Ferramenta para conversão e migração automatizada de dados entre SGBDs

Gabriel Radzewicz de Queiroz, André Fiorin

Instituto Federal de Educação, Ciência e Tecnologia Farroupilha (IFFar)
Frederico Westphalen – RS – Brasil

gabriel.rq@proton.me, andre.fiorin@iffar.edu.br

Abstract. *This article describes the development of a tool for migrating data between different Database Management Systems (DBMSs), focusing on PostgreSQL and Firebird. The development follows a client-server architecture, and uses the technologies Java and Spring Boot for the backend, and Sveltekit for the frontend. It is concluded that the developed tool is capable of migrating data in an automated, free and simple manner, having limitations regarding the scope of migrated data.*

Resumo. *Este artigo descreve o desenvolvimento de uma ferramenta para migração de dados entre diferentes Sistemas Gerenciadores de Bases de Dados (SGBDs), focando em PostgreSQL e Firebird. O desenvolvimento segue uma arquitetura cliente-servidor, e utiliza as tecnologias Java e Spring Boot para o backend, e SvelteKit para o frontend. Conclui-se que a ferramenta desenvolvida é capaz de migrar os dados de forma automatizada, gratuita e simples, possuindo limitações quanto ao escopo dos dados migrados.*

1. Introdução

A migração de dados é um desafio constante para empresas que atuam com desenvolvimento e manutenção de software. Esse processo costuma gerar custos elevados e dificuldades técnicas consideráveis, especialmente devido a problemas de compatibilidade entre sistemas e ao tempo necessário para execução das tarefas envolvidas.

No contexto brasileiro, os SGBDs (Sistemas Gerenciadores de Bases de Dados) Firebird e PostgreSQL são amplamente utilizados, especialmente em aplicações corporativas de pequeno e médio porte. O Firebird se destaca pela forte presença em sistemas legados, em função de sua facilidade de implantação e baixo custo, enquanto o PostgreSQL tem sido frequentemente adotado em processos de modernização tecnológica por sua robustez e aderência a padrões. Nesse sentido, a migração entre esses dois SGBDs configura um cenário recorrente na realidade nacional.

Atualmente existem diversas ferramentas disponíveis no mercado que se propõem a realizar a migração entre diferentes Sistemas de Gerenciadores de Banco de Dados (SGBD). Como exemplos podem ser citadas ferramentas como DBCOnvert, FBExport, pgloader e AWS DMS.

Com a constante evolução das tecnologias de gerenciamento de dados, muitas dessas ferramentas se tornam obsoletas, o que exige processos de migração para soluções mais modernas e compatíveis com as demandas atuais. Além disso, Santos Neto et al (2013) destacam que existem muitas oportunidades para melhorias no desenvolvimento de novas soluções para migração de dados.

Diante deste contexto, o presente trabalho propõe o desenvolvimento de uma ferramenta de migração de dados modular, gratuita e de código aberto e com interface focada na clareza do processo de migração. A solução visa facilitar a transferência de dados entre os SGBDs PostgreSQL e Firebird. Essa proposta busca preencher lacunas deixadas por ferramentas já existentes no mercado ou propostas por trabalhos anteriores, como a exigência de licenças custosas, a falta de interfaces modernas e intuitivas, e a incapacidade de realizar a migração bidirecional dos dados.

Ressalta-se que este trabalho não propõe uma ferramenta genérica e universal para migração entre quaisquer SGBDs, mas sim uma solução focada nesse contexto específico, permitindo maior aprofundamento técnico, embora com uma arquitetura modular e extensível.

O texto que segue está organizado da seguinte forma: a Seção 2 apresenta os conceitos sobre migração de dados e SGBDs, que envolvem a temática central deste trabalho. Na Seção 3 são discutidos trabalhos correlatos, onde são abordadas, tanto pesquisas relacionadas, quanto ferramentas existentes no mercado. A Seção 4 descreve a metodologia utilizada e as etapas de desenvolvimento da ferramenta proposta. Na Seção 5 são apresentados os resultados obtidos, bem como os testes realizados. Por fim, a Seção 6 apresenta as considerações finais, seguido das referências bibliográficas utilizadas como base para o desenvolvimento da pesquisa.

2. Referencial teórico

Nesta seção são apresentados os principais conceitos relacionados às atividades e tecnologias utilizadas no desenvolvimento deste trabalho.

2.1. Migração de dados

As constantes mudanças nas tecnologias para gerenciamento de dados fazem com que ferramentas consideradas a melhor opção para armazenamento em determinado momento, sejam consideradas obsoletas e insuficientes noutro. Em situações como essa, se torna necessária a migração dos dados para tecnologias mais recentes ou superiores, processo que pode custar caro e ser tecnicamente difícil [Santos Neto et al. 2013].

Além disso, a IBM (2025) constata que, seja por motivos econômicos, seja por motivos técnicos, muitas empresas optam por migrar seus dados e cargas de trabalho para plataformas em nuvem, abandonando os chamados "ambientes *on-premises*", para obter melhor performance, custos, ou acesso a tecnologias mais modernas. Nesse contexto, a AWS (2025b) descreve a migração de dados da seguinte maneira:

> A migração de dados ocorre quando você move dados de um ambiente de computação ou sistema de armazenamento para outro. [...] O objetivo da migração de dados é mover os dados com eficiência e rapidez para evitar ou minimizar a interrupção das operações comerciais. [...] A migração de dados também pode envolver considerações de arquitetura de armazenamento para fatores como valores de dados ausentes ou alteração dos tipos de dados.

2.2. SGBDs

De acordo com Ramakrishnan e Gherke (2009, apud SOUZA, 2020, p. 19),

> Sistemas Gerenciadores de Bancos de Dados são softwares que promovem a interação entre usuários e a base de dados permitindo a aplicação de comandos no banco de dados como: criação, inserção, remoção e modificação, seleção, entre outros. Também são responsáveis por garantir a integridade e a segurança da informação, além de ter a função de recuperar as informações em caso de falha no sistema.

Para o desenvolvimento deste trabalho, o escopo foi restrito aos SGBDs PostgreSQL e Firebird. Essas ferramentas foram escolhidas por serem sistemas amplamente utilizados e que, embora trabalhem com bases de dados relacionais, apresentam desafios técnicos significativos no que se refere à migração e à interoperabilidade entre si, como diferenças de tipos de dados e estruturas internas. Esses desafios demandam a aplicação de conceitos de modelagem de dados, engenharia de software e integração de sistemas heterogêneos.

2.2.1. PostgreSQL

De acordo com a documentação da versão 17.4 do PostgreSQL (2025), é um sistema de gerenciamento de banco de dados objeto-relacional, de código aberto que suporta grande parte do padrão ANSI SQL e que oferece diversas funcionalidades modernas e suporte a extensões desenvolvidas por usuários. A ferramenta se apresenta como a mais avançada base de dados de código aberto disponível.

Segundo a Thomson Data, empresa que oferece soluções baseadas em análise de dados, o PostgreSQL é utilizado por grandes empresas como Netflix, Uber, Twitch, Instagram, Spotify e Reddit, demonstrando seu sucesso comercial e forte presença no mercado [Thomson Data 2025].

2.2.2. Firebird

Segundo a página web da própria ferramenta, Firebird é uma base de dados relacional que oferece muitas das funções do padrão ANSI SQL, compatível com diversos sistemas operacionais, como Windows, Linux e outras plataformas Unix, com excelente performance e funcionalidades. O projeto é economicamente independente, de código aberto e desenvolvido por apoiadores. [Firebird 2025a]

Ainda, segundo a documentação mais recente do Firebird [Firebird 2025b], é difícil contar quantas empresas utilizam a ferramenta devido sua característica *open source* e sem licença, porém, sabe-se que algumas empresas utilizam Firebird em seus serviços, como a Broadview Software Ltd (fornecedor de sistemas de informação e controle), Deutsche Presse-Agentur GmbH (maior agência de notícias da Alemanha), KIMData (sistemas de inteligência de negócios e *data warehousing* para hospitais alemães) e a U.S. Navy (em sistemas de gerenciamento e logísticas).

O Quadro 1 apresenta uma comparação geral entre os dois SGBDs, em suas versões mais recentes na data de desenvolvimento do presente trabalho.

Quadro 1. Comparação entre os SGBDs PostgreSQL e Firebird

Critério	PostgreSQL 18.x	Firebird 5.x
Interface com linguagens	Sim. Ex.: C, C++, Python, Java, Delphi, C#, etc.	Sim. Ex.: Java, Delphi, C, C++, etc.

Stored Procedures	Sim	Sim
Triggers	Sim	Sim
Cursores	Sim	Sim
Transactions	Sim	Sim
Bloqueios	Sim	Sim
Ferramentas disponíveis	Sim. Ex.: PgAdmin IV, Barman, etc.	Sim. Ex.: IBExpert
Compatibilidade	Sim. Ex.: Windows, Linux, etc.	Sim. Ex.: Windows, Linux, etc.
Sub-selects	Sim	Sim
Views	Sim	Sim
Sequências	Sim	Sim
Orientado a Objetos	Sim	Não
Unions	Sim	Sim
Backup	Sim	Sim
Full-text search	Sim	Não
Insert múltiplo	Sim	Não
Replicação	Sim	Sim

Fonte: SILVA (2011, p. 51)

Essas são algumas das principais semelhanças e diferenças entre o PostgreSQL e o Firebird, valendo destacar que as maiores diferenças se encontram nos tipos de dados e na sintaxe de cada SGBD. Dadas essas características, a ferramenta deve ser capaz de tratar as particularidades dos SGBDs no processo de migração; por exemplo, ao converter uma base de dados PostgreSQL para Firebird, datas com fuso horário e UTC, formatadas no padrão ISO 8601 devem ser tratadas (pois o Firebird não suporta esse formato), e *strings* muito grandes precisam ser subdivididas e concatenadas, a fim de não exceder o limite de tamanho para *string literals* do Firebird.

3. Trabalhos correlatos

Nessa seção são apresentados trabalhos com propostas semelhantes e ferramentas de migração de dados existentes no mercado.

3.1. Trabalhos relacionados

Queiroz et al. (2022), propuseram um *middleware* flexível para migração automática entre bases de dados relacionais e não-relacionais. A ferramenta apresenta uma arquitetura modular, com funções bem isoladas e definidas. A solução apresentou

performance melhor que abordagens anteriores citadas no próprio trabalho, considerando o banco de dados Northwind.

A solução proposta por Queiroz et al. (2022) é interessante, pois apesar de ter como objetivo criar um *middleware* para migração de um SGBD para uma base de dados de grafos (Dgraph), sua abordagem é flexível, performática, e bem arquitetada, permitindo ser expandida. Além disso, a ferramenta permite que tabelas específicas sejam exportadas, e dispensa o uso de pacotes adicionais, por usar *queries* diretas. Ainda, segundo os autores, o tempo de execução é excelente, e a confiabilidade alta.

Kaluba e Nyirenda (2024), utilizaram estratégias e ferramentas já existentes para migração de dados, porém se aproveitaram de uma arquitetura de microsserviços para o processo de migração. Em sua abordagem, cada migração de uma base de dados origem para uma base de dados alvo é tratada como um microsserviço, fazendo com que um conjunto de pequenos serviços trabalhe em conjunto, de maneira coesa, para alcançar o objetivo final da migração. O *framework* desenvolvido pelos autores é capaz de migrar dados com perda mínima, e entre qualquer par de sistemas de migração de bases de dados.

A abordagem proposta por Kaluba e Nyirenda (2024) primeiramente executa um serviço de métricas na base de dados origem, depois segue um processo de ETL (*Extract*, *Transform*, *Load*), definido em um serviço próprio, que interage com um serviço de replicação para copiar os dados das tabelas para um arquivo JSON. Um serviço de transformação transforma os dados de origem em um formato compatível com a base de dados destino (para a conversão de dados entre diferentes SGBDs, um dicionário de tipos de dados é usado), e então um outro serviço carrega os dados no novo SGBD. Por fim, um serviço de validação interage com o serviço de métricas para comparar as bases de dados de origem e destino.

3.2. Ferramentas de migração no mercado

No âmbito das ferramentas profissionais disponíveis no mercado, existem diversas alternativas. Para este trabalho, foram selecionadas as ferramentas com maior ênfase nos SGBDs definidos no escopo da pesquisa e com relevância reconhecida no mercado.

A primeira delas é o DBConvert, que se apresenta como uma ferramenta de conversão e sincronização de bases de dados, com suporte para cerca de 30 tipos de bancos de dados, incluindo provedores de serviços em nuvem. A versão mais completa (DBConvert Studio) é uma ferramenta comercial paga, contando com versões de teste [DBConvert 2025].

No contexto das ferramentas em nuvem, o AWS DMS é um serviço que realiza a migração de bancos de dados *on-premises* para soluções gerenciadas em nuvem da AWS. A ferramenta é capaz de migrar dados de diversos SGBDs de origem para as opções de bases de dado alvo disponíveis nos serviços AWS RDS e AWS EC2. A ferramenta pode ser testada gratuitamente no *free tier* da AWS, e possui precificação sob demanda, baseada em horas de uso [AWS 2025a].

Além disso, os SGBDs costumam possuir ferramentas próprias ou de terceiros para migração de dados. O Firebird possui a ferramenta FBExport, capaz de exportar e

importar dados de bases de dados Firebird [FBExport 2025]. Já o PostgreSQL possui a ferramenta pgloader, que permite importar dados de arquivos CSV ou de outros SGBDs, como MySQL, SQLite e MS SQL Server® [pgloader 2025]. Nenhuma dessas ferramentas é capaz de realizar a migração bidirecional de dados entre PostgreSQL e Firebird.

4. Metodologia e desenvolvimento

O presente trabalho tem por caráter o desenvolvimento tecnológico, com foco na criação de uma ferramenta para migração de dados. Para isso, inicialmente foi realizada uma pesquisa exploratória sobre os principais desafios enfrentados durante o processo de migração de dados, com base em relatos de empresas, trabalhos acadêmicos e bibliografia relacionada. Também foi realizada uma análise dos principais requisitos e etapas envolvidas nesse processo, a fim de identificar quais são mais relevantes e devem ser levadas em consideração para a implementação da ferramenta. A Figura 1 detalha as etapas metodológicas utilizadas no desenvolvimento do trabalho.

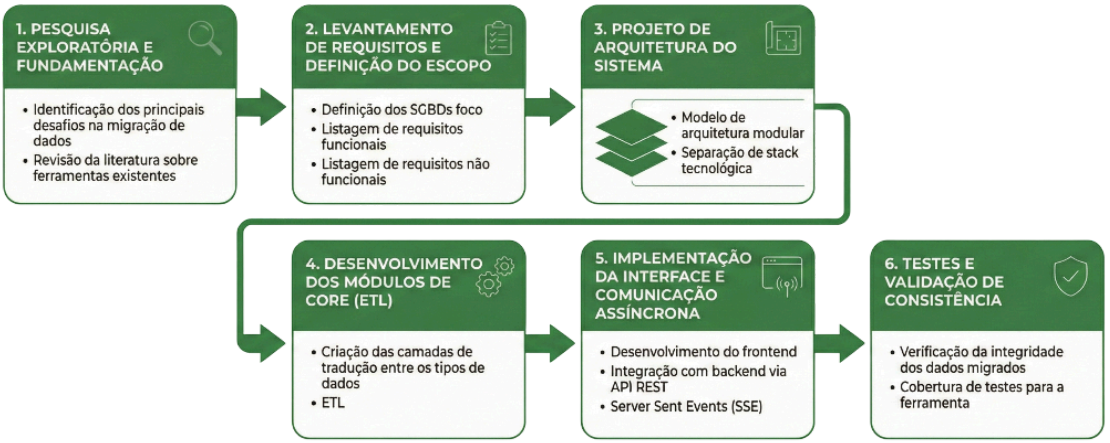

Figura 1. Etapas de desenvolvimento. Fonte: do autor.

Queiroz et al. (2022) mencionam que a performance da migração é uma dificuldade apresentada nas abordagens revisadas, e afirmam que sua abordagem consegue realizar a migração mais rapidamente.

Além disso, Santos Neto et al. (2013) elencam os seguintes requisitos para o desenvolvimento de uma boa ferramenta de migração de dados, que podem ser tratados como desafios na implementação de tal ferramenta:

- Migração envolvendo estruturas diferentes;
- Migração paralela ou distribuída;
- Migração Incremental;
- Reengenharia de dados;
- Migração envolvendo paradigmas diferentes;
- Migração entre bases de dados com modelagens diferentes;
- Testabilidade;
- Boa usabilidade.

A partir dessa análise, foram definidos os requisitos funcionais e não funcionais do sistema proposto. Em seguida, realizou-se a modelagem, que serviu como base para o desenvolvimento da API e da interface gráfica. Por fim, foram realizados testes e a validação.

4.1. Requisitos do sistema

Com base na análise de ferramentas existentes e nos desafios de interoperabilidade [Queiroz et al. 2022], o desenvolvimento da solução foi norteado por requisitos que visam simplicidade e eficiência.

Requisitos Funcionais (RF):

- **RF01:** Conectar-se a diferentes SGBDs (PostgreSQL e Firebird).
- **RF02:** Realizar a extração automática do esquema e dos dados da origem.
- **RF03:** Mapear tipos de dados de forma compatível entre os SGBDs.
- **RF04:** Executar a carga dos dados no destino respeitando a integridade referencial.

Requisitos Não Funcionais (RNF):

- **RNF01:** Interface que permita a visibilidade das etapas ETL de fácil operação (SvelteKit).
- **RNF02:** Processamento assíncrono para lidar com grandes volumes de dados sem travar a interface.
- **RNF03:** Independência de plataforma através do uso de Java (Spring Boot).

4.2. Arquitetura e tecnologias empregadas

A solução foi construída utilizando uma arquitetura cliente-servidor, pensado de forma que o *frontend* e o *backend* possam evoluir de forma independente.

O sistema proposto conta com uma interface genérica, capaz de se comunicar com drivers específicos de cada SGBD. No *backend*, optou-se pelo ecossistema Java com Spring Boot 3, utilizando o Spring Data JDBC para a camada de persistência. Essa escolha permite que a ferramenta se comunique com diversos bancos de dados através de drivers JDBC genéricos, facilitando a expansão para outros SGBDs no futuro. O Spring Data abstrai as APIs Java JDBC, tornando a comunicação com diferentes SGBDs mais fácil e mais bem integrada ao ecossistema do Spring.

No *frontend*, utilizou-se o *framework* SvelteKit, que oferece uma interface reativa e performática. A comunicação entre a interface e a API é realizada via requisições HTTP (*Hypertext Transfer Protocol*).

4.3. Implementação da API e controle de fluxo

A comunicação entre as camadas é feita através de uma API REST. O núcleo do sistema reside no gerenciamento de estados da migração, que assegura que cada etapa ocorra na ordem correta.

A API é responsável por todo o processamento da migração de dados, incluindo a comunicação com as bases de dados, o processo de ETL e a análise de consistência da migração.

Na etapa de extração, dados e metadados são obtidos da base de dados de origem e armazenados em um formato JSON que permite sua descrição. A etapa de transformação inclui o mapeamento automático ou manual dos tipos de dados, seguido da adaptação dos nomes das colunas e conversão dos formatos dos dados (de acordo com um mapa de conversão de tipos), considerando apenas dados no padrão SQL. Por fim, na etapa de carga, os dados transformados são inseridos na base de dados destino.

Um módulo específico foi desenvolvido para validar a consistência dos dados migrados. Esse módulo coleta metadados da base de dados de origem e destino, e os compara, a fim de avaliar a integridade e possíveis perdas ou alterações durante o processo. A Figura 2 apresenta o esquema de funcionamento da API.

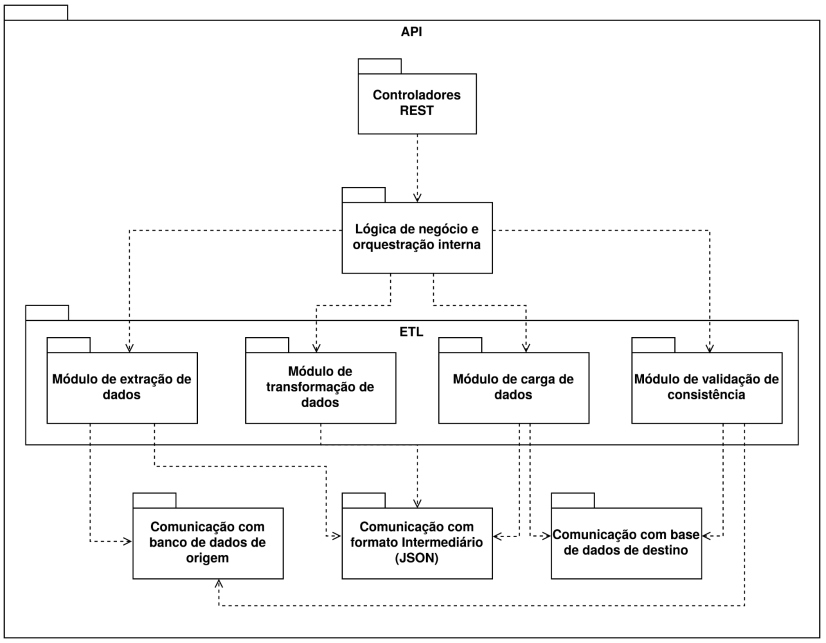

Figura 2. Diagrama UML de pacotes da API. Fonte: do autor.

As funcionalidades da API são expostas por controladores REST (*Representational State Transfer*), acessíveis a partir de requisições HTTP usando os métodos e rotas disponibilizadas, listadas e descritas abaixo:

- **POST /api/v1/migrations:** registra uma nova migração, espera receber um identificador (nome) para esse registro, dados de conexão com a base de dados de origem e destino, e o nome do mapa de conversão como corpo da requisição;
- **POST /api/v1/migrations/{id}/extract:** inicia a etapa de extração, espera receber o ID da migração previamente iniciada como parâmetro de rota;
- **POST /api/v1/migrations/{id}/transform:** inicia a etapa de transformação, espera receber o ID da migração previamente iniciada como parâmetro de rota;
- **POST /api/v1/migrations/{id}/load:** inicia a etapa de carga, espera receber o ID da migração previamente iniciada como parâmetro de rota;

- **POST /api/v1/migrations/{id}/validate:** inicia a etapa de validação de consistência, esperar receber o ID da migração previamente iniciada como parâmetro de rota;
- **GET /api/v1/migrations/{id}/status:** retorna o status da migração previamente iniciada com o ID informado como parâmetro de rota;
- **GET /api/v1/migration/{id}/sse:** comunica atualizações de estado da migração de ID informado como parâmetro de rota, por meio de *Server Sent Events*;
- **GET /api/v1/migrations/{id}/sql:** retorna os scripts SQL de DDL gerados na etapa de transformação para a migração com o ID informado como parâmetro de rota, de maneira paginada. Pode receber parâmetros de consulta para controlar a página atual e a quantia de *scripts* retornados;
- **PUT /api/v1/migrations/{id}/sql:** atualiza o conteúdo dos *scripts* SQL de DDL informados, pertencentes a migração com ID especificado como parâmetro de rota. Espera receber uma lista contendo o nome do arquivo SQL e seu novo conteúdo, para cada arquivo, como corpo da requisição.

Para garantir que as requisições feitas a API não fiquem bloqueadas, aguardando a conclusão das etapas de ETL, essas operações são executadas de maneira não bloqueante (assíncrona, por meio de concorrência), retornando imediatamente. Assim, o controle do estado da migração é realizado por meio de uma máquina de estados, permitindo a identificação do estado atual e informações associadas (como quaisquer dados e metadados necessários para acompanhamento do processo). Essas informações sobre a migração, por sua vez, são armazenadas em um repositório acessível em qualquer momento do processo de ETL (uma base de dados ou armazenamento em memória, por exemplo), pois são usados para o controle e identificação da migração durante o todo o processo. A Figura 3 apresenta a máquina de estados que representa o ciclo de vida de uma migração.

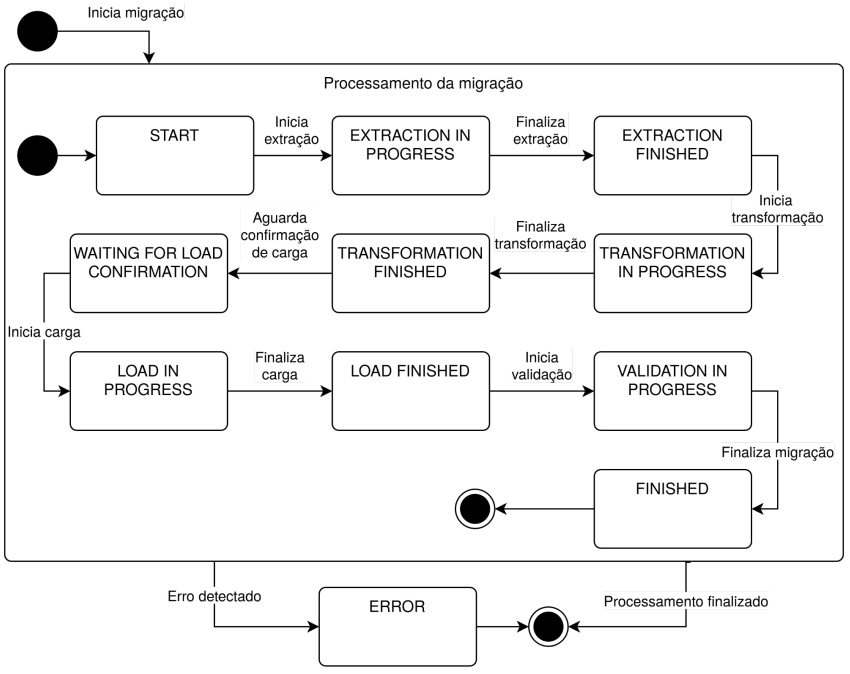

Figura 3. Diagrama UML de máquina de estados de uma migração. Fonte: do autor.

Os estados são atualizados para cada entrada armazenada no repositório de migrações ativas, conforme as etapas de ETL ocorrem. Apenas os estados *FINISHED* e *ERROR* são considerados finais. O estado que mantém a migração aguardando confirmação do usuário (*WAITING FOR LOAD CONFIRMATION*) existe para que o usuário possa revisar e alterar os *scripts* de DDL gerados pela ferramenta antes de prosseguir com a carga de dados. As etapas do processo de ETL são unidirecionais, isto é, não é possível reexecutar etapas já concluídas, devendo o sistema validar esses casos, impedir a execução e retornar uma resposta de erro para a requisição.

Vale mencionar que a etapa de validação não é obrigatória, porém é considerada de grande importância e, portanto, deve ser executada automaticamente pela orquestração da migração implementada na interface gráfica *web*. Recomenda-se que também seja executada nos casos de uso programáticos (via API), a fim de garantir a consistência do processo de migração.

4.4. O processo de ETL (Extração, Transformação e Carga)

A operação principal da ferramenta segue o modelo ETL, dividida nos seguintes módulos internos:

4.4.1 Módulo de Extração

Este módulo é responsável por ler os metadados do banco de origem (tabelas, colunas, chaves primárias e estrangeiras) e converter essas informações em uma representação intermediária (JSON).

Nessa etapa, a ferramenta se conecta a base de dados de origem e extrai os dados de todas as colunas, assim como metadados descritivos (informações de *schema*, conjunto de caracteres, tabelas, colunas, tipos de dados, chaves primárias e estrangeiras, *etc*).

A fim de reduzir o tempo de execução, a etapa de extração é implementada com execução concorrente, utilizando um *pool* de *threads* para obter os dados de diversas tabelas simultaneamente. Os metadados são obtidos a partir da interface de conexão com a base de dados do JDBC.

4.4.2 Módulo de Transformação

Este módulo é responsável pela conversão dos dados da base de dados de origem em dados compatíveis com a base de dados de destino. Neste momento também é realizada a geração dos *scripts* SQL de DDL (*Data Definition Language*) e DML (*Data Manipulation Language*) para criação de tabelas e inserção dos dados, respectivamente.

Nessa etapa, os tipos SQL genéricos (correspondentes aos tipos definidos na classe *Types* do pacote java.sql) obtidos na extração de metadados da base de dados de origem são mapeados para os tipos correspondentes da base de dados destino, por meio de um mapa de conversão. Nessa etapa também é calculada a ordem de criação das tabelas na base de dados destino, por meio de um algoritmo de ordenação topológica, a fim de evitar conflitos na etapa de carga.

Os mapas de conversão, utilizados para mapear os tipos de dados da base de dados de origem para a base de dados destino, por sua vez, são representados em

arquivos JSON, seguindo uma estrutura de mapa na qual a chave representa um tipo de dados genérico (conforme descrito acima), e o valor representa o tipo de dados no SGBD representado. Um mapa de conversão para uma base de dados PostgreSQL é representado na Figura 4.

Figura 4. Exemplo de mapa de conversão do PostgreSQL. Fonte: do autor.

4.4.3 Módulo de Carga

Na sequência, o módulo de carga executa o processo final de ETL, responsável por carregar os dados transformados na base de dados destino.

Essa etapa executa os *scripts* de DDL para criar a estrutura da base de dados, e executa os *scripts* de DML para inserir os dados extraídos. São usadas técnicas de leitura/escrita com *buffers* e *lazy loading* para processar grandes quantidades de registros em lotes, otimizando o uso de memória e acelerando a inserção no destino durante o processamento da carga de dados.

Após a carga, o sistema realiza uma conferência entre o número de registros extraídos e inseridos para garantir que não houve perda de informação e validar a consistência dos dados migrados. Nessa etapa é realizada uma validação estrutural (quantidade de tabelas e colunas, nomes de tabelas e colunas e tipos de dados) e volumétrica (quantidade de dados inseridos nas tabelas) simples. Cabe ressaltar que a etapa de validação é opcional.

É importante destacar que as etapas do processo de ETL podem ser controladas manualmente, e não necessariamente são executadas de forma encadeada, contanto que a ordem de execução seja respeitada: extração, transformação, carga e validação. Isso é necessário para que o usuário da ferramenta tenha controle do processo de migração, seja por meio da API ou da interface gráfica, permitindo, por exemplo, que os *scripts* de DDL gerados após a transformação sejam manipulados para correções/ajustes, antes de serem submetidos à execução da etapa de carga. O fluxo dos dados durante a execução de um processo de migração pode ser observado na Figura 5.

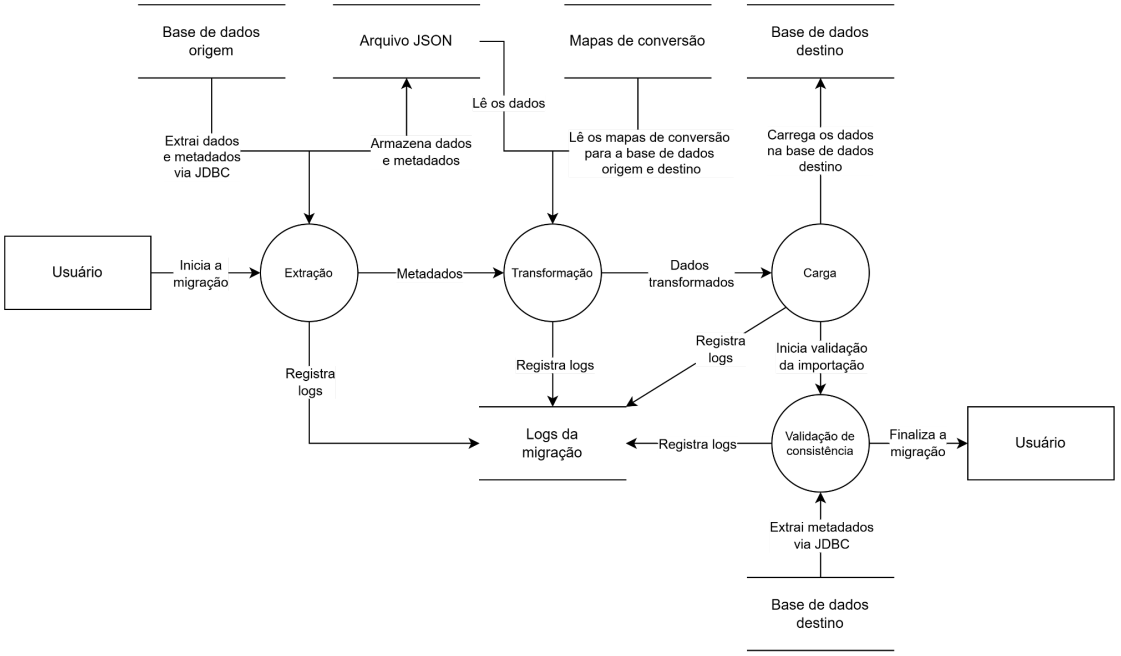

Figura 5. Fluxo de dados do processo de migração. Fonte: do autor.

O estado atual de uma migração registrada pode ser consultado de duas maneiras a partir da API. É exposta uma rota que retorna o estado de uma migração a partir de seu ID, permitindo consultas esporádicas ou até mesmo a implementação de um sistema de *polling* para atualizações regulares; também é exposta uma rota que implementa *Server Sent Events* (SSE), permitindo que a API comunique em tempo real atualizações de estado para uma migração, a partir de seu ID.

Os metadados, lidos a partir das interfaces JDBC, são mapeados para as classes apresentadas na Figura 6, que por sua vez são utilizadas durante o processo de migração para descrição das bases de dados origem e destino e conversão dos tipos de dados (por meio dos mapas de conversão).

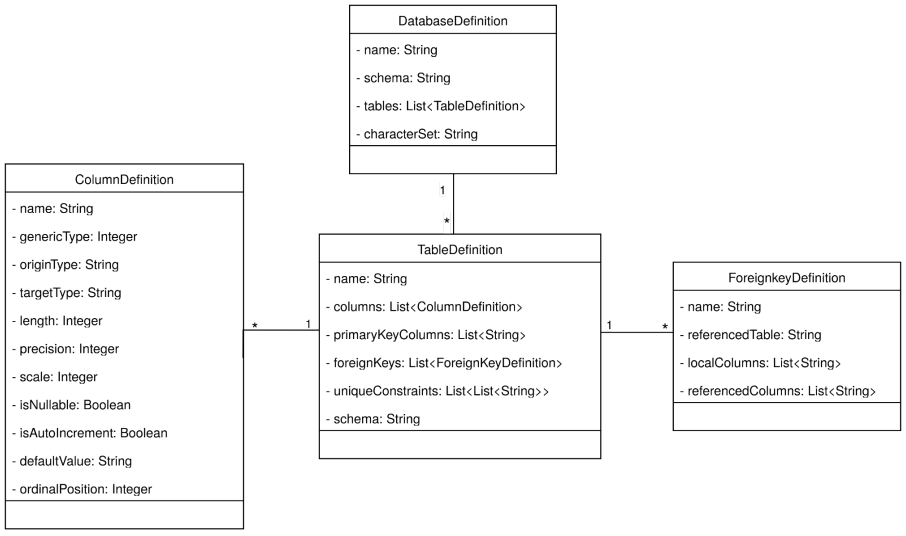

Figura 6. Diagrama UML de classes utilizadas no processo de migração. Fonte: do autor.

Os dados brutos extraídos são armazenados em um ou mais arquivos JSON, permitindo que sejam manipulados durante o processo de ETL e que um cache local seja mantido durante todo o processo.

4.5. Testes e Documentação

A fim de garantir a integridade e manutenibilidade do código produzido, foram desenvolvidos testes específicos para o *backend* e *frontend* da ferramenta. Para o *backend*, foram desenvolvidos testes unitários (JUnit) e de integração (MockMvc) cobrindo os utilitários, serviços, repositórios e controladores da aplicação. Para o *frontend*, foram desenvolvidos testes unitários e de componentes (Vitest, testing-library-svelte e jsdom) a fim de garantir o correto funcionamento e cobertura dos componentes e utilitários da interface *web*. Foram realizados também testes de sistema, com resultados discutidos na sessão 5.2.

Além disso, as rotas da API foram documentadas com a especificação OpenAPI (OAS), um formato agnóstico para descrever APIs RESTful, e Swagger, permitindo o acesso a uma documentação interativa.

5. Resultados obtidos

A ferramenta desenvolvida pode ser utilizada de maneira programática (por meio da API) ou através da interface gráfica *web*. A seguir, são apresentados o fluxo de uso da interface e os resultados obtidos nos testes aplicados.

5.1. Funcionalidades e fluxo de uso

A interface gráfica foi projetada para permitir o uso completo da ferramenta, priorizando a redução na carga cognitiva da interação. Além disso, a interface possibilita o acompanhamento do processo de migração de dados em tempo real, de forma clara e objetiva.

Ao acessar a página *web* da ferramenta, o usuário pode iniciar um processo de migração preenchendo um identificador para tal (Figura 7). O identificador será usado pela ferramenta apenas para nomear o processo de migração.

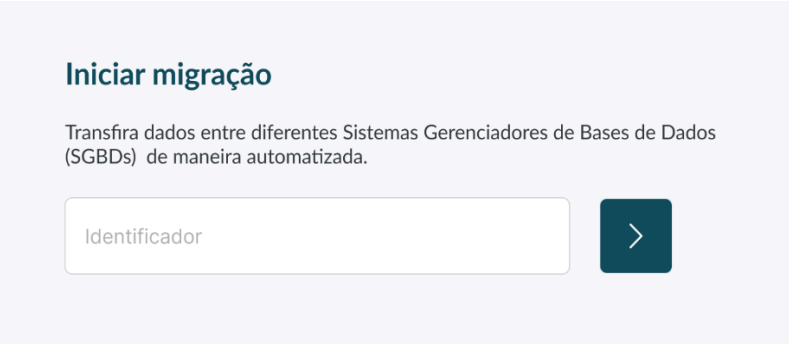

Figura 7. Página inicial da interface *web*. Fonte: do autor.

Após preencher o identificador e prosseguir, o usuário é direcionado para a página com os formulários onde devem ser informados os dados de conexão com a base de dados de origem e destino, bem como o SGBD utilizado por cada uma (Figura 8).

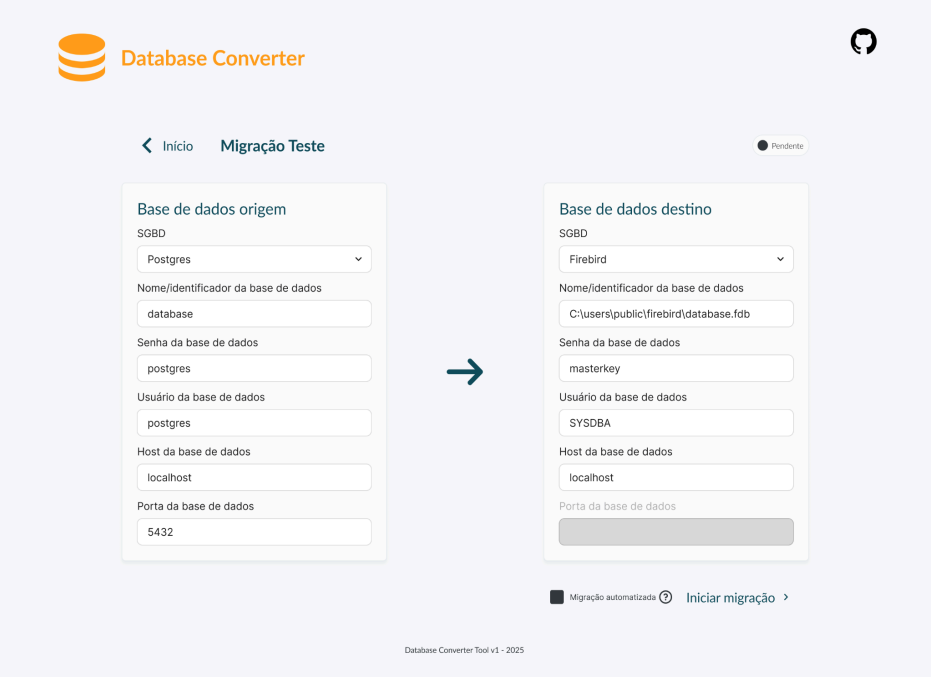

Figura 8. Página de configuração da migração. Fonte: do autor.

Depois de preencher os dados solicitados, o usuário pode iniciar o processo de migração. A opção de "migração automatizada" permite ao usuário executar o processo de ETL sem que a ferramenta solicite a confirmação de carga dos dados.

Por fim, o usuário é levado a página de *status*, apresentada na Figura 9. Nessa tela as informações e *logs* associados ao processo de migração são apresentados em tempo real, para que o usuário possa acompanhar o processo de migração.

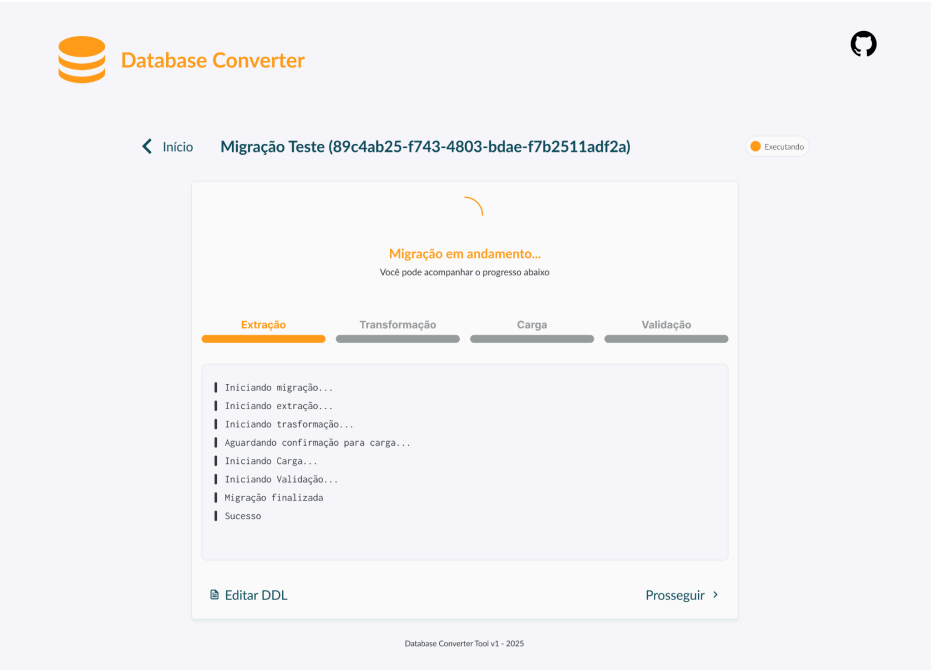

Figura 9. Página de configuração da migração. Fonte: do autor.

Para migrações iniciadas sem a opção de “migração automatizada”, a ferramenta irá pausar o processo antes de iniciar a etapa de carga dos dados, permitindo ao usuário editar e verificar os *scripts* de DDL gerados antes de resumir o processo de migração

5.2. Resultados dos testes

A validação do sistema proposto ocorreu por meio de testes controlados, com execução de cenários de migração simulados. A verificação de integridade dos dados foi realizada a partir da comparação entre os dados de origem e destino.

Foram realizados testes de migração usando a base de dados de amostra pagila[1], implementação da base de dados de amostra sakila (MySQL) em PostgreSQL. A base de dados pagila ocupa 16.36 MB de espaço, considerando sua implementação até a data do presente trabalho. Os testes buscaram medir o tempo necessário para realizar uma migração completa da base de dados pagila de PostgreSQL para Firebird, e do resultado dessa migração novamente para PostgreSQL. Também foi considerado o retorno da análise de consistência.

Os testes foram realizados em uma máquina executando o sistema operacional Windows 10, com 8 GB de memória RAM e um processador Intel Core i5-10400, com as versões 18.1 do PostgreSQL e 5.0 do Firebird. Os resultados obtidos são apresentados no Quadro 2.

Quadro 2. Resultados dos testes de migração com a ferramenta

PostgreSQL → Firebird		
Teste	**Duração**	**Espaço em disco**
T1	01:07:735	9.98 MB
T2	01:06:466	9.98 MB
T3	01:04:132	9.98 MB
Firebird → PostgreSQL		
T1	00:39:125	13.96 MB
T2	00:38:671	13.96 MB
T3	00:38:878	13.96 MB

Fonte: do autor.

Os resultados demonstram que a ferramenta é capaz de realizar a migração da base de dados pagila do PostgreSQL para o Firebird em uma média de tempo de 01:06:111. Executando o processo inverso, migrando os dados previamente convertidos para o Firebird novamente para o PostgreSQL, a média de tempo foi de 00:38:891, demonstrando uma possível maior eficiência do PostgreSQL, especialmente na etapa de carga dos dados.

[1]Disponível em: https://github.com/devrimgunduz/pagila

Foi observado que o espaço em disco final das bases de dados migradas do Firebird para o PostgreSQL apresentou tamanho inferior ao da base de dados original utilizada nos testes. Essa diferença ocorre pelo fato de que determinadas colunas da tabela *'film'*, especialmente *'release_year'* e *'fulltext'* não foram migradas por utilizarem tipos de dados específicos do SGBD PostgreSQL que não fazem parte do escopo de conversão da ferramenta. Ressalta-se que a ausência dessas colunas foi identificada e controlada durante o processo de migração, não comprometendo a integridade dos demais dados, os quais foram migrados com sucesso e devidamente validados.

Além disso, a ferramenta foi testada com a versão 9.2 do PostgreSQL no processo inverso de migração, obtendo êxito na conversão de uma base de dados para o Firebird, o que evidencia a viabilidade da abordagem bidirecional proposta.

6. Considerações finais

O presente trabalho teve como proposta o desenvolvimento de uma ferramenta gratuita e de código aberto para migração de dados entre os SGBDs PostgreSQL e Firebird. Considera-se que tal proposta foi alcançada, e que a ferramenta desenvolvida cumpre com os objetivos propostos.

A ferramenta foi desenvolvida para realizar as etapas de um processo de ETL completo, através de módulos específicos. Também foi desenvolvido um módulo para validação de consistência e integridade da migração dos dados. A API foi integrada a uma interface gráfica que facilita o uso da ferramenta.

Comparada a outras ferramentas disponíveis e trabalhos relacionados, a ferramenta desenvolvida oferece uma interface intuitiva e moderna, segue padrões modulares que simplificam melhorias e é capaz de realizar a migração bidirecional entre os SGBDs PostgreSQL e Firebird. Contudo, é importante destacar as limitações da ferramenta: não converte os valores padrão definidos nas colunas; não converte tipos de dados específicos de um SGBD ou definidos por usuário; exige que a base de dados de destino seja previamente criada; não considera o *schema* da base de dados (para bases de dados que utilizam *schemas*, como o PostgreSQL) e não é capaz de resumir um processo de migração pausado anteriormente, nem listar todas as migrações ativas (apesar de ser capaz de realizar múltiplas migrações ao mesmo tempo).

Os códigos-fonte produzidos neste trabalho estão disponíveis em repositórios públicos. Os repositórios correspondentes ao *backend* da ferramenta e à implementação do *frontend* podem ser acessados em:

- *Backend*: https://github.com/Gabriel-RQ/database-converter-backend;
- *Frontend*: https://github.com/Gabriel-RQ/database-converter-frontend.

Para trabalhos futuros, sugere-se o aprofundamento nas limitações da ferramenta, bem como a realização de otimizações voltadas à melhoria de desempenho e ao uso de memória. Também é recomendada a ampliação do suporte a outros SGBDs, tornando a ferramenta compatível com um conjunto maior de tecnologias. Além disso, pode ser benéfico investigar soluções que permitam tratar as particularidades de cada SGBD de forma genérica e centralizada durante o processo de migração, evitando a

introdução de código específico no núcleo da ferramenta e garantindo maior modularidade e facilidade de manutenção.

Referências

AWS (2025a) "Migração de banco de dados para a nuvem | AWS Database Migration Service (AWS DMS)", Disponível em: https://aws.amazon.com/pt/dms, Acesso em: 9 maio 2025.

AWS (2025b) "O que é migração de dados? - Explicação sobre planos de migração de dados", Disponível em: https://aws.amazon.com/pt/what-is/data-migration, Acesso em: 7 maio 2025.

DBConvert (2025) "Database conversion and synchronization software", Disponível em: https://dbconvert.com, Acesso em: 9 maio 2025.

FBExport (2025) "Tools for Firebird developers", Disponível em: https://fbexport.sourceforge.net, Acesso em: 9 maio 2025.

Firebird (2025a) "About Firebird", Disponível em: https://www.firebirdsql.org/pt/about-firebird, Acesso em: 7 maio 2025.

Firebird (2025b) "Who uses Firebird?", Disponível em: https://www.firebirdsql.org/file/documentation/papers_presentations/html/who-uses.html, Acesso em: 21 maio 2025.

IBM (2025) "Migração de dados", Disponível em: https://www.ibm.com/br-pt/think/topics/data-migration, Acesso em: 7 maio 2025.

Kaluba, Z. and Nyirenda, M. (2024) "Database Migration Service With A Microservice Architecture", IOSR Journal of Computer Engineering, v. 26, n. 1, p. 48–58, Disponível em: https://www.iosrjournals.org/iosr-jce/papers/Vol26-issue1/Ser-4/F2601044858.pdf.

PostgreSQL Global Development Group (2025) "PostgreSQL 17.0 Documentation", Disponível em: https://www.postgresql.org/files/documentation/pdf/17/postgresql-17-US.pdf, Acesso em: 7 maio 2025.

pgloader (2025) "pgloader", Disponível em: https://pgloader.io, Acesso em: 9 maio 2025.

Queiroz, J. S. et al. (2022) "AMANDA: A Middleware for Automatic Migration between Different Database Paradigms", Applied Sciences, v. 12, n. 12, p. 6106–6106, 16 jun. 2022, Disponível em: https://www.mdpi.com/2076-3417/12/12/6106.

Santos Neto, P. A. dos, Rodrigues Neto, J., Ribeiro Júnior, F. das C. Oliveira, P. A. (2013) "Requisitos para Ferramentas de Migração de Dados", In: Simpósio Brasileiro de Sistemas de Informação (SBSI), 9., João Pessoa, p. 887–898, Porto Alegre: Sociedade Brasileira de Computação. DOI: https://doi.org/10.5753/sbsi.2013.5749.

Silva, W. C. (2011) "SGBDs LIVRES: um estudo comparativo entre os sistemas gerenciadores de banco de dados Firebird, MySQL e PostgreSQL", Trabalho de

Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Piauí, Teresina, 48 f., Disponível em: https://repositorio.uespi.br/handle/123456789/121?show=full.

Souza, R. W. L. (2020) "Análise de desempenho de bancos de dados relacionais a partir da construção de um sistema de benchmarks sintético", Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) – Universidade Tecnológica Federal do Paraná, Curitiba, 78 f., Disponível em: https://riut.utfpr.edu.br/jspui/bitstream/1/28844/1/benchmarksintetico.pdf.

Thomson Data (2025) "Companies That Use PostgreSQL", Disponível em: https://www.thomsondata.com/customer-base/companies-that-use-postgresql.php, Acesso em: 9 maio 2025.