UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

CURSO BACHARELADO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

ARTUR SANTOS NOGUEIRA

INTEGRAÇÃO DO MAMUTE SEISMIC TOOLS AO OPENDTECT VIA PLUGIN NATIVO EM C++

PAU DOS FERROS

2026

ARTUR SANTOS NOGUEIRA

INTEGRAÇÃO DO MAMUTE SEISMIC TOOLS AO OPENDTECT VIA PLUGIN NATIVO EM C++

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Orientador: Ítalo Augusto Souza de Assis, Prof. Dr.

PAU DOS FERROS

2026

N778i Nogueira, Artur Santos.
Integração do mamute seismic tools ao opendtect via plugin em C++ / Artur Santos Nogueira. - 2026.
38 f. : il.

Orientador: Ítalo Augusto Souza de Assis.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2026.

1. Integração de Software. 2. Mamute. 3. OpendTect. 4. Processamento Sísmico. 5. Plugin C++. I. Assis, Ítalo Augusto Souza de, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

ARTUR SANTOS NOGUEIRA

INTEGRAÇÃO DO MAMUTE SEISMIC TOOLS AO OPENDTECT VIA PLUGIN NATIVO EM C++

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Defendida em: 01/07/2026

BANCA EXAMINADORA

Ítalo Augusto Souza de Assis, Prof. Dr. (UFERSA)
Presidente

João Batista Fernandes, Me. (UFRN)
Membro Examinador

Samuel Xavier de Souza, Prof. Dr. (UFRN)
Membro Examinador

Peterson Nogueira Santos, Prof. Dr. (UFBA)
Membro Examinador

RESUMO

Este trabalho apresenta o desenvolvimento de um plugin nativo em C++ para o OpendTect que integra as ferramentas de processamento sísmico do Mamute Seismic Tools. O Mamute é um sistema de simulação e processamento sísmico que implementa modelagem sísmica acústica e visco-acústica. Presentemente, o Mamute é controlado exclusivamente via linha de comando,exigindo que o usuário alterne entre ambientes de processamento e interpretação sem integração direta. A solução proposta elimina essa fragmentação ao permitir configuração, execução, monitoramento e visualização de simulações sísmicas diretamente no OpendTect. O plugin é organizado em cinco componentes principais: estruturas de parâmetros, gerenciador de comandos externos, fluxo de preparação e execução da modelagem, monitor de execução e interface gráfica. A integração tem início pela modelagem sísmica visco-acústica, validada com dados sintéticos, servindo como base para a integração incremental da RTM e da FWI. O desenvolvimento contribui para ampliar o alcance do Mamute junto a usuários da plataforma.

Palavras-chave: integração de software; processamento sísmico; OpendTect; Mamute; plugin C++

ABSTRACT

This work presents the development of a native C++ plugin for OpendTect that integrates the seismic processing tools of the Mamute Seismic Tools. Mamute is a seismic simulation and processing system that implements acoustic and visco-acoustic seismic modeling. Currently, Mamute is controlled exclusively through a command-line interface, requiring users to alternate between processing and interpretation environments without direct integration. The proposed solution eliminates this fragmentation by enabling the configuration, execution, monitoring, and visualization of seismic simulations directly within OpendTect. The plugin is organized into five main components: parameter structures, external command manager, modeling preparation and execution workflow, execution monitor, and graphical user interface. Integration begins with visco-acoustic seismic modeling, validated with synthetic data, serving as a foundation for the incremental integration of RTM and FWI. The development contributes to expanding Mamute's reach among platform users.

Keywords: software integration; seismic processing; OpendTect; Mamute; C++ plugin

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A exploração sísmica de reflexão é a principal técnica geofísica para o imageamento da subsuperfície, utilizada na prospecção de recursos como petróleo e gás natural (Yilmaz, 2001). Um levantamento sísmico de reflexão compreende três etapas: aquisição, processamento e interpretação.

Na aquisição, fontes de energia artificial, como explosivos em terra ou canhões de ar comprimido no mar, geram ondas sísmicas que se propagam pela subsuperfície. Ao encontrar interfaces entre camadas com diferentes impedâncias acústicas, parte da energia é refletida e registrada por receptores na superfície (Yilmaz, 2001).

No processamento, os dados passam por operações para atenuar ruídos, corrigir distorções geométricas e produzir a imagem da subsuperfície. Entre as operações de maior custo computacional, destacam-se a Migração Reversa no Tempo (RTM) (Baysal; Kosloff; Sherwood, 1983) e a Inversão de Forma de Onda Completa (FWI) (Tarantola, 1984).

Na interpretação, especialistas analisam dados como imagem sísmica, sismogramas e modelos de velocidade para identificar estruturas geológicas como falhas, dobras e reservatórios. Plataformas como o OpendTect oferecem o ambiente para essa etapa, com recursos de interpretação, visualização e extração de atributos sísmicos (dGB Earth Sciences, 2024a).

O Mamute é um sistema de processamento sísmico (Fernandes *et al.*, 2025). Ele implementa modelagem sísmica, RTM e FWI com ferramentas de computação de alto desempenho (HPC, do inglês: *High Performance Computing*), sendo executado em ambientes de processamento paralelo com suporte a OpenMP e MPI. (Barros *et al.*, 2025) identifica a integração do Mamute com plataformas de interpretação, como o OpendTect como um próximo passo para a ferramenta.

Atualmente, o Mamute é controlado exclusivamente via linha de comando. Isso obriga o usuário a alternar entre o ambiente de processamento e alguma ferramenta de interpretação sísmica ao longo do fluxo de trabalho, sem integração direta entre as ferramentas. Problema semelhante é reportado para outras ferramentas sísmicas de linha de comando, como o Seismic Unix (Lorenzo *et al.*, 2025; Ribeiro *et al.*, 2025).

Este trabalho propõe desenvolver um plugin nativo em C++ para o OpendTect que integra as ferramentas do Mamute à plataforma, permitindo ao usuário configurar, executar e acompanhar simulações sísmicas diretamente no ambiente de interpretação. A integração tem início pela modelagem sísmica visco-acústica, com extensão prevista para RTM e FWI, de modo a integrar

progressivamente todas as ferramentas do Mamute.

1.1 Problematização

O Mamute opera exclusivamente via linha de comando, exigindo que o usuário alterne entre ambientes distintos para configurar simulações, executar o processamento e visualizar os resultados. Esse fluxo fragmentado aumenta a chance de erros manuais e dificulta o uso por pesquisadores com menor experiência em terminal de comando.

O OpendTect oferece uma plataforma consolidada para interpretação sísmica, com suporte nativo a plugins em C++. No entanto, ainda não há uma integração direta com o Mamute que permita executar simulações sísmicas e analisar os resultados em um único ambiente.

Dessa forma, o problema deste trabalho é: como integrar as ferramentas de processamento sísmico do Mamute ao OpendTect para permitir configuração, execução, monitoramento e interpretação de simulações diretamente na plataforma.

1.2 Objetivos

Para solucionar a fragmentação do fluxo de trabalho entre o processamento e a interpretação sísmica, e facilitar o uso da ferramenta Mamute, esta seção define o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um plugin nativo em C++ para o OpendTect que integre as ferramentas do Mamute Seismic Tools à plataforma, permitindo a configuração, execução e visualização de simulações sísmicas diretamente no OpendTect.

1.2.2 Objetivos Específicos

1. Implementar a interface gráfica do plugin para configuração dos parâmetros de modelagem sísmica, estruturando uma arquitetura extensível que abra caminho para a futura integração dos métodos de RTM e FWI do Mamute.
2. Validar o plugin com dados sintéticos de modelagem sísmica visco-acústica.
3. Disponibilizar no plugin um fluxo integrado para configuração, execução, monitoramento e interpretação dos resultados gerados pelo Mamute.

1.3 Justificativa

A integração do Mamute ao OpendTect via plugin elimina a necessidade de alternância entre ambientes e de conversões intermediárias de dados, reduzindo erros operacionais e o tempo de configuração das simulações.

O Mamute é uma ferramenta de alto desempenho computacional que implementa métodos sísmicos avançados, como RTM e FWI. Usuários típicos de ferramentas de processamento e interpretação sísmica, como geofísicos e pesquisadores da área, costumam trabalhar em plataformas integradas e com interfaces visuais. Nesse contexto, a integração com o OpendTect amplia o alcance da ferramenta ao torná-la mais acessível para usuários que já utilizam essa plataforma como ambiente principal de trabalho.

Atualmente, o projeto está realizando a integração da ferramenta de modelagem, porém, a incorporação do Mamute ao OpendTect de RTM e FWI permanece como continuidade natural do trabalho, sem perder de vista o objetivo de integrar com o tempo todas as ferramentas do Mamute ao OpendTect.

1.4 Organização do Texto

O Capítulo 2 apresenta a revisão teórica, abordando os conceitos de sísmica de reflexão, modelagem sísmica, RTM e FWI para mostrar a relevância dos métodos sísmicos suportados pelo Mamute, além de introduzir o OpendTect e conceitos importantes para o desenvolvimento do *plugin*. O Capítulo 3 apresenta os trabalhos relacionados, analisando outras propostas focadas na resolução do problema de fragmentação no fluxo de trabalho sísmico. O Capítulo 4 descreve a metodologia de desenvolvimento e as decisões de integração do plugin ao OpendTect. O Capítulo 5 apresenta os resultados obtidos. O Capítulo 6 apresenta as conclusões do trabalho.

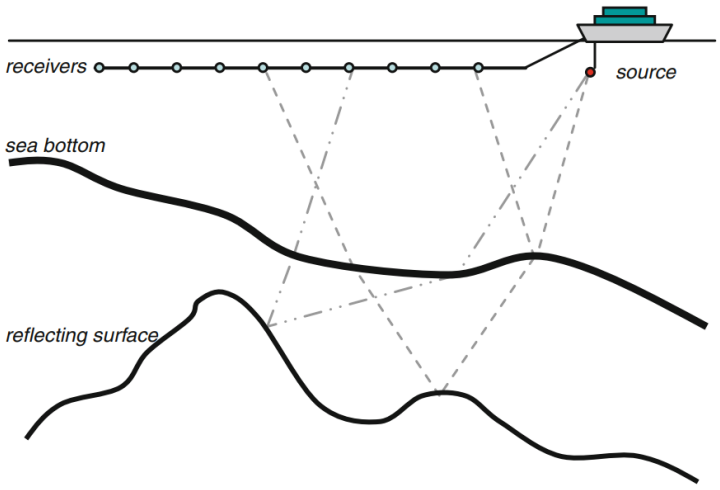
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para a compreensão do desenvolvimento do *plugin* de integração. A fundamentação inicia com os conceitos fundamentais dos métodos computacionais de sísmica de reflexão e os métodos computacionais de modelagem sísmica, RTM e FWI, contextualizando os algoritmos que o *plugin* visa tornar mais acessíveis. Em seguida, são descritas as ferramentas Mamute, OpendTect e os conceitos de integração que viabilizam a solução proposta neste trabalho.

2.1 Sísmica de Reflexão

A sísmica de reflexão é a principal técnica geofísica para imageamento da subsuperfície, utilizada na exploração de recursos como petróleo e gás natural (Yilmaz, 2001). O método consiste em três etapas: aquisição, processamento e interpretação dos dados.

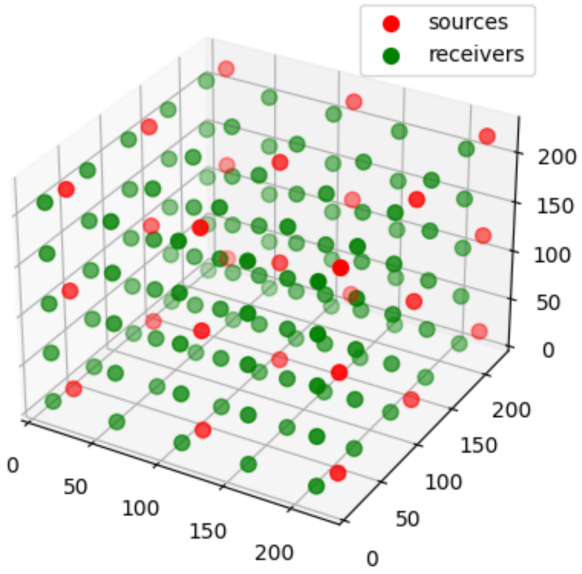
Figura 1 – Esquema de aquisição sísmica marítima evidenciando a propagação e reflexão das ondas.



Fonte: Adaptado de Panetta *et al.* (2012).

Na etapa de aquisição, uma fonte de energia artificial, como um canhão de ar no mar ou uma explosão em terra, gera uma onda sísmica que se propaga pela subsuperfície (Figura 1). Ao encontrar interfaces entre camadas com diferentes contrastes acústicos, parte da energia é refletida até a superfície e registrada por detectores como hidrofones ou geofones. Esse processo é repetido para múltiplas posições de fontes e receptores. O sinal de um único detector é chamado de traço sísmico; um conjunto de traços forma um sismograma (Yilmaz, 2001).

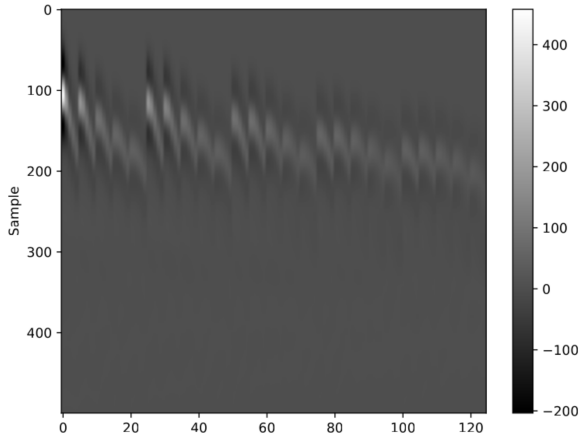
Figura 2 – Representação da malha da geometria de aquisição



Fonte: Adaptado de Fernandes *et al.* (2025).

A geometria de aquisição, que define posições de fontes e receptores, é um elemento fundamental para a qualidade dos dados coletados. Ela determina a cobertura e o ângulo de incidência das ondas sísmicas, influenciando diretamente a resolução e a fidelidade da imagem sísmica resultante (Yilmaz, 2001). A Figura 2 ilustra um exemplo de geometria de aquisição sísmica e a Figura 3 apresenta um exemplo de sismograma resultante.

Figura 3 – Representação de um sismograma sintético



Fonte: Adaptado de Fernandes *et al.* (2025).

Na fase de processamento, os dados registrados na aquisição passam por operações que atenuam ruídos, corrigem distorções geométricas e produzem uma imagem da subsuperfície. A

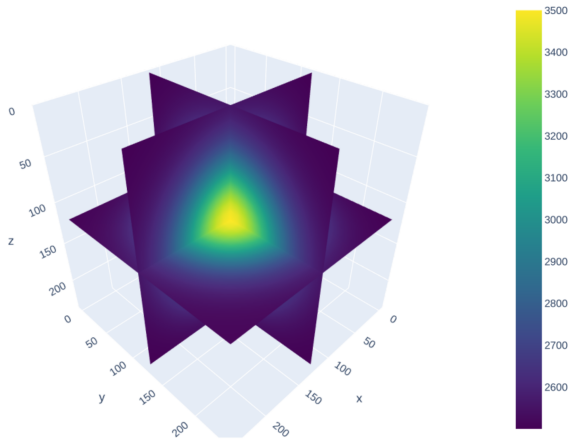
migração reposiciona eventos de reflexão em suas corretas localizações espaciais na subsuperfície, corrigindo distorções geométricas e colapsando difrações. Esses procedimentos possuem um custo computacional elevado para alguns métodos de processamento sísmico. Esse custo se dá a partir da necessidade de resolver numericamente equações diferenciais parciais que regem a propagação da onda sísmica e de aplicar métodos de otimização em grandes conjuntos de dados. Com isso, surge a necessidade de implementar essas operações em ambientes de computação de alto desempenho para lidar com a complexidade computacional e o grande volume de dados envolvidos. O Mamute é um exemplo de ferramenta desenvolvida para atender a essa demanda, empregando técnicas de programação paralela, como OpenMP e MPI, para otimizar o desempenho nessas infraestruturas (Fernandes *et al.*, 2025).

Na interpretação, especialistas identificam estruturas geológicas no dado migrado. Ferramentas como o OpendTect fornecem ambientes para essa etapa, com funcionalidades como o rastreamento de horizontes, extração de atributos sísmicos e visualização 3D (dGB Earth Sciences, 2024a).

2.2 Modelagem Sísmica

A modelagem sísmica é um dos métodos fundamentais de processamento sísmico. Ela constitui o motor numérico central que compõe métodos avançados de imageamento e inversão, como a RTM e a FWI. Nesse contexto, a modelagem é uma técnica utilizada para simular numericamente a propagação de ondas sísmicas em modelos computacionais da subsuperfície.

Figura 4 – Representação de um modelo de velocidades



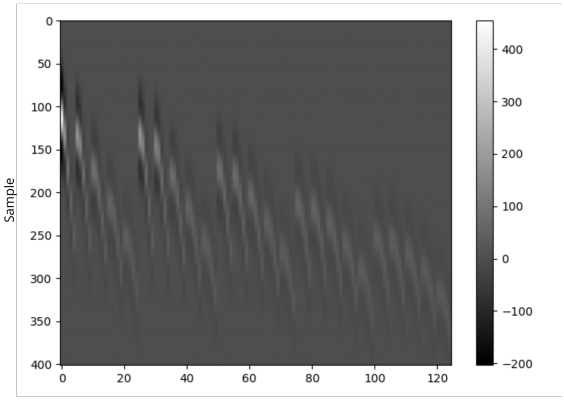
Fonte: Adaptado de Fernandes *et al.* (2025).

Para a simulação ocorrer, é necessário definir um modelo que represente as propriedades físicas da subsuperfície. O modelo de velocidades é o mais comum, que mapeia a velocidade de propagação da onda sísmica em cada ponto de uma malha discretizada. Contudo, para simulações mais realistas, outros modelos que incorporam propriedades físicas, como o fator de qualidade (utilizado na modelagem visco-acústica), são utilizados. Porém, a inclusão desses parâmetros, além de aumentar a fidelidade da simulação, também aumenta o custo computacional e o consumo de memória, exigindo a utilização de arquiteturas de processamento de alto desempenho com ferramentas como o Mamute para viabilizar a execução dessas simulações.

O objetivo é calcular o campo de deslocamento ou pressão provocado por uma fonte de energia ao longo de um modelo de propriedades. As equações diferenciais parciais que regem a propagação de ondas são resolvidas por métodos numéricos como diferenças finitas, elementos finitos ou espectrais (Virieux; Operto, 2009).

Como entrada, a modelagem recebe o modelo de velocidades, a geometria de aquisição, os parâmetros da fonte sísmica e a assinatura do sinal, além dos seus respectivos parâmetros de posicionamento. Como saída, produz traços sísmicos sintéticos organizados em sismogramas. No contexto deste trabalho, a modelagem sísmica é a primeira operação do Mamute integrada ao OpendTect via *plugin*. A Figura 4 apresenta um exemplo de modelo de velocidades e a Figura 5 apresenta o sismograma sintético obtido.

Figura 5 – Representação de um sismograma sintético obtido por modelagem visco-acústica.



Fonte: Adaptado de Fernandes *et al.* (2025).

2.2.1 Diferença entre Modelagem Acústica e Visco-acústica

A simulação computacional da propagação de ondas pode ser realizada com diferentes níveis de precisão. A modelagem acústica é uma aproximação mais simples que trata a subsuperfície como um meio ideal, onde a onda viaja sem perder energia por atrito (Yilmaz, 2001). Para executar esse tipo de simulação, o software exige basicamente um único modelo de entrada principal: o modelo de velocidades.

Por outro lado, a modelagem visco-acústica é uma representação mais realista, ao considerar que as rochas absorvem e dissipam parte da energia da onda sísmica, fenômeno conhecido como atenuação. Para modelar essa perda de energia numericamente, são frequentemente utilizados métodos matemáticos como o método τ (Blanch; Robertsson; Symes, 1995). A execução passa a exigir um segundo arquivo obrigatório: o modelo do Fator de Qualidade (Qp, do inglês: quality factor). O Qp é uma medida adimensional que quantifica a atenuação do meio, sendo inversamente proporcional à fração de energia dissipada por ciclo de onda. Quanto menor for o valor de Qp, maior a atenuação, enquanto valores elevados indicam meios onde a onda se propaga com pouca perda de energia. Esse modelo pode ser gerado através do modelo de velocidades, relação empírica descrita por (Li, 2017).

$$Q_p = 3{,}516 \times 10^{-6} \cdot V_p^{2,2} \tag{2.1}$$

onde:

- Q_p é o fator de qualidade do meio (adimensional);
- v é a velocidade de propagação da onda sísmica compressional em cada ponto da malha, dada em metros por segundo (m/s).

2.3 Migração Reversa no Tempo

A Migração Reversa no Tempo (RTM, do inglês *Reverse Time Migration*) é um algoritmo de migração sísmica proposto por Baysal, Kosloff e Sherwood (1983) e baseado na solução numérica bidirecional da equação da onda. Diferentemente de métodos de migração anteriores, a RTM destaca-se por ser adequada para o imageamento de estruturas geológicas com geometria complexa, o que a torna adequada para imageamento de estruturas geológicas com geometria complexa e meios com forte variação lateral de velocidade (Yilmaz, 2001).

Para cada tiro sísmico, a RTM executa três etapas. Na primeira, o campo de ondas da fonte é propagado para frente no tempo a partir da posição da fonte. Em seguina, na segunda, os dados registrados nos receptores são retropropagados para trás no tempo. Enfim, na terceira, uma condição de imagem é aplicada para identificar os refletores: nos pontos da malha onde ambos os campos de onda coincidem no tempo, há um refletor, e esses pontos formam a imagem sísmica (Claerbout, 1985).

A RTM requer armazenar ou reconstruir o campo de ondas da fonte em todos os instantes de tempo para aplicar a condição de imagem, o que impõe custo elevado de memória e processamento. Técnicas de *checkpointing* reduzem esse custo, armazenando o campo em instantes selecionados e reconstruindo os demais quando necessário (Symes, 2007).

2.4 Inversão de Forma de Onda Completa (FWI)

A Inversão de Forma de Onda Completa (FWI, do inglês *Full Waveform Inversion*) é um método iterativo de otimização que estima o modelo de velocidades da subsuperfície (Tarantola, 1984; Virieux; Operto, 2009). O método parte de um modelo inicial e, a cada iteração, executa uma modelagem sísmica para gerar dados sintéticos. A diferença entre esses dados sintéticos e os dados sísmicos observados é utilizada para atualizar o modelo na direção que reduz esse erro. O processo se repete até que o resíduo atinja um critério de convergência.

O gradiente que orienta a atualização do modelo pode ser calculado pelo método do estado adjunto, que obtém essa informação com apenas duas simulações da equação da onda por tiro sísmico: uma direta, para gerar os dados sintéticos, e uma reversa, para propagar os resíduos de volta ao modelo (Plessix, 2006). A atualização pode ser realizada por métodos como o Gradiente Conjugado ou um método Quasi-Newton, que aceleram a convergência em relação ao gradiente descendente simples (Virieux; Operto, 2009).

Por executar ao menos duas modelagens sísmicas completas (direta e reversa) a cada iteração e para cada tiro sísmico, a FWI é um método computacionalmente intensivo. Essa característica torna necessária sua implementação em sistemas computacionais de alto desempenho.

2.5 Mamute

O Mamute é uma ferramenta para processamento sísmico em ambientes de alto desempenho computacional (HPC). O Mamute implementa modelagem sísmica, Migração Reversa no

Tempo (RTM) (Baysal; Kosloff; Sherwood, 1983) e Inversão de Forma de Onda Completa (FWI) (Tarantola, 1984; Virieux; Operto, 2009), com suporte aos modelos de propagação de ondas: acústica e visco-acústica. A ferramenta é de código aberto e está disponível em repositório público[1], e sua arquitetura, bem como as características de implementação, estão detalhadas no *preprint* de Fernandes *et al.* (2025).

Para viabilizar a execução de simulações que exigem alto custo computacional e grande consumo de memória, o Mamute explora o paralelismo em múltiplos níveis. Desenvolvido em C++, o sistema utiliza OpenMP para implementar o paralelismo de memória compartilhada (distribuindo o processamento das malhas entre as *threads*) e o MPI para o paralelismo de memória distribuída. Essa combinação permite que o Mamute seja executado em uma variedade de arquiteturas HPC, desde nós de um supercomputador até *clusters* (Silva *et al.*, 2025).

A arquitetura de execução do Mamute é orientada por arquivos de configuração. Para executar uma simulação, o usuário deve fornecer um conjunto de arquivos de entrada que incluem:

- Um arquivo de configuração principal (`modeling.toml`) que contém os parâmetros numéricos e caminhos dos arquivos de entrada e saída;
- Arquivos binários que representam as propriedades físicas da subsuperfície, como o modelo de velocidades (`velocity_model.bin`) e o modelo de fator de qualidade (`qp_model.bin`);
- Arquivos de geometria de aquisição como a posição da fonte (`src_coord.bin`) e dos receptores (`rcv_coord_*.bin`);
- Arquivos estruturados (como `src_rcv.json`) descrevendo a geometria de aquisição e (`vel.json`) descrevendo o modelo de velocidades;
- O arquivo contendo a assinatura da fonte sísmica (`source.bin`).

Após o processamento, o sistema gera como saída os sismogramas sintéticos em formato binário (nomeados como `dobs_*.bin`) correspondentes a cada tiro sísmico.

No Mamute, o controle do fluxo de execução ocorre estritamente via linha de comando. Na prática, a preparação dos dados exige a execução de scripts em Python auxiliares (por exemplo, `mamute.py data.generate_vel.py`). A execução da modelagem sísmica é realizada manualmente por meio de um comando específico (`mamute modeling`). Por fim, a visualização dos resultados depende da execução de scripts adicionais para ler e plotar os dados binários gerados (como: `mamute.py data.plot_seismogram.py`).

[1] https://lappsufrn.gitlab.io/mamute/dev/

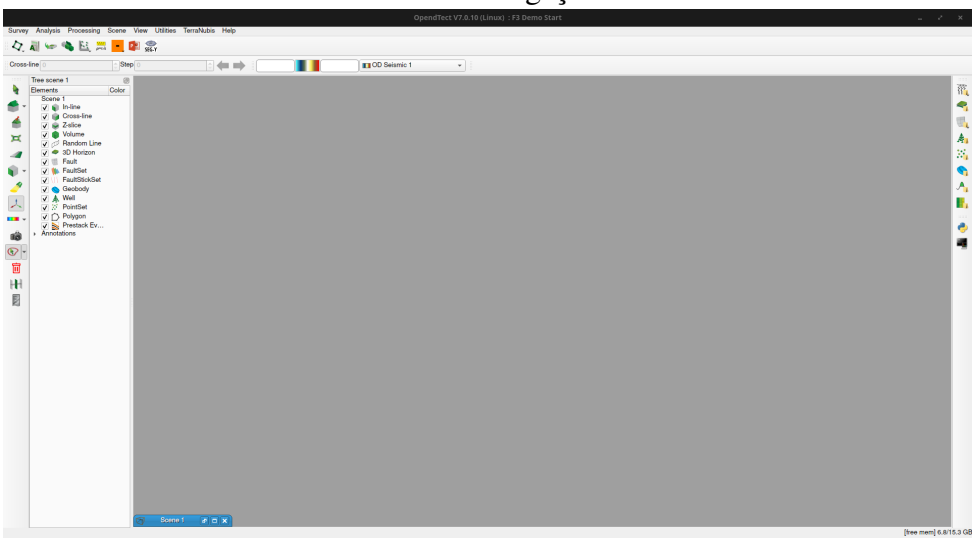
Embora essa abordagem seja adequada para usuários experientes, ela impõe uma barreira de usabilidade para geofísicos e profissionais da área. A necessidade de editar configurações em arquivos de texto, executar múltiplos comandos e alternar entre ambientes de processamento e interpretação sem integração direta pode ser um desafio. Essa limitação de interface evidencia a necessidade de uma solução que integre o Mamute a uma plataforma gráfica consolidada na indústria para interpretação de dados, como OpendTect (dGB Earth Sciences, 2024a).

2.6 OpendTect

O OpendTect é uma plataforma de interpretação sísmica de código aberto utilizada na indústria do petróleo e na academia (dGB Earth Sciences, 2024a). Antes de se consolidar como um ambiente de desenvolvimento extensível, o *software* destaca-se por fornecer um robusto conjunto de ferramentas nativas para a visualização e interpretação de dados geofísicos.

Entre suas funcionalidades, destacam-se a visualização de volumes sísmicos em 2D e 3D, o rastreamento de horizontes, a extração de atributos sísmicos, a análise de poços e muitas outras (dGB Earth Sciences, 2024a). Outro ponto relevante é que o OpendTect possui código aberto sob a licença GNU GPL, permitindo o acesso a recursos avançados de processamento de dados para pesquisadores e estudantes. A Figura 6 ilustra a interface gráfica padrão do sistema.

Figura 6 – Interface de interpretação do OpendTect exibindo um volume sísmico e ferramentas de navegação.



Fonte: Autoria própria (2026).

A flexibilidade do OpendTect permite a extensão de suas funcionalidades por meio de um sistema de *plugins* em C++. Esses *plugins* podem ser desenvolvidos para adicionar novas

funcionalidades, integrar ferramentas externas ou personalizar a interface de acordo com as necessidades do usuário (dGB Earth Sciences, 2024b). Esse sistema de *plugins* é implementado por meio de bibliotecas de carregamento dinâmico como `.dll` no Windows e `.so` no Linux, permitindo que cada *plugin* seja desenvolvido, distribuído e atualizado independentemente do núcleo do sistema (dGB Earth Sciences, 2024b; dGB Earth Sciences, 2024c).

A arquitetura do OpendTect classifica os *plugins* como *EARLY* ou *LATE*. Os plugins *EARLY* são responsáveis por tarefas de *backend*, como operações de processamento de dados, cálculos matemáticos e gerenciamento de arquivos sem interação visual direta, sendo carregados antes da interface gráfica. Os *plugins LATE* são responsáveis por criar a interface com o usuário, sendo carregados após a interface estar disponível. O ecossistema de *plugins* do OpendTect já conta com extensões de terceiros que integram ferramentas externas à plataforma (Izzatdin; Jaafar; Gilal, 2017).

3 TRABALHOS RELACIONADOS

No contexto da integração entre ferramentas de processamento sísmico e interfaces gráficas, alguns trabalhos relevantes foram identificados na literatura. Ribeiro *et al.* (2025) propõem o BotoSeis-web, uma aplicação web para processamento de dados sísmicos que encapsula os programas do Seismic Unix em uma interface com visualizador integrado. O sistema elimina a necessidade de uso direto da linha de comando ao organizar o fluxo de processamento por meio de uma API de *backend* e um *frontend* interativo. O trabalho parte do mesmo problema abordado neste trabalho: ferramentas de processamento sísmico controladas por linha de comando impõem barreiras de uso a pesquisadores e profissionais da indústria sem experiência nesse ambiente. A diferença em relação a este trabalho é a arquitetura adotada: BotoSeis-web utiliza uma interface web, enquanto o *plugin* proposto aqui é nativo ao OpendTect, integrando processamento e interpretação em um único ambiente de interpretação sísmica 3D.

Lorenzo *et al.* (2025) apresentam o SeismicUnixGui (SUG), uma interface gráfica *open-source* para os programas do Seismic Unix, desenvolvida para permitir que usuários sem experiência realizem processamento sísmico. O SUG utiliza *wrappers* desenvolvidos na linguagem Perl para encapsular os módulos do Seismic Unix. Através da interface, o SUG gera e executa automaticamente os comandos necessários para processar os dados sísmicos, eliminando a necessidade de interação direta com a linha de comando. A principal diferença em relação a este trabalho é que o SeismicUnixGui é uma aplicação independente, enquanto o *plugin* proposto aqui é uma integração ao OpendTect.

Cabieces *et al.* (2022) apresentam o ISP (do inglês: *Integrated Seismic Program*, ou Programa Sísmico Integrado), um software com interface gráfica que integra múltiplas ferramentas de sismologia em um único ambiente. O ISP é desenvolvido como uma aplicação independente escrita em Python, utilizando bibliotecas como PyQt para a interface gráfica e ObsPy para a manipulação de dados, encapsulando e executando códigos de terceiros sob um mesmo ambiente. A distinção em relação a esta pesquisa segue a mesma linha do SUG: o ISP é uma aplicação independente, enquanto este trabalho propõe um *plugin* integrado ao OpendTect, unificando o processamento e a interpretação no ambiente de trabalho do geofísico.

Izzatdin, Jaafar e Gilal (2017) analisam o OpendTect em relação à sua compatibilidade com outros pacotes de processamento sísmico, discutindo como a arquitetura extensível da plataforma permite integrar ferramentas externas via *plugins*. O trabalho demonstra que o OpendTect é uma plataforma consolidada para integração processamento-interpretação no contexto

acadêmico.

O DUG Insight (DUG Technology, 2023) unifica processamento sísmico, imageamento e interpretação em uma única plataforma com integração a *clusters* HPC, sem que o usuário precise alternar entre ferramentas. Este trabalho aplica o mesmo conceito no contexto *open-source*, integrando o Mamute ao OpendTect via *plugin* nativo.

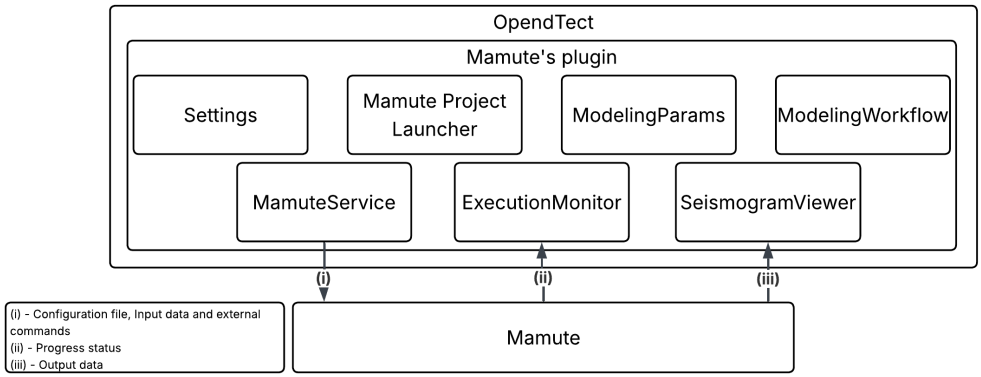
4 METODOLOGIA

Este capítulo descreve a metodologia de desenvolvimento do *plugin* de integração do Mamute ao OpendTect. Inicialmente, a arquitetura e o fluxo de trabalho aqui apresentados foram estruturados com foco na modelagem visco-acústica. Contudo, devido aos parâmetros operacionais em comum, essa mesma lógica servirá de base para a integração das futuras aplicações da RTM e da FWI. A Seção 4.1 descreve os componentes e responsabilidades do *plugin*, enquanto a Seção 4.2 discute as principais decisões de projeto tomadas durante o desenvolvimento.

4.1 Arquitetura do *Plugin*

A Figura 7 apresenta um esquema das interações entre o OpendTect, o *plugin* e o Mamute, indicando onde os dados são gerados e lidos.

Figura 7 – Esquema das interações entre OpendTect, *plugin* MamutePlugin e Mamute.



Fonte: Autoria própria (2026).

O *plugin* MamutePlugin é organizado em oito componentes principais. Para a configuração global do ambiente, o primeiro componente é o `Settings`, que fornece um diálogo simples para a definição do caminho de instalação do Mamute.

O componente `Mamute Project Launcher` gerencia o diretório do projeto. Ele oferece as funcionalidades de criar projetos novos, copiar e abrir projetos existentes, além de fornecer uma lista de projetos abertos recentemente.

O componente `ModelingParams` é um conjunto de estruturas de parâmetros, definidas no *namespace* Mamute. A estrutura contém os parâmetros da simulação: dimensões da malha,

espaçamento, origem, arquivos do modelo de velocidade e do modelo Qp (fator de qualidade), parâmetros da fonte sísmica, número de fontes, número de receptores e diretório do projeto.

O componente `MamuteService` é uma classe de utilidade que fornece métodos para execução de comandos externos ao sistema operacional. Este componente é o responsável direto por chamar tanto os scripts Python auxiliares quanto os executáveis do Mamute em C++, além de gerenciar a escrita dos arquivos de configuração. O componente `ModelingWorkflow` utiliza os parâmetros de configuração necessários do `ModelingParams` e aciona o `MamuteService` para executar a rotina de geração dos arquivos binários dos modelos da geometria de aquisição, velocidade, Qp, assinatura da fonte e do arquivo de configuração, além de coordenar a execução da modelagem visco-acústica.

O componente `ExecutionMonitor` monitora a execução da modelagem em tempo real. Esse monitor funciona como uma janela de progresso da execução. Ele implementa a leitura incremental do arquivo de *log*, monitorando o *status* da execução e disponibilizando os indicadores de progresso (*Waiting for log file*, *Connected* e *Process finished*). O fechamento dessa janela não interrompe a modelagem: o usuário pode reabri-la a partir do botão `Monitor` na interface principal.

O componente `SeismogramViewer` é o responsável por identificar os arquivos de saída da execução no diretório do projeto. Ele permite que o usuário selecione os sismogramas no diretório do projeto, veja algumas informações sobre aquele arquivo, como uma representação gráfica 2D da malha da geometria de aquisição e a visualização dos sismogramas direto do plugin para o OpendTect.

O componente da camada de interface gráfica é implementado com as classes do módulo `uiBase` do OpendTect. Esse componente fornece a interface principal da janela de trabalho, que integra as páginas *Acquisition Geometry*, *Velocity Model* e *Visco-Acoustic Modeling* em uma interface unificada. A interface inclui a verificação dos parâmetros de entrada, gerenciamento de etapas do fluxo de trabalho e funcionalidades de gerenciamento de projeto, como abrir projetos recentes, criar novos projetos, copiar projetos e selecionar o diretório do projeto. A integração fornece um diálogo que verifica automaticamente os sismogramas disponíveis no diretório do projeto, permitindo a seleção e visualização dos arquivos de saída.

4.2 Características Arquiteturais do Plugin

O desenvolvimento do *plugin* passou por revoluções arquiteturais relevantes, motivadas pela necessidade de ampliar as funcionalidades oferecidas sem comprometer a usabilidade da interface. Esta seção descreve as principais características arquiteturais da implementação proposta neste trabalho.

4.2.1 Persistência das Configurações de Ambiente

Para otimizar o fluxo de trabalho e evitar reconfigurações repetitivas do sistema, tornou-se necessária a persistência das configurações de ambiente. A decisão de projeto consistiu em externalizar o caminho de instalação do software em uma estrutura de configuração global. Dessa forma, o sistema coleta esse dado uma única vez e, persistentemente, o mantém entre as diferentes sessões de uso, até que seja alterado.

4.2.2 Módulo de Gerenciamento de Projetos

Para garantir uma melhor separação de responsabilidades no *plugin*, estabeleceu-se o requisito de desacoplar o gerenciamento dos repositórios das configurações da modelagem. Para atender a essa necessidade, foi feito um módulo independente de gerenciamento, atuando como um inicializador de projetos. Esse módulo concentra somente as operações de abertura de projetos recentes, criação de novos projetos, cópia de novos projetos e seleção do diretório de trabalho. Com isso, separamos bem as responsabilidades de cada janela da interface.

4.2.3 Fluxo de Trabalho

Da mesma forma que o Módulo de Gerenciamento de Projetos, foi feita uma separação de responsabilidades na interface do fluxo de trabalho. A geração de artefatos como os arquivos dos modelos de velocidade e geometria de aquisição é realizada em páginas diferentes da execução da modelagem visco-acústica. Na interface principal, foi adicionada uma barra lateral onde é possível selecionar as funcionalidades implementadas. Com essa estrutura, é possível separar as etapas de pré-processamento (configuração e geração da geometria de aquisição e do modelo de velocidade) da etapa de processamento que concentra os parâmetros de entrada da simulação e sua execução.

4.2.4 Extensibilidade como critério de projeto

A organização da interface principal também foi guiada pela necessidade de extensibilidade futura. Ao integrar novas funcionalidades do Mamute, como RTM e FWI, é necessário somente adicionar as novas páginas à barra lateral. Essas futuras etapas poderão reutilizar diretamente os artefatos já gerados pelas páginas existentes: a geometria de aquisição, o modelo de velocidade e os sismogramas sintéticos produzidos pela modelagem visco-acústica são entradas comuns a RTM e FWI, de modo que a execução sequencial das etapas na interface principal reduz retrabalho e mantém a consistência dos dados entre as fases do processamento.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos do desenvolvimento do *plugin* de integração do Mamute ao OpendTect. Os recursos implementados pela integração são detalhados, demonstrando como o sistema gerencia a configuração dos parâmetros, a geração dos arquivos de entrada, o monitoramento e a visualização dos resultados da modelagem sísmica.

Como resultado do desenvolvimento, o *plugin* adota um modelo de navegação baseado em uma interface orientada a tarefas, com páginas dedicadas para configuração da geometria de aquisição, modelo de velocidade e parâmetros de modelagem visco-acústica. No nível de integração, a arquitetura mantém a separação clara entre a camada de interface, a camada de preparação dos arquivos e parâmetros de entrada, como o arquivo de configuração, modelo de velocidade, geometria de aquisição, assinatura da fonte e modelo de atenuação, e a camada de execução e monitoramento da modelagem.

5.1 Evolução Arquitetural da Solução

A interface do *plugin* adota um modelo de navegação baseado em `Workflow Navigator`, composto por páginas dedicadas para `Acquisition Geometry`, `Velocity Model` e `Visco-Acoustic Modeling`. Essa reorganização reduziu a complexidade de interação e tornou o fluxo de preparação da modelagem mais explícito ao usuário. A evolução aqui consolidada refere-se à transição de uma interface com uma única tela para uma estrutura modularizada e distribuída em camadas.

No nível de integração, a execução permanece baseada no `MamuteService` que, gerenciado pelo `ModelingWorkflow`, é responsável por orquestrar a geração dos arquivos de entrada, escrita de arquivos de configuração e execução do Mamute. Como consequência, a arquitetura atual mantém separação clara entre:

- Camada de interface (configuração e validação): Responsável pela captura dos dados digitados pelo usuário e pela validação em tempo real dos tipos de dados (como impedir valores negativos ou campos vazios em dimensões de malha e espaçamentos), gerenciando o estado dos botões de execução conforme a integridade dos campos;
- Camada de preparação de artefatos: Encarregada de estruturar e salvar os arquivos físicos de parametrização e dados operacionais (como `modeling.toml`, `vel.json` e `src_rcv.json`), traduzindo o estado da interface para os formatos interpretáveis pelo motor de simulação;

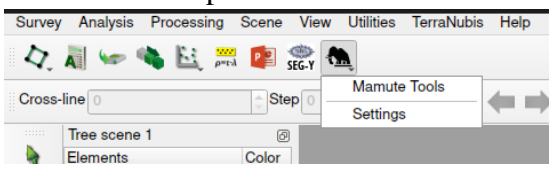
- Camada de execução e monitoramento da modelagem: Responsável por gerenciar o ciclo de vida do processo externo do Mamute, controlando o disparo assíncrono, a leitura contínua dos arquivos de `log` e a interceptação de erros de execução.

Essa separação favorece a manutenção e a evolução incremental da solução, preservando compatibilidade.

5.2 Resultados Funcionais do Fluxo de Uso

O fluxo de utilização do *plugin* pelo usuário final segue os passos descritos a seguir:

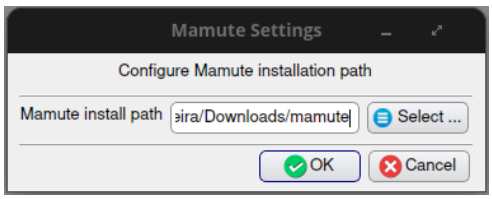
Figura 8 – Botão `Mamute Seismic Tools` na barra de ferramentas do OpendTect.



Fonte: Autoria própria (2026).

1. Abrir o OpendTect. O *plugin* MamutePlugin é carregado automaticamente e adiciona o menu *Mamute Seismic Tools* e um botão na barra de ferramentas, conforme a Figura 8.

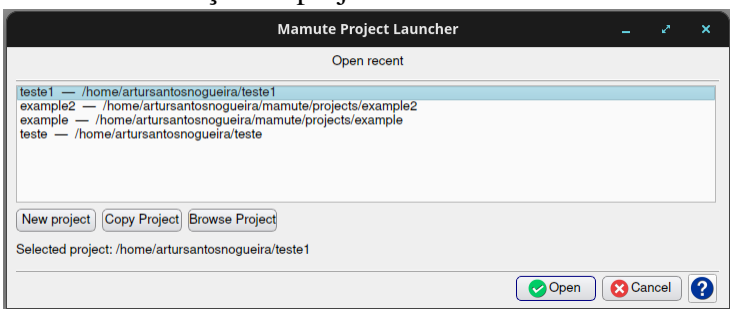
Figura 9 – Tela de configuração do caminho de instalação do Mamute.



Fonte: Autoria própria (2026).

2. Na primeira utilização, o usuário deve acessar `Settings` no menu do *plugin* para configurar o caminho de instalação do Mamute, como apresentado na Figura 9. Essa configuração é persistida entre sessões por meio da API `Settings` do OpendTect, no arquivo ~/.od/settings_Mamute, eliminando a necessidade de reconfiguração repetitiva a cada abertura do OpendTect.
3. No `Mamute Project Launcher`, o usuário gerencia o ambiente de trabalho escolhendo

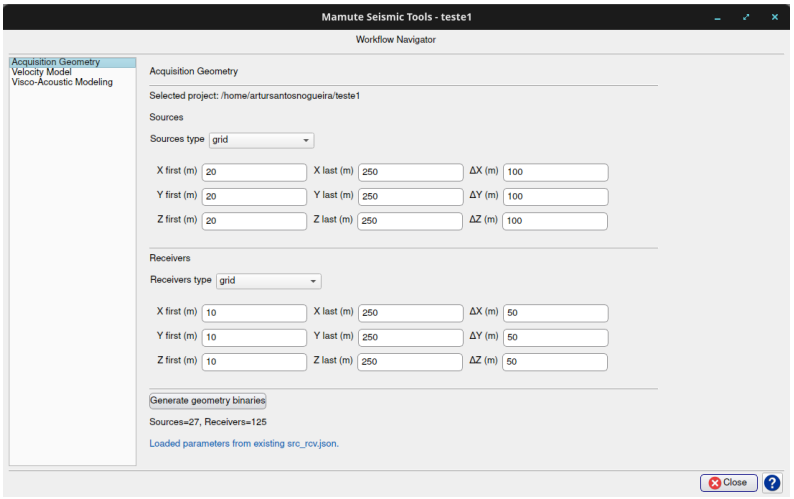
Figura 10 – `Mamute Project Launcher` para seleção e criação de projetos.



Fonte: Autoria própria (2026).

entre: abrir um projeto existente, criar um novo projeto, copiar um projeto existente ou selecionar um projeto recente, conforme ilustrado na Figura 10.

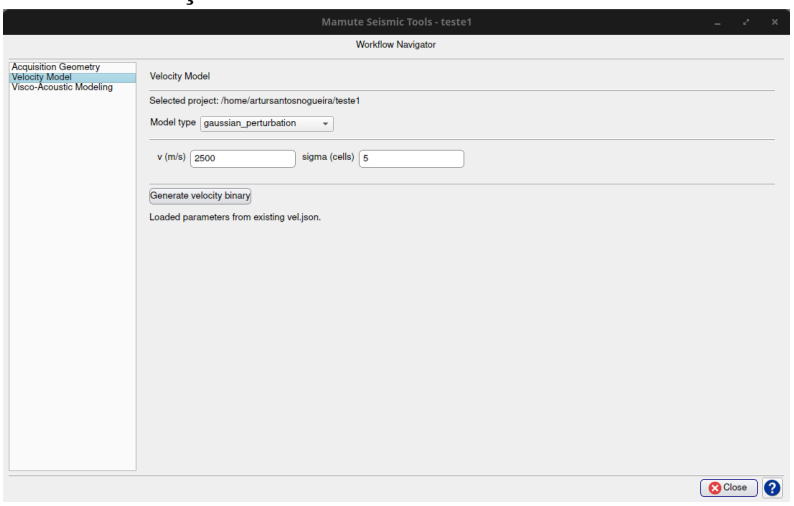
Figura 11 – Página `Acquisition Geometry` para configuração da geometria de aquisição.



Fonte: Autoria própria (2026).

4. Na janela principal, configuram-se inicialmente os parâmetros da aba `Acquisition Geometry` e aciona-se `Generate geometry binaries`, conforme a Figura 11. Nessa etapa, a interface verifica as dimensões informadas, armazena as configurações no arquivo `src_rcv.json` e gera os arquivos binários da geometria. Cabe destacar que esta etapa é opcional: caso o usuário já disponha de arquivos binários da geometria previamente, ele pode avançar no fluxo sem a necessidade de nova geração.
5. A etapa consiste em configurar a aba `Velocity Model` e acionar `Generate velocity binary`, como ilustrado na Figura 12. Nessa etapa, a interface valida os parâmetros do tipo de modelo de velocidade escolhido, grava internamente o arquivo `vel.json` e gera o

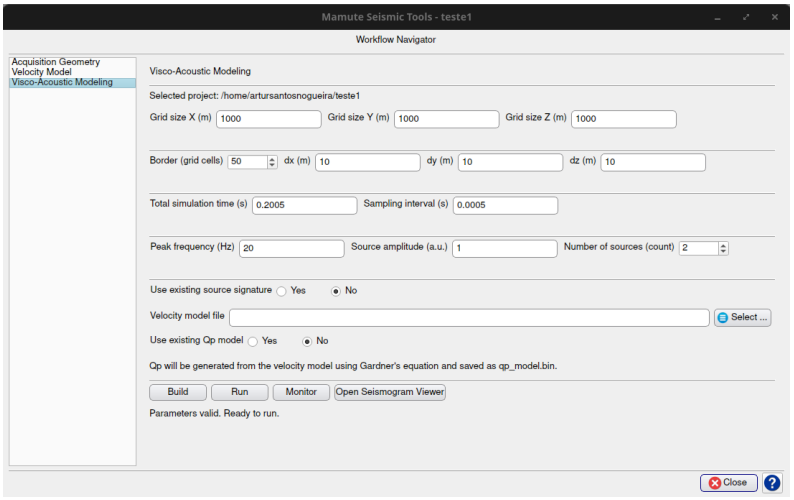
Figura 12 – Página `Velocity Model` para configuração e geração do modelo de velocidade.



Fonte: Autoria própria (2026).

arquivo binário referente ao modelo de velocidades.

Figura 13 – Página `Visco-Acoustic Modeling` para configuração da simulação.



Fonte: Autoria própria (2026).

6. Na aba `Visco-Acoustic Modeling`, o usuário deve preencher os parâmetros necessários para iniciar a modelagem, incluindo dimensões da malha, espaçamentos, parâmetros da fonte e tempo total, conforme a Figura 13.
7. Com os parâmetros preenchidos, o usuário deve clicar no botão `Build` para iniciar o processo de preparação e consolidação. O *plugin* realiza uma validação para verificar se os arquivos binários necessários, como o da geometria de aquisição e o do modelo de velocidade, estão presentes no projeto. Se algum estiver faltando, o processo é interrompido

e uma mensagem de erro é exibida. Caso os arquivos estejam presentes, o botão executa sequencialmente as seguintes etapas de preparação:

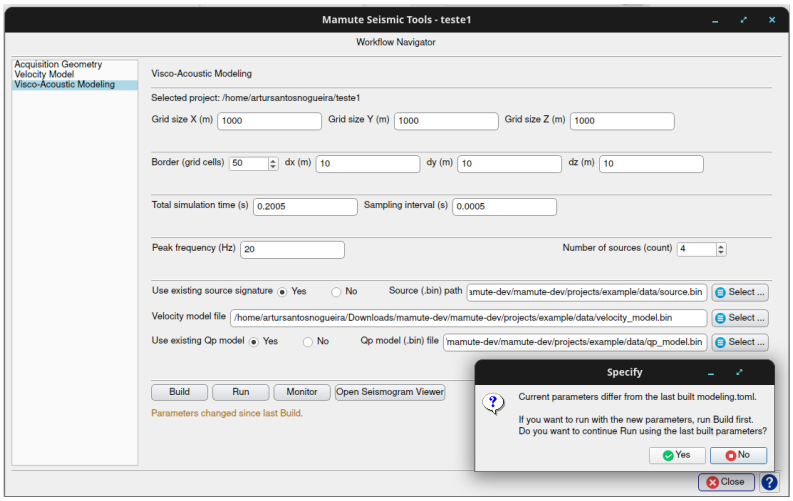
a) `Save Parameters`: grava o arquivo `modeling.toml` com os parâmetros da simulação;

b) `Source`: gera a *wavelet* da fonte sísmica caso não esteja sendo utiliza uma fonte já existente;

c) `Velocity`: valida o modelo de velocidade presente no projeto;

d) `Qp`: gera ou utiliza o modelo Qp existente.

Figura 14 – Página `Visco-Acoustic Modeling` chamando `Run` após alterar parâmetros sem executar `Build`.

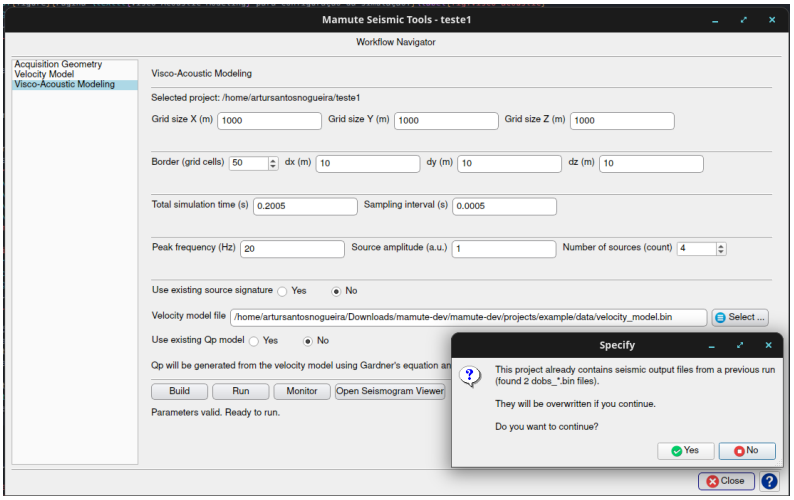


Fonte: Autoria própria (2026).

8. Após a execução do `Build`, qualquer modificação nos parâmetros invalida o estado da preparação. Se o usuário tentar iniciar a simulação pelo botão `Run`, o sistema exibirá um alerta recomendando uma nova execução do `Build` para garantir que as alterações sejam aplicadas.
9. Com os parâmetros validados, o botão `Run` executa a modelagem sísmica. O *plugin* invoca o Mamute como processo externo. Antes do disparo, os arquivos binários de saída de simulações anteriores (`dobs_*.bin`) presentes na pasta do projeto são removidos mediante confirmação do usuário, assim evitando mistura de resultados de execuções anteriores.
10. Ao iniciar a modelagem, o monitor de progresso (`ExecutionMonitor`) é aberto automaticamente, conforme a Figura 16, implementando a leitura do arquivo de `log` gerado pelo Mamute em tempo real.

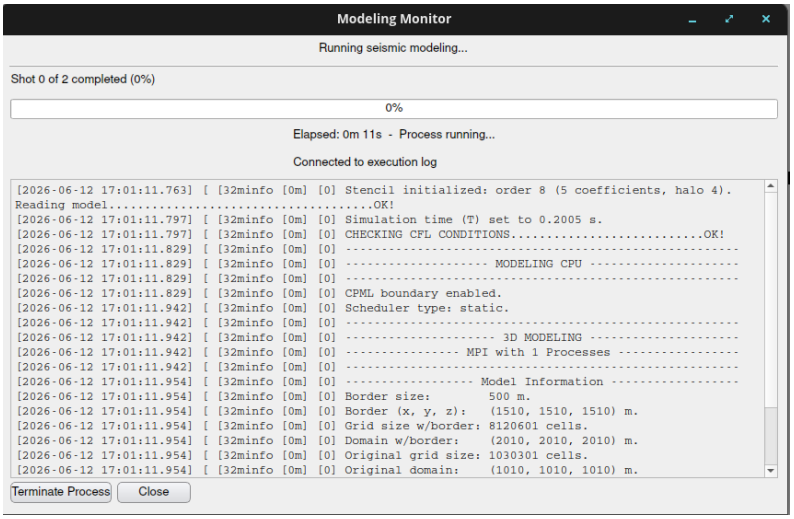
    Ele se torna a principal janela de acompanhamento da execução e implementa leitura do

Figura 15 – Página `Visco-Acoustic Modeling` com aviso da sobrescrita dos arquivos `dobs_*.bin` já existentes.



Fonte: Autoria própria (2026).

Figura 16 – Monitor de progresso da execução da modelagem.



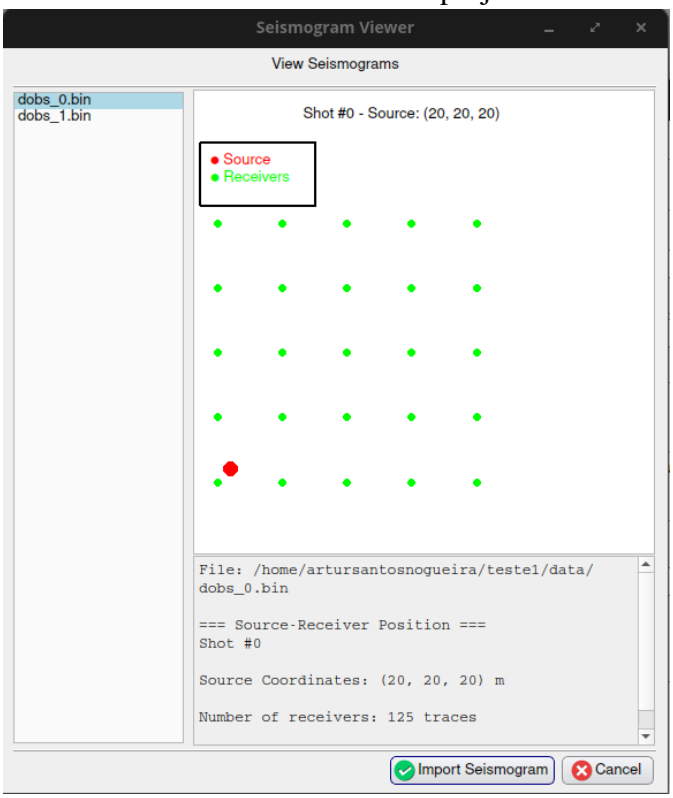
Fonte: Autoria própria (2026).

arquivo de *log*, exibindo a saída do Mamute em tempo real. Um indicador de `status` na parte superior do monitor mostra:

- `Waiting for log file...`: processo iniciado, mas o log ainda não foi criado;
- `Connected`: log conectado e sendo lido em tempo real;
- `Process finished`: processo concluído, com sucesso ou com falha.

O usuário pode fechar essa janela sem encerrar a modelagem. Quando isso ocorre, a execução continua em segundo plano, e o monitor pode ser reaberto pelo botão `Monitor` na janela principal da modelagem.

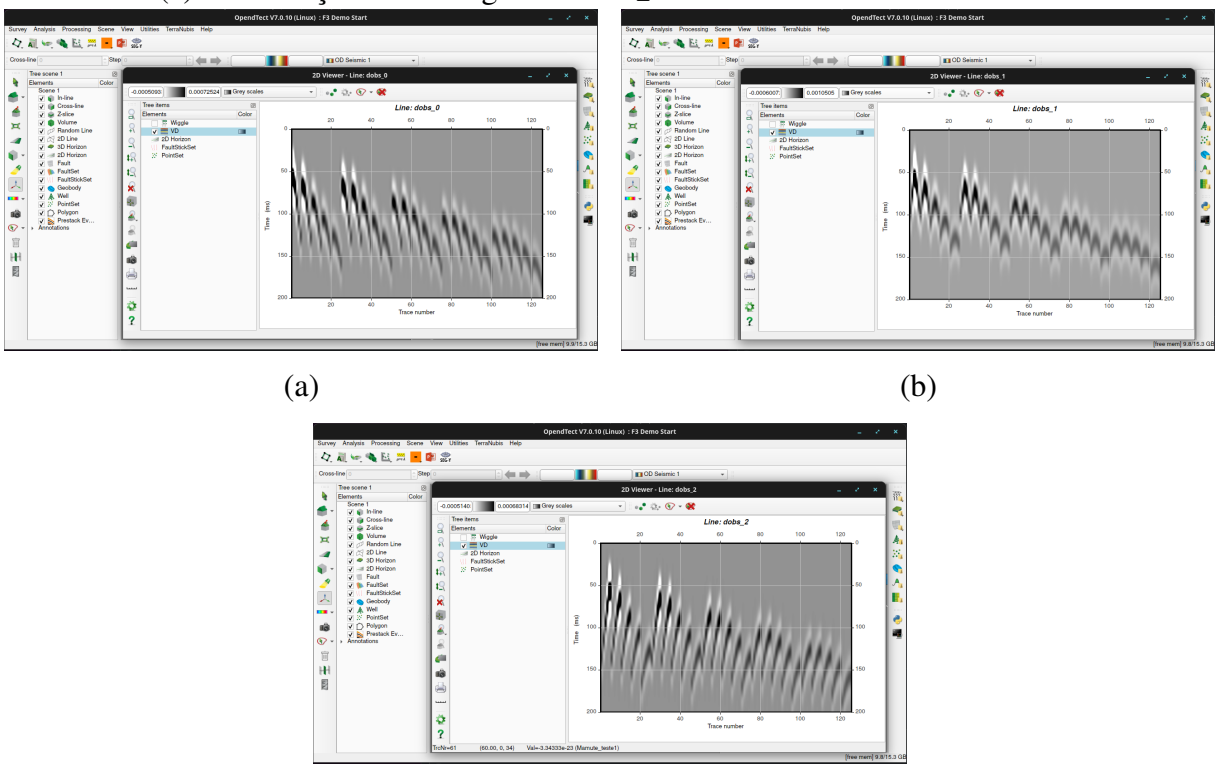
Figura 17 – Janela de seleção de sismogramas disponíveis no diretório do projeto.



Fonte: Autoria própria (2026).

11. Após a conclusão da modelagem, o `Seismogram Viewer` é aberto automaticamente pelo monitor, quando a presença dos arquivos no diretório do projeto é detectada e a modelagem considerada encerrada. O visualizador, então, realiza uma busca automática de todos os arquivos `dobs_*.bin` presentes no diretório do projeto. O botão `Open Seismogram Viewer` na janela principal também pode ser acionado a qualquer momento para abrir o visualizador manualmente.
    - Exibe a geometria de aquisição (posições de fonte e receptores) graficamente;

Figura 18 – (a) visualização do sismograma `dobs_0`, (b) visualização do sismograma `dobs_1` e (c) visualização do sismograma `dobs_2`.



(a) (b)

(c)

Fonte: Autoria própria (2026).

- Oferece opções para importar o sismograma para o OpendTect e plotá-lo em um visualizador 2D.

12. Após o usuário selecionar um sismograma e clicar em `Plot Seismogram`, o visualizador 2D é aberto, exibindo o sismograma selecionado, como ilustrado na Figura 18.

5.3 Síntese dos Resultados Obtidos

Em síntese, os resultados indicam que a integração Mamute–OpendTect transformou a experiência de uso do Mamute e aprimorou o fluxo de trabalho geofísico, evoluindo de um processo fragmentado e dependente de linha de comando para uma solução integrada e orientada a tarefas. Antes da integração, a execução do Mamute exigia que o usuário manipulasse manualmente os arquivos de configuração e interagisse diretamente com o terminal para configurar e executar a modelagem. Com o *plugin*, todo o processo foi internalizado no OpendTect. Dessa forma, a integração cumpre o objetivo de tornar o uso do Mamute mais acessível e facilitar o ambiente de trabalho para geofísicos e profissionais da área.

6 CONCLUSÃO

Este trabalho apresentou a integração do Mamute ao OpendTect por meio de um *plugin* nativo em C++. Até o momento, o escopo deste trabalho focou na integração da modelagem sísmica. O foco central foi transformar um fluxo originalmente dependente de linha de comando em uma experiência de uso guiada por interface gráfica.

Com a arquitetura atual, foi possível consolidar uma operação orientada a produto. Por meio dela, o usuário configura geometria, modelo de velocidade e parâmetros de modelagem. Em seguida, é possível executar as etapas de preparação e acompanhar a simulação em um ciclo integrado de *Build*, *Run*, *Monitor* e *View*.

Os resultados obtidos mostram que a solução alcançou um estado funcional e estável. A interface oferece validações em tempo real das entradas, organização mais clara e controle de consistência entre parâmetros. Também é feita verificação de arquivos de entrada necessários para execução, acompanhamento do processo e visualização dos resultados.

6.1 Trabalhos Futuros

Como continuidade deste trabalho, propõe-se a validação do plugin com a utilização de modelos de velocidades da literatura geofísica, como o modelo Marmousi (Versteeg, 1994), a expansão da integração para os módulos de RTM e FWI, abrangendo as demais funcionalidades do Mamute. Planeja-se adicionar ao *plugin* o carregamento de sismogramas e volumes 3D diretamente no OpendTect, adicionar suporte a múltiplas execuções simultâneas e dar suporte à execução em nuvem e em supercomputadores. Outro ponto importante é realizar a análise de usabilidade do *plugin* para saber se ele realmente atende ao que propõe.

REFERÊNCIAS

BARROS, T. *et al.* An open-source high-performance computing software for 3d geophysical modeling and inversion. In: **Anais do 19º CISBGF**. SBGf, 2025. Disponível em: https://doi.org/10.22564/19cisbgf2025.047.

BAYSAL, E.; KOSLOFF, D.; SHERWOOD, J. Reverse-time migration. **Geophysics**, v. 48, n. 11, p. 1514–1524, 1983. Disponível em: https://doi.org/10.1190/1.1441434.

BLANCH, J. O.; ROBERTSSON, J. O. A.; SYMES, W. W. Modeling of a constant q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique. **Geophysics**, v. 60, n. 1, p. 176–184, 1995. Disponível em: https://doi.org/10.1190/1.1443744.

CABIECES, R. *et al.* Integrated Seismic Program (ISP): A New Python GUI-Based Software for Earthquake Seismology and Seismic Signal Processing. **Seismological Research Letters**, v. 93, n. 3, p. 1895–1908, 2022. Disponível em: https://doi.org/10.1785/0220210205.

CLAERBOUT, J. F. **Imaging the Earth's Interior**. Blackwell Scientific Publications, 1985. Disponível em: https://doi.org/10.1111/j.1365-246X.1986.tb01086.x.

dGB Earth Sciences. **OpendTect**: open source seismic interpretation software. 2024. Disponivel em: https://dgbes.com/software/opendtect.

dGB Earth Sciences. **OpendTect plugin development**. 2024. Disponivel em: https://doc.opendtect.org/7.0.0/doc/Programmer/Default.htm.

dGB Earth Sciences. **OpendTect source code repository**. 2024. Disponivel em: https://github.com/OpendTect/OpendTect.

DUG Technology. **DUG Insight: A modern approach to seismic data interpretation**. 2023. Oil Review Middle East.

FERNANDES, J. B. *et al.* **Mamute: high-performance computing for geophysical methods**. 2025. Disponível em: https://arxiv.org/abs/2502.12350.

IZZATDIN, A.; JAAFAR, J.; GILAL, A. The study of opendtect seismic data interpretation and visualization package in relation to seismic interpretation and visualization models. **International Journal of Computer Science and Network Security (IJCSNS)**, v. 17, p. 124–134, 2017.

LI, Q.-Z. Subsurface attenuation of high-frequency signals and the empirical Vp-Q relation. In: LI, Q.-Z. (Ed.). **High-Resolution Seismic Exploration**. Society of Exploration Geophysicists, 2017. cap. 3, p. 39–58. Disponível em: https://doi.org/10.1190/1.9781560803508.

LORENZO, J. M. *et al.* Introducing Novices to Active-Source Seismology via SeismicUnixGui. **Seismological Research Letters**, v. 97, n. 1, p. 578–584, 2025. Disponível em: https://doi.org/10.1785/0220250080.

PANETTA, J. *et al.* Accelerating time and depth seismic migration by cpu and gpu cooperation. **International Journal of Parallel Programming**, v. 40, n. 3, p. 290–312, 2012. Disponível em: https://doi.org/10.1007/s10766-011-0185-2.

PLESSIX, R.-E. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. **Geophysical Journal International**, v. 167, n. 2, p. 495–503, 2006. Disponível em: https://doi.org/10.1111/j.1365-246X.2006.02978.x.

RIBEIRO, J. *et al.* Botoseis-web: An interactive web application for seismic data processing using seismic unix. In: **Anais do 19º CISBGF**. SBGf, 2025. Disponível em: https://doi.org/10.22564/19cisbgf2025.085.

SILVA, F. H. S. da *et al.* Auto-tuning for openmp dynamic scheduling applied to full waveform inversion. **Computers & Geosciences**, v. 202, 2025. Disponível em: https://doi.org/10.1016/j.cageo.2025.105932.

SYMES, W. W. Reverse time migration with optimal checkpointing. **Geophysics**, v. 72, n. 5, p. SM213–SM221, 2007. Disponível em: https://doi.org/10.1190/1.2742686.

TARANTOLA, A. Inversion of seismic reflection data in the acoustic approximation. **Geophysics**, v. 49, n. 8, p. 1259–1266, 1984. Disponível em: https://doi.org/10.1190/1.1441754.

VERSTEEG, R. The marmousi experience: Velocity model determination on a synthetic complex data set. **The Leading Edge**, v. 13, n. 9, p. 927–936, 1994. Disponível em: https://doi.org/10.1190/1.1437051.

VIRIEUX, J.; OPERTO, S. An overview of full-waveform inversion in exploration geophysics. **Geophysics**, v. 74, n. 6, p. WCC1–WCC26, 2009. Disponível em: https://doi.org/10.1190/1.3238367.

YILMAZ, Ö. **Seismic Data Analysis**: Processing, inversion, and interpretation of seismic data. Society of Exploration Geophysicists, 2001. v. 2. Disponível em: https://doi.org/10.1190/1.9781560801580.