UNIVERSIDADE FEDERAL DO PARANÁ

ALEXANDRE DO AMARAL SEVERINO

ALYSSON LUÍS MARTINS NARLOCH

GUSTAVO MOLITOR PORCIDES

SPECTative 2.0: FERRAMENTA PARA LEITURA E VISUALIZAÇÃO DE IMAGENS CINTILOGRÁFICAS

CURITIBA

2012

ALEXANDRE SEVERINO

ALYSSON LUÍS MARTINS NARLOCH

GUSTAVO MOLITOR PORCIDES

SPECTative 2.0: FERRAMENTA PARA LEITURA E VISUALIZAÇÃO DE IMAGENS CINTILOGRÁFICAS

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) da Universidade Federal do Paraná como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas

Orientador: Prof. Dr. Lucas Ferrari de Oliveira

CURITIBA

2012

RESUMO

O SPECTative é uma ferramenta que foi desenvolvida para dar apoio ao médico na visualização de imagens geradas através de equipamentos de Medicina Nuclear (SPECT). Com o uso desta ferramenta open source, é possível analisar as imagens sem a necessidade de um equipamento específico. O software proposto permite que o usuário visualize o exame, no formato DICOM, obtido através do equipamento citado acima, como uma série de imagens 2D e realizar o alinhamento destas imagens, com a imagem modelo e exibir o mapa polar. Além disso, para melhorar a exibição das fatias nos três eixos (X, Y e Z), o software permite que o usuário aplique pseudo-cores, mude o contraste e modifique a fatia a ser mostrada, de forma dinâmica. Para gerir o projeto a equipe fez uso da metodologia ágil Scrum. Para o desenvolvimento do software foram utilizadas as bibliotecas ITK, VTK e QT programáveis na linguagem C++.

ABSTRACT

SPECTative is a tool developed to support doctors in the visualization of images generated by nuclear medicine equipment. With this open source tool, it will be possible to analyze and visualize the images without a special equipment. . The proposed software allows its users to visualize the exams, in DICOM format, obtained using the equipment cited above, as a series of 2D images e and perform its registration with the model image and display its polar map. Moreover, to improve the slices' display in the three axis (X, Y and Z), the software allows users to apply pseudo-colors, change the contrast and modify the slice that is being displayed in a dynamic way.For the project management the team used the agile methodology SCRUM.For the software development were used the C++ libraries ITK, VTK and QT.

LISTA DE ABREVIATURAS E SIGLAS

2D – Bidimensional

3D – Tridimensional

CMake – Sistema de gerenciamento do processo de compilação

CT (Computed Tomography) – Tomografia Computadorizada

DICOM (*Digital Imaging and Communications in Medicine)* – Protocolo de Imageamento Digital e Comunicações na área médica

IAM – Infarto Agudo do Miocárdio

ITK (*Insight Segmentation and Registration Toolkit)* – Conjunto de Ferramentas de Registro e Segmentação de imagens

JPEG (*Joint Photographic Experts Group*) – Método de compressão de imagens

MPEG-2 (*Moving Picture Experts Group*) – Métodos de compressão de áudio e vídeo

MRI (*Magnetic Resonance Imaging*) – Método de Ressonância Magnética

PET (*Positron Emission Tomography*) – Tomografia por Emissão de Positrons

QT – framework multiplataforma para o desenvolvimento em C++ que possibilita a compilação para diversas plataformas

RGB (*Red Green Blue*) – Modelo de cores baseado nas cores vermelha, verde e azul

SPECT (*Single Photon Emission Computed Tomography)* – Tomografia Computadorizada por Emissão de Fóton Único

UML (*Unified Modeling Language)* – Linguagem-padrão para a elaboração da estrutura de projetos de software

VTK (*VisualizationToolKit)* – Conjunto de Ferramentas para Visualização

SUMÁRIO

1 INTRODUÇÃO

1.1 INTRODUÇÃO

Segundo (SARMENTO-LEITE et al, 2001) doenças do sistema circulatório são uma das principais causas de morte em países desenvolvidos. Por exemplo, o Infarto Agudo do Miocárdio (IAM) leva aproximadamente um milhão de pessoas a óbito anualmente nos Estados Unidos. Na Inglaterra, a incidência é de 2,6 por mil habitantes ao ano (MELO et AL, 2004)

Geralmente, os pacientes que sobrevivem a um infarto necessitam de acompanhamento para avaliar o risco de reincidência (MULTICENTER, 1983). Alguns dos métodos utilizados na detecção e acompanhamento desses pacientes fazem parte de uma especialidade médica conhecida como medicina nuclear.

Exames como cintilografia, SPECT e CT fazem uso de radiação para obter as imagens. A radiação pode ser inserida no paciente, intravenosamente ou oralmente, ou aplicada no paciente pelo equipamento. Nas duas modalidades é necessário o uso de detectores externos para a obtenção das imagens.

Este trabalho foca-se nas imagens de SPECT de perfusão do miocárdio. Esse exame é bastante usado para predizer o estado do coração de pacientes pós-infarto. As imagens obtidas permitem ao médico avaliar a necessidade de cirurgia ou tratamento, pois diferencia as regiões do coração que tem tecido vivo das que estão mortas (PATZER, 2011).

As imagens fornecidas pelo SPECT são imagens volumétricas, sendo, diferentemente de imagens comuns, representadas em três dimensões. Para a visualização, a imagem é dividida em diversas fatias, nos três eixos, onde cada uma das fatias é um deslocamento no seu respectivo eixo. Ela é uma reprodução do fluxo sanguíneo na parede do miocárdio (BARROS, 2007).

Essas imagens são salvas no formato DICOM, que é um padrão de tratamento e armazenamento de dados médicos (MILDENBERGER et al, 2002). Além da imagem, o formato DICOM guarda, no mesmo arquivo, dados sobre o paciente, exame e equipamento utilizado como um cabeçalho.

Para a leitura e manipulação dessas imagens são necessários softwares específicos, uma vez que os visualizadores devem seguir esse padrão.

Além disso, para a correta visualização da imagem é necessária que as estruturas estejam nos lugares corretos. Esse processo, conhecido como alinhamento, geralmente é feito manualmente por um especialista (PATZER, 2011).

Além disso, existem diversas técnicas que modificam o modo como as imagens são exibidas. Uma das mais conhecidas é o mapa polar (GARCIA et al, 1985). Esta técnica transforma as fatias 3D em uma imagem bidimensional circular, facilitando a visualização do dano causado pelo infarto. Outra técnica bastante utilizada é a pseudo-cor que, levando em conta que a visão humana é menos sensível a variação de tons de cinza do que de cores, aplica cores nos pixels da imagem, que originalmente são em escala de cinza (LIUMING et al, 2004).

Isso motivou o desenvolvimento de um software que além de permitir a visualização das imagens, fizesse o alinhamento automático e possibilitasse a geração de mapas polares e aplicação de pseudo-cores.

1.1 OBJETIVOS DO PROJETO

Este projeto tem o objetivo de detalhar todas as etapas do desenvolvimento de um software de visualização e manipulação de imagens DICOM, desde a escolha das ferramentas e levantamento de requisitos até o desenvolvimento da aplicação.

O objetivo do SPECTative é permitir a visualização e manipulação de imagens de SPECT de perfusão do miocárdio no formato DICOM.

A ferramenta possui os seguintes objetivos específicos:

- Abrir imagens tridimensionais no formato DICOM.

- Desenvolver um visualizador que permita a exibição de até dois arquivos diferentes simultaneamente, mostrando os seus três eixos.

- Permitir a aplicação de pseudo-cores distintas para as imagens.

- Realizar o alinhamento automático de até duas imagens simultaneamente com uma imagem modelo previamente alinhada por um especialista, possibilitando que o usuário salve essas imagens em um novo arquivo.

- Controles para a alteração do contraste das imagens na fase de visualização e alinhamento.

- Geração de mapa polar de até duas imagens com a exibição da subtração das mesmas.

2 FUNDAMENTAÇÃO TEÓRICA

Para a elaboração do software proposto foi necessário o uso de tecnologias e ferramentas, as quais serão descritas nesta seção.

2.1 Unified Modeling Language

De acordo com Grady Booch et al. (2005), a UML (Unified Modeling Language) é uma linguagem-padrão que surgiu da necessidade de unificar a forma de representar o sistema conceitual e fisicamente. Sendo uma linguagem, a UML fornece um vocabulário e um conjunto de regras para expressar o comportamento

dos diferentes aspectos do sistema. Com ela, é possível modelar o sistema de modo a abstrair e exemplificar o seu funcionamento, especificá-lo, usá-lo como guia para seu desenvolvimento e documentar seus componentes. Com isso, esta linguagem é uma ferramenta importante na elaboração de projetos de software.

Para tornar a modelagem possível, a UML dispõe de uma série de diagramas que permitem mostrar e descrever o sistema a nível de especificação. Neste trabalho serão utilizados os seguintes diagramas:

- Diagrama de classes: faz a representação estática do sistema com a apresentação das classes, seus respectivos relacionamentos, interfaces e colaboração. Além disso, o diagrama de classes abstrai o comportamento das funcionalidades do sistema (Booch et al., 2005).
- Diagrama de casos de uso: este diagrama é responsável por demonstrar o comportamento que deseja-se que o sistema tenha, sem descrever como será a implementação. Para isso, o caso de uso faz a representação de atores (pessoas ou sistemas automatizados) interagindo com as funcionalidades do sistema (casos de uso). Esta ferramenta é muito utilizada para validar o projeto do sistema junto ao usuário (Booch et al., 2005).
- Diagrama de sequência: este diagrama mostra a ordem sequêncial da transmissão das mensagens e os fluxos presentes no sistema (Booch et al., 2005).

2.2 Linguagem C e C++

Segundo Schildt (1998), C é uma linguagem de programação estruturada de nível médio, pois possui os recursos de linguagens de alto nível, como Pascal ou Basic, e ao mesmo tempo dispõe do controle e flexibilidade presentes na linguagem assembly. C é amplamente compatível com os diferentes sistemas operacionais disponíveis e por ter a característica de ser de nível médio dá ao programador o suporte a manipulação de bits, bytes e endereços. De acordo com Schildt (1998), C foi feita para desenvolvedores, o que implica que sua arquitetura foi projetada para dar total liberdade na programação, poucos incômodos, estruturas em blocos e um conjunto compacto de palavras. Já o C++ é uma linguagem orientada a objetos que

contém todos os recursos de C e se caracteriza pelo bom desempenho de suas aplicações (Davis, 2004). Além disso, o C++ é uma das linguagens mais conhecidas e muitas das aplicações que estão sendo executadas nos computadores foram desenvolvidas em C++.

2.3 O FORMATO DICOM

Com o passar dos anos, a computação se tornou relevante na vida da população, com suas inovações. Na área médica a situação não foi diferente. Com toda essa inovação tecnológica a medicina passou a fazer uso de processos automatizados, principalmente na geração de imagens e modelos 3D, a partir de dispositivos sofisticados como CT (Computed Tomography), MRI (Magnetic Resonance Imaging), SPECT (Single Photon Emission Computed Tomography) e PET (Positron Emission Tomography) (MUSTRA et al, 2008). Com o aumento da geração de imagens médicas, surgiu a necessidade de padronizar seu formato. Para isso, grupos de diferentes áreas da medicina criaram o padrão DICOM (Digital Imaging and Communication in Medicine), cada um com um foco diferente, para viabilizar a compatibilidade das imagens entre diferentes equipamentos e facilitar a transferência.

Além da imagem propriamente dita, segundo Mario Mustra et al (2008), o conteúdo do arquivo DICOM é composto por um cabeçalho, com informações a respeito do equipamento em que a imagem foi gerada, dados do paciente, nome da instituição, entre outras. Além disto, o arquivo tem suporte a imagens com múltiplas dimensões, JPEG e até mesmo vídeo MPEG-2. Dessa forma, com o padrão DICOM é possível acessar imagens de exames em qualquer computador convencional e permite que o paciente seja transferido para outro hospital sem perda do exame realizado anteriormente.

No presente trabalho estão sendo usadas imagens DICOM de exames de SPECT.

2.4 QT

O QT é uma tecnologia desenvolvida pela empresa Nokia com o simples intuito de prover uma plataforma livre para a criação de interfaces gráficas (GUI). A idéia principal deste framework é "*write once, compile anywhere*", ou seja, escreva seu código uma única vez e compile em qualquer ambiente. Assim, é possível criar aplicações para ambientes Windows, Mac OS, Linux, Solaris e HP (BLANCHETTE et al, 2008).

2.5 *Visualization Toolkit*

VTK (*Visualization Toolkit*) é uma biblioteca implementada em C++, orientada a objetos, para computação gráfica, visualização e processamento de imagens. Além disso, o VTK é *open source* e portável, tendo compatibilidade com Windows, Linux e Mac OS e da suporte as linguagens Tcl/Tk, Python e Java (PERINI et al, 2009). De acordo com OLIVEIRA et al. (2007), o VTK dispõe diversos algoritmos de visualização podendo ser encontrados métodos escalares, vetorial e volumétrico. O que tornou o VTK uma tecnologia com um desenvolvimento mais acessível é o fato de sua arquitetura não prover uma comunicação tão próxima com o hardware, como ocorre com o OpenGL e o DirectX. Este maior nível de abstração permite que pessoas de áreas não correlatas com a computação desfrutem dos recursos de renderização que o VTK oferece.

2.6 Insight Segmentation and Registration Toolkit

O ITK (*Insight Segmentation and Registration Toolkit*) é uma biblioteca *open source* implementada em C++, orientada a objetos, focada no alinhamento e segmentação de imagens médicas. Apesar de ser voltada para aplicações médicas, a tecnologia pode ser usada para a resolução dos mais diversos problemas (OLIVEIRA et al, 2007). O ITK se destaca por adotar a programação genérica em

suas funcionalidades, o que torna a programação mais flexível. Além disso, YOO et al. (2010) ressalta que a programação genérica possibilita uma maior velocidade sobre a evolução da biblioteca por parte da comunidade que a mantém, tornando o ITK único em seu segmento.

Além de suportar os tipos nativos (long, unsigned int, float, etc) o ITK é multi-componentes (intensidade, intensidade/α, RGB, RGB/α, etc) e multidimensional (x-y-z, x-y-z-time, x-y-z-scale, etc) (YOO, 2010). Como o ITK não possui embutido um modelo de visualização de imagens, se faz necessário o uso de outra tecnologia, como o VTK, por exemplo, para exibir o resultado do processamento realizado (Oliveira et al, 2007).

2.7 Cmake

O CMake é um sistema de código aberto que permite ao usuário gerenciar todo o procedimento de compilação do sistema. Toda a configuração de chamada as bibliotecas e arquivos necessários para a geração do executável é definida pelo usuário no arquivo chamado CMakeLists.txt. A partir disto é que são gerados os arquivos padrão de compilação. Ao executar o CMake, todas as diretivas passadas no arquivo CMakeLists são localizadas e armazenadas no cache do sistema e são apresentadas ao usuário as opções a serem selecionadas para a compilação. O CMake tem a característica de trabalhar bem com estruturas complexas de diretórios e dar suporte a múltiplas bibliotecas dentro de um mesmo arquivo de configuração. Além disso, com ele é possível compilar dinâmica ou estaticamente (CMake, 2012).

3 METODOLOGIA

3.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

Todo o desenvolvimento desse trabalho foi guiado pela metodologia ágil Scrum, a qual descreve uma série de atividades a serem desempenhadas pela

equipe para o cumprimento das metas. Como a equipe buscava uma metodologia dinâmica e com uma abordagem incremental para gerir as atividades do projeto, o Scrum mostrou-se a opção mais adequada.

3.1.1 METODOLOGIA ÁGIL – SCRUM

Segundo Martins (2007), o nome Scrum veio das partidas de Rugby, em que os jogadores se reuniam em círculos para definir a próxima jogada. Da mesma forma, em projetos de TI, as atividades devem ser planejadas e executadas em pequenos marcos, com o objetivo de compor uma solução completa ao final do processo de desenvolvimento.

A metodologia chamou atenção pelo fato de não estar presa a um modelo “fechado”, o qual determina exatamente como as fases serão aplicadas, diferenciando-se das técnicas tradicionais. Entretanto, é uma abordagem com foco em mudanças e na criação de um produto com valor agregado, fazendo uso de práticas incrementais e iterativas.

3.1.1.1 O CICLO DO PROCESSO

Martins (2007) define o ciclo básico do Scrum possuindo a definição da visão, do Backlog de Produto, Backlog do Sprint e execução do Sprint, como pode ser observado na figura abaixo:

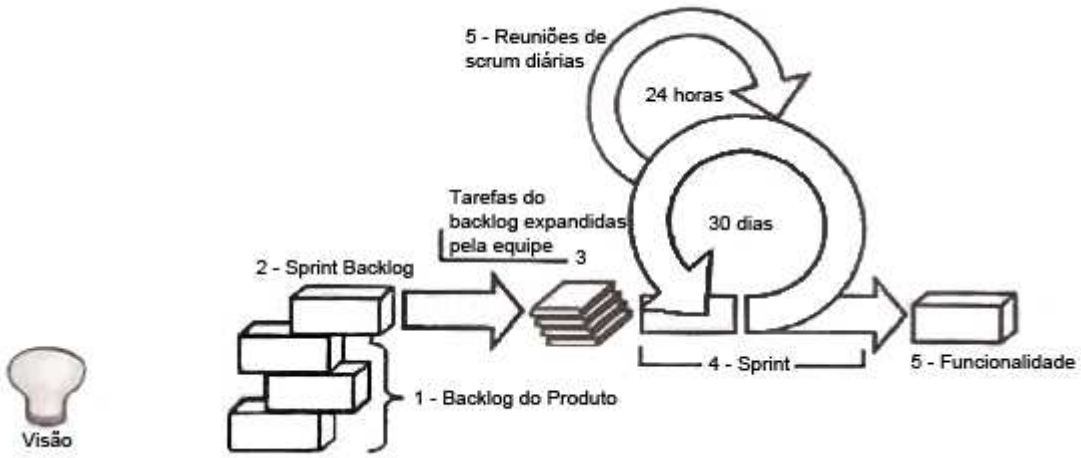


Figura 1 - O ciclo do processo ágil SCRUM

A primeira atividade do ciclo do projeto é conhecida como visão e é caracterizada pela definição dos principais elementos, técnicos ou de negócio, que deverão compor o produto final e ela faz a demonstração do seu resultado desejado. Dado o entendimento inicial do sistema, é criada uma lista, chamada Backlog de Produto, para descrever e priorizar as funcionalidades que o futuro produto irá conter. Em seguida, os itens levantados passam por uma avaliação, para selecionar quais a equipe é capaz de entregar, dentro do prazo estipulado, e assim elaborar o Backlog do Sprint. Como já mencionado anteriormente, o Scrum segue uma abordagem iterativa, a qual o sistema é elaborado em pequenas partes. Cada fase (iteração) é chamada de Sprint. O Backlog do Sprint define as etapas de desenvolvimento do sistema, com o conjunto de funcionalidades que cada módulo conterá. Ao final do fechamento de cada Sprint é criado um sub-sistema funcional e será apresentado aos clientes e demais stackholders para serem feitas sugestões e solicitarem adaptações. Com isso, o Backlog de Produto e do Sprint são modificados, atendendo assim as novas demandas. Ao final do processo do Scrum a equipe tem um software completo para oferecer ao cliente e o projeto é encerrado.

3.1.1.2 PAPÉIS

O Scrum possui uma hierarquia de responsabilidades, sendo composta pelo dono do produto, o Mestre Scrum e a equipe do projeto (Martins, 2007), como serão descritas a seguir.

O dono do produto é a pessoa que faz frente ao projeto defendendo o propósito de cada um que acredita em seu resultado. Além disso, é ele quem busca recursos financeiros, estabelece os requisitos, determina os objetivos e planeja as entregas.

Já o Mestre Scrum é reconhecido como um líder e é responsável por prover e disseminar as práticas estabelecidas pela metodologia, sendo um facilitador no cumprimento das metas e garante que os envolvidos no projeto façam uso correto do Scrum. Assim, suas principais funções são as de facilitar a comunicação entre equipe de desenvolvimento e Dono do Produto, estimular a criatividade da equipe e sempre deixar todos os envolvidos no projeto informados a respeito do desempenho da equipe.

Diferente dos dois casos anteriores, a equipe do projeto tem a missão de transformar os requisitos funcionais e não funcionais em um software. As equipes são pró-ativas e independentes. Elas não se limitam apenas às pessoas da área de desenvolvimento, permitindo também a participação de integrantes externos, como o cliente, por exemplo. Na maioria dos casos elas são compostas por 5 a 10 integrantes e cada um possuindo uma habilidade diferente.

3.1.1.3 FASES

De acordo com Martins (2007), o processo completo do Scrum é definido pelas fases de pré-game, game e pós-game.

O pré-game é composto pela etapa de planejamento e arquitetura. É na fase de planejamento que a equipe faz a análise, fazendo o levantamento das

informações a respeito do novo produto, elaboração de estimativas de custo, prazo, definição de papéis e infra-estrutura, prioridade de atividades, análise de risco e aprovação do projeto pelo responsável. Com isso, o planejamento auxilia os stackholders para que se adaptem ao novo ambiente e fornece um meio de assegurar o compromisso entre desenvolvedores e clientes, investidores, etc. Além disso, o planejamento será usado durante toda finalização de iteração como forma de parâmetro de avaliação do que foi entregue.

A arquitetura consiste em promover uma análise de como cada item irá se comportar ao ser finalizado no desenvolvimento e se ele terá um impacto positivo para o usuário. Além disso, a arquitetura serve para melhorar os itens que foram classificados como ineficientes e será proposta uma nova abordagem.

Após a definição do sistema é iniciada a fase “game” sendo um processo de execução das atividades das Sprints elaboradas no pré-game. Esta fase é guiada por reuniões diárias “em pé”, geralmente em torno de 15 minutos, para garantir o correto cumprimento dos requisitos e padrões estabelecidos e o game é encerrado ao final da conclusão da última Sprint.

Para finalizar o projeto, há o pós-game, que valida o software de acordo com a definição de prazo, custo, qualidade e concorrência de mercado. Ao ser aprovado, o produto entra em concepção para entrega. Nesta etapa é elaborada a integração, conjunto de testes, documentação, material de treinamento e de marketing, dentre outros.

3.2 PLANO DE ATIVIDADE

Como já abordado nas seções anteriores, este trabalho visa o desenvolvimento de uma ferramenta de visualização de imagens de cintilografia para uso médico. Para que isto seja possível é necessário a elaboração de um planejamento para deixar claro as etapas a serem seguidas e cumpridas para se alcançar o objetivo.

Antes de qualquer coisa, a equipe desenvolveu um plano geral para demonstrar os módulos que o projeto será composto, como pode ser visualizado na WBS apresentada na Figura 2.

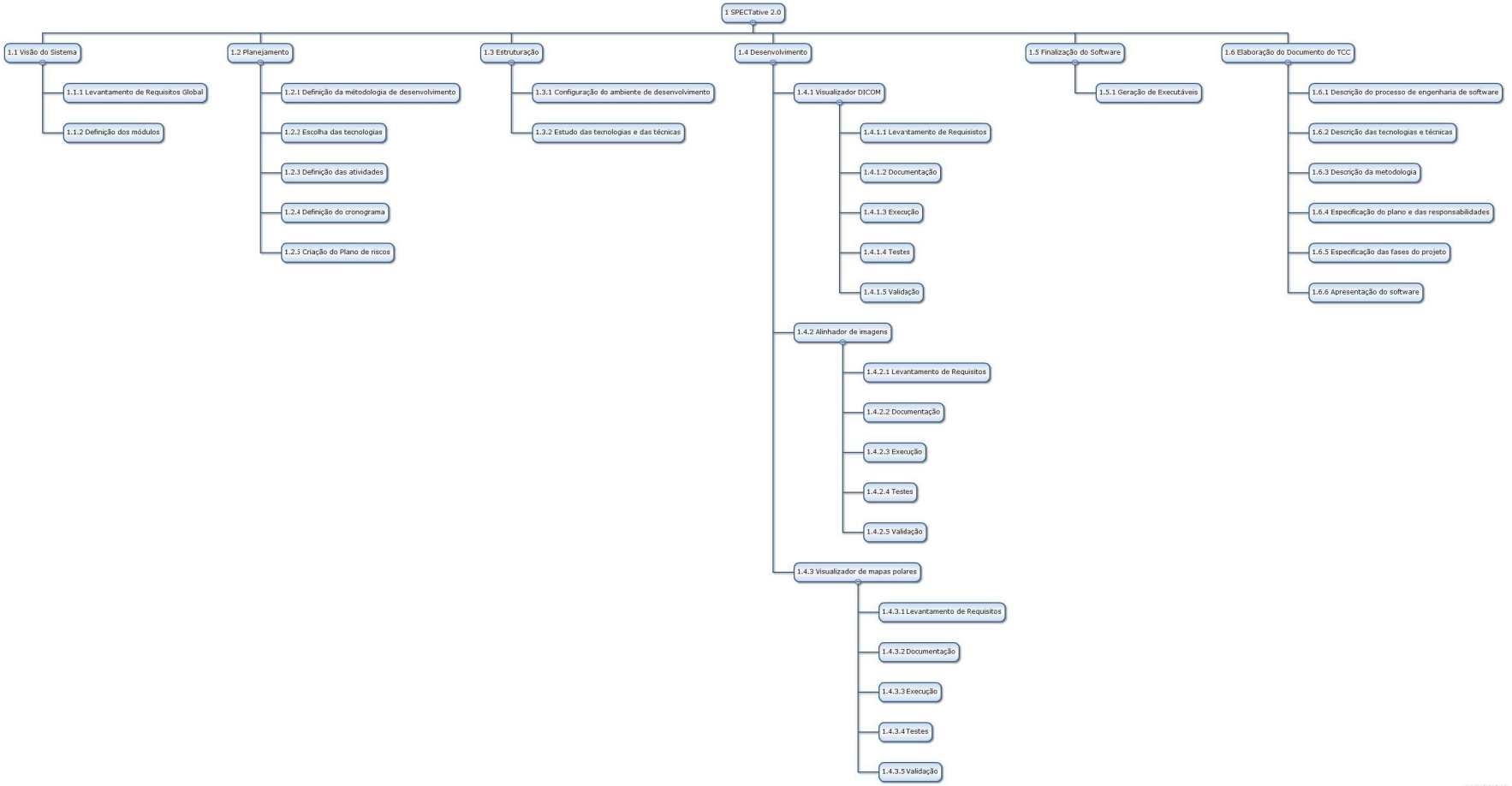
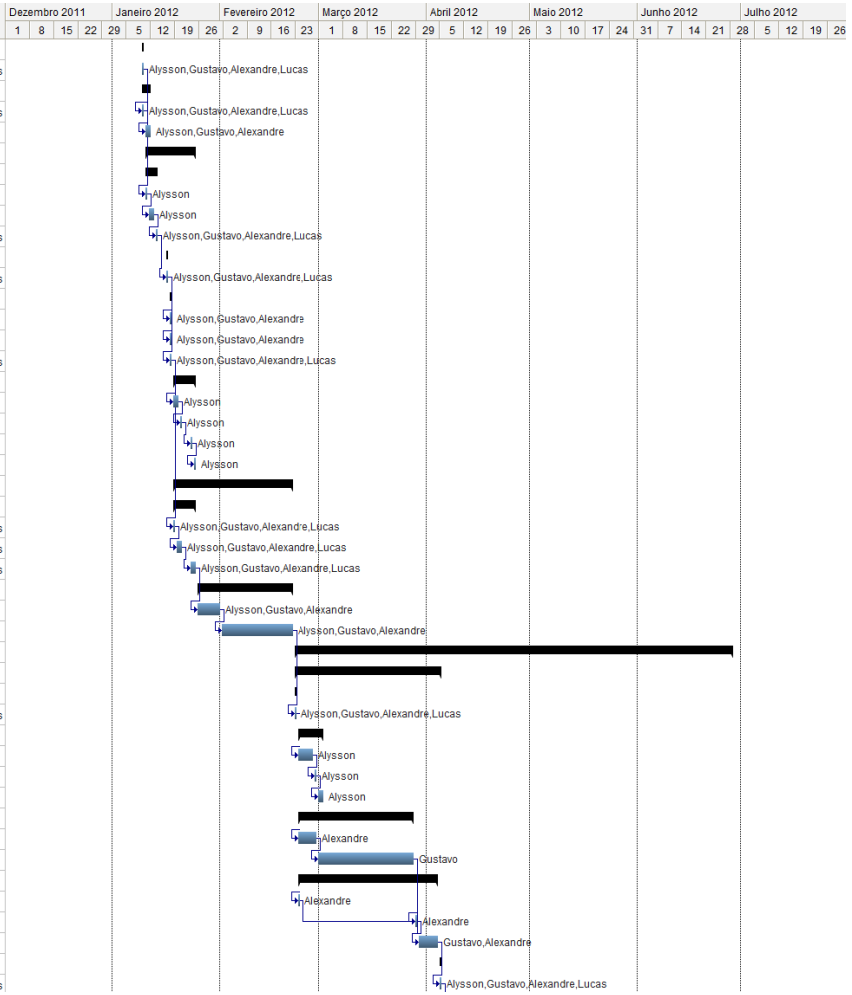

1 SPECTative 2.0
1.1 Visão do Sistema
1.1.1 Levantamento de Requisitos Global
1.1.2 Definição dos módulos
1.2 Planejamento
1.2.1 Definição da métodologia de desenvolvimento
1.2.2 Escolha das tecnologias
1.2.3 Definição das atividades
1.2.4 Definição do cronograma
1.2.5 Criação do Plano de riscos
1.3 Estruturação
1.3.1 Configuração do ambiente de desenvolvimento
1.3.2 Estudo das tecnologias e das técnicas
1.4 Desenvolvimento
1.4.1 Visualizador DICOM
1.4.1.1 Levantamento de Requisistos
1.4.1.2 Documentação
1.4.1.3 Execução
1.4.1.4 Testes
1.4.1.5 Validação
1.4.2 Alinhador de imagens
1.4.2.1 Levantamento de Requisitos
1.4.2.2 Documentação
1.4.2.3 Execução
1.4.2.4 Testes
1.4.2.5 Validação
1.4.3 Visualizador de mapas polares
1.4.3.1 Levantamento de Requisitos
1.4.3.2 Documentação
1.4.3.3 Execução
1.4.3.4 Testes
1.4.3.5 Validação
1.5 Finalização do Software
1.5.1 Geração de Executáveis
1.6 Elaboração do Documento do TCC
1.6.1 Descrição do processo de engenharia de software
1.6.2 Descrição das tecnologias e técnicas
1.6.3 Descrição da metodologia
1.6.4 Especificação do plano e das responsabilidades
1.6.5 Especificação das fases do projeto
1.6.6 Apresentação do software
www.wbstool.com

Figura 2 – WBS do projeto

Como pode ser observado, o gráfico de Gantt representa o projeto de uma forma global e as etapas não estão totalmente descritas. O principal objetivo deste gráfico é abstrair o projeto de modo a dividi-lo em pacotes menores e simplificar seu desenvolvimento.

Para especificar mais detalhadamente as atividades, os responsáveis e as datas, a equipe elaborou o gráfico de Gantt, Figura 3.

Nome	Duração	Início	Fim	Predecessores	Recursos
Levantamento de requisitos global	**1d**	**09/01/2012**	**09/01/2012**		
Reunião com o orientador/cliente para conhecer o problema	1d	09/01/2012	09/01/2012		Alysson,Gustavo,Alexandre,Lucas
Definição dos módulos	**3d**	**09/01/2012**	**11/01/2012**		
Definir os módulos que o sistema irá conter	1d	09/01/2012	09/01/2012	2	Alysson,Gustavo,Alexandre,Lucas
Elaborar o esboço das telas	2d	10/01/2012	11/01/2012	2,4	Alysson,Gustavo,Alexandre
Planejamento	**11d?**	**10/01/2012**	**24/01/2012**		
Definição da metodologia de desenvolvimento	**4d?**	**10/01/2012**	**13/01/2012**		
Levantamento das principais metodologias	1d?	10/01/2012	10/01/2012	2	Alysson
Estudo das metodologias levantadas	2d?	11/01/2012	12/01/2012	8	Alysson
Reunião para definição da melhor metodologia	1d	13/01/2012	13/01/2012	9	Alysson,Gustavo,Alexandre,Lucas
Escolha das tecnologias	**1d**	**16/01/2012**	**16/01/2012**		
Reunião com orientador para que sejam apresentas as tecnologias r	1d	16/01/2012	16/01/2012	10	Alysson,Gustavo,Alexandre,Lucas
Elaboração do cronograma	**1d**	**17/01/2012**	**17/01/2012**		
Definir o que cada integrante do grupo deve fazer no projeto	1d	17/01/2012	17/01/2012	12	Alysson,Gustavo,Alexandre
Definir as etapas do projeto	1d	17/01/2012	17/01/2012	12	Alysson,Gustavo,Alexandre
Definir as datas de entrega de cada funcionalidade	1d	17/01/2012	17/01/2012	12	Alysson,Gustavo,Alexandre,Lucas
Criação do plano de riscos	**5d?**	**18/01/2012**	**24/01/2012**		
Estudo teórico sobre planejamento de riscos	2d?	18/01/2012	19/01/2012	16	Alysson
Listar os riscos pertinentes ao projeto	1d?	20/01/2012	20/01/2012	18	Alysson
Definição da prioridade de cada risco	1d?	23/01/2012	23/01/2012	19	Alysson
Criação do plano de contingência	1d?	24/01/2012	24/01/2012	20	Alysson
Estruturação	**25d?**	**18/01/2012**	**21/02/2012**		
Configuração do ambiente de desenvolvimento	**5d?**	**18/01/2012**	**24/01/2012**		
Instalação do Linux	1d?	18/01/2012	18/01/2012	16	Alysson,Gustavo,Alexandre,Lucas
Instalação das bibliotecas ITK, VTK e QT	2d?	19/01/2012	20/01/2012	24	Alysson,Gustavo,Alexandre,Lucas
Integração das bilbiotecas	2d?	23/01/2012	24/01/2012	25	Alysson,Gustavo,Alexandre,Lucas
Estudo das tecnologias e das técnicas	**20d?**	**25/01/2012**	**21/02/2012**		
Levantamento bibliográfico	5d?	25/01/2012	31/01/2012	26	Alysson,Gustavo,Alexandre
Aprender o funcionamento básico das bibliotecas	15d?	01/02/2012	21/02/2012	28	Alysson,Gustavo,Alexandre
Desenvolvimento	**91d?**	**22/02/2012**	**27/06/2012**		
Visualizador DICOM	**31d?**	**22/02/2012**	**04/04/2012**		
Levantamento de requisitos	**1d**	**22/02/2012**	**22/02/2012**		
Reunião com orientador/cliente para detalhar o funcionamento do	1d	22/02/2012	22/02/2012	29	Alysson,Gustavo,Alexandre,Lucas
Documentação	**6d?**	**23/02/2012**	**01/03/2012**		
Elaboração e especificação dos diagramas de casos de uso	3d?	23/02/2012	27/02/2012	33	Alysson
Elaboração do diagrama de classe	1d?	28/02/2012	28/02/2012	35	Alysson
Elaboração dos diagramas de sequência	2d?	29/02/2012	01/03/2012	36	Alysson
Execução	**24d?**	**23/02/2012**	**27/03/2012**		
Criação da interface	4d?	23/02/2012	28/02/2012	33	Alexandre
Criação do algoritmo de visualização	20d?	29/02/2012	27/03/2012	39	Gustavo
Testes	**29d?**	**23/02/2012**	**03/04/2012**		
Criação do plano de teste	1d?	23/02/2012	23/02/2012	33	Alexandre
Teste do módulo	1d?	28/03/2012	28/03/2012	40,42	Alexandre
Correção de falhas	4d?	29/03/2012	03/04/2012	40,43	Gustavo,Alexandre
Validação	**1d**	**04/04/2012**	**04/04/2012**		
Apresentação do módulo ao orientador/cliente	1d	04/04/2012	04/04/2012	44	Alysson,Gustavo,Alexandre,Lucas



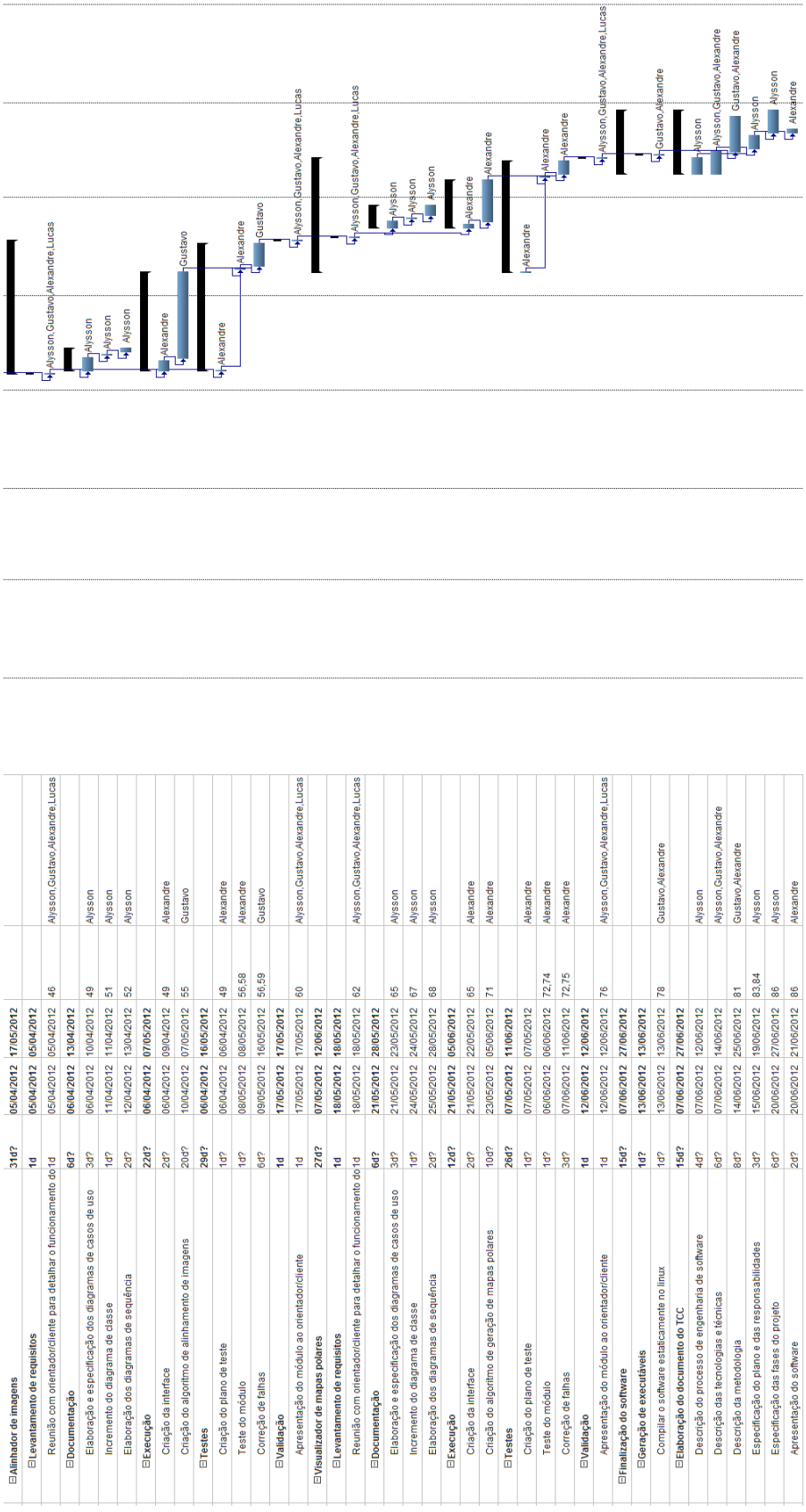


Figura 3 – Gráfico de Gantt

Como pode ser visto na figura acima, primeiramente foi definida a data inicial e final do projeto, sendo iniciado dia 09 de janeiro e finalizado até dia 21 de junho de 2012. O primeiro passo definido pelo grupo foi o de entender e abstrair o sistema que deve ser construído, junto ao cliente/orientador. Com a ideia geral bem elaborada, foi feito o planejamento para estabelecer as ferramentas a serem usadas, alocação dos integrantes, definição das datas e o responsável por cada entrega e levantar os riscos eminentes ao projeto. Após estabelecer a base do projeto, deu-se início a preparação técnica e de aprendizado necessário para desenvolver a aplicação. A partir disto é que a equipe pôde partir para a fase de desenvolvimento propriamente dita. Como o projeto foi desenvolvido aliado a metodologia ágil scrum, o desenvolvimento foi dividido em três estágios, cada um entregando ao final um módulo funcional do sistema. Como descrito no gráfico de Gantt, as três etapas foram compostas por análise, documentação, execução, testes e validação. Ao final disto, o desenvolvimento do sistema estará finalizado. Em seguida, entra a concepção do pacote do software, que será a geração do executável que deverá funcionar no sistema operacional Linux. Para concluir o projeto, a equipe entra na fase de elaboração do documento do TCC, que engloba as atividades de descrição de todas as técnicas, metodologias, planejamento e conduta de desenvolvimento.

3.3 PLANO DE RISCOS

Segundo Pressmann (1995), os fatos presentes neste momento são resultados de alguma ação que foi executada a tempos atrás, ou seja, tudo o que fazemos hoje nos resulta em benefícios ou prejuízos futuros. Desta forma, para que os prejuízos sejam minimizados, é necessário que todos os riscos envolvidos no processo sejam identificados e ocorram mudanças de comportamento para criar oportunidades a partir dessas incertezas. Com isso, pode-se dizer que o projeto de sucesso não é aquele que apenas teve um bom planejamento, preço, prazo e qualidade. Para seu total êxito, todas as situações de imprevisibilidade deverão ser identificadas, planejadas, executadas e gerenciadas ao longo do projeto (Martins, 2007).

Assim, este plano fará um levantamento dos riscos mais prováveis e esta seção está dividida em: identificação de riscos, análise qualitativa e planejamento de respostas a riscos. O plano de riscos pode ser encontrado no Apêndice – Plano de Riscos.

3.3.1 Identificação de Riscos

De acordo com Martins (2007), a fase de identificação de riscos consiste em fazer o levantamento das incertezas que podem estar presentes ao longo do desenvolvimento do projeto, e nela todas as pessoas envolvidas devem participar. Neste trabalho, as entradas para este processo foi o backlog do produto e o cronograma.

Como modelo, o presente trabalho segue a sugestão do Pressmann (1995), que define de forma macro a divisão dos riscos em três categorias, sendo elas: riscos do projeto, riscos técnicos e riscos de negócio.

Na primeira categoria se enquadram os riscos voltados a custo, cronograma, pessoas e requisitos. Já os riscos técnicos englobam problemas de projeto, desenvolvimento, interfaces, controle e manutenção. Por último, os riscos de negócio descrevem as incertezas que podem ocorrer por parte dos clientes e usuários.

3.3.2 Análise Qualitativa de Riscos

Esta etapa é responsável por identificar a chance de um risco ocorrer e seus impactos sobre o projeto (Martins, 2007). A priorização dos riscos identificados para este projeto foram feitos baseados na visão da equipe a respeito do projeto como um todo. A prioridade e probabilidade de ocorrência de cada risco foram enquadradas nas seguintes classes: riscos de baixa, média e alta probabilidade. Na classe baixa, a chance de confirmação do risco fica entre 1 e 3 e para que o

problema não se alastre é apenas feito um controle simples. No segundo caso, o valor aumenta para 4 a 6, necessitando de uma monitoração constante. Já os riscos de alto grau possuem valores iguais ou maiores a 7 e para evitá-los devem ser executadas ações diárias ou mudanças de comportamento sobre o projeto, como descrito por Martins (2007).

3.3.3 Planejamento de Respostas a Riscos

O planejamento de respostas a riscos é um processo realizado para descrever a forma com que os riscos serão tratados, com o intuito de invalidar ou simplesmente minimizar seus impactos (Martins, 2007). Neste trabalho foi elaborado o plano de contingência que descreve os caminhos a serem seguidos após a ocorrência do problema e quando deverá ser executada.

3.4 RESPONSABILIDADES

No início do projeto foram definidas as principais funções que cada um deveria exercer ao longo do trabalho. Na decisão inicial foi definido que o Alysson seria o analista e responsável pelo projeto, o Alexandre seria o criador das interfaces e responsável pelos testes e o Gustavo ficaria como desenvolvedor do visualizador e do alinhador. Além disso, o projeto seria orientado pelo professor Lucas Ferrari de Oliveira.

Apesar de ter sido feito a alocação do pessoal como descrito anteriormente, no decorrer do desenvolvimento do projeto se fez necessário reordenar a atribuição de algumas atividades, como será descrito no item 3.8. Desta forma, cada pessoa teve um papel dentro da confecção do software e da sua documentação, como descrito abaixo.

- Alysson
    - Levantamento de requisitos

  - Definição da metodologia ágil
  - Elaboração da documentação
- Gustavo
  - Desenvolvimento do visualizador
  - Desenvolvimento do alinhador
- Alexandre
  - Desenvolvimento das interfaces
  - Criação da exibição de mapas polares
  - Elaboração dos testes

3.5 MATERIAIS

Para a concepção desse trabalho foram utilizados diferentes softwares, bibliotecas de desenvolvimento e computadores, como será exibido logo abaixo.

Como base, foram utilizados três notebooks com as seguintes configurações:

- Processador Intel Core i3 2.27GHz, 2Gb de memória RAM e 500Gb de disco rígido;
- Processador Intel Core2Duo 2.2GHz, 2Gb de memória RAM e 320Gb de disco rígido;
- Processador Core2Duo 2.1GHz, 3Gb de memória RAM e 320Gb de disco rígido.

Para a criação do software optou-se pelo uso do Linux Debian e Ubuntu 12.04 LTS. Com isso, para construção do sistema foram utilizadas as seguintes bibliotecas e ferramentas:

- ITK (*Insight Toolkit*) – versão 3.20.0
- VTK (*Visualization Toolkit*) – versão 5.8.0
- ITK VTK Glue
- QT – versão 4.8.2
- CMake – versão 2.8.7
- XMedcon (visualizador de imagens médicas) – versão 0.10.7

De modo a apoiar o desenvolvimento da aplicação, foi feito o uso da IDE Code::Blocks e o editor Vim. Da mesma forma, a documentação e o planejamento foram feitos com as ferramentas exibidas abaixo:

- WBSTool – versão on-line
- Gantter – versão 3.0
- Astah Professional – versão 6.6.3
- BROffice - versão 3.2.1
- Google Docs
- Balsamiq versão - 1.6.69

3.6 ARQUITETURA DO SOFTWARE

O software foi desenvolvido seguindo a padronização de código proposta pelo *ITK Style Guide* do *Insight Software Consortium*. A arquitetura do software buscou se aproximar de uma abordagem MVC, imitando outras aplicações semelhantes. As GUIs ficaram como classes do tipo *view* e apenas fazem a interação com o usuário. Já a troca de informações entre a *view* e o *model* é feita

pelas classes que chamamos de *Interactor*, que são do tipo *control*. As classes do tipo *model* foram desenvolvidas de modo que uma seja facilmente conectável à outra. Por exemplo o leitor de imagens desenvolvido em ITK pode ser ligado tanto ao filtro de geração de mapa polar quanto ao filtro de alinhamento de imagens que, por sua vez, pode ser ligado à ele mesmo e depois à janela de renderização do visualizador ou do alinhador. Isso é possível graças aos métodos *SetInput* e *GetOutput* presentes em todas as classes onde são necessários e que, no caso do ITK, usam, exclusivamente, o tipo *itkImage* para entrada e saída, assim como no caso do VTK é usado o tipo *vtkImageData*. Devido ao fato de os tipos usados pelo ITK e VTK serem diferentes foi necessário usar métodos para conversão. Esses métodos são encontrados no ITKVTKGlue, que faz a ligação entre as duas bibliotecas.

3.7 DESENVOLVIMENTO DO PROJETO

Como já visto anteriormente no item 3.2.1.1, o trabalho foi desenvolvido tendo como base a metodologia ágil Scrum. De praxe, a equipe primeiramente estudou os requisitos levantados pelo cliente, para, em seguida, fazer uma pesquisa minunciosa pelas melhores opções de tecnologias. Em seguida estimou-se um cronograma, o qual foi validado por todos os membros. Em paralelo com a familiarização das técnicas de utilização das tecnologias selecionadas e preparação dos ambientes de desenvolvimento (SO Ubuntu e Debian com as bibliotecas QT, VTK e ITK), foi estudado, elaborado e determinado o Plano de Riscos. Para abstrair o sistema, foram desenhadas algumas telas que serviriam de esboço para o produto, as quais foram devidamente validadas pelo cliente. As tarefas foram divididas entre os membros por aptidão: o Alexandre foi atribuído as atividades referentes a tecnologia QT e o Gustavo às bibliotecas VTK e ITK, pois ambos tinham mais facilidades e conhecimento prévio nestas áreas. Enquanto isso, o Alysson demonstrou facilidade

em fazer análises de requisito, riscos, e outros planejamentos afins, tendo a documentação do projeto sido em maior parte elaborada por ele. Dessa forma, o Alexandre ficou responsável pelo desenvolvimento de todas as interfaces do produto, com base no que havia sido esboçado pelo Alysson, ao mesmo tempo que o Gustavo ficou como principal responsável pelo desenvolvimento de algoritmos relacionados aos processamentos das imagens, tendo sido eventualmente ajudado pelo Alexandre. Além disso, o Alexandre ficou responsável pelo desenvolvimento do código de geração de mapa polar. Apesar da ótima paridade da Análise e Documentação do produto com o seu cronograma, alguns atrasos puderam ser observados na parte de desenvolvimento de algoritmos complexos, o que já era esperado devido sua complexidade. O Gustavo teve problemas de atraso com a entrega do algoritmo de Alinhamento de Imagens, devido à sua dificuldade tanto teórica quanto prática. Da mesma forma, surgiram problemas por conta de ser difícil encontrar fontes de pesquisa e aplicá-las. De acordo com o presente no Risco 1, que pode ser encontrado no Apêndice, o qual prevê a situação pela qual o Gustavo passou, foi feita a realocação de recursos no auxílio da solução do problema. No entanto, optou-se por alocar o Alexandre na tarefa de desenvolver o código de Geração de Mapa Polar (anteriormente sob responsabilidade também do Gustavo). Havia ciência da equipe de que tal estratégia aumentava em muito as chances de acontecer o previsto no Risco 2, já que o Alexandre não possuía conhecimento suficiente das bibliotecas VTK e ITK, essenciais para que se pudesse desenvolver o código do Mapa Polar, gerando um “efeito cascata” de riscos. No entanto, apesar de também atrasar o cronograma com essa tarefa, o Alexandre conseguiu obter o resultado esperado logo depois do fim do prazo, possibilitando uma correção posterior sem maiores dificuldades.

3.8 ESTIMATIVAS DO PROJETO

Para o trabalho proposto, optou-se pelo uso da técnica de Pontos de Caso de Uso para realizar a estimativa de tempo do projeto, pois é uma técnica simples e bastante utilizada em projetos orientados a objetos.

3.8.1 Pontos de Caso de Uso – UCP

Segundo Monteiro (2005), a estimativa UCP foi criada por Karner com o objetivo de estimar o tempo que um projeto de software levaria para ser desenvolvido. Esta técnica surgiu com base na Análise de Pontos de Função (FPA), MK II FPA e no processo Objectory. Devido sua facilidade de implementação, a estimativa de Pontos de Caso de Uso tem feito com que as empresas tenham maior interesse em aplicar este tipo de técnica nos projetos de software (MONTEIRO, 2005). Para a elaboração da estimativa o processo é dividido em seis etapas: (i) classificação dos atores; (ii) classificação dos casos de uso; (iii) cálculo dos pontos de casos de uso não-ajustados; (iv) cálculo dos fatores técnicos; (v) cálculo dos fatores ambientais; e (vi) cálculo dos pontos de caso de uso.

3.8.1.1 Classificação dos Atores

Nesta primeira etapa os atores envolvidos com o sistema deverão ser classificados por níveis de complexidade.

Complexidade do Ator	**Descrição**	**Peso**
Simples	Médico que usa computador rotineiramente e não tem dificuldades	1
Médio	Médico que faz uso do computador apenas quando precisa	2
Complexo	Médico que não tem domínio em usar o computador	3

Após a definição das possíveis classes que o usuário do sistema proposto pode ser classificado, é feito o cálculo do UAW (*Unadjusted Actor Weight*).

Atores	Quantidade	Peso	Peso Total
Médio	1	2	1 * 2 = 2
TOTAL (UAW = ∑Peso Total dos Atores)			2

3.8.1.2 Classificação dos Casos de Uso

Como no caso anterior, antes de calcular os pesos para cada caso de uso, deve-se estabelecer o que torna um caso de uso complexo ou não. Para isso, foram usadas características referentes ao grau de impacto da funcionalidade na análise da imagem que está sendo mostrada ao usuário.

Complexidade do Caso de Uso	Descrição	Peso
Simples	A funcionalidade **não** impacta na análise do conteúdo da imagem	5
Médio	A funcionalidade impacta **pouco** na análise do conteúdo da imagem	10
Complexo	A funcionalidade impacta **muito** na análise do conteúdo da imagem	15

A partir do momento em que se tem definido a complexidade que os casos de uso podem assumir é possível calcular o UUCW (*Unadjusted Use Case Weight*).

Complexidade	Quantidade	Peso	Peso Total
Simples	3	5	15
Médio	2	10	20
Complexo	3	15	45
TOTAL (UUCW = ∑Peso Total dos Casos de Uso)			**80**

3.8.1.3 Pontos de Casos de Uso Não Ajustados

Para se obter o UUCP (*Unadjusted Use Case Points*) deve-se somar o resultado da complexidade dos atores com a complexidade dos casos de uso.

UUCP = ∑UAW + ∑UUCW = 2 + 80 = 82

3.8.1.4 Cálculo dos Fatores Técnicos

Para efetuar o cálculo dos fatores técnicos serão utilizados os ajustes calculados pela Análise por Pontos de Função. Com isso é possível calcular o ajuste para os fatores técnicos, os quais irão representar os requisitos funcionais do sistema (MONTEIRO, 2005). O valor dos pesos pode variar de 0 a 5, sendo 0 (zero) um requisito insignificante ao sistema, 3 indica relevância média e 5 grande influência e alta complexidade.

Fator	Descrição	Peso	Nota	Peso Total
T1	O sistema é distribuído	2,0	0	0
T2	Desempenho da Aplicação.	1,0	3	3
T3	Eficiência do Usuário-Final.	1,0	5	5
T4	Processamento Interno Complexo	1,0	5	5
T5	Reusabilidade do código em outras aplicações.	1,0	3	3

T6	Facilidade de Instalação.	0,5	3	1,5
T7	Usabilidade.	0,5	5	2,5
T8	Portabilidade.	2,0	3	6
T9	Facilidade de Manutenção.	1,0	3	3
T10	Concorrência.	1,0	1	1
T11	Características especiais de segurança.	1,0	1	1
T12	Facilidades especiais de treinamento.	1,0	1	1
TFator				**32**

TCF = 0.6 + (0.01 * TFator) = 0.6 + (0.01 * 32) = 0.92

3.8.1.5 Cálculo dos Fatores Ambientais

Diferente do cálculo dos fatores técnicos, o valor representado pelos fatores ambientais irão descrever os requisitos não-funcionais do sistema, como a experiência da equipe, estabilidade do projeto e a motivação dos integrantes (MONTEIRO, 2005). O pesos estipulados para os fatores ambientais são os mesmos que os utilizados pelo cálculo dos fatores técnicos, diferenciando apenas no significado. Por exemplo, para o fator F2, 0 (zero) indica que a equipe não tem conhecimento a respeito do assunto referente ao projeto, 3 indica que os integrantes já tiveram alguma experiência e 5 mostra que a equipe já possui domínio na área correlata.

Fator	Descrição	Peso	Nota	Peso Total
F1	Familiaridade com o processo de desenvolvimento de software	1,5	3	4,5
F2	Experiência na aplicação	0,5	1	5
F3	Experiência com OO, na linguagem e na técnica	1,0	2	2

	de desenvolvimento			
F4	Capacidade do líder de análise	0,5	5	2,5
F5	Motivação	1,0	5	5
F6	Requisitos estáveis	2,0	4	8
F7	Trabalhadores com dedicação parcial	-1,0	5	-5
F8	Dificuldade na linguagem de programação	-1,0	3	-3
EFator				**19**

EF = 1.4 + (-0.03 * EFator) = 1.4 + (-0.03 * 19) = 0.83

3.8.1.6 Cálculo dos Pontos de Caso de Uso

Após encontrar o valor dos pontos de casos de uso não ajustados (UUCP), dos fatores técnicos (TCF) e dos fatores ambientais e possível obter o tamanho do projeto, como mostrado na equação abaixo:

UCP = UUCP * TCF * EF = 82 * 0.92 * 0.83 = 62.61

3.8.1.7 Estimativa de Esforço do Projeto

De acordo com MONTEIRO (2005), para chegar a estimativa do esforço total do projeto, é necessário fazer o uso de um fator de produtividade. Este fator é dado em horas e é escolhido o número de horas de acordo com a produtividade da equipe. Para este trabalho será utilizada 28 horas por caso de uso como fator de produtividade. Com isto, tem-se a estimativa de esforço do projeto.

Esforço = UCP * PROD = 62 * 28 = 1753.22 horas

Este esforço indica que se cada integrante da equipe trabalhar durante 3 horas no projeto, resultando em 9 horas diárias, o projeto será desenvolvido em aproximadamente 195 dias.

4 APRESENTAÇÃO DO SOFTWARE

4.1 INSTALAÇÃO

Atualmente, o sistema desenvolvido só está preparado para ser executado na plataforma Linux. Assim, para que os passos descritos a seguir funcionem, o usuário deverá ter o sistema operacional Linux. Como a equipe usou o Ubuntu e o Debian, não há como garantir o perfeito funcionamento com outras distribuições do linux.
OBS: É possível que durante a instalação das bibliotecas seja necessário a instalação de outros arquivos dependentes. Além disso, o tempo de compilação das bibliotecas geralmente não é muito rápido, podendo levar algumas horas.

Para rodar a aplicação, o primeiro passo é a instalação do CMake, o qual permitirá a compilação das bibliotecas QT, VTK, ITK e ItkVtkGlue. Para realizar o download do CMake basta acessar o link http://www.cmake.org/cmake/resources/software.html e baixa-lo, na versão para Linux. Até o momento final da escrita deste trabalho o CMake se encontrava na versão 2.8.9. Para efetuar a instalação basta seguir os passos descritos na página que pode ser acessada pelo link http://www.cmake.org/cmake/help/install.html.

Com o CMake instalado, é possível obter a biblioteca do QT para criação da interface do sistema. Para obtê-la, basta acessar o site http://qt-project.org/downloads e baixar o aquivo Qt libraries 4.8.3 for Linux/X11 (223 MB). As etapas de instalação podem ser vistas no próprio site do QT (http://qt-project.org/doc/qt-4.8/install-x11.html).

A próxima etapa é a instalação do VTK, responsável pela exibição das imagens, o qual deverá ser adquirido no site da KitWare, acessando o link vtk-

5.8.0.tar.gz da página http://vtk.org/VTK/resources/software.html. Para rodar a aplicação desenvolvida, deve-se baixar a versão 5.8.0 do VTK, pois foi com esta versão que o software foi desenvolvido e ele pode apresentar incompatibilidade com outras versões.

Antes de compilar o VTK, é necessário configurá-lo, com o uso do CMake. Para isso, acesse o diretório em que o VTK foi descompactado e execute o comando “ccmake .”.

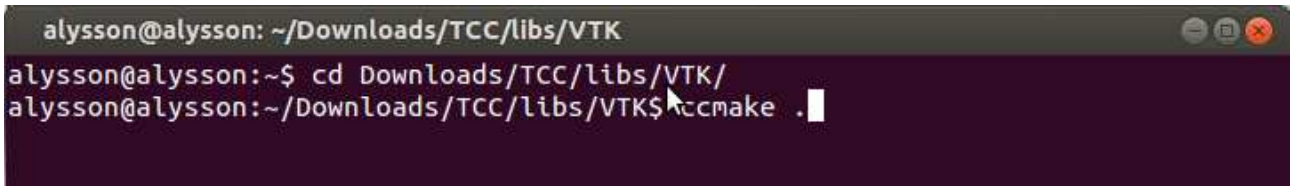


Figura 4 – Comando para acessar as configurações de compilação do VTK

Em seguida irá aparecer a tela de configuração.

Figura 5 – Tela de pré-configuração do VTK

Pressione a tecla “c” para realizar a configuração inicial.

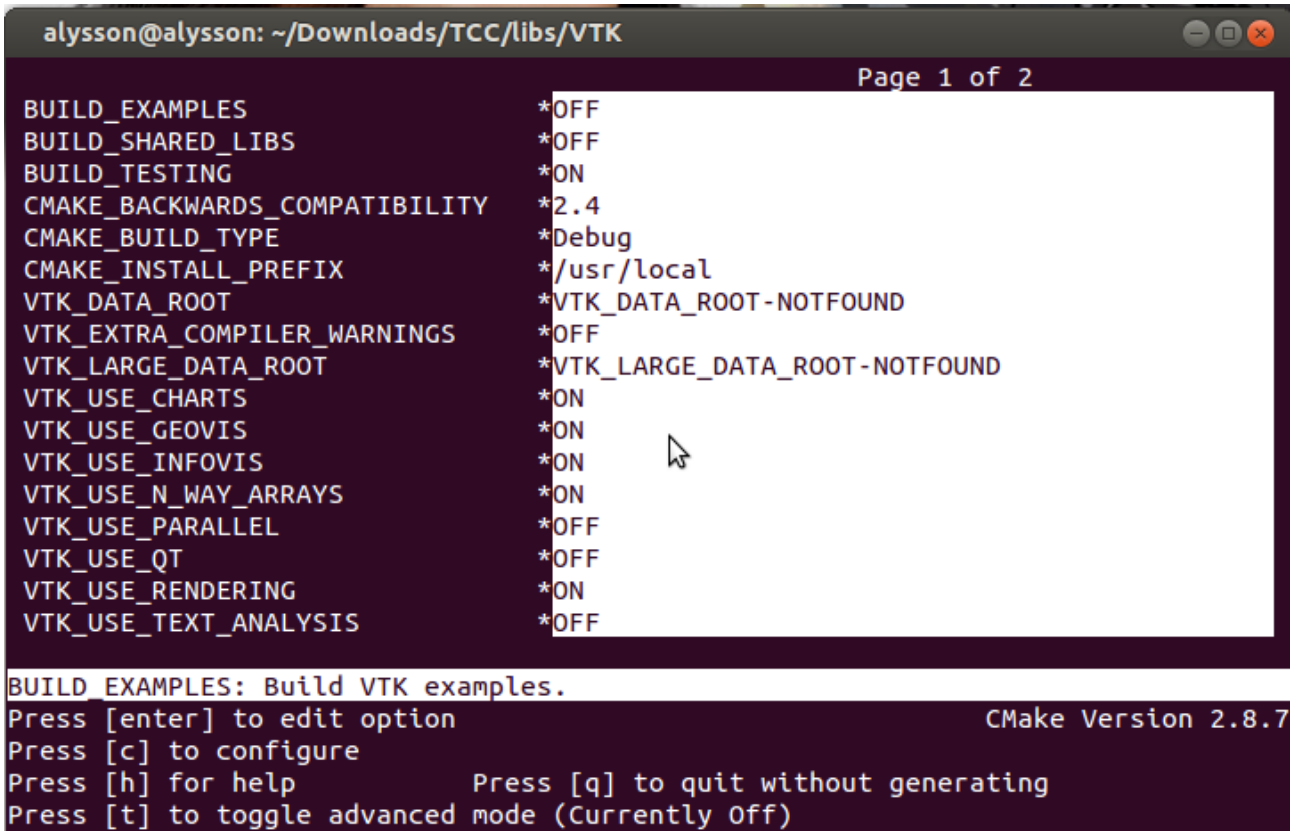


Figura 6 – Configuração do diretório do VTK

Para que o CMake compile o VTK é necessário fornecer o diretório que o VTK se encontra, inserindo no campo “VTK_DATA_ROOT”. Além disso, como deseja-se utilizar a imagem gerada dentro de uma interface QT, o parâmetro “VTK_USE_QT” deve ser alterado para “ON”. Em seguida, pressione a tecla “c” e depois a tecla “g”. Com a finalização da configuração do VTK é só rodar o comando “make” ou “make -j <numero_de_nucleos_processador>”, para compilar utilizando todos os núcleos do processador e diminuir o tempo de compilação.

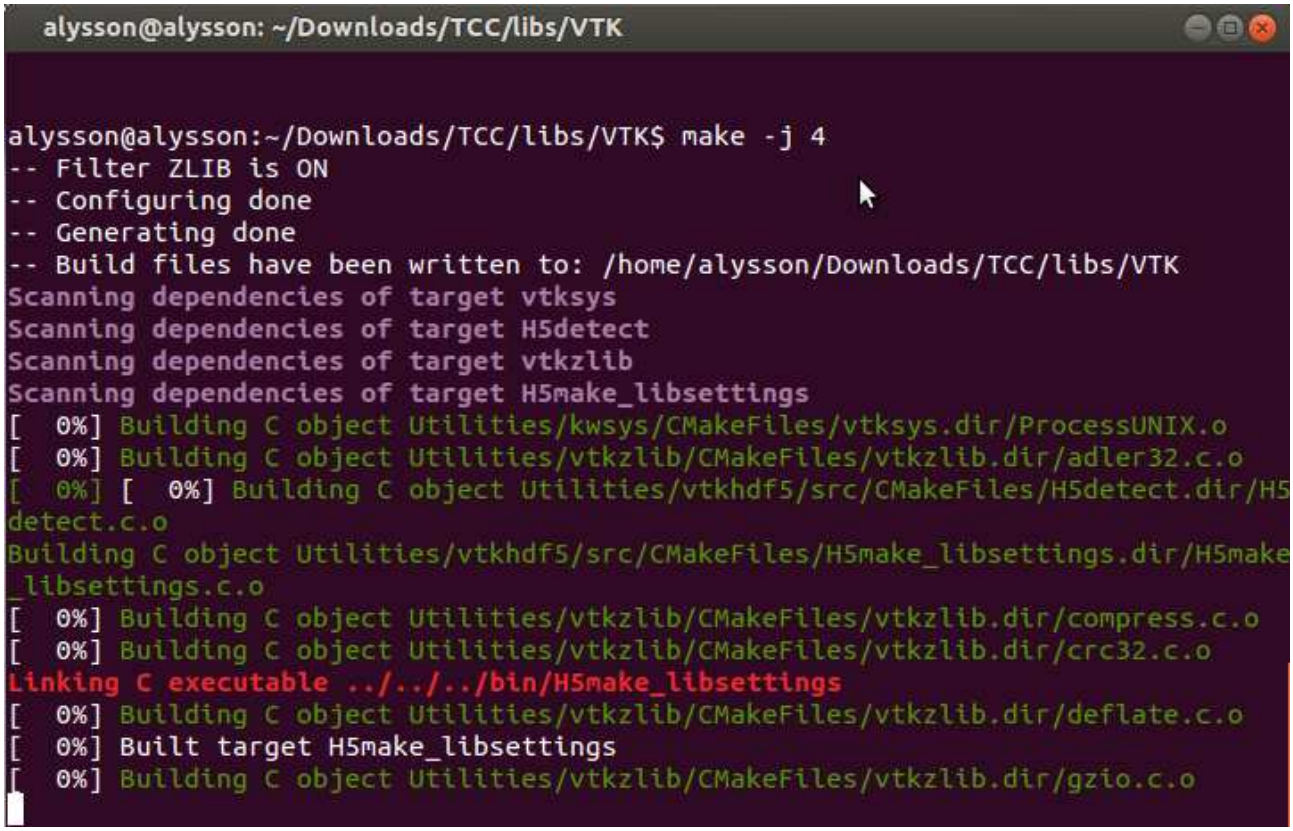


Figura 7 - Tela de compilação do VTK

Com o VTK funcionando, dirija-se a página de download do ITK (http://itk.org/ITK/resources/software.html) e baixe os arquivos pelo link InsightApplications-4.2.0.tar.gz (hosted at SourceForge). Da mesma forma que o VTK, o ITK deve ter a versão 4.2.0 para que a aplicação funcione sem problemas. Para iniciar o processo, acesse a página em que a pasta do ITK foi descompactada e execute o comando “ccmake .”.

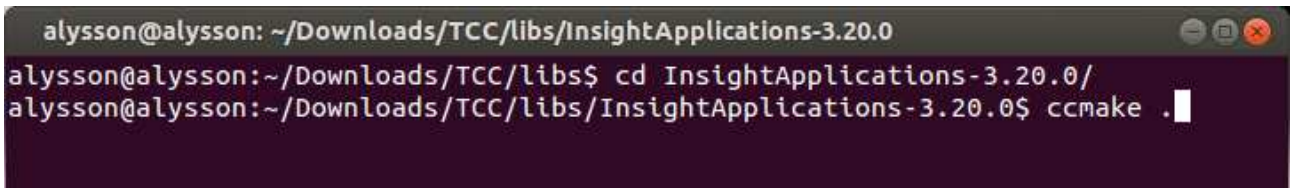


Figura 8 – Comando para acessar as configurações de compilação do ITK

Em seguida será exibida a tela de configuração.

Figura 9 – Tela de pré-configuração do ITK

Pressione a tecla “c”. Irá aparecer o erro informando que o diretório do ITK não foi encontrado.

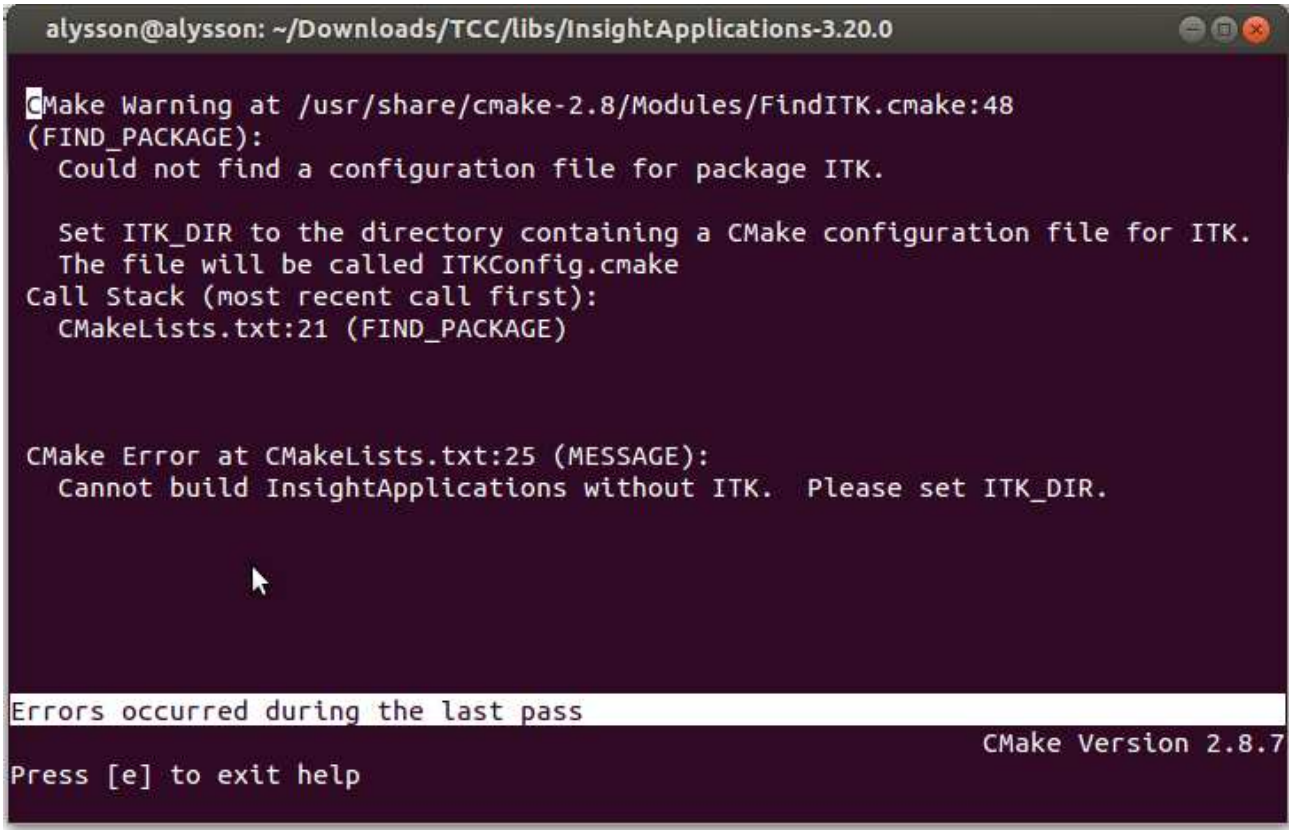


Figura 10 – Tela de erro indicando que o diretório do ITK não foi encontrado

Aperte a tecla “e” para prosseguir.

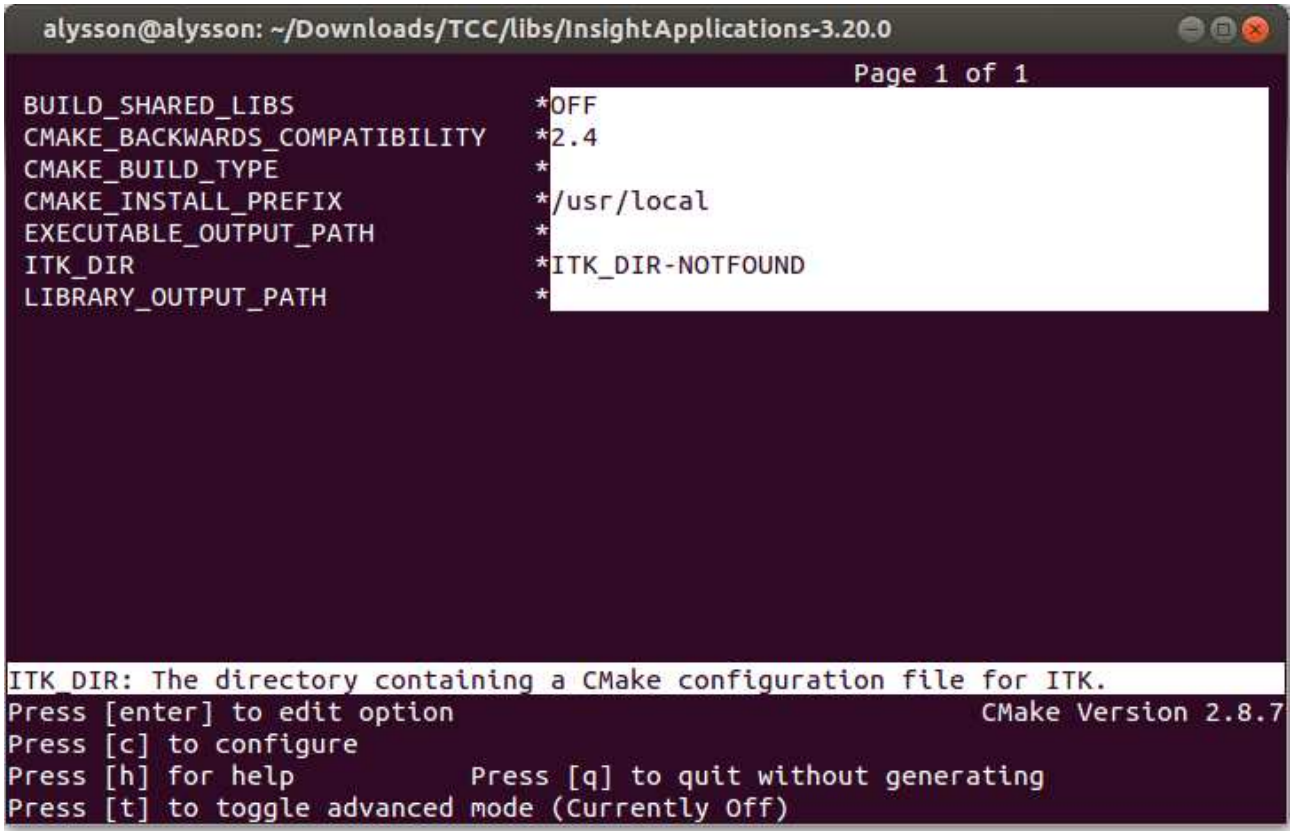


Figura 11 - Tela de configuração do diretório do ITK

Para indicar o diretório onde se encontram os arquivos do ITK basta passar o caminho completo da pasta do ITK, no parâmetro “ITK_DIR”. Após, pressione a tecla “c”. Novamente será mostrada a configuração, porém com mais itens. Desça até encontrar o item “USE_VTK” e coloque “ON”.

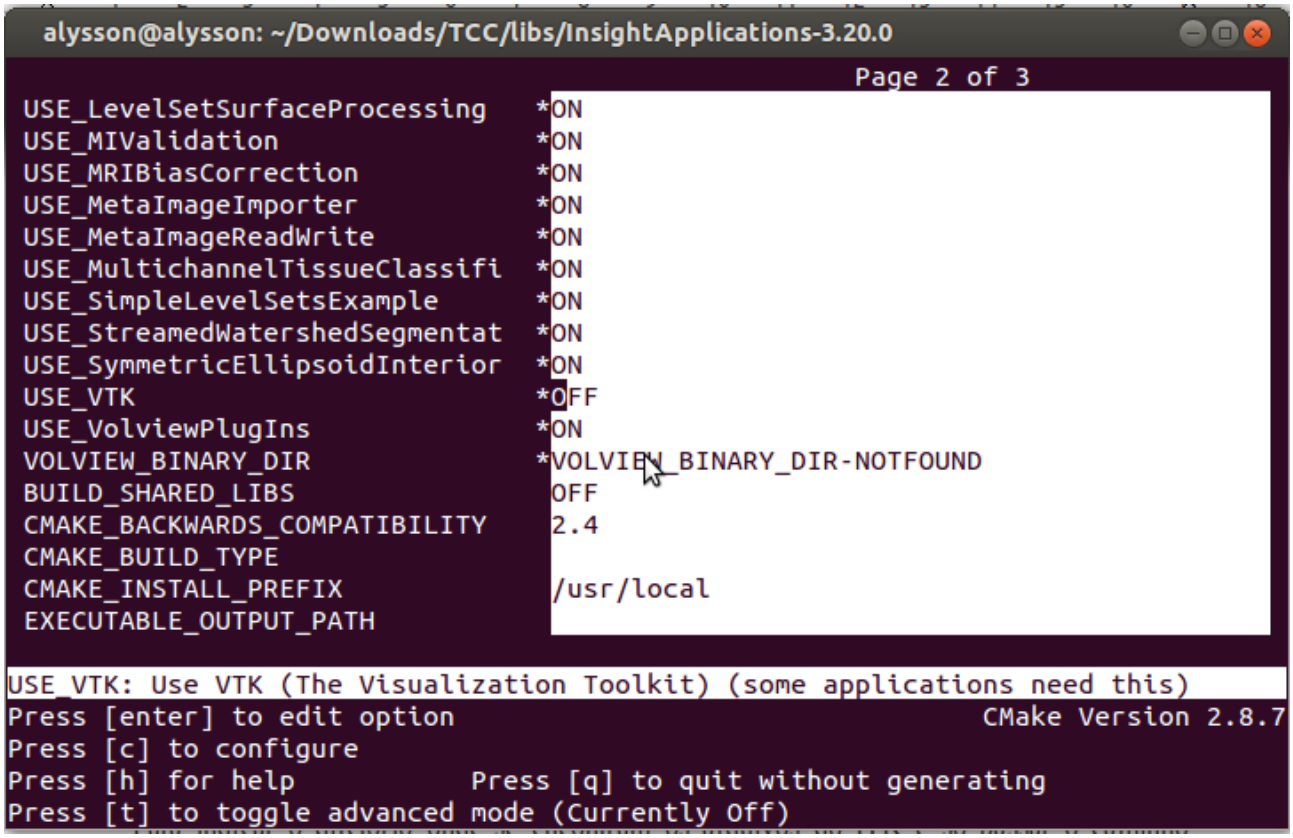


Figura 12 – Tela de configuração para selecionar a opção de integração do VTK com o ITK

Aperte “c” novamente e a tela de configuração irá aparecer de novo, só que dessa vez deverá informar o diretório do VTK no item “VTK_DIR”.

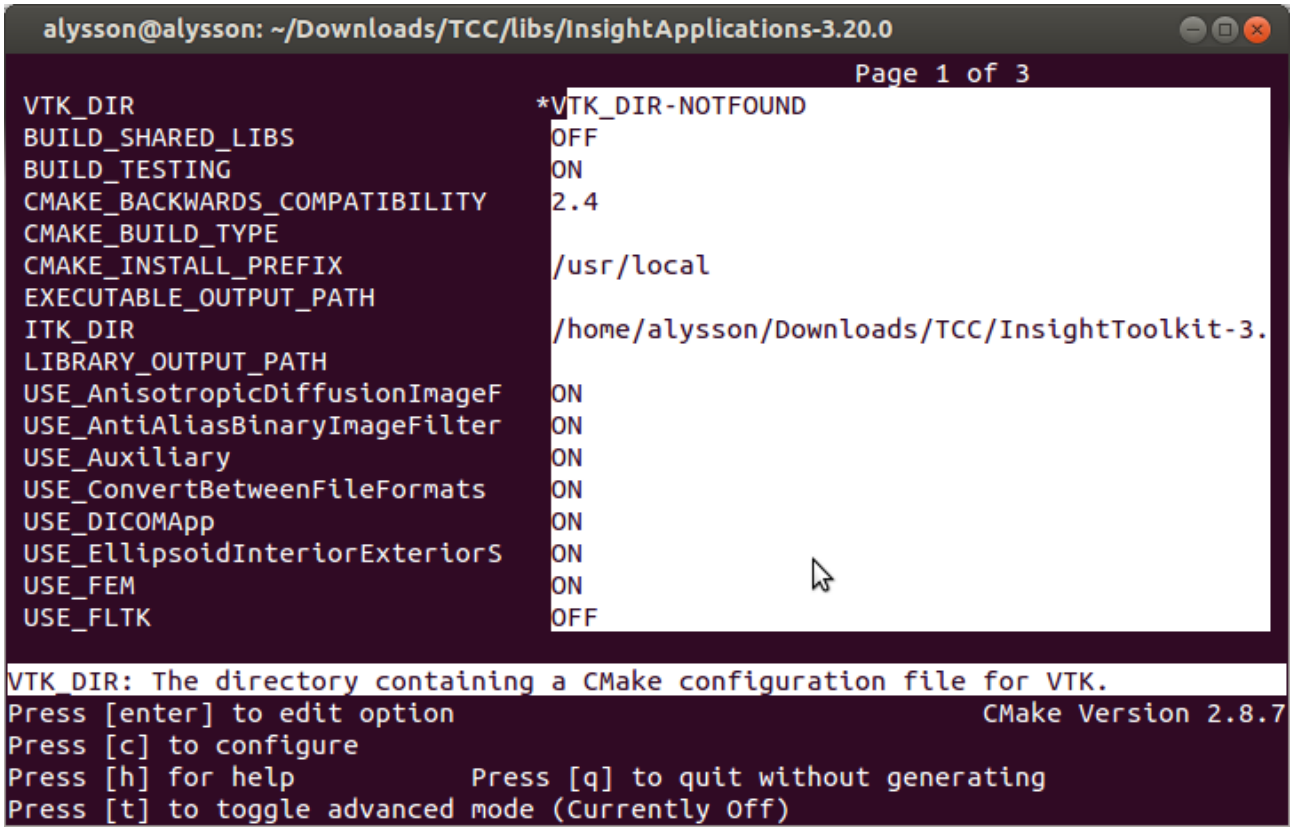


Figura 13 – Tela em que o usuário indica o diretório do VTK para integração com o ITK

Pressione “c” e em seguida “g” para finalizar o processo de configuração do ITK. Com a finalização da configuração do VTK é só rodar o comando “make” ou “make -j <numero_de_nucleos_processador>”, para compilar utilizando todos os núcleos do processador e diminuir o tempo de compilação.

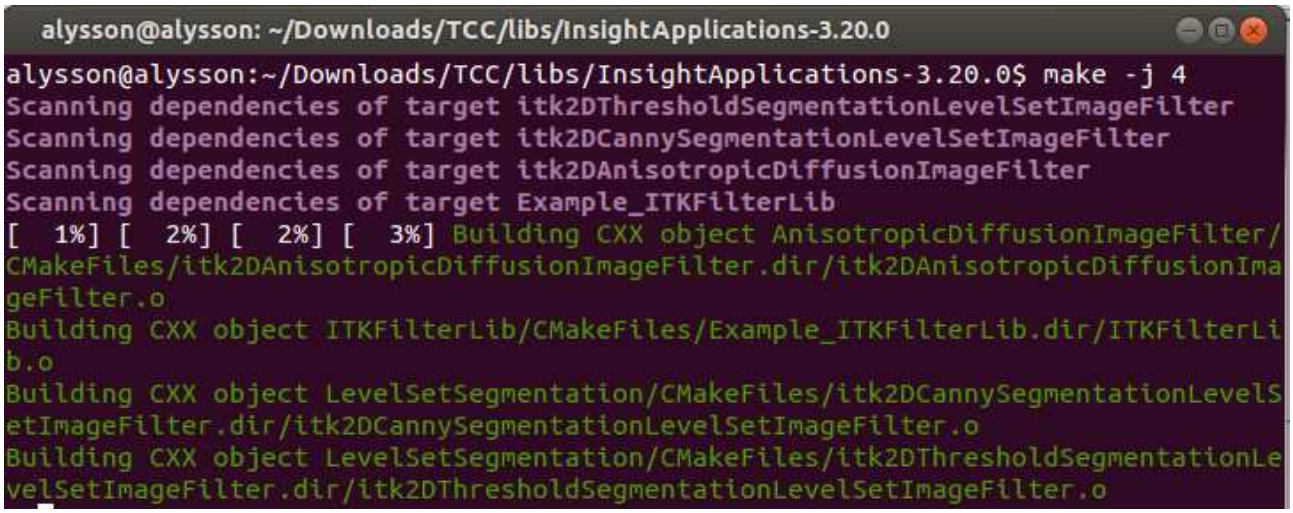


Figura 14 – Execução da compilação do ITK

A partir do momento em que se tem instaladas as bibliotecas QT, VTK e ITK é necessário usar a biblioteca ItkVtkGlue para integrar o ITK com o VTK. Como não foi localizada a página oficial de download do ItkVtkGlue, o arquivo de download pode ser obtido pelo link http://www.4shared.com/zip/sDCyIW1K/tcc_itkvtkglue.html.

Da mesma forma que nas instalações anteriores, primeiro deve acessar o diretório que o ItkVtkGlue foi descompactado e executar o comando “ccmake .”.

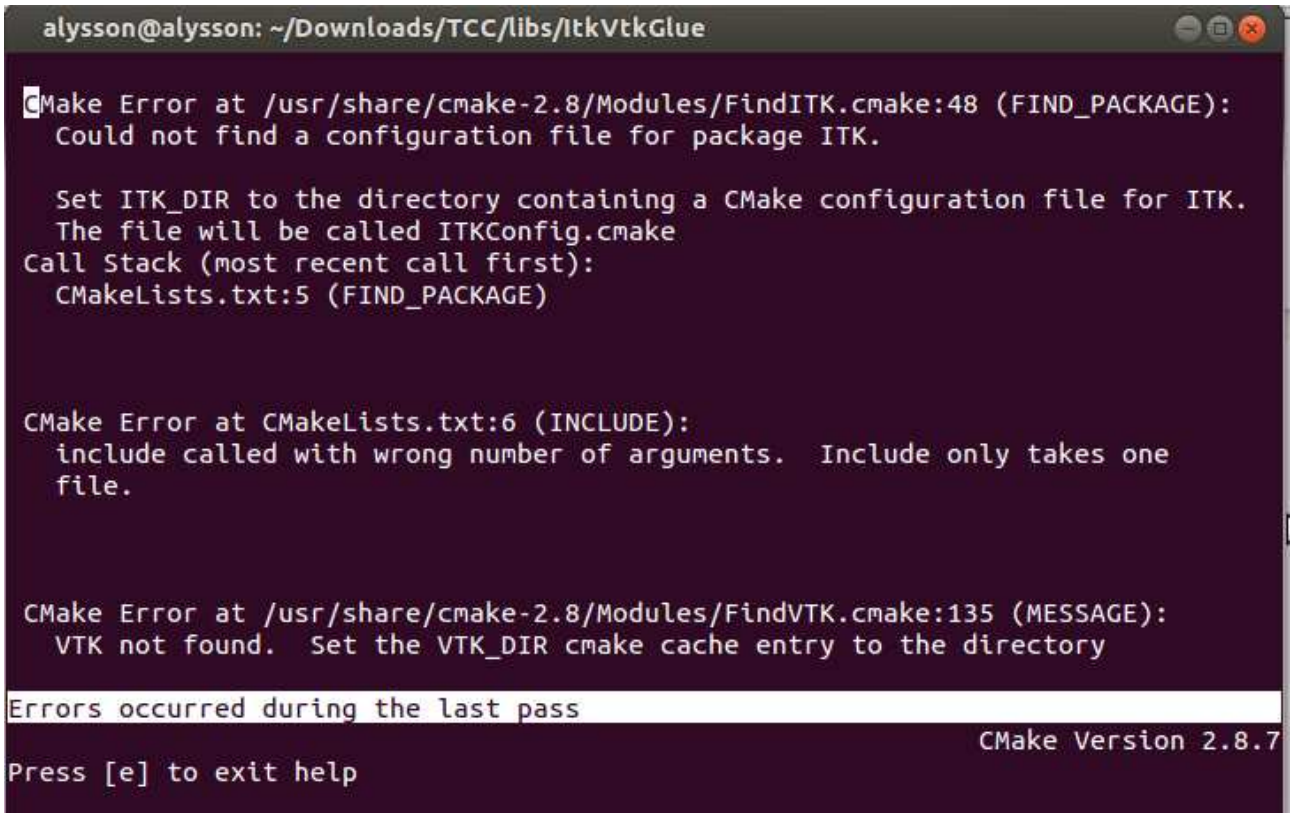


Figura 15 – Erro ao acessar a pré-configuração do ItkVtkGlue

Ao aparecer o erro aperte “e” e indique o diretório do VTK e do ITK, como apresentado na imagem abaixo.

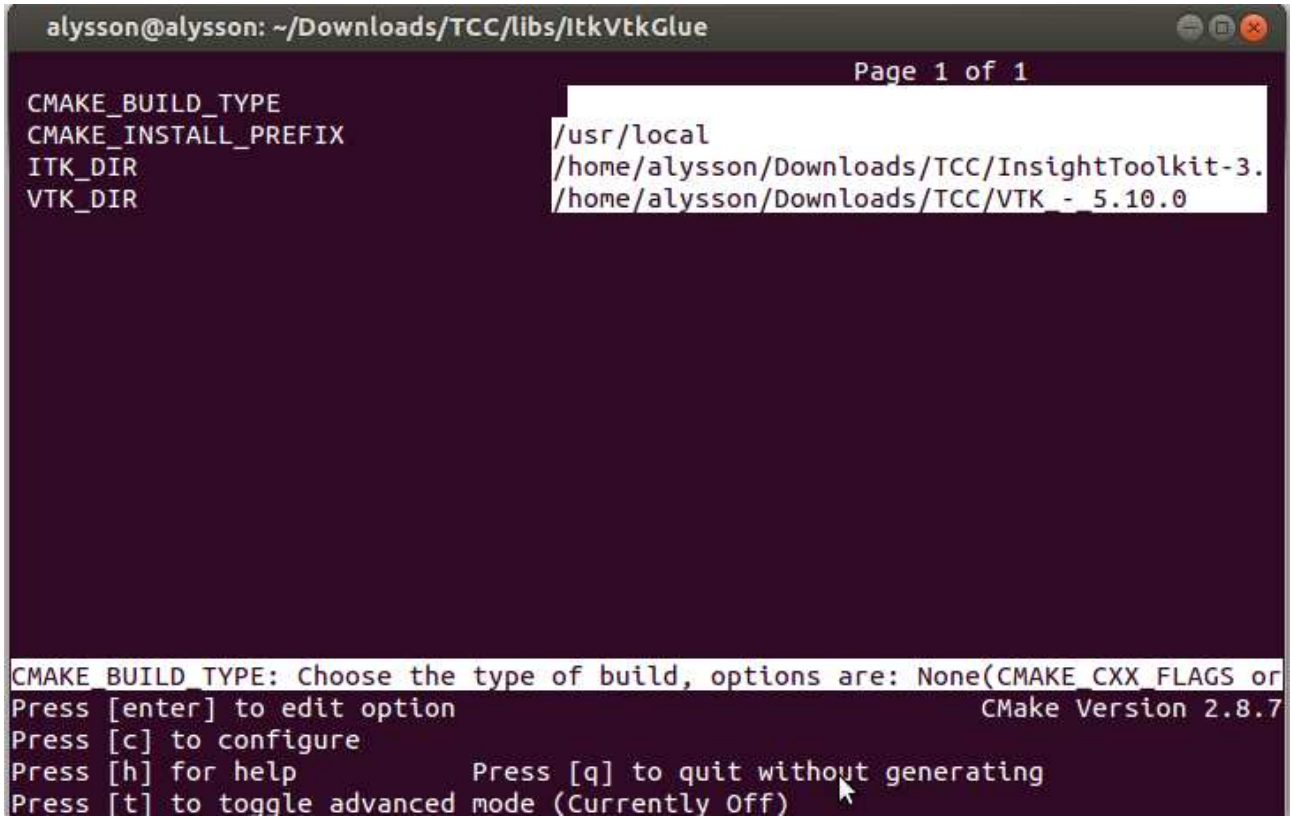


Figura 16 – Tela em que o usuário indica o diretório do ITK e do VTK para serem ligados pelo ItkVtkGlue

Aperte “c” e em seguida “g” para finalizar a configuração. Ao finalizar é só rodar o comando “make” ou “make -j <numero_de_nucleos_processador>”.

Figura 17 – Compilação do ItkVtkGlue

Ao finalizar essas etapas o computador está pronto para compilar a aplicação. Para isso, é só colocar o CD fornecido pela equipe e copiar a pasta para um diretório qualquer do seu computador.

Como na instalação das bibliotecas, primeiro abra o terminal e acesse o diretório da aplicação.

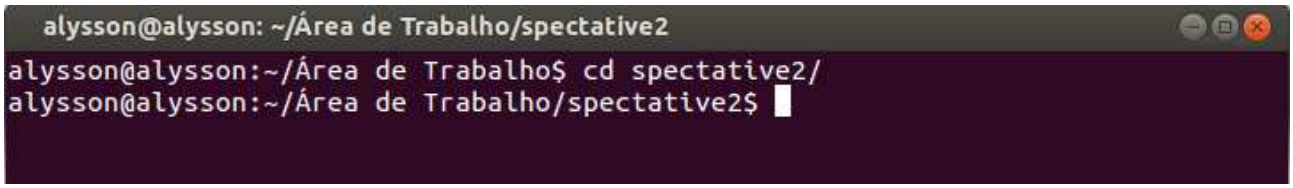


Figura 18 – Acessando o diretório da aplicação

Para gerar os arquivos de configuração, como nos casos anteriores, execute o comando “ccmake .”.

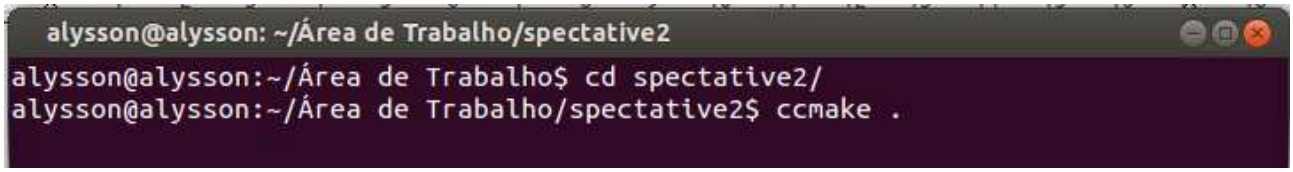


Figura 19 – Chamada da pré-configuração da aplicação

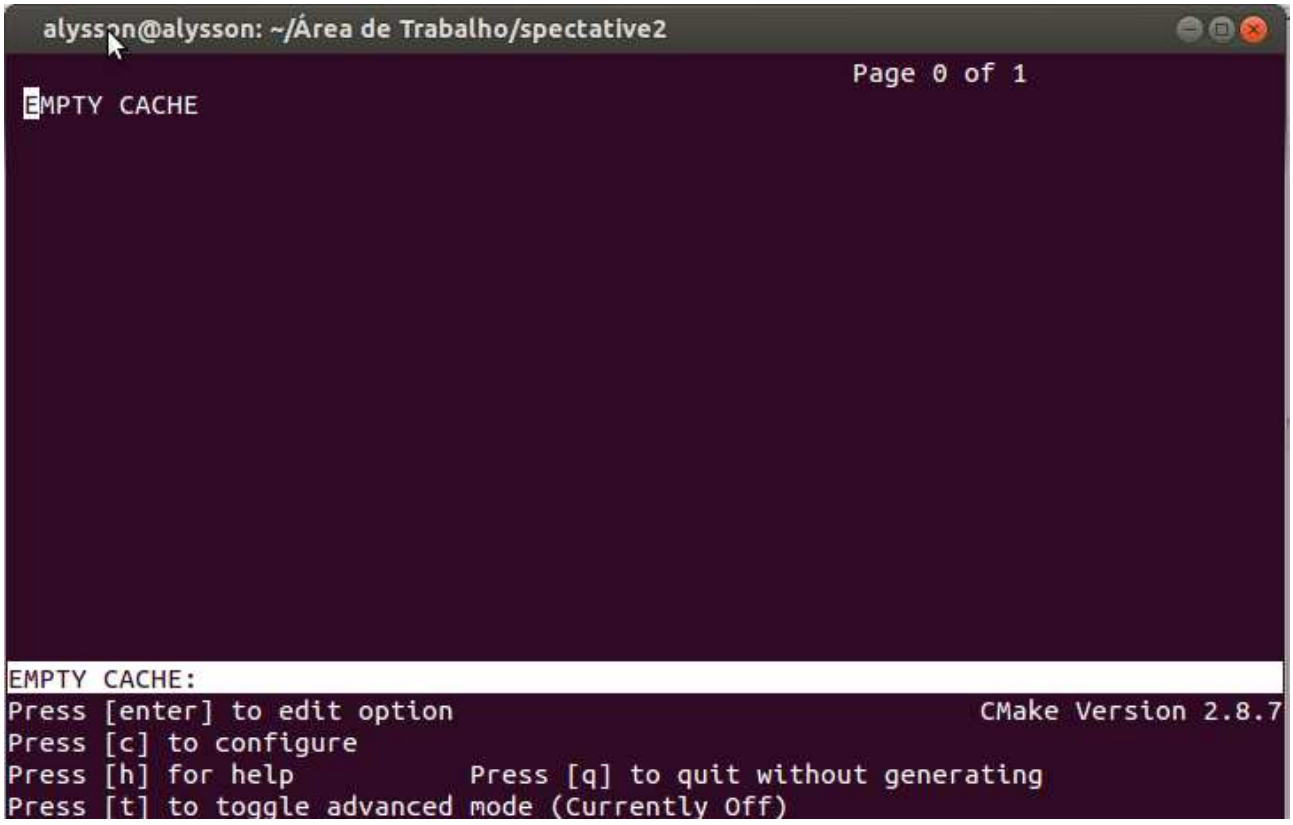


Figura 20 – Pré-configuração da aplicação

Pressione “c” para dar início ao processo.

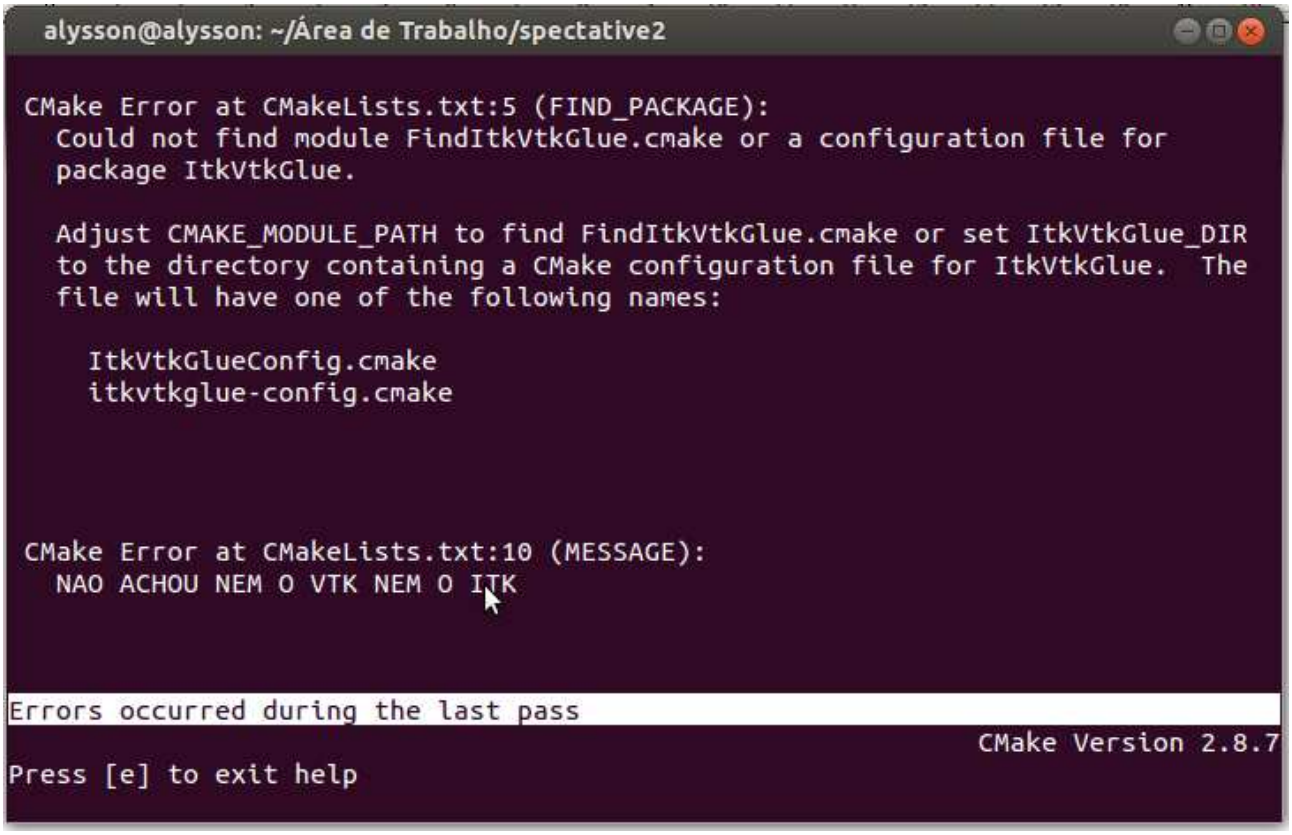


Figura 21 – Tela que emite a mensagem informando que o ItkVtkGlue não foi detectado

Em seguida irá aparecer a mensagem de erro informando o usuário que o ItkVtkGlue não foi localizado. Pressione a tecla “e” para prosseguir o processo.

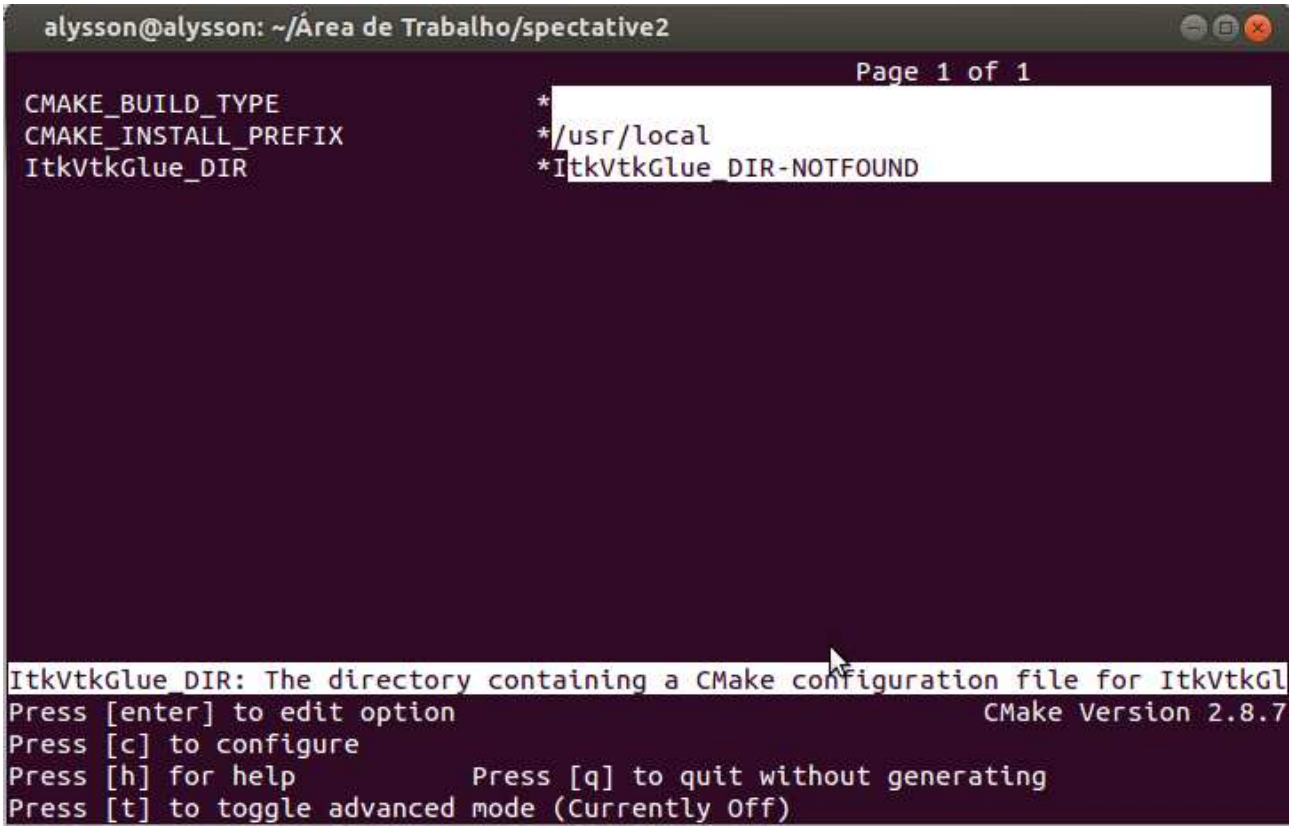


Figura 22 – Tela em que o usuário indica o diretório do ItkVtkGlue

Primeiro é necessário fornecer o caminho completo do ItkVtkGlue alterando o parâmetro “ItkVtkGlue_DIR” e apertar a tecla “c”.

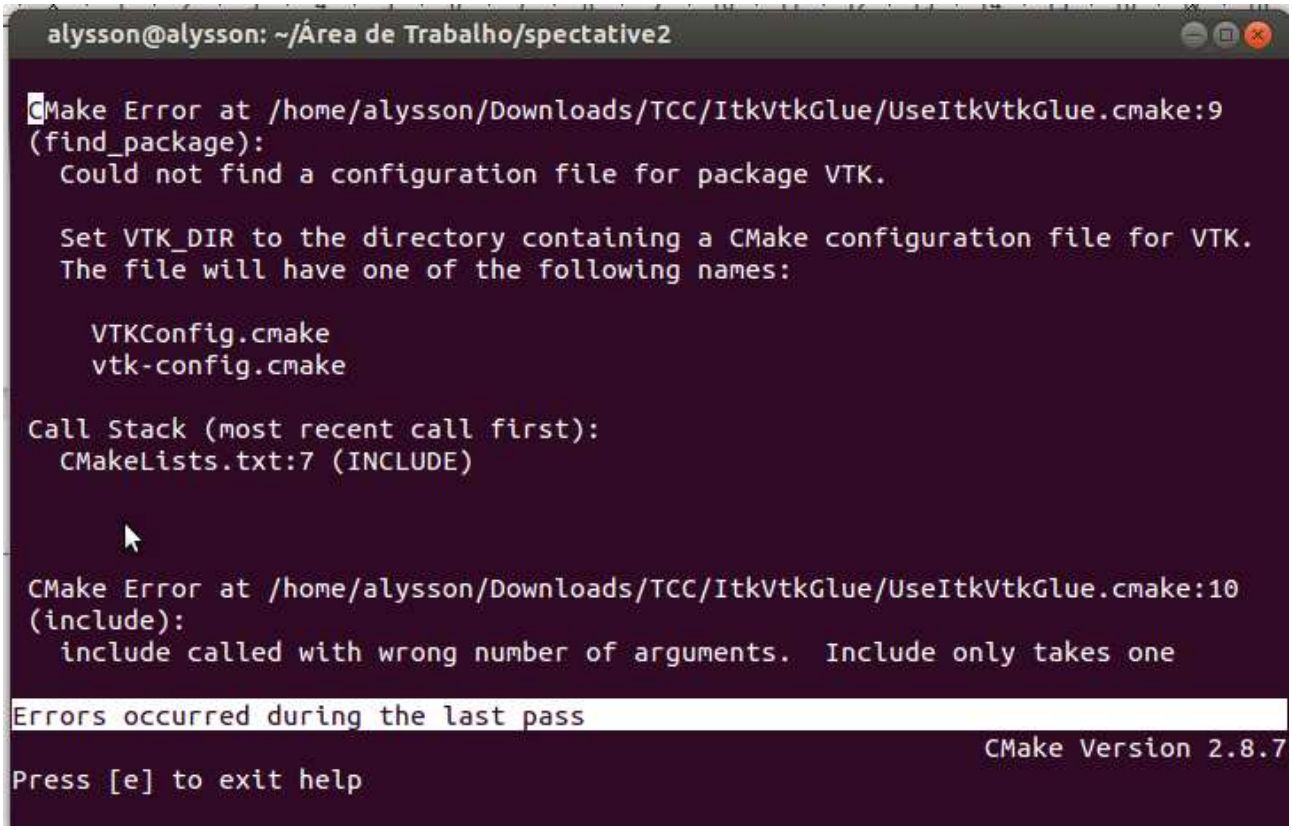


Figura 23 – Tela que indica que o diretório do VTK não foi encontrado

Novamente ocorreu um erro, pois o diretório do VTK ainda não foi indicado. Para configura-lo, aperte “e”.

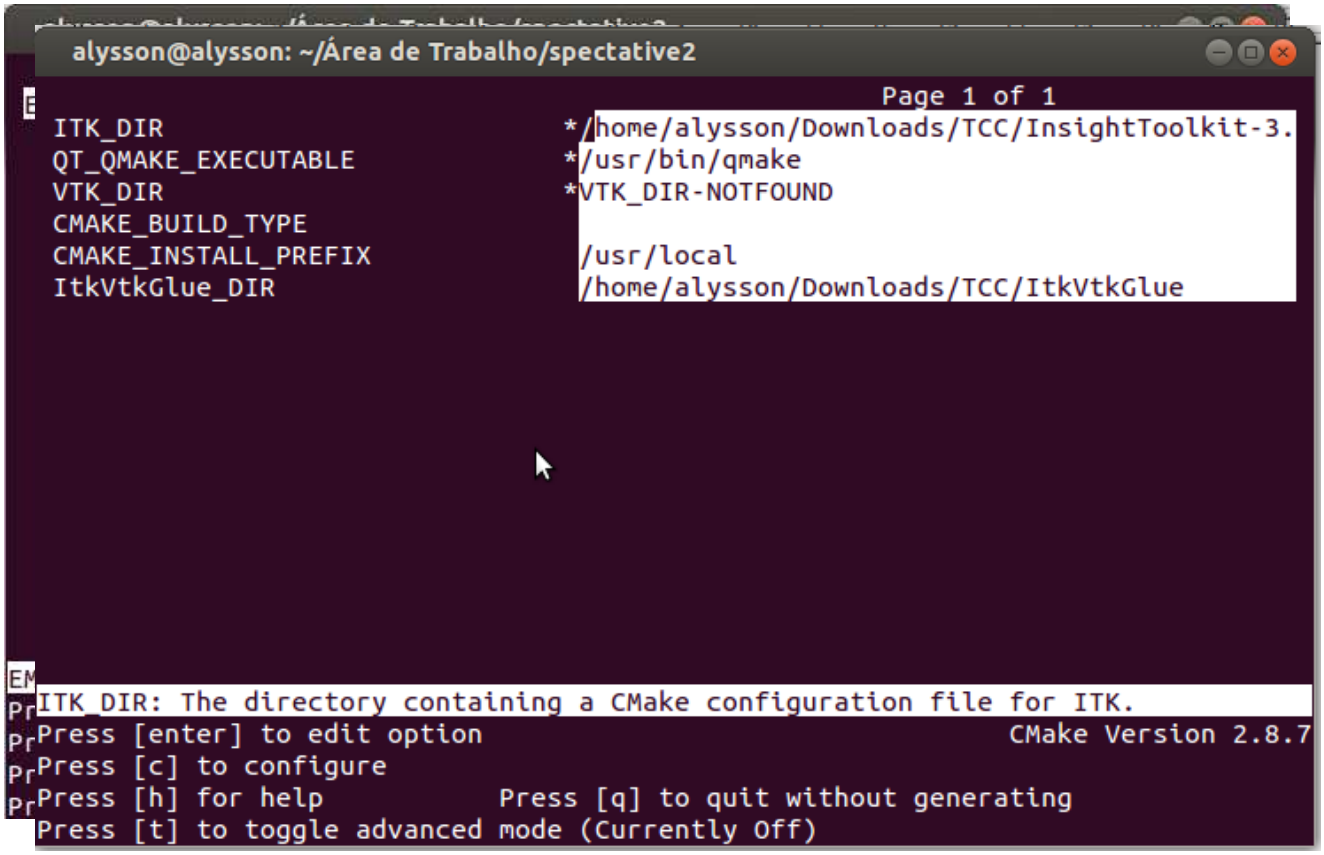


Figura 24 – Tela em que o usuário indica o diretório do VTK

Forneça o caminho completo do diretório do VTK, no item “VTK_DIR” e pressione “c”.

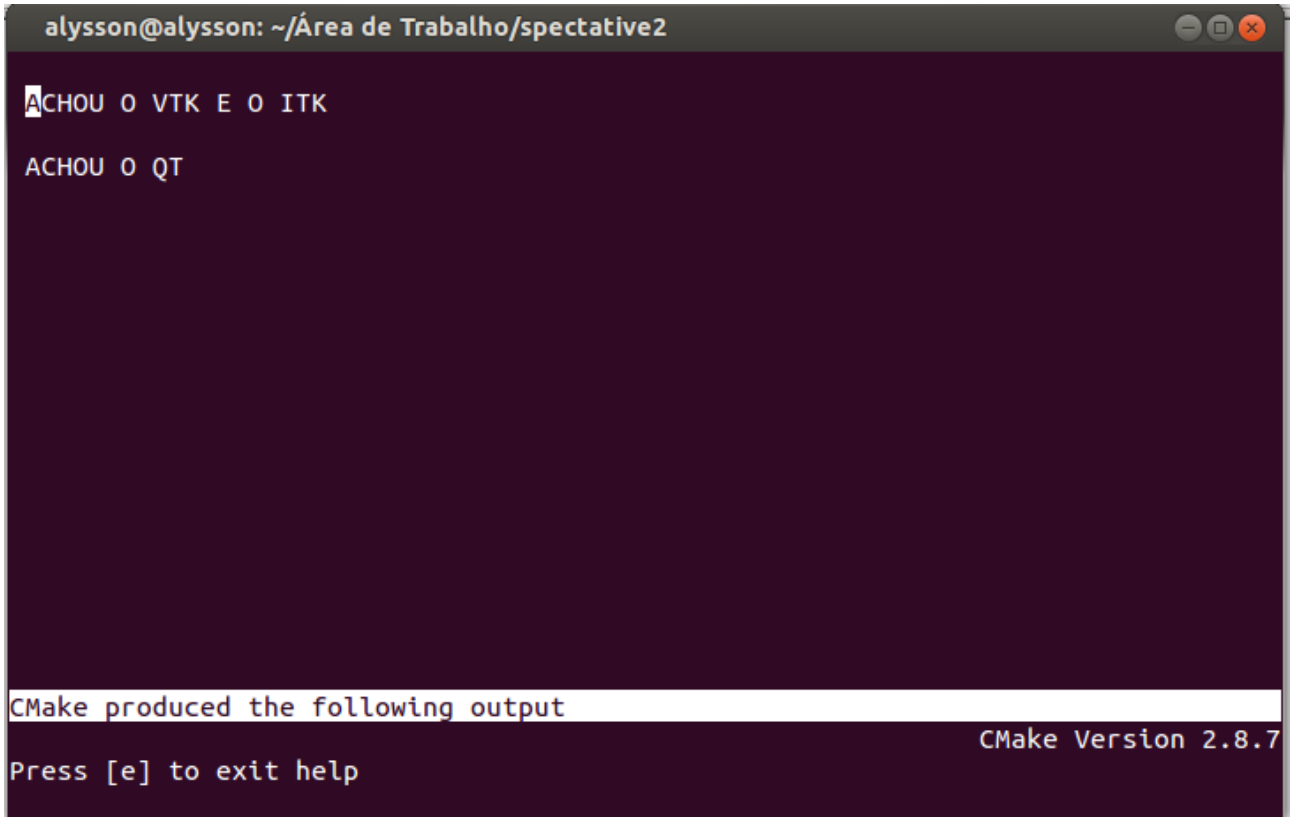


Figura 25 – Tela em que o usuário é informado que tanto o ITK quanto o ITK foram encontrados

Para dar continuidade ao processo aperte “e”.

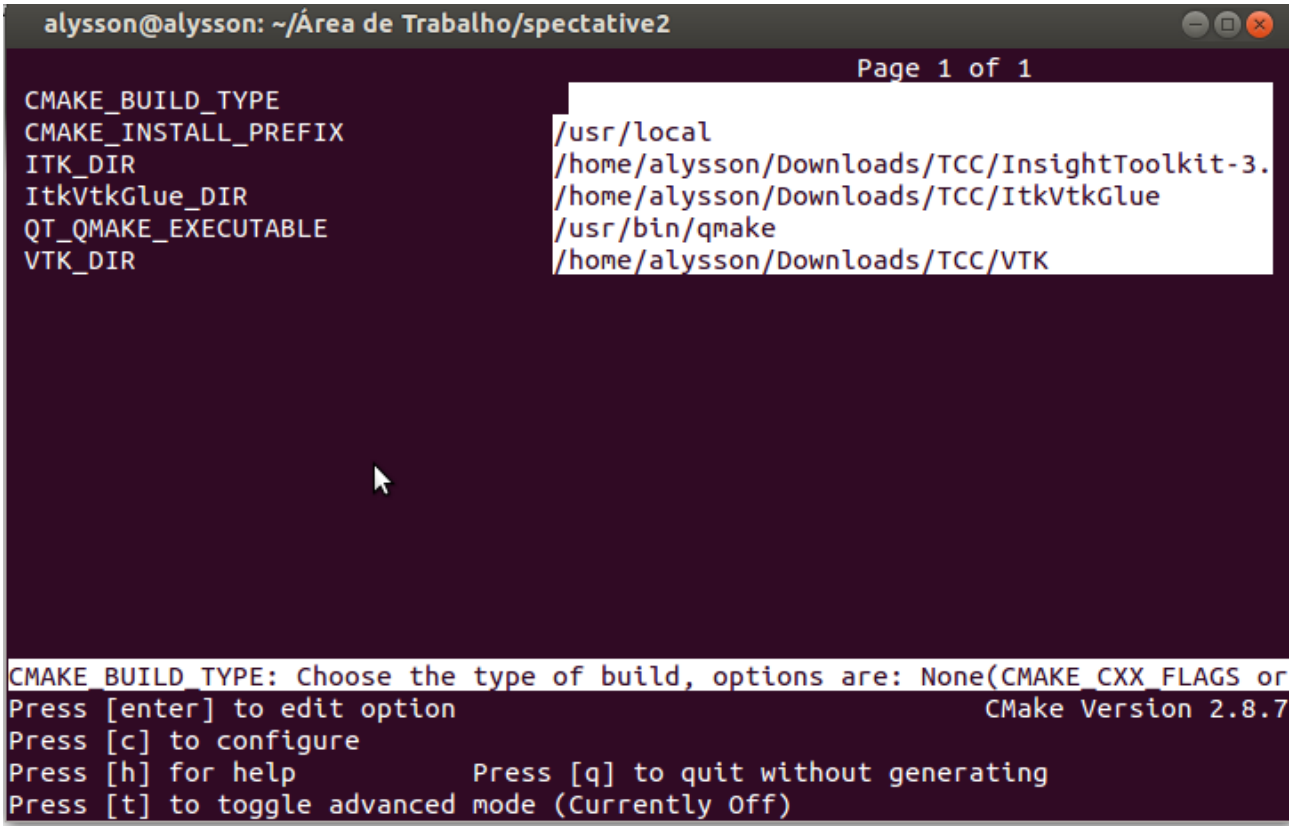


Figura 26 – Tela de finalização da configuração da aplicação

Como pode ser visto, todas as bibliotecas foram encontradas. Agora é só apertar “g” para finalizar a configuração.

Para gerar o executável do software basta executar o comando “make” ou “make -j <numero_nucleos_processador>”.

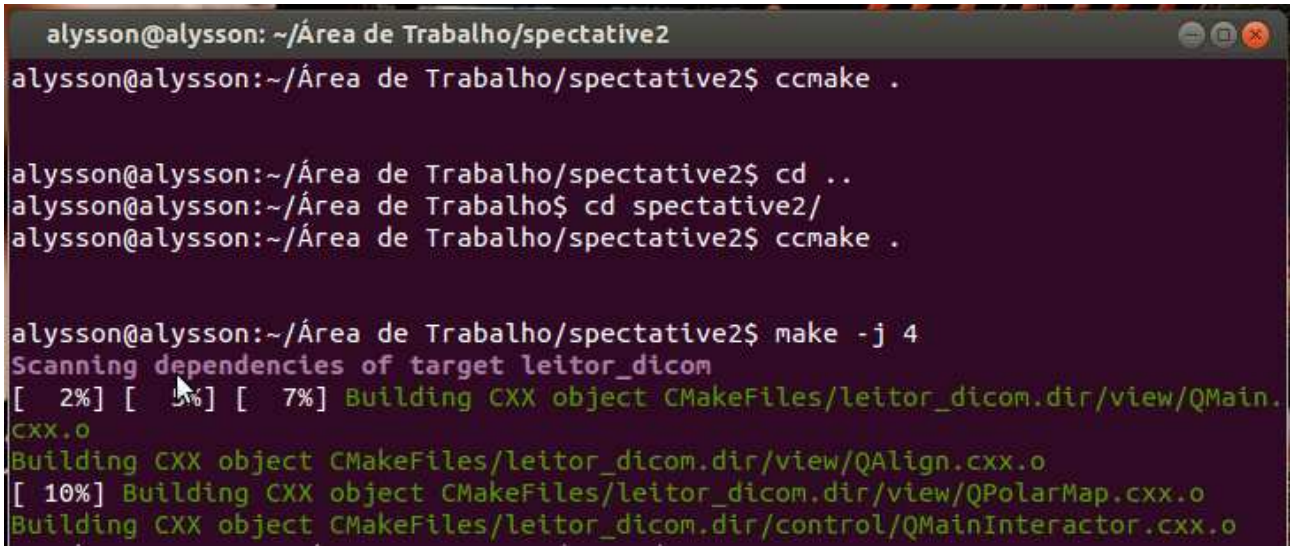


Figura 27 – Tela de compilação da aplicação

4.2 USO DO SOFTWARE

Para acessar o sistema, acesse a pasta em que o projeto foi compilado e execute o comando “./leitor_dicom” para acessar a aplicação.

Ao executar o programa, a interface da tela principal do sistema será apresentada (Figura 28).

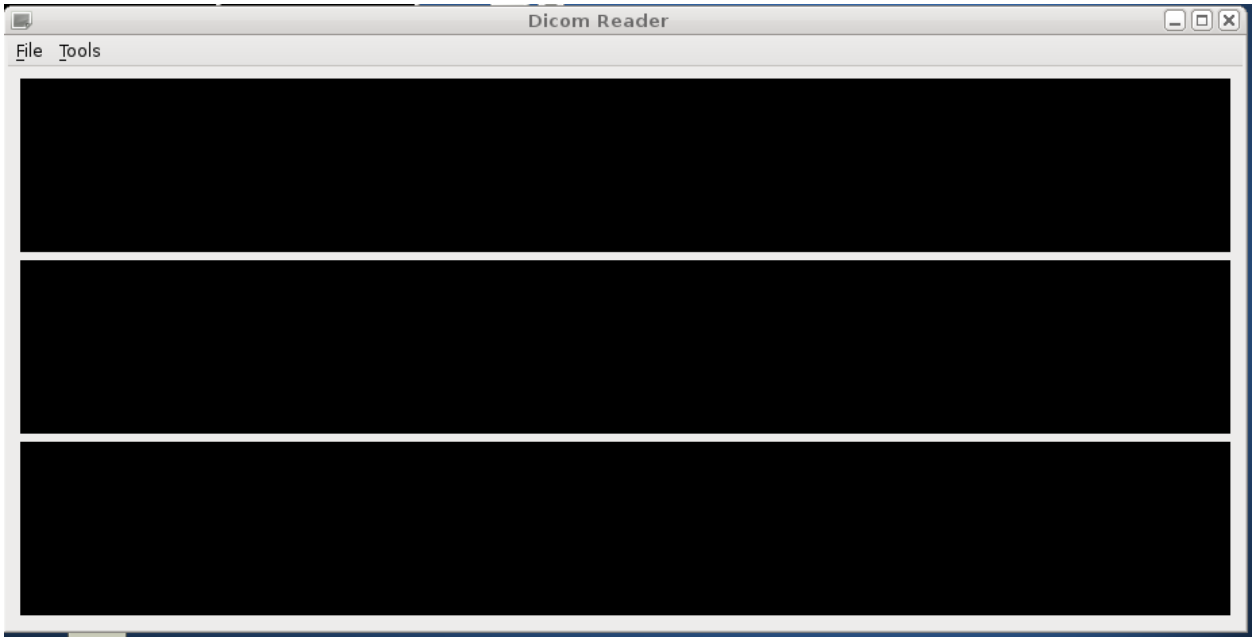


Figura 28 – Tela principal

O menu superior do sistema conta com dois itens: “*File*” e “*Tools*”. O item “*File*” possui sub-itens para abertura e fechamento de duas imagens (“*First Image*” e “*Second Image*”) e a opção “*Exit*”, para sair do programa, enquanto o item “*Tools*” dispõe dos sub-itens “*Image Align*” e “*Polar Mapping*”.

Ao selecionar a opção de abrir “*First Image*” ou “*Second Image*”, a tela de escolha de arquivo será apresentada (Figura 29). O usuário poderá então selecionar um arquivo do tipo DICOM para carregar no sistema.

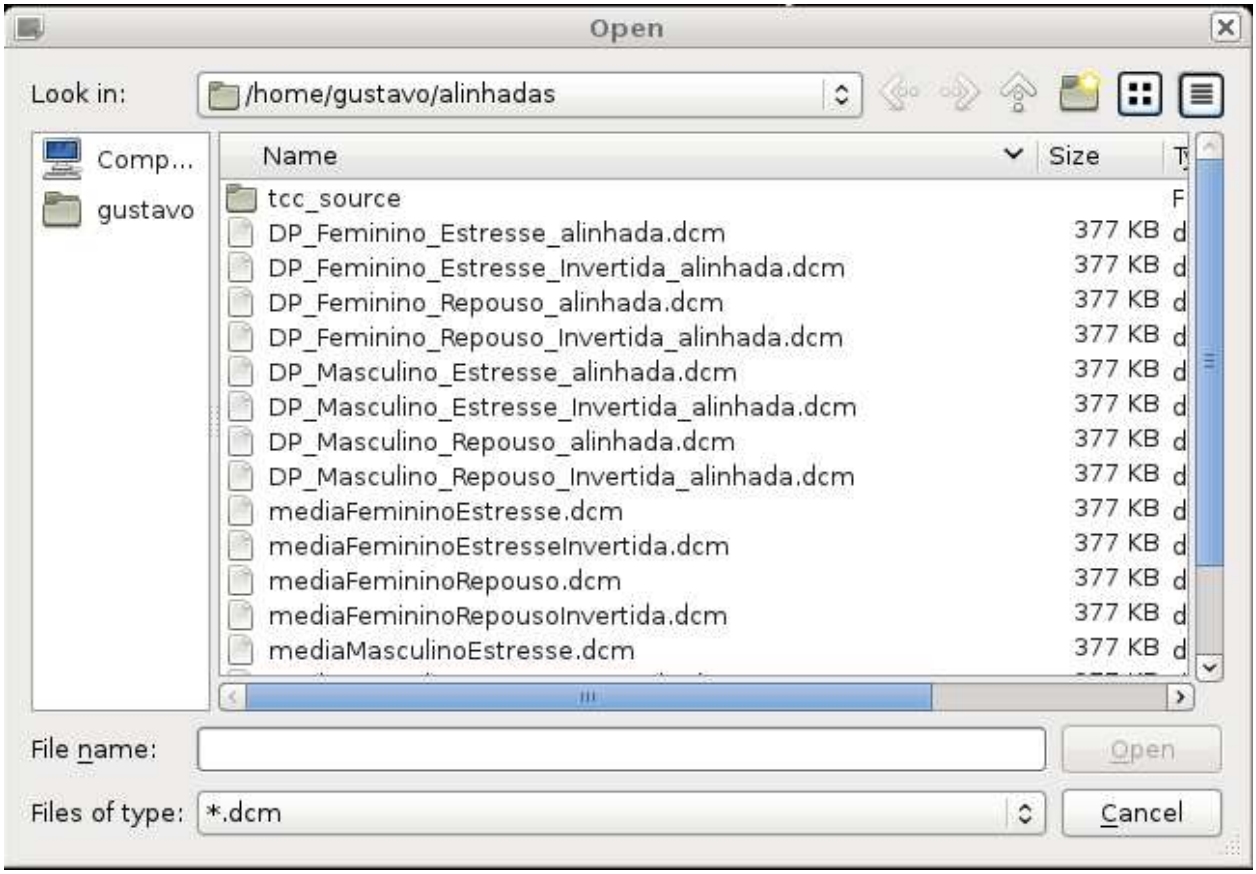


Figura 29 – Abertura de arquivo

Caso o usuário selecione um arquivo válido de imagem DICOM, a tela principal do sistema será atualizada de forma a renderizá-la (Figura 30). Como pode ser observado, junto com o preenchimento das “tiras” de imagens, até então vazias, ficam disponíveis o *Slider* de intensidade da imagem e um *SpinBox* diretamente à sua direita que tem o seu valor, os quais atualizam os componentes de visualização em tempo real a medida que o usuário modifica seus valores. Mais à direita, são revelados os *SpinBoxes* com *Label* “*Min*” e “*Max*”, os quais representam o *range* do *Slider* de intensidade, ou seja, o alcance do mesmo. Por último, temos a *ComboBox* relacionada às pseudo-cores, das quais falaremos posteriormente.

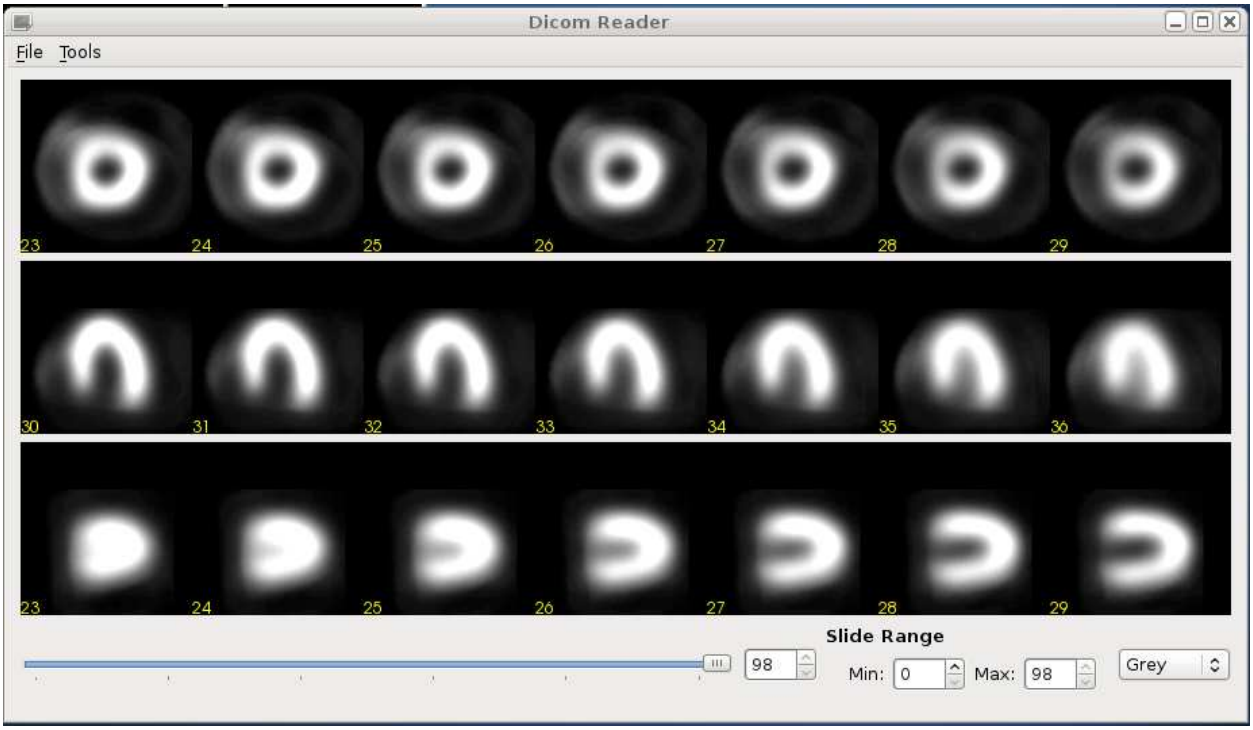


Figura 30 – Tela principal com uma imagem aberta

Tendo duas imagens abertas, a tela principal passa a mostrar ambas, uma sobre a outra nos componentes de visualização. Os componentes relacionados à intensidade dessa segunda imagem também são apresentados, assim como o *ComboBox* de pseudo-cores (Figura 31).

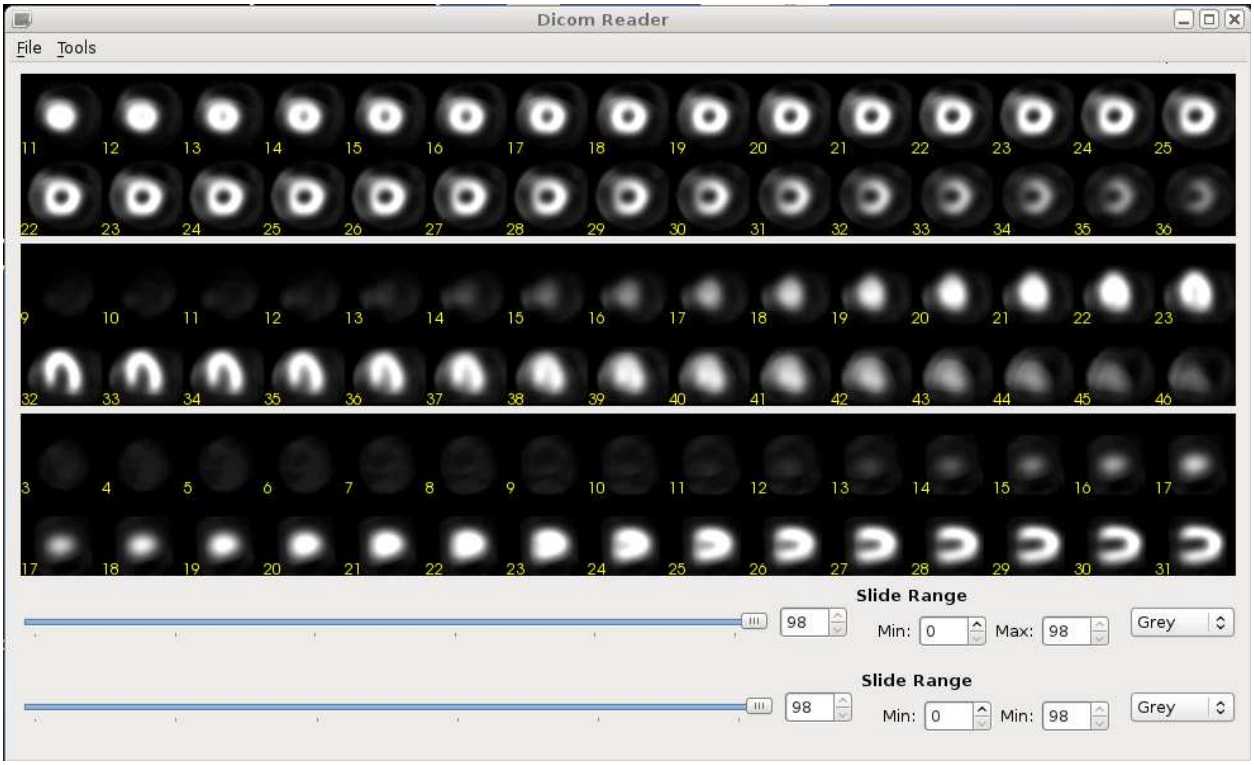


Figura 31 – Tela principal com duas imagens abertas

As pseudo-cores podem ser selecionadas nas suas respectivas *ComboBoxes* de forma independente. Isso significa que o usuário poderá escolher uma cor diferente para cada imagem aberta (Figuras 32 e 33).

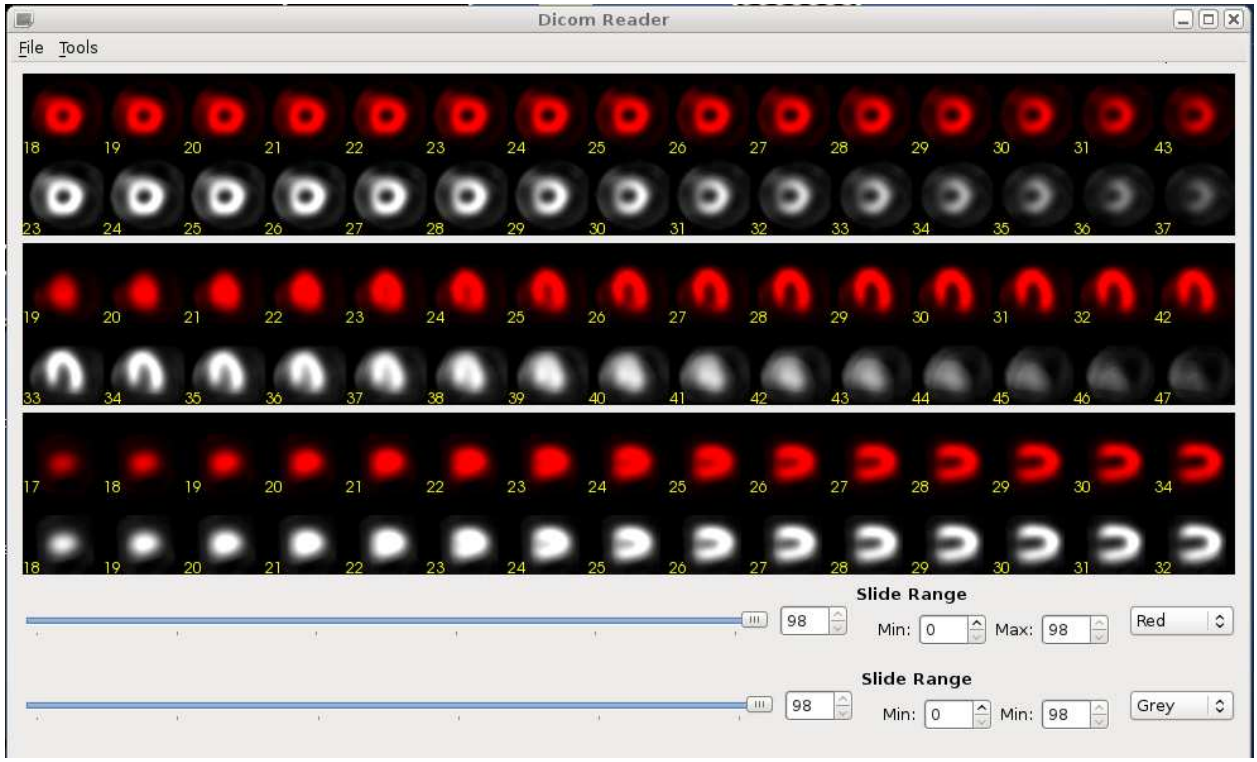


Figura 32 – Apenas uma das imagens tem o filtro de cores aplicado

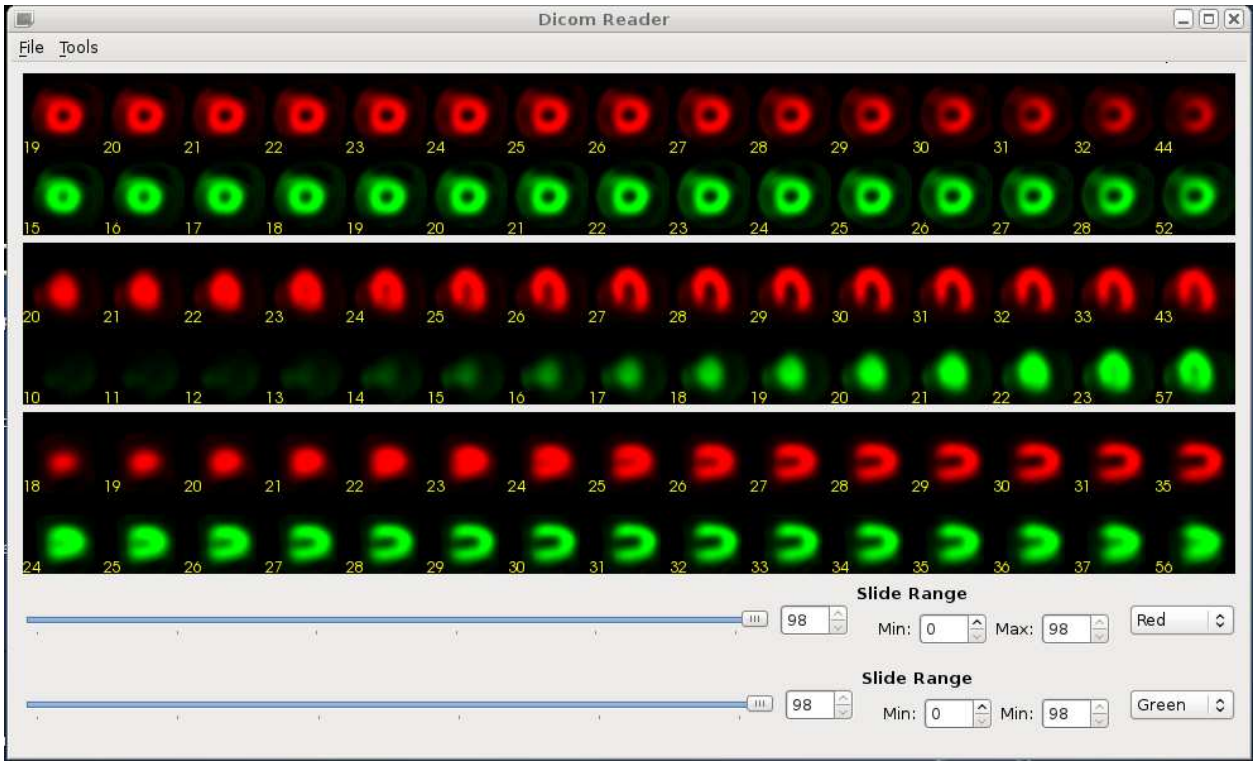


Figura 33 – Ambas as imagens possuem um filtro de cor aplicado, sendo uma de cada cor

Ao fechar a janela do sistema, o usuário receberá um pedido de confirmação (Figura 34).

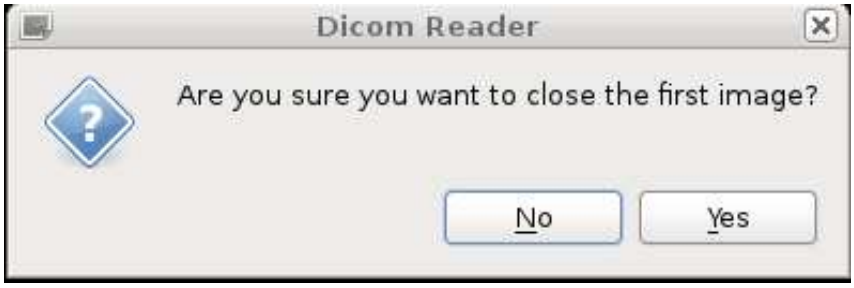


Figura 34 – Tela de pedido de confirmação do término do programa

Ao clicar no sub-item “*Image Align*” do menu “*Tools*”, é apresentado ao usuário uma nova tela (Figura 35). Essa interface mostra um menu “*File*” com as opções “*AlignImages*”, “*Save First Image*” e “*Save Second Image*”.

Esta tela possui três janelas de renderização de imagens, assim como três conjuntos de controles de contraste.

A janela superior exibe a imagem modelo, já alinhada pelo especialista. As imagens de entrada serão comparadas com ela.

A janela central mostra a imagem que foi aberta na janela de visualização com “*First Image*” e a janela inferior mostra a “*Second Image*”.

Caso o usuário não tenha aberto uma imagem na janela anterior ou tenha aberto e fechado, ela não será mostrada nessa janela e os seus respectivos controles serão desabilitados.

O menu *File* possui três itens: *Align Images, Save First Image* e *Save Second Image*. O primeiro item chama os métodos que irão alinhar as imagens abertas com a imagem modelo. Os outros dois abrirão uma janela para escolher onde o usuário deseja salvar a respectiva imagem de resultado.

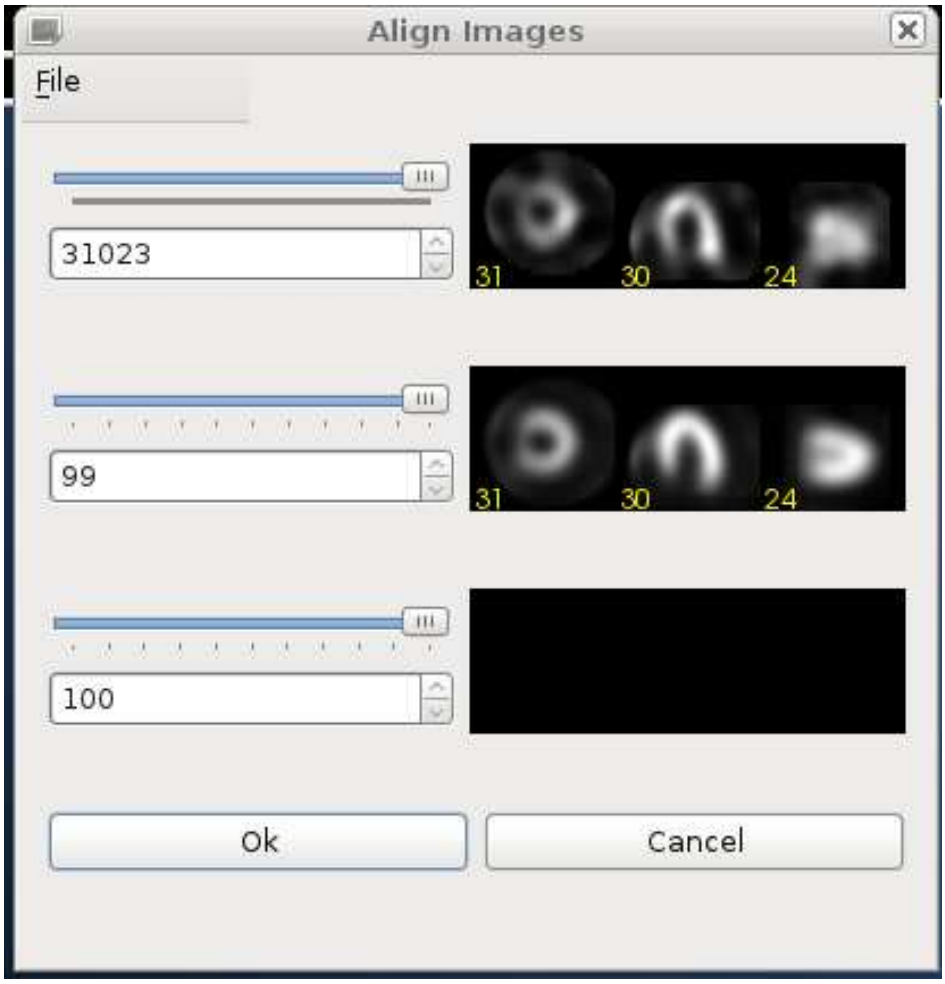


Figura 35 – Tela de Alinhamento (com apenas uma imagem carregada)

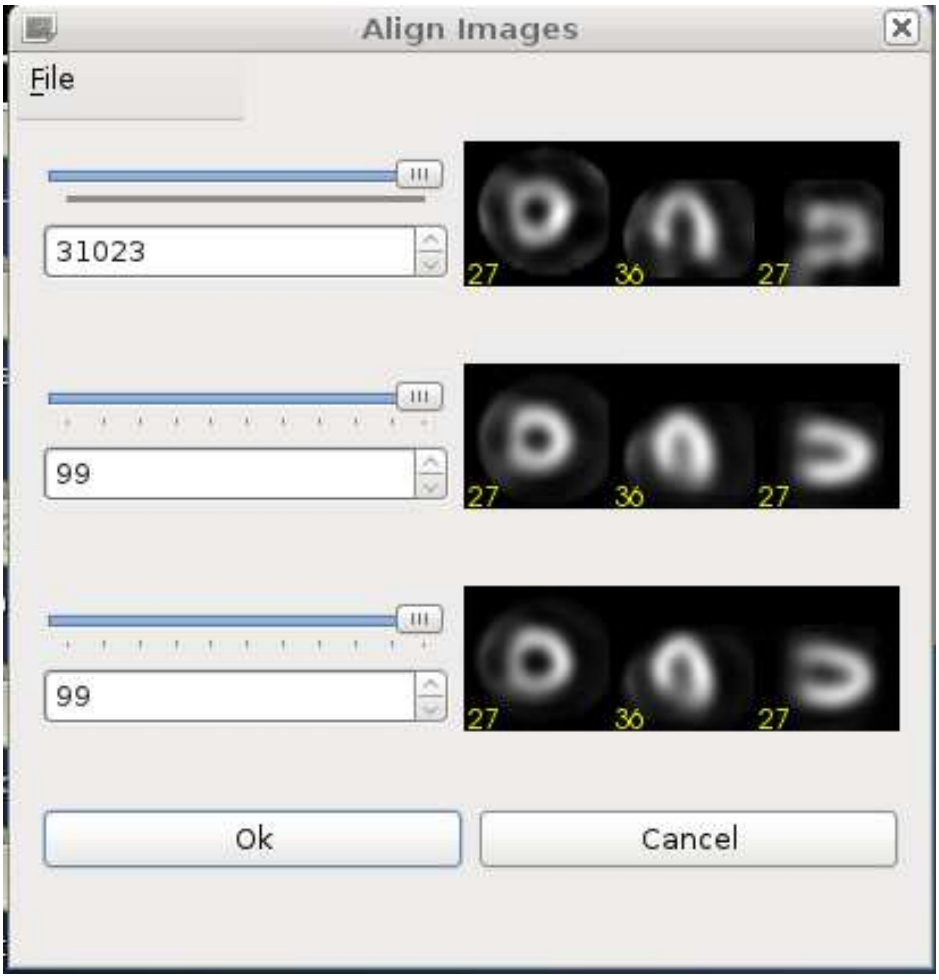


Figura 36 – Tela de Alinhamento (com duas imagens carregadas)

Da tela principal, ainda no menu “*Tools*”, é possível acessar, através do sub-item “*Polar Mapping*”, a tela onde será gerado o mapa polar com um imagem (Figura 37) ou com duas (Figura 38). No caso do mapa polar ser gerado com apenas uma imagem, ele os outros dois componentes ficarão vazios. Já no caso de duas imagens estarem carregadas, dois mapas polares serão gerados nos componentes superiores e a subtração matemática dos mesmos será obtida no terceiro componente (logo abaixo).

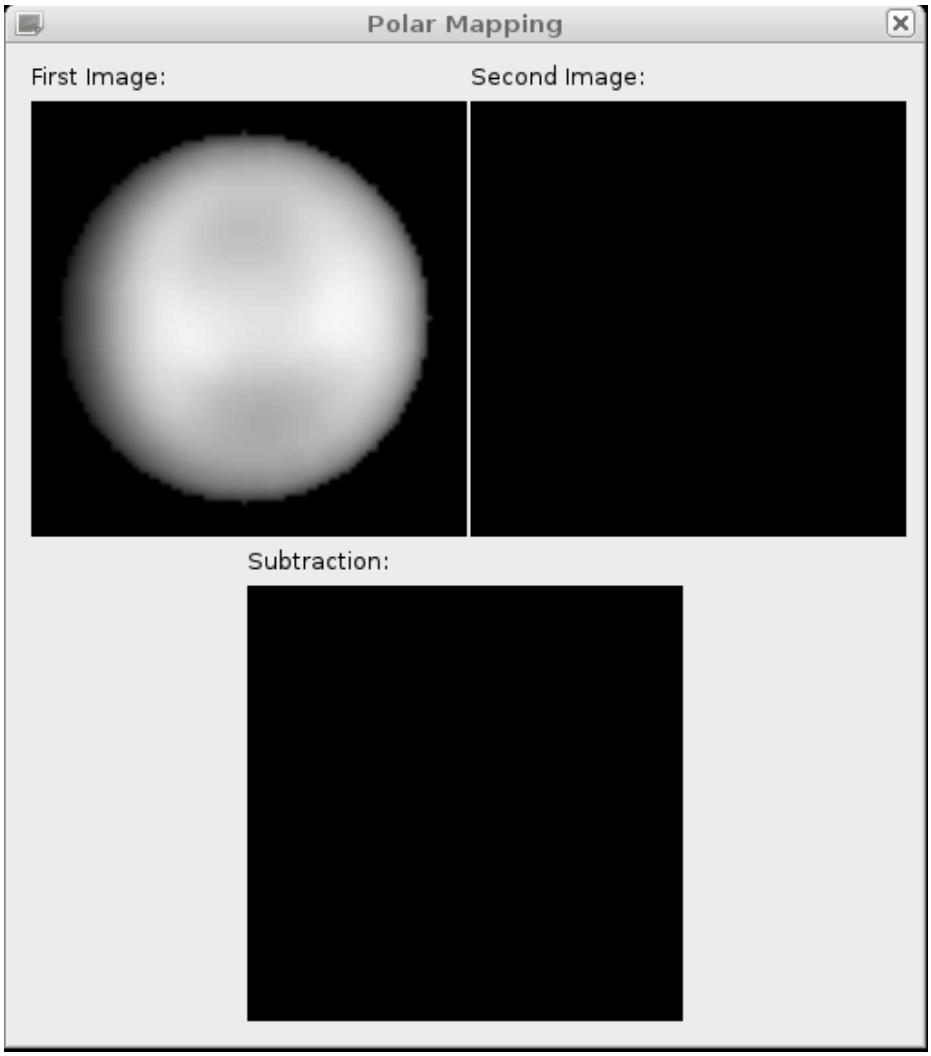


Figura 37 – Tela de Geração de Mapa Polar (com uma imagem carregada)

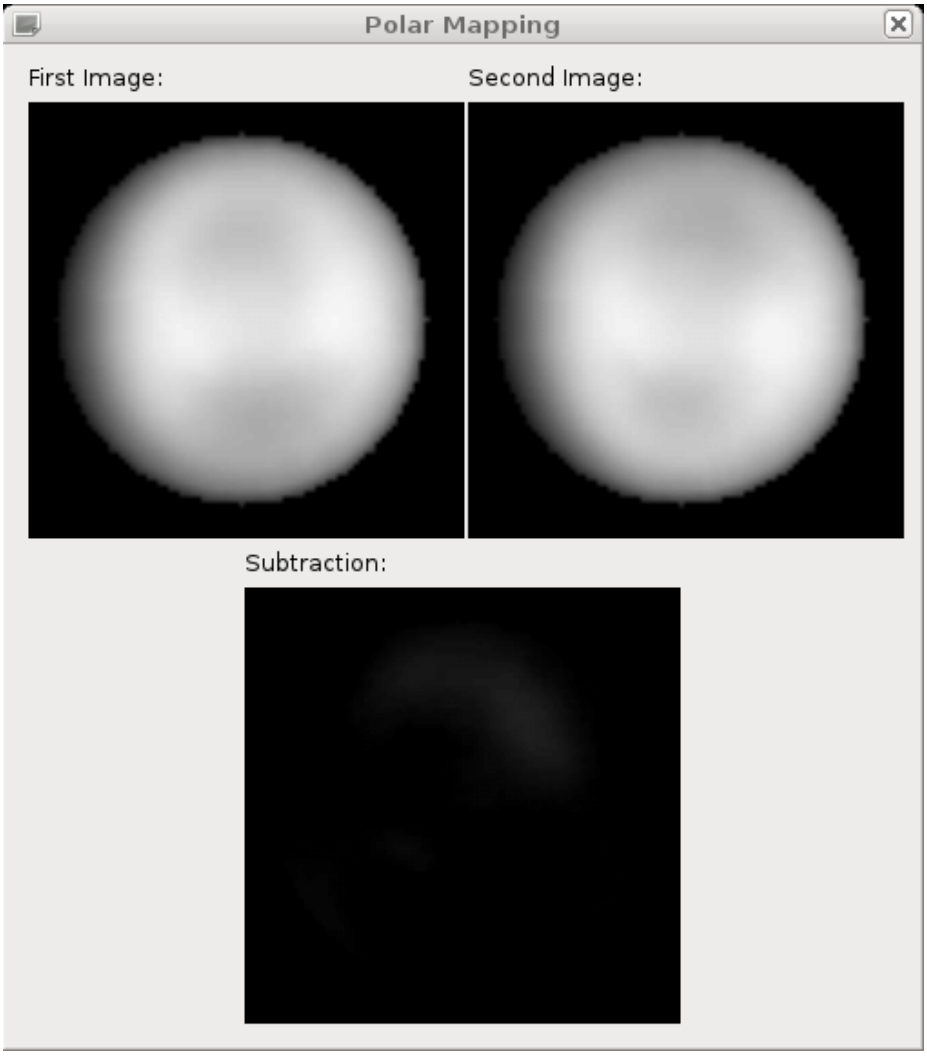


Figura 38 – Tela de Geração de Mapa Polar (com duas imagens)

5 CONSIDERAÇÕES FINAIS

Com o SPECTative, pôde-se alcançar os resultados esperados: uma alternativa gratuita de Auxílio ao Diagnóstico em Cardiopatias, com técnicas avançadas de processamento de imagens tais quais alinhamento 3D, pseudo-coloração e mapa polar (ou “Bull’s Eye”). Ferramenta essa que poderá ser utilizada por médicos para substituir concorrentes proprietários do mercado que exigem equipamentos proprietários para serem executados o quepode encarecer os exames cardiovasculares e diagnóstico de cardiopatias. O software foi desenvolvido de forma a facilitar seu entendimento do código e arquitetura por futuros interessados em dar continuidade ao projeto.

A aplicação tem como outra vantagem o fato de agregar em si várias funcionalidades muitas vezes não encontradas em um único lugar, tornando o trabalho do médico muito mais oneroso, tendo em vista que o mesmo teria que tratar as imagens, alinhá-las, ajustar intensidade e gerar o mapa polar através de vários diferentes softwares. Consideramos que, para trabalhos futuros, pode-se averiguar os seguintes pontos:

• Tornar o projeto multiplataforma. O projeto foi desenvolvido com o objetivo de ser funcional no maior número possível de sistemas operacionais. E, de fato, ele possui tal potencial por utilizar apenas bibliotecas com suporte a, pelo menos, Windows e Linux. No entanto, apenas uma versão Linux foi gerada e testada. Fica em aberto a tarefa de torná-lo totalmente portável.

• Melhor precisão matemática. Apesar de funcional e suficientemente eficaz, ainda há o que melhorar nos cálculos matemáticos responsáveis pela geração do Mapa Polar, da colorização e do ajuste de intensidade das renderizações.

• Gravar alterações de alinhamento e resultado do Mapa Polar. Seria realmente interessante ter essas funcionalidades, as quais realmente só não foram implementadas no projeto por não se encaixarem no cronograma. Tais itens seriam de grande ajuda para os médicos para facilitar a portabilidade dos resultados de exames.

REFERÊNCIAS

BARROS, R. C; **Desenvolvimento de ferramenta para análise quantitativa de imagens de cintilografia de perfusão miocárdica através de técnicas de processamento de imagens.** Pelotas: UFPel 2007.

BOOCH, G.; RUMBAUGH, J. JACOBSON, I. **UML:** guia do usuário. Rio de Janeiro: Elsevier, 2005.

BLANCHETTE, J.; SUMMERFIELD, M. **C++ GUI programming with Qt 4-2nd Ed.** Massachusetts, 2008.

CMAKE: Cross Plataform Make. Disponível em http://www.cmake.org/cmake/project/about.html. Acesso em 21 de julho de 2012.

DAVIS, S. R. **C++ For Drummies.** Indiana: Wiley Publishing, 2004.

GARCIA, E. V.; VAN TRAIN, Kenneth F.; MADDAHI, Jamshid; PRIGENT, Florence; FRIEDMAN, John; AREEDA, Joseph; WAXMAN, Alan; BERMAN, Daniel S. **Quantification of Rotational Thallium-201 Myocardial Tomography**. Journal of Nuclear Medicine, v. 26, n. 1, p. 17-26. Janeiro, 1985.

LIUMING, F; NING, L; WEIHUA, D. **Application of Pseudo Color Enhancement Method To Geotechnical CT Image Analysis**. Chinese Journal of Rock Mechanics and Engineering, V.23, p2257-2257. 2004.

MARTINS, J. C. C. **Técnicas para gerenciamento de projetos de software.** Rio de Janeiro: Brasport, 2007.

MELO, E. C. P.; TRAVASSOS, C.; CARVALHO, M. S. **Qualidade dos dados sobre óbitos por infarto agudo do miocárdio**. Revista de Saúde Pública. V.38, p385-391. 2004.

MILDENBERGER, P.; EICHELBERG, M.; MARTIN, E.. **Introduction to the DICOM standard**. European Radiology, V. 12, p920-927. 2002.

MONTEIRO, T. C; **Pontos de Caso de Uso técnicos (TUCP):** Uma extensão da UCP. Fortaleza: UNIFOR, 2005.

MULTICENTER Postinfarction Research Group. **Risk stratification and survival after myocardial infarction**. N. Eng. J. Med, p.309-331, 1983.

MUSTRA, M.; DELAC, K.; GRGIC, M. **Overview of the DICOM Standard**. Zagreb: 50th International Symposium ELMAR-2008, 2008.

OLIVEIRA, L. F.; ZANCHET, B. A.; BARROS, Rodrigo C.; GOMES, V.V; FUJII, S.Y; VORTMANN, C.H; PATZER, G.P.. **Utilização das Bibliotecas VTK e ITK no processamento de Imagens Médicas**. Porto Alegre: 8º Fórum Internacional de Software Livre - VIII Workshop sobre Software Livre, 2007.

PATZER, G. P; **Método automático para criação de mapas polares baseado em alinhamento de imagens.** Santo André: UFABC, 2011.

PERINI, A. P.; OLIVEIRA, L. F.; SIQUEIRA, R. B.; MACHADO, H. R.; CARNEIRO, A. A. O. **Neuronavegador cirúrgico guiado por IRM, baseado num transdutor de posição magnético.** Revista Brasileira de Engenharia Biomédica, 2009.

PRESSMAN, R. S. **Engenharia de Software.** São Paulo: Makron Books do Brasil Editoria Ltda, 1995.

Processing API: A Technical Report on ITK – the Insight Toolkit. Amsterdam: Studies in Health Technology and Informatics, vol. 85, 2010.

SARMENTO-LEITE, Rogério; KREPSKY, Ana Maria; GOTTSCHALL, Carlos A. M. **Acute myocardial infarction**: one century of history. Arquivos Brasileiros de Cardiologia, V.77, p602-610. 2001.

SCHILDT, H. **C++:** The Complete Reference Third Edition. McGraw-Hill, 1998.

YOO, T. S.; ACKERMAN, M. J.; SCHROEDER, W.; CHALANA, V.; AYLWARD, S.; METAXAS, D.; WHITAKER, R. **Engineering and Algorithm Deseign for na Image** MARTINS, J. C. C. **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML.** Rio de Janeiro: Brasport Livros e Multimidia Ltda, 2007.

APÊNDICE

1 BACKLOG DO PRODUTO

Para definição das atividades foi feito o uso do backlog do produto para servir de complemento ao gráfico de GANTT, detalhando as atividades a serem desempenhadas por cada integrante da equipe.

SPRINT BACKLOG			
SPRINT 1 (Duração: 1 dia) - Estória Técnica (parte 1)			
Nome	**Item**	**Descrição Detalhada**	**Responsável**
Levantamento de Requisitos	Entrevista com o cliente	1) Elaboração de uma relação de perguntas; 2) Realizar a entrevista com o cliente para obter o propósito do produto e obter informações para dar início ao desenvolvimento.	Alysson
SPRINT 2 (Duração: 26 dias) - Estória Técnica (parte 2)			
Nome	**Item**	**Descrição Detalhada**	**Responsável**
Definição do backlog inicial do produto	Definir a primeira visão obtida a respeito do sistema	1) Análise dos dados obtidos com a primeira fase de levantamento de requisitos e definir os módulos que irão compor o sistema final; 2) Dividir os requisitos em pequenas atividades.	Alysson
Elaboração do cronograma	Definição das datas	Criar um cronograma com as atividades a serem desenvolvidas, os responsáveis e a data de início e fim.	Alysson
Escolha da tecnologia	Escolha da tecnologia a ser empregada	Fazer pesquisa a fim de encontrar as melhores tecnologias a serem empregadas no desenvolvimento do produto	Alexandre, Alysson e Gustavo
Elaboração de esboço das telas	Desenho inicial das futuras telas do produto	Fazer desenhos simples e explicativos do que cada tela do produto conterá e suas funcionalidades (uma prévia de documentação).	Alexandre, Alysson
Ambiente de Desenvolvimento	Preparação do Ambiente de Desenvolvimento	Instalação do CMake e das bibliotecas QT, VTK e ITK	Alexandre, Alysson e Gustavo
Familiarização com as técnicas	Estudo detalhado da parte técnica e teórica	Estudo das técnicas de processamento de imagens associadas aos recursos oferecidos pelas tecnologias e de acordo com as necessidades listadas na primeira análise.	Alexandre, Alysson e Gustavo
Plano de riscos	Criação do plano de riscos	1) Estudo teórico sobre planejamento de riscos; 2) Lista-los; 3) Definição de prioridade; 4) Criação de plano de contingência	Alysson
SPRINT 3 (Duração: 31 dias)			

Nome	Item	Descrição Detalhada	Responsável
Elaboração do Menu Principal	Criação do menu que estará presente na tela principal	Conter os itens: "File" (com sub-itens: Open: "First Image", e "Second Image", Close: “First Image”, “Second Image” e Exit), “Tools” (com sub-itens: Image Aligner e PolarMapping)	Alexandre
Interface de Visualização Tela Principal	Criação da Interface de Visualização da Tela Principal	A interface será composta por: 1) Três componentes de visualização, cada um com três sequências horizontais de 7 imagens com o tamanho 64x64 cada. Em uma sequência será mostrada a imagem na perspectiva X, outra Y e a última Z do respectivo componente. O primeiro e o segundo componente farão a exibição das imagens que o usuário irá selecionar e o terceiro mostrará a imagem após o processo de alinhamento (o alinhamento será concluído no quarto sprint); 2) Um "Slider" de intensidade para cada imagem logo abaixo dos componentes de visualização; 3) Dois "spinboxes" para cada imagem, ao lado de seus respectivos "Sliders" de intensidade, representando o "range" mínimo e máximo de intensidade de cada imagem; 4) Um combobox para cada imagem contendo opções de pseudo-cores a serem aplicadas nas imagens.	Alexandre
Leitura de imagens DICOM	Elaboração de algoritmo para a leitura e conversão de imagens DICOM	Desenvolvimento de código o qual visa em imagens compatíveis com a interface para serem devidamente apresentadas	Gustavo
Visualização de Imagens DICOM	Interface para abrir dois arquivos DICOM. Cada um representando uma das imagens.	Criar um seletor de arquivos para o usuário escolher uma ou duas imagens no formato DICOM, apenas este formato, para ser mostrada.	Gustavo
Testes	Criação de casos de teste para este módulo e sua execução	De acordo com o Sprint Backlog do sistema serão criados os casos de teste prevendo todas as falhas que podem existir e relatar os problemas que forem encontrados.	Alexandre e Gustavo
Documentação	Elaboração dos diagramas correntes a esta sprint	Todos os requisitos do sistema serão transformados em representações gráficas com o uso dos diagramas da UML	Alysson
Validação	Apresentação do produto ao cliente/usuário para verificar se o que foi desenvolvido era o esperado	O cliente/usuário fará uso do módulo recém desenvolvido e testado para fazer a análise do cumprimento com suas necessidades e realizar a aprovação ou solicitar alterações	Lucas
Correções	Reparo ou melhoria de alguma funcionalidade do sistema	No caso de algum requisito não ser implementado no sistema, após a validação do cliente/usuário, serão feitas as necessárias correções e adaptações	Todos
SPRINT 4 (Duração: 31 dias)			
Nome	**Item**	**Descrição Detalhada**	**Responsável**
Alinhamento de Imagens	Criação da funcionalidade de alinhamento de imagens	Criar a funcionalidade que permitirá ao usuário selecionar o alinhamento da imagem 1 ou da imagem 2 com o modelo (previamente carregado pelo software). Em seguida deve ser mostrado o resultado no terceiro componente.	Gustavo
Interface de Alinhamento de Imagens	Criação da Interface de Visualização da tela de Alinhamento de Imagens	A interface será composta por: 1) Três componentes de visualização cada um com 3 imagens do tamanho 64x64.	Alexandre
Seleção de "Slices"	Inserir interação sobre as imagens permitindo fazer a troca de "slice" em tempo real	Criar mudança de slice dinamicamente de modo que o usuário mude o slice que está sendo mostrado em tempo real com o scroll do mouse ou pelo respectivo slider.	Gustavo

Controlador de intensidade de imagem	Um controlador de intensidade para cada imagem no formato "Slider"	1) Um "Slider" para cada imagem representando a intensidade da mesma. Ao modificar o valor do componente, a imagem deve ter alteração em sua intensidade atualizada imediatamente; 2) Um "Spinbox" contendo o valor exato deve ficar à direita do "Slider" em seu auxílio para uma atribuição mais precisa de valores	Alexandre e Gustavo
Testes	Criação de casos de teste para este módulo e sua execução	De acordo com o Sprint Backlog do sistema serão criados os casos de teste prevendo todas as falhas que podem existir e relatar os problemas que forem encontrados.	Alexandre e Gustavo
Documentação	Elaboração dos diagramas correntes a esta sprint	Todos os requisitos do sistema serão transformados em representações gráficas com o uso dos diagramas da UML	Alysson
Validação	Apresentação do produto ao cliente/usuário para verificar se o que foi desenvolvido era o esperado	O cliente/usuário fará uso do módulo recém desenvolvido e testado para fazer a análise do cumprimento com suas necessidades e realizar a aprovação ou solicitar alterações	Lucas
Correção	Reparo ou melhoria de alguma funcionalidade do sistema	No caso de algum requisito não ser implementado no sistema, após a validação do cliente/usuário, serão feitas as necessárias correções e adaptações	Todos
SPRINT 5 (Duração:27 dias)			
Nome	**Item**	**Descrição Detalhada**	**Responsável**
Interface de Visualização de Mapas Polares	Criação da Interface do módulo de visualização de mapas polares	Criar um componente de visualização com tamanho 2x3 em que as 2 primeiras colunas receberam a imagem 1 e 2 e a terceira será o mapa polar de uma delas ou resultante de operações matemáticas de subtração.	Alexandre
Geração de Mapas Polares	Implementação do algoritmo de visualização de mapas polares	Gerar um mapa polar para cada imagem. Caso haja duas, fazer operação de subtração e mostrar o resultado em um terceiro componente.	Alexandre
Testes	Criação de casos de teste para este módulo e sua execução	De acordo com o Sprint Backlog do sistema serão criados os casos de teste prevendo todas as falhas que podem existir e relatar os problemas que forem encontrados.	Alexandre e Gustavo
Documentação	Elaboração dos diagramas correntes a esta sprint	Todos os requisitos do sistema serão transformados em representações gráficas com o uso dos diagramas da UML	Alysson
Validação	Apresentação do produto ao cliente/usuário para verificar se o que foi desenvolvido era o esperado	O cliente/usuário fará uso do módulo recém desenvolvido e testado para fazer a análise do cumprimento com suas necessidades e realizar a aprovação ou solicitar alterações	Lucas
Correção	Reparo ou melhoria de alguma funcionalidade do sistema	No caso de algum requisito não ser implementado no sistema, após a validação do cliente/usuário, serão feitas as necessárias correções e adaptações	Todos
SPRINT 6 (Duração: 17 dias)			
Nome	**Item**	**Descrição Detalhada**	**Responsável**
Geração de Executáveis	Gerar executável final do software		Todos
Manual do usuário	Desenvolvimento de um contexto de ajuda ao usuário dentro do sistema		Alexandre

2 PLANO DE RISCOS

Nesta seção serão mostrados todos os riscos envolvidos nesse trabalho e o conjunto de atividades que fazem parte de sua prevenção e manutenção.

Risco 1: Falta de conhecimento da tecnologia escolhida

- **Categoria:** Técnico
- **Prioridade:** Média
- **Prevenção:** Ter desde o início do projeto a definição das tecnologias a serem usadas para buscar um prévio conhecimento.
- **Impacto:** Atrasar todo o projeto para adquirir os conhecimentos necessários para desenvolver o sistema.
- **Plano de Contingência:** No momento que os desenvolvedores do projeto identificarem que não possuem o conhecimento mínimo para desenvolverem o produto, todas às atividades referentes ao desenvolvimento da aplicação serão paradas. Com isso, a equipe irá buscar pessoas e materiais de apoio para aprenderem a tecnologia e voltarem as atividades. A partir disto o cronograma será restabelecido e o desenvolvimento do software prosseguirá.
- **Quando atuar:** Quando um erro não for solucionado dentro de uma semana.

Risco 2: Alocar os integrantes em atividades que não são compatíveis com suas habilidades

- **Categoria:** Projeto e Técnico
- **Prioridade:** Baixa
- **Prevenção:** Deixar o mais livre possível a escolha das atividades entre a equipe.
- **Impacto:** O cronograma será comprometido e o software desenvolvido da forma incorreta e mal documentado.

- **Plano de Contingência:** Quando algum integrante do grupo começar a apresentar baixo rendimento será feita uma reunião com a equipe para que ele exponha suas dificuldades. Caso fique claro que o indivíduo não gosta e/ou não tem habilidade para desempenhar a atividade a qual foi designado, haverá troca de função ou o grupo dará mais atenção às dificuldades desse integrante
- **Quando atuar:** No final de cada sprint após passar todos esses períodos de tentativas e ao final não apresentar resultados positivos.

Risco 3: Má distribuição de atividades

- **Categoria:** Projeto
- **Prioridade:** Média
- **Prevenção:** Avaliar o tempo médio de execução de cada atividade e atribuir uma quantidade de tarefas que dêem um número próximo de horas por integrante.
- **Impacto:** O indivíduo que ficar responsável por mais atividades poderá sofrer uma sobrecarga e não dar conta de executar todas suas funções, prejudicando as entregas.
- **Plano de Contingência:** A equipe se encontrará para verificar se a alocação de tarefas está justa. Caso fique claro que está irregular, as atividades serão redistribuídas.
- **Quando atuar:** Ao ser identificado que as atividades foram mal distribuídas.

Risco 4: Faltar tempo para finalizar o projeto

- **Categoria:** Projeto
- **Prioridade:** Média

- **Prevenção:** Elaborar um planejamento antes de dar início a qualquer implementação para prever os possíveis percalços que podem ocorrer.
- **Impacto:** O software será entregue com menos funcionalidades do que o prometido ou o projeto será cancelado.
- **Plano de Contingência:** A equipe fará uma reunião e o escopo deverá ser reformulado para reduzir a quantidade de funcionalidades, sem prejudicar sua qualidade.
- **Quando atuar:** No momento que a equipe perceber que o projeto não é viável para o tempo disponível.

Risco 5: Informações imprecisas

- **Categoria:** Negócio
- **Prioridade:** Média
- **Prevenção:** Escolher a técnica de coleta de requisitos mais compatível com a situação e colocar para desempenhar esta tarefa o integrante da equipe que tem maior capacidade de comunicação.
- **Impacto:** Comprometimento de todo o desenvolvimento do produto. Além disso, o cliente/usuário irá receber um produto que não atende suas demandas e o software deverá ser redesenvolvido, gastando muito mais tempo que o necessário. Da mesma forma, a omissão de informações por parte do cliente terão impacto em todo o projeto, desde a elaboração do cronograma até a fase de testes.
- **Plano de Contingência:** Fazer um novo estudo para identificar a causa do problema. Caso o problema exista devido o uso de uma técnica ineficiente, o analista fará estudos durante 1 ou 2 dias para encontrar melhores técnicas. Se a falta de informações foi causada pela falta de comunicação do cliente o analista tentará marcar uma nova entrevista para tentar uma abordagem mais eficiente de levantamento de requisitos.

- **Quando atuar:** Quando a dificuldade for encontrada.

Risco 6: Falta de comunicação e comprometimento por parte dos integrantes da equipe

- **Categoria:** Projeto
- **Prioridade:** Média
- **Prevenção:** Realizar encontros semanais para que todos informem o andamento das suas atividades.
- **Impacto:** Dificuldade em saber o andamento do trabalho, problemas podem não ser emitidos ao restante do grupo, sendo manifestados em falhas futuras e atrapalhar o andamento do projeto. Além disso, o resultado final do produto pode não atender a demanda do cliente/usuário
- **Plano de Contingência:** Fazer uma nova reunião para expor o que cada integrante está fazendo, definir novas atividades e agendar novas reuniões.
- **Quando atuar:** No momento que a falta de comunicação se refletir em falta de produtividade e resultado.

Risco 7: Falta da padronização do desenvolvimento do sistema e problemas de integração

- **Categoria:** Técnico
- **Prioridade:** Média
- **Prevenção:** Definir ao início do projeto padrões de desenvolvimento e estudar formas de integrar o sistema.
- **Impacto:** Dificuldade na integração do sistema quando compartilhado entre os desenvolvedores.
- **Plano de Contingência:** Elaborar uma nova padronização e exigir que todos façam seu uso.

- **Quando atuar:** Quando o problema se manifestar.

Risco 8: Atraso em atividades dependentes de resultados anteriores

- **Categoria:** Projeto
- **Prioridade:** Média
- **Prevenção:** Dar grande atenção ao planejamento do projeto.
- **Impacto:** Quando houverem atividades que dependem da execução de sua predecessora todo o cronograma ficará comprometido, havendo um efeito dominó.
- **Plano de Contingência:** No momento em que forem identificados problemas na execução de atividades interligadas a equipe se reunirá para avaliar o escopo do projeto e o desempenho do grupo. Se o problema for de escopo acima do prazo suas atividades mais importantes deverão ser priorizadas e as menos importantes descartadas. Se a equipe demonstrar dificuldades de desenvolvimento o grupo deverá parar para aprender os conteúdos necessários e se necessário o escopo deverá ser reavaliado.
- **Quando atuar:** Após o atraso passar de 5 dias

Risco 9: Utilizar metodologias de desenvolvimento que não sejam adequadas às características do projeto

- **Categoria:** Técnico
- **Prioridade:** Baixa
- **Prevenção:** Levantar os prós e contras das metodologias mais difundidas na área e escolher a mais adequada ao tipo de projeto.
- **Impacto:** O projeto poderá será desenvolvido com mais ou menos processos, resultando em um desenvolvimento menos eficiente.
- **Plano de Contingência:** A equipe fará uma pausa no projeto e investirá 1 ou 2 dias no estudo de uma nova abordagem. Ao encontrar uma nova

metodologia ela será testada durante 3 dias e se apresentar um bom resultado será utilizada na gestão do projeto.

- **Quando atuar:** No primeiro momento em que for identificado que a metodologia não responde aos objetivos da equipe e do projeto.

Risco 10: Investir tempo em tarefas menos relevantes

- **Categoria:** Projeto
- **Prioridade:** Baixa
- **Prevenção:** Fazer um levantamento detalhado de todas as atividades para ter mais chances de saber precisamente a prioridade de cada uma.
- **Impacto:** Haverá perda de tempo de modo a deixar as tarefas mais críticas acumuladas para o final do projeto e comprometerá o calendário.
- **Plano de Contingência:** Haverá uma nova reunião para definição de uma outra priorização das atividades para que sempre os detalhes fiquem para o final.
- **Quando atuar:** No momento que o problema for encontrado.

Risco 11: Elaborar uma documentação técnica demais

- **Categoria:** Técnico
- **Prioridade:** Baixa
- **Prevenção:** Realizar todos os trabalhos com o conhecimento do cliente.
- **Impacto:** O cliente não irá compreender a documentação, tirar conclusões precipitadas e presumir que algumas informações já foram adquiridas pelo analista, prejudicando a coleta de requisitos e ao futuro desenvolvimento.
- **Plano de Contingência:** Após o cliente demonstrar que a documentação está difícil de compreender o analista da equipe buscará formas mais simples de demonstrar o funcionamento do sistema ao cliente/usuário.

- **Quando atuar:** Logo que o cliente/usuário manifestar alguma falha ou dificuldade no entendimento na documentação

Risco 12: Elaborar um guia do usuário com pouca didática

- **Categoria:** Técnico
- **Prioridade:** Baixa
- **Prevenção:** Ao iniciar o guia fazer uma demonstração exemplar do guia ao usuário.
- **Impacto:** Quando o usuário fizer uso do manual para esclarecer alguma dúvida ele muito provavelmente não conseguirá saná-la e terá que entrar em contato com os desenvolvedores.
- **Plano de Contingência:** Quando a equipe souber que a documentação não está sendo um facilitador aos usuários ela passará por uma fase de reformulação.
- **Quando atuar:** Em seguida que o problema for identificado.

Risco 13: O software não rodar na máquina do cliente

- **Categoria:** Técnico
- **Prioridade:** Média
- **Prevenção:** Instalar e executar o software em outras máquinas antes de fazer a implantação.
- **Impacto:** A equipe de desenvolvimento do projeto perderá credibilidade com o cliente e gastará mais tempo que o necessário para gerar uma versão compatível com o seu computador.
- **Plano de Contingência:** Ao surgir o problema a equipe investirá tempo para gerar o software e encontrar a forma correta de preparar o ambiente de modo a ser compatível com o computador do cliente/usuário.

- **Quando atuar:** Após a identificação do problema.

Risco 14: Existência de falhas críticas

- **Categoria:** Negócio e Técnico
- **Prioridade:** Alta
- **Prevenção:** Adquirir conhecimento em técnicas de programação para evitar a criação de códigos “remendados” e elaborar rotinas de testes para que falhas sejam encontradas e corrigidas enquanto o produto está em desenvolvimento.
- **Impacto:** O sistema será entregue com muitas falhas e quando se manifestarem irão emitir erros ou oferecer um resultado inadequado que comprometa sua análise e a vida do paciente.
- **Plano de Contingência:** Durante todas as sprints do projeto o integrante responsável pela elaboração e execução de testes fará uma bateria de testes que para buscar brechas não encontradas pelos desenvolvedores o que irá contribuir para a qualidade do software. Caso as falhas persistam, o analista de testes fará novos casos de teste e os desenvolvedores poderão solucionar os erros.
- **Quando atuar:** A ação corretiva será executada logo que o problema for apresentado.

Risco 15: Alto custo de processamento e lentidão para executar a aplicação

- **Categoria:** Técnico
- **Prioridade:** Baixa
- **Prevenção:** Fazer uso de tecnologias que tenham característica de priorizar o desempenho, além de fazer uso de técnicas de programação de alto desempenho.
- **Impacto:** O uso do software pode ser suspenso pelo usuário e necessitar um novo desenvolvimento ou correção.

- **Plano de Contingência:** Se o software estiver finalizado e cumprindo com seu propósito, porém, seu desempenho for abaixo do esperado, será aberta uma sprint no projeto para a refatoração do código de modo a utilizar técnicas de alto desempenho para atingir o resultado desejado.
- **Quando atuar:** Após a identificação do problema

Risco 16: O cliente pode não estar disponível a equipe

- **Categoria:** Negócio
- **Prioridade:** Baixa
- **Prevenção:** Manter um bom relacionamento com o cliente durante o projeto.
- **Impacto:** Com a ausência do cliente/usuário no desenvolvimento do projeto a primeira fase de coleta de requisitos será tomada como o documento a ser seguido durante todo o projeto e todos os questionamentos que surgirem serão solucionados pela equipe.
- **Plano de Contingência:** Caso não haja disponibilidade por parte do cliente, a equipe insistirá para realizar um encontro para debater sobre dúvidas encontradas e resultados adquiridos para contribuir com o projeto. Se o cliente não colaborar com o projeto, todas as decisões e análises de resultado ficarão a cargo da equipe.
- **Quando atuar:** Após a equipe precisar do cliente para algum esclarecimento ou demonstração de resultados e o encontro se mostrar difícil.

Risco 17: Desistência projeto por parte do cliente

- **Categoria:** Negócio
- **Prioridade:** Baixa
- **Prevenção:** Manter um relacionamento transparente com o cliente e a equipe se empenhar para fazer um bom produto.
- **Impacto:** O projeto é dado como encerrado e tudo o que foi feito é perdido.

- **Plano de Contingência:** Se o cliente se mostrar insatisfeito e desejar desistir do projeto a equipe se reunirá para tentar uma nova negociação. Para isso, um dos integrantes do grupo será designado para se encontrar com o cliente e tentar convencê-lo de que o produto tem qualidade e atenderá a todos seus propósitos.
- **Quando atuar:** Logo que o projeto for recusado.

3 DIAGRAMA DE CLASSES

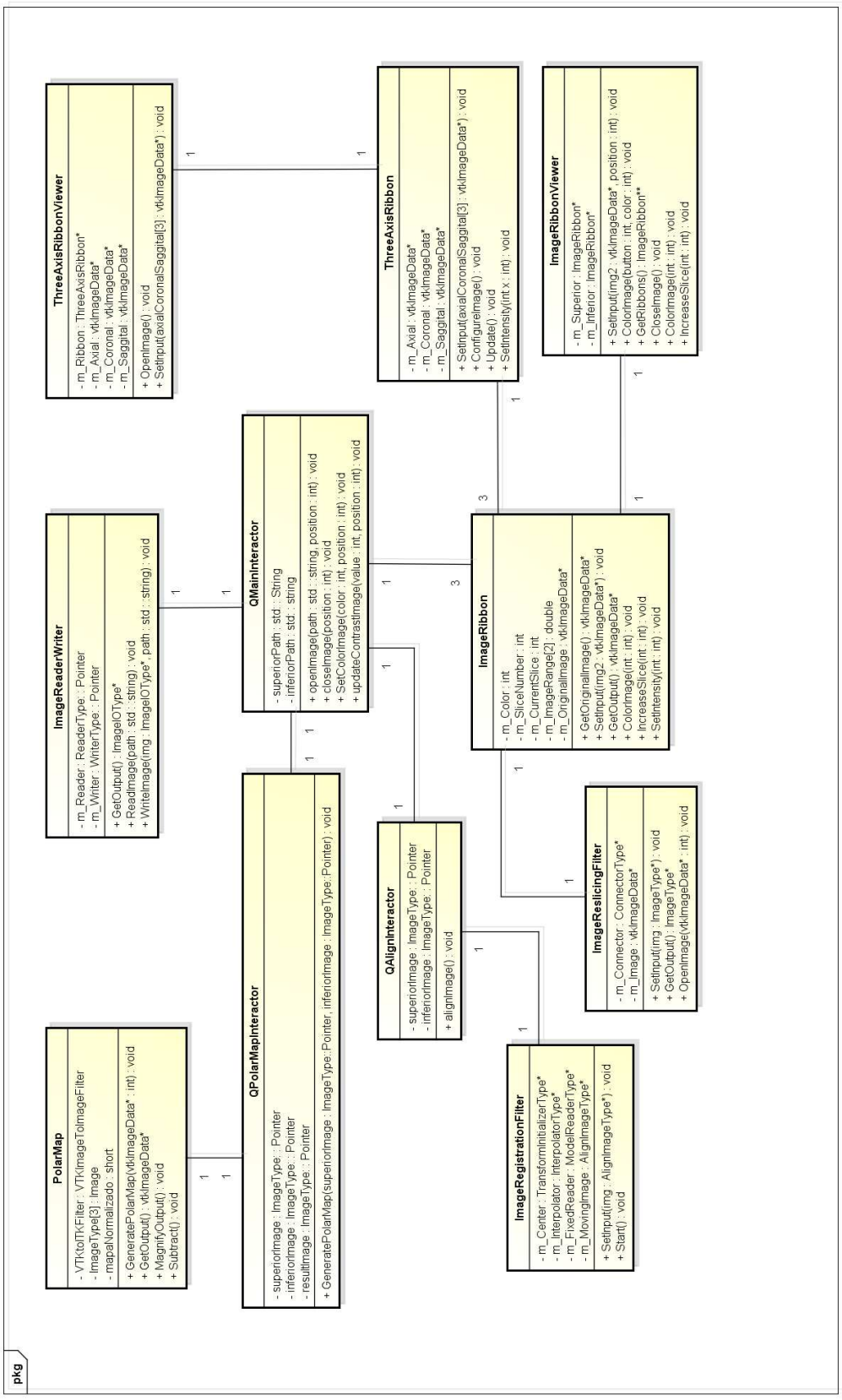


Figura 39 – Diagrama de classes

4 DIAGRAMA DE CASOS DE USO

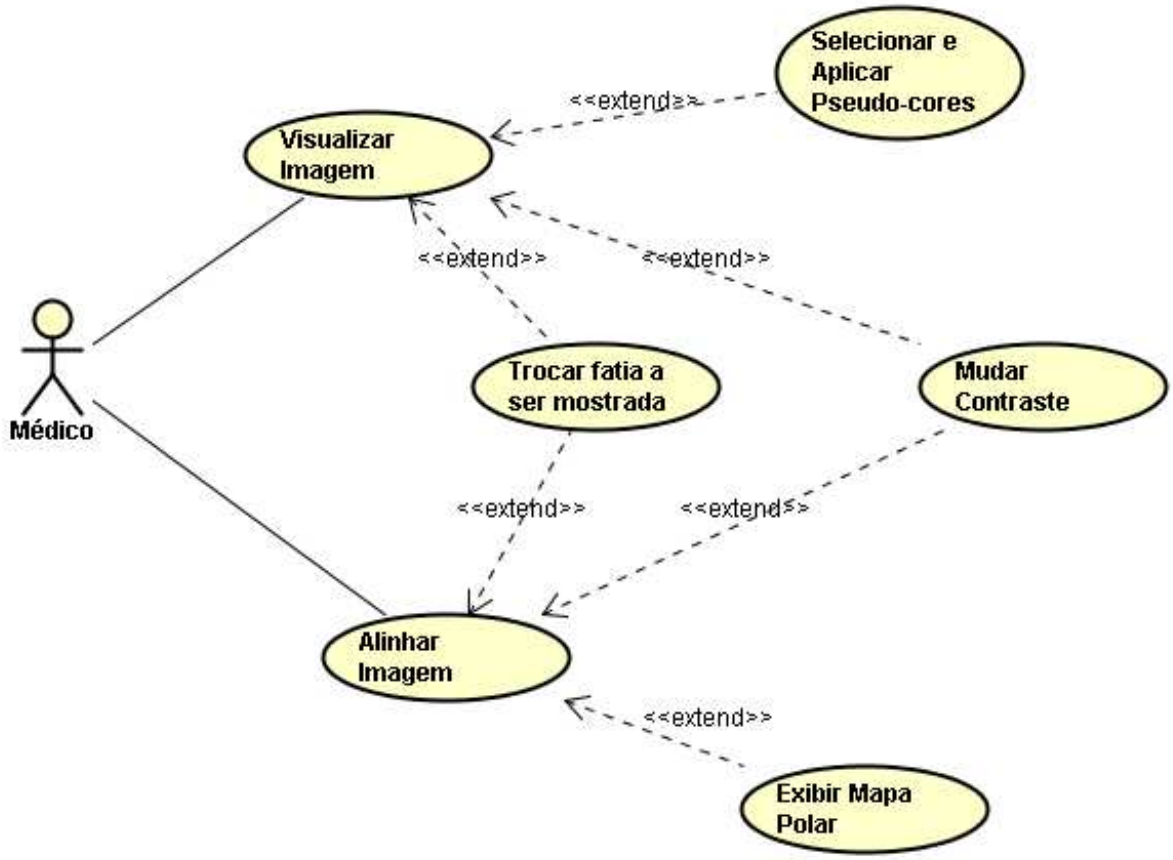


Figura 40 – Diagrama de casos de uso

5 ESPECIFICAÇÃO DE CASOS DE USO

Após o levantamento das funcionalidades do sistema e da especificação dos casos de uso que o sistema terá, cada caso de uso será descrito e exemplificado pelo uso de interfaces e fluxos de ações, como será visto abaixo.

UC01 Visualizar Imagem

Este caso de uso tem a função de exibir uma ou duas imagens nos três diferentes eixos e permitir que o usuário altere o contraste, aplique pseudo-cores ou mude a fatia que está sendo mostrada.

DV01 – Interface Principal com a exibição de uma imagem DICOM

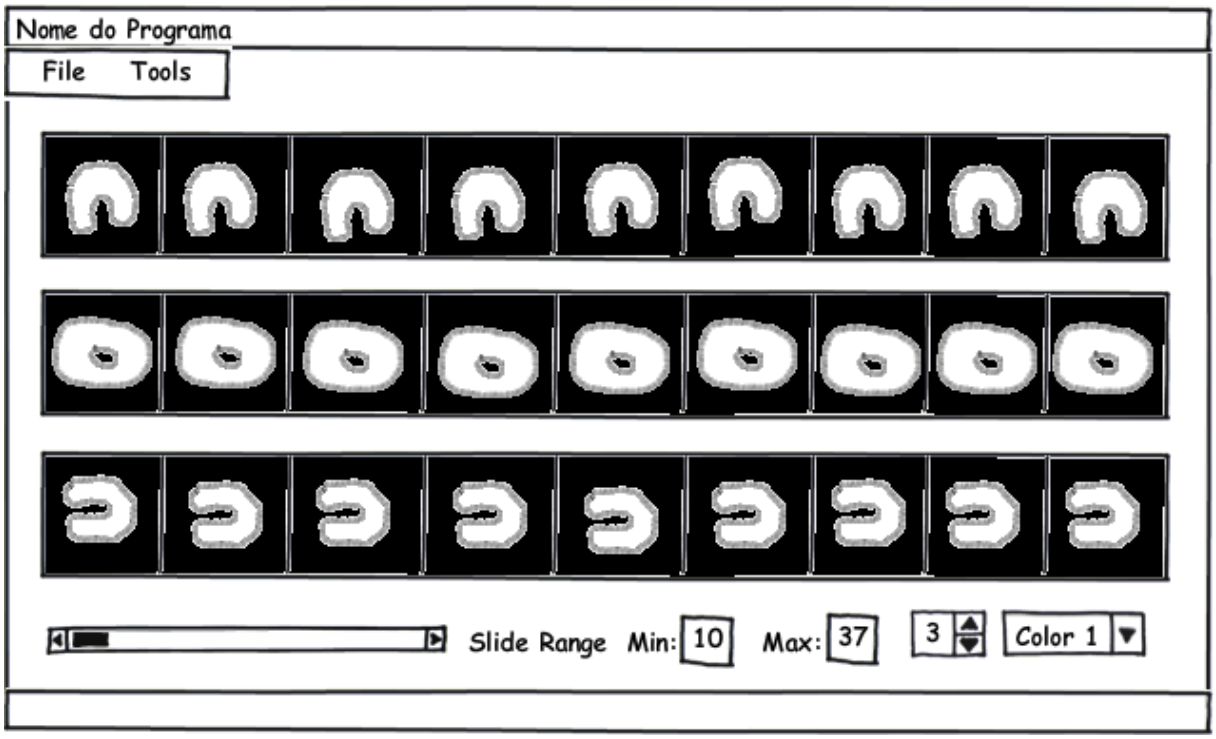


Figura 41 – Esboço da tela principal com a imagem carregada nos eixos X, Y e Z

DV02 – Interface Principal com a exibição de duas imagens DICOM

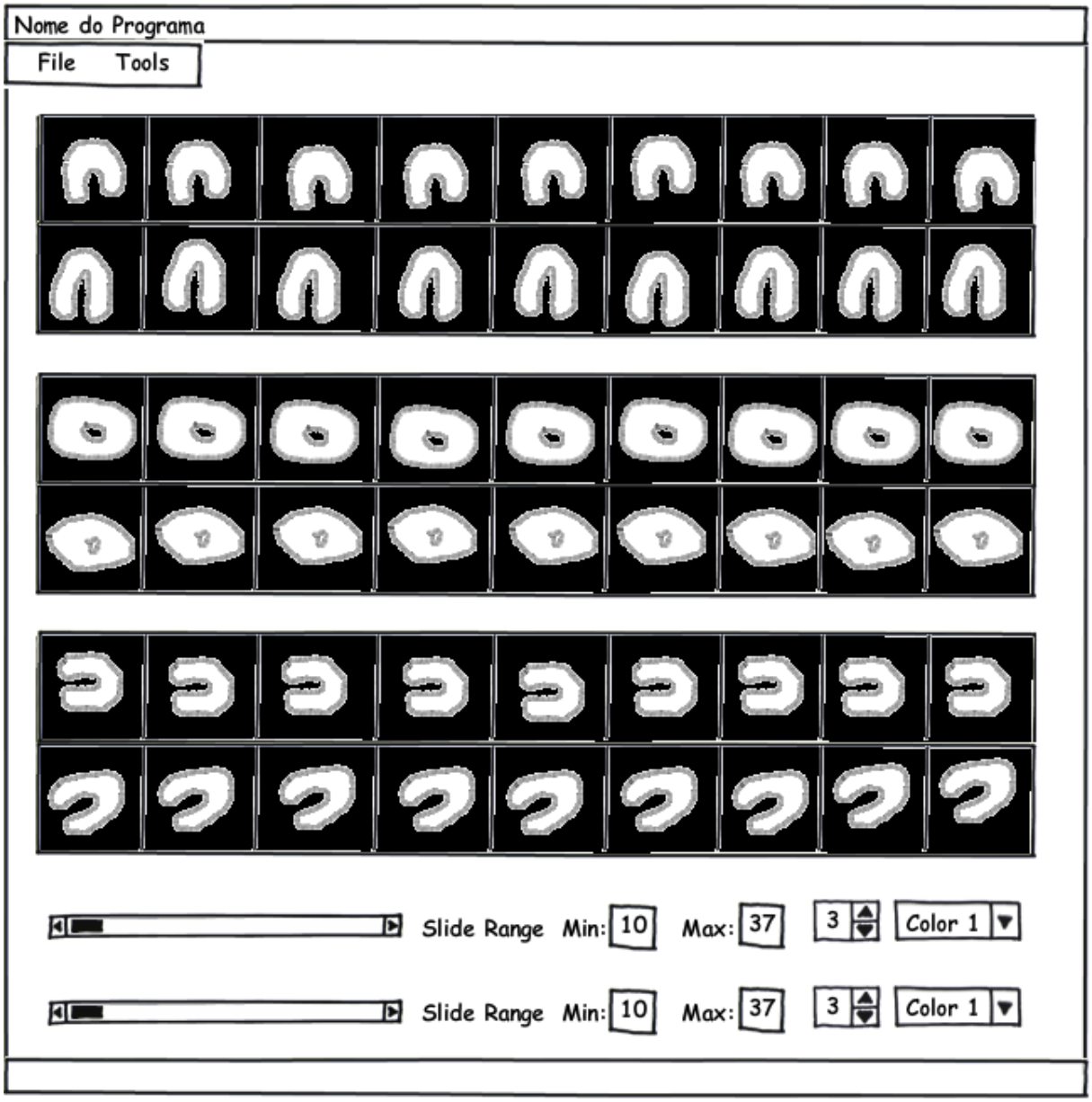


Figura 42 – Esboço da tela principal com duas imagens carregadas nos eixos X, Y e Z

DV03 – Seletor de Arquivos

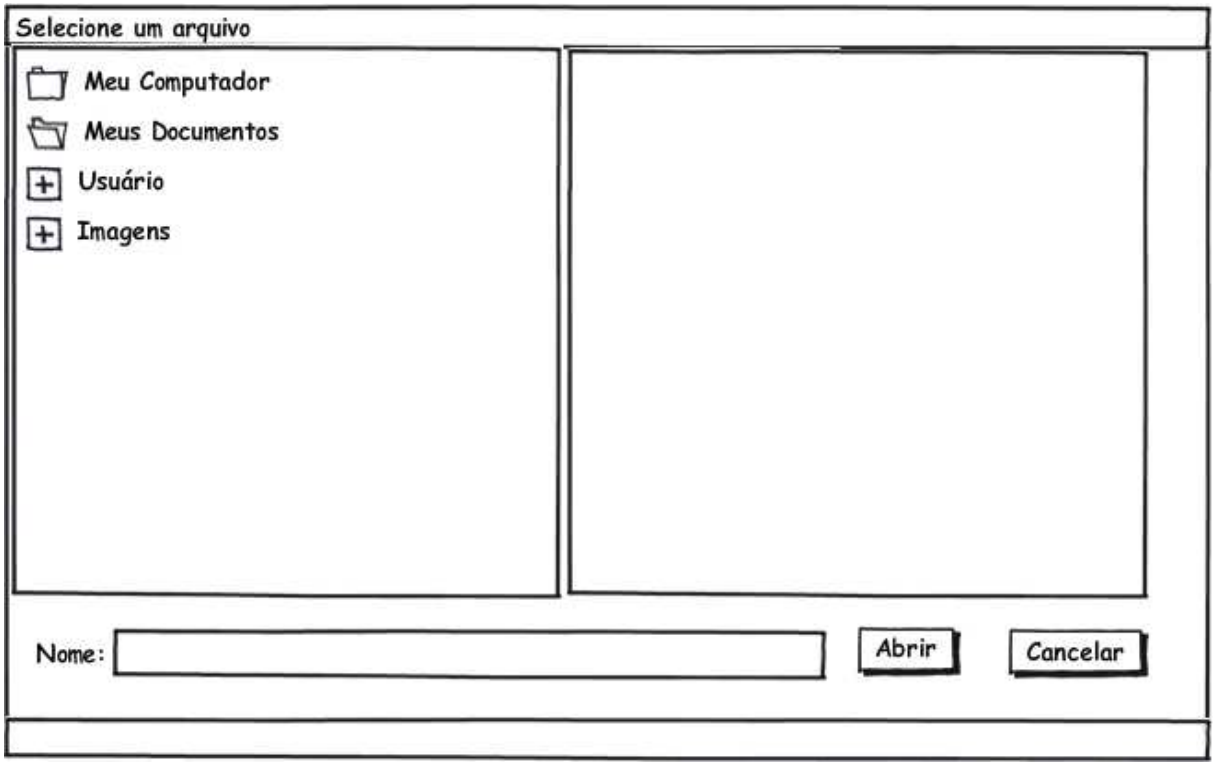


Figura 43 – Esboço da tela do Seletor de Arquivos

Pré-condições

Este caso de uso só é executado se:

1. A imagem deve ser selecionada pelo usuário (**DV03**)

Pós-condições

Ao terminar a execução deste caso de uso o sistema deverá:

1. Exibir a imagem selecionada pelo usuário nos três diferentes eixos

Ator Primário

Médico

Fluxo de Eventos Principal

1. O sistema apresenta a tela principal com a exibição da imagem selecionada (**DV01** ou **DV02**)
2. O usuário seleciona uma pseudo-cor (**A1**)
3. O usuário muda o valor de contraste (**A2**)
4. O usuário muda a fatia que está sendo mostrada (**A3**)
5. O usuário abre a imagem 2 (**A4**)
6. O caso de uso é encerrado

Fluxos Alternativos

A1. Combo Box de “Pseudo-cores”

1. O sistema chama o caso de uso “**UC02 – Selecionar e Aplicar Pseudo-cores**”
2. O sistema volta ao fluxo principal

A2. Spinbox de alteração de contraste

1. O sistema chama o caso de uso “**UC03 – Mudar Contraste**”
2. O sistema volta ao fluxo principal

A3. Rolagem do scroll do mouse sobre a imagem nos diferentes eixos

1. O sistema chama o caso de uso “**UC04 – Trocar fatia a ser mostrada**”
2. O sistema mostra a imagem com a fatia que foi selecionada
3. O sistema volta ao fluxo principal

A4. Abrir a segunda imagem

1. O usuário deve selecionar a imagem de entrada (**DV03**)
2. O sistema mostra a tela **DV2** com ambas as imagens carregadas
3. O sistema volta ao fluxo principal

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso.

UC02 Selecionar e Aplicar Pseudo-cores

Este caso de uso serve para o sistema modificar a coloração das imagens que estão sendo mostradas, melhorando assim sua visualização.

DV01 – Exibição da imagem DICOM na tela principal

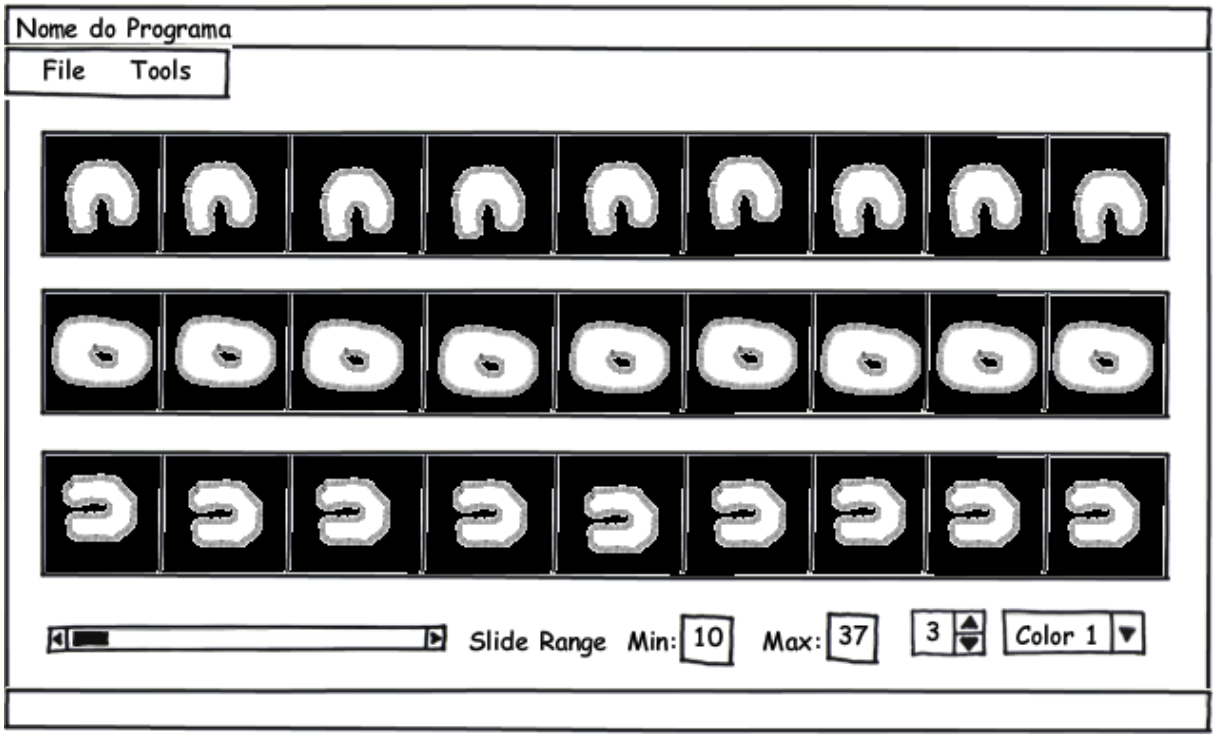


Figura 44 – Esboço da tela principal com a imagem carregada nos eixos X, Y e Z

DV02 – Exibição da imagem DICOM com pseudo-cor

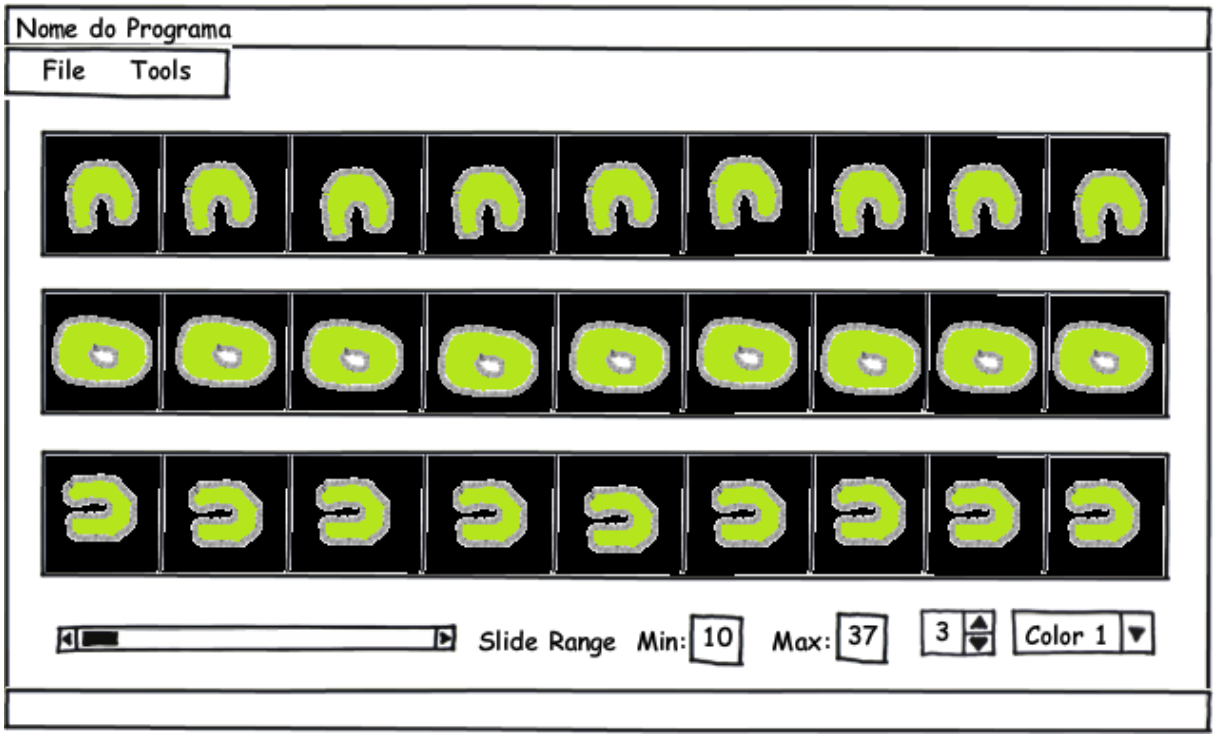


Figura 45 – Esboço da tela principal com a exibição da imagem com uma pseudo-cor aplicada

Pré-condições

Este caso de uso só é iniciado se:

1. A imagem deve ser selecionada pelo usuário

Pós-condições

Após terminar a execução deste caso de uso o sistema deve:

1. Exibir a imagem com a pseudo-cor selecionada (**DV02**)

Ator Primário

Médico

Fluxo de Eventos Principal

1. O sistema apresenta a imagem carregada pelo usuário (**DV01**)
2. O usuário escolhe a cor no combobox de pseudo-cores
3. O sistema apresenta a imagem na tela principal com a pseudo-cor selecionada (**DV02**)
4. O caso de uso é encerrado

Fluxos Alternativos

Não há fluxos alternativos para este caso de uso

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso

UC03 Mudar Contraste

Este caso de uso serve para o sistema modificar o contraste das imagens que estão sendo mostradas, melhorando assim sua visualização.

DV01 – Exibição da imagem DICOM na tela principal

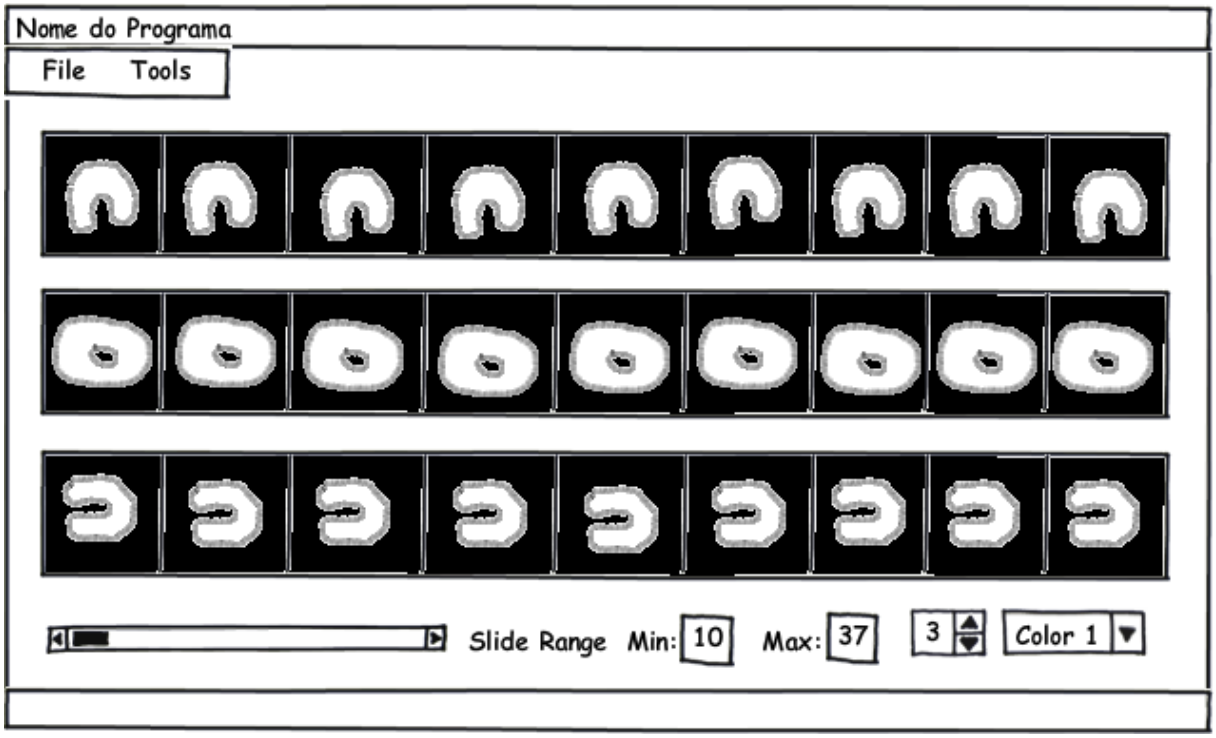


Figura 46 – Esboço da tela principal com a imagem carregada nos eixos X, Y e Z

DV02 – Exibição da imagem DICOM com o contraste alterado

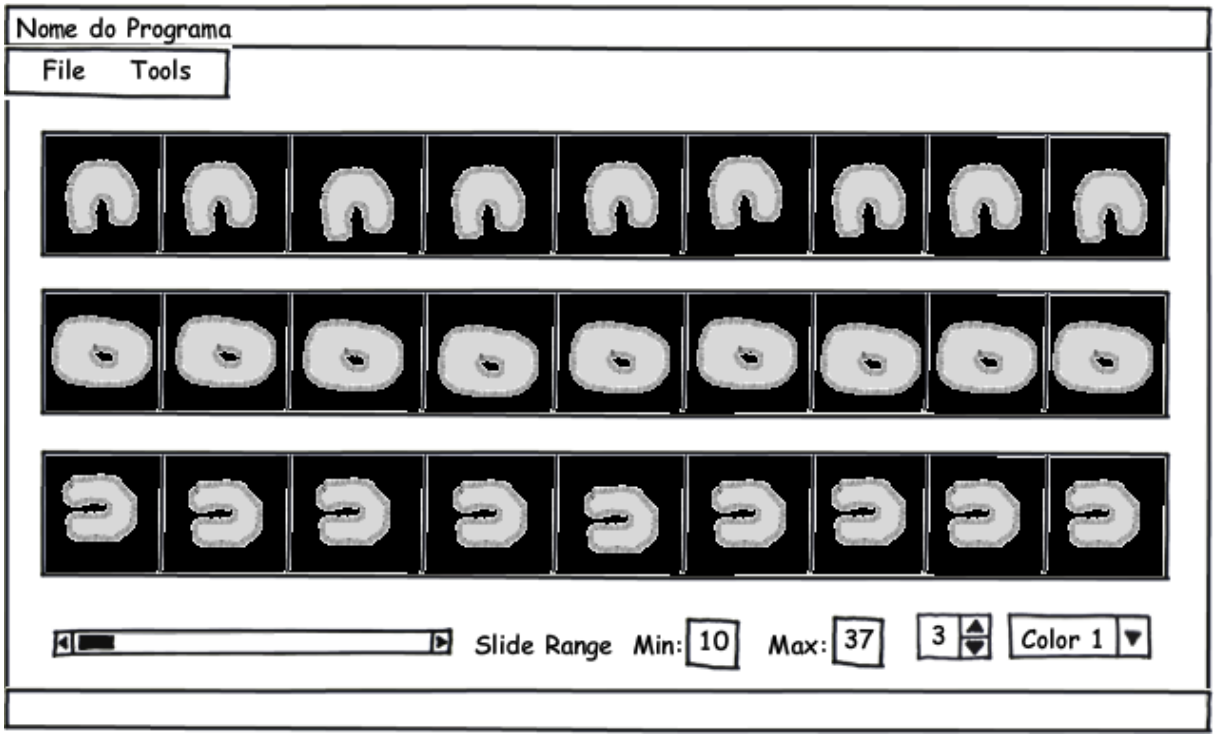


Figura 47 – Esboço da tela principal com a exibição da imagem com o contraste modificado

Pré-condições

Este caso de uso só é iniciado se:

1. A imagem deve ser selecionada pelo usuário

Pós-condições

Após terminar a execução deste caso de uso o sistema deve:

1. Exibir a imagem com a o novo valor de contraste

Ator Primário

Médico

Fluxo de Eventos Principal

1. O sistema exibe e imagem carregada pelo usuário (**DV01**)
2. O usuário movimenta o spinbox de contraste
3. O sistema apresenta a imagem com o novo contraste aplicado na tela de visualização (**DV02**)
4. O caso de uso é encerrado

Fluxos Alternativos

Não há fluxos alternativos para este caso de uso

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso

UC04 Trocar fatia a ser mostrada

Este caso de uso serve para o sistema modificar a fatia que está sendo mostrada.

DV01 - Exibição da imagem DICOM na tela principal

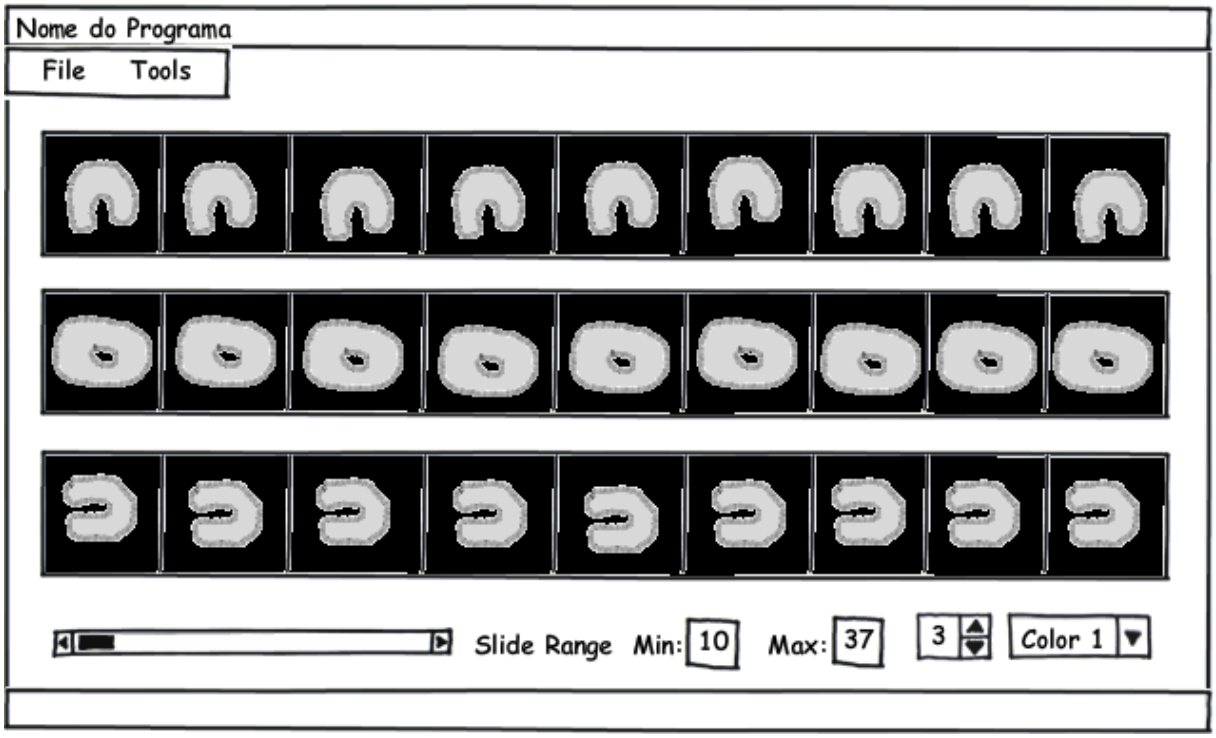


Figura 48 – Esboço da tela principal com a imagem carregada nos eixos X, Y e Z

DV02 - Exibição da imagem DICOM da tela DV01 com outra fatia selecionada

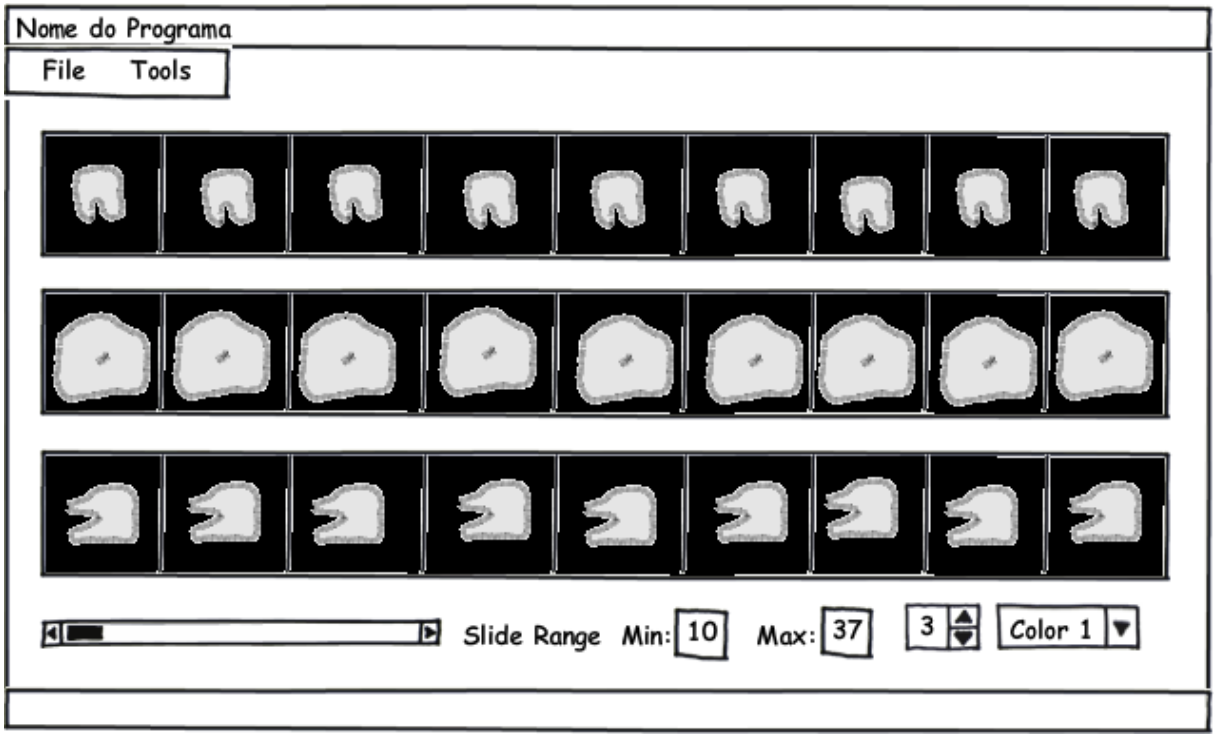


Figura 49 – Esboço da tela principal com a exibição da imagem com outra fatia a mostra

Pré-condições

Este caso de uso só é iniciado se:

1. A imagem deve ser selecionada pelo usuário

Pós-condições

Após terminar a execução deste caso de uso o sistema deve:

1. Exibir a imagem com a fatia que foi selecionada (**DV02**)

Ator Primário

Médico

Fluxo de Eventos Principal

1. O sistema exibe e imagem carregada pelo usuário (**DV01**)
2. O usuário movimenta o scroll do mouse
3. O sistema apresenta a imagem com a fatia selecionada pelo usuário na tela de visualização (**DV02**)
4. O caso de uso é encerrado

Fluxos Alternativos

Não há fluxos alternativos para este caso de uso

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso

UC05 Alinhar Imagem

Este caso de uso permite que o usuário alinhe a imagem do paciente com a imagem modelo.

DV01 – Interface principal do alinhador

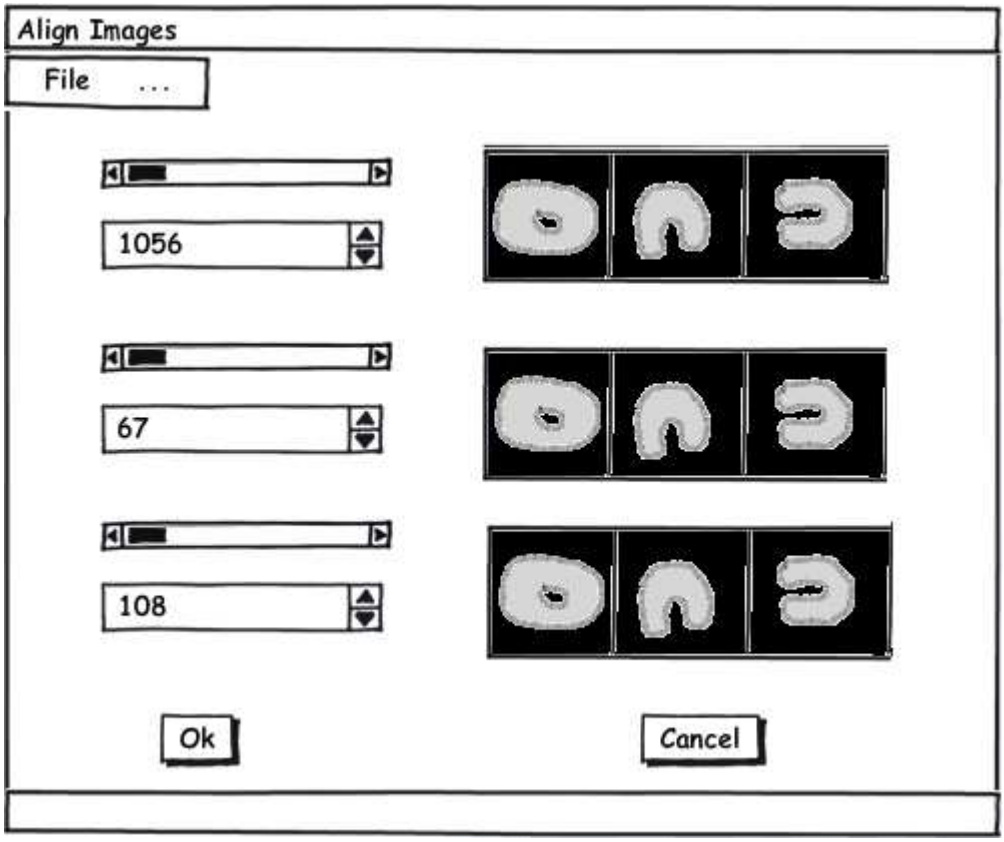


Figura 50 – Esboço da tela principal do alinhador

DV02 – Tela para armazenamento de arquivo

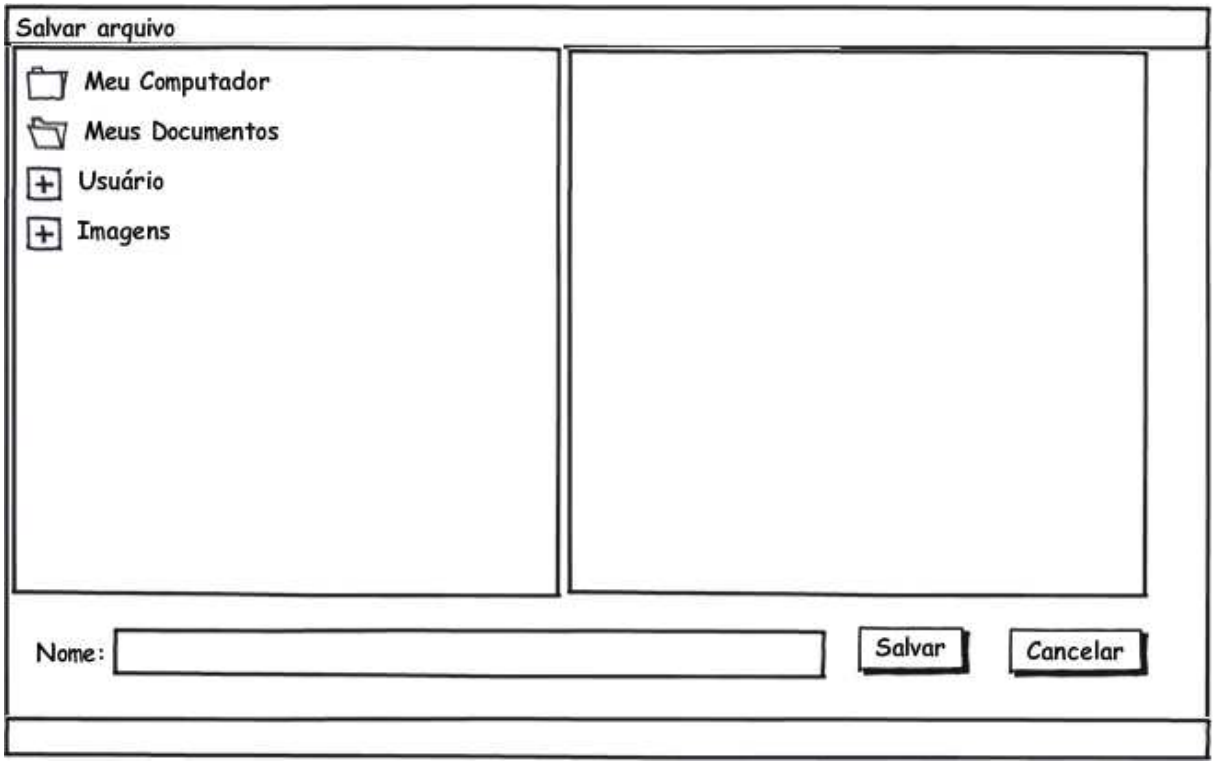


Figura 51 – Esboço da tela para o usuário salvar a imagem

Pré-condições

Este caso de uso só é iniciado após:

1. A imagem deve ser selecionada pelo usuário

Pós-condições

Ao finalizar este caso de uso o sistema deverá:

1. Exibir a imagem alinhada em relação a imagem modelo

Ator Primário

Médico

Fluxo de Eventos Principal

1. Exibir a tela (**DV01**)
2. O usuário clica na opção “Align Image”
3. O usuário muda o valor de contraste (**A1**)
4. O usuário muda a fatia que está sendo mostrada (**A2**)
5. O usuário clica no botão “Exibir Mapa Polar” (**A3**)
6. O usuário clica em “Salvar Imagem” (**A4**)
7. O caso de uso é encerrado

Fluxos Alternativos

A1. Scroll Bar ou Numeric Stepper de alteração de contraste

1. O sistema chama o caso de uso “**UC03 – Mudar Contraste**”
2. O sistema volta ao fluxo principal

A2. Rolagem do scroll do mouse sobre a imagem nos diferentes eixos

1. O sistema chama o caso de uso “**UC04 – Trocar fatia a ser mostrada**”
2. O sistema mostra a imagem com a fatia que foi selecionada

A3. Exibir Mapa Polar

1. O sistema executa o caso de uso “**UC06 – Exibir Mapa Polar**”
2. O sistema volta ao fluxo principal

A4. Salvar Imagem

1. O sistema exibe a tela (**DV02**) para o usuário escolher o diretório em que a imagem será armazenada
2. A imagem é salva
3. O sistema volta ao fluxo principal

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso

UC06 Exibir Mapa Polar

Este caso de uso permite que o usuário visualize a imagem alinhada na perspectiva de mapa polar.

DV1 – Tela principal do sistema

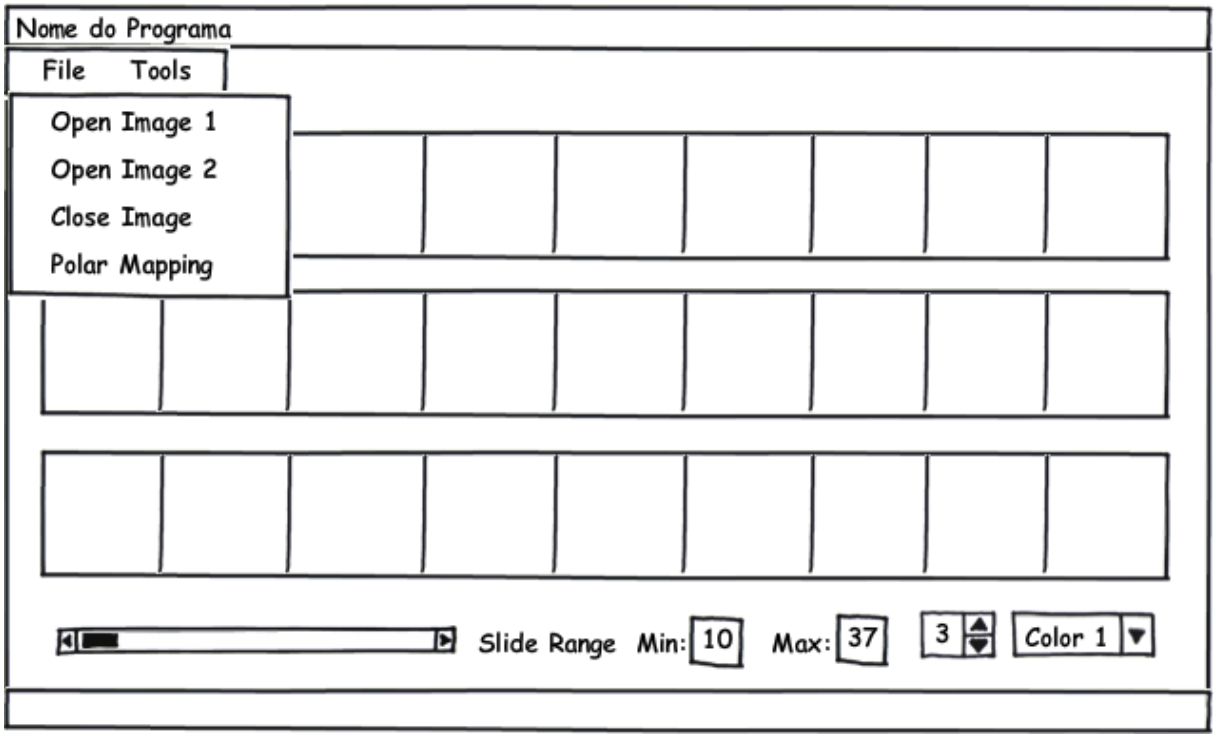


Figura 52 – Esboço da tela principal com o menu a vista

DV02 – Interface do mapa polar

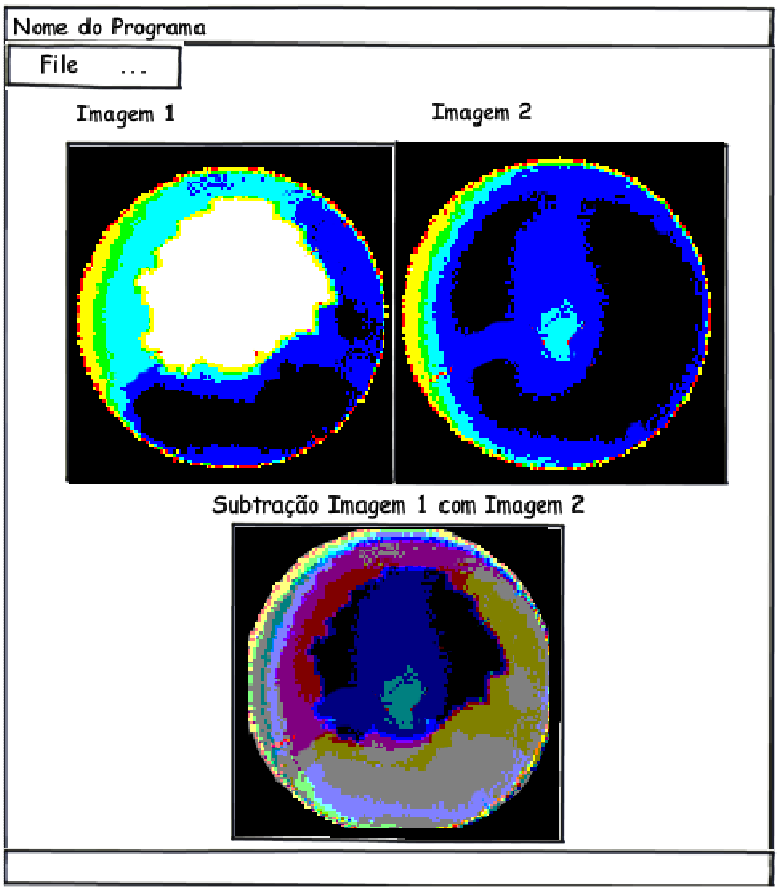


Figura 53 – Esboço da tela de geração de mapa polar

Pré-condições

Este caso de uso só entra em execução após:

1. O caso de uso “**UC06 - Alinhar Imagem**” estiver finalizado

Pós-condições

Ao finalizar este caso de uso o sistema deverá:

1. Exibir a imagem alinhada em forma de mapa polar

Ator Primário

Médico

Fluxo de Eventos Principal

1. O usuário clica em “Pollar Mapping” na interface principal
2. O sistema apresenta a tela (**DV02**) com o mapa polar criado
3. O caso de uso é encerrado

Fluxos Alternativos

Não há fluxos alternativos para este caso de uso

Fluxos de Exceção

Não há fluxos de exceção para este caso de uso

6 DIAGRAMAS DE SEQUÊNCIA

Para exemplificar o fluxo que cada funcionalidade possui serão apresentados os diagramas de sequência a seguir.

UC01 Visualizar Imagem

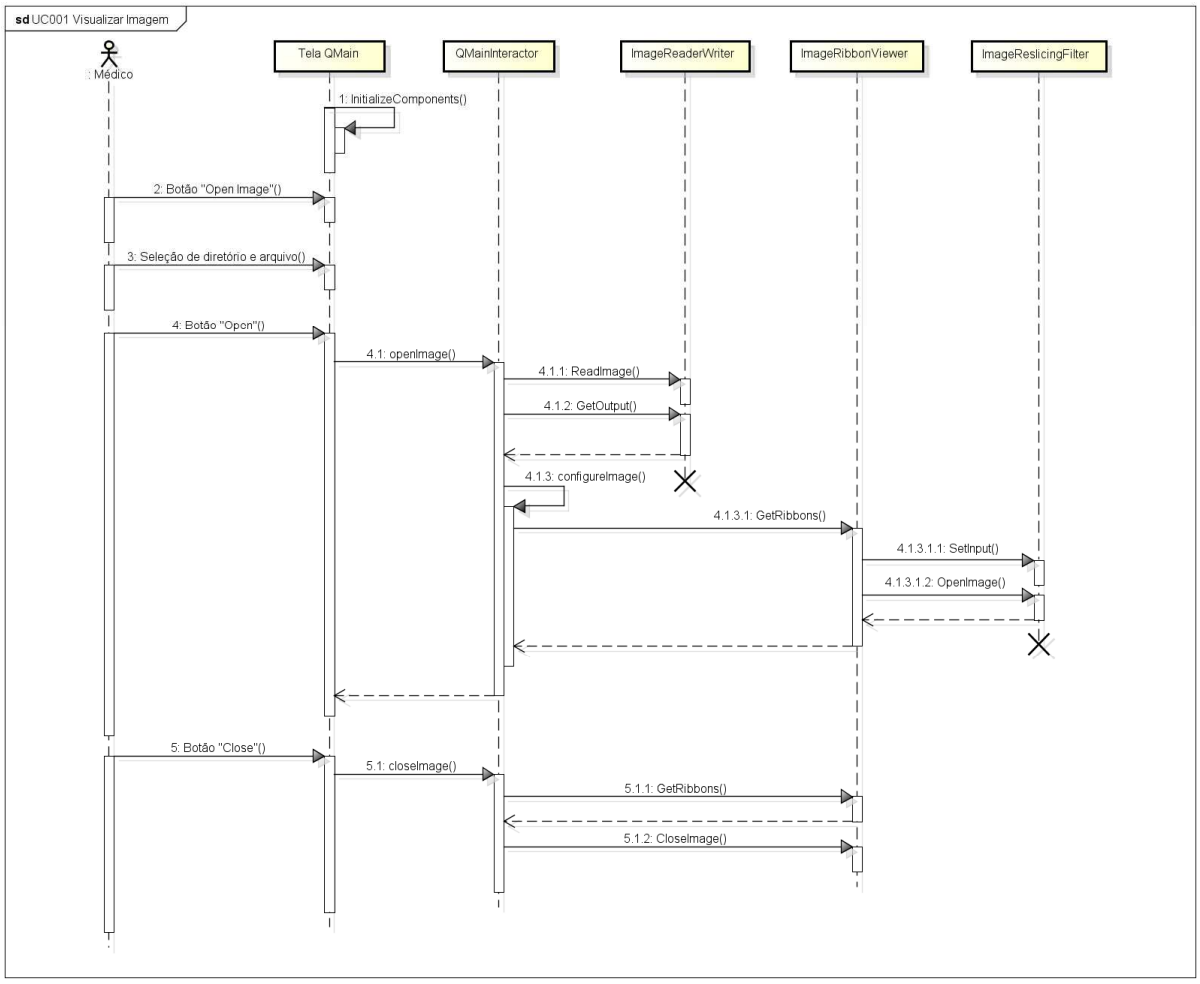


Figura 54 – Diagrama de sequência do caso de uso “Visualizar Imagem”

UC02 Alterar Pseudo-cores

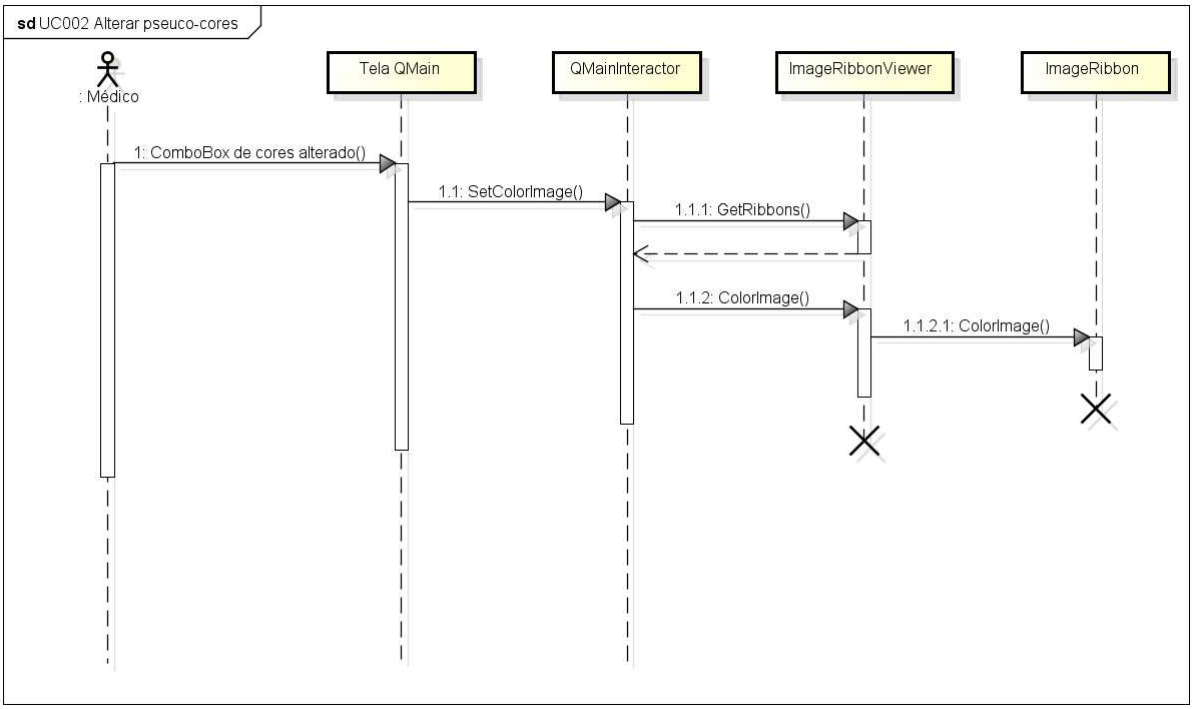


Figura 55 – Diagrama de sequência do caso de uso “Alterar Pseudo-cores”

UC03 Trocar fatia a ser mostrada

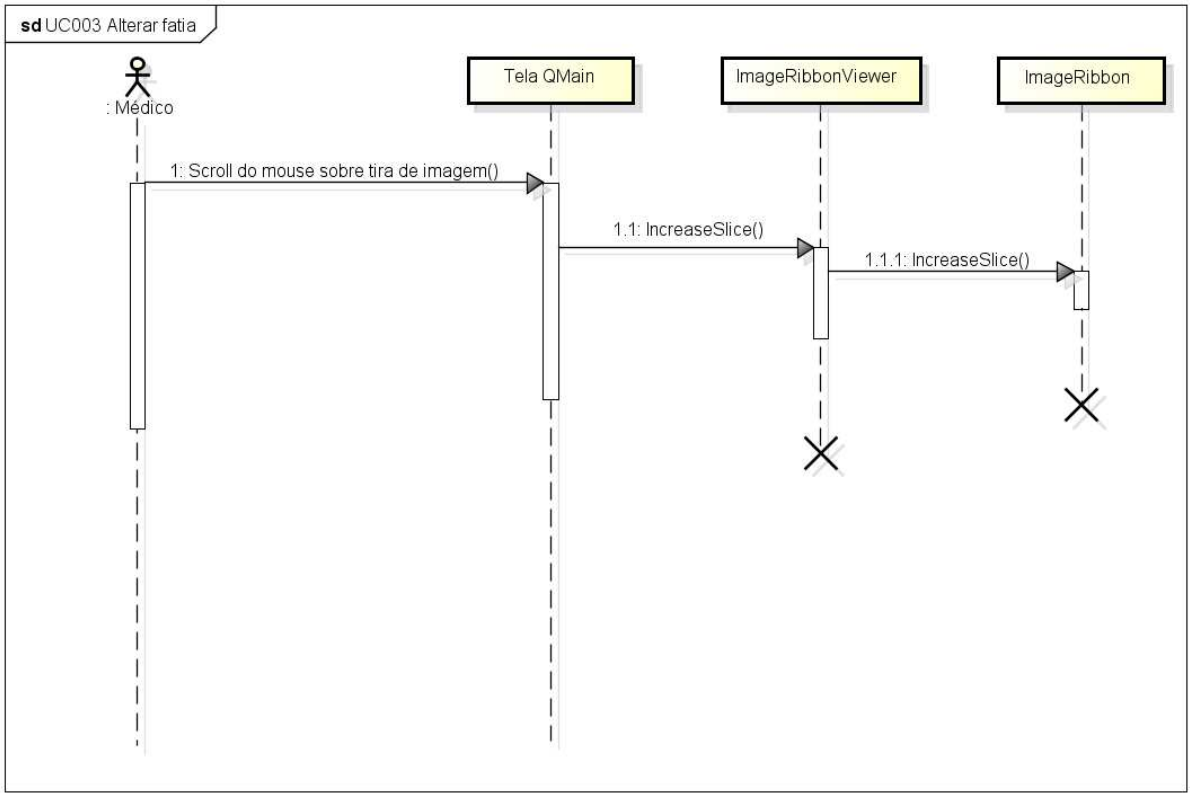


Figura 56 – Diagrama de sequência do caso de uso “Trocar fatia a ser mostrada”

UC04 Mudar Contraste

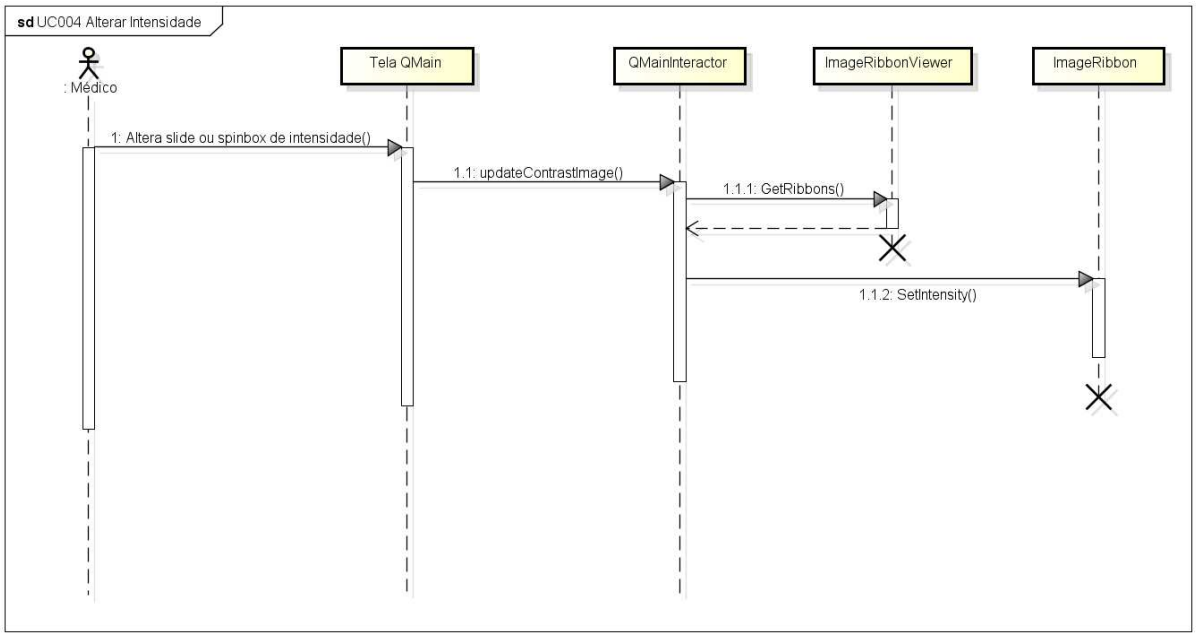


Figura 57 – Diagrama de sequência do caso de uso “Mudar Contraste”

UC05 Alinhar Imagem

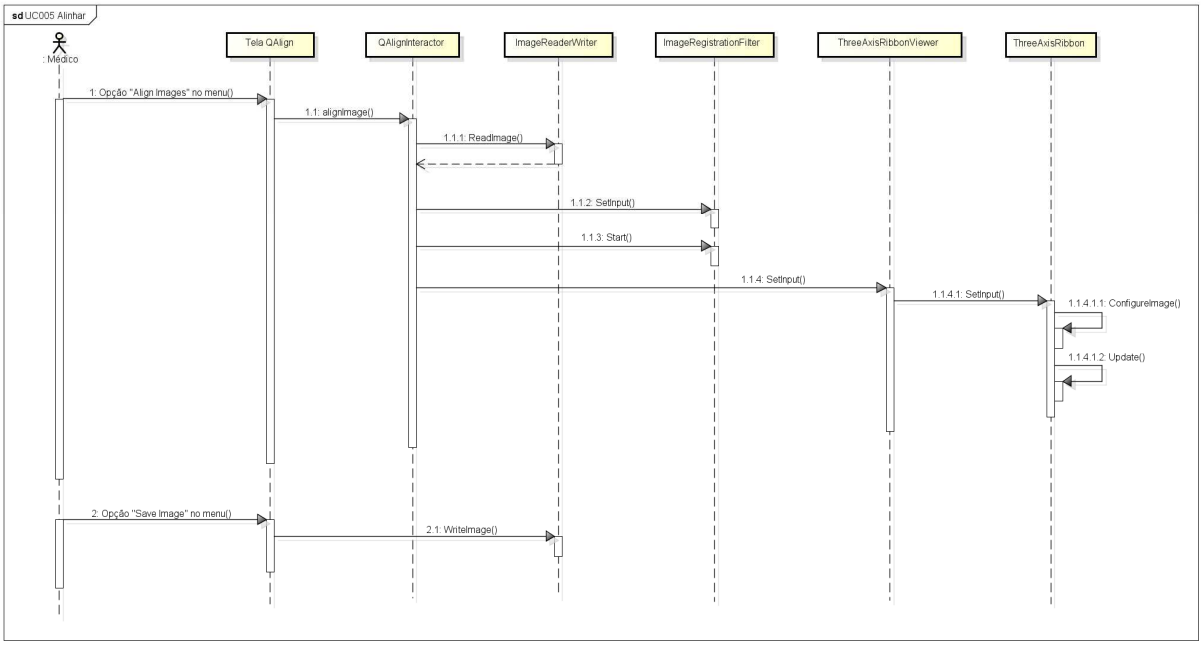


Figura 58 – Diagrama de sequência do caso de uso “Alinhar Imagem”

UC06 Gerar Mapa Polar

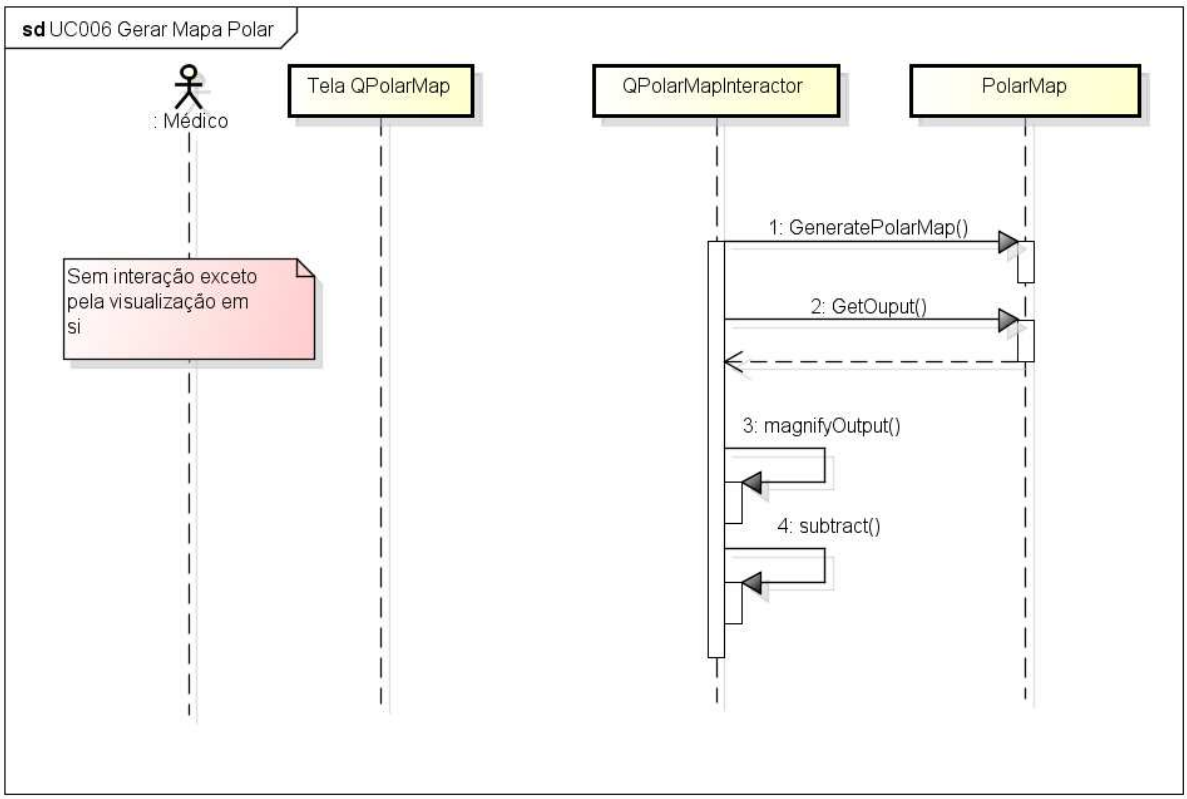


Figura 59 – Diagrama de sequência do caso de uso “Gerar Mapa Polar”