UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE INFORMÁTICA

VICTOR PICUSSA

# SISTEMA DE INTEGRAÇÃO DE DADOS COM INTERFACE GRÁFICA E MAPEAMENTOS AUTOMATIZADOS

CURITIBA
2021

VICTOR PICUSSA

# SISTEMA DE INTEGRAÇÃO DE DADOS COM INTERFACE GRÁFICA E MAPEAMENTOS AUTOMATIZADOS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação da Universidade Federal do Paraná como parte das exigências para a obtenção do título de Bacharelado em Ciência da Computação.

Orientador: Prof. Dr. Marcos Didonet Del Fabro

CURITIBA
2021

*"Em algum lugar, algo incrível está esperando para ser descoberto."*
*Carl Sagan*

## AGRADECIMENTOS

Divido aqui os meus agradecimentos a todos que foram importantes para esse capítulo da minha vida, que colaboraram de alguma maneira para eu alcançar meus objetivos, e que fizeram e farão parte da minha história. Agradeço, então:

- aos professores do Departamento de Informática e de Exatas que conduziram as matérias e passaram seus ensinamentos, encontrando-se sempre disponíveis para ajudar, definindo um curso único.
- à minha mãe, Elza Maria Juliatto Picussa, que me incentivou e apoiou em todas as decisões, ajudando no dia a dia, fazendo eu ser a melhor versão de mim.
- à meu pai, Valmor Picussa, que me introduziu no mundo da tecnologia e permitiu que eu seguisse meus sonhos.
- ao meu professor orientador, Prof. Marcos Didonet Del Fabro, que esteve sempre disponível a ajudar com a ideia proposta e a construção do trabalho, além de ter passados seus ensinamentos durante o curso, como também um excelente profissional coordenando projetos no C3SL.
- aos meus grandes amigos, Axel Rocha Valene e Gustavo Hornig de Meira, que dividiram suas experiências e compartilharam a graduação, as dificuldades e as conquistas durante o caminho.
- às diversas amizades que obtive durante a graduação, que fizeram parte de estudos e trabalhos, que criaram dias mais interessantes e divertidos, e que espero um dia poder dividir minhas experiências novamente.

## RESUMO

A diminuição do custo e tempo de produção de dados no inicio do século XXI transformou o mundo digital, promovendo um crescimento constante de dados, como também a democratização da utilização através dos dados abertos. Esses dados abertos, geralmente, não seguem um padrão e não possuem documentações, o que causa uma dificuldade de manipulação e análise. Para resolver esse problema existem sistemas de integração de dados e métodos de mapeamento. Muitos dos métodos de mapeamento são aplicados manualmente, o que demanda tempo e tem alto custo. Os sistemas disponíveis atualmente possuem diversas características, porém, nenhum aplica ao mesmo tempo uma interface gráfica, um mapeamento automatizado, e a disponibilização de código aberto. O objetivo desse trabalho é apresentar um sistema com essas características. Dessa forma, o sistema disponibiliza uma interface gráfica de usuário sobre uma ferramenta denominada HOTMapper, utilizando um método de geração de mapeamento de dados automatizada. Telas são definidas utilizando a ferramenta de integração de dados, como também é apresentado a tela que aplica o método de mapeamento automatizado. Por fim, é visualizado todo o fluxo de execução do sistema de integração de dados com a aplicação do mapeamento automatizado e, como será visto, é obtido sucesso em todo o processo de teste confirmando a composição das características citadas anteriormente.



**Palavras-chave**: interface gráfica; integração de dados; dados abertos; mapeamento de dados.

## ABSTRACT

The decrease in the cost and time of data production at the beginning of the 21st century has transformed the digital world, promoting a constant growth of data, as well as the democratization of use through open data. These open data, generally, they do not follow a pattern and do not have documentation, which makes manipulation and analysis difficult. To solve this problem there are data integration systems and mapping methods. Many of the mapping methods are applied manually, which is time consuming and costly. The systems currently available have several characteristics, however, none of them apply a graphical interface, an automated mapping, and the availability of open source at the same time. The objective of this work is to present a system with these characteristics. Thus, the system provides a graphical user interface on a tool called HOTMapper, using an automated data mapping generation method. Screens are defined using the data integration tool, as well as the screen that applies the automated mapping method. Finally, the entire execution flow of the data integration system is visualized with the application of the automated mapping and, as will be seen, success is obtained in the entire test process, confirming the composition of the characteristics mentioned above.



**Keywords**: graphical interface; data integration; open data; data mapping.

## LISTA DE FIGURAS

# LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| BD | Banco de dados |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| CSV | Comma-separated Value |
| XML | Extensible Markup Language |
| CRLF | Carriage Return Line Feed |
| CLI | Command Line Interface |
| GUI | Graphical User Interface |
| ETL | Extract, Transform and Load |
| SQL | Structured Query Language |
| ML | Machine Learning |
| JSON | Javascript Object Notation |

## SUMÁRIO

# 1 INTRODUÇÃO

A produção de dados está em constante crescimento no mundo digital desde a popularização da internet. Os conjuntos de dados estão cada vez maiores e disponibilizam cada vez mais informações. A produção dos dados tornou-se menos custosa a partir do século XXI (KITCHIN, 2014), em termos de tempo e dinheiro, permitindo o crescimento da geração de dados abertos, que tornam-se cada vez mais populares no contexto mundial e político.

Os dados abertos são uma forma de democratizar o acesso a informação por qualquer usuário ou empresa que possua alguma noção de análise e manipulação de dados. Para dados serem considerados abertos, devem seguir convenções de acessibilidade, disponibilidade, compartilhamento, reuso e universalidade (FOUNDATION, 2015). Com a grande quantidade de dados abertos e não padronizados, sistemas de integração de dados são progressivamente mais importantes para analisar e agrupar conjuntos de dados.

Os sistemas de integração de dados aplicam um processo de transformação de dados que utilizam mapeamento de dados. Esses mapeamentos são, usualmente, definidos manualmente. A geração manual desses mapeamento pode ser longa e custosa, estando a mercê de erros humanos na definição das correlações de dados. Aplicações automatizadas de mapeamento de dados são apresentadas em estudos como (MILLER, 2018) e (BERLIN; MOTRO, 2002), utilizando valores dos conjuntos de dados para correlacioná-los. Os sistemas de integração de dados em geral, como os que serão visualizados neste trabalho, possuem diversas funcionalidades que complementam a integração, porém, nenhum constitui simultaneamente das funcionalidades de mapeamento automatizado, interface gráfica e código aberto.

O objetivo desse presente trabalho é apresentar um sistema de integração de dados que define uma interface gráfica sobre o *HOTMapper* (EHRENFRIED et al., 2019), uma ferramenta de integração de dados históricos, para facilitar e promover acessibilidade a ferramentas de integração de dados, apresentando e implementando um método automatizado de geração de mapeamentos de dados integrado ao sistema. Para atingir esse objetivo será definida uma arquitetura do sistema proposto, apresentando as telas e detalhes implementados que determinam a interface gráfica com funcionalidades de uma ferramenta de integração de dados, além da geração de mapeamentos básicos e automatizados. Mais do que percorrer todas as funcionalidades das telas, da aplicação da ferramenta de integração de dados e do funcionamento do mapeamento automatizado através das fórmulas, o sistema será testado com conjuntos de dados governamentais relevantes, demonstrando o fluxo de geração de mapeamento básico, como também o mapeamento automatizado, a análise das saídas, e um fluxo da execução do sistema a partir da configuração de tabela e mapeamento resultado.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 Banco de Dados

Os bancos de dados são repositórios que persistem conjuntos de dados ou informações através de arquivos que, estruturados de certa maneira, possibilitam a aplicação de funções de busca, de alteração, e remoção de dados com sistemas de gerenciamento (DATE, 2004). Sistemas que manipulam grandes quantidades de dados para processamento utilizam os bancos de dados para armazenar e buscar de forma eficiente as informações, como também para definir e organizar os dados.

Os dados armazenados nos arquivos de banco de dados estão estruturados de forma que é necessário um sistema para que usuários possam acessá-los, como um sistema de gerenciamento de banco de dados (SGBD), representado na Figura 1. Esses sistemas permitem a manutenção dos registros através de operações para armazenar novos dados, alterar dados existentes e aplicar exclusões de dados (DATE, 2004). Muitos dos SGBDs também providenciam a execução de fórmulas matemáticas na busca de dados como média, mediana, etc, que podem ser utilizadas para análises mais complexas de dados.

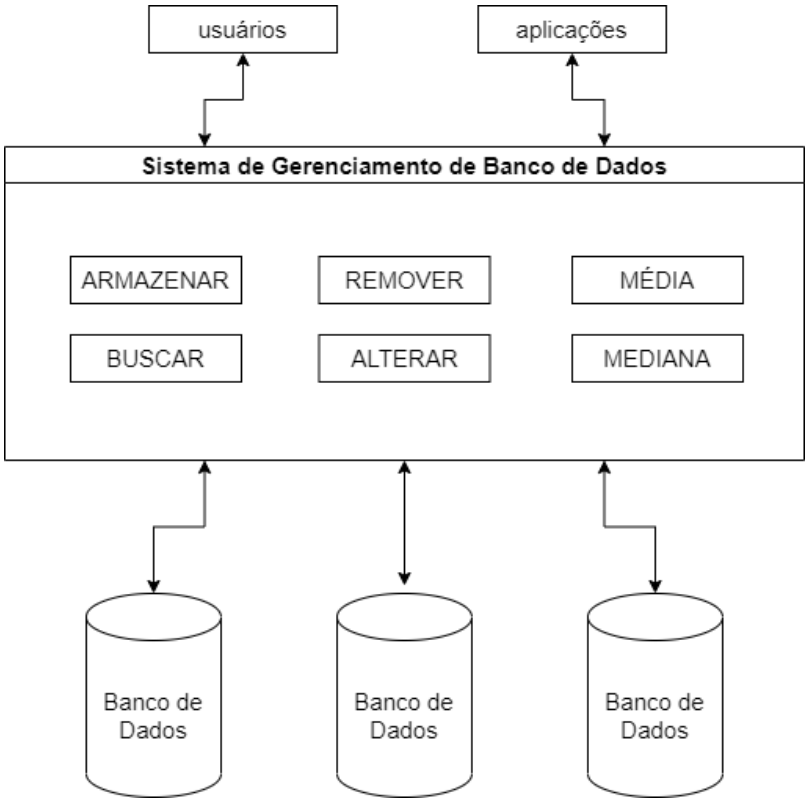



Figura 1 – Exemplo da conexão entre SGBD e BDs.

Fonte: autoria própria, 2021.

O modelo mais popular de banco de dados é o relacional, onde é definido a partir do modelo relacional de dados (CODD, 1970). Os bancos de dados relacionais aplicam a estruturalização dos dados a partir de tabelas (ou relações) que são formadas por linhas e colunas respeitando restrições de integralidade (DATE, 2004). As tabelas representam entidades específicas (e.g. docentes ou alunos) que armazenam dados relacionados a ela. Cada entrada de um conjunto de dados é uma linha representada na tabela do banco de dados relacional; essas linhas também podem ser chamadas de registros ou tuplas de uma tabela e são divididas pelas colunas. As colunas de uma tabela definem os atributos dos dados, separando e organizando um conjunto de dados dando uma definição para cada entrada (e.g. colunas que representam nome, idade, endereço, telefone, etc) como também o tipo do dado da coluna (e.g. texto, inteiro, ponto flutuante, etc). Dessa forma, uma tupla que contém todos os atributos relacionados a ela será representada como uma entrada de uma tabela na visualização dos bancos de dados relacionais.

Considerando a Figura 2, pode-se observar uma tabela representando a entidade *Alunos* de um banco de dados relacional. Essa tabela é composta pelas colunas: *ID* - campo inteiro que identifica uma tupla; *Nome* - campo texto que constitui o nome do aluno; *Idade* - campo inteiro que define a idade do aluno; *IRA* - campo ponto flutuante que apresenta a média das notas do aluno. Além das colunas, a tabela também possui 5 tuplas que armazenam os dados, como por exemplo a primeira tupla, que é identificada pelo ID 1, representa o aluno Victor com 25 anos de idade e IRA de 0.75.

Para poder inserir dados em um banco de dados existem diversas técnicas, porém, a mais popular e mais eficiente é com a manipulação de arquivos CSV com scripts que facilitam a persistência dos dados com SGBDs.

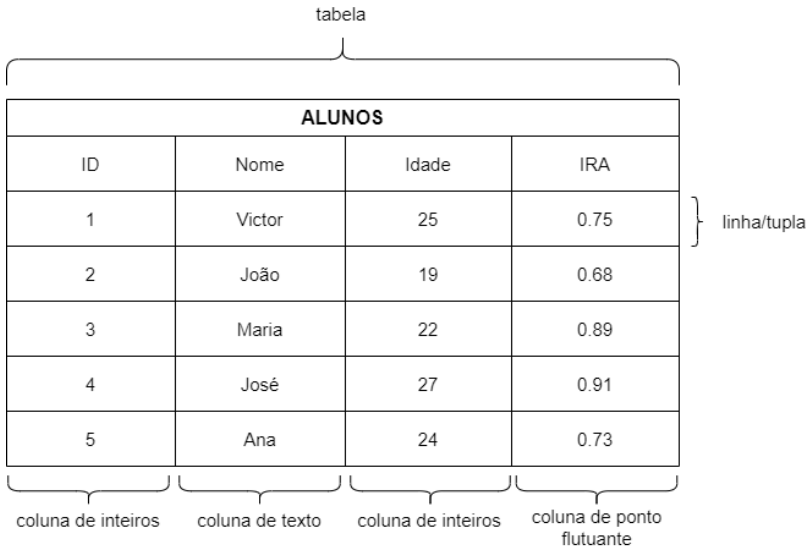



Figura 2 – Exemplo de uma tabela de um banco de dados relacional.

Fonte: autoria própria, 2021.

## 2.2 Arquivos CSV

Arquivos texto regulamentados pelo RFC 4180 (SHAFRANOVICH, 2005), são arquivos do tipo CSV (Comma-Separated Values) utilizados na criação de planilhas digitais em softwares como LibreOffice Calc e Microsoft Excel. Esse tipo de arquivo é definido por um conjunto de dados separados por um delimitador, como a vírgula, onde o dado entre cada vírgula é uma coluna da tabela. A separação de cada tupla de uma tabela é presentada pela quebra de linha, podendo ser "*CRLF*" (Windows) ou o caractere especial "$\backslash n$" (Linux). Como um formato opcional adicional aos arquivos CSV que representam uma tabela, a primeira linha pode definir um cabeçalho para dar nome as colunas de dados (REPICI, 2004) e, para isso, é necessário que haja o mesmo número de campos com separador para cada linha do resto do arquivo.

Como os delimitadores dos arquivos CSV não são especificados formalmente (SHAFRANOVICH, 2005), pode-se utilizar outros delimitadores além da vírgula, como ponto-e-vírgula, barra vertical, e outros. Essa pluralidade de delimitadores não é uma complicação para os softwares que permitem a visualização de arquivos CSV, já que esses softwares adicionam um campo para o usuário especificar qual delimitador o arquivo que será visualizado utiliza. No entanto, é essencial que dados que possuem o mesmo caractere que o delimitador do arquivo estejam entre aspas para não ocorrer um problema na visualização da planilha.

A figura 3 demonstra um arquivo CSV e sua representação em uma planilha digital. Esse arquivo é constituído da tabela *Docentes*, que em sua primeira linha define um cabeçalho contendo todas as colunas que definem os dados. As colunas são estabelecidas pelo delimitador mais comum, a vírgula. Além disso, o CSV contém 3 tuplas respeitando as regras de delimitação, que são os dados separados por vírgula, e número de campos, que são as quantidades de colunas da tabela que é representada.

**docentes.csv**

```
1  ID, Nome, Idade, Matérias
2  1, Marcos, 35, "Banco de dados, TAP"
3  2, Douglas, 42, "Astronomia, Filosofia"
4  3, Ana, 56, "Geografia, Climatologia"
```

**DOCENTES**

| ID | Nome | Idade | Matérias |
|---|---|---|---|
| 1 | Marcos | 35 | Banco de dados, TAP |
| 2 | Douglas | 42 | Astronomia, Filosofia |
| 3 | Ana | 56 | Geografia, Climatologia |

Figura 3 – Exemplo de um arquivo CSV e sua representação em tabela.

Fonte: autoria própria, 2021.

## 2.3 Dados Abertos

Os dados são entendidos de forma geral como informações geradas pela abstração do mundo (data, hora, localizações, músicas, filmes, etc) a partir de representações como texto, números, sons, imagens e outros. A produção tradicional de dados até o século XX era custosa e demandava muito tempo para serem gerados, analisados e interpretados por qualquer instituição ou indivíduo (KITCHIN, 2014). A partir do final do século XX e inicio do século XXI, a produção de dados cresceu de forma surpreendente - consequência da diminuição do custo e tempo de geração de dados - gerando cada vez mais dados abertos e acessíveis.

Os dados abertos surgem a partir da democratização dos dados promovendo uma maior interoperabilidade de sistemas e maior acessibilidade por pessoas físicas. O livre acesso, manipulação e utilização de dados são características inerentes aos dados abertos que devem, no máximo, referir-se a fonte original e o compartilhamento das mesmas licenças.

Baseando-se nas definições de código aberto e software livre, dados abertos devem seguir alguns requisitos (FOUNDATION, 2015):

- **Acessibilidade e Disponibilidade**: os dados devem ser estar sempre disponíveis e em sua completude, sem custos além da reprodução para a pessoa física ou jurídica que fará uso do mesmo, promovendo a acessibilidade a qualquer indivíduo ou coletivo, através da internet. Qualquer informação adicional, como licenças, devem acompanhar os dados.
- **Compartilhamento e reuso**: permissões de reutilização e redistribuição devem ser incluídas na licença e precisam aplicar-se a todos na propagação dos dados sem termos legais adicionais.
- **Universalidade**: qualquer usuário deve estar livre para utilizar, modificar, compilar e compartilhar os dados sem discriminar qualquer pessoa ou grupo ou restringir qualquer finalidade ou campo de atuação.

Embora o termo "dados abertos" tenha surgido em 1995 (ISOTANI; BITTENCOURT, 2015), a atenção na definição do conceito e na visibilidade dos dados abertos demorou a ser discutida. Em 2007, ativistas e usuários da internet juntaram-se para conceituar *Dados Abertos Públicos ou Governamentais*. Em 2008 os Estados Unidos apresentou um memorando sobre *Transparência e Dados Governamentais*, além da criação de um portal digital (U.S. General Services Administration, 2009) para disponibilização de dados abertos a todos os cidadãos norte-americanos. Em 2011, o governo brasileiro fundou a *Open Government Partnership* como também um portal digital de dados abertos (Governo Brasileiro, 2011).

A publicação dos conjuntos de dados abertos é comumente feita com arquivos do tipo CSV. Verificando o portal de dados abertos brasileiro, existem 10387 conjuntos de dados, dos quais 7171 disponibilizam o formato CSV (acessado em 11 de junho de 2021). Em comparação ao portal brasileiro, o portal norte-americano disponibiliza 299769 conjuntos de dados onde 17739 possuem arquivos em CSV (acessado em 11 de junho de 2021). O portal

norte-americano possui uma notável diferença na quantidade de conjuntos de dados e conjuntos que disponibilizam CSV, essa diferença é dada pela grande produção de dados climatológicos que utilizam somente o formato HTML para web.

Apesar do formato CSV ser popular e descomplicado na utilização e manuseio de dados abertos, a grande produção e transformação de dados ao longo dos anos apresenta uma dificuldade de relacionamento de conjuntos de dados que fazem parte de um mesmo contexto, problema o qual é resolvido com abordagens de integração de dados.

## 2.4 Integração de Dados

O objetivo da integração de dados é unificar o acesso a diferentes conjuntos de dados para facilitar a visualização pelo usuário (LENZERINI, 2002). Muitas técnicas de unificação utilizam a resolução de perguntas pela conjunção de predicados e *views*, "recursos que são utilizados por um integrador internamente para ajudar a responder perguntas" (ULLMAN, 2000). Para a aplicação dessa unificação em banco de dados são utilizados os esquemas, que são estruturas que descrevem a organização de um conjunto de dados, indicando como as entidades de um banco de dados se relacionam.

Uma das primeiras formas da utilização da integração de dados aplicava a criação de um esquema global [1], a partir de esquemas locais, para o relacionamento dos dados (BATINI; LENZERINI; NAVATHE, 1986). Essa técnica é utilizada com federação de dados [2], já que os esquemas de dados são bem definidos e a coleção de dados é estática (não sofre mutações nos dados) (HAAS; LIN; ROTH, 2002). Uma técnica recente explora métodos de aprendizado de máquina para extrair e concatenar características com a finalidade de unificar os dados (LI; NGOM, 2015). Outra abordagem apresenta uma estrutura de índices [3] distribuída que resolve o problema de busca de tabelas juntáveis (tabelas que podem ser mescladas em uma única tabela) como um problema de busca de domínio em que cada atributo de uma tabela é um domínio (MILLER, 2018). Um importante passo que as abordagens apresentadas utilizam para a integração de dados é o mapeamento de atributos, que busca classificar os conjuntos de dados a fim de unificar as informações.

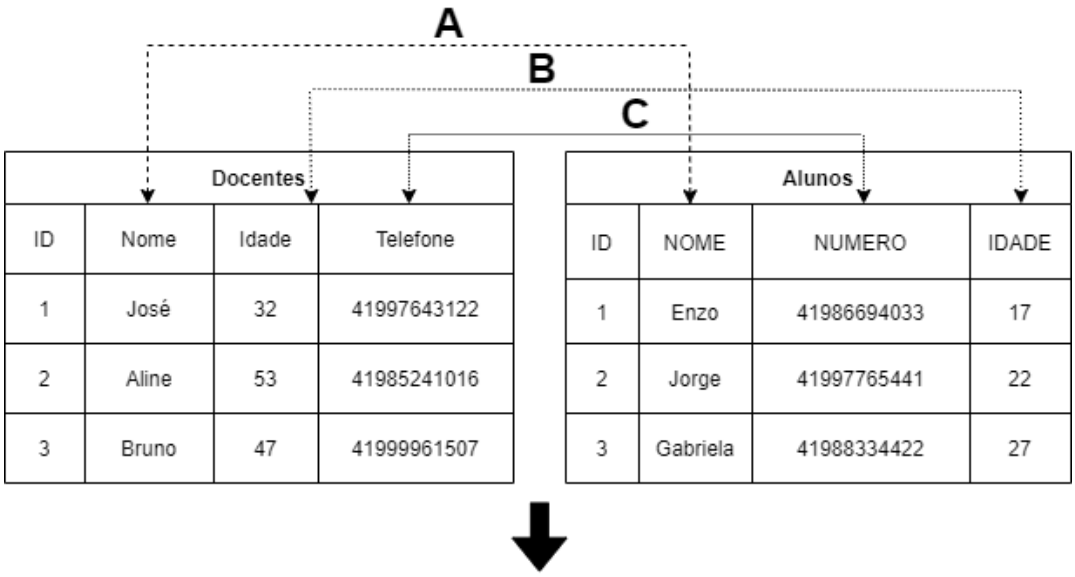
### 2.4.1 Mapeamento de Dados

A integração de conjuntos de dados com o métodos de mapeamento de atributos é feito através do alinhamento de esquemas com diferentes estruturações. Esses métodos recebem entidades discretas (e.g. tabelas e elementos XML) como entrada e retornam as relações entre elas (SHVAIKO; EUZENAT, 2005). De forma abrangente, o processo de mapeamento envolve analisar todos os atributos de conjuntos de dados heterogêneos (dados que possuem alguma

[1] coleção de esquemas relacionados em um único esquema
[2] agregação de dados de fontes diferentes em um banco de dados único
[3] conjunto de valores que referenciam chaves para otimizar a busca de dados

discrepância estrutural), com todas as suas tuplas ou uma amostra suficiente, para verificar similaridades que possibilitam inferir a correspondência de atributos.

Seguindo a Figura 4, podemos ver a integração de dois conjuntos de dados através do mapeamento de seus arquivos no formato CSV. Para uma visualização mais clara foi utilizado o formato de tabelas. O mapeamento dos atributos é realizado com a análise dos conteúdos e tipo dos dados, definido então como: o mapeamento *A* corresponde à coluna *Nome* da tabela *Docentes* com a coluna *NOME* da tabela *Alunos*; o mapeamento B corresponde à coluna *Idade* da tabela *Docentes* com a coluna *IDADE* da tabela *Alunos*; o mapeamento C corresponde à coluna *Telefone* da tabela *Docentes* com a coluna *NUMERO* da tabela *Alunos*. Esse mapeamento gera a tabela *Pessoas*, a qual apresenta a integração dos dados com os atributos que continham alguma similaridade.




**Pessoas**

| ID | Nome | Idade | Contato |
|---|---|---|---|
| 1 | José | 32 | 41997643122 |
| 2 | Aline | 53 | 41985241016 |
| 3 | Bruno | 47 | 41999961507 |
| 4 | Enzo | 17 | 41986694033 |
| 5 | Jorge | 22 | 41997765441 |
| 6 | Gabriela | 27 | 41988334422 |

Figura 4 – Exemplo de integração de dados com um mapeamento de atributos.

Fonte: autoria própria, 2021.

É muito comum que mapeamentos de dados como o apresentado anteriormente sejam feitos de forma manual, o que é ineficiente e demorado. Muitas das abordagens de mapeamento podem ser aplicadas de forma automatizada como o *Clio* (FAGIN et al., 2009) e o *Exsmal* (CHUKMOL; RIFAIEH; BENHARKAT, 2005).

### 2.4.2 Mapeamentos Automáticos

Os mapeamentos automáticos são, cada vez mais, uma necessidade no processamento de dados abertos. As técnicas automatizadas removem a necessidade do usuário fazer o processo de correlação de dados manualmente. A aplicação de mapeamentos automatizados com dados abertos históricos, isto é, dados que possuem informações anuais, semestrais, etc. que estão sujeitos a mudanças estruturais a cada produção de um conjunto de dados subsequente, facilita a integração de dados.

A aplicação de mapeamentos utilizando sistemas automatizados diminui o custo do processamento de dados como também diminui o tempo gasto pelo usuário para mapear dados. Alguns sistemas de mapeamentos automatizados lidam com a interação do usuário para decisões no processo de mapeamentos, como na limpeza de dados (MILLER, 2018) - processo de remoção ou alteração de dados corrompidos (RAHM; DO, 2000). Os mapeamentos automatizados podem utilizar diversas técnicas entre conjuntos de dados heterogêneos, como a comparação de dados de registros e seus tipos, como também a comparação do nome de colunas, caso exista.

### 2.4.3 String Matching

A comparação entre caracteres, palavras ou textos é conhecido como *string matching*. As técnicas de *string matching* são dividas em: *exact matching* - quando utiliza-se uma busca da coincidência exata entre palavras; *approximate matching* - quando utiliza-se uma busca da coincidência aproximada entre palavras. Essas técnicas de comparação são muito utilizadas com os métodos de mapeamento para buscar a correlação de conjuntos de dados e gerar a unificação dos dados (SINGLA; GARG, 2012).

A Figura 5 apresenta a aplicação de um método de aproximação de correlação de *string*. No lado esquerdo da imagem tem-se o padrão de texto que será buscado no conjunto de palavras que estão no lado direito. Como esse método busca uma aproximação, ele retornará pelo menos uma palavra, a qual é a mais parecida dentro do conjunto de palavras providenciado.

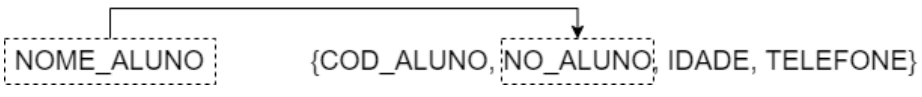



Figura 5 – Exemplo da aplicação de um método de *string matching*.

Fonte: autoria própria, 2021.

## 2.5 Interface

A interface pode ser descrita como uma fronteira de comunicação entre componentes de diferentes sistemas, os quais fazem a troca de informações (HOOKWAY, 2014). As interfaces podem ser definidas em terminais, com linhas de comando como os CLIs (Command Line Interface, ou Interface de Linha de Comando), como também utilizando interfaces gráficas como os GUIs (Graphical User Interface, ou Interface Gráfica do Usuário). Essas interfaces permitem que usuários e outros sistemas utilizem funcionalidades intermediárias para a execução de comandos de outros sistemas de forma acessível. A Figura 6 apresenta, no lado esquerdo, uma CLI, e no lado direito, uma GUI.

As interfaces por linha de comando requerem um conhecimento muito mais profundo do usuário em relação ao sistema operacional que está utilizando, como também da ferramenta. Dessa forma, os CLIs apresentam uma acessibilidade menor aos usuários e podem necessitar de softwares de terceiros para possibilitar a utilização de todas as suas funções. Já as interfaces gráficas podem ser mais acessíveis, com telas que compõe funcionalidades que um CLI não pode implementar, tornando assim a utilização mais eficiente de sistemas.

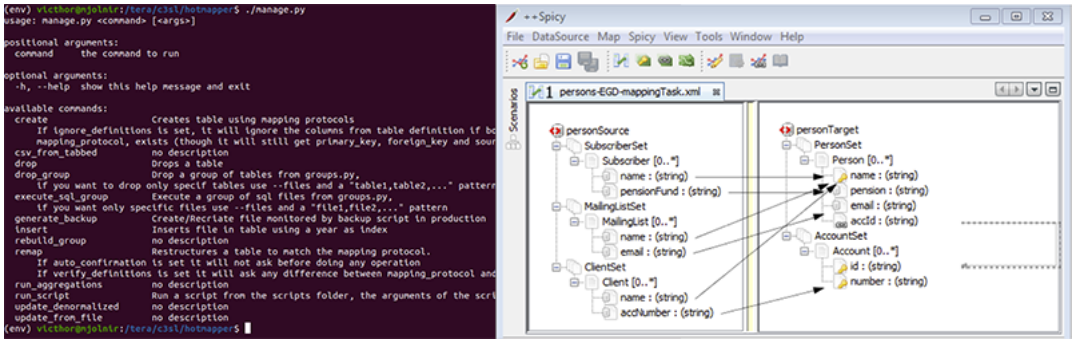



Figura 6 – Exemplo de CLI e GUI.

# 3 TRABALHOS RELACIONADOS

A grande produção de dados e a necessidade da integração, de forma a simplificar a analise e manipulação dos dados, providenciou diversas interfaces para extrair, transformar e armazenar dados (ETL - *Extract, Transform and Load*) em inúmeras áreas da indústria ao longo dos anos. Algumas das interfaces serão exploradas nesse capítulo para analisar suas técnicas e aplicabilidades na integração de dados.

## 3.1 Clio

Um dos mais importantes e populares projetos na área de integração de dados é o *Clio* (FAGIN et al., 2009), que apresenta uma forma de simplificar a integração da informação através de técnicas de mapeamentos e transformação de dados. O projeto foi o pioneiro a usar esquemas de mapeamento para especificar a relação entre conjuntos de dados.

O algoritmo para gerar os mapeamentos apresentado pelo projeto busca testar se elementos de um conjunto máximo de correspondências, onde um conjunto de dados correlaciona a outro, equivalem a mesma associação lógica, nos esquemas fonte e alvo dos conjuntos de dados. Olhando tanto para os pares de associações lógicas das correspondências de fonte como de alvos, o algoritmo inclui as correspondências que utilizam somente os elementos dentro dessas associações. As associações são formas de busca em esquemas singulares, implicando a relação entre todos os elementos.

A Figura 7 exibe um mapeamento de esquemas entre as relações *Concessoes* com os atributos: *gid*, *recipiente*, *quantidade*, *supervisor* e *gerente*; *Companhias* com os atributos: *nome*, *endereco* e *ano*; *Contatos* com os atributos: *cid*, *email* e *telefone*; *Organizacoes* com os atributos: *codigo* e *ano*; *Fundos* com os atributos: *fid* e *finid*; *Financas* com os atributos: *finid*, *despesas* e *telefone*.

```
para cada g em concessoes, c em companhias, s em contatos, m em contatos
    onde g.recipiente = c.nome e g.supervisor = s.cid
        e g.gerente = m.cid
existe o em organizacoes, f em o.fundos, i em financas
    onde f.finId = i.finId
com c.nome = o.codigo e g.gid = f.fid e g.quantidade = i.despesas
```

Figura 7 – Exemplo de um esquema de mapeamento gerado pelo *Clio*.

Fonte: traduzido de (FAGIN et al., 2009).

O projeto emprega o algoritmo proposto com o framework [1] de troca de dados

[1] abstração que unifica códigos entre diversos projetos disponibilizando uma funcionalidade genérica

desenvolvido em (FAGIN et al., 2003), porém, não apresenta uma interface de utilização. Apesar disso, existem projetos inspirados no *Clio* como a ferramenta de código aberto *++Spicy*.

## 3.2 ++Spicy

A ferramenta de código aberto para a segunda geração de mapeamento de esquemas e troca de dados, o *++Spicy* (MARNETTE et al., 2011), apresenta uma interface para trabalhar com a fusão de dados, a limpeza de dados e cenários ETL. Por ser uma segunda geração, a ferramenta aprimora sua primeira versão e permite a criação de cadeias de mapeamentos como também introduz dependências funcionais (condições entre conjuntos de atributos) nos esquemas alvos.

A Figura 8 é uma imagem da GUI definida pelo *++Spicy*. Essa interface apresenta um mapeamento com arquivo XML entre as relações fonte: *Subscriber* (Inscrito), *MailingList* (Liste de Correios) e *Client* (Cliente); e as relações alvo: *Person* (Pessoa) e *Account* (Conta). Seguindo a imagem, os mapeamentos são definidos de forma que os atributos *name* (nome) das relações fonte correspondam ao atributo *name* na relação alvo *Person*; o atributo *pensionFund* (fundo de pensão) da fonte *Subscriber* corresponde ao atributo *pension* (pensão) do alvo *Person*; o atributo *email* da fonte *MailingList* corresponde ao atributo *email* do alvo *Person*; o atributo *accNumber* (número de acc) da fonte *Client* corresponde ao atributo *number* (número) do alvo *Account*. O mapeamento é definido manualmente pelo usuário e as transformações dos dados são aplicadas pela ferramenta.

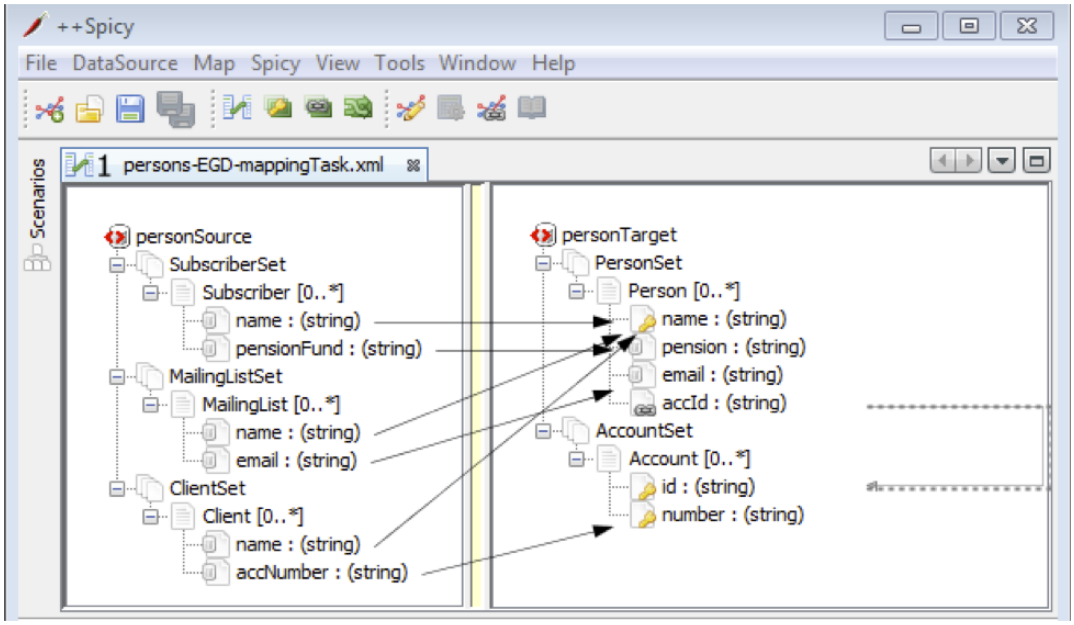



Figura 8 – GUI de mapeamento do *++Spicy*.

Fonte: (MARNETTE et al., 2011).

## 3.3 Metamorfose

Outra abordagem de interface de mapeamento de dados, com uma demonstração da aplicação em conjuntos de dados abertos, é o *Metamorfose* (KUSZERA; PERES; FABRO, 2018), um framework de transformação de dados baseado no Apache Spark (ZAHARIA et al., 2016). Com esse framework o usuário consegue performar ações de ETL para a integração de dados com um método de mapeamento de esquemas através de uma interface gráfica intuitiva. Essa ferramenta permite persistir os dados do resultado das transformações em arquivos no formato CSV, como também em tabelas de bancos de dados relacionais.

Um exemplo da aplicação do mapeamento do *Metamorfose* pode ser vista na Figura 9. Um mapeamento base é definido a partir de um conjunto de dados, no caso os dados de matrículas da educação brasileira de 2013 obtida pelo INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira), e uma tela da interface gráfica é disponibilizada ao usuário para completar o mapeamento. O lado esquerdo em (1) da figura apresenta os atributos obtidos do conjunto de dados a partir de uma configuração de transformação definida pelo usuário. Essas transformações podem simplesmente repetir o atributo ou fazer a concatenação com a palavra reservada *$LIST*, um conjunto de atributos correlacionados, ou até mesmo inserir valores constantes com *$VALUE*. Do lado direito em (2) o usuário define os atributos do esquema alvo do mapeamento de dados como também o tipo dos dados e da transformação a ser aplicada. As transformações são divididas no formato *CASTING*, que fará a cópia exata dos dados, como também *JAVASCRIPT*, que utiliza um código na linguagem javascript definido pelo usuário para transformar a entrada como em (4).

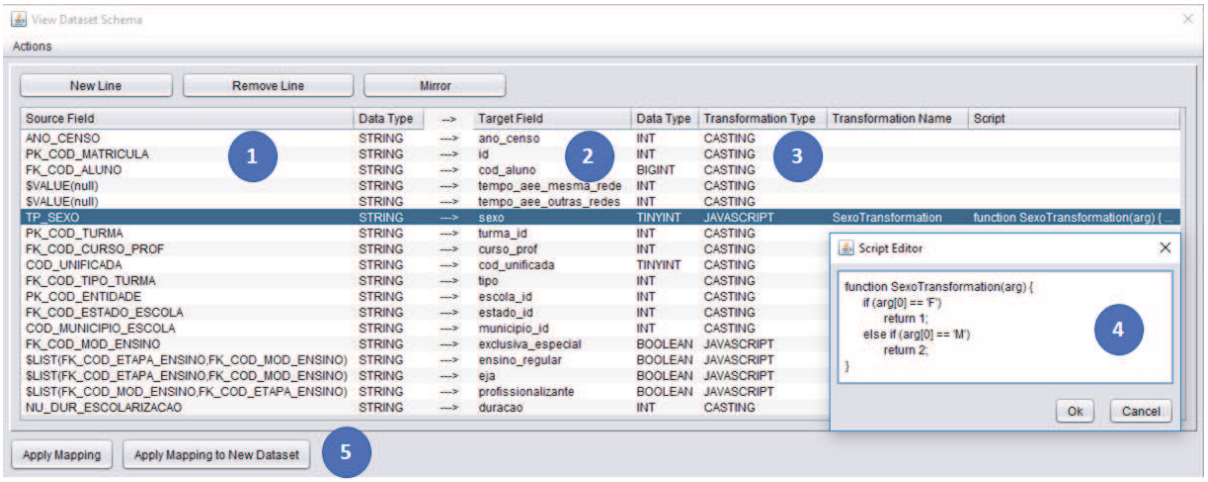



Figura 9 – GUI de mapeamento do *Metamorfose*.

Fonte: (KUSZERA; PERES; FABRO, 2018).

## 3.4 HOTMapper

Uma abordagem interessante de código aberto que é utilizada como base para esse trabalho é a interface de linha de comando *HOTMapper* (EHRENFRIED et al., 2019), ferramenta de mapeamento de dados abertos históricos. Essa ferramenta resolve um problema específico de integração de dados visando dados abertos históricos desnormalizados, que não possuem um padrão de dados. A integração é feita a partir da entrada de conjuntos de dados no formato de arquivo CSV, já o resultado da transformação é persistida em um banco de dados relacional com o SGBD *MonetDB*.

A Figura 10 apresenta a arquitetura do HOTMapper. A CLI recebe arquivos em formato CSV de conjuntos de dados e protocolos de mapeamentos de dados de entrada, implementando também funcionalidades como: criar tabelas a partir de relações pré-definidas *(create)*; inserir dados em uma tabela seguindo o mapeamento relacionado *(insert)*; atualizar uma dada tabela com os dados e mapeamento de entrada *(update)*; remapear uma tabela para atualizar suas definições *(remap)*; deletar a tabela do banco de dados *(drop)*. Dessa forma, o CLI dispõe de uma forma simplificada de integração de conjuntos de dados históricos através de uma interface com a possibilidade de manipular conjuntos de dados com milhões de tuplas e centenas de atributos.

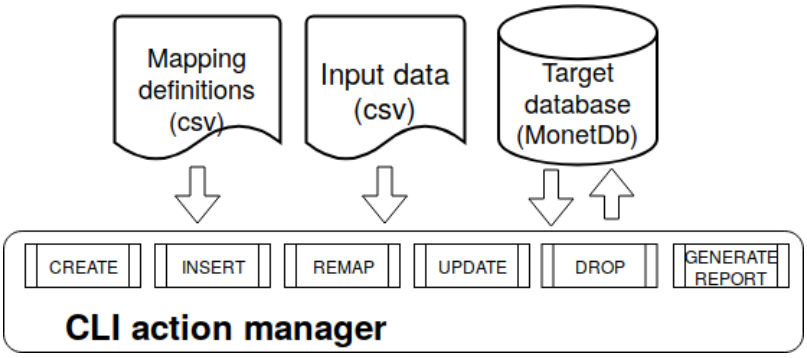


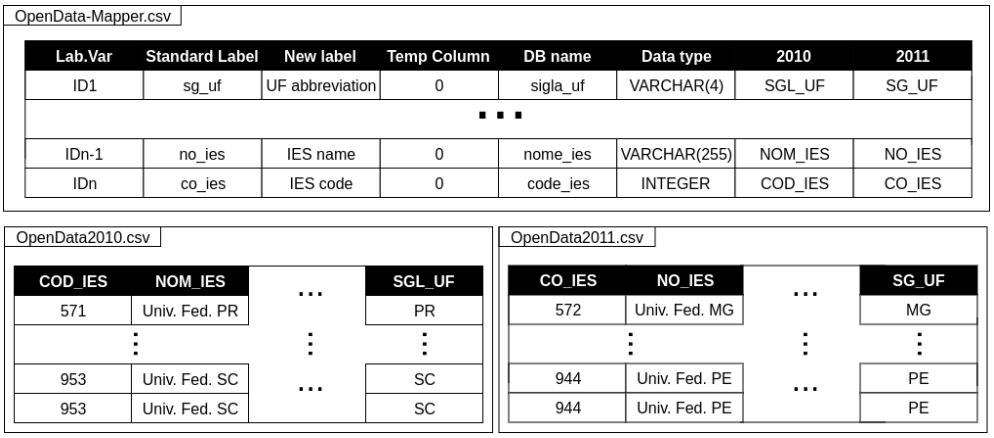

Figura 10 – Arquitetura geral do *HOTMapper*.

Fonte: (EHRENFRIED et al., 2019).

Além de arquivos de conjuntos de dados de entrada, a interface de linha de comando também recebe protocolos de mapeamento. Os protocolos de mapeamento são arquivos no formato CSV padronizados para serem interpretados pela ferramenta para aplicar as transformações de dados definidas.

O formato da definição de protocolos de mapeamento utilizada pelo CLI para executar integrações é demonstrada na Figura 11. O mapeamento contido no arquivo *OpenData-Mapper.csv* define uma relação com atributos padronizados: *Lab.Var*, um identificador de uma tupla do mapeamento; *Standard Label*, rótulo padrão de um atributo descrito na tupla; *New Label*, uma descrição do atributo; *Temp Column*, define se o atributo é temporário na aplicação

do mapeamento; *DB name*, nome do atributo na relação do banco de dados; *Data type*, tipo de dados que o atributo no banco de dados receberá. Além disso, para cada conjunto de dados históricos é definida uma coluna, como *2010* e *2011*. Essas colunas podem ser definidas de duas formas: uma equivalência de atributos do mapeamento, onde o atributo a ser manipulado corresponde ao atributo original do mapeamento, o que significa que o dado não será modificado no processo de integração de dados, como por exemplo o atributo *SG_UF* na coluna de *2011*; uma transformação dos dados a partir da definição de afirmações em SQL (Structured Query Language, ou Linguagem de Consulta Estruturada), em que será aplicada uma transformação nos dados no processo de integração.



OpenData-Mapper.csv

| Lab.Var | Standard Label | New label | Temp Column | DB name | Data type | 2010 | 2011 |
|---|---|---|---|---|---|---|---|
| ID1 | sg_uf | UF abbreviation | 0 | sigla_uf | VARCHAR(4) | SGL_UF | SG_UF |
| ▪ ▪ ▪ | | | | | | | |
| IDn-1 | no_ies | IES name | 0 | nome_ies | VARCHAR(255) | NOM_IES | NO_IES |
| IDn | co_ies | IES code | 0 | code_ies | INTEGER | COD_IES | CO_IES |



Figura 11 – Exemplo de um protocolo de mapeamento e os dados correlacionados.

Fonte: (EHRENFRIED et al., 2019).

A utilização e manipulação dos protocolos de mapeamento e do CLI serão discutidos em mais detalhes no capítulo de apresentação do sistema desenvolvido nesse trabalho.

## 3.5 Automatch

Existem diversos métodos de integração de dados utilizando mapeamento de esquemas, porém, muitos são aplicados de formas manuais. A implementação de técnicas com ML (Machine Learning, ou Aprendizado de Máquinas) permitem a produção de mapeamento automatizados que simplificam e agilizam a integração de dados.

Uma das abordagens em mapeamentos automatizados é apresentada em (BERLIN; MOTRO, 2002). Os autores desenvolvem um sistema chamado *Automatch*, baseado no método Bayesiano de aprendizado de máquina. Esse sistema utiliza um dicionário de atributos mapeados a partir de esquemas para gerar informações probabilísticas. As informações probabilísticas são geradas na comparação de cada atributo de texto de um esquema com todos os atributos de outro esquema, gerando assim uma pontuação da probabilidade de correspondência entre atributos. A pontuação é gerada através da Fórmula 1, baseada no Teorema de Bayes. Já com os atributos numéricos é aplicado um função de densidade normal, assumindo que existe uma

distribuição normal dos dados. A aplicação dessa função também irá gerar uma pontuação, para então somar-se as outras pontuações gerando a pontuação geral dos esquemas.

$$M(X, A) = \frac{P(A)}{P(V)} . \prod_{k=1}^{n} P(v_k|A) \quad (1)$$

As probabilidades da fórmula são descritas como:

- *M(X, A)*: probabilidade posterior de *X* mapear para *A*.
- *X*: atributo do esquema a ser mapeado.
- *A*: atributo do dicionário de atributos.
- *V*: conjunto de atributos observados em *X*.
- *P(A)*: probabilidade precedente de *X* mapear para *A*.
- *P(V)*: probabilidade de observar valores *V* em *X*.
- *P($v_k$)*: probabilidade de observar um valor *v* de *V* em *X*, tal que *X* é mapeável para *A*.

A Figura 12 demonstra um grafo onde os nós correspondem aos atributos, as arestas são as correspondências de atributos e os pesos nas arestas são as probabilidades. Os conjuntos $R_1$ e $R_2$ são os esquemas a serem mapeados. Os nós *($B_1, B_2$)* e *($C_1, C_2$)* são os atributos dos esquemas fonte e *($A_1, A_2, A_3$)* os atributos do dicionário. Os pesos *($w_1 - w_{12}$)* são as pontuações da probabilidades de mapeamento entre atributos a partir do dicionário. Um exemplo do mapeamento de atributos seria entre $B_1$ e $C_2$ por $A_1$ e entre $B_2$ e $C_1$ por $A_2$, com a pontuação geral de $w_1 + w_8 + w_4 + w_9$.

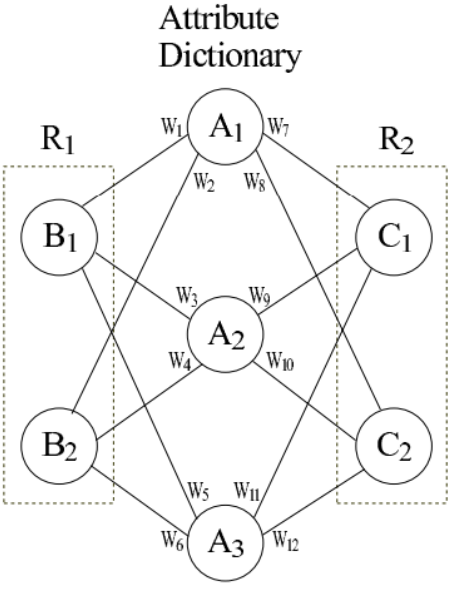



Figura 12 – Grafo exemplo representando a pontuação entre atributos.

Fonte: (BERLIN; MOTRO, 2002).

Uma abordagem baseada nesse sistema é discutida em (OLINI, 2019). A implementação, que será mais detalhada no capítulo de abordagem, apresenta uma forma mais simplificada do *Automatch* que será utilizada como base nesse trabalho para a criação de mapeamentos automatizados.

## 3.6 Considerações

Visualizando a Tabela 1 de comparações das abordagens apresentadas, podemos analisar as diferenças e semelhanças dos métodos de integração de dados. Existem abordagens que utilizam interfaces gráficas de integração de dados como o *++Spicy* e o *Metamorfose*, outras empregam um formato mais simples como uma interface por linhas de comando como o *HOTMapper* e algumas não apresentam interfaces mas disponibilizam os métodos de mapeamento como o *Clio* e o *Automatch*. A aplicação de interfaces é muito importante para permitir a integrações de dados de maneiras mais simplificadas e alcançar uma maior quantidade de usuários promovendo mais acessibilidade. Outro ponto é a implementação de técnicas automatizadas de mapeamento como definidas na abordagem do *Automatch*. Grande parte das interfaces tem a geração de mapeamentos básicos para permitir que o usuário aplique suas próprias transformações e definições como no *Metamorfose* e o *++Spicy*, porém, nenhuma das interfaces analisadas utiliza uma técnica de geração de mapeamentos automatizados. Outra característica interessante que será tratada por esse trabalho e é analisada pelas ferramentas *Metamorfose* e *HOTMapper* é a aplicação em dados abertos brasileiros. Poucos trabalhos com aplicações em interfaces fazem análises com dados abertos brasileiros na área de integração de dados. Finalmente, ferramentas como o *++Spicy* e o *HOTMapper* disponibilizam seus códigos promovendo uma colaboração de terceiros no desenvolvimento e manutenção do projetos, o que é uma característica muito interessante quando se trata da manipulação de dados abertos.

Visto as diferenças e semelhanças, o próximo capítulo propõe uma abordagem de um sistema de integração de dados utilizando um conjunto de características baseadas nas ferramentas apresentadas, como na Tabela 1, com o objetivo de simplificar a integração de dados e promover mais eficiência na utilização de ferramentas de integração, fazendo análises no cenário brasileiro de dados abertos.

| Ferramenta | Interface | Mapeamento Automatizado | Analise em Dados Abertos Brasileiros | Código Aberto |
|---|---|---|---|---|
| Clio | Não disponível | Não | Não | Não |
| ++Spicy | Gráfica | Não | Não | Sim |
| Metamorfose | Gráfica | Não | Sim | Não |
| Automatch | Não disponível | Sim | Não | Não |
| HOTMapper | Linha de comando | Não | Sim | Sim |

Tabela 1 – Tabela de comparação de ferramentas de integração de dados.

# 4 SISTEMA DE INTEGRAÇÃO DE DADOS COM INTERFACE GRÁFICA E MAPEAMENTOS AUTOMATIZADOS

A produção e utilização de dados abertos em projetos de integração de dados são de grande importância no cenário mundial, principalmente com a aplicação de dados abertos governamentais. A abordagem apresentada nesse trabalho é focada na integração de dados promovida pelo C3SL (Centro de Computação Científica e Software Livre) (DIRENE. et al., 2016), utilizando dados anuais de Microdados do INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira) (INEP, 2021) por meio da ferramenta *HOTMapper* (EHRENFRIED et al., 2019). A arquitetura apresentada a seguir apresenta uma interface gráfica que atua sobre o *HOTMapper*, contendo as funcionalidades de ETL, e a aplicação de um método de mapeamento automatizado baseado no *Automatch*, com o objetivo de simplificar a integração de dados e dar mais acessibilidade às ferramentas.

## 4.1 Arquitetura

A arquitetura do sistema apresentado na Figura 13 possui suas telas divididas em duas partes, onde a primeira é composta por *documentação*, *configurações* e *funções* que disponibilizam as funcionalidades do *HOTMapper*; já a segunda por *mapeamento* que implementa funcionalidades básicas de mapeamento como também o mapeamento automatizado. A tela de *preferências* apresenta as configurações da interface.

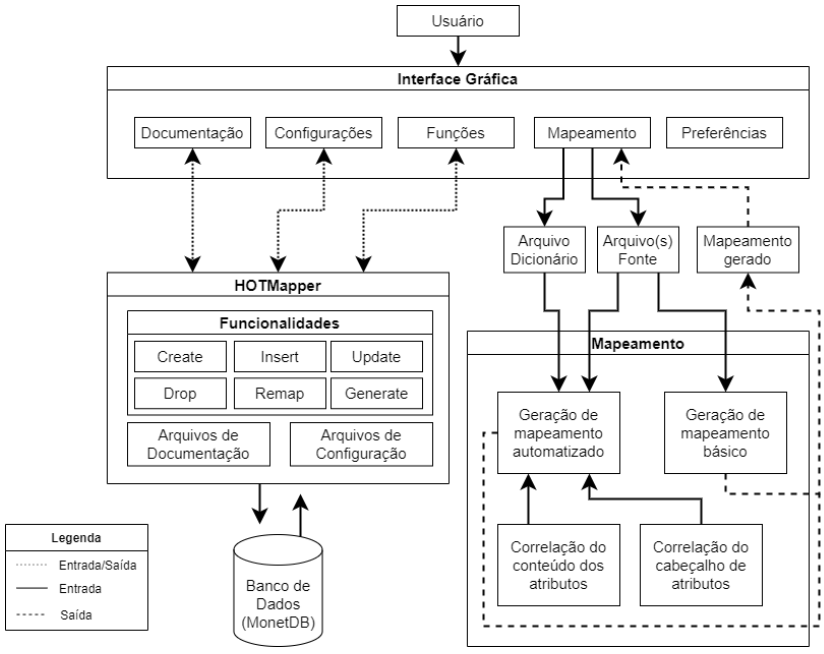



Figura 13 – Arquitetura de interface da abordagem.

Fonte: autoria própria, 2021.

A partir do fluxo é possível visualizar que a interface gráfica faz o intermédio entre o usuário e a ferramenta de integração de dados, como também com a funcionalidade que implementa os métodos de mapeamento. De maneira geral, a tela *Documentação* apresenta um arquivo de documentação da ferramenta de integração de dados. A tela *Configurações* define um menu de arquivos e um visualizador para configurar a ferramenta de integração. As funcionalidade da ferramenta de integração de dados são apresentadas na tela *Funções*, através de uma lista, como também funcionalidades da própria interface gráfica. Já a tela *Mapeamento* divide suas funcionalidades em botões como a criação de um mapeamento básico, carregar um mapeamento para edição e a geração de mapeamentos automatizados. Por fim, a tela *Preferências* server uma uma edição da própria interface gráfica para alterar o visual conforma a vontade do usuário. Essa interface será abordada em mais detalhes a seguir.

## 4.2 Implementação

A interface gráfica aqui apresentada é implementada usando os frameworks ReactJS e Electron, que utilizam a linguagem de programação Javascript, e são utilizados para a parte visual e para a integração com o sistema operacional, respectivamente.

A implementação define um arquivo no formato JSON (*Javascript Object Notation*) de configuração de ferramentas para produzir as telas e informações na interface gráfica. Um exemplo do arquivo de configuração é apresentado no Código 4.1. O arquivo de configuração define um conjunto de objetos que representam as ferramentas, onde a chave é o identificador da ferramenta, e valor são os dados para a integração com a interface gráfica. Os objetos que representam as ferramentas possuem as seguintes chaves:

- *name*: define o nome da ferramenta.
- *folder*: define o nome da pasta da ferramenta para o acesso do sistema.
- *download*: define o link do repositório da ferramenta para o sistema carregá-lo.
- *links*: conjunto de vetores que definem links para os repositórios da ferramenta.
- *documentation*: define o caminho de acesso ao arquivo de documentação da ferramenta.
- *settings*: define um conjunto de pastas de arquivos de configuração da ferramenta.
- *cli*: define o caminho de acesso ao arquivo de interface da ferramenta.
- *execute*: define o comando de execução da ferramenta.

```
{
    "hotmapper": {
        "name": "HOTMapper",
        "folder": "hotmapper",
        "download": "link-repositorio-ferramenta",
        "links": [
            ["gitlab", "link-repositorio-gitlab"],
            ["github", "link-repositorio-github"]
        ],
        "documentation": "README.md",
        "settings": ["table-definitions"],
        "cli": "manage.py",
        "execute": "./manage.py"
    }
}
```

Código 4.1 – Exemplo de arquivo de definição da interface gráfica.

Com as informações definidas nesse arquivo de configurações de ferramentas, a interface gráfica gera as telas e aplica as funcionalidades da ferramenta de integração de dados carregada. Dessa forma, utilizando a definição do *HOTMapper* como exemplo, a tela Documentação que será detalhada em seguida carrega o arquivo "README.md" para disponibilizar ao usuário a documentação da ferramenta da integração de dados. Com isso, todo o sistema depende da definição correta de uma ferramenta no JSON apresentado.

### 4.2.1 Documentação

A interface disponibiliza um acesso rápido para a documentação da ferramenta através da tela *Home* como apresentado na Figura 14. O arquivo de documentação da ferramenta está presente nas pastas de arquivos do *HOTMapper* e pode ser encontrado em seus repositórios. Contido na tela inicial está o nome da ferramenta carregada através de um título, além do arquivo de documentação da ferramenta que é carregado pela interface, definido no arquivo de configuração de ferramentas, em um visualizador de arquivos *markdown* - linguagem simples de marcação de texto para formatar arquivos. Esse arquivo é apresentado conforme é estruturado pela ferramenta e não pode ser alterado pela interface gráfica.

Além do arquivo de documentação, a tela inicial da interface também dispõe o acesso rápido aos repositórios da ferramenta carregada através dos ícones inferiores da tela, como as plataformas *Gitlab* e *Github* do *HOTMapper*. Esses repositórios são definidos no arquivo de configurações como visto anteriormente.

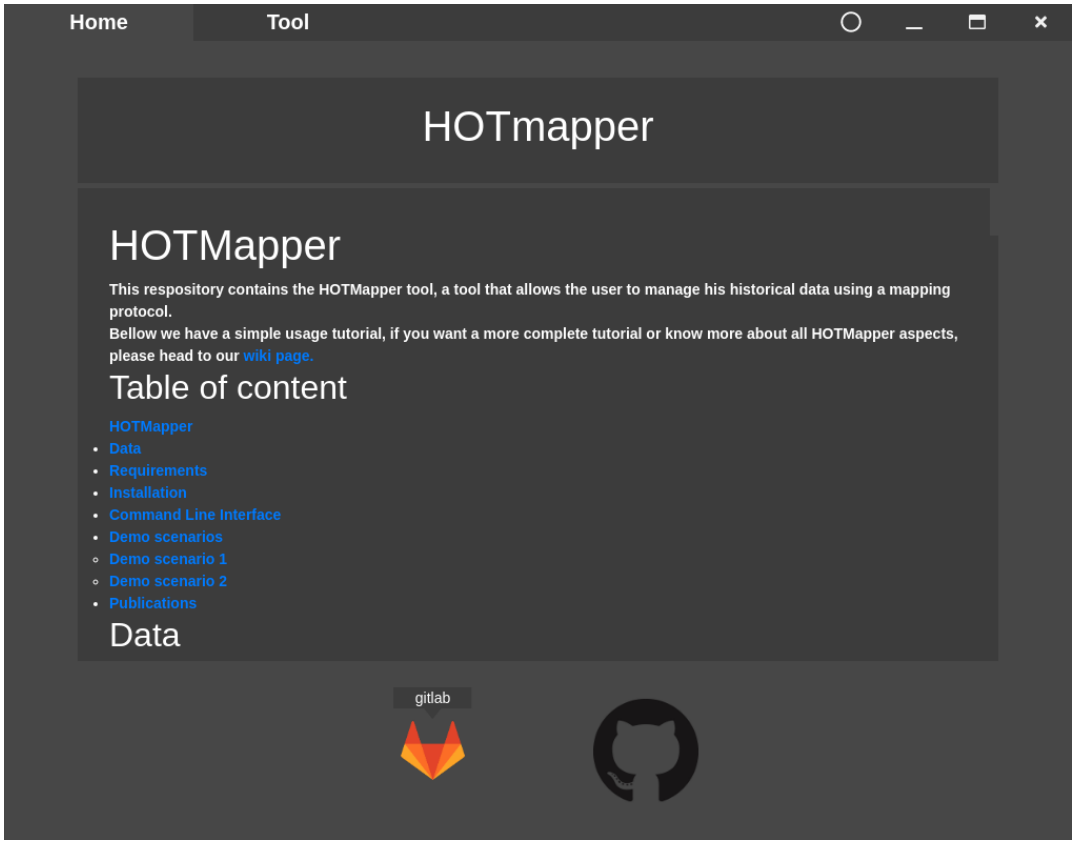



Figura 14 – Tela de documentação da interface gráfica.

### 4.2.2 Configurações

Uma funcionalidade que é integrada na interface é a disponibilização de um acesso rápido aos arquivos de configuração da ferramenta e, para isso, a tela *configuration* na seção *Tool* implementa um visualizador de arquivos *JSON*.

Os arquivos que são carregados pela interface através de um sistema que faz a leitura da pasta definida no Código 4.1, estão contidos nas pastas definidas no arquivo de configuração de ferramentas da interface gráfica. Um exemplo de arquivo de configuração é o que define tabelas que serão utilizadas pelo *HOTMapper*, como o Código 4.2 que exibe um arquivo *JSON* da tabela "Massa de Agua". Essas definições de tabelas são compostas pelas seguintes chaves:

- *pairing_description*: recebe como valor a descrição da tabela.
- *data_source*: recebe como valor a fonte dos dados.
- *pk*: recebe um conjunto dos atributos que definem as chaves primárias da tabela.
- *foreing_keys*: recebe como valor um conjunto de objetos de atributos que definem as chaves estrangeiras da tabela.
- *columns*: recebe como valor conjuntos de colunas da tabela e suas definições.

```
{
    "pairing_description" = "Massa de Agua",
    "data_source" = "Regiao Hidrografica Parana em dados.gov
        .br",
    "pk" = ["id"],
    "foreign_keys" = [{
        "keys": ["id"],
        "reference_columns": ["id"],
        "reference_table": "agua_parana"
    }],
    "columns" = {
        "id": ["INT", "OBJID"],
        "ano_censo": ["INT", "ANO"],
        "nome": ["VARCHAR(64)", "NOMEOG"]
    }
}
```

Código 4.2 – Exemplo de arquivo de definição de tabela.

A Figura 15 demonstra a tela de configurações da ferramenta. O lado esquerdo da interface apresenta todos os arquivos carregados através de um menu de seleção, como também disponibiliza o botão *delete* para remover um arquivo e o botão *new* para criar um novo arquivo. Quando um arquivo é selecionado no menu esquerdo ele é carregado no visualizador de arquivos *JSON* na parte direita da interface. É possível fazer edições nos arquivos a partir do visualizador, podendo salvar através do botão *save* ou restaurar o estado inicial do arquivo no botão *restore*.

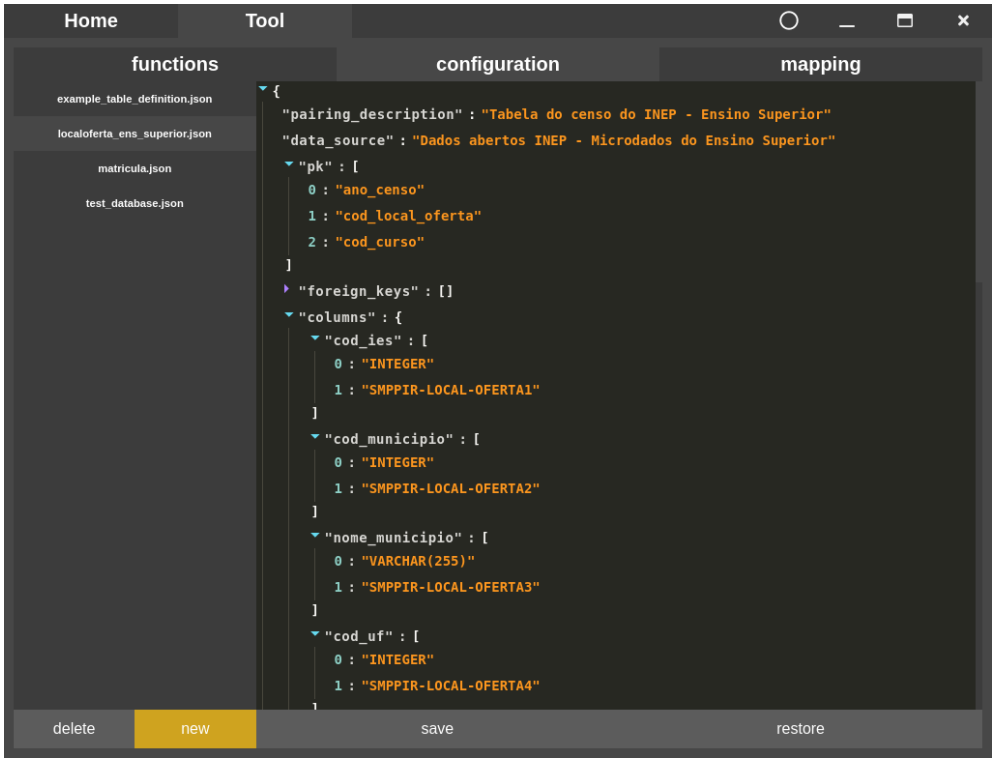



Figura 15 – Tela de configurações da interface gráfica.

### 4.2.3 Funções

As funcionalidades implementadas pelo *HOTMapper* são definidas em um arquivo específico que funciona como a interface por linha de comandos da ferramenta. Esse arquivo, que é definido no arquivo de configurações das ferramentas da interface, é interpretado pelo sistema da interface gráfica para gerar formulários de entrada na tela *functions* na seção *Tool* como visto na Figura 16.

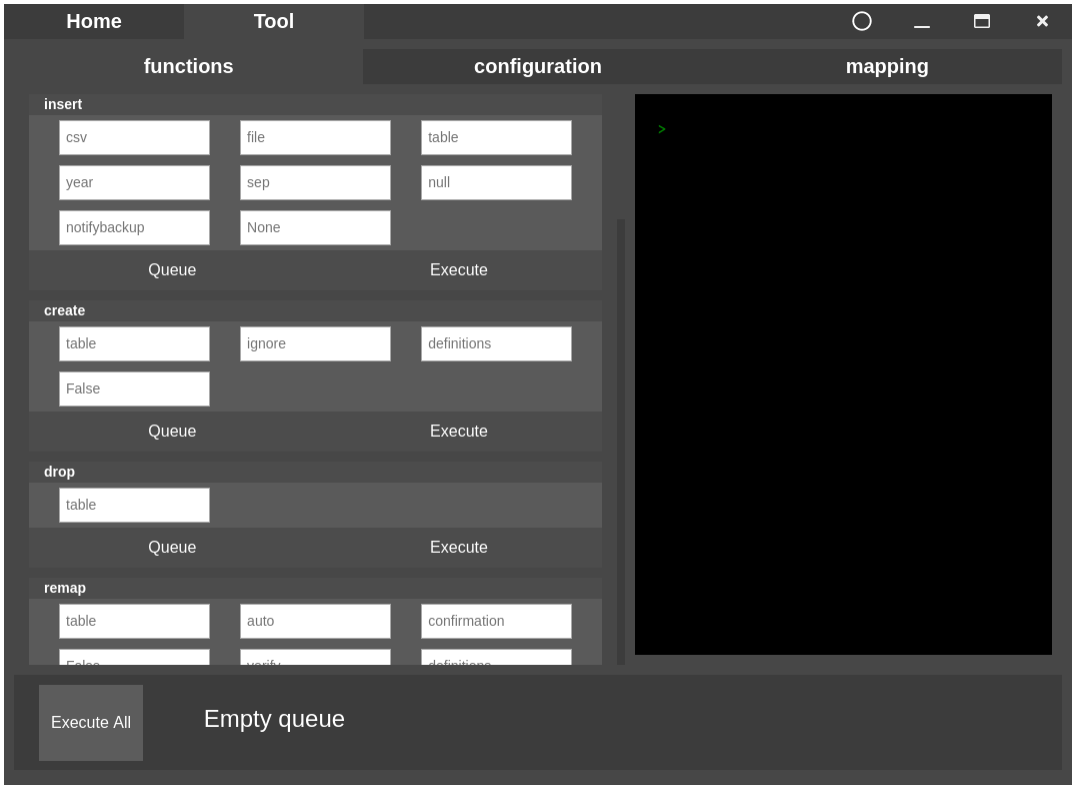



Figura 16 – Tela de funções da interface gráfica.

Visualizando o lado esquerdo da interface, os formulários que disponibilizam as funções do *HOTMapper*, apresentados na seção 3.4, são implementados em caixas divididos em:

- título na parte superior: utiliza o nome da função carregada pelo formulário (insert, create, drop, remap, etc).
- caixas de entrada: definem os parâmetros que as funções recebem (como *csv, file, table, year, sep* que diferem para cada função).
- botão *execute*: executa o comando específico da função com as entradas passadas pelo usuário.
- botão *queue*: adiciona a função e suas entradas em uma fila de execução que pode ser visualizada na parte inferior da tela.

Como cada formulário é uma função, a quantidade de caixas de entrada equivale a quantidade de parâmetros da funcionalidade do *HOTMapper* representada na interface

gráfica. Nem todas as caixas de entrada precisam ser preenchidas para executar uma função com sucesso, o que pode mudar para cada funcionalidade da ferramenta em questão. Dessa forma, essas funções podem ser executadas através do preenchimento dos campos conforme a documentação da ferramenta, gerando um comando de execução quando a função é adicionada à fila de execução através do botão *queue* ou da execução imediata através do botão *execute*.

A parte inferior da tela *functions* implementa a funcionalidade de fila de execução de funções. Essa parte apresenta uma caixa com o botão *Execute All* para executar todas as funções contidas na fila de execução, como também uma caixa da própria fila. Quando o botão *Execute All* é acionado, todas as funções presentes na fila de execução são executados sequencialmente até a fila esvaziar. A fila é apresentada em uma caixa com todas as funções contidas nela, como visto na Figura 17, formatadas como:

- título na parte superior: é o nome da função armazenada no segmento da fila.
- parâmetros: são os parâmetros que serão utilizados para executar a função definida.
- botão *remove*: remove a função da fila de execução.

A fila de execução armazena todos as funções e parâmetros adicionados através do botão *queue* nas caixas de funções. A execução da fila é feita da mesma forma que a execução imediata de uma função, através da geração de um comando de execução, porém, é gerado para cada função.

A saída da execução das funções do *HOTMapper* chamadas pelo sistema são apresentadas em um visualizador na parte direita da interface gráfica. Esse visualizador apresenta as saídas das execuções em formato de texto, como também os registros das execuções. Os registro são padronizados por data, horário, função e parâmetros da forma "===> <data> - <horário>: Command execution <cli><função><parâmetros>".

A execução de uma função pelo sistema pode ser exemplificada utilizando a função *insert* e seus parâmetros apresentados na Figura 17. A função e os parâmetros são concatenados com espaços em uma *string*, adicionando a chamada de execução do *HOTMapper* resultando na *string* "*./manage.py insert /tera/c3sl/archives/2014_DM_LOCAL_OFERTA_YEAR.CSV localoferta_ens_superior 2014 –sep=\|*". Antes da execução da função, é registrada a ação no visualizador lateral da interface gráfica, adicionando a data, horário, e a *string* de execução ao registro como visto na primeira linha do visualizador. Por fim, a *string* gerada é então executada pela ferramenta, e a saída de execução é disponibilizada também no visualizador lateral da GUI. Da mesma maneira, essa função poderia ser adicionada à fila de execução, junto a outras funções, para executar uma sequência de funções as quais teriam seus registros e saídas apresentadas no visualizador.

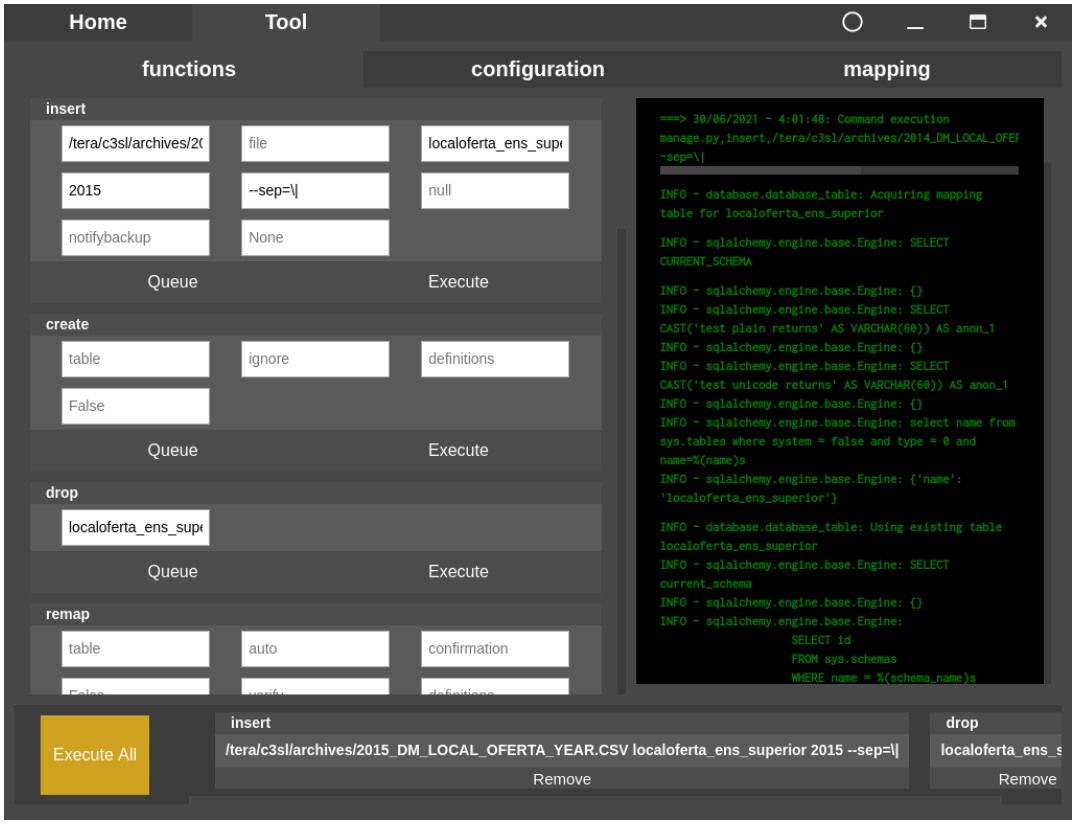



Figura 17 – Exemplo de execução na tela de funções da interface gráfica.

### 4.2.4 Preferências

A tela de preferências é acessada através do botão com ícone circular no canto superior direito da interface gráfica. O objetivo da tela de preferências é permitir o usuário aplicar alterações na interface gráfica. A figura 18 apresenta essa tela utilizando as definições padrões de tema e fonte. Essas alterações são divididas em:

- fontes: tamanho das fontes dos textos da interface podendo ser nos formatos: pequeno, médio e grande.
- temas: as cores de todos os componentes da interface que definem temas do sistema, como um tema contendo cores escuras e um tema contendo cores claras.

Além das definições de preferências da interface gráfica, a tela *preferences* também apresenta um ícone na parte inferior central que permite o acesso rápido ao repositório do sistema em questão.

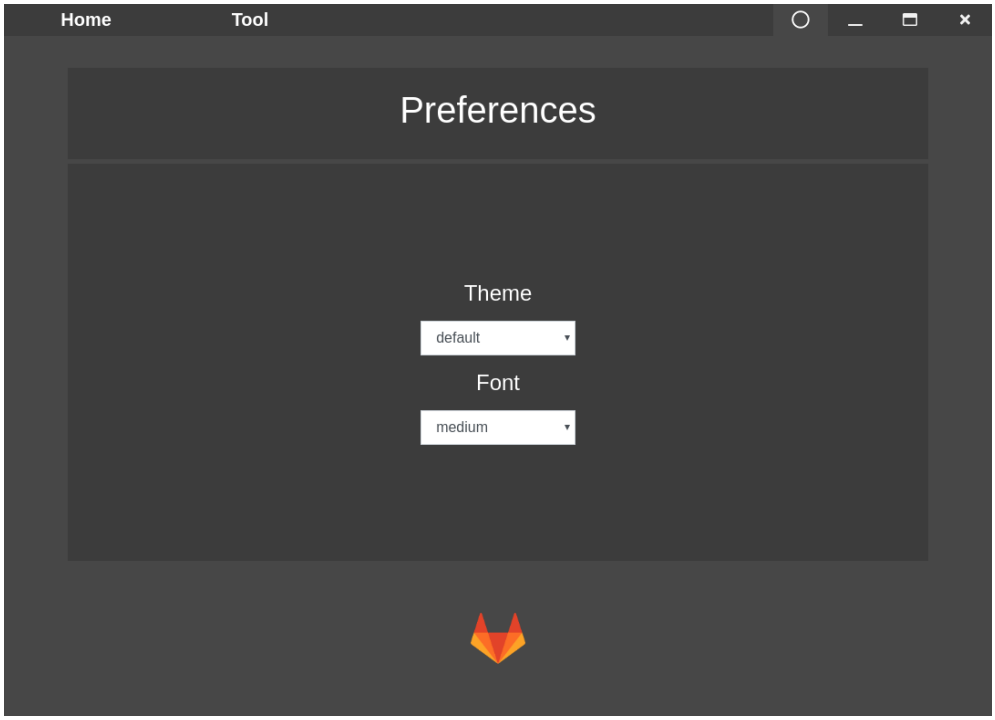



Figura 18 – Tela de preferências da interface gráfica.

A Figura 19 apresenta a tela *Home* com um exemplo de um tema claro para a interface gráfica, com a fonte de tamanho pequeno.

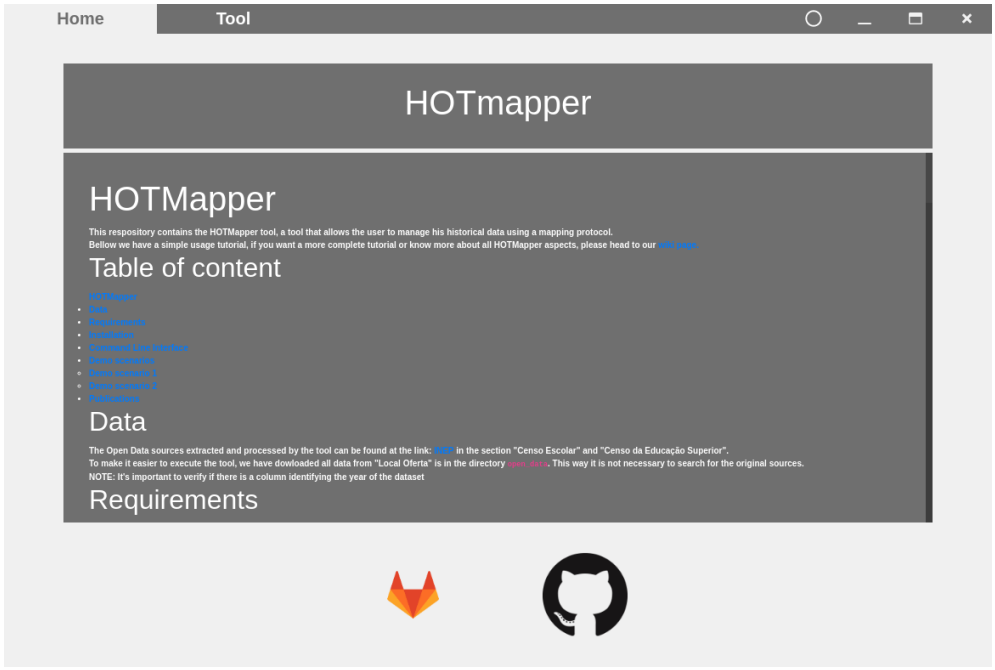



Figura 19 – Tela inicial com tema claro.

### 4.2.5 Mapeamento

O acesso rápido a protocolos de mapeamento e geração de novos mapeamentos são objetivos desse trabalho na integração de dados e, para isso, a GUI implementa a sub-tela *mapping* na tela *Tool*. Essa tela pode ser visualizada na Figura 20 onde é dividida em três partes: uma seção de ações que contém os botões *New Mapping* (produção de mapeamentos básicos), *Load Mapping* (carrega arquivos de mapeamento para edição) e *Auto Mapping* (produção de mapeamentos automatizados); uma seção para disponibilizar os dados processsados ou protocolos de mapeamento; uma seção para visualizar os dados brutos carregados. Quando não há dados carregados pelo sistema, as seções de visualização de dados apresentam somente uma mensagem de aviso.

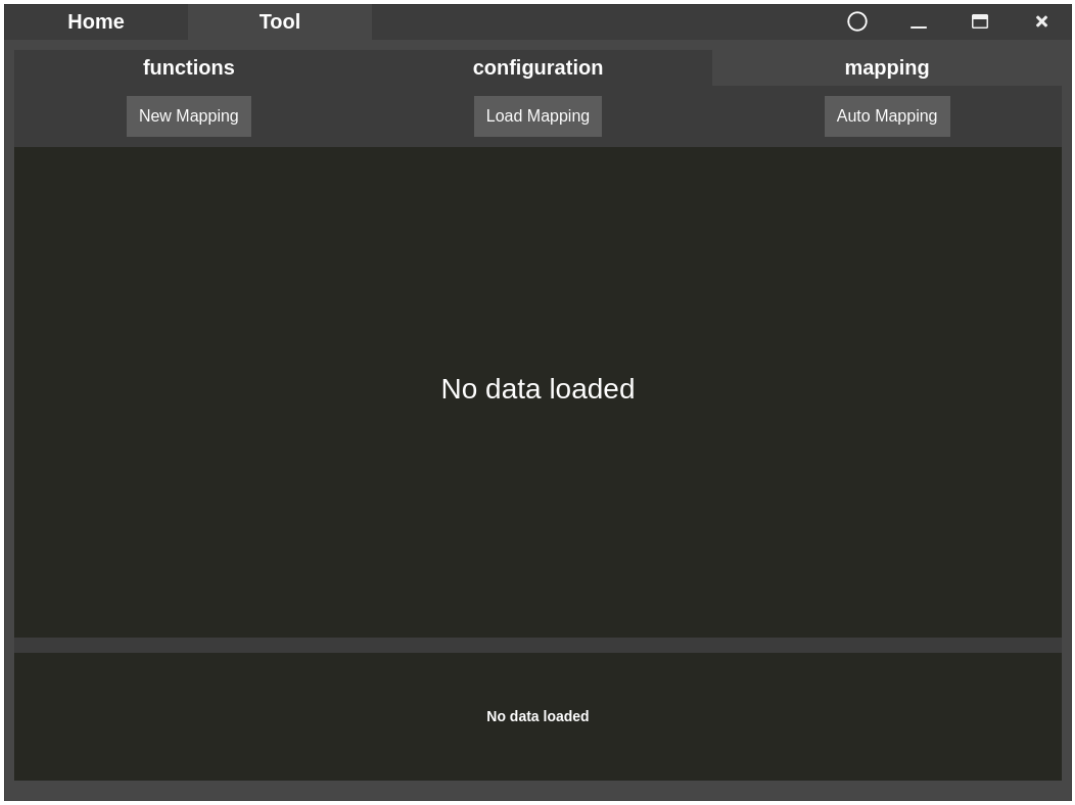



Figura 20 – Tela de mapeamento da interface gráfica.

Antes de apresentar o funcionamento de cada botão da tela de mapeamento é preciso esclarecer a criação e aplicação dos protocolos de mapeamento da ferramenta *HOTMapper*.

Os protocolos de mapeamento são tabelas essenciais para que a ferramenta aplique as transformações nos dados carregados e, como visto no capítulo 3, os protocolos possuem seis colunas padrões que definem, respectivamente: identificador da linha da tabela; rótulo padrão do atributo dos dados brutos; descrição do atributo; valor que define se o atributo é, ou não, temporário [1] na aplicação do mapeamento; rótulo que o atributo recebe no banco de dados;

[1] atributo que é criado no inicio de uma função do *HOTMapper* e descartado ao final

tipo de dados do atributo definido no BD. Esses protocolos de mapeamento devem definir a mesma tabela apresentada em um arquivo de configurações de tabela, como visto na tela de *configuration* apresentado na Figura 15, para que a ferramenta aplique a transformação de dados.

Como a ferramenta *HOTMapper* faz a integração de dados históricos, os protocolos de mapeamento também definem as colunas de ano para transformação de dados. Um exemplo de um protocolo carregado pela interface gráfica através do botão *Load Mapping* pode ser visto na Figura 21. O arquivo apresenta as colunas padrões e as colunas ano de 2010 a 2016 do mapeamento de dados da tabela "Local de Oferta do Ensino Superior". Essas colunas recebem os rótulos dos dados brutos do ano que representam, como o rótulo *CO_CURSO*, ou um código SQL de transformação de dados, como "*CASE WHEN ("CO_CURSO") = 1 THEN "Informatica" END*". Os protocolos podem definir atributos que não existem em certos períodos, como o atributo "NO_LOCAL_OFERTA" que é inexistente nos anos de 2010 a 2012.

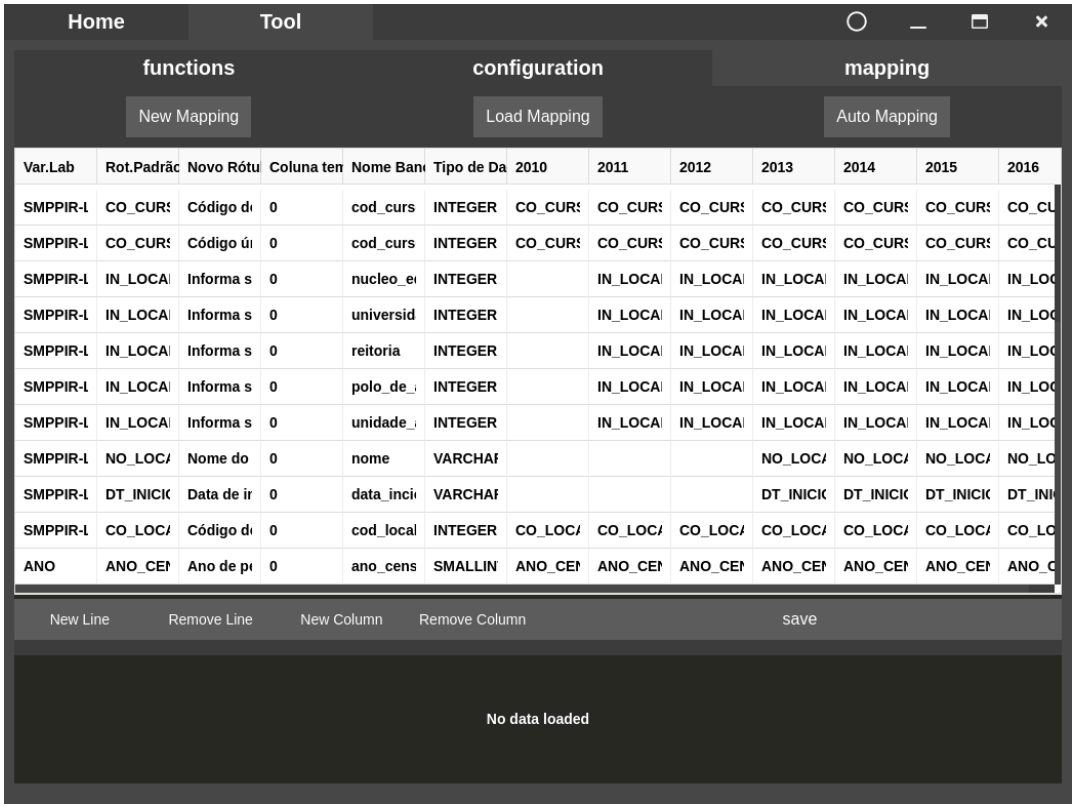



Figura 21 – Exemplo do carregamento de um protocolo de mapeamento na interface gráfica.

Sabendo da padronização de um protocolo de mapeamento, a interface gráfica permite a geração de novos arquivos de mapeamento através do botão *New Mapping* que produz uma nova tela baseada na ferramenta *Metamorfose* (KUSZERA; PERES; FABRO, 2018).

O processo de geração de um protocolo de mapeamento básico requer a passagem de um arquivo de dados pelo usuário. Como os mapeamentos são feitos sobre conjuntos de dados históricos, é necessário que o usuário dê entrada do ano que define o arquivo de dados, como

é visto na Figura 22. Com essas informações, o sistema gera um protocolo de mapeamento padronizado para a utilização do *HOTMapper*, com as seis colunas obrigatórias e a coluna que define os atributos do ano especificado. As colunas "rótulo padrão", "coluna temporária", "tipo de dado" e o ano do arquivo carregado são preenchidas automaticamente pelo sistema, respectivamente, da forma: rótulo de cada atributo do conjunto de dados em cada linha; o caractere "0" para todas as linhas; a *string* "VARCHAR(255)" para definir os tipos de dados de cada linha; rótulo padrão dos atributos. O arquivo gerado é então disponibilizado em uma tabela na interface gráfica para a edição pelo usuário.

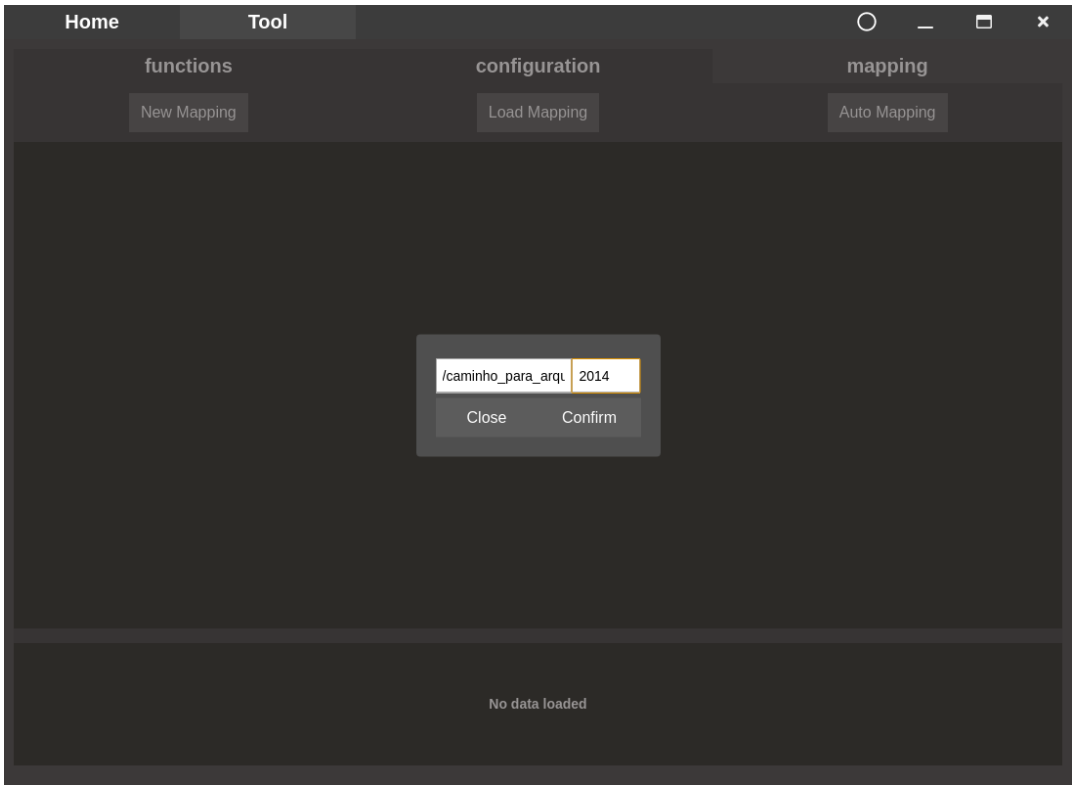



Figura 22 – Exemplo de entrada na tela de mapeamento.

A Figura 23 demonstra uma tabela sendo preenchida na interface, gerada através do conjunto de dados de "Local de Oferta do Ensino Superior" do ano de 2014. Além da tela apresentar a tabela do mapeamento, também exibe uma tabela com duas linhas dos dados carregados, na parte inferior, para permitir o usuário analisar rapidamente os dados de cada atributo.

A geração de mapeamentos básicos é um facilitador na criação de mapeamentos, já que o usuário não necessita criar todo o documento manualmente, porém, essa funcionalidade aplica-se somente a criação inicial dos mapeamentos. Para a adição de mais períodos (e.g. colunas de ano) na tabela, é disponibilizado a função de mapeamento automatizado.

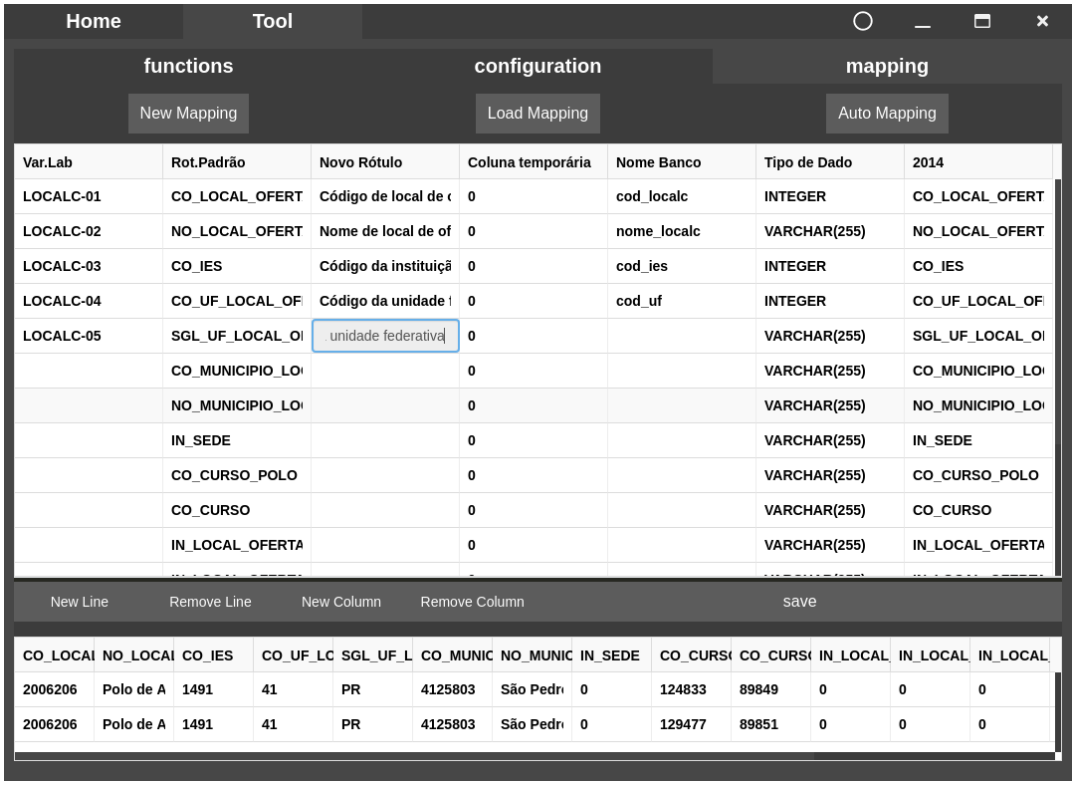



Figura 23 – Exemplo de um protocolo de mapeamento básico gerado na interface gráfica.

#### 4.2.5.1 Mapeamento Automatizado

Além da geração de protocolos de mapeamentos básicos, a interface gráfica disponibiliza o botão *Auto Mapping* que aplica um mapeamento automatizado utilizando o método apresentado em (OLINI, 2019) com algumas modificações. Esse método utiliza dois conjuntos de dados em formato CSV para a geração das colunas ano de um protocolo de mapeamento.

A técnica de mapeamento automatizado utiliza uma base dicionário (arquivo CSV de um conjunto de dados) para criar um dicionário de atributos e uma conjunto de dados cliente para encontrar correlações de atributos. Essa técnica é implementada pelo Algoritmo 1 que recebe o conjuntos de dados como entrada e retorna um dicionário de atributos. Inicialmente, para atributos do tipo texto ou numérico discreto (valores numéricos inteiros, finitos), o algoritmo faz a limpeza dos dados carregados removendo acentos e alterando os caracteres para caixa baixa, permitindo assim a contagem da presença de cada valor por coluna armazenando os valores e contagens na base dicionário. Já para cada atributo do tipo numérico contínuo (valores numéricos flutuantes, fracionários) os valores são basicamente armazenados e separados pelos atributos de aparição. Os atributos booleanos (valores verdadeiros ou falsos, representados por 0 e 1) são considerados como atributos texto e lidos da mesma maneira.

**Algoritmo 1:** Construção do dicionário de atributos retirado de (OLINI, 2019)

```
Function constroiDicionarioAtributos(baseDic):
    abre arquivo da base dicionario baseDic
    valoresCols = [ ]
    colsDicAtr = [ ]
    for nomeColuna in cabecalho(baseDic) do
        colunas da base dicionário
        insere nomeColuna em colsDicAtr[]
        for linha in linhas(baseDic) do
            if i in indices_colunas_tipo_texto_num_disc then
                valor = sanitizacao(linha[i])
                if valor in valoresCols[i] then
                    valoresCols[i][valor]+= 1
                else
                    valoresCols[i][valor] = 1
                end
            else
                insere linha[i] em valoresCols[i]
            end
        end
    end
    dicAtr = {}
    totalOcorrDicAtr = [ ]
    for i in tamanho(colunas) do
        if i in indices_colunas_tipo_texto_num_disc then
            ordena decrescentemente valoresCols[i] pelo número de ocorrências
            valoresAtrDic, numTotalOcorr = getTop80(valoresCols[i])
            dicAtr[colsDicAtr[i]] = valoresAtrDic
            insere numTotalOcorr em totalOcorrDicAtr[i]
        else
            media = media(valoresCols[i])
            desvioPadrao = desvioPadrao(valoresCols[i])
            dicAtr[colsDicAtr[i]][media] = media
            dicAtr[colsDicAtr[i]][desvioPadrao] = desvioPadrao
        end
    end
    return dicAtr, totalOcorrDicAtr
```

Após a leitura e limpeza dos dados do tipo texto e tipo numérico discreto, os valores são ordenados crescentemente e são mantidos somente os que compõe 80% do valor total de aparições, que são os valores mais relevantes, diminuindo a quantidade de valores no dicionário. Esse processo de seleção de valores é implementado pelo Algoritmo 2 e é utilizado no processo de criação do dicionário de atributos. Já para atributos do tipo numérico contínuo é feito a média e desvio padrão dos valores, armazenando-os no dicionário de atributos.

**Algoritmo 2:** Processo de seleção dos valores mais relevantes retirado de (OLINI, 2019)

```
Function 80dic:
    numTotalOcorr = soma(valores(dicionario))
    numTotalOcorrContabilizadas = 0.0
    dicAux =  while chave in chaves(dic) do
        dicAux[chave] = dic[chave]
        numTotalOcorrContabilizadas+ = dic[chave]
        remove chave de dic
    end
    numTotalOcorrAtrDic = soma(valores(dicAux))
    return dicAux, numTotalOcorrAtrDic
```

Após a criação do dicionário de atributos, o conjunto de dados cliente é processado aplicando a limpeza e armazenagem dos valores distintos de cada atributo assim como feito na base dicionário.

Com o dicionário de atributos e a base cliente preparados, a correlação de atributos é iniciada para definir o mapeamento de dados. Esse processo é implementado pelo Algoritmo 3. Inicialmente é verificado se o atributo da base cliente é texto/numérico discreto ou numérico contínuo para decidir o processo de execução. Para os atributos numéricos contínuos é feito a busca na distribuição normal de valores de mesmo tipo no dicionário de atributos, que é obtida através da função densidade de probabilidade, e é somado a variável de resultados. Já para os atributos do tipo texto ou numérico discreto é feito a busca do valor no dicionário de atributos e, caso exista, é aplicado o somatório de probabilidades resultantes. A modificação do algoritmo de (OLINI, 2019), destacado no código entre "Parte Modificada", aplica o cálculo da distância de Levenshtein para ambos os tipos de atributos que, então, é somado ao resultado das probabilidades adicionando a média dos valores.

A geração da pontuação de atributos no formato texto e atributos do tipo numérico discreto é feita através da Fórmula 2. Essa fórmula é baseada na mesma apresentada em (BERLIN; MOTRO, 2002), alterando a multiplicação pela soma das probabilidades dos valores *v* estarem em *A*, assumindo que o mapeamento de *X* e *A* é válido.

$$M(X, A) = \frac{P(A)}{P(V)} . \sum_{k=1}^{n} P(v_k|A) \tag{2}$$

A equação para obter a probabilidade de observar os valores *V* em *X*, *P(V)*, é apresentada na Fórmula 3. Essa equação é obtida sabendo que $M(X, A) + M(X, \neg A) = 1$.

$$P(V) = (P(A) . \sum_{k=1}^{n} P(v_k|A) + P(\neg A) . \sum_{k=1}^{n} P(v_k|\neg A)) \tag{3}$$

Com a equação aplicada no mapeamento de dados, o conjunto *A* representa o dicionário de atributos e o conjunto *X* o conjunto de dados. Dessa forma, a equação possui as seguintes probabilidades:

- *M(X, A)*: probabilidade posterior de *X* mapear para *A*, sendo o mapeamento válido após observar valores de *X*.
- *P(A)*: probabilidade precedente de *X* mapear para *A*, sendo o mapeamento válido antes de observar qualquer valor de *X*. Essa probabilidade é estimada pela proporção de valores na base dicionário que estão presentes no dicionário de atributos.A proporção utilizada é de 80% por apresentar os valores mais representativos, portanto, *P(A)* sempre será 0.8.
- *P(*$\neg A$*)*: probabilidade precedente de *X* mapear para *A*, sendo o mapeamento válido antes de observar qualquer valor de *X*. Essa probabilidade é a negação de *P(A)* apresentando 20% dos valores mais representados, portanto, *P(*$\neg A$*)* será sempre 0.2.
- *P(*$v_k|A$*)*: probabilidade de um valor $v_k$ de *X* estar em *A*, assumindo que o mapeamento é válido. Para o cálculo das probabilidades é feita a racionalidade entre a quantidade de ocorrências de um valor para um atributo no dicionário de atributos sobre o total de ocorrências de valores para o atributo no dicionário. A probabilidade inicia-se em *0.0* e somam-se as probabilidades subsequentes, aumentando a probabilidade de correlação de atributos de *X* e *A* ser correta, onde o mapeamento é válido.
- *P(*$v_k|\neg A$*)*: probabilidade de um valor $v_k$ de *X* estar em *A*, assumindo que o mapeamento é inválido. O cálculo do racional de *P(*$v_k|A$*)* é subtraído de *P(*$v_k|\neg A$*)*. A probabilidade inicia-se em *1.0* e é subtraído das probabilidades subsequentes, representando a diminuição da probabilidade de correlação de atributos de *X* e *A* ser correta, onde o mapeamento é inválido.

Para o cálculo da pontuação de atributos no formato numérico contínuo utiliza-se a Fórmula 4. Essa fórmula é apresentada em (BERLIN; MOTRO, 2002) aplicando a função de densidade de probabilidade nos atributos, assumindo uma distribuição normal dos valores.

$$M(X, A) = \sum_{k=1}^{n} PDF(v_k|A) \tag{4}$$

Os elementos da equação de pontuação de atributos numéricos contínuos são definidos como:

- *M(X, A)*: probabilidade de *X* mapear para *A*.
- *PDF(*$v_k|A$*)*: probabilidade do valor $v_k$ estar contido na distribuição normal de valores do conjunto *A*.

Além dos cálculos de pontuação utilizando os dados dos atributos de um conjunto de dados, é adicionado uma métrica de pontuação para aprimorar a precisão do mapeamento de atributos manipulando os nomes dos atributos. Essa métrica é obtida através de um método

de *string matching* que gera uma pontuação na comparação entre dois textos.

A distância de Levenshtein, ou distância de edição, é uma métrica de aproximação de *string* definida pelo número mínimo de operações (remover, editar ou adicionar um caractere) para transformar uma *string* em outra *string*. Essa métrica é aplicada no mapeamento verificando cada atributo do dicionário de atributos com todos os atributos do conjunto de dados. A Fórmula 5 define a distância de Levenshtein.

A formula da distância de edição é divida em quatro casos onde são definidos para o mapeamento de atributos como, visualizando da parte superior à inferior:

$$lev(a,b) = \begin{cases} |a| & \text{se } |b| = 0, \\ |b| & \text{se } |a| = 0, \\ lev(cauda(a), cauda(b)) & \text{se } a[0] = b[0], \\ 1 + min \begin{cases} lev(cauda(a), b) \\ lev(a, cauda(b)) \\ lev(cauda(a), cauda(b)) \end{cases} & \text{senão.} \end{cases} \tag{5}$$

- **Primeiro caso**: ocorre quando o tamanho de *b* é vazio, retornando o tamanho de *a*.
- **Segundo caso**: ocorre quando o tamanho de *a* é vazio, retornando o tamanho de *b*.
- **Terceiro caso**: ocorre quando o primeiro caractere de *a* é igual ao primeiro caractere de *b*, retornando a função de Levenshtein com as entradas da cauda de *a* e cauda de *b*, onde a cauda são todos os caracteres da *string*, exceto o primeiro.
- **Quarto caso**: ocorre quando o tamanho de *a* e *b* são positivos e o primeiro caractere de *a* é diferente do primeiro caractere de *b*, retornando 1 mais o valor mínimo entre as funções da distância de Levenshtein das *strings* a serem comparadas.

A aplicação do método de Levenshtein pode ser visto na seção de código descrita como *(((0.8/pv) * resultados[j]) + lev(colsDicAtr[j],colsBaseCliente[i])*, na Figura 3. Essa parte do código faz a soma da pontuação de um atributo *j* com a métrica de Levenshtein gerada com os atributos relacionados, definindo então a pontuação final da correlação de atributos.

Finalmente, com a seleção das melhores correlações de atributos a partir das pontuações geradas é, então, criado um protocolo de mapeamento padronizado que é obtido pelo sistema da interface gráfica. Esse protocolo de mapeamento é disponibilizado em um tabela onde o usuário pode aplicar edições e salvar o resultado final.

A Figura 24 apresenta o resultado do mapeamento automatizado, com a tabela do protocolo de mapeamento e as tabelas dos dados base e cliente, respectivamente. O mapeamento em questão foi gerado utilizando um protocolo básico de mapeamento produzido a partir dos dados de "Local de Oferta" de 2014 e os dados de 2015 para o mapeamento automatizado.

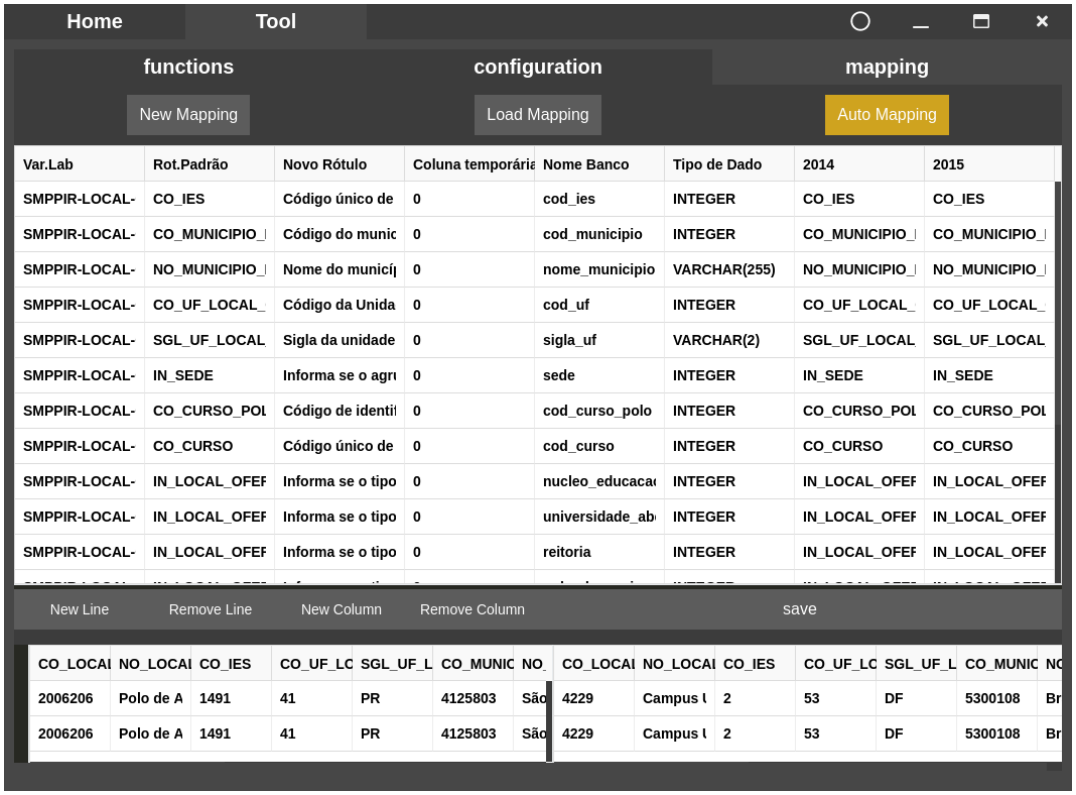



Figura 24 – Exemplo de um protocolo de mapeamento automatizado gerado na interface gráfica.

Com a interface gráfica apresentada e todas as funcionalidades detalhadas, é necessário verificar o fluxo de funcionamento com entradas e saídas específicas. Para isso será visualizada uma aplicação de um teste sobre dados abertos governamentais brasileiros, utilizando as definições de tabelas, o estabelecimento de mapeamento básico e automatizado, como também a execução das funções do *HOTMapper*.

**Algoritmo 3:** Correlação de atributos modificado de (OLINI, 2019)

```
Function correlacionaAtributos(colsBaseCliente,
 valoresColsBaseCliente, colsDicAtr, dicAtr, totalOcorrDicAtr):
    for i in tamanho(colsBaseCliente) do
        resultados = [ ]
        if i in indices_colunas_tipo_num_con_base_cliente then
            for valorBuscado in valoresColsBaseCliente[i] do
                for j in tamanho(colsDicAtr) do
                    if j in indices_colunas_tipo_num_con_dic_atr then
                        resultados[j]+= (norm(dicAtr[j]).pdf(valorBuscado)
                    end
                end
            end
            melhorCorrelacao = [ ]
            for j in tamanho(colsDicAtr) do
                if tamanho(melhorCorrelacao) == 0 OU resultados[j] >
                 melhorCorrelacao[1] then
                    melhorCorrelacao = [colsDicAtr[j], resultado]
                end
            end
            novasColunas[melhorCorrelacao[0]] = colsBaseCliente[i]
        else
            resultadosNegados = [ ]
            for j in tamanho(colsDicAtr) do
                inicializa resultados[j] = 0.0
                inicializa resultadosNegados[j] = 1.0
            end
            for valorBuscado in valoresColsBaseCliente[i] do
                for coluna in colsDicAtr do
                    if valorBuscado in dicAtr[coluna] then
                        ocorrencias = dicAtr[coluna][valorBuscado]
                        relacaoAoTotal = ocorrencias/totalOcorrDicAtr[coluna]
                        resultados[coluna]+= relacaoAoTotal
                        resultadosNegados[coluna] = relacaoAoTotal
                    end
                end
            end
            melhorCorrelacao = [ ]
            — Parte Modificada —
            for j in tamanho(colsDicAtr) do
                pv = (0.8 * resultados[j]) + (0.2 * resultadosNegados[coluna])
                resultados[j] = (((0.8/pv) * resultados[j]) + lev(colsDicAtr[j],
                 colsBaseCliente[i]))/2
                if tamanho(melhorCorrelacao) == 0 OU resultados[j] >
                 melhorCorrelacao[1] then
                    melhorCorrelacao = [colsDicAtr[j], resultado]
                end
            end
            novasColunas[melhorCorrelacao[0]] = colsBaseCliente[i]
            — Parte Modificada —
        end
    end
    criaProtocoloMapeamento(novasColunas)
    return
```

# 5 TESTES E RESULTADOS

A interface gráfica e o fluxo de execução são testados utilizando conjuntos de dados do INEP (INEP, 2021), seguindo os passos de criação das tabelas, geração de protocolos de mapeamento básico e automatizado, como também a inserção de dados no banco de dados. Os conjuntos de dados testados são referentes à "Local de Oferta da Educação Superior", para anos de 2018 (base dicionário) e 2019 (base cliente). Além dos testes de execução do sistema, foi analisado também o tempo de execução do algoritmo de mapeamento automatizado, aplicando diferentes limitações comparando os resultados.

A Tabela 2 apresenta a definição dos dados de "Local de Oferta da Educação Superior" dos anos de 2018 e 2019. Os dados mostram a quantidade de cada tipo de atributo de cada relação, a quantidade de atributos e a quantidade de tuplas.

| | 2018 | 2019 |
|---|---|---|
| Atributos | 50 | 50 |
| Tuplas | 767725 | 1254045 |
| Tipo Texto | 1 | 1 |
| Tipo Numérico Discreto | 6 | 6 |
| Tipo Numérico Contínuo | 1 | 1 |
| Tipo Booleano | 42 | 42 |

Tabela 2 – Tabela de definições dos dados de Local de Oferta.

Fonte: autoria própria, 2021.

Inicialmente, o teste do sistema é feito gerando um protocolo de mapeamento, conforme exemplificado na Figura 11, utilizando os dados de 2018 como base. O protocolo inicial é gerado com a colunas padrões e uma coluna com o ano de 2018, onde define todos os atributos do conjunto de dados apresentados em 17 linhas. Esse protocolo é, então, preenchido de acordo com a padronização do HOTMapper (EHRENFRIED et al., 2019) para gerar as transformações corretas do conjunto de dados ao banco de dados.

Utilizando o protocolo de mapeamento, é definido também um arquivo de configuração da tabela para possibilitar a criação no banco de dados e inserção dos dados históricos. Essa tabela segue a padronização das definições de tabelas do *HOTMapper*, com os atributos apresentados no arquivo de mapeamento referenciando a coluna de identificação, as chaves primárias e a descrição da tabela. É possível visualizar a tela do resultado final do processo de geração do protocolo de mapeamento básico na Figura 25.

Após definir o protocolo de mapeamento e o arquivo de configuração da tabela é, então, aplicado a geração de mapeamento automático para os dados de 2019. O processo de mapeamento automatizado é iniciado passando os arquivos de dados de 2018 e 2019, como

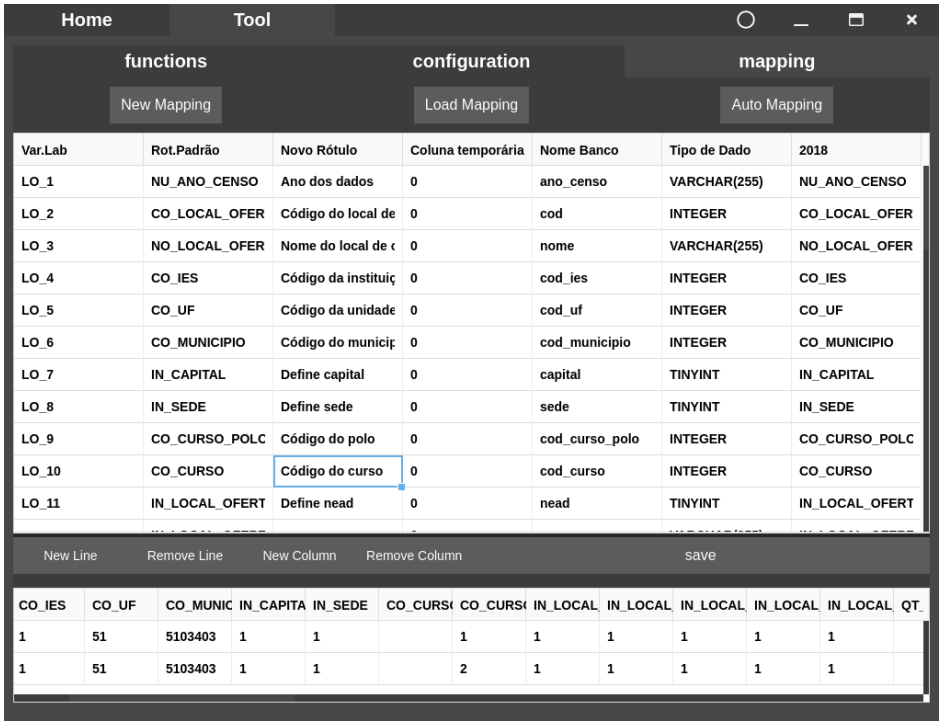



Figura 25 – Teste da geração de protocolo de mapeamento básico na interface gráfica.

também o protocolo de mapeamento no formulário carregado ao acionar o botão *Auto Mapping*. A execução do algoritmo é iniciada e, com as melhores correlações dos atributos é gerado a coluna de 2019 no protocolo de mapeamento. Antes de apresentar a tela e o resultado do mapeamento automatizado, será feito uma análise da aplicação do algoritmo.

Analisando as pontuações de correlação dos atributos do algoritmo já mencionado, podemos comparar as métricas e seleção de resultados. A saída tem-se, primeiramente, o atributo a ser comparado com o dicionário de atributos, em seguida os atributos do dicionário de atributos com suas pontuações (valor que representa a correlação entre atributos). As pontuações são divididas em: a pontuação total do atributo; P1 que apresenta a pontuação da análise do conjunto de dados; P2 que apresenta a pontuação da comparação do nome dos atributos.

O Código 5.1 apresenta as correlações para o único atributo do tipo texto da base cliente, que é definido como *NO_LOCAL_OFERTA*. Analisando todas as colunas e resultados, a correspondência com o atributo de mesmo nome da base dicionário é ideal. A pontuação para ambos P1 e P2 é 1.00, o que significa que os algoritmos encontraram 100% de correlação entre esses atributos. Visualizando a correlação com os outros atributos, como o *CO_LOCAL_OFERTA*, a pontuação de P1 é de 0.00, apresentando uma correlação acurada entre os dados de texto e os dados inteiros do atributo comparado, já P2 define 0.93, representando a correspondência das *strings* onde altera-se somente o primeiro caractere. Dessa forma, tem-se que a análise e resultado obtiveram sucesso na correlação de atributos do tipo texto.

```
1  ####################################
2  Cliente: NO_LOCAL_OFERTA
3  Dicionario: NU_ANO_CENSO: 0.18 - P1: 0.00 P2: 0.37
4  Dicionario: CO_LOCAL_OFERTA: 0.47 - P1: 0.00 P2: 0.93
5  Dicionario: NO_LOCAL_OFERTA: 1.00 - P1: 1.00 P2: 1.00
6  Dicionario: CO_IES: 0.14 - P1: 0.00 P2: 0.29
7  Dicionario: CO_UF: 0.15 - P1: 0.00 P2: 0.30
8  Dicionario: CO_MUNICIPIO: 0.15 - P1: 0.00 P2: 0.30
9  Dicionario: IN_CAPITAL: 0.24 - P1: 0.00 P2: 0.48
10 Dicionario: IN_SEDE: 0.14 - P1: 0.00 P2: 0.27
11 Dicionario: CO_CURSO_POLO: 0.18 - P1: 0.00 P2: 0.36
12 Dicionario: CO_CURSO: 0.17 - P1: 0.00 P2: 0.35
13 ...
```

Código 5.1 – Pontuações da correlação de atributo texto.

O Código 5.2 apresenta as correlações dos atributos do tipo numérico discreto da base cliente, que são definidos por *CO_LOCAL_OFERTA*, *CO_IES*, *CO_UF*, *CO_MUNICIPIO*. Os atributos selecionados para o mapeamento correspondem aos atributos ideais, porém, é possível visualizar que existem pontuações altas para atributos que não são ideias para a solução. O atributo *CO_LOCAL_OFERTA* que possui sua correspondência correta, também tem a correlação com o atributo *CO_IES* com uma pontuação de P1 de 0.82, a qual apresenta uma fraqueza do algoritmo de correlação onde os valores numéricos discretos sequencial podem ser relacionados sem possuir o mesmo contexto. Esse problema surge na correlação de atributos diferentes que possuem sequencias iguais ou parecidas, onde a pontuação é menor conforme a diferença entre sequencias. A métrica de correlação de nome de atributos pode ajudar a diminuir a questão da correlação utilizando valores, como visto com o atributo *CO_IES* que possui uma correspondência P1 de 1.00 com *IN_LOCAL_OFERTA_POLO*, porém, a pontuação P2 faz com que não seja a melhor opção para o mapeamento, fazendo com que a média dos valores de P1 e P2 tenha um resultado de correlação baixo.

A utilização das métricas de correlação de valores de atributos, adjunto a correlação de nomes de atributos, complementam para a escolha de um mapeamento mais preciso através da aplicação da média dos valores. Com as métricas aplicadas separadamente, o resultado de correlação apresentaria problemas em casos específicos, principalmente nos atributos do tipo numérico discreto, como visto entre os atributos *CO_IES* e *IN_LOCAL_OFERTA_POLO*.

```
1  ####################################
2  Cliente: CO_LOCAL_OFERTA
3  ...
4  Dicionario: CO_LOCAL_OFERTA: 1.00 - P1: 1.00 P2: 1.00
5  Dicionario: CO_IES: 0.60 - P1: 0.82 P2: 0.38
6  Dicionario: CO_MUNICIPIO: 0.21 - P1: 0.04 P2: 0.37
7  Dicionario: IN_CAPITAL: 0.20 - P1: 0.00 P2: 0.40
8  Dicionario: CO_CURSO_POLO: 0.27 - P1: 0.11 P2: 0.43
9  Dicionario: CO_CURSO: 0.23 - P1: 0.03 P2: 0.43
10 ...
11 ####################################
12 Cliente: CO_IES
13 ...
14 Dicionario: CO_LOCAL_OFERTA: 0.22 - P1: 0.05 P2: 0.38
15 Dicionario: CO_IES: 1.00 - P1: 1.00 P2: 1.00
16 Dicionario: CO_UF: 0.71 - P1: 0.87 P2: 0.55
17 Dicionario: IN_LOCAL_OFERTA_POLO: 0.61 - P1: 1.00 P2: 0.23
18 Dicionario: QT_COMPUTADOR_DISCENTE: 0.51 - P1: 0.67 P2: 0.36
19 Dicionario: IN_ACESSIBILIDADE: 0.51 - P1: 0.76 P2: 0.26
20 ...
21 ####################################
22 Cliente: CO_UF
23 ...
24 Dicionario: CO_UF: 1.00 - P1: 1.00 P2: 1.00
25 Dicionario: CO_MUNICIPIO: 0.23 - P1: 0.00 P2: 0.47
26 Dicionario: IN_SEDE: 0.47 - P1: 0.77 P2: 0.17
27 Dicionario: CO_CURSO_POLO: 0.32 - P1: 0.20 P2: 0.44
28 Dicionario: QT_COMPUTADOR_DISCENTE: 0.56 - P1: 0.90 P2: 0.22
29 Dicionario: IN_ACESSIBILIDADE: 0.46 - P1: 0.83 P2: 0.09
30 ...
31 ####################################
32 Cliente: CO_MUNICIPIO
33 ...
34 Dicionario: NO_LOCAL_OFERTA: 0.15 - P1: 0.00 P2: 0.30
35 Dicionario: CO_UF: 0.23 - P1: 0.00 P2: 0.47
36 Dicionario: CO_MUNICIPIO: 1.00 - P1: 1.00 P2: 1.00
37 Dicionario: CO_CURSO: 0.25 - P1: 0.00 P2: 0.50
38 Dicionario: IN_PISCINA: 0.18 - P1: 0.00 P2: 0.36
39 Dicionario: IN_CINEMA: 0.14 - P1: 0.00 P2: 0.29
40 ...
```

Código 5.2 – Pontuações da correlação de atributos numéricos discretos.

Para o atributo do tipo numérico contínuo *NU_ANO_CENSO*, o Código 5.3 apresenta a pontuação das correlações dos atributos. A correlação obtida é a ideal para o mapeamento, apresentando uma pontuação de 0.98. Já a correlação para os atributos incorretos para o

mapeamento apresentaram uma pontuação baixa. Dessa forma, o algoritmo obteve êxito em mapear o atributo numérico contínuo.

```
1  ######################################
2  Cliente: NU_ANO_CENSO
3  Dicionario: NU_ANO_CENSO: 0.98 - P1: 0.97 P2: 1.00
4  Dicionario: CO_LOCAL_OFERTA: 0.10 - P1: 0.00 P2: 0.20
5  Dicionario: NO_LOCAL_OFERTA: 0.13 - P1: 0.00 P2: 0.27
6  Dicionario: CO_IES: 0.17 - P1: 0.00 P2: 0.34
7  Dicionario: CO_UF: 0.08 - P1: 0.00 P2: 0.17
8  Dicionario: CO_MUNICIPIO: 0.12 - P1: 0.00 P2: 0.25
9  Dicionario: IN_CAPITAL: 0.04 - P1: 0.00 P2: 0.09
10 Dicionario: IN_SEDE: 0.12 - P1: 0.00 P2: 0.25
11 Dicionario: CO_CURSO_POLO: 0.12 - P1: 0.00 P2: 0.24
12 Dicionario: CO_CURSO: 0.21 - P1: 0.00 P2: 0.42
13 ...
```

Código 5.3 – Pontuações da correlação de atributo numérico contínuo.

Os atributos do tipo booleano apresentaram a limitação do algoritmo de correlação de atributos utilizando valores das colunas, limitação que é resolvida pela correlação dos nomes dos atributos, como visto nos atributos do tipo numérico discreto.

A execução do algoritmo de mapeamento com os conjuntos de dados de 2018 e 2019 com, respectivamente, 767725 e 1254045 tuplas, levou aproximadamente 3 horas e 6 minutos em um sistema com um processador Intel Xeon E3-1200 3.10GHz e 8GB de memória.

A Tabela 3 apresenta uma comparação da aplicação do mapeamento automatizado sem limitações com uma aplicação de limitação da utilização de, no máximo, 300000 linhas dos conjuntos de dados. Essa comparação é feita sobre alguns atributos da base cliente com dois atributos da base dicionário. O atributo *NO_LOCAL_OFERTA* na aplicação sem limitação apresenta uma pontuação de 1.00 comparando com o mesmo atributo no dicionário, já na aplicação com limitação apresenta uma pontuação de 0.99. Da mesma maneira, o atributo *CO_IES* apresenta uma variação de 0.02 entre as comparações sem e com limitação. Apesar da diferença na variação da pontuação das correlações dos atributos, a aplicação do método com limitação apresentou uma execução de aproximadamente 18 minutos para a geração do resultado, o que significa que pode ser uma aplicação interessante se a pequena diferença de precisão é menos importante que o tempo de execução.

A aplicação final do sistema de integração de dados utiliza a limitação de 300000 linhas para a geração de mapeamentos automatizados. A decisão da limitação dá-se pelo tempo de definição de um mapeamento, já que para a utilização diária da ferramenta, a geração de mapeamentos de dados deve ser relativamente rápida, pois o sistema ficará bloqueado enquanto o processo não é finalizado.

| Atributo Cliente | Sem Limitação | | Com Limitação | |
|---|---|---|---|---|
| | NO_LOCAL_OFERTA | CO_IES | NO_LOCAL_OFERTA | CO_IES |
| NO_LOCAL_OFERTA | 1.00 | 0.14 | 0.99 | 0.14 |
| CO_LOCAL_OFERTA | 0.47 | 0.56 | 0.47 | 0.55 |
| CO_IES | 0.14 | 1.00 | 0.14 | 0.98 |
| CO_UF | 0.13 | 0.30 | 0.13 | 0.28 |
| CO_MUNICIPIO | 0.15 | 0.22 | 0.15 | 0.22 |

Tabela 3 – Tabela de comparação de pontuações das aplicações.

Fonte: autoria própria, 2021.

Finalmente, após o protocolo de mapeamento ser concluído a tela apresentada na Figura 26 disponibiliza a tabela para alterações. O usuário pode visualizar o resultado final e aplicar modificações conforme necessário. Por fim, esse protocolo é salvo na pasta de mapeamentos do *HOTMapper* e está pronto para ser aplicado na transformação de dados.

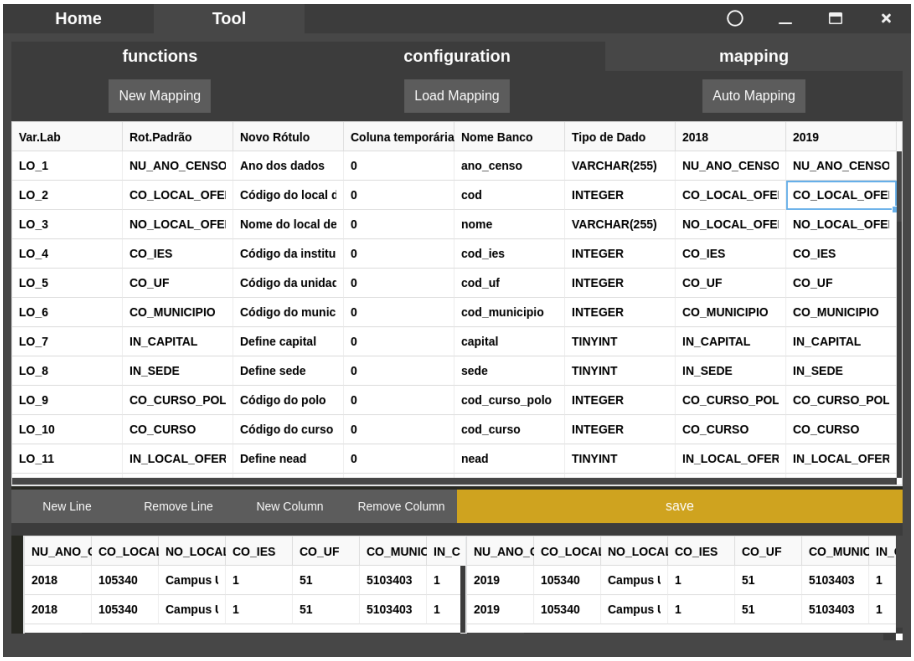



Figura 26 – Teste da geração de protocolo de mapeamento automatizado na interface gráfica.

Para concluir o processo de teste é feito a criação da tabela e a inserção dos dados de 2018 e 2019 da tabela de "Local de Oferta do Ensino Superior". Esse processo é apresentado na Figura 27 que utiliza o sistema de fila para executar os comandos de inserção de dados. Após a inserção das funções na lista, o sistema requisita a execução sequencial das funções pelo HOTMapper. Como visto no visualizador de saída do sistema, o processo obteve sucesso em criar a tabela e inserir ambos os dados de 2018 e 2019.

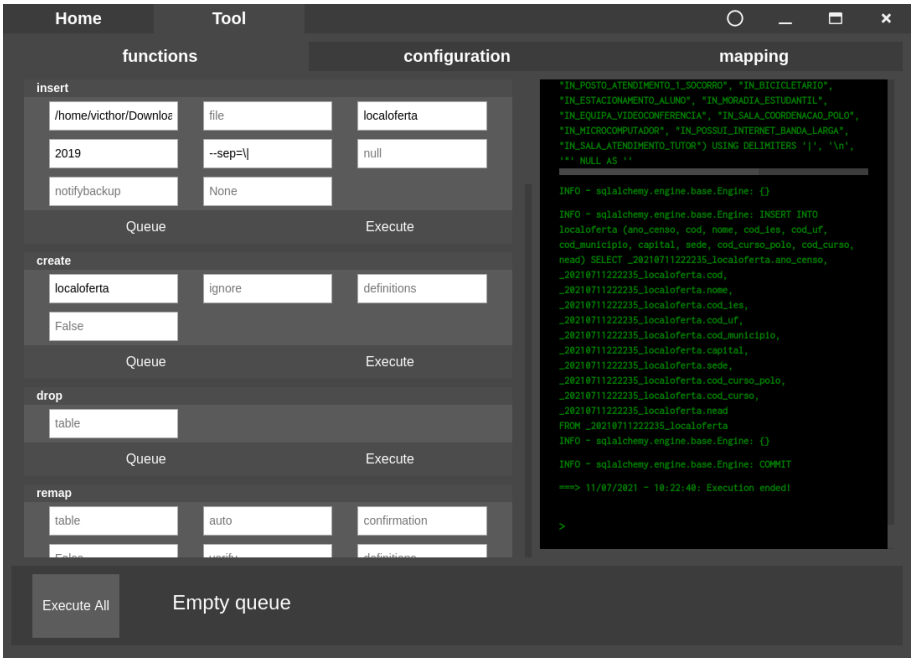



Figura 27 – Teste da execução de funcionalidades da integração de dados na interface gráfica.

### 5.0.1 Considerações Finais

A aplicação dos testes no sistema de integração de dados desenvolvido apresentou bons resultados, explicitando todo o fluxo de uma definição de tabelas até a inserção dos dados em um banco de dados. Dessa forma, foi possível visualizar que houve uma aprimoramento do algoritmo desenvolvido em (OLINI, 2019), com o emprego da métrica de distância de Levenshtein, onde casos como o atributo CO_IES do conjunto de dados de Local de Oferta apresentou a correlação de atributos correta entre a base cliente e o dicionário de atributos. Para CO_IES, o método base de correlação encontra um problema com outros atributos que apresentam uma pontuação de 1.0, gerando um caso de decisão, porém, com a segunda métrica adicionada ao mapeamento automatizado, o resultado final é o esperado. Além disso, a limitação da quantidade de tuplas no mapeamento automatizado permitiu uma execução do processo em menor tempo (de aproximadamente 3 horas para 20 minutos), em detrimento da precisão das pontuações (aproximadamente 2%), que foi uma decisão final de projeto.

O sistema de integração de dados aplicou uma interface gráfica sobre o *HOTMapper* com sucesso, apresentando um aprimoramento da utilização da ferramenta de integração de dados que possui somente um CLI. De certa forma, essa interface gráfica facilitou a utilização da ferramenta já que possui diversas funcionalidades em um único programa. As funcionalidades permitiram definir uma tabela na tela *configuration*, gerar um mapeamento básico e um mapeamento automatizado na tela *mapping* e executar a criação e inserção de dados na tela *functions*, tudo em uma única interface. O mesmo processo, sem a utilização da interface gráfica, necessita navegar pelas pastas de arquivos do *HOTMapper* para acessar as definições de tabelas, como também buscar os mapeamentos e executar os comandos do CLI em um

terminal. Dessa forma, a interface gráfica torna mais eficiente a utilização da ferramenta.

Finalmente, os testes no sistema demonstraram que o conjunto de características visto na Figura 4 foram implementados com sucesso. Com isso, o sistema apresentou uma interface gráfica, mapeamentos automatizados, análise em dados abertos brasileiros, como também possui código aberto, características que as ferramentas analisadas não constituem simultaneamente.

| Ferramenta | Interface | Mapeamento Automatizado | Analise em Dados Abertos Brasileiros | Código Aberto |
|---|---|---|---|---|
| Clio | Não disponível | Não | Não | Não |
| ++Spicy | Gráfica | Não | Não | Sim |
| Metamorfose | Gráfica | Não | Sim | Não |
| Automatch | Não disponível | Sim | Não | Não |
| HOTMapper | Linha de comando | Não | Sim | Sim |
| Sistema Apresentado | Gráfica | Sim | Sim | Sim |

Tabela 4 – Tabela final de comparação de ferramentas de integração de dados.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

Os dados abertos tornam-se cada vez mais importantes no cenário político ao passar dos anos, e a aplicação de métodos de análise e integração de dados são cada vez mais relevantes. A aplicação de uma interface gráfica em um sistema de integração de dados, adjacente a um método automatizado de geração de mapeamento de dados, trouxe resultados esperados quanto a aplicação desejada.

A GUI proposta nesse trabalho definiu telas para todas as funcionalidades da ferramenta de integração de dados carregada, como também telas para a visualização da documentação, criação de mapeamentos e edição de arquivos de configuração, onde todas possuem suas importâncias dentro do sistema geral. A tela de documentação facilita o acesso rápido ao entendimento da ferramenta carregada. A tela de configurações permite a visualização rápida e edição de arquivos de configuração da ferramenta. A tela de funcionalidade disponibiliza todas as funções da ferramenta, como também um sistema de execução em fila e um visualizador da saída da ferramenta. A tela de mapeamento disponibiliza funcionalidades para criar, carregar e gerar mapeamentos de formas automatizadas. Dessa forma, a interface aplica funcionalidades importantes para produzir uma maior acessibilidade à ferramenta de integração de dados, como também facilitar a geração de mapeamentos.

Os testes e resultados obtidos apresentaram sucesso na aplicação do sistema, desde a geração de um protocolo de mapeamento até a inserção dos dados em um banco de dados. As correlações de atributos, no processo de mapeamento automatizado, foram as ideais para o mapeamento dos conjuntos de dados utilizados. Os atributos do tipo texto e numérico contínuo obtiveram pontuações e distinções de atributos adequadas. Já os atributos do tipo numérico discreto apresentaram uma fraqueza na correlação de atributos a partir dos conjuntos de dados, porém, a métrica de correlação dos nomes gerou uma classificação final ideal para a escolha do melhor atributo.

O sistema pode se beneficiar de diversas melhorias e aplicações em trabalhos futuros. O aprimoramento do mapeamento automatizado pode ser feito aplicando alguma forma de definir automaticamente os tipos das colunas dos conjuntos de dados, removendo o processo manual de definição pelo usuário. Outra melhoria na parte dos mapeamentos é a possibilidade da geração dos mapeamentos sem a necessidade de um protocolo de mapeamento base, tornando o processo mais rápido e independente. Além disso, a possibilidade de adicionar mais de dois conjuntos de dados para geração de um protocolo de mapeamento com vários anos seria uma adição interessante à ferramenta. Finalmente, definir uma melhor modularização do sistema permitiria aplicar a interface gráfica a outras ferramentas de integração de dados, gerando uma agregação de ferramentas de integração de dados em uma única interface.

# Referências


BATINI, C.; LENZERINI, M.; NAVATHE, S. A comparative analysis of methodologies for database schema integration. **ACM Comput. Surv.**, v. 18, p. 323–364, 12 1986. Citado na página 19.

BERLIN, J.; MOTRO, A. Database schema matching using machine learning with feature selection. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2002. p. 452–466. Citado 5 vezes nas páginas 14, 27, 28, 44 e 45.

CHUKMOL, U.; RIFAIEH, R.; BENHARKAT, A.-N. Exsmal: Edi/xml semi-automatic schema matching algorithm. In: . [S.l.: s.n.], 2005. p. 422–425. Citado na página 21.

CODD, E. F. A relational model of data for large shared data banks. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 13, n. 6, p. 377–387, jun. 1970. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/362384.362685>. Citado na página 16.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados 8ª Edição**. [S.l.]: Elsevier Editora Ltda., 2004. Citado 2 vezes nas páginas 15 e 16.

DIRENE., A. I. et al. C3sl - from education to public transparency, fifteen years developing computer systems for the brazilian society. In: INSTICC. **European Space project on Smart Systems, Big Data, Future Internet - Towards Serving the Grand Societal Challenges - EPS Rome 2016,**. [S.l.]: SciTePress, 2016. p. 50–72. ISBN 978-989-758-207-3. Citado na página 30.

EHRENFRIED, H. V. et al. Hotmapper: Historical open data table mapper. In: **EDBT**. [S.l.: s.n.], 2019. p. 550–553. Citado 5 vezes nas páginas 14, 26, 27, 30 e 49.

FAGIN, R. et al. Clio: Schema mapping creation and data exchange. p. 198–236, 01 2009. Citado 2 vezes nas páginas 21 e 23.

FAGIN, R. et al. Data exchange: Semantics and query answering. 06 2003. Citado na página 24.

FOUNDATION, O. K. **Defining Open in Open Data, Open Content And Open Knowledge**. 2015. Disponível em: <http://opendefinition.org/od/2.1/pt-br/>. Citado 2 vezes nas páginas 14 e 18.

Governo Brasileiro. **Portal Brasileiro de Dados Abertos**. 2011. Disponível em: <https://dados.gov.br/>. Citado na página 18.

HAAS, L. M.; LIN, E. T.; ROTH, M. A. Data integration through database federation. **IBM Systems Journal**, v. 41, n. 4, p. 578–596, 2002. Citado na página 19.

HOOKWAY, B. Chapter 1: The subject of the interface. **Interface. MIT Press**, p. 1–58, 2014. Citado na página 22.

INEP. Microdados. In: . [S.l.: s.n.], 2021. Citado 2 vezes nas páginas 30 e 49.

ISOTANI, S.; BITTENCOURT, I. I. **Dados Abertos Conectados: Em busca da Web do Conhecimento**. [S.l.]: Novatec Editora, 2015. Citado na página 18.

KITCHIN, R. **The data revolution: Big data, open data, data infrastructures and their consequences**. [S.l.]: Sage, 2014. Citado 2 vezes nas páginas 14 e 18.

KUSZERA, E. M.; PERES, L. M.; FABRO, M. D. D. Metamorfose: a data transformation framework based on apache spark. 2018. Citado 2 vezes nas páginas 25 e 40.

LENZERINI, M. Data integration: A theoretical perspective. In: . [S.l.: s.n.], 2002. p. 233–246. Citado na página 19.

LI, Y.; NGOM, A. Data integration in machine learning. 11 2015. Citado na página 19.

MARNETTE, B. et al. ++ spicy: an open-source tool for second-generation schema mapping and data exchange. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 4, n. 12, p. 1438–1441, 2011. Citado na página 24.

MILLER, R. J. Open data integration. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 11, n. 12, p. 2130–2139, 2018. Citado 3 vezes nas páginas 14, 19 e 21.

OLINI, L. Desenvolvimento de abordagem de mapeamento de esquemas em cenário de dados abertos educacionais. 2019. Citado 6 vezes nas páginas 29, 42, 43, 44, 48 e 55.

RAHM, E.; DO, H. H. Data cleaning: Problems and current approaches. **IEEE Data Eng. Bull.**, v. 23, n. 4, p. 3–13, 2000. Citado na página 21.

REPICI, J. How-to: The comma separated value (csv) file format. 2004. Disponível em: <http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm>. Citado na página 17.

SHAFRANOVICH, Y. **Common Format and MIME Type for Comma-Separated Values (CSV) Files**. RFC Editor, 2005. RFC 4180. (Request for Comments, 4180). Disponível em: <https://rfc-editor.org/rfc/rfc4180.txt>. Citado na página 17.

SHVAIKO, P.; EUZENAT, J. A survey of schema-based matching approaches. In: **Journal on data semantics IV**. [S.l.]: Springer, 2005. p. 146–171. Citado na página 19.

SINGLA, N.; GARG, D. String matching algorithms and their applicability in various applications. **International journal of soft computing and engineering**, v. 1, n. 6, p. 218–222, 2012. Citado na página 21.

ULLMAN, J. D. Information integration using logical views. **Theoretical Computer Science**, v. 239, n. 2, p. 189–210, 2000. ISSN 0304-3975. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304397599002194>. Citado na página 19.

U.S. General Services Administration. **U.S. Open Data Portal**. 2009. Disponível em: <https://www.data.gov/>. Citado na página 18.

ZAHARIA, M. et al. Apache spark: a unified engine for big data processing. **Communications of the ACM**, ACM New York, NY, USA, v. 59, n. 11, p. 56–65, 2016. Citado na página 25.