Aplicação Web para Migrar Dados de Bancos Relacionais para Bancos NoSQL Orientados a Documentos

Luan Felipe Marmentini[1], **Evandro Kuszera**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos – PR – Brazil

luanf2003@gmail.com, evandrokuszera@utfpr.edu.br

Abstract. *To simplify the use of a data migration tool between relational databases and document-oriented NoSQL databases, this work developed a web-based graphical application to support the configuration and execution of the migration process. The tool performs data migration with the support of a Large Language Model (LLM), based on the structure of the relational database and current access patterns. The graphical interface improves usability by making the configuration, execution, and monitoring stages more accessible. The application's front-end was developed using Next.js, while the back-end was implemented with Spring Boot. The communication between the two layers is handled via HTTP requests and Server-Sent Events (SSE), ensuring a dynamic and responsive user experience.*

Resumo. *Visando facilitar o uso de uma ferramenta de migração de dados entre bancos relacionais e bancos NoSQL orientados a documentos, foi desenvolvida uma aplicação gráfica web para a configuração e execução desse processo. A ferramenta migra os dados com o apoio de uma large language model (LLM), com base na estrutura do banco relacional e nos padrões de acesso atuais. A interface desenvolvida facilita a manipulação da ferramenta, tornando mais acessíveis às etapas de configuração, execução e monitoramento da migração. A comunicação entre o front-end, desenvolvido em Next.js, e o back-end, implementado com Spring Boot, ocorre por meio de requisições HTTP e Server-Sent Events (SSE), garantindo uma experiência dinâmica e responsiva ao usuário.*

1. Introdução

Banco de dados relacionais são amplamente utilizados para armazenar dados de aplicações diversas. No entanto, apresentam limitações em cenários com alta escalabilidade e com dados semi-estruturados, abrindo espaço para o uso de bancos NoSQL [Stonebraker et al. 2007].

Bancos de dados NoSQL (do inglês, *Not only SQL*) [Sadalage and Fowler 2013] diferem dos bancos relacionais em termos de arquitetura, modelo de dados e linguagem de consulta. Os bancos de dados NoSQL oferecem maior flexibilidade, não obrigado a definição de um esquema antes da inserção de dados. Procuram diminuir o tempo de execução de consultas e inserções mas, por outro lado, transferem o gerenciamento do esquema de dados e consistência de dados para a aplicação.

Muitos projetos optam atualmente por uma solução híbrida, utilizando bancos de dados relacionais e não relacionais para atender diferentes requisitos. Converter e migrar os dados entre diferentes bancos de dados

acaba se tornando uma necessidade nesses projetos. Diferentes abordagens ([Santos and Costa 2016, Lee and Zheng 2015, Vajk et al. 2013, Zhao et al. 2014, Serrano et al. 2015, Karnitis and Arnicans 2015, Jia et al. 2016, Mior et al. 2016]) foram propostas para converter e migrar os dados de um banco de dados relacional para um NoSQL. Cabe ao desenvolvedor analisar e selecionar a melhor solução no contexto específico, o que pode ser tedioso e propenso a erros de implementação.

Neste artigo é apresentada uma ferramenta visual para migração de dados, baseada em uma ferramenta de migração de SQL para NoSQL desenvolvida anteriormente pelos autores [Marmentini and Kuszera 2025]. A partir de metadados da base relacional e um conjunto de consultas (carga de trabalho) é utilizado um LLM (*Large Language Model*) para analisar o contexto, sugerir esquemas de dados e gerar código para migrar dados. A interface visual visa simplificar o processo de uso da ferramenta de migração, facilitando as definição das configurações necessárias para sua utilização. Os resultados mostram que a interface visual é capaz de manipular de forma adequada e intuitiva a ferramenta de migração de dados, possibilitando o usuário de receber *feedbacks* em tempo real sobre o processo de migração.

2. Fundamentação Teórica e Trabalhos Relacionados

Banco de dados relacionais (RDBs) são baseados em tabelas, linhas e colunas, na qual cada linha representa uma instância de uma entidade da aplicação e cada coluna um de seus atributos. De forma diferente, bancos de dados NoSQL [Sadalage and Fowler 2013] não seguem o modelo relacional, sendo geralmente classificados de acordo com sua arquitetura, modelo de dados e linguagem de consulta. Os principais modelos de dados são: orientado a colunas, chave-valor, grafo e orientado a documentos [Lóscio et al. 2011].

Este trabalho tem como foco a conversão de bancos relacionais para bancos de dados NoSQL orientados a documentos. Os bancos de dados orientados a documentos se caracterizam por persistir os dados em formato JSON (*Javascript Object Notation*), permitindo armazenar dados estruturados e semi-estruturados, possibilitando representar os dados de múltiplas formas, de forma normalizada e/ou desnormalizada. Neste trabalho é utilizado o banco de dados MongoDB[1], um dos bancos de dados NoSQL amplamente utilizado pela comunidade.

Os grandes modelos de linguagem (LLMs do inglês, *Large Language Models*), como ChatGPT-4 e Gemini são ferramentas para processamento de linguagem natural (PLN), que é uma área da inteligência artificial que tem como foco a interpretação da linguagem humana por computadores. A interação com as LLMs é realizada por meio do *prompt*, uma descrição textual de uma tarefa a ser realizada. É através da interpretação desse texto que o modelo de linguagem compreende a tarefa a ser executada e gera uma resposta. O processo de elaboração dos *prompts* é conhecido como *prompt engineering*, e envolve o uso de técnicas, como o *few-shot* e o *chain-of-thought*, por exemplo, para aumentar a precisão e a relevância das respostas geradas pelas LLMs.

As LLMs possuem a capacidade de generalizar a partir de grandes conjuntos de dados, permitindo seu uso em diversos cenários, incluindo o processo de migração entre bases de dados [Minaee et al. 2025]. Essas características permitem seu uso no processo

[1] https://www.mongodb.com/

de migrar dados entre bancos de dados relacionais e não relacionais, por meio de técnicas de *prompt engineering*, para extrair metadados do RDB, analisar consultas existentes (carga de trabalho) e sugerir esquemas de dados que priorizem determinados padrões de acesso, fornecendo subsídios para a migração de dados.

3. Abordagem de migração de dados existente

A ferramenta de migração de dados [Marmentini and Kuszera 2025] que este trabalho estende tem por objetivo automatizar o processo de conversão e migração de dados entre banco de dados relacional e banco de dados não relacional, especificamente o MongoDB. Ela realiza etapas do processo de migração, incluindo a identificação dos metadados do banco de origem, geração de artefatos intermediários, até a migração efetiva dos dados para a base de destino. Com base nos metadados extraídos e com informações do *workload* do banco de origem, a ferramenta faz uso de uma LLM para definir o esquema de dados no banco de destino.

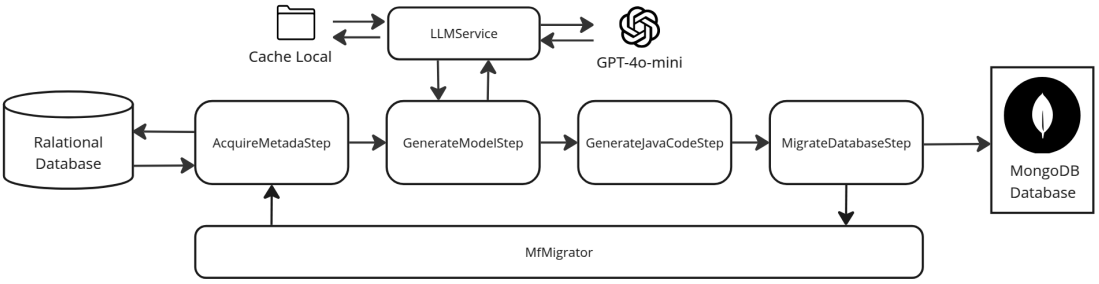

Figura 1. Arquitetura da ferramenta de migração

A Figura 4 apresenta a arquitetura da ferramenta. Ela é implementada na linguagem Java e estruturada como uma *pipeline*, na qual cada etapa da migração recebe um conjunto de parâmetros de entrada e retorna um conjunto de parâmetros de saída. Além disso, cada etapa pode ser executada em conjunto ou de forma individual, possibilitando adaptabilidade nas execuções do processo de migração. Por ser uma biblioteca, sua execução depende da configuração e implementação de uma aplicação externa responsável por controlar o fluxo de execução da *pipeline*, fornecendo os parâmetros necessários para cada etapa e manipulando os resultados retornados.

4. Ferramenta visual para migração de dados

A ferramenta visual proposta neste artigo tem o objetivo de facilitar a utilização da abordagem de migração dos autores, tornando as etapas de configuração e controle do processo de migração mais dinâmica e intuitiva, fornecendo uma apresentação clara e objetiva das informações. Anteriormente, todas as operações de configuração e monitoramento da migração precisavam ser realizadas por meio de código Java, o que exigia conhecimento técnico específico por parte dos usuários. A introdução da interface visual representa, portanto, um avanço significativo na acessibilidade e no potencial de adoção da ferramenta por um público mais amplo.

4.1. Arquitetura

A Figura 2 apresenta a arquitetura da aplicação *web*. A interface foi desenvolvida como uma aplicação web, possibilitando executar em diversas plataformas. Há dois compo-

nentes principais: cliente e servidor. As seções seguintes descrevem os componentes em detalhes.

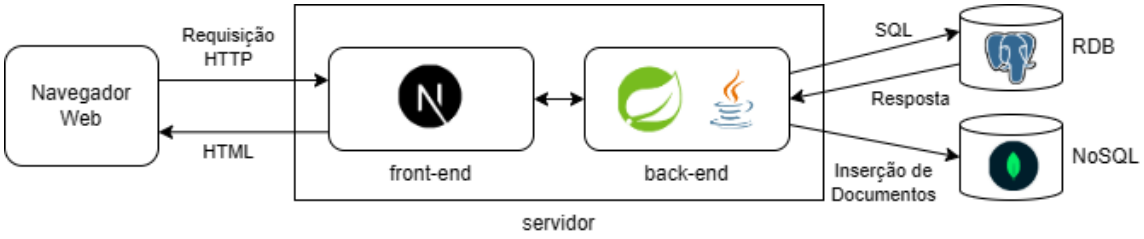

Figura 2. Arquitetura da ferramenta visual desenvolvida

4.2. Servidor

Ambiente responsável por processar as requisições do cliente, executar o processo de migração e gerenciar os dados. É composto pelo componentes *front-end* e *back-end*. O **front-end** é responsável por receber as requisições do cliente, repassando as requisições para o *back-end* quando necessário. O *front-end* foi desenvolvido em *TypeScript* utilizando o *framework Next.Js*. O **back-end** é uma API REST que controla as execuções das etapas da *pipeline* de migração. Esse componente foi desenvolvido utilizando a linguagem *Java* junto ao *framework Spring Boot* para a criação de *endpoints* HTTP para a comunicação com o componente *Front-end*.

4.3. Cliente

Ambiente onde o usuário tem interação e controle do processo de migração por meio de um navegador web, podendo configurar e monitorar a execução da migração.

4.4. Bancos de Dados de Origem e de Destino

A ferramenta visual permite a configuração das bases de dados de origem e destino. Assume-se que essas bases já existem no ambiente do usuário. O banco de dados relacional é a base de origem e o banco de dados MongoDB é a base de destino.

4.5. Fluxo de comunicação

Como a arquitetura da ferramenta segue o modelo cliente-servidor, a comunicação entre o cliente e o servidor é baseada em dois mecanismos principais: requisições HTTP e conexões SSE (*Server-Sent Events*). As requisições HTTP são utilizadas para enviar os dados de configuração ao servidor, como as credenciais de acesso aos bancos de dados e preferências sobre o processo de migração por exemplo. Essas requisições possibilitam uma comunicação com o front-end simplificada e possibilidade de integração com outros serviços. O *Server-Sent Events* (SSE) estabelece comunicação em tempo real do servidor para com o cliente, permitindo que o usuário acompanhe cada etapa do processo, sem a necessidade de realizar novas requisições.

Cada etapa de migração possui parâmetros de entrada que precisam ser informados corretamente para a execução dos processos de migração. A ferramenta visual fornece interfaces web para que o usuário envie as informações necessárias e elas sejam validadas antes de serem processadas. O back-end recebe uma requisição HTTP com os parâmetros de entrada e retorna uma resposta com parâmetros de saída, que são utilizados como o corpo (*payload*) da requisição seguinte.

5. Usando a ferramenta visual em cenário de migração de dados

Nesta seção é apresentado o uso da ferramenta visual para realizar a migração de dados entre bases relacional e NoSQL orientadas a documentos. A ferramenta, por meio do uso de LLMs para auxiliar no processo de análise do esquema de origem (relacional) e o conjunto de consultas existentes (carga de trabalho), propõe um esquema de destino e gera código Java para a execução da migração dos dados de origem para o destino, além de disponibilizar o código gerado para a livre utilização pelo usuário.

5.1. Cenário exemplo

O cenário de exemplo considera um banco de dados relacional PostgreSQL que armazena dados de voos entre aeroportos de origem e destino, conforme apresentado na Figura 3. O objetivo é migrar os dados para uma base de dados NoSQL orientada a documentos. Esse cenário será utilizado para exemplificar o funcionamento da ferramenta de migração em conjunto com a aplicação web desenvolvida.

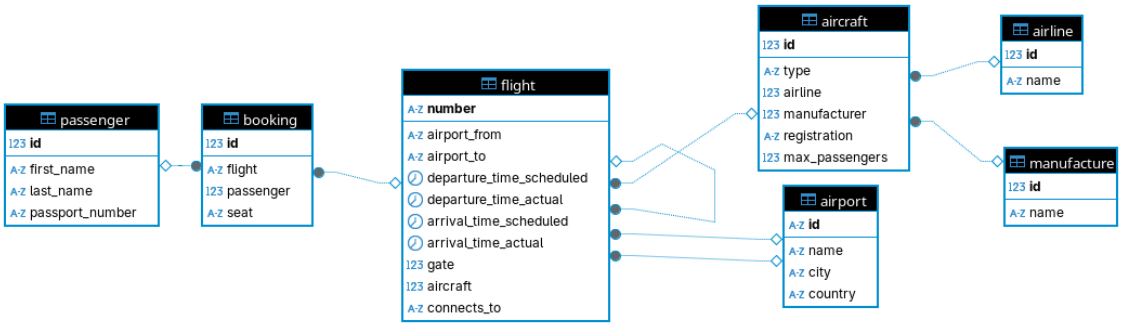

Figura 3. Esquema do banco de dados relacional de origem.

5.2. Etapas da migração

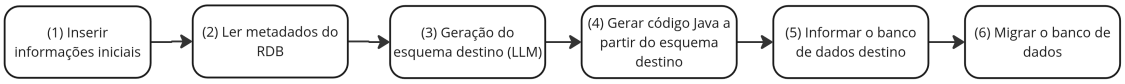

Figura 4. Etapas do processo de migração executado pela ferramenta

A Figura 4 apresenta as etapas executadas pela abordagem de migração, sendo: (1) inserir informações iniciais, (2) ler metadados do banco de dados relacional, (3) geração do esquema destino (LLM), (4) gerar código Java a partir do esquema destino, (5) informar o banco de dados destino, (6) migrar o banco de dados. A seguir essas etapas serão descritas em maiores detalhes.

5.2.1. Inserir informações iniciais

A primeira etapa consiste na configuração inicial da ferramenta para executar a migração de dados. As informações entrada necessárias são apresentadas a seguir.

Credenciais de acesso ao banco de dados de origem (RDB): Primeiramente o usuário deve informar a *string* de conexão do banco de dados de origem (compatível com

o JDBC, informando usuário, senha e nome do banco de dados). A ferramenta valida as informações e acusa se a conexão foi estabelecida com sucesso.

Workload: O usuário opcionalmente pode informar um conjunto de consultas SQL que representa a carga de trabalho habitual do banco de dados. Para cada consulta é necessário informar a frequência de execução (valor em porcentagem). Essas informações são utilizadas pela LLM para adaptar o modelo de dados destino conforme carga de trabalho. As informações inseridas não são validadas antes de serem utilizadas pela ferramenta. A Figura 5 mostra a interface de inserção de *workloads* contendo duas consultas. Atualmente as consultas são inseridas manualmente, mas como trabalho futuro será implementada extração automática de consultas a partir do banco de dados relacional.

Figura 5. Tela para a inserção do *workload*

Customizar geração do esquema destino: Nessa fase são fornecidas informações que visam dar controle ao usuário de como o esquema será gerado pela LLM. É possível informar ao LLM para (*i*) permitir que documentos possuam referências; (*ii*) priorizar desempenho em detrimento da redundância de dados; (*iii*) fornecer informações adicionais que devem ser interpretadas pela LLM.

Configurações da LLM: Nesta fase o usuário deve informar qual modelo de LLM será utilizado, bem como a chave da API válida para a realização da requisição por parte da ferramenta. Atualmente apenas o *gpt-4o-mini* é suportado pela ferramente gráfica. No entanto, é possível adaptar outros modelos da OpenAI, como o *gpt-o4-mini* e *gpt-3o*.

Cada fase acima tem uma página dedicada a inserção das informações solicitadas. Todas as informações inseridas são armazenadas no *localStorage* do navegador[2]. Essa estratégia permite que os dados sejam mantidos mesmo com o recarregamento (*refresh*) das páginas ou o encerramento da aplicação.

5.2.2. Ler metadados do banco de dados relacional

Nesta fase, a ferramenta chama o *back-end* para extrair os metadados da base relacional e identificar as cardinalidades mínimas, máximas e médias de cada relacionamento entre tabelas identificadas. Esses dados são serializados em formato JSON e são enviados ao *front-end* como resposta a requisição realizada.

Assim que as cardinalidades são recebidas pelo front-end, as mesmas são persistidas localmente utilizando o *localStorage* e exibidas ao usuário em formato de lista, como mostra a Figura 6. Atualmente as cardinalidades não podem ser editadas.

[2]https://developer.mozilla.org/pt-BR/docs/Web/API/Window/localStorage

Figura 6. Relacionamentos da base de origem identificados pela ferramenta.

5.2.3. Geração do esquema destino por meio da LLM

O cliente (*front-end*) envia novamente uma requisição HTTP ao servidor (*back-end*) com os metadados e as cardinalidades do banco de dados de origem. O servidor, através de chamadas de funções da ferramenta, envia um *prompt* para a LLM configurada pelo usuário. Como resultado, é gerada uma representação do esquema de dados destino baseada no formato JSON Schema[3], enriquecida com propriedades adicionais. Essas propriedades adicionais estabelecem mapeamentos entre elementos do esquema relacional de origem e elementos do esquema de destino (MongoDB), incluindo como certos atributos ou relacionamentos devem ser implementados no banco de destino. Após gerado o esquema de dados, o JSON Schema é enviado ao *front-end* como resposta da requisição HTTP.

Para possibilitar uma compreensão melhor do esquema gerado, no *front-end*, um grafo é criado a partir do esquema de dados recebido do *back-end*. Nesse grafo, os vértices representam as coleções, listando também as propriedades dos documentos. As arestas representam os relacionamentos entre os documentos, podendo ser de dois tipos: (*i*) contínuos: relação de aninhamento entre documentos e (*ii*) segmentado: relação de referência entre documentos. Considerando o cenário exemplo, a Figura 7 mostra o grafo criado pela LLM a partir do esquema do banco de dados relacional e demais instruções fornecidas.

5.2.4. Gerar código Java a partir do esquema destino

Após a visualização do esquema pelo usuário e posterior confirmação, a ferramenta utiliza um algoritmo que transforma o JSON Schema gerado pela LLM em código-fonte Java executável. Além de ser utilizada para converter e persistir os dados entre a base relacional e a base orientada a documentos, o código gerado é exibido no front-end por meio de uma caixa de texto. O usuário pode utilizar esse código em suas próprias aplicações.

5.2.5. Informar o banco de dados destino

Antes de iniciar a migração dos dados, o usuário deve informar o banco de dados de destino. Há uma página dedicada a inserção do *host*, porta, nome de usuário e senha de

[3]https://json-schema.org/

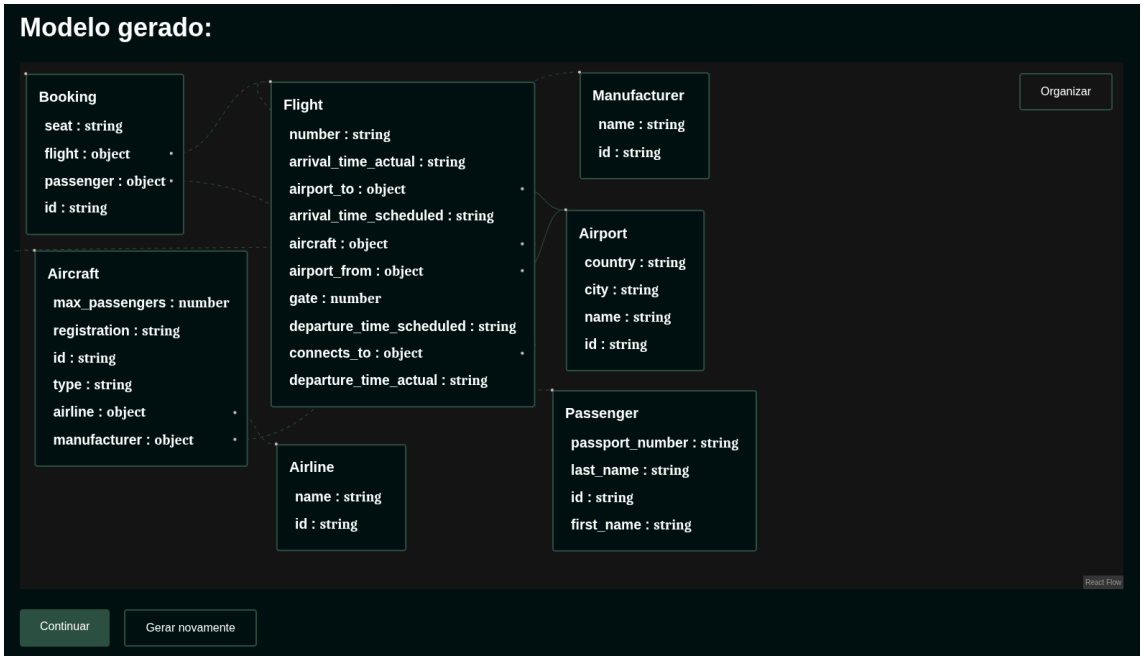

Figura 7. Grafo representando o modelo de dados gerado pela LLM

uma instância do MongoDB aonde os dados serão persistidos. A ferramenta valida as informações e acusa se a conexão foi estabelecida com sucesso.

5.2.6. Migração de dados

O cliente envia uma nova requisição HTTP ao servidor, contendo as credenciais de acesso ao banco de dados de destino e o código-fonte gerado. O processo de migração é iniciado e uma conexão do tipo SSE (Server-Sent Events) é estabelecida para permitir que o servidor notifique o cliente, em tempo real, sobre o andamento da migração (qual tabela está sendo processada, por exemplo).

Na sequência, são disparados processos para compilação de código, depois extração, conversão e persistência dos dados. De forma mais detalhada: *i)* as classes geradas são compiladas e armazenadas em memória; *ii)* a partir das classes Java compiladas são realizadas consultas sobre a base de dados relacional (origem), resultando na carga de objetos em memória; *iii)* esses objetos com dados do banco relacional são convertidos para objetos que representam o modelo de destino (NoSQL); *iv)* por fim, os objetos convertidos são persistidos no banco de dados de destino (MongoDB).

A Figura 8 mostra a interface de *feedback* exibida ao usuário durante a execução do cenário de teste.

5.2.7. Relatório de migração

Ao final da migração é gerado um relatório do processo executado e de artefatos que foram gerados, incluindo: *i)* representação do banco de dados de origem em formato SQL; *ii)* relacionamentos e cardinalidades identificados pela ferramenta em formato JSON, re-

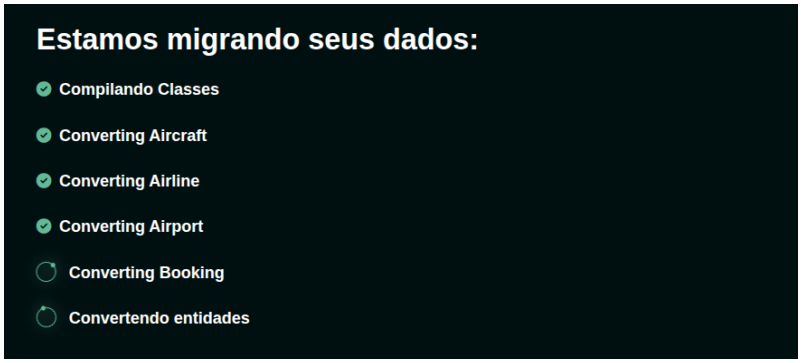

Figura 8. Interface de *feedback* sobre o processo de conversão e persistência dos dados

sultado da etapa *Ler metadados do banco de dados relacional* (Seção 5.2.2); *iii) workload* informado pelo usuário; *iv)* parágrafo gerado pela LLM explicando o modelo gerado e as razões pelas quais foi estruturado de tal forma; *v)* JSON Schema representando o modelo de dados de destino gerado pela LLM; *vi)* contagem de registros migrados separados por tabela.

Após a exibição do relatório o processo de migração é encerrado, não havendo interações possíveis de serem realizadas pelo usuário.

5.3. Resultados obtidos

A interface gráfica contribuiu para melhorar a usabilidade da ferramenta de migração proposta pelo autores em trabalhos anteriores, possibilitando configurar e acompanhar a execução do processo de migração, além de visualizar graficamente o modelo de dados gerado pela LLM. Apesar da ferramenta de migração permitir a interação via código Java, o uso da interface gráfica desenvolvida nestes artigo facilita o uso e execução de migração de dados por usuários não especialistas em programação.

6. Considerações Finais

Este trabalho apresentou o desenvolvimento de uma interface visual voltada para o gerenciamento do processo de migração de dados entre bancos de dados relacionais e NoSQL orientado a documentos. A aplicação foi projetada para que o usuário configure e acompanhe as etapas da migração por meio de uma interface gráfica intuitiva. Desenvolvida como uma aplicação web, seguindo o modelo cliente-servidor, integra e orquestra a ferramenta de migração desenvolvida pelos autores em trabalhos anteriores. A interface permite que o usuário forneça os parâmetros necessários para cada etapa e acompanhe o progresso da migração.

Além disso, não foi desenvolvido um mecanismo persistência de dados que suporte o armazenamento do histórico de migrações realizadas, sendo armazenado apenas as configurações do último processo de migração executado. Sendo assim, a inclusão das funcionalidades de editar o modelo de dados gerado, bem como um histórico de migração em que o usuário possa facilmente reexecutar um processo anterior são propostas de desenvolvimento para trabalhos futuros.

Como trabalho futuro, pretende-se desenvolver histórico de migrações realizadas

e também funcionalidade de editar o modelo de dados gerado pela LLM, a fim de permitir que o usuário customize o processo de migração.

Referências

Jia, T., Zhao, X., Wang, Z., Gong, D., and Ding, G. (2016). Model Transformation and Data Migration from Relational Database to MongoDB. In *2016 IEEE International Congress on Big Data (BigData Congress)*, pages 60–67.

Karnitis, G. and Arnicans, G. (2015). Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation. In *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*, pages 113–118.

Lee, C. and Zheng, Y. (2015). SQL-to-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2022–2026.

Lóscio, B. F., OLIVEIRA, H. d., and PONTES, J. d. S. (2011). Nosql no desenvolvimento de aplicações web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, 10(1):11.

Marmentini, L. and Kuszera, E. (2025). Abordagem de migração de bases relacionais para bases orientadas a documentos apoiada por llm. In *Anais do XL Simpósio Brasileiro de Bancos de Dados*, pages 441–454, Porto Alegre, RS, Brasil. SBC.

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. (2025). Large language models: A survey.

Mior, M. J., Salem, K., Aboulnaga, A., and Liu, R. (2016). NoSE: Schema design for NoSQL applications. In *2016 IEEE 32nd ICDE*, pages 181–192.

Sadalage, P. and Fowler, M. (2013). *NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota*. Novatec Editora.

Santos, M. Y. and Costa, C. (2016). Data Models in NoSQL Databases for Big Data Contexts. In Tan, Y. and Shi, Y., editors, *Data Mining and Big Data*, pages 475–485, Cham. Springer International Publishing.

Serrano, D., Han, D., and Stroulia, E. (2015). From Relations to Multi-dimensional Maps: Towards an SQL-to-HBase Transformation Methodology. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 81–89.

Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., and Helland, P. (2007). The end of an architectural era (it's time for a complete rewrite). In *Proceedings of the 33rd VLDB, University of Vienna, Austria, 2007*, pages 1150–1160.

Vajk, T., Fehér, P., Fekete, K., and Charaf, H. (2013). Denormalizing data into schema-free databases. In *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 747–752.

Zhao, G., Li, L., Li, Z., and Lin, Q. (2014). Multiple Nested Schema of HBase for Migration from SQL. In *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 338–343.