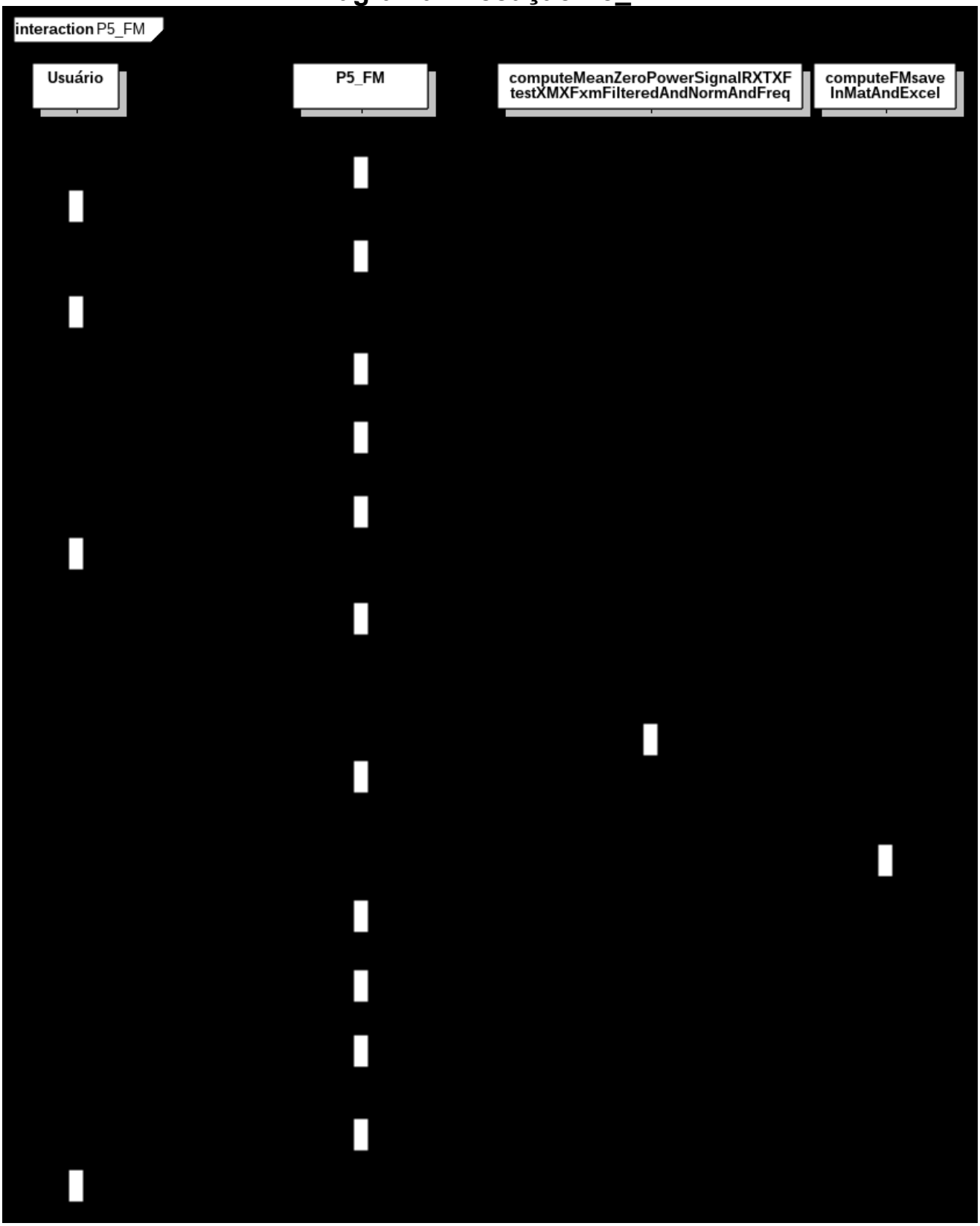


Diagrama Execução P6_NOVO_COR

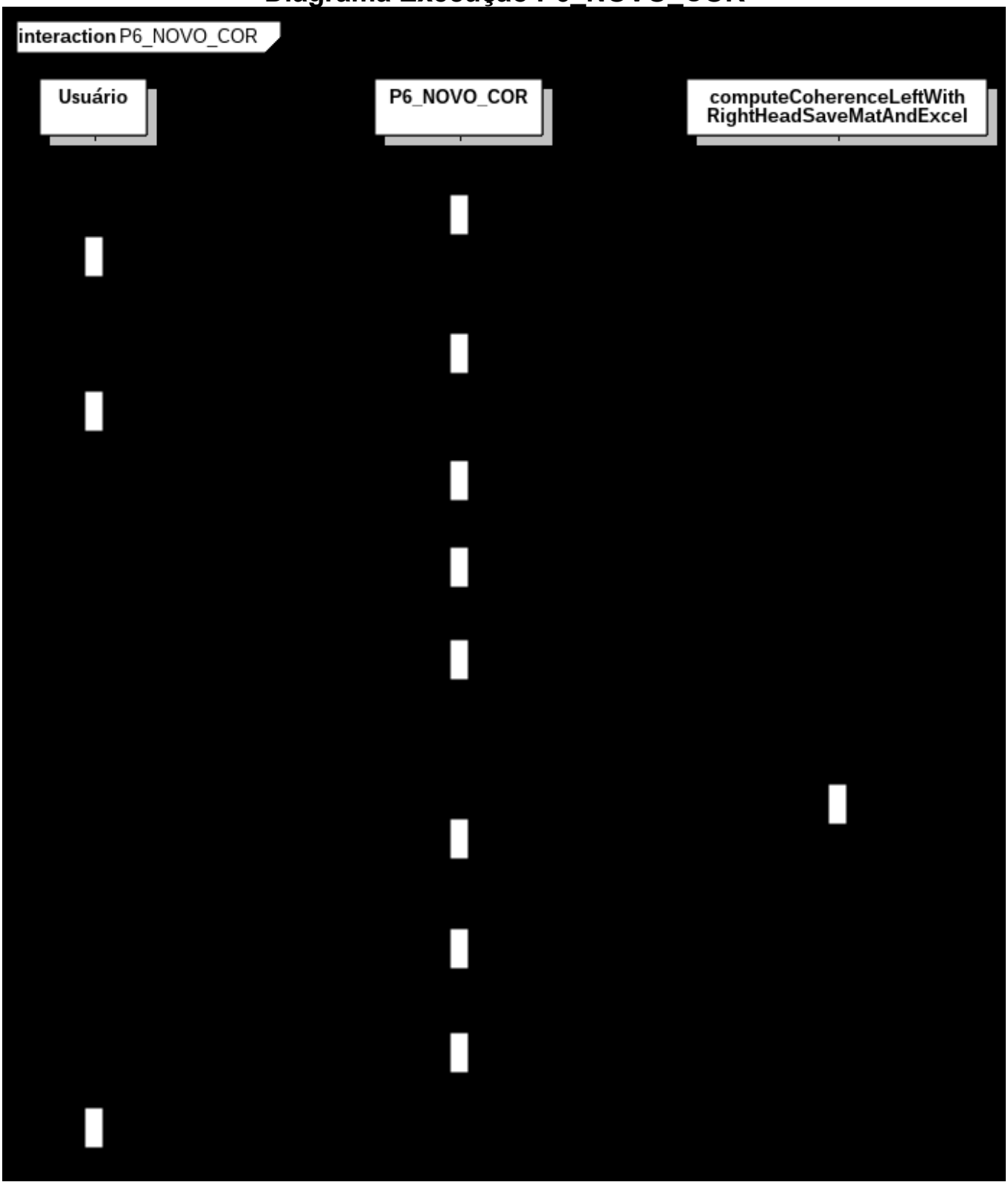


Diagrama Execução computeMeanZeroPowerSignalRXTXFtestXMXFxmFilteredAndNormAndFreq

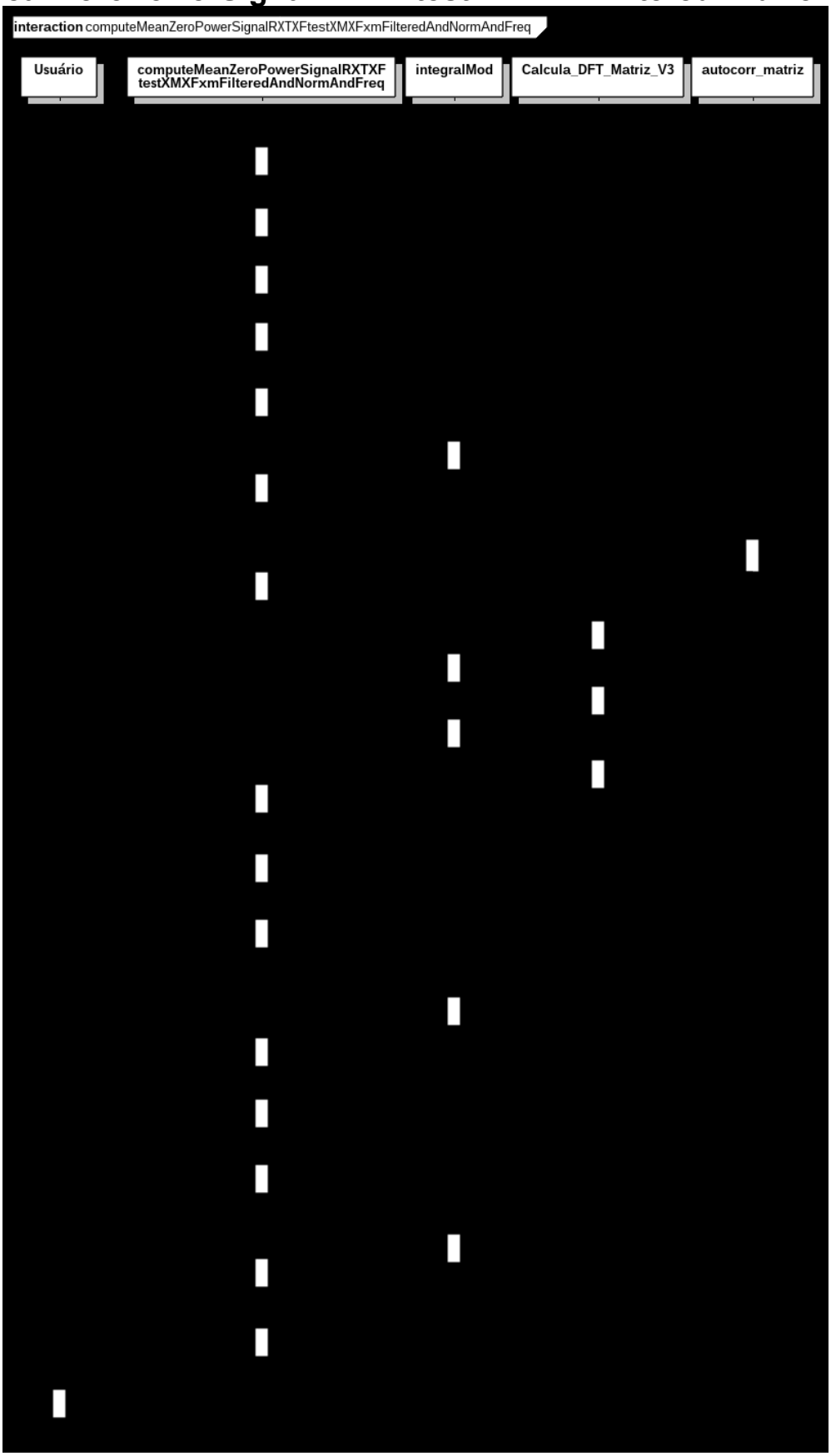


Diagrama Execução computePCPsaveInMatAndExcel

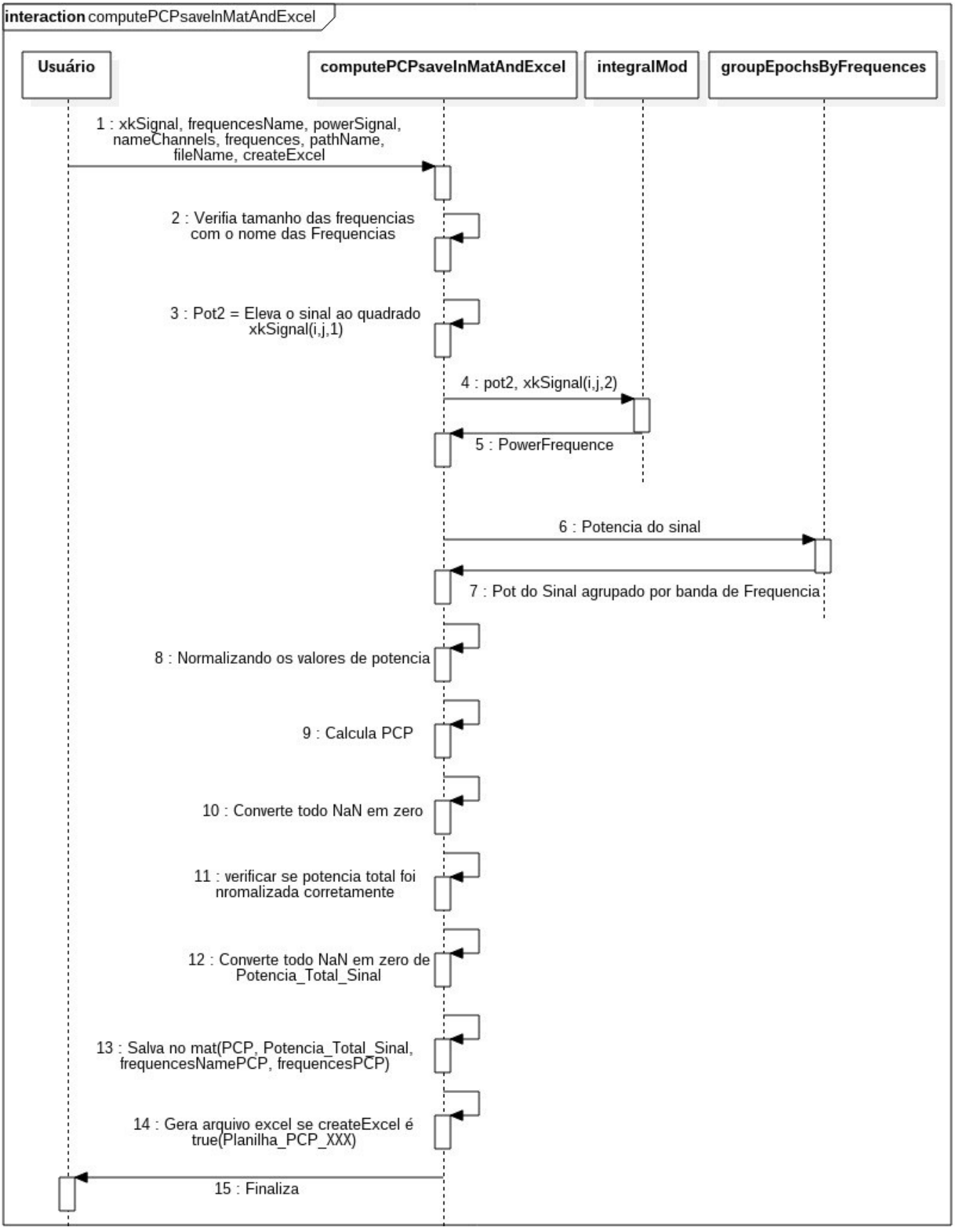


Diagrama Execução computeFMsaveInMatAndExcel

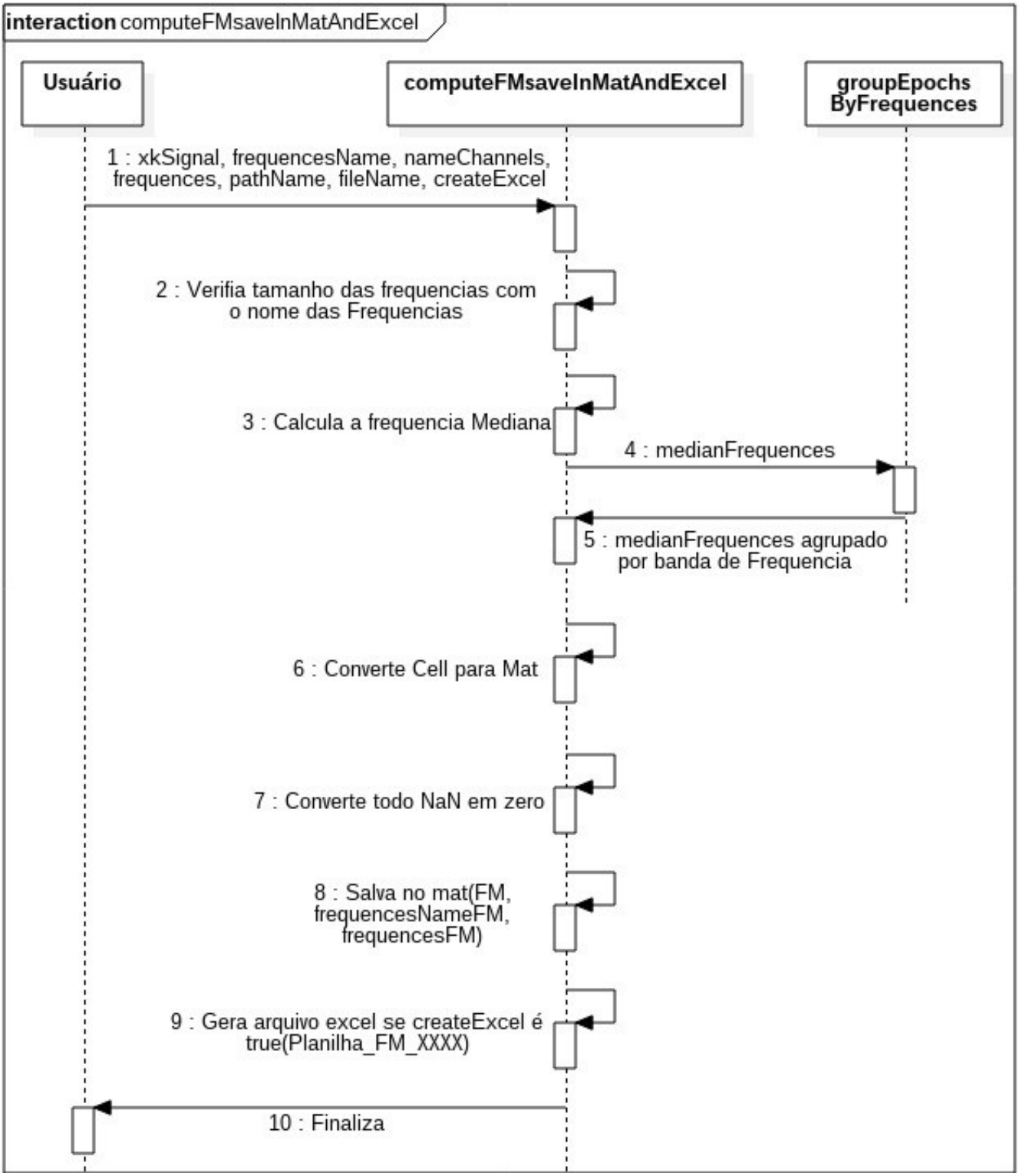


Diagrama Execução computeCoherenceLeftWithRightHeadSaveMatAndExcel

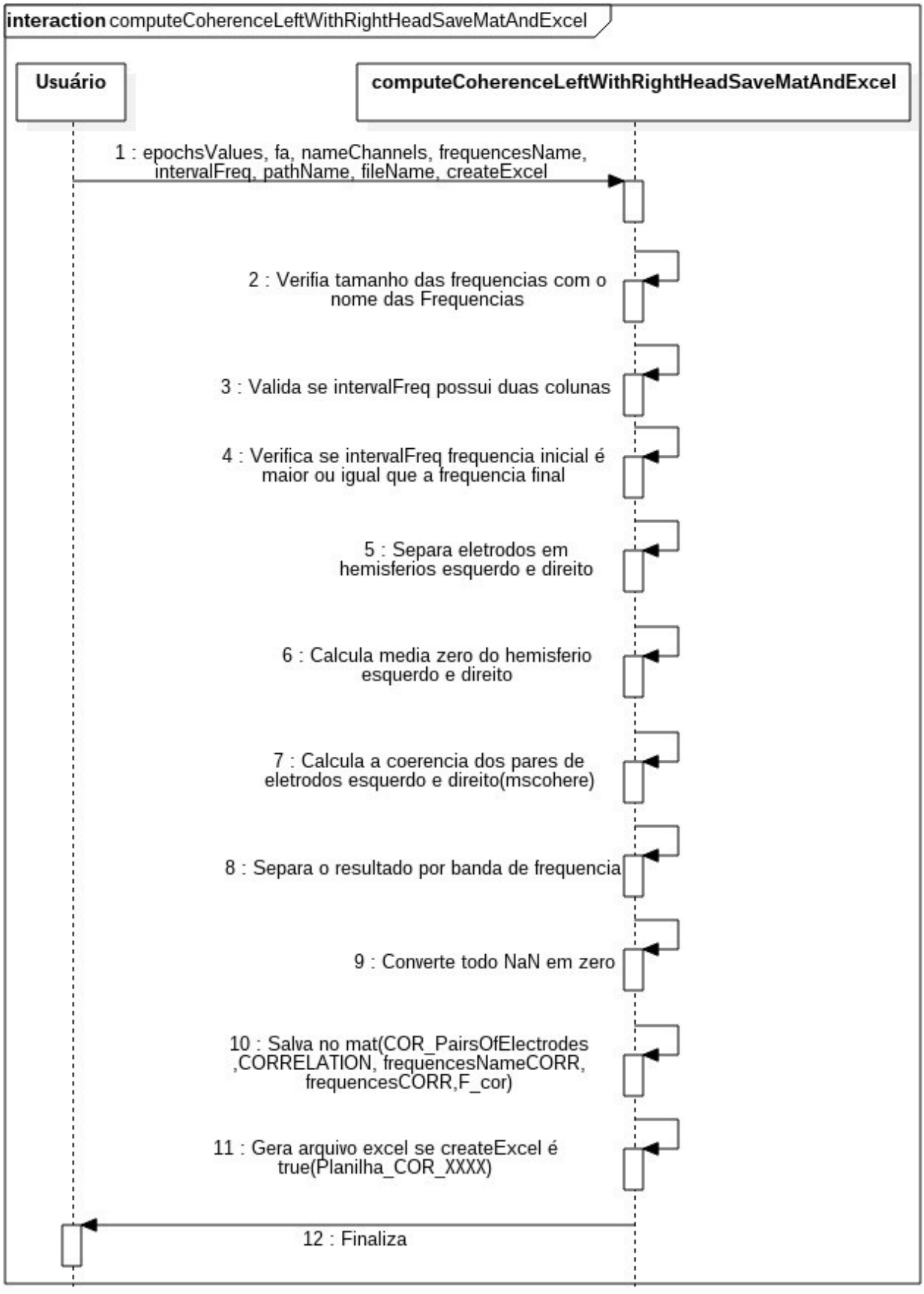


Diagrama Execução saveFileMat parte 1

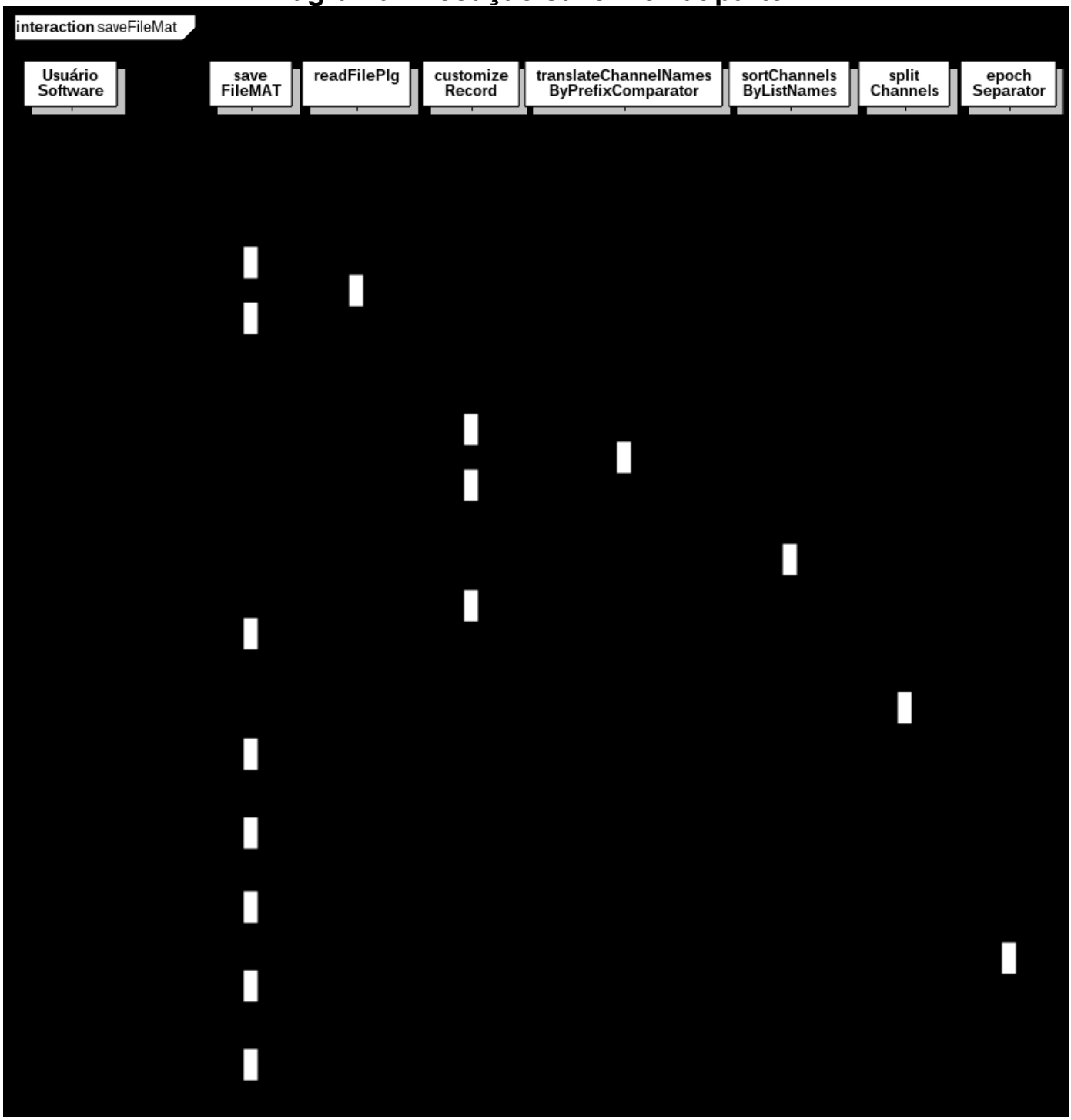


Diagrama Execução saveFileMat parte 2

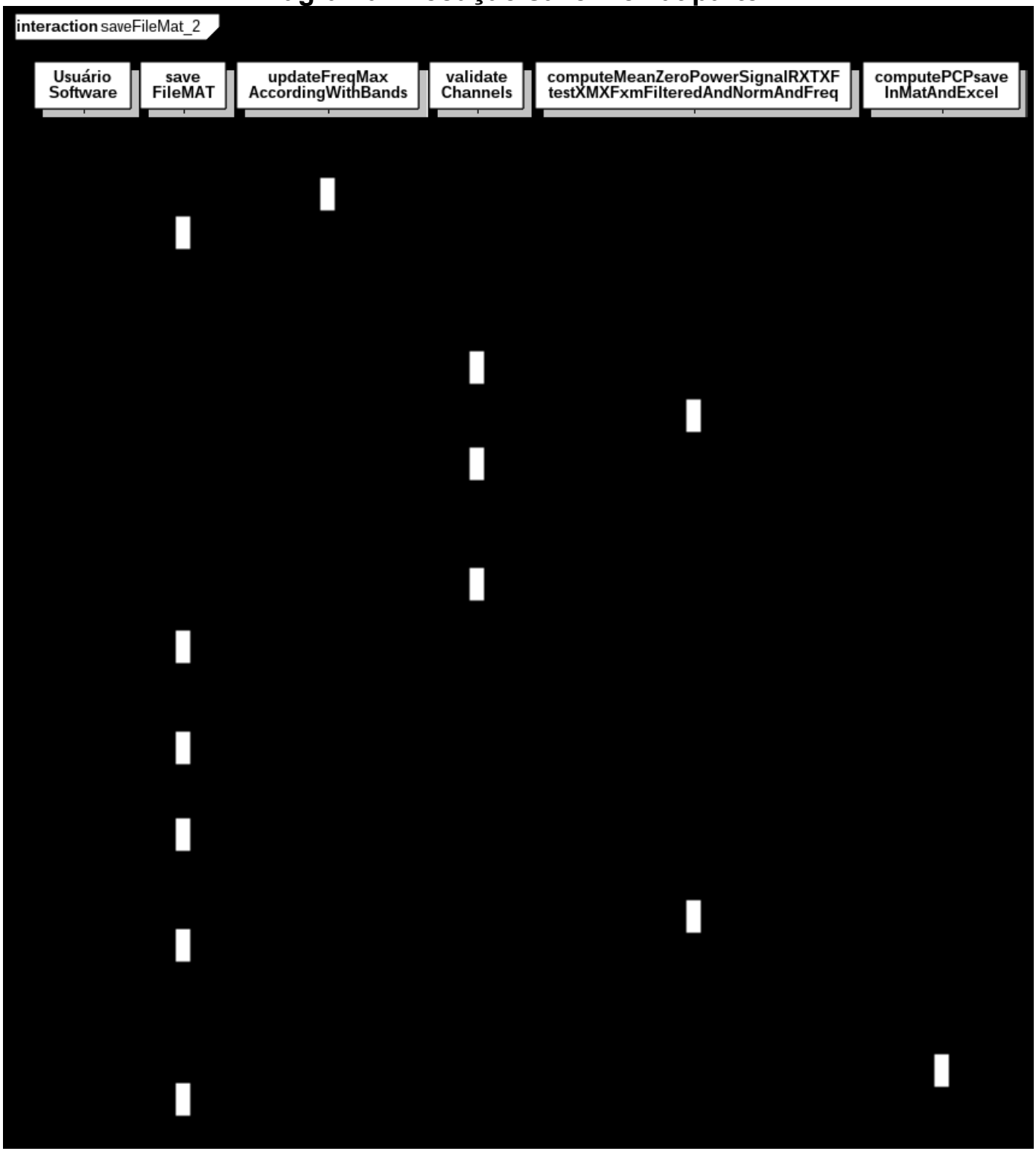


Diagrama Execução saveFileMat parte 3

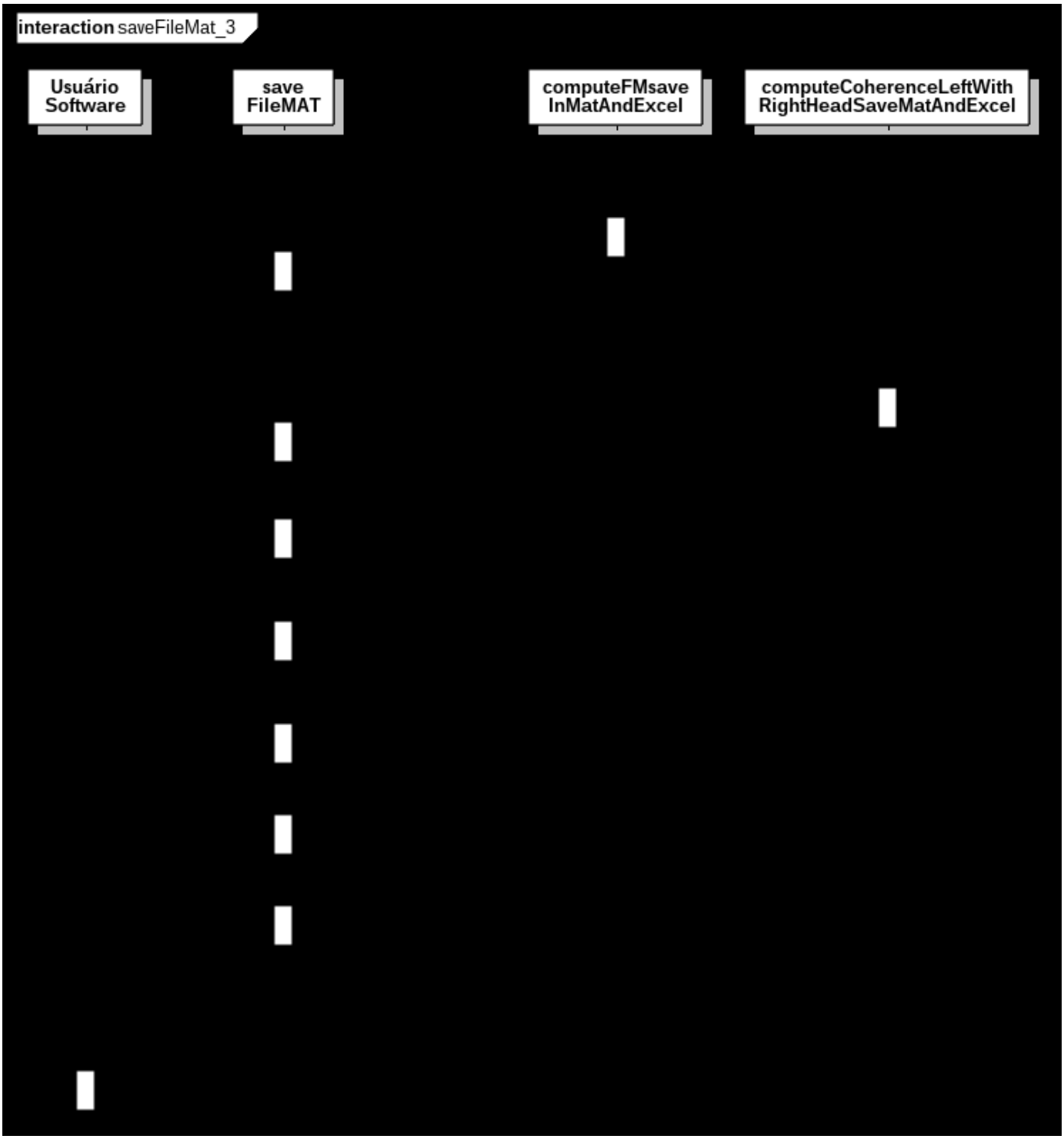
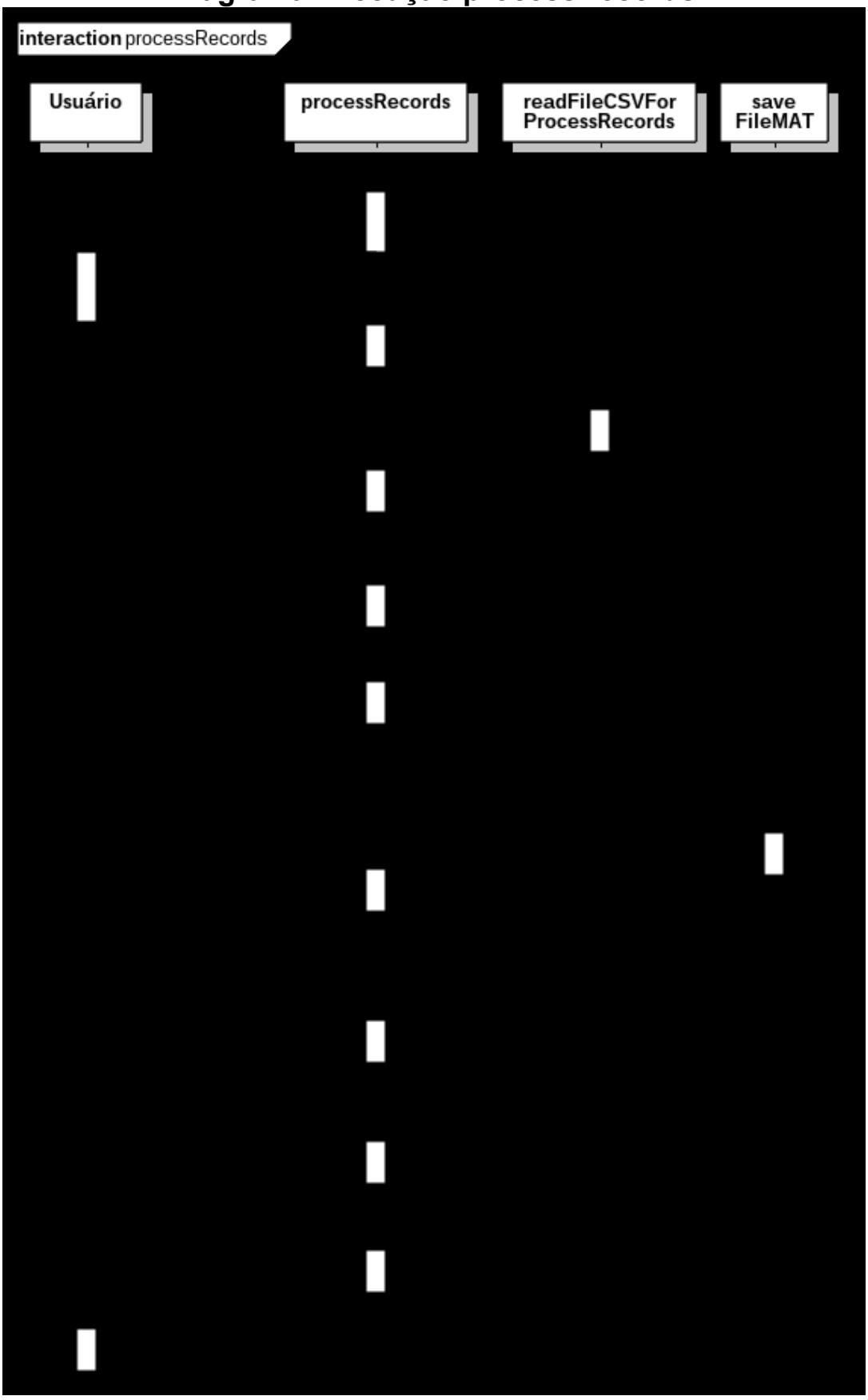


Diagrama Execução processRecords



ANEXO B – Arquivo.CSV de entrada para o processRecords

Nome Arquivo PLG	Duração Epocas (seg)	Qtd. Epo-cas	Ep1	Ep2	Ep3	Ep4	Ep5	Ep6	Ep7	Ep8	Ep9	Ep10	Qtd Ruido-sos	Canal Ruido 1	Canal Ruido 2	Canal Ruido 3	Normal /Coma	Fil-tro	Nome Saída	Gerar Excel
0011901'	2	10	01:49	01:51	01:53	01:55	01:58	02:00	02:14	02:23	02:41	02:45					COMA	35	C0011901_ME_OUTROS	SIM
0013501'	2	10	12:28	12:30	12:32	12:34	12:36	12:38	12:42	12:49	12:56	13:13	3	FP1	FP2	F4	COMA	35	C0013501_ME_COMA_METABOLICO	SIM
0303001'	2	10	03:56	03:58	04:05	04:07	04:26	05:13	06:06	06:21	08:27	09:29					COMA	35	C0303001_ME_OUTROS	SIM
0372501'	2	10	15:44	16:11	17:23	17:44	18:21	18:36	18:48	18:50	19:00	19:02	3	F4	F8	Oz	COMA	35	C0372501_ME_OUTROS	SIM
0373701'	2	10	09:53	09:55	09:57	10:01	10:03	10:28	10:35	11:02	11:52	12:17	2	T5	F7		COMA	35	C0373701_COMA_METABÓLICO	SIM
0375201'	2	10	10:36	10:38	10:40	10:42	10:44	11:23	11:28	11:35	12:16	12:23	1	Oz			COMA	35	C0375201_ME_COMA_METABÓLICO	SIM
0726001'	2	10	00:32	00:34	00:36	00:38	00:46	01:00	01:39	02:05	02:45	03:01	2	FP2	F4		COMA	35	C0726001_ME_OUTROS	SIM
0932801'	2	10	00:01	00:03	00:05	00:07	00:22	00:27	00:44	01:04	01:15	01:38	1	FP2			COMA	35	C0932801_ME_OUTROS	SIM
1045901'	2	10	02:54	02:56	02:58	03:00	03:10	04:36	05:23	06:30	06:50	09:23					COMA	35	C1045901_ME_AVE	SIM
1048001'	2	10	00:48	00:50	00:52	00:54	01:33	02:24	03:03	05:00	07:26	07:39					COMA	35	C1048001_ME_TCE	SIM
1054001'	2	10	00:56	04:04	04:37	05:43	06:08	06:54	07:25	07:27	07:29	07:31					COMA	35	C1054001_ME_AVE	SIM
1054501'	2	10	01:05	01:07	01:09	01:11	01:35	02:12	03:23	04:50	05:59	08:02					COMA	35	C1054501_ME_TCE	SIM

ANEXO C – Arquivo CSV de resultado do processamento do ANEXO B apenas com as colunas adicionais

Nome Arquivo PLG	Conversão?	Canais Ruidosos	Exame Valido (POUCO RUIDO)?	PCP	Frequência Mediana	Coerência	FreqMax Adotada
0011901'	SUCCESS		SIM	SIM	SIM	SIM	30.00
0013501'	SUCCESS		SIM	SIM	SIM	SIM	30.00
0303001'	SUCCESS	F7	SIM	SIM	SIM	SIM	30.00
0372501'	SUCCESS	F7	NAO	SIM	SIM	SIM	30.00
0373701'	SUCCESS	F7	SIM	SIM	SIM	SIM	30.00
0375201'	SUCCESS	F7	SIM	SIM	SIM	SIM	30.00
0726001'	SUCCESS		SIM	SIM	SIM	SIM	30.00
0932801'	SUCCESS		SIM	SIM	SIM	SIM	30.00
1045901'	SUCCESS		SIM	SIM	SIM	SIM	30.00
1048001'	SUCCESS		SIM	SIM	SIM	SIM	30.00
1054001'	SUCCESS		SIM	SIM	SIM	SIM	30.00
1054501'	SUCCESS		SIM	SIM	SIM	SIM	30.00

ANEXO D – Arquivo CSV de entrada para o teste de comparação de versão do software

Referência	Novo
C1_REF	C1
C2_REF	C2
C3_REF	C3
C4_REF	C4
C5_REF	C5

ANEXO E – Arquivo CSV de saída para o teste de comparação de versão do software do ANEXO D - Erro Relativo

Ref	File	fa	t	xn	typeChannels	nameChannels	remainingChannels	remainingChannelsName	remainingChannelsType	epochsTimes	epochsValues	EpochsValuesZerosInNoiseChannel	nameChannelsNoise	PCP	powerSignal	FM	F_corr	COR_PairsOfElectrodes	COERENCIA	TimeExecution
C1_REF	C1	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM -[erros aceitavel (qtd=9268 erro=1.4099e-12)]	SIM -[erros aceitavel (qtd=1609 erro=3.9442e-14)]	'0.38902'
C2_REF	C2	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM -[erros aceitavel (qtd=8154 erro=3.9553e-12)]	SIM -[erros aceitavel (qtd=1426 erro=4.2079e-14)]	'1.8591'
C3_REF	C3	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM -[erros aceitavel (qtd=3935 erro=9.0001e-13)]	SIM -[erros aceitavel (qtd=1064 erro=9.4774e-14)]	'1.4102'
C4_REF	C4	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM -[erros aceitavel (qtd=9793 erro=9.4818e-13)]	SIM -[erros aceitavel (qtd=2674 erro=2.1214e-13)]	'0.8746'
C5_REF	C5	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM -[erros aceitavel (qtd=3562 erro=4.3362e-14)]	SIM -[erros aceitavel (qtd=999 erro=4.7907e-15)]	'1.2036'

ANEXO F – Arquivo CSV de saída para o teste de comparação de versão do software do ANEXO D – Friedman

Ref	File	fa	Epochs Times	epochs Values	Epochs Values Zeros In Noise Channel	PCP	Power Signal	FM	F_cor	COR_PairsOfElectrodes	COERENCIA	t	xn	remaining Channels	timeExecution
C1_REF	C1	SIM()	SIM()	SIM()	SIM ()	SIM()	SIM()	SIM()	SIM ()	NAO [minPreturn=0.015777]	NAO (band=1{NAO [minPreturn=0.0455]}||band=2{NAO [minPreturn=0.014306]}||band=3{NAO [minPreturn=0.011412]}||band=4{NAO [minPreturn=0.011412]}||)				'86.8449'
C2_REF	C2	SIM()	SIM()	SIM()	SIM ()	SIM()	SIM()	SIM()	SIM ()	NAO [minPreturn=0.015777]	NAO (band=1{NAO [minPreturn=0.014306]}||band=2{NAO [minPreturn=0.0046777]}||band=3{SIM [P aceitavel (qtd=24 minPreturn=0.05778)]}||band=4{NAO [minPreturn=0.019631]}||)				'10.9019'
C3_REF	C3	SIM()	SIM()	SIM()	SIM ()	SIM()	SIM()	SIM()	SIM ()	SIM [P aceitavel (qtd=32 minPreturn=0.058291)]	NAO (band=1{NAO [minPreturn=0.0455]}||band=2{NAO [minPreturn=0.0455]}||band=3{NAO [minPreturn=0.0455]}||band=4{NAO [minPreturn=0.0455]}||)				'3.9163'
C4_REF	C4	SIM()	SIM()	SIM()	SIM ()	SIM()	SIM()	SIM()	SIM ()	NAO [minPreturn=0.0055589]	NAO (band=1{SIM [P aceitavel (qtd=21 minPreturn=0.095581)]}||band=2{NAO [minPreturn=0.0046777]}||band=3{NAO [minPreturn=0.019631]}||band=4{NAO [minPreturn=0.0015654]}||)				'8.0407'
C5_REF	C5	SIM()	SIM()	SIM()	SIM ()	SIM()	SIM()	SIM()	SIM ()	NAO [minPreturn=0.035939]	NAO (band=1{NAO [minPreturn=0.0455]}||band=2{NAO [minPreturn=0.0455]}||band=3{NAO [minPreturn=0.0455]}||band=4{NAO [minPreturn=0.0455]}||)				'3.988'

ANEXO G – Guia de Navegação na Plataforma de Cadastro desenvolvida

Para conectar-se à aplicação deve-se acessar o seguinte link:

https://ufu-ppgeb-eeg.herokuapp.com/

A primeira tela que será exibida é um popUp solicitando o usuário e senha para entrar na aplicação (Ilustração 1).

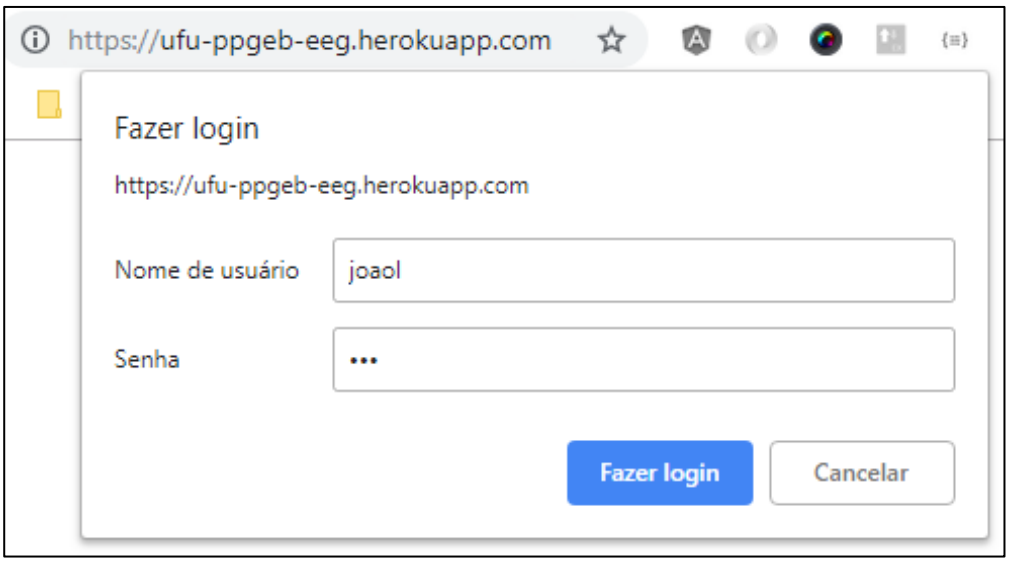


Ilustração 1 - Tela de login

Após o usuário digitar o usuário e senha válido ele será direcionado para a tela inicial da aplicação que é um Dashboard que ainda está em elaboração (Ilustração 2).

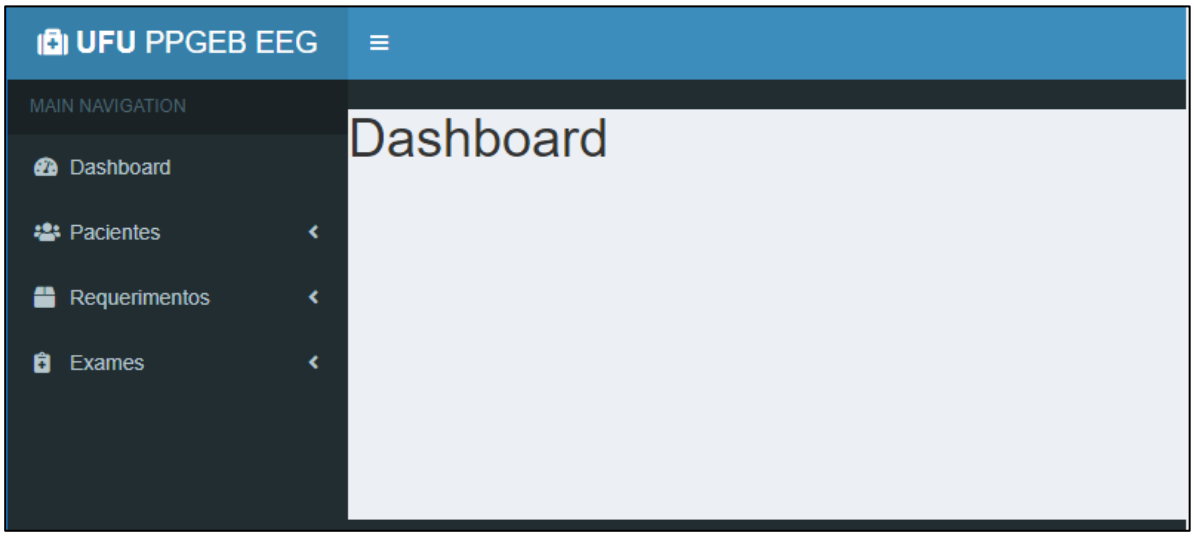


Ilustração 2 - Tela inicial da aplicação Dashboard

Nesta tela, na extremidade lateral à esquerda temos os menus de navegação da aplicação, que são:

- Pacientes
  - Cadastro (Ilustração 3 e Ilustração 4)
  - Busca (Ilustração 6 e Ilustração 7)
- Requerimentos
  - Busca (Ilustração 13)
- Exames
  - Busca (Ilustração 10)

Na tela de cadastro de Paciente (Ilustração 3 e Ilustração 4) o usuário deve preencher todos os campos obrigatórios e clicar no botão adicionar. Caso alguns dos campos obrigatórios não estejam preenchidos ele ficará marcado em vermelho, conforme Ilustração 3 onde os campos CPF e Sexo não foram preenchidos. Caso o cadastro seja realizado com sucesso, no canto superior à esquerda irá aparecer uma tarja verde informado que o paciente foi cadastrado com sucesso e o usuário é, então, redirecionado automaticamente para a tela de edição deste paciente (Ilustração 5).

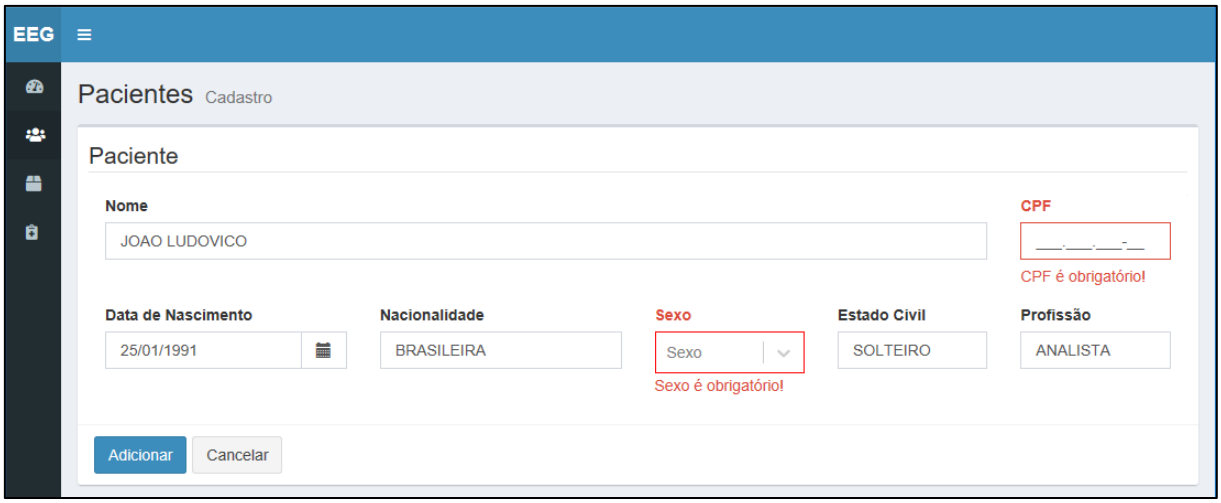


Ilustração 3 - Tela de cadastro de pacientes (versão desktop)

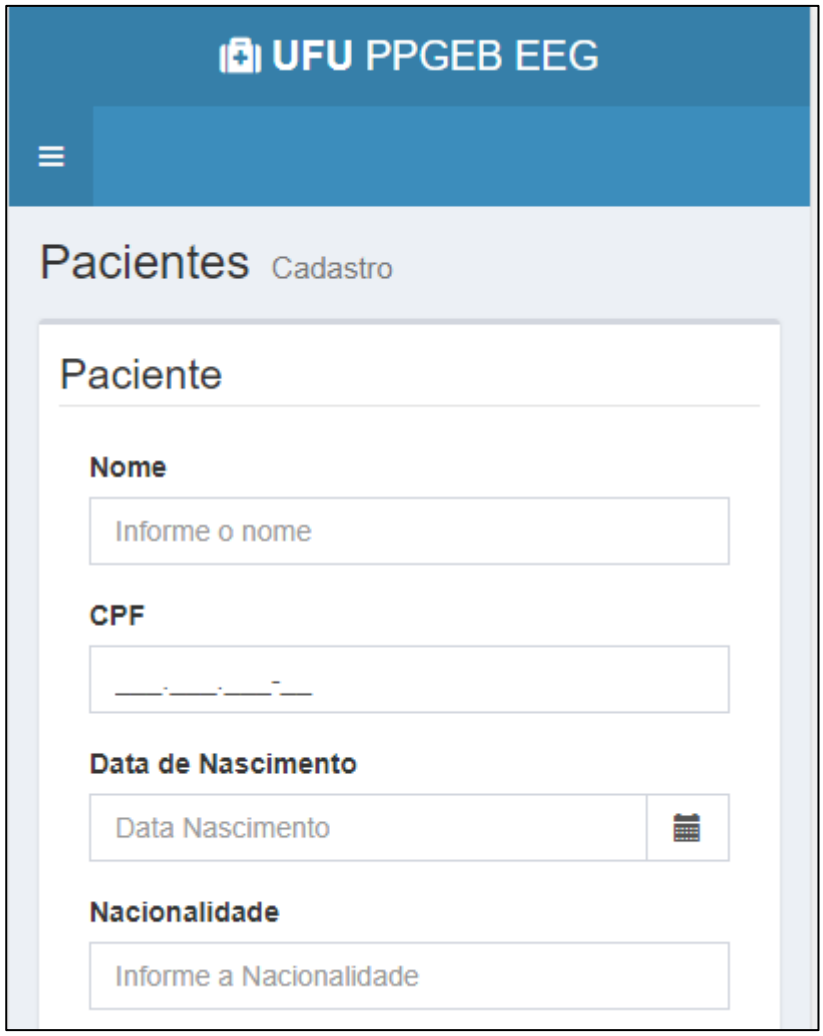


Ilustração 4 - Tela de cadastro de pacientes (versão mobile)

A tela de edição de dados do paciente (Ilustração 5) é praticamente idêntica à tela de cadastro, com a adição apenas dos campos de criação e atualização, bem como de duas abas na parte inferior da janela, abas estas que são os exames de EEG relacionados ao paciente, assim como a aba de requerimento de exames para o mesmo. Em cada aba existe uma tabela exibindo os exames/requerimento, na qual, se o usuário clicar em um dos itens, é redirecionado para a tela de edição do Exame ou Requerimento, assim como o botão “Adicionar” na cor verde, que redireciona o usuário para a tela de cadastro de Exame ou Requerimento.

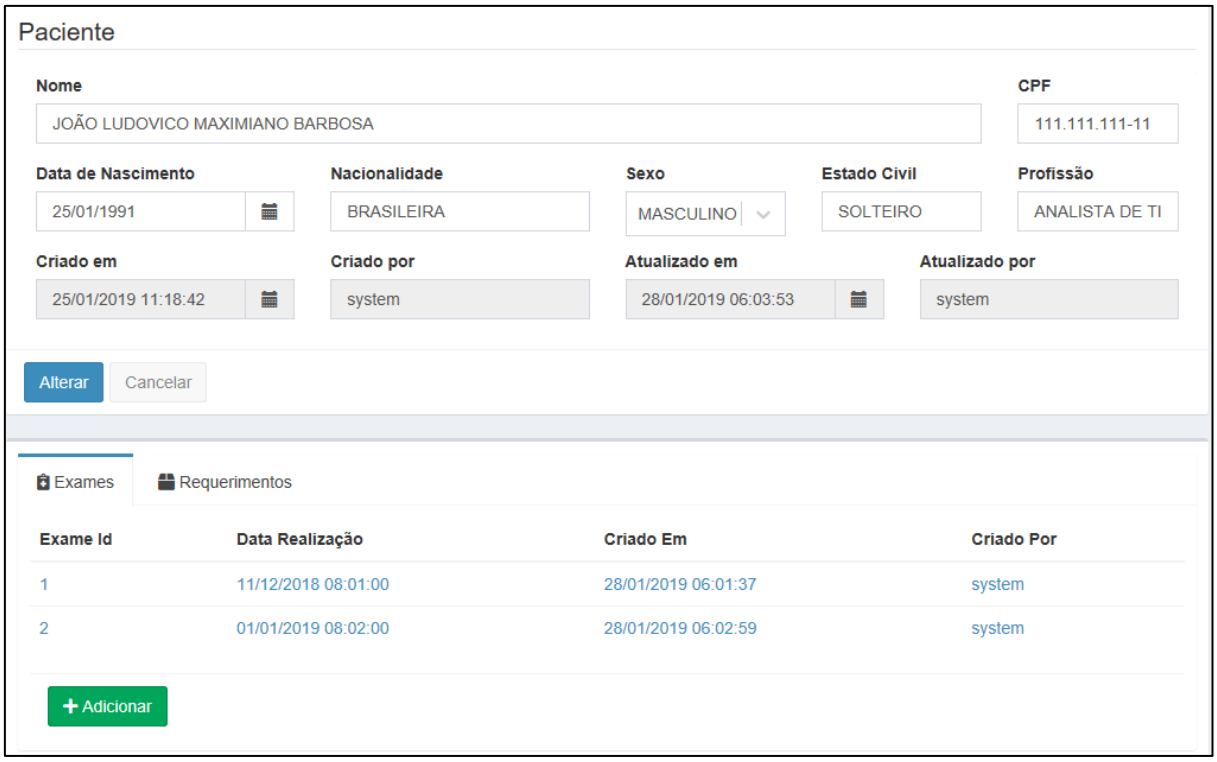


Ilustração 5 -Tela de Edição de dados do Pacientes

Para a tela de busca de pacientes (Ilustração 6**Erro! Fonte de referência não encontrada.** e Ilustração 7**Erro! Fonte de referência não encontrada.**) foram criados dois campos de consulta, que são: o Nome e o CPF do paciente. O usuário deve preencher pelo menos um dos campos para consultar, caso contrário o sistema irá marcar os campos em vermelho e informar que pelo menos um dos campos deve ser preenchido. Caso os parâmetros informados para a consulta retornem algum resultado, eles serão mostrados logo abaixo aos campos da consulta em uma tabela e, caso o usuário clique em um destes itens da tabela, ele será redirecionado para a tela de edição do paciente em questão.

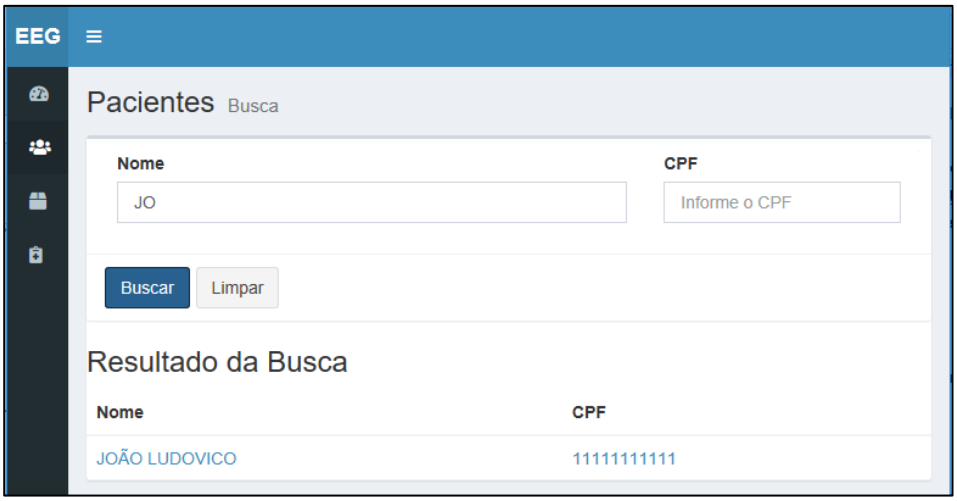


Ilustração 6 - Tela de consulta de pacientes (versão desktop)

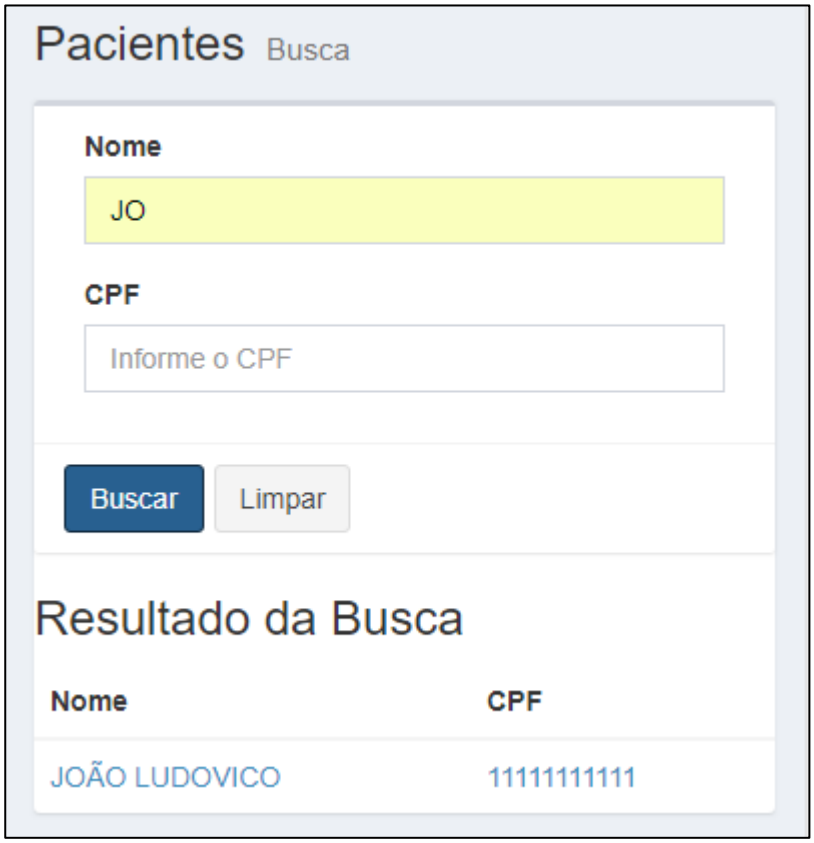


Ilustração 7 - Tela de consulta de pacientes (versão mobile)

Para o cadastro de um novo exame o usuário primeiramente deve consultar um paciente e, logo em sequência, entrar em sua tela de edição (Ilustração 5), ir na aba “Exames” e clicar no botão “Adicionar”. Dessa forma, ele será redirecionado para a tela de cadastro de Exames (Ilustração 8) onde, na parte superior da tela, serão

exibidas as informações do paciente em questão e, logo abaixo, os campos relacionados ao cadastro do exame, nos quais o usuário deve preencher, pelo menos, o campo de data da realização. Após preencher os campos o usuário deve clicar no botão adicionar; caso o cadastro seja realizado com sucesso, no canto superior direito irá aparecer uma tarja na cor verde informando que o cadastro foi realizado com sucesso e o usuário será redirecionado para a tela de edição de exame; caso ocorra um erro ao salvar o exame será exibido uma tarja em vermelho, no canto superior direito da tela, informando que ocorreu um erro.

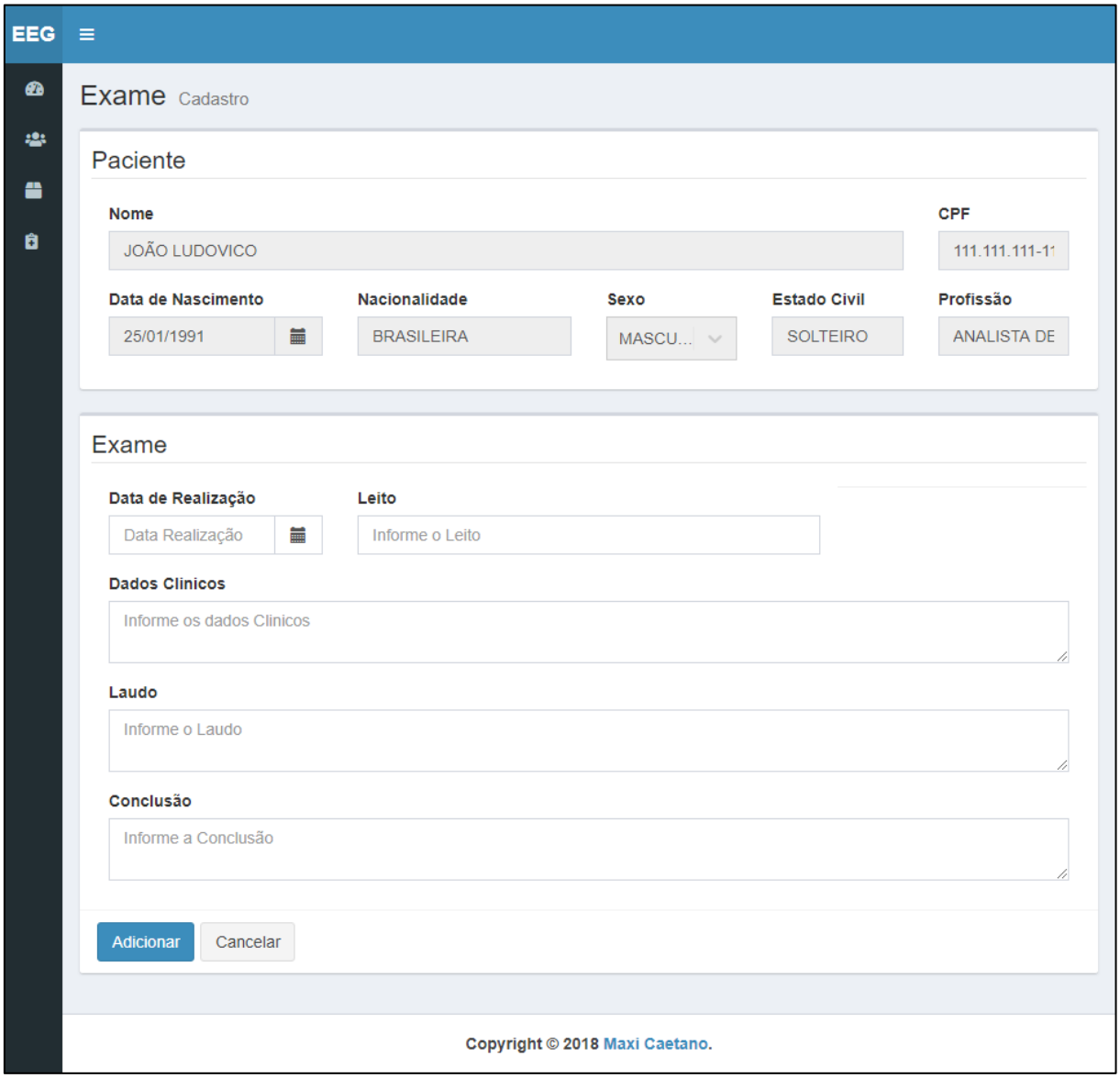


Ilustração 8 - Tela de Cadastro de Exames

A tela de edição de Exame é idêntica à tela de cadastro, com a adição dos campos de criação e atualização, mais a adição de 4 abas (Ilustração 9): Medicamentos, Equipamentos, Épocas e Requerimentos. A aba de Medicamentos e Equipamentos são idênticas, nas quais o usuário pode selecionar vários medicamentos/equipamentos e, caso o medicamento/equipamento ainda não esteja cadastrado, o usuário deve selecionar a opção “outro”, conforme Ilustração 9, preencher o campo Nome e descrição. Assim, quando o backEnd receber esta requisição ele entenderá que deve adicioná-lo no banco de dados. Já a aba de Épocas é composta apenas de 4 campos: minutos, segundos, duração e descrição. Todos estes campos são obrigatórios. A aba Requerimentos só irá aparecer caso o exame esteja vinculado a um requerimento; nesta aba é exibida uma tabela com o requerimento vinculado, caso o usuário clique nesta linha de tabela ele será redirecionado para a tela de edição do requerimento.

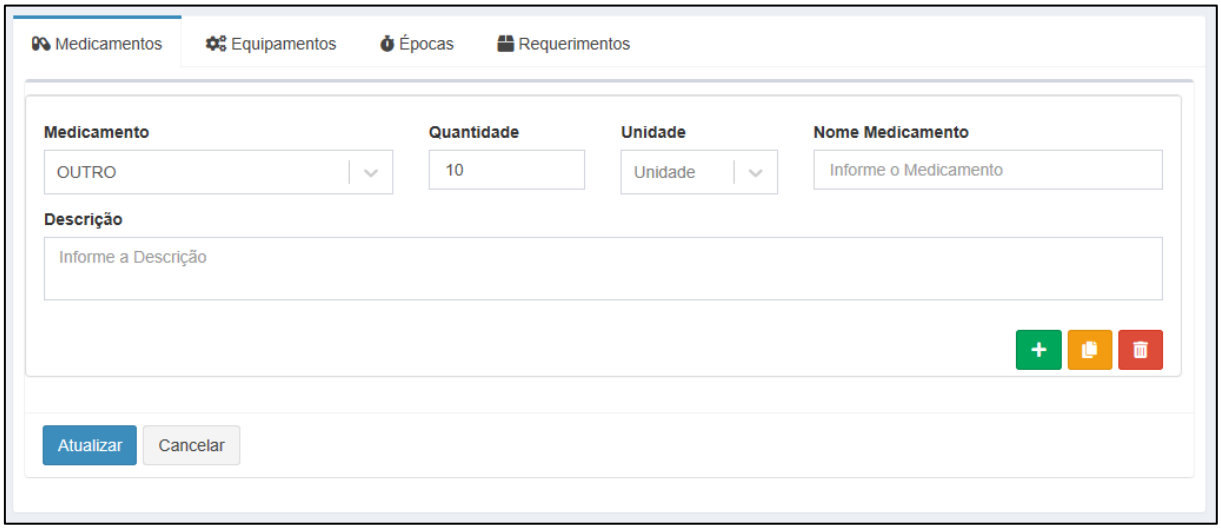


Ilustração 9 - Abas da tela de edição de exames

Existe também a tela de busca de exames (Ilustração 10) composta por dois campos: o ID do exame e o leito onde o exame foi realizado. O usuário deve preencher pelo menos um dos campos, caso contrário o sistema irá apresentar os campos em vermelho e irá informar que pelo menos um dos campos deve ser informado para realizar a consulta. O resultado da consulta é apresentado logo abaixo dos campos de busca, conforme pode ser observado na Ilustração 10. Se o usuário clicar em um dos itens do resultado da consulta ele é redirecionado para a tela de edição daquele exame. Caso os campos informados para a consulta não retornem nenhum resultado

irá ser apresentado no canto superior direito uma tarja na cor laranja informando que não foi encontrado nenhum resultado.

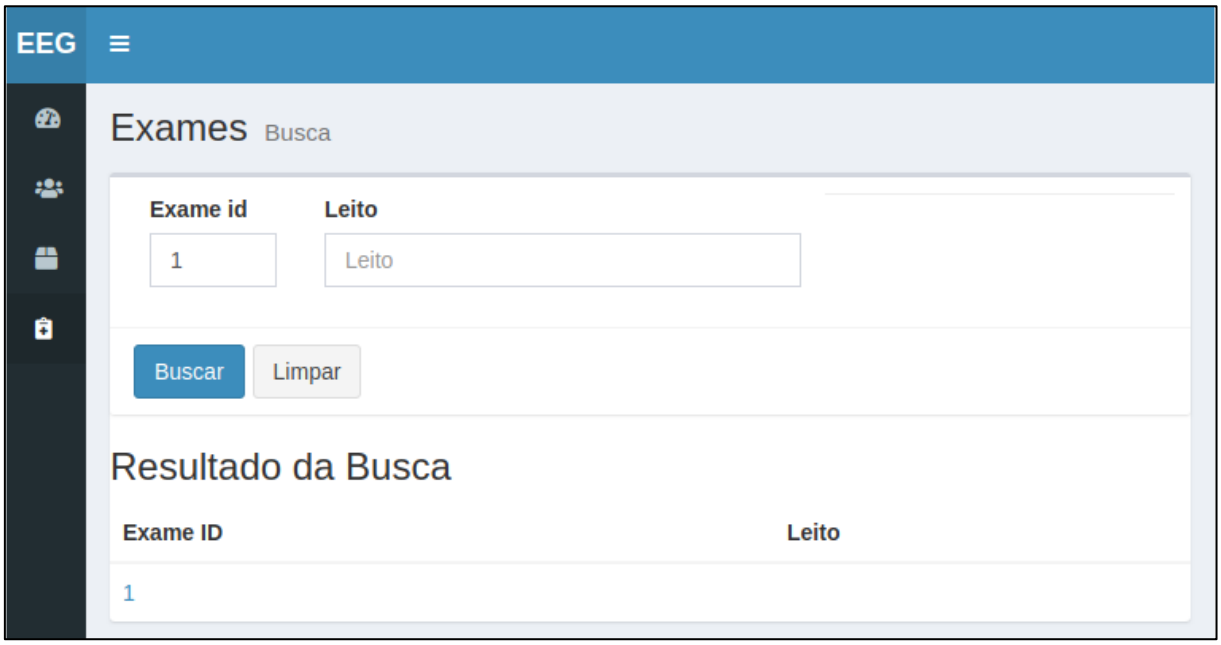


Ilustração 10 - Tela de consulta de exames

Para se cadastrar um novo requerimento o usuário, primeiramente, deve consultar um paciente e, logo em sequência, entrar em sua tela de edição (Ilustração 5) e ir na aba “Requerimentos”, clicando no botão “Adicionar”. Desta maneira, ele será redirecionado para a tela de cadastro de Requerimentos (Ilustração 11) onde, na parte superior da tela, serão exibidas as informações do paciente em questão e, logo abaixo, os campos relacionados ao cadastro do requerimento, nos quais o usuário deve preencher pelo menos os campos: Prontuário ID, Requisição ID, Data do Pedido, Setor, Medico Solicitante e Usuário. Após preencher os campos o usuário deve clicar no botão adicionar; caso o cadastro seja realizado com sucesso, no canto superior direito irá aparecer uma tarja na cor verde informando que o cadastro foi realizado com sucesso e o usuário será redirecionado para a tela de edição de requerimento; caso ocorra um erro ao salvar o requerimento será exibido uma tarja em vermelho no canto superior direito da tela informando que ocorreu um erro.

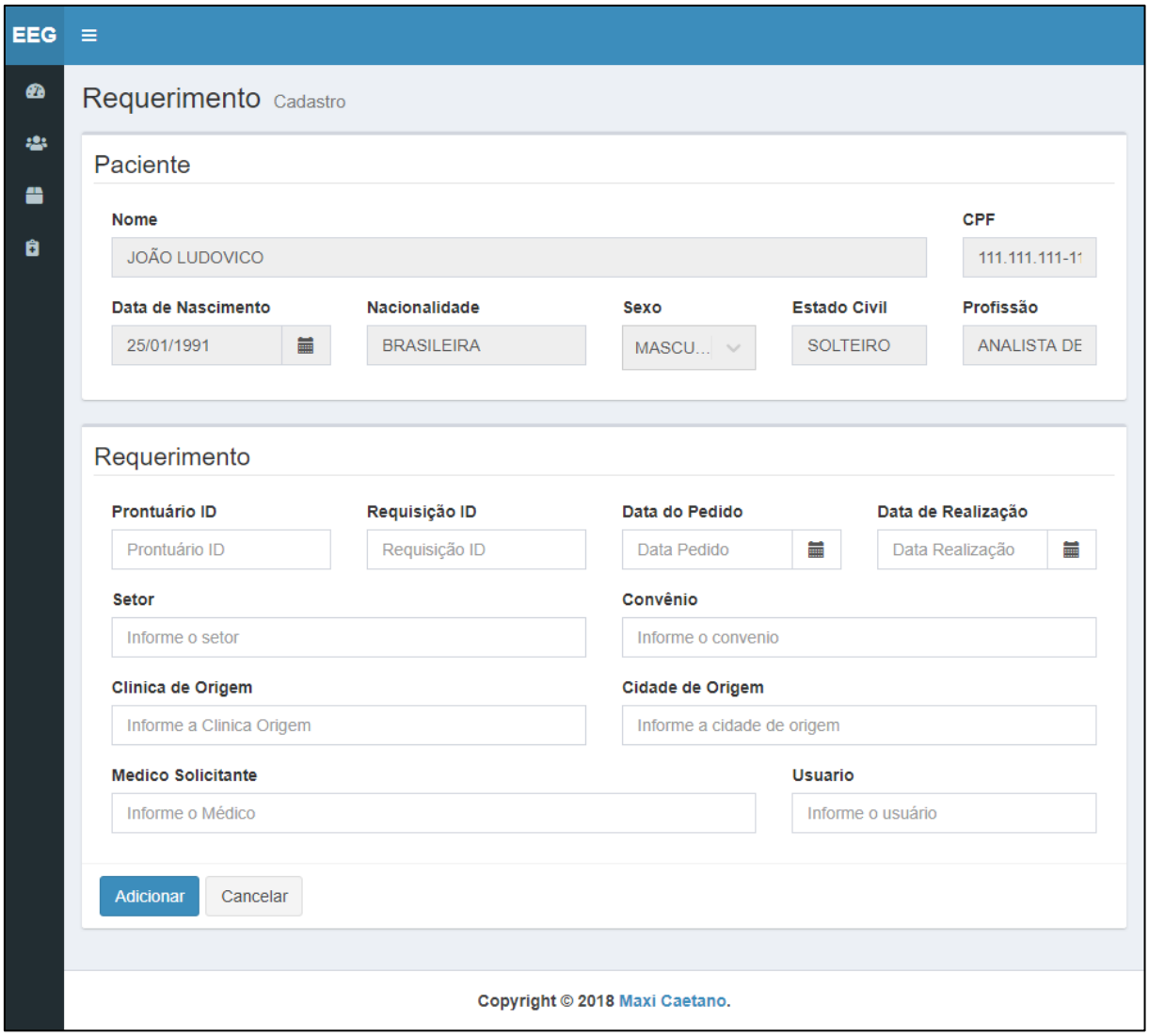


Ilustração 11 - Tela de Cadastro de requerimentos

A tela de edição de Requerimentos é idêntica à tela de cadastro, com a adição dos campos de criação, atualização, mais a adição da aba “Exames” (Ilustração 12). Esta aba possui o comportamento idêntico à aba “Exames” da edição de pacientes (Ilustração 5), nela é exibida uma tabela com os Exames relacionados ao requerimento em edição, na qual, se o usuário clicar em qualquer um dos itens da tabela, ele é redirecionado para a tela de edição do exame clicado e, se o usuário clicar no botão adicionar, é redirecionado para a tela de cadastro de exames (Ilustração 8).

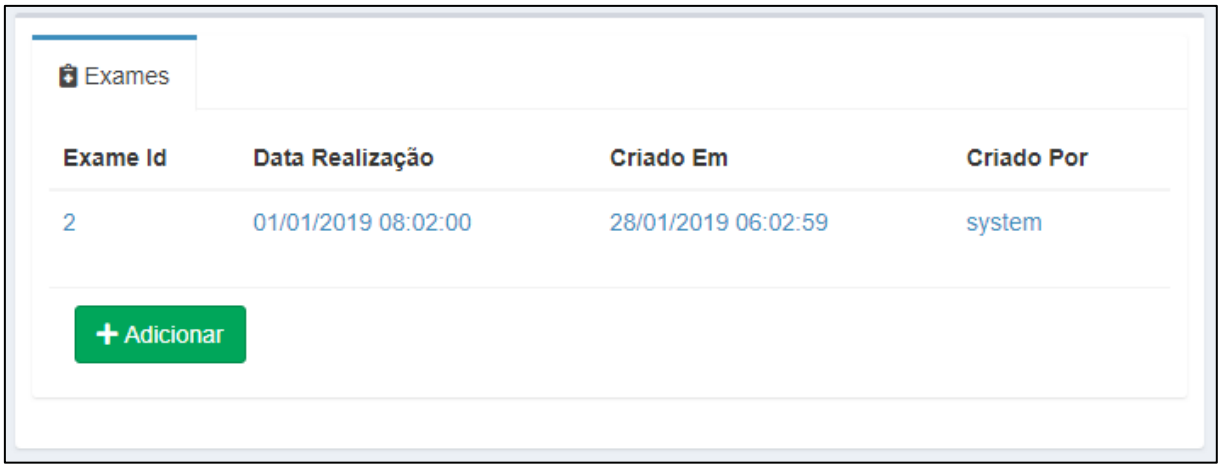


Ilustração 12 - Aba "Exames" da tela de edição de Requerimentos

Existe, também, a tela de busca de requerimentos (Ilustração 13) composta por três campos: Prontuário ID, Requisição ID e Medico Solicitante. O usuário deve preencher pelo menos um dos campos, caso contrário o sistema irá apresentar os campos em vermelho e irá informar que pelo menos um dos campos deve ser informado para realizar a consulta. O resultado da consulta é apresentado logo abaixo dos campos de busca, conforme pode ser observado na Ilustração 13. Se o usuário clicar em um dos itens do resultado da consulta ele será redirecionado para a tela de edição daquele requerimento. Caso os campos informados para a consulta não retornem nenhum resultado irá ser apresentado no canto superior direito uma tarja na cor laranja informando que não foi encontrado nenhum resultado.

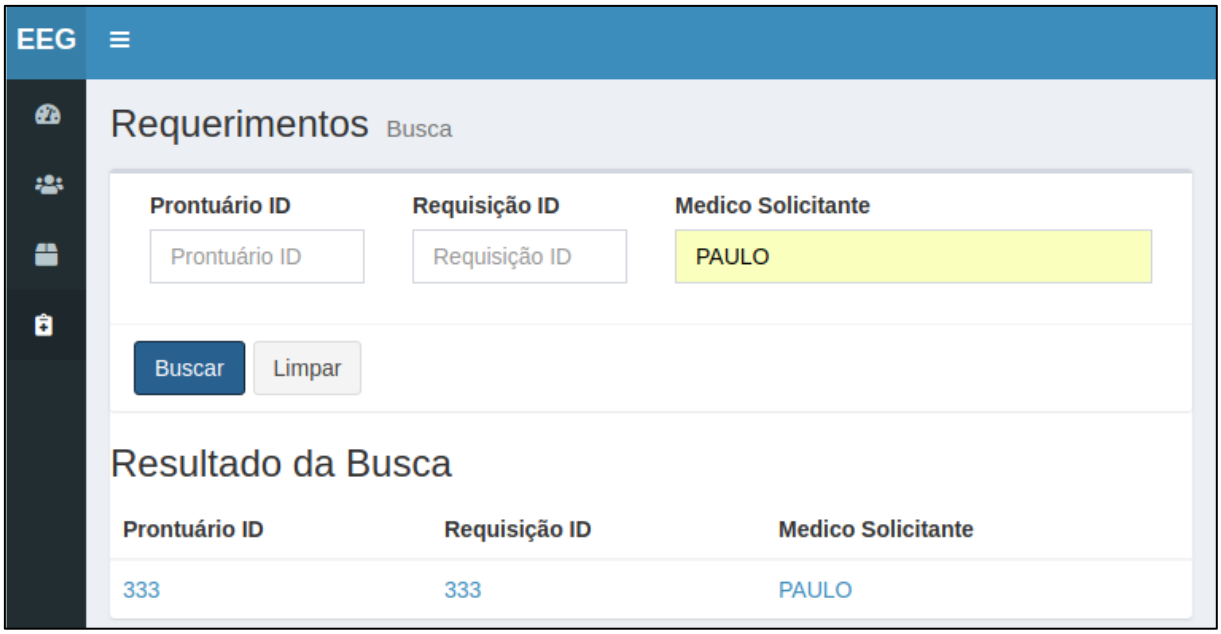


Ilustração 13 - Tela de consulta de Requerimentos