Felipe Cruz Luiz

Incorporação do Modelo de Agregados na ferramenta brModeloWeb

Brasil
2025

Felipe Cruz Luiz

Incorporação do Modelo de Agregados na ferramenta brModeloWeb

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de Bacharel, do curso de Ciências da Computação na Universidade Federal de Santa Catarina.
Orientador: Prof. Dr. Ronaldo dos Santos Mello
Coorientador: Milton Bittencourt de Souza Neto

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Ciência da Computação

Brasil
2025

Incorporação do Modelo de Agregados na ferramenta brModeloWeb

Felipe Cruz Luiz

2025

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de Bacharel, do curso de Ciências da Computação na Universidade Federal de Santa Catarina.

Banca examinadora:

Orientador: Prof. Dr. Ronaldo dos Santos Mello
Universidade Federal de Santa Catarina - UFSC

Coorientador: Milton Bittencourt de Souza Neto

Prof. Dr. Renato Fileto
Universidade Federal de Santa Catarina - UFSC

Prof. Dr. Angelo Augusto Frozza
Instituto Federal Catarinense IFC

Resumo

Os bancos de dados *NoSQL* surgiram no início do século XXI para atender às necessidades de ambientes *web* modernos, oferecendo uma abordagem mais flexível e escalável em comparação com os bancos de dados relacionais. Uma das principais abordagens para projeto de bancos de dados NoSQL é a modelagem de dados baseada em agregados. Essa modelagem organiza conjuntos de dados complexos em estruturas compactas, tratando coleções de informações como uma única unidade lógica. Isso não apenas facilita a manipulação dos dados, mas também torna mais eficiente a operação em *clusters* distribuídos, com suporte aprimorado à replicação e fragmentação. Um esquema de agregados pode ser definido como uma abstração intermediária a partir de uma modelagem conceitual, como um diagrama entidade-relacionamento, ou definida diretamente por um projetista de dados. A partir desta modelagem de agregados pode-se, então, gerar, de uma forma mais simplificada, o projeto lógico para algum modelo de dados *NoSQL*. Este trabalho propõe o desenvolvimento de um módulo para a ferramenta *BrModeloWeb* que permite a modelagem interativa de esquemas de agregados, para, a partir dela, ser possível gerar esquemas lógicos para bancos de dados *NoSQL*. Por meio de uma interface interativa, ela permite que os usuários manipulem conceitos do modelo de agregados de forma intuitiva, criando representações lógicas de dados agregados. A implementação desse módulo busca beneficiar tanto a academia quanto à indústria de *software*. Na academia, ela pode ser utilizada para fins educacionais, permitindo que estudantes compreendam melhor a modelagem de bancos *NoSQL*. Já na indústria, a proposta visa agilizar o processo de desenvolvimento de soluções que exigem bancos de dados altamente escaláveis, contribuindo para um *design* mais eficiente e adaptável de sistemas complexos.

Palavras-chave: Banco de dados NoSQL. Modelagem em Agregados. *brModeloWeb*.

Lista de ilustrações

Figura 1 – Exemplo de esquema de dados representado no modelo de agregados. Fonte: (LIMA; MELLO, 2015) 11
Figura 2 – brModelo V3 para *desktop*. Fonte: Autor 15
Figura 3 – *brModeloNext para desktop*. Fonte: Autor 17
Figura 4 – Página inicial da ferramenta *brModeloWeb*. Fonte: brModeloWeb . . . 18
Figura 5 – Exemplo de modelagem de um *eCommerce* usando *Moon Modeler*. Fonte: Datensen 20
Figura 6 – Um exemplo de modelagem de dados para documentos usando *Hackolade Studio*. Fonte: Hackolade Studio 21
Figura 7 – Exemplo de modelagem de dados para documentos usando a ferramenta DbSchema. Fonte: DbSchema 23
Figura 8 – Imagem de tutorial JointJS Fonte: https://docs.jointjs.com/learn/features/containers-and-grouping/ 26
Figura 9 – Arquitetura do Sistema Fonte: Autor 28
Figura 10 – Menu principal. Fonte: Autor 29
Figura 11 – Criação de uma Coleção. Fonte: Autor 29
Figura 12 – Cardinalidades de atributo e bloco. Fonte: Autor 30
Figura 13 – Restrição de disjunção de blocos. Fonte: Autor 30
Figura 14 – Aninhamento de Blocos. Fonte: Autor 31
Figura 15 – Persistência de modelagens. Fonte: Autor 31
Figura 16 – Estudo de caso: *login*. Fonte: Autor. 34
Figura 17 – Estudo de caso: definição das coleções. Fonte: Autor. 35
Figura 18 – Estudo de caso: painel lateral. Fonte: Autor. 36
Figura 19 – Estudo de caso: exemplo de atributo de referência. Fonte: Autor. . . . 37
Figura 20 – Estudo de caso: definição dos atributos. Fonte: Autor. 37

Lista de abreviaturas e siglas

NoSQL	*Not SQL ou Not Only SQL*
ASCII	*American Standard Code for Information Interchange*
SQL	*Structured Query Language*
ACID	*(Atomic, Consistent, Isolated, Durable)*
BASE	*(Basic Availability, Soft state, Eventual consistency)*
BD	Banco de Dados
TCC	Trabalho de Conclusão de Curso
BLOB	*Binary Large OBject*
XML	*Extensible Markup Language*
JSON	*JavaScript Object Notation*
BSON	*Binary JavaScript Object Notation*
SGBD	Sistema de Gerenciamento de Banco de Dados
Ecommerce	Comércio eletrônico
DBA	Administrador de banco de dados
TI	Tecnologia da informação
HTML	*HyperText Markup Language*
ER	Entidade Relacionamento
EER	Entidade Relacionamento Estendido
UX	Experiência do Usuário
API	*Application Programming Interface*

Sumário

Lista de ilustrações 7

1 INTRODUÇÃO 5
1.1 Motivação 5
1.2 Objetivos 7
1.2.1 Objetivo geral 7
1.2.2 Objetivos específicos 7
1.3 Metodologia 8
1.4 Organização do Trabalho 8

2 FUNDAMENTAÇÃO TEÓRICA 9
2.1 NoSQL 9
2.2 Modelo de dados orientados a agregados 9
2.3 Banco de dados chave-valor 11
2.4 Banco de dados orientado a colunas 12
2.5 Banco de dados orientados a documentos 13
2.6 Projeto Lógico de Banco de Dados 14
2.7 *brModelo* e suas versões 14
2.7.1 *brModelo* – Primeira Versão 14
2.7.2 *brModeloNext* 16
2.7.3 *brModeloWeb* 16

3 TRABALHOS RELACIONADOS 19
3.1 *Moon Modeler* 20
3.2 *Hackolade Studio* 21
3.3 DbSchema 23

4 DESENVOLVIMENTO DO MÓDULO 25
4.1 Projeto 25
4.2 Tecnologias e Ferramentas Utilizadas 25
4.2.1 *JavaScript* 25
4.2.2 *JointJS* 26
4.2.3 *AngularJS* 27
4.2.4 *Node.js* 27
4.2.5 *MongoDB* 27
4.3 Arquitetura do Sistema 27

4.4 Funcionalidades do Módulo 28

5 ESTUDO DE CASO 33
5.1 Domínio: Museu 33
5.2 Modelagem 34

6 CONCLUSÃO 39

REFERÊNCIAS 41
Apêndice A – Artigo Resumido 43
Apêndice B – Código da Aplicação 47

1 Introdução

O termo *NoSQL ("Not only SQL")* surgiu nos anos 1990 com um banco de dados relacional de código aberto criado por Carlo Strozzi, que não utilizava *SQL (Structured Query Language)* como linguagem de consulta (CATTELL, 2011).

Bancos de dados *NoSQL* não seguem o modelo relacional, nem possuem um esquema fixo, o que facilita a inclusão de novos campos e o armazenamento de dados semiestruturados ou não estruturados. Eles foram criados para atender à demanda por escalabilidade horizontal e alta performance em ambientes distribuídos, permitindo a expansão por meio da adição de novos servidores.

Mais do que uma tecnologia específica, *NoSQL* representa um movimento em direção à persistência poliglota, na qual diferentes tipos de bancos de dados são utilizados de acordo com os requisitos da aplicação (SADALAGE; FOWLER, 2013). Enquanto bancos relacionais priorizam consistência e integridade, os bancos *NoSQL* colocam flexibilidade e desempenho em primeiro plano, assumindo uma abordagem de esquema na leitura — em que a estrutura dos dados é definida na aplicação e não no banco.

Nesse contexto, a modelagem de dados em agregados tem papel fundamental. Um agregado é uma coleção de dados tratados como uma unidade lógica, o que simplifica operações e otimiza o desempenho em bancos *NoSQL*, especialmente nos modelos documento, colunares e chave-valor. Essa abordagem facilita a replicação, particionamento e manipulação de dados complexos de forma eficiente (RUIZ; MORALES; MOLINA, 2015).

Em resumo, a modelagem em *NoSQL* se afasta da rigidez relacional, adotando estruturas mais dinâmicas e orientadas ao domínio da aplicação, capazes de suportar os desafios de escalabilidade e variedade de dados do mundo moderno.

1.1 Motivação

BD *NoSQL* embora ganhem em agilidade em relação aos bancos tradicionais relacionais, carecem em esquemas bem definidos de como os dados são acomodados.

Para quem desenvolve aplicações, saber como lidar com os dados, como acessá-los e como estão estruturados é primordial para a evolução no desenvolvimento da aplicação, pois sem um esquema o desenvolvedor não tem uma base bem definida para se guiar

durante o desenvolvimento e isso pode atrasar o projeto.

A flexibilidade de esquema é um dos principais fatores que contribuem para a popularidade dos bancos de dados *NoSQL* (SADALAGE; FOWLER, 2013). Apesar de essa flexibilidade parecer vantajosa, ter informações sobre o esquema é crucial, especialmente durante o desenvolvimento de uma aplicação. Sem isso, o acesso aos dados pode se tornar mais difícil e complexo.

Frequentemente, as informações de esquema em bancos de dados*NoSQL* estão embutidas no código da aplicação, o que pode criar problemas significativos se múltiplos aplicativos desenvolvidos por diferentes pessoas, precisarem acessar o mesmo BD (SADALAGE; FOWLER, 2013). Portanto, é fundamental ter uma compreensão clara e confiável da estrutura do BD. Sem esse conhecimento, o desenvolvimento de novas aplicações ou a realização de análises de dados significativas podem se tornar tarefas árduas e, por vezes, inviáveis.

Com base nessa motivação, este Trabalho de Conclusão de Curso (TCC) visa desenvolver um módulo que permita a criação de esquemas de dados em agregados. Isso permitirá a representação, em alto nível, de diferentes tipos de bancos de dados *NoSQL*, como os orientados a documentos, chave-valor e colunar. Dessa forma, pretende-se facilitar a tarefa de projeto destes tipos de bancos de dados. A intenção é desenvolver essa ferramenta como um componente da ferramenta *brModeloWeb*[1], que é uma ferramenta amplamente utilizada principalmente na comunidade acadêmica para o projeto de bancos de dados.

Desde sua criação, a ferramenta *brModelo* (CÂNDIDO, 2005) tem sido amplamente utilizada em cursos de graduação na área da Computação para o ensino e o projeto de bancos de dados relacionais. Com o tempo, surgiram diversas demandas por versões mais acessíveis e modernas da ferramenta. Essas demandas resultaram no lançamento da *brModeloNext*(MENNA; RAMOS; MELLO, 2011), em 2011, reimplementada em *Java* e, posteriormente, no desenvolvimento da *brModeloNext NoSQL*, apresentada em 2015 como uma proposta para lidar com os desafios do projeto de esquemas em bancos *NoSQL*. Essa versão introduziu a ideia de esquemas de agregados como uma abstração para representar modelos chave-valor, orientado a colunas e orientado a documentos.

No entanto, até então, todas essas versões eram ferramentas *desktop*, o que limitava o seu acesso e uso em ambientes educacionais com restrições de instalação. A primeira tentativa de romper essa barreira ocorreu em 2016 com o lançamento da *brModeloWeb*, uma versão mais leve e acessível via navegador, facilitando seu uso em laboratórios de

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

informática e por usuários com diferentes sistemas operacionais.

Apesar dessa evolução, a *brModeloWeb* ainda não oferece suporte à modelagem de bancos *NoSQL*. Isso representa uma limitação, especialmente considerando o crescimento do uso de bancos *NoSQL* em aplicações modernas. Esses bancos, embora ofereçam agilidade e flexibilidade em relação aos bancos relacionais, carecem de esquemas bem definidos, o que pode comprometer o desenvolvimento de aplicações que dependem de uma estrutura clara dos dados.

A flexibilidade de esquema, embora atrativa, pode dificultar a manutenção e a colaboração entre diferentes desenvolvedores, especialmente quando o conhecimento sobre a estrutura dos dados está disperso ou embutido no código da aplicação (SADALAGE; FOWLER, 2013). Assim, fornecer uma forma gráfica e de alto nível para representar esses esquemas é fundamental para garantir a clareza e facilitar o desenvolvimento.

Diante desse cenário, este Trabalho de Conclusão de Curso (TCC) propõe o desenvolvimento de um módulo para a ferramenta *brModeloWeb* que permita a modelagem de bancos *NoSQL* por meio da criação de esquemas de agregados. Essa iniciativa visa preencher uma lacuna existente na ferramenta atual, ampliando suas funcionalidades e tornando-a ainda mais útil no contexto acadêmico e profissional, especialmente para aqueles que trabalham com bancos de dados não relacionais.

1.2 Objetivos

1.2.1 Objetivo geral

O presente TCC tem por objetivo o desenvolvimento de um módulo para a ferramenta *brModeloWeb* que permite a modelagem de dados agregados.

1.2.2 Objetivos específicos

Para alcançar o objetivo final do projeto, os seguintes objetivos específicos são requeridos:

- Análise das ferramentas de modelagem similares visando subsídios para o projeto do módulo proposto;
- Desenvolvimento do módulo proposto;
- Demonstração da interação com o módulo proposto através de um estudo de caso.

1.3 Metodologia

Partiu-se de um levantamento bibliográfico sobre *NoSQL* e modelagem por agregados e de um estudo comparativo de ferramentas similares para extrair requisitos e definir o escopo do módulo. Em seguida procedeu-se à especificação arquitetural e à implementação incremental das funcionalidades essenciais, com ciclos curtos de desenvolvimento, testes e refinamentos. Por fim, realizou-se um estudo de caso em um domínio de aplicação.

1.4 Organização do Trabalho

Com os objetivos e a metodologia definidos, os capítulos seguintes descrevem as etapas do trabalho: o capítulo de fundamentação apresenta conceitos sobre *NoSQL*, modelos por agregados e a evolução do *brModelo*; o capítulo de trabalhos relacionados analisa ferramentas como *Moon Modeler, Hackolade e DbSchema* para justificar requisitos e diferenciais; o capítulo de desenvolvimento documenta o projeto, as decisões tecnológicas, a arquitetura e as funcionalidades implementadas; o capítulo de estudo de caso demostra a interação com o módulo proposto e, por fim, a conclusão integra as evidências técnicas e empíricas, destacando as contribuições do módulo e apontando direções para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos necessários para o entendimento deste TCC.

2.1 NoSQL

Os bancos de dados *NoSQL* foram criados para oferecer alto desempenho (tanto em termos de velocidade quanto de escala) e alta disponibilidade em troca da perda da característica ACID *(Atomic, Consistent, Isolated, Durable)* dos bancos de dados tradicionais, optando por manter uma característica mais fraca conhecida como BASE *(Basic Availability, Soft state, Eventual consistency)* (TUDORICA; BUCUR, 2011).

O termo *NoSQL*, refere-se a tipos de bancos de dados não relacionais. Diferentemente dos bancos de dados relacionais baseados em tabelas e regras fixas, os bancos de dados *NoSQL* armazenam dados em formatos não tabulares. Eles utilizam um modelo de esquema flexível que é compatível com uma ampla gama de dados não estruturados, como documentos, pares chave-valor, colunas largas, grafos, entre outros. As organizações geralmente optam por bancos de dados *NoSQL* devido à capacidade de lidar com grandes volumes de dados que não se ajustam a um modelo relacional, oferecer um esquema dinâmico que permite maior flexibilidade, escalabilidade horizontal e facilidade no desenvolvimento. Os bancos de dados *NoSQL* surgiram como uma resposta aos desafios que grandes empresas enfrentaram no início do século XXI em relação ao processamento e armazenamento de vastas quantidades de dados(BARBOSA, 2013).

Por não priorizar a consistência, esses bancos de dados acabam perdendo as características ACID dos BD relacionais, em troca de manter um BD BASE. Ao adotar as propriedades BASE, esses BD garantem que as atualizações são eventualmente propagadas para todos os nós, mas que existem garantias limitadas sobre a consistência das leituras. Com isso, esses BD conseguem garantir alto desempenho em operações de leitura e escrita, além de garantir alta disponibilidade e escalabilidade (SADALAGE; FOWLER, 2013).

2.2 Modelo de dados orientados a agregados

Agregado é um conceito oriundo do Design Orientado ao Domínio *(Domain-Driven Design)* (EVANS, 2004). Nesse cenário, um agregado reúne objetos relacionados e os trata como uma única unidade. Essa unidade é usada para manipulação de dados e gestão da

consistência. Essa definição se alinha perfeitamente com o funcionamento de bancos de dados de chave-valor, documentos e famílias de colunas (SADALAGE; FOWLER, 2013). O modelo de agregados atua como ponte entre a modelagem relacional e as abordagens *NoSQL*, por isso uma representação relacional pode ser convertida em agregados que, por sua vez, evoluem para esquemas orientados a documentos, chave-valor ou a famílias de colunas.

No modelo relacional, as informações a serem armazenadas são divididas em um conjunto de tuplas. Uma tupla é uma estrutura de dados que contém um valor, não permitindo o aninhamento de outras tuplas (uma tupla dentro de outra) e nem de lista de valores.

A orientação agregada — utilizada pelos modelos de dados chave-valor, documento e família de colunas — os dados são tratados como unidades e tem uma estrutura mais complexa do que um conjunto de tuplas. Um agregado permite o aninhamento de listas e de outras estruturas de dados (SADALAGE; FOWLER, 2013).

O modelo de agregados foi formalizado por (LIMA; MELLO, 2015). Neste trabalho, também foi definida uma notação gráfica para ele, conforme exemplificado na Figura 1 para dados no domínio de um Museu.

O modelo de agregados é ideal para abstração de dados nos tipos de modelos de dados *NoSQL* orientado a documentos, orientado a colunas e chave-valor, pois todos estes representam objetos complexos com uma chave de acesso (identificador). Esses três modelos compartilham essa característica.

Os conceitos principais do modelo de agregados são: (i) coleção; (ii) bloco; (iii) atributo; (iv) relacionamento de agregação; (v) relacionamento de referência; e (vi) restrição de disjunção. A coleção representa um objeto do mundo real (normalmente uma entidade) e possui um identificador (atributo com prefixo ID_).

Uma coleção pode referenciar outra coleção ou a si mesma. Exemplos são *Obra* e *Salão* na Figura 1. Um bloco é um componente interno a uma coleção ou outro bloco, podendo ter uma cardinalidade associada, o que significa que várias ocorrências podem existir. Um exemplo na 1 é o bloco opcional *autoria*.

Blocos e coleções podem ter atributos que descrevem seus dados, podendo ser mono ou multivalorados. Um exemplo é o atributo monovalorado *nacionalidade* no bloco *autoria* e o atributo multivalorado *horários* na coleção *Salão*.

Os relacionamentos no modelo de agregados são de dois tipos: agregação e refe-

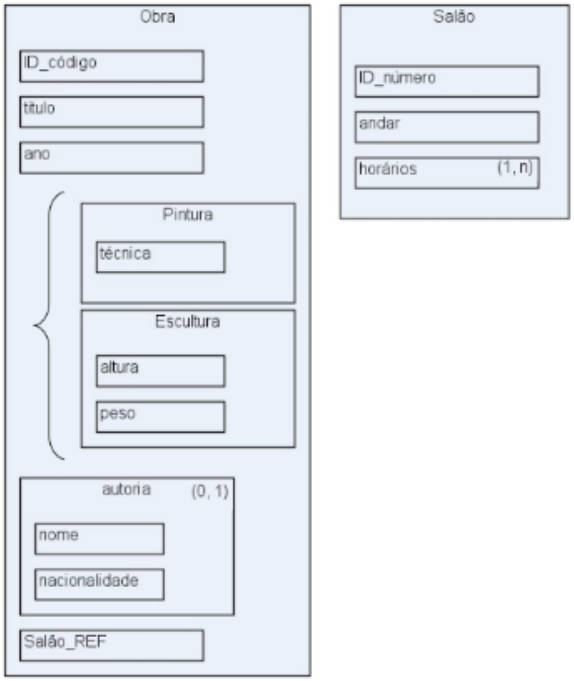

Figura 1 – Exemplo de esquema de dados representado no modelo de agregados. Fonte: (LIMA; MELLO, 2015)

rência. O relacionamento de agregação é representado por blocos aninhados em coleções ou outros blocos, indicando a existência de um objeto dentro de outro, como *Pintura* e *Escultura* na coleção *Obra*. O relacionamento de referência liga coleções através de um atributo com sufixo _REF, como o atributo *Salao_REF* na Figura 1, que indica o salão no qual a obra pode estar exposta.

A restrição de disjunção denota blocos mutuamente exclusivos em uma coleção, úteis para representar especializações disjuntas em um diagrama EER (Entidade Relacionamento Estendido). Blocos mutuamente exclusivos são indicados por um "abre chave" envolvendo todos os blocos que devem ser disjuntos. Um exemplo na Figura 1 são os blocos *Pintura* e *Escultura* na coleção *Obra*.

2.3 Banco de dados chave-valor

A ideia de pares chave-valor existe na computação há muitas décadas, é uma estrutura de dados ou conceito comum usado no desenvolvimento de sistemas de arquivos. Neste tipo de BD, os dados são armazenados como um par de chave e valor. Cada uma das chaves é única em uma coleção. O acesso aos valores é feito por meio de associação de chave-valor. As chaves precisam ser mantidas em um armazenamento de dados que possa

ser acessado rapidamente, por exemplo, uma tabela *hash*[1] (INDRAWAN-SANTIAGO, 2012).

O valor, do par chave-valor, é um *BLOB*[2] *(Binary Large OBject)* que o BD apenas armazena, sem se preocupar ou saber o que há dentro dele (SADALAGE; FOWLER, 2013).

Além de suportar armazenamento em massa, os BD chave-valor oferecem um alto desempenho em operações de leitura e escrita concorrentes. Alguns dos BD chave-valor mais populares são *Riak*[3], *Redis*[4], *MemcachedDB*[5], *BerkeleyDB*[6], *Amazon DynamoDB*[7] e *Voldemort*[8].

2.4 Banco de dados orientado a colunas

Um BD orientado a colunas utiliza o modelo de tabela, mas não suporta associação entre tabelas. Ele armazena dados por coluna, sendo que cada coluna serve como índice. Ao realizar consultas, acessa apenas as colunas necessárias para reduzir operações de entrada e saída. As consultas são processadas de forma concorrente, com cada coluna sendo tratada por um processo separado. Este modelo beneficia-se de tipos de dados semelhantes e boa compressão, sendo ideal para aplicações de agregação e armazenamento de dados (HAN et al., 2011).

Os BD orientado a colunas, conhecidos também como BD de família de colunas, oferecem um alto desempenho e uma arquitetura altamente escalável. Esses BD podem ser considerados como um tipo específico de BD chave-valor. Porém, são mais complexos e, neste caso, muda-se o paradigma de orientação a registros ou tuplas, como utilizado no modelo relacional, para orientação a atributos ou colunas.

As tabelas são constituídas por colunas, famílias de colunas e supercolunas.

A coluna é uma unidade atômica de informação, sendo expressa como um par chave-valor. Supercoluna é o agrupamento de colunas que podem ser obtidas juntas ou

[1] Uma tabela de dispersão (*hash*) é uma estrutura de dados especial, que associa chaves de pesquisa a valores.
[2] É um tipo de dado utilizado em bancos de dados para armazenar grandes volumes de informações binárias.
[3] <http://riak.com/products/>
[4] <https://redis.io//>
[5] <https://memcachedb.org/>
[6] <https://www.oracle.com/database/berkeley-db/db.html>
[7] <https://aws.amazon.com/pt/dynamodb/>
[8] <https://www.project-voldemort.com/voldemort/>

que possuem uma associação semântica entre si. Família de colunas é o agrupamento de colunas e de supercolunas que armazenam o mesmo tipo de dados, sendo a semelhança mais próxima da tabela no modelo relacional (INDRAWAN-SANTIAGO, 2012).

A estrutura de um BD colunar representa seu esquema. No entanto, em contraste com um BD relacional, é mais versátil quando se trata de adicionar novas colunas ou supercolunas. BD populares nesta categoria são: *HBase*[9], *Hypertable*[10], *Google BigTable*[11], *Cassandra*[12].

2.5 Banco de dados orientados a documentos

Os BD orientados a documentos suportam dados mais complexos do que os armazenamentos chave-valor. O termo "armazenamento de documentos"pode ser confuso: enquanto esses sistemas podem armazenar "documentos"no sentido tradicional (artigos, arquivos de texto, etc.), aqui "documento"significa um objeto autocontido e flexível (sem esquema rígido), capaz de conter campos aninhados, arrays e tipos variados, isto é, um objeto de dados que não requer referências explícitas ou ponteiros para outros objetos ou documentos externos para ser entendido ou manipulado dentro do sistema de BD (CATTELL, 2011).

Ao contrário dos armazenamentos chave-valor, esses sistemas geralmente suportam índices secundários, múltiplos tipos de documentos (objetos) por BD, e documentos ou listas aninhados. Como outros sistemas *NoSQL*, os armazenamentos de documentos não fornecem propriedades transacionais ACID.

Os BD orientados a documentos não estão preocupados com o alto desempenho em operações de leitura e escrita concorrentes, mas sim, em garantir um armazenamento eficiente de grandes volumes de dados e um bom desempenho em operações de consulta (HAN et al., 2011).

Esses BD podem ser considerados como uma subcategoria dos BD chave- valor. Sendo a diferença entre esses dois, o fato de que, enquanto os BD chave-valor armazenam valores como BLOB, os BD orientados a documentos armazenam os valores como documentos, os quais podem ser armazenados em um formato de troca de dados padrão, como XML (Extensible Markup Language), JSON (JavaScript Object Notation) ou BSON (Bi-

[9] <https://hbase.apache.org/>
[10] <https://www.hypertable.org/>
[11] <https://cloud.google.com/bigtable/>
[12] <https://cassandra.apache.org/>

nary JavaScript Object Notation)(HASHEM; RANC, 2016).

Geralmente, nos bancos de dados orientados a documentos, não há um esquema rígido, pois o número de campos não é fixo e novos campos podem ser incorporados a um documento de forma dinâmica. Um documento pode ser visto como uma estrutura de dados organizada em forma de árvore, que inclui valores simples (como *string*[13], *booleano*[14] , numérico), documentos aninhados e coleções de dados. BD populares nessa categoria são o *MongoDB*[15], *CouchDB*[16] e *Apache Solr*[17].

2.6 Projeto Lógico de Banco de Dados

O projeto de BD segue três fases principais: conceitual, lógica e física. Começando pelo projeto conceitual, é criado um esquema de alto nível que representa as informações de um domínio. Este esquema é então refinado nos níveis seguintes (projeto lógico) até alcançar o modelo específico no qual o BD será implementado fisicamente.

O modelo lógico descreve a classe dos dados manipulados pelo sistema de gerenciamento de banco de dados (SGBD), como modelos hierárquicos, orientados a objetos, relacionais e *NoSQL*.

Por fim, o projeto físico é totalmente dependente do modelo específico de armazenamento do SGBD escolhido. Essa metodologia em três níveis é amplamente aceita no projeto tradicional de BD, seguindo uma abordagem *top-down* que facilita a modelagem das informações de uma aplicação de forma clara e natural, baseando-se nos fatos do mundo real e seus relacionamentos.

2.7 *brModelo* e suas versões

A ferramenta *brModelo* é um projeto *Open Source* para a modelagem de bancos de dados desenvolvido e mantido pelo Grupo de Banco de Dados da UFSC [18]. Ela está disponível em diferentes versões, que são detalhadas a seguir.

2.7.1 *brModelo* – Primeira Versão

A primeira versão do *brModelo*4 foi criada em 2005, como parte do trabalho de conclusão de curso de Carlos Henrique Cândido, intitulado "Aprendizagem em banco

[13] Uma cadeia de caracteres.
[14] É um tipo de dado primitivo que possui dois valores, que podem ser considerados como 0 ou 1, falso ou verdadeiro.
[15] <https://www.mongodb.com/>
[16] <https://https://couchdb.apache.org/>
[17] <https://solr.apache.org/>
[18] https://github.com/gbd-ufsc

de dados: implementação de ferramenta de modelagem E.R."[19]. O objetivo primordial era construir uma aplicação que auxiliasse no ensino e na aprendizagem de técnicas de modelagem de bancos de dados relacionais. Para isso, foram adotados os conceitos e a representação gráfica sugeridos por (HEUSER, 2009).

Entre suas principais funcionalidades estavam: desenvolvimento de modelos conceituais baseados no modelo Entidade-Relacionamento (ER), conversão semiautomática do modelo ER para o modelo lógico relacional e geração do esquema físico em SQL, independente do sistema gerenciador de banco de dados relacional utilizado.

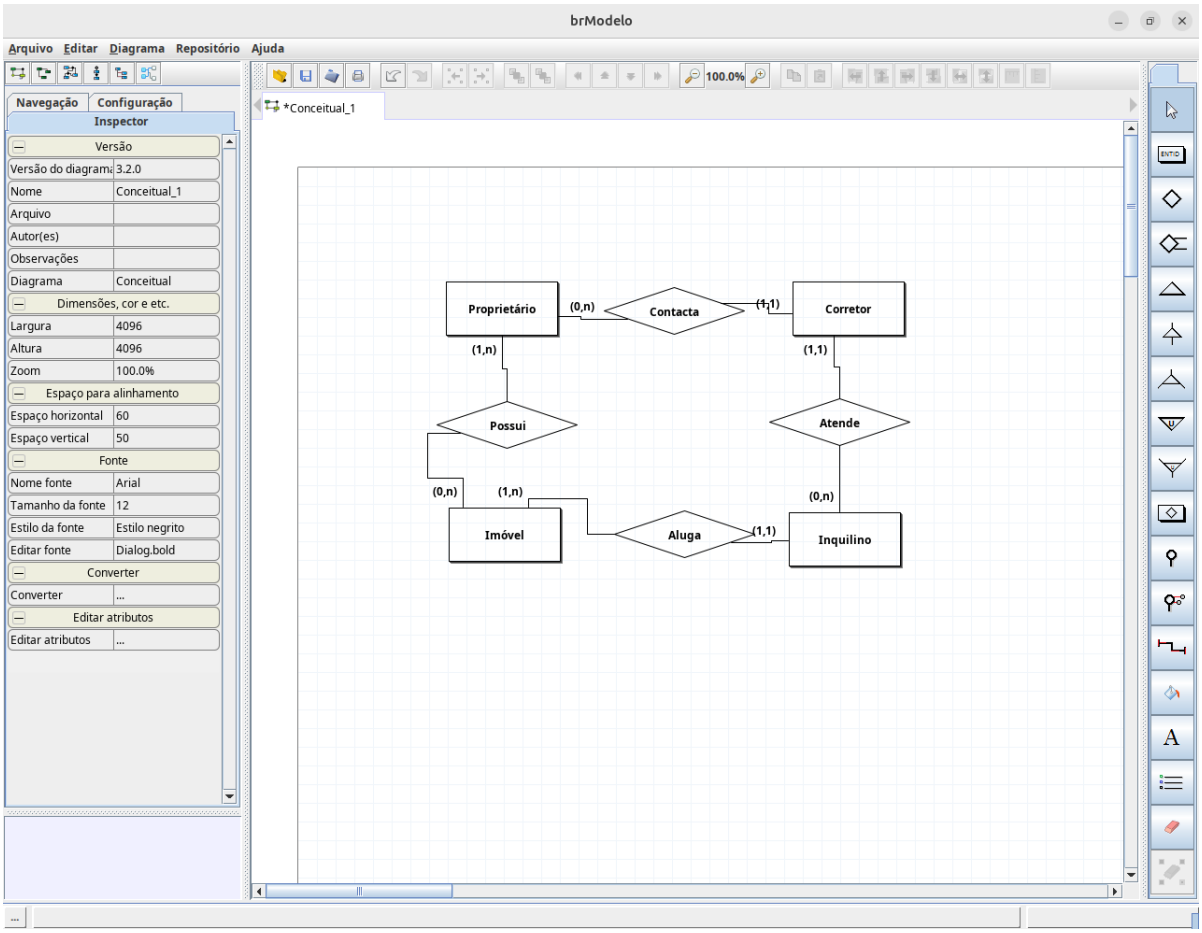

Figura 2 – brModelo V3 para *desktop*.
Fonte: Autor

A ferramenta rapidamente ganhou popularidade no meio acadêmico por ser gratuita, acompanhar todas as fases do projeto de banco de dados e suportar uma modelagem conceitual bastante completa em relação ao modelo ER. Isso a diferenciava das demais ferramentas disponíveis na época. Entretanto, a primeira versão do *brModelo* apresentava limitações importantes em termos de portabilidade, já que foi desenvolvida em *Object Pascal/Delphi*[20], restringindo sua execução ao sistema operacional *Windows*[21]. Essa de-

[19] (CÂNDIDO, 2005)
[20] Linguagens de programação.
[21] Sistema operacional.

pendência de um sistema proprietário contrariava a proposta de ser uma solução gratuita e acessível para professores e alunos.

2.7.2 *brModeloNext*

Com o intuito de superar o problema da portabilidade, iniciou-se, em 2011, o desenvolvimento do *brModeloNext*3, uma nova versão multiplataforma da ferramenta (MENNA; RAMOS; MELLO, 2011). O projeto, tema de trabalhos acadêmicos no INE/UFSC, trouxe também avanços significativos na interface, incorporando melhorias baseadas em avaliações heurísticas e testes de usabilidade.

Para garantir a execução em diferentes sistemas operacionais, optou-se pela linguagem *Java*. A biblioteca *JGraph* foi empregada para a representação gráfica dos modelos, e a arquitetura seguiu o padrão *Model-View-Controller*, facilitando a manutenção e a separação das responsabilidades do código. Entre as novas funcionalidades destacam-se:

- Modelagem em múltiplas janelas, permitindo ao usuário visualizar e comparar diferentes modelos simultaneamente;
- Edição *in-place*[22], tornando mais ágil a criação e a modificação de atributos em entidades e tabelas a partir de uma janela sobreposta ao componente;
- *Tooltips*[23] para facilitar o acesso rápido a propriedades dos objetos e aos ícones da interface.

Apesar de resolver o problema de portabilidade e trazer melhorias significativas na usabilidade, o *brModeloNext* não alcançou a mesma popularidade da versão original, em parte devido à demora na liberação de uma versão estável para uso.

Cabe ressaltar que o *brModeloNext* possui uma versão capaz de modelar bancos de dados agregados 3, contemplando conceitos como coleções aninhadas e estruturas típicas de bancos de dados *NoSQL*. No entanto, esta capacidade está restrita à versão *desktop*, pois não existe uma versão *web* da ferramenta com suporte a modelagem de agregados.

2.7.3 *brModeloWeb*

Após o desenvolvimento das versões anteriores, surgiu o *brModeloWeb*[24], uma nova iniciativa focada em oferecer uma ferramenta acessível via navegador. Diferente das versões prévias, o *brModeloWeb* foi desenvolvido com tecnologias modernas para a *web*, eliminando a dependência de sistemas operacionais específicos.

[22] Significa que algo está em sua posição correta, ou está sendo realizado sem a necessidade de criar cópias ou um espaço extra.

[23] São elementos de interface que aparecem para fornecer informações adicionais sobre um item específico quando o usuário passa o mouse sobre ele.

[24] <https://www.brmodeloweb.com/lang/pt-br/index.html>

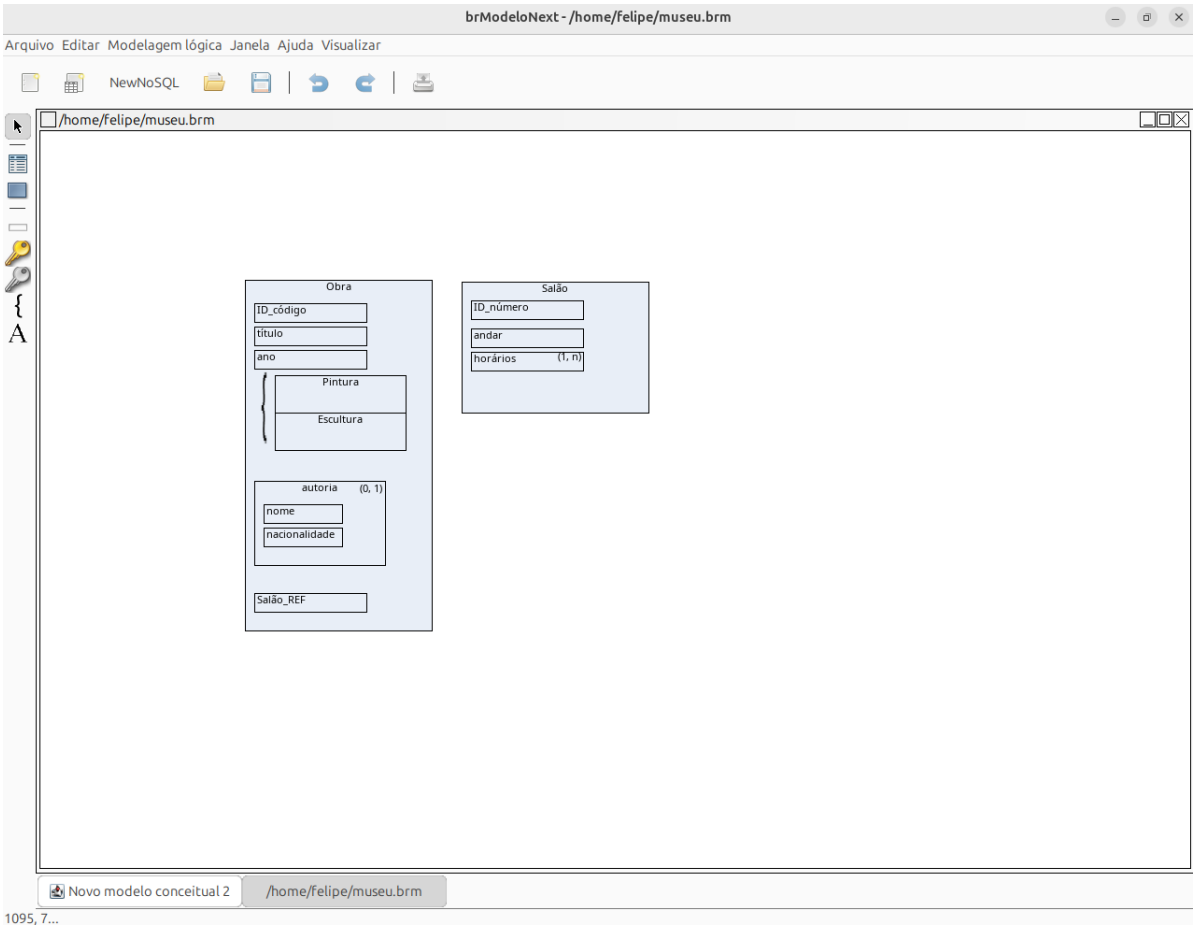

Figura 3 – *brModeloNext para desktop.*
Fonte: Autor

Apesar desse avanço em termos de acessibilidade e portabilidade, até o momento o *brModeloWeb* mantém seu foco exclusivamente na modelagem de bancos de dados relacionais, sem suporte nativo à modelagem de bancos de dados agregados ou estruturas *NoSQL*. Ou seja, enquanto o *brModeloNext* já contempla recursos para modelagem de agregados na versão *desktop*, o *brModeloWeb* ainda não oferece tal funcionalidade, restringindo-se ao paradigma relacional tradicional.

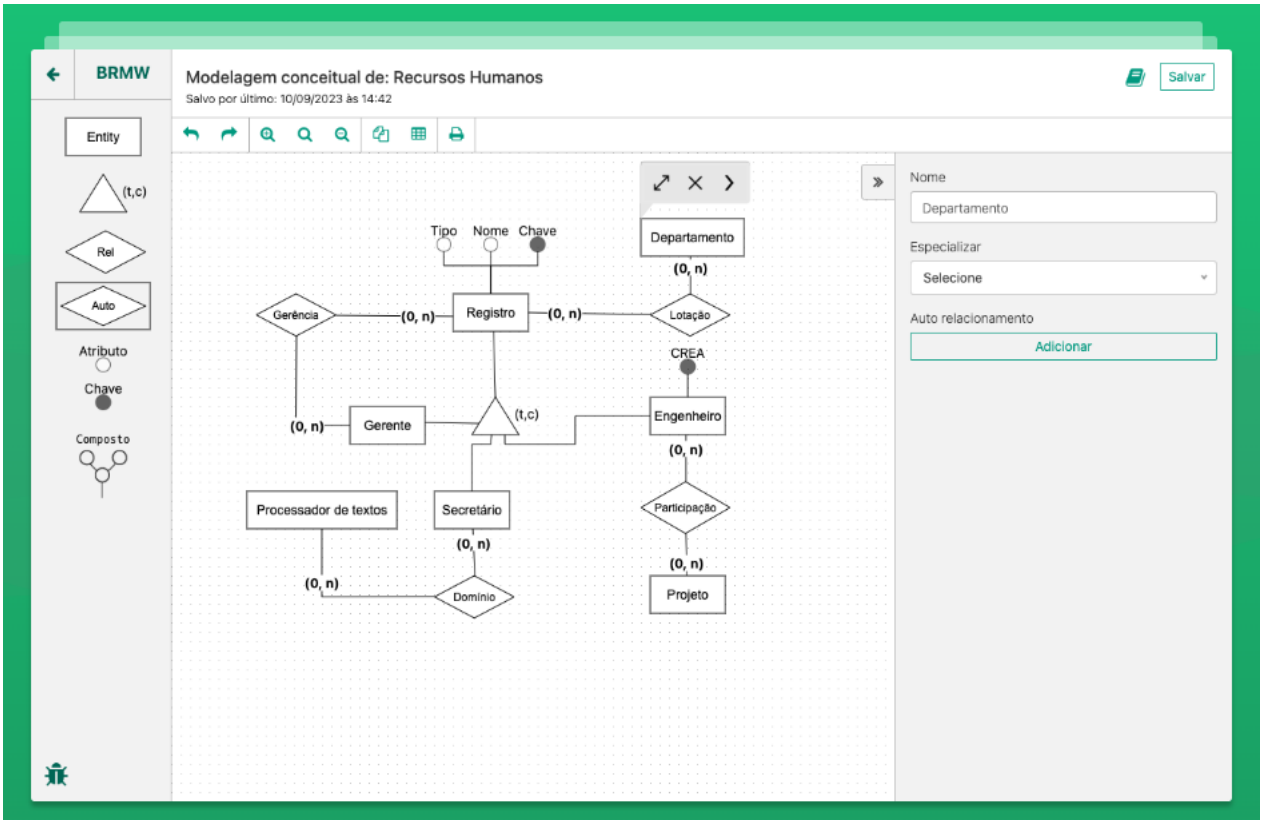

Figura 4 – Página inicial da ferramenta *brModeloWeb*. Fonte: brModeloWeb

3 Trabalhos relacionados

Durante o desenvolvimento deste trabalho, foi essencial analisar ferramentas já consolidadas no mercado que oferecem soluções semelhantes. Entre as opções disponíveis, três ferramentas se destacaram por sua relevância e abrangência de funcionalidades: *Moon Modeler, Hackolade Studio e DBSchema*. A escolha desses trabalhos relacionados foi motivada por suas abordagens práticas, recursos voltados para bancos de dados *NoSQL* e interfaces intuitivas, que serviram como referência na definição de requisitos e funcionalidades do projeto.

O *Moon Modeler*[1] se destacou pela sua interface amigável e foco em bancos de documentos como *MongoDB e Firestore*[2]. Sua representação gráfica clara e funcionalidades de exportação de modelo ajudaram a entender boas práticas de UX[3] para ferramentas de modelagem. Além disso, sua capacidade de gerar *scripts* e diagramas prontos trouxe importantes contribuições para a automatização do processo de criação de modelagem. Já o *Hackolade Studio*[4] foi escolhido por seu suporte abrangente a diversos tipos de bancos *NoSQL*, incluindo bancos de documentos, chave-valor e orientados a colunas. Isso o torna mais próximo do escopo do módulo desenvolvido. Ele também oferece uma análise de esquema que facilita a visualização de estruturas semiestruturadas, algo relevante para bases de dados flexíveis como as utilizadas em ambientes *NoSQL*.

Por fim, o *DbSchema*[5] foi incluído por ser uma ferramenta mais generalista, porém com um bom suporte a bancos relacionais e não-relacionais. Ele demonstra bem como uma abordagem unificada pode facilitar a transição entre modelos e a colaboração em times com diferentes necessidades. Além disso, suas opções de visualização de dados em tempo real e sincronização com o banco tornam a ferramenta uma referência em integração de modelagem com operação.

Essas três ferramentas forneceram contribuições valiosas sobre boas práticas de experiência do usuário, funcionalidades prioritárias e abordagens técnicas eficientes para modelagem *NoSQL*, e por isso foram essenciais como trabalhos relacionados neste projeto.

---

1 https://www.datensen.com/blog/docs/about-moon-modeler-data-modeling-tool
2 <https://firebase.google.com/>
3 Experiência do Usuário.
4 https://studio.hackolade.com/
5 https://dbschema.com/tutorials.html

3.1 *Moon Modeler*

Moon Modeler é uma ferramenta de modelagem de dados para *MongoDB*. Também pode ser utilizada com soluções como *Cosmos DB*[6], *AWS Amazon DocumentDB*[7] e outros BD orientados a documentos. O *software* pertence à categoria de ferramentas de modelagem de dados e *design* de esquemas e pode ser usado para criar e visualizar estruturas de BD.

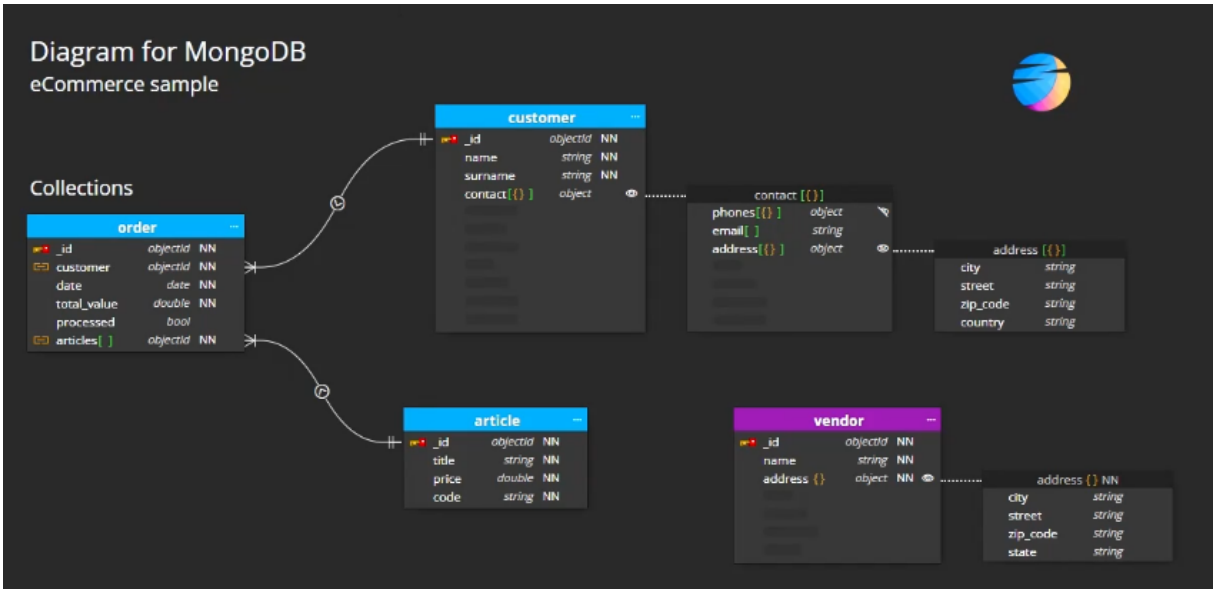

Figura 5 – Exemplo de modelagem de um *eCommerce* usando *Moon Modeler*. Fonte: Datensen

A Figura 5 mostra um exemplo de modelagem para um *eCommerce* modelado para *MongoDB* usando a ferramenta. Ela mostra as relações de *order*, tanto com *customer* quanto com *article*, e para *vendor* uma representação aparte.

A ferramenta permite a criação de diagramas para bancos de dados *NoSQL*, como *MongoDB*, *Mongoose ODM*[8] e outros. Oferece uma interface gráfica intuitiva para definir coleções, campos e tipos de dados.

Ainda, ela pode gerar *scripts* de criação de esquemas para os bancos de dados suportados, facilita a sincronização entre a modelagem visual e o BD real, reduzindo o risco de erros e inconsistências, permite a exportação das modelagens para vários formatos, incluindo JSON e outros formatos legíveis por máquina e oferece a capacidade de importar esquemas existentes para facilitar a atualização e manutenção.

6 <https://cosmos.azure.com/>
7 <https://aws.amazon.com/pt/documentdb/>
8 <https://mongoosejs.com/>

Em resumo, o *Moon Modeler* é uma ferramenta poderosa e versátil que pode ajudar equipes de desenvolvimento a gerenciar e manter esquemas de bancos de dados baseados em documentos de forma mais eficiente e colaborativa.

O módulo da ferramenta *brModeloWeb* para modelagem de agregados atua como ponte entre modelos relacionais e bancos *NoSQL*, oferece interface *web* e portabilidade. Por ser *open source* e orientado a agregados, ela se mostra mais versátil que o *Moon Modeler*, que atende apenas bancos orientados a documentos, não dispõe de versão *web*, é de código fechado e cobra uma licença fixa de 99 dólares, o módulo desenvolvido elimina custos de licenciamento e facilita colaboração e adaptação a diferentes cenários.

3.2 *Hackolade Studio*

Hackolade Studio é uma ferramenta especializada em modelagem de dados e *design* de esquemas para bancos de dados *NoSQL*, bancos de dados baseados em documentos e outras tecnologias de armazenamento de dados não relacionais.

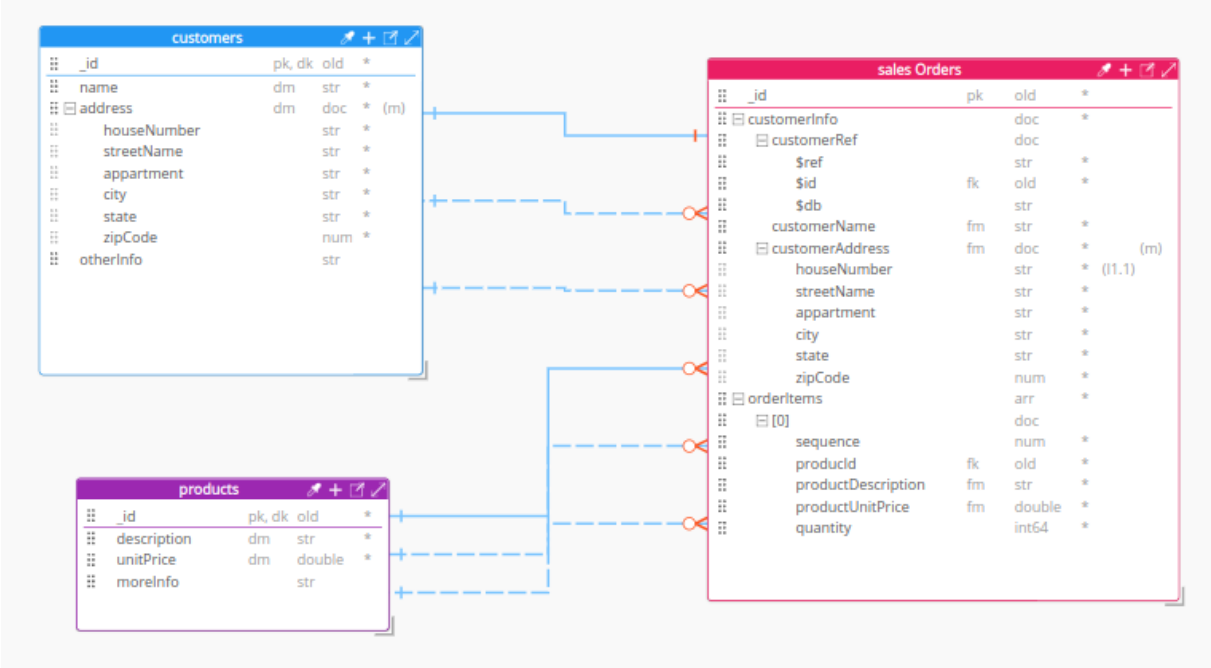

Figura 6 – Um exemplo de modelagem de dados para documentos usando *Hackolade Studio*. Fonte: Hackolade Studio

A Figura 6 demonstra um exemplo de modelagem para um sistema de vendas modelado para *MongoDB* usando a ferramenta *Hackolade Studio*. No exemplo *sales Orders* contém todas as informações, incluindo o *customer* e os produtos da venda.

Ela oferece uma interface gráfica avançada para modelagem de dados, permitindo aos usuários criar, modificar e visualizar esquemas de BD de forma intuitiva. Suporta

modelagem conceitual, lógica e física, facilitando a transição entre diferentes níveis de abstração.

Ainda, ela suporta uma ampla gama de bancos de dados, tendo em vista que a ferramenta não faz modelagem para agregados, mas sim para todos os tipos de modelagens *NoSQL, como MongoDB, Cassandra, DynamoDB, Couchbase*[9], *Neo4j*[10], entre outros, permite a geração automática de *scripts* de criação de esquemas e a sincronização entre os modelos visuais e os bancos de dados reais, e suporta engenharia reversa, permitindo a importação de esquemas existentes para facilitar a atualização e manutenção.

É uma ferramenta útil para projetos que envolvem a integração de dados entre diferentes sistemas e tecnologias de armazenamento, assim cono projetos que necessitam migração de dados e esquemas entre diferentes plataformas de BD e a modernização de sistemas legados.

Hackolade Studio é uma ferramenta poderosa e versátil para modelagem de dados e *design* de esquemas, especialmente adequada para ambientes que utilizam uma variedade de tecnologias de BD, tanto *NoSQL* quanto relacionais. Ele oferece uma ampla gama de funcionalidades que ajudam a aumentar a produtividade, garantir a qualidade dos dados e facilitar a colaboração em equipe.

Contudo, assim como a ferramenta *Moon Modeler* citada na seção 3.1, o *Hackolade Studio* é uma ferramenta voltada para a indústria e não tem uma versão gratuita para fins acadêmicos.

Diferenciais do módulo da ferramenta *brModeloWeb* para modelagem em agregados:

- Roda no navegador e é portátil, permitindo acesso imediato sem instalação.
- É *open source*, favorecendo transparência, contribuição e customização do código.
- Adota o modelo de agregados(que é um modelo intermediário para modelagens orientadas a documentos, família de coluna e chave-valor).
- Elimina custos de licenciamento, atraindo universidades, *startups* e projetos com orçamento restrito.

O módulo *brModeloWeb* prioriza portabilidade, abertura e flexibilidade conceitual com modelo de agregados, sendo ideal para ensino, prototipagem e equipes que precisam customizar e colaborar via *web*. O *Hackolade Studio* oferece uma solução comercial completa com exportadores e recursos prontos para produção, indicada para projetos que demandam conectores e suporte empresarial.

[9] <https://www.couchbase.com/>
[10] <https://neo4j.com/>

3.3 DbSchema

DbSchema é uma ferramenta de *design* de esquemas e gerenciamento de bancos de dados, que oferece uma interface visual para modelagem de dados, documentação e colaboração. A Figura 7 apresenta um exemplo de modelagem para um sistema de vendas modelado para *MongoDB* usando a ferramenta *DbSchema*. Na figura 7, o exemplo *inventory* contém todas as informações sobre o registro de venda, incluindo o *customer*, produtos da venda, local da compra, cupom de desconto e forma de pagamento.

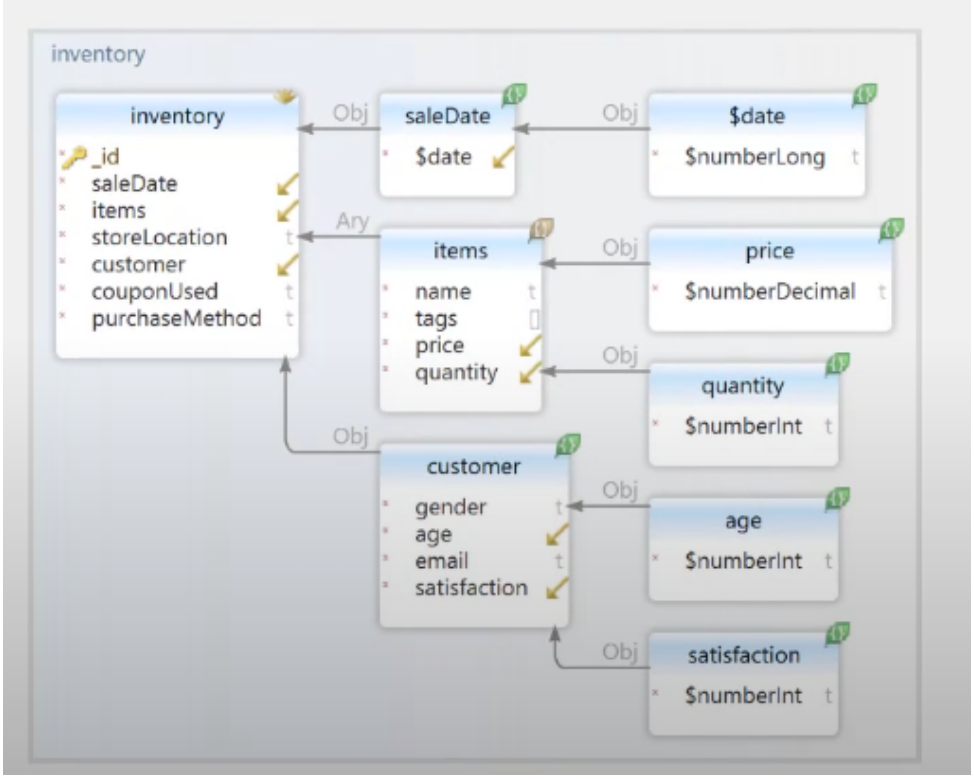

Figura 7 – Exemplo de modelagem de dados para documentos usando a ferramenta DbSchema. Fonte: DbSchema

Ela possui uma interface gráfica para a criação e modificação de esquemas de BD. Também permite a modelagem de dados de forma visual, criando diagramas ER (Entidade-Relacionamento) e diagramas de relacionamento de tabelas, suporta uma ampla gama de bancos de dados, incluindo bancos de dados relacionais como *MySQL*[11], *PostgreSQL*[12], *SQL Server*[13], *Oracle*[14], *SQLite*[15] e bancos de dados *NoSQL* como *MongoDB*, possui geração de *scripts SQL* para criação e atualização de esquemas de BD, sincronização entre a modelagem visual e o BD real, permitindo engenharia reversa para importar esquemas existentes, e inclui ferramentas para explorar e visualizar dados, faci-

[11] <https://www.mysql.com/>
[12] <https://www.postgresql.org/>
[13] <https://www.microsoft.com/pt-br/sql-server/sql-server-downloads>
[14] <https://www.oracle.com/br/>
[15] <https://www.sqlite.org/>

litando a criação de consultas SQL e a visualização de resultados. Também oferece um editor de consultas visual que ajuda a construir consultas SQL de forma intuitiva.

Para bancos *NoSQL*, o *DbSchema* fornece modelagem visual de coleções e documentos, engenharia reversa para inferir esquemas a partir dos dados existentes, um construtor visual de consultas, explorador/ editor de documentos e ferramentas para gestão de índices e importação/exportação de dados. Essas funcionalidades permitem visualizar a estrutura semi-estruturada (JSON/BSON), navegar e editar documentos, e sincronizar a modelagem visual com o banco real, facilitando tanto a análise quanto a manutenção de esquemas flexíveis.

Ela é adequada a diferentes cenários de uso, desde desenvolvimento de aplicações até manutenção de sistemas legados e migração de dados. Ideal para desenvolvedores que precisam modelar, criar e manter esquemas de BD de forma eficiente.

O *DbSchema* é uma ferramenta robusta e versátil para modelagem de dados e gerenciamento de esquemas de BD. Sua interface intuitiva, compatibilidade com diversos bancos de dados e ferramentas avançadas de documentação e colaboração tornam-no uma escolha poderosa para desenvolvedores, *DBAs* (Database Administrator) e equipes de TI (Tecnologia da informação) que trabalham com *design* e manutenção de bancos de dados.

Contudo, é uma ferramenta voltada para a plataforma *desktop*[16], não tem uma versão *web* disponível, o que tira flexibilidade da ferramenta. A ferramenta *DbSchema* oferece uma edição gratuita, porém com limitações relevantes para uso profissional: recursos como *design* lógico independente de SGBD, edição *offline* e salvamento de modelos em arquivo, geração de documentação *HTML5*[17], sincronização de esquemas (com geração de *scripts* de migração), construtor visual de consultas e ferramentas de navegação/gestão de dados não estão disponíveis na versão gratuita.

Em suma, a proposta deste trabalho se diferencia das versões anteriores da *brModelo* e dos trabalhos relacionados por oferecer um módulo inédito da *brModeloWeb* para suportar a modelagem de agregados e também por permitir a modelagem lógica de bancos de dados *NoSQL* em uma ferramenta *open source*,*web* e gratuita.

[16] Computador de mesa.
[17] Linguagem de Marcação de Hipertexto 5.

4 Desenvolvimento do Módulo

4.1 Projeto

O presente trabalho consistiu no desenvolvimento de um módulo para a ferramenta *brModeloWeb*, voltado à modelagem de bancos de dados agregados, com ênfase em bancos *NoSQL*. O objetivo foi permitir a criação de modelagens que contemplam diferentes paradigmas de bancos *NoSQL*, como documentos, família de colunas e chave-valor, proporcionando flexibilidade e recursos avançados para usuários que necessitam modelar bancos de dados não relacionais.
O módulo amplia as capacidades da ferramenta *brModeloWeb*, permitindo a definição de atributos, atributos de referência a coleções, controle de cardinalidade de atributos e coleções, disjunção mútua de coleções e aninhamento entre coleções. Dessa forma, oferece ao usuário uma interface rica e intuitiva para a representação visual de modelos agregados, alinhada com a evolução dos bancos de dados modernos.

4.2 Tecnologias e Ferramentas Utilizadas

O desenvolvimento do módulo fez uso de um conjunto de tecnologias e ferramentas escolhidas por suas características de desempenho, flexibilidade e aderência ao contexto de aplicações *web* modernas. A seguir, detalha-se cada ferramenta e sua importância no projeto.

4.2.1 *JavaScript*

JavaScript é uma das linguagens de programação mais utilizadas no mundo, especialmente no desenvolvimento *web*. Sua principal característica é a execução no lado do cliente (navegador), permitindo a criação de interfaces dinâmicas e interativas sem necessidade de recarregar a página. Além disso, *JavaScript* é altamente versátil, sendo utilizado também em servidores (através de plataformas como *Node.js*[1]). No contexto deste projeto, o *JavaScript* foi fundamental para implementar funcionalidades do *frontend*, manipular elementos da interface e interagir com as bibliotecas utilizadas, proporcionando uma experiência responsiva ao usuário.

[1] É um ambiente de tempo de execução de código aberto e multiplataforma que permite executar código *JavaScript* fora do navegador.

4.2.2 *JointJS*

JointJS é uma biblioteca *JavaScript* especializada para a criação e manipulação de diagramas gráficos interativos. Ela permite modelar visualmente estruturas complexas, como diagramas de bancos de dados, fluxogramas, redes e outros tipos de grafos. Por ser altamente customizável, *JointJS* facilita a integração de elementos personalizados e a implementação de interações intuitivas, como arrastar e soltar, conexões entre objetos e edição visual. No projeto, *JointJS* foi utilizada para oferecer ao usuário uma ferramenta gráfica robusta para modelagem de bancos agregados, tornando o processo mais visual e amigável.

Para os desenhos dos *containers* utilizados na ferramenta, utilizou-se os exemplos da página da ferramenta *JointJS*8, que conta com milhares de exemplos para uso da ferramenta de diagramação.

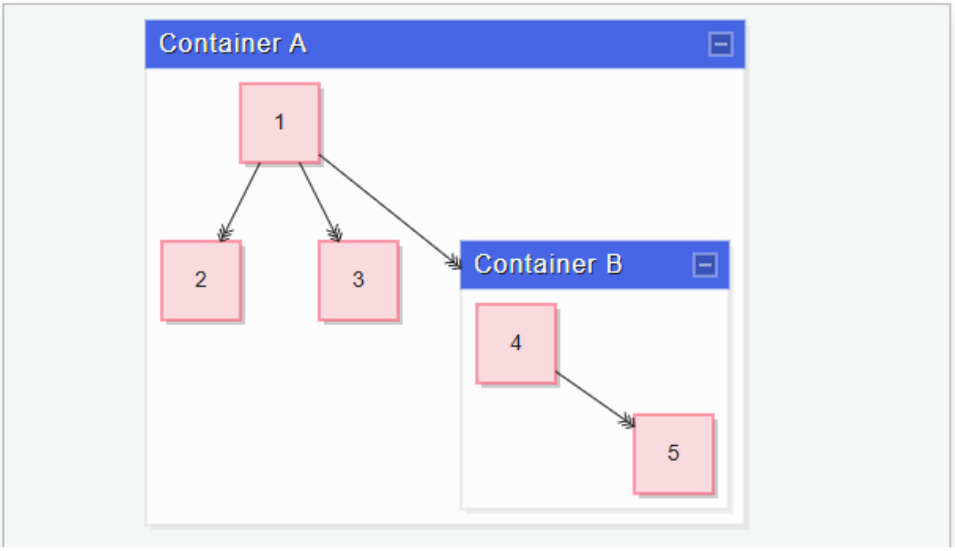

Figura 8 – Imagem de tutorial JointJS Fonte: https://docs.jointjs.com/learn/features/containers-and-grouping/

4.2.3 *AngularJS*

AngularJS é um *framework JavaScript* de código aberto mantido pelo *Google*, projetado para facilitar o desenvolvimento de aplicações *web* de página única *(Single Page Applications – SPA)*. Entre seus principais benefícios, destacam-se a ligação bidirecional de dados *(two-way data binding)*[2], injeção de dependências, modularização do código e roteamento de páginas. No sistema desenvolvido, *AngularJS* foi responsável por estruturar a interface do usuário, organizar a lógica de apresentação e promover interatividade e dinamismo à aplicação. Isso permitiu o desenvolvimento de interfaces ricas, escaláveis e com excelente separação entre lógica e apresentação.

4.2.4 *Node.js*

Node.js é uma plataforma para execução de código *JavaScript* no lado do servidor, baseada no motor V8 do *Google Chrome*. Diferente das abordagens tradicionais, *Node.js* utiliza um modelo assíncrono e orientado a eventos, o que resulta em alta eficiência no processamento de múltiplas requisições simultâneas. *Node.js* é largamente empregado para construir servidores *web*, APIs e aplicações em tempo real. No projeto, *Node.js* foi utilizado para implementar o *backend*, processando requisições dos clientes, realizando autenticação, manipulando dados e orquestrando a integração com o banco de dados. Sua adoção contribuiu para uma arquitetura moderna, escalável e de fácil manutenção.

4.2.5 *MongoDB*

MongoDB é um banco de dados *NoSQL* orientado a documentos, reconhecido por sua flexibilidade na modelagem de dados e escalabilidade horizontal. Em vez de armazenar dados em tabelas e linhas, como bancos de dados relacionais, o *MongoDB* utiliza documentos no formato BSON (uma extensão do JSON), o que facilita o armazenamento de estruturas complexas e aninhadas. Ele é especialmente indicado para aplicações que exigem rapidez em operações de leitura/escrita e não exigem um esquema rígido. No contexto deste projeto, *MongoDB* foi escolhido para persistir os modelos criados e informações associadas à modelagem agregada, permitindo fácil evolução do sistema conforme novas necessidades surgem, sem grandes alterações estruturais.

4.3 Arquitetura do Sistema

A arquitetura do sistema9 segue o modelo cliente-servidor. A interface cliente do usuário foi desenvolvida utilizando *AngularJS*, integrando a biblioteca *JointJS* para a

[2] A "two-way data binding"(ou ligação de dados bidirecional) é um recurso de desenvolvimento que sincroniza automaticamente os dados entre a interface do usuário (UI) e o modelo de dados (lógica) em uma aplicação.

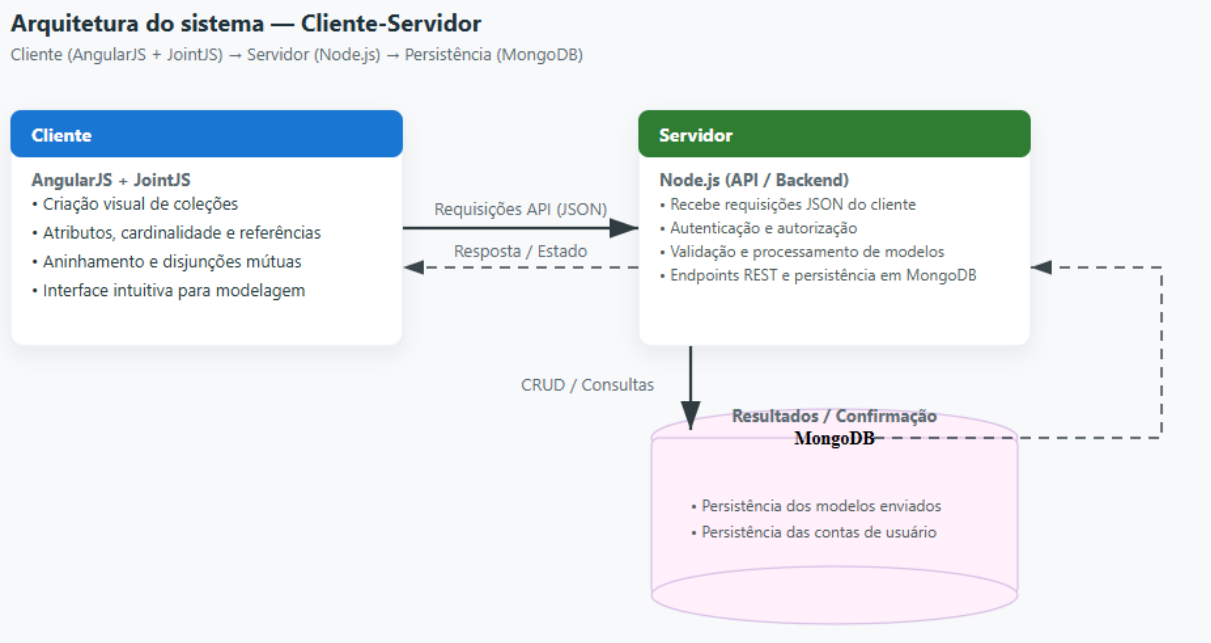

Figura 9 – Arquitetura do Sistema
Fonte: Autor

construção dos diagramas de modelagem. O usuário pode criar coleções, definir atributos (incluindo atributos de referência), especificar cardinalidade de atributos e coleções, configurar disjunções mútuas entre coleções e realizar o aninhamento de coleções de forma visual e intuitiva.

Já o componente servidor (*backend*), implementado em *Node.js*, é responsável por processar as requisições provenientes do cliente, realizar operações de autenticação e persistência dos dados no *MongoDB*, além de fornecer as APIs necessárias para manipulação dos modelos.

4.4 Funcionalidades do Módulo

O módulo desenvolvido oferece um conjunto de funcionalidades para a modelagem de dados no modelo de agregados. Para acessar o módulo, primeiramente o usuário deve fazer *login* e escolher a opção NoSQL no menu principal da ferramenta *brModeloWeb*, como exemplificado na Figura 10. Feito isso, o usuário é direcionado para uma tela de edição dos conceitos do modelo de agregados, que corresponde ao módulo desenvolvido.

As funcionalidade do módulo são as seguintes:

- *Criação de Coleções*: Permite ao usuário adicionar novas coleções ao modelo, com suporte à definição de atributos simples e de referência. A Figura 11 ilustra a definição de uma coleção na interface com o usuário da *brModeloWeb*.

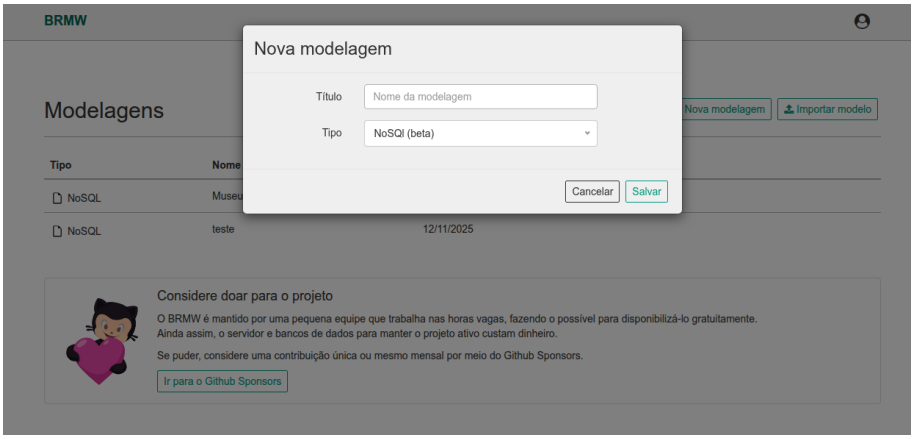

Figura 10 – Menu principal. Fonte: Autor

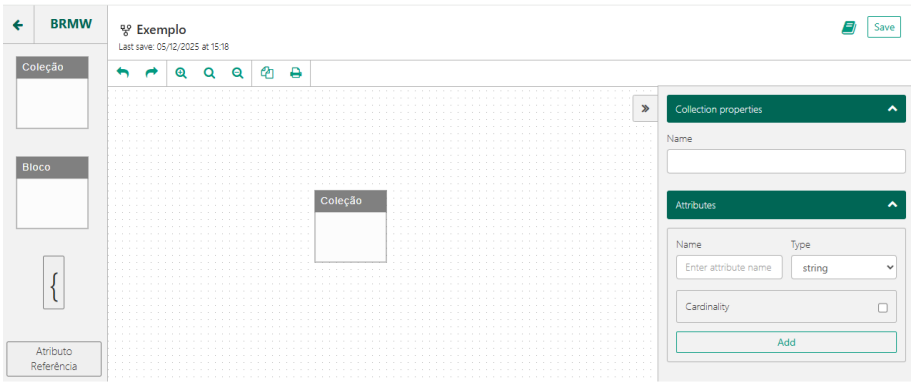

Figura 11 – Criação de uma Coleção. Fonte: Autor

- *Cardinalidade de Atributos e Blocos*: Possibilita a configuração de cardinalidade mínima e máxima tanto para atributos quanto para blocos agregados. A Figura 12 apresenta um exemplo para o caso de um bloco.

- *Restrição de Disjunção de Blocos*: Permite a definição de blocos mutuamente exclusivos dentro de uma coleção ou bloco, conforme ilustra a Figura 13.

- *Aninhamento de Blocos*: Permite a inclusão de blocos dentro de outras coleções ou blocos, modelando estruturas de dados complexas e hierárquicas. A Figura 14 exemplifica essas definições de aninhamento.

O módulo proposto segue o mesmo padrão visual já presente na ferramenta *brModeloWeb*, ou seja, uma interface gráfica intuitiva para a construção, visualização e edição dos esquemas de agregados, facilitando o entendimento e a manutenção dos esquemas criados. Além disso, todos os modelos de dados desenvolvidos podem ser salvos e recuperados posteriormente, garantindo flexibilidade e continuidade no processo de modelagem.

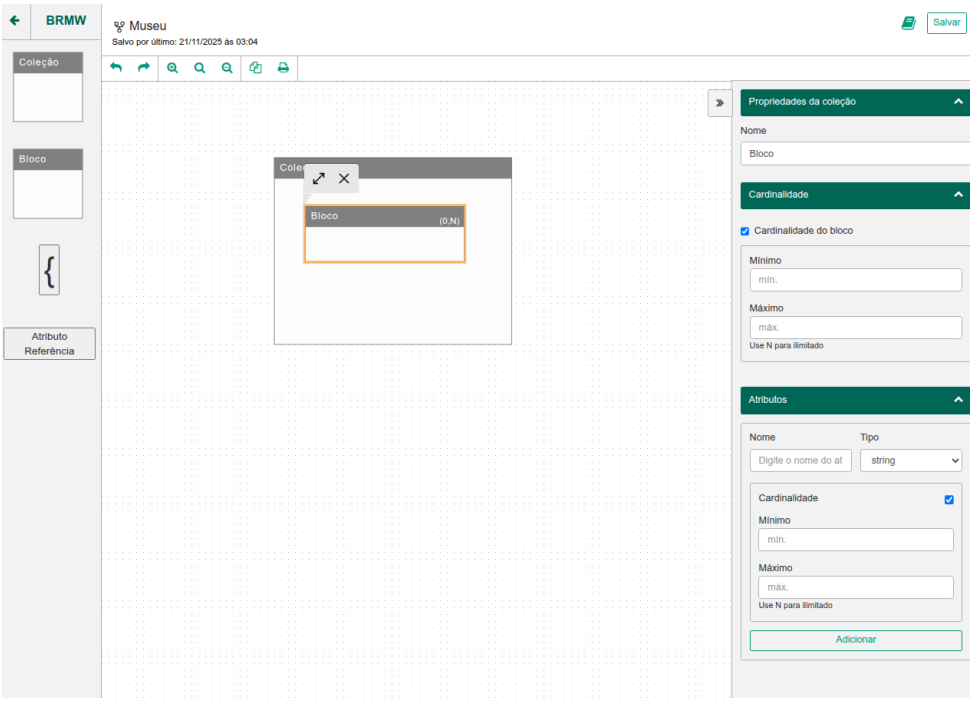

Figura 12 – Cardinalidades de atributo e bloco. Fonte: Autor

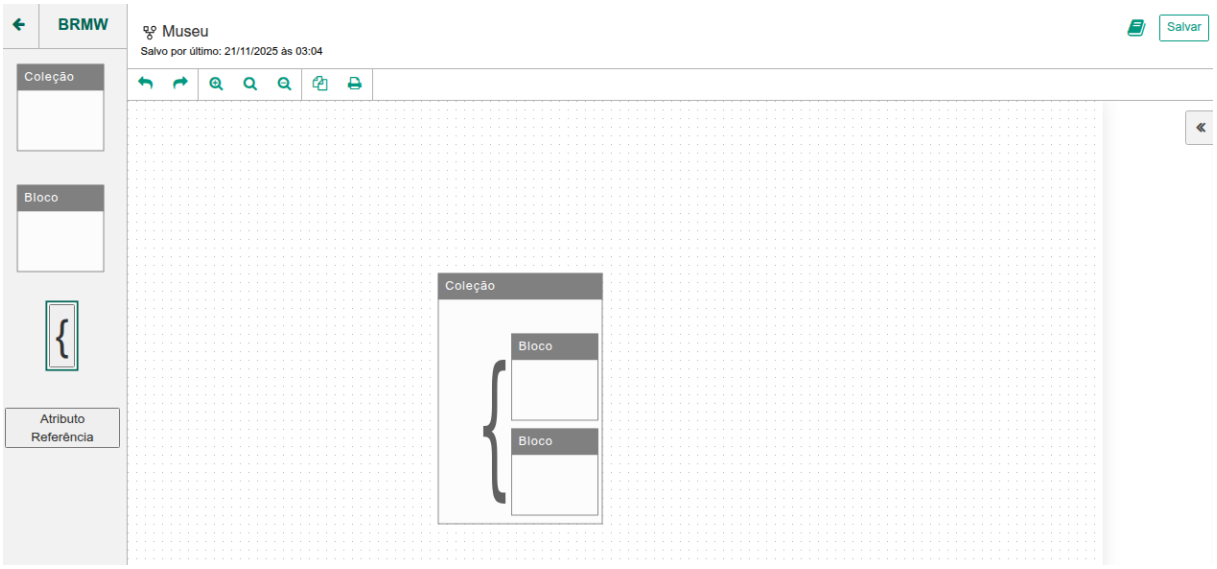

Figura 13 – Restrição de disjunção de blocos. Fonte: Autor

Um botão no canto superior direito permite o salvamento das modelagens, como ilustra a Figura 15.

Dessa forma, o módulo proposto oferece uma solução para a modelagem de dados agregados, tornando a ferramenta *brModeloWeb* ainda mais flexível, abrangente e alinhada às necessidades atuais de desenvolvimento de sistemas que utilizam bancos de dados *NoSQL*.

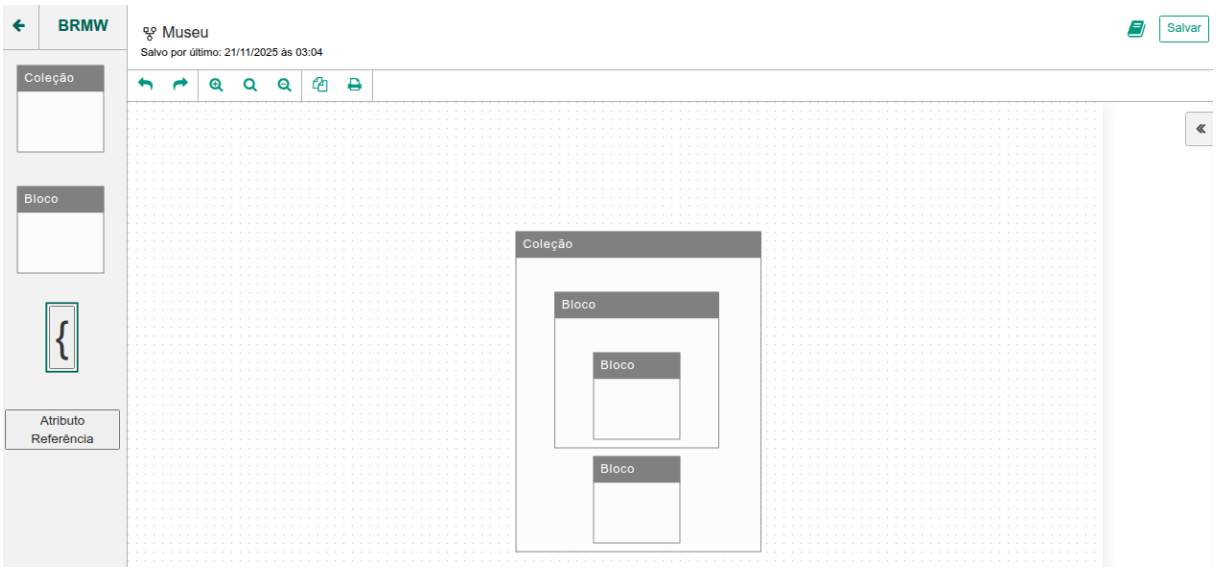

Figura 14 – Aninhamento de Blocos. Fonte: Autor

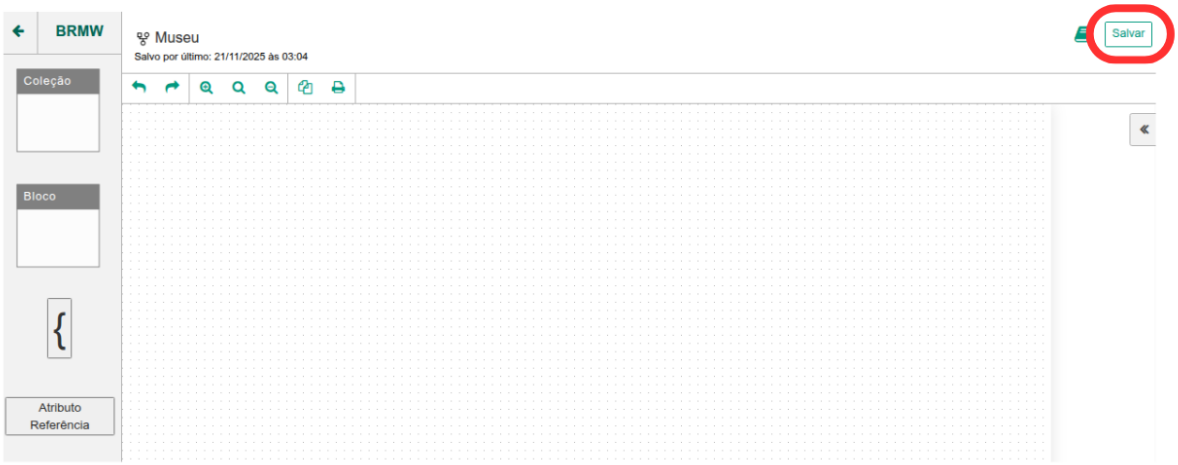

Figura 15 – Persistência de modelagens. Fonte: Autor

5 Estudo de Caso

Este capítulo apresenta um estudo de caso com o objetivo de facilitar a compreensão de utilização do módulo proposto. Inicialmente é apresentada uma descrição dos requisitos de dados de um domínio escolhido, seguido da construção da modelagem de agregados na ferramenta *brModeloWeb*.

5.1 Domínio: Museu

Cada obra no museu possui um código, um título e um ano. Obras são pinturas ou esculturas. No primeiro caso, é importante saber a área (m2) e o tipo (por exemplo, impressionista). No caso das esculturas, elas se especializam em esculturas de pedra ou de metal, podendo uma escultura ser de ambos os materiais. Para qualquer escultura, é importante saber peso e altura. Para esculturas de metal, é importante saber o seu tempo de vida estimado e o tipo de metal. Para esculturas de pedra, é importante saber o tipo de pedra e o país de onde a pedra foi extraída.

Uma obra pode estar exposta em um único salão do museu, em uma posição específica neste salão. Um salão, que geralmente abriga várias obras, é identificado por um número e está em um andar do museu. Certos dados a respeito dos autores de cada obra também devem ser mantidos: código, nome, sexo e nacionalidade (país). Uma obra é produzida por apenas um autor, porém, pode existir mais de uma obra de um mesmo autor no museu.

No museu trabalham restauradores de obras, que possuem id, CPF, nome e salário. Além disso, é necessário saber quais são os seus dias de atendimento no museu durante a semana. Um restaurador pode estar trabalhando em uma única obra, ou seja, realizando sua manutenção. Uma obra, caso esteja em manutenção, está nas mãos de apenas um restaurador. Para cada manutenção deve-se registrar a data de início e a data prevista de término do trabalho, uma descrição do serviço feito e um custo previsto para realizar a manutenção. Uma manutenção pode estar utilizando uma ou mais matérias-primas. Uma matéria-prima possui um código, um nome e uma quantidade em estoque. Uma matéria-prima pode estar sendo utilizada em várias manutenções, em uma certa quantidade. Fornecedores são responsáveis pelo fornecimento de uma ou mais matérias-primas ao museu. Nada impede que a mesma matéria- prima seja fornecida por mais de um fornecedor. Para cada fornecedor, é importante saber CNPJ, nome e endereço (rua, número, complemento, cidade, UF e CEP).

5.2 Modelagem

Para a criação de uma modelagem de agregados na *brModeloWeb*, um usuário deve inicialmente criar uma conta e posteriormente realizar um *login* na ferramenta, como mostra a Figura 16. Na sequência ele seleciona a opção de menu referente à criação de uma modelagem *NoSQL*, como exemplificado no capítulo anterior (Figura 10), para então entrar na interface de criação de esquemas de agregados.

Figura 16 – Estudo de caso: *login*. Fonte: Autor.

Seguidos esses passos, pode-se iniciar a criação de um esquema. Para este estudo de caso, após a leitura do enunciado foi possível identificar as seguintes coleções: *Obra, Restaurador, Matérias-Primas, Fornecedores, Manutenção, Autor* e *Salão*. Para representá-las na ferramenta, arraste os itens disponíveis no menu à esquerda para a área de modelagem. Depois de adicionadas, renomeie as entidades para que façam sentido no contexto do modelo - selecione o objeto e altere o campo NOME no painel à direita.

Um aspecto importante do enunciado é a presença de duas restrições de disjunção: as obras são classificadas como *Pinturas* ou *Esculturas*, e as *Esculturas* podem ser compostas por pedra, metal ou por ambos. Para representar essa restrição na ferramenta, selecione os dois blocos simultaneamente (pressionando *Ctrl* enquanto clica em cada um) e, em seguida, clique no ícone de restrição de disjunção no painel à esquerda. O resultado da modelagem até este momento é mostrado na Figura 17.

Na sequência, o usuário deve identificar e definir os atributos de coleções e blocos. Após a análise do texto, foi possível identificar os seguintes atributos:

- Autor: Código, Nome, Sexo, Nacionalidade, REF_Obra.
- Salão: Numero, Andar, REF_Obra.

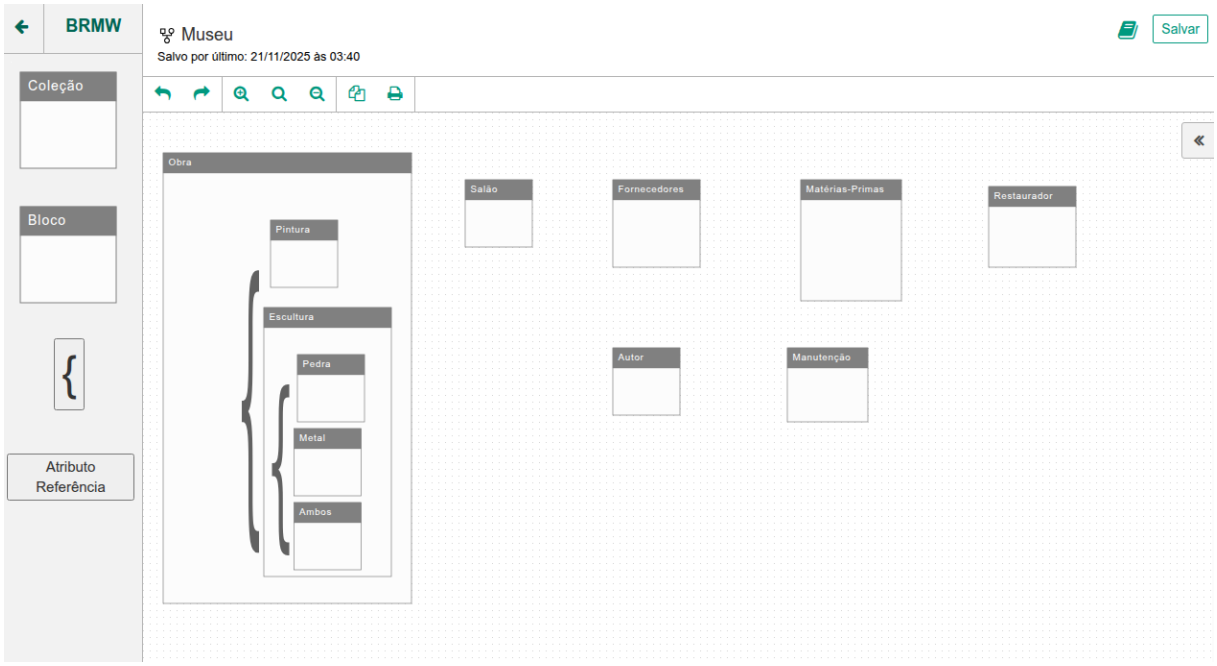

Figura 17 – Estudo de caso: definição das coleções. Fonte: Autor.

- Restaurador: ID, Salário, CPF, Nome, Dias_de_Atendimento e REF_Manutenção.
- Fornecedor: CNPJ, Nome, Rua, Numero, Complemento, CEP, Cidade, UF.
- Endereço: Rua, Número, Complemento, Cidade, UF, CEP.
- Matéria-Prima: Código, Nome, Quantidade_em_Estoque e REF_Fornecedores.
- Obra: Código, Titulo, Ano, REF_Autor, REF_Restaurador.
- Escultura: Peso, Altura.
- Pintura: Área, Tipo.
- Metal: Tipo_de_Metal, Tempo_de_Vida_Estimado.
- Pedra: Tipo_de_Pedra, País_de_Origem.
- Exposição: Posição, REF_Salão.
- Manutenção: REF_Obra, REF_Restaurador, Data_início, Data_termino, Descrição, Custo, REF_Matéria-Prima.

Para criar um atributo em uma coleção ou um bloco, clique sobre ela com o botão esquerdo do *mouse*. O painel lateral à direita exibirá a seção atributos, na qual você pode inserir o nome do atributo e selecionar o seu tipo. Para criar um bloco, basta arrastar o conceito de bloco no menu à esquerda para a área de modelagem. A Figura 18 ilustra a criação de um bloco e de atributos.

Os atributos também podem ter cardinalidade, assim como os blocos. Clicando sobre um bloco, por exemplo, o painel lateral mostra as opções para edição, como exemplificado também na Figura 18.

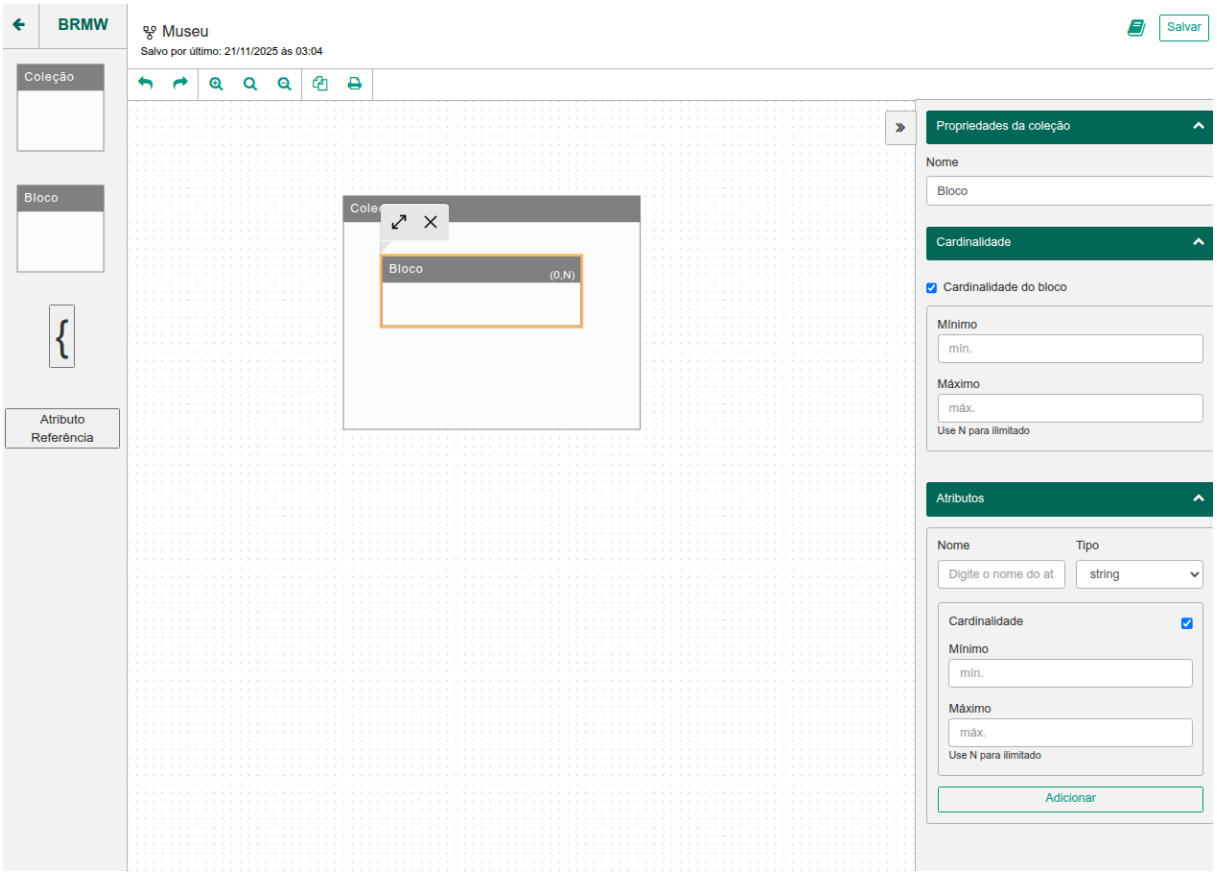

Figura 18 – Estudo de caso: painel lateral. Fonte: Autor.

No modelo diagramático de agregados não existe ligação direta entre coleções, como no modelo relacional. Ao invés disso, as coleções estabelecem relações por meio de atributos de referência. Conforme ilustrado na Figura 19, algumas coleções contêm atributos nomeados segundo o padrão ref_<nome_da_coleção>, que apontam para outra coleção. Para definir um atributo de referência, o usuário deve clicar no botão atributo referência na barra de tarefas localizado na lateral esquerda da interface, e então seguir as instruções que serão exibidas na tela: primeiramente selecionar qual coleção será referenciada e após isso selecionar qual coleção ou bloco receberá o atributo de referência.

Por fim, a Figura 20 apresenta a modelagem final de agregados para o domínio exemplo. Cabe ressaltar que essa não é a única possível modelagem de agregados para este domínio, sendo este resultado uma possível interpretação do domínio feita pelo projetista de dados.

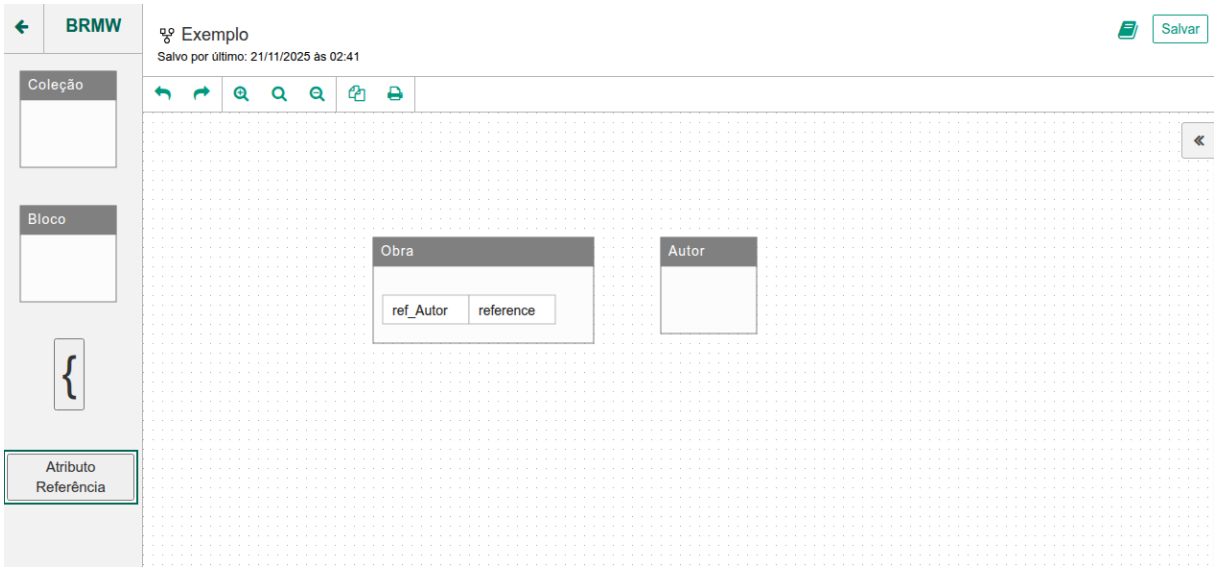

Figura 19 – Estudo de caso: exemplo de atributo de referência. Fonte: Autor.

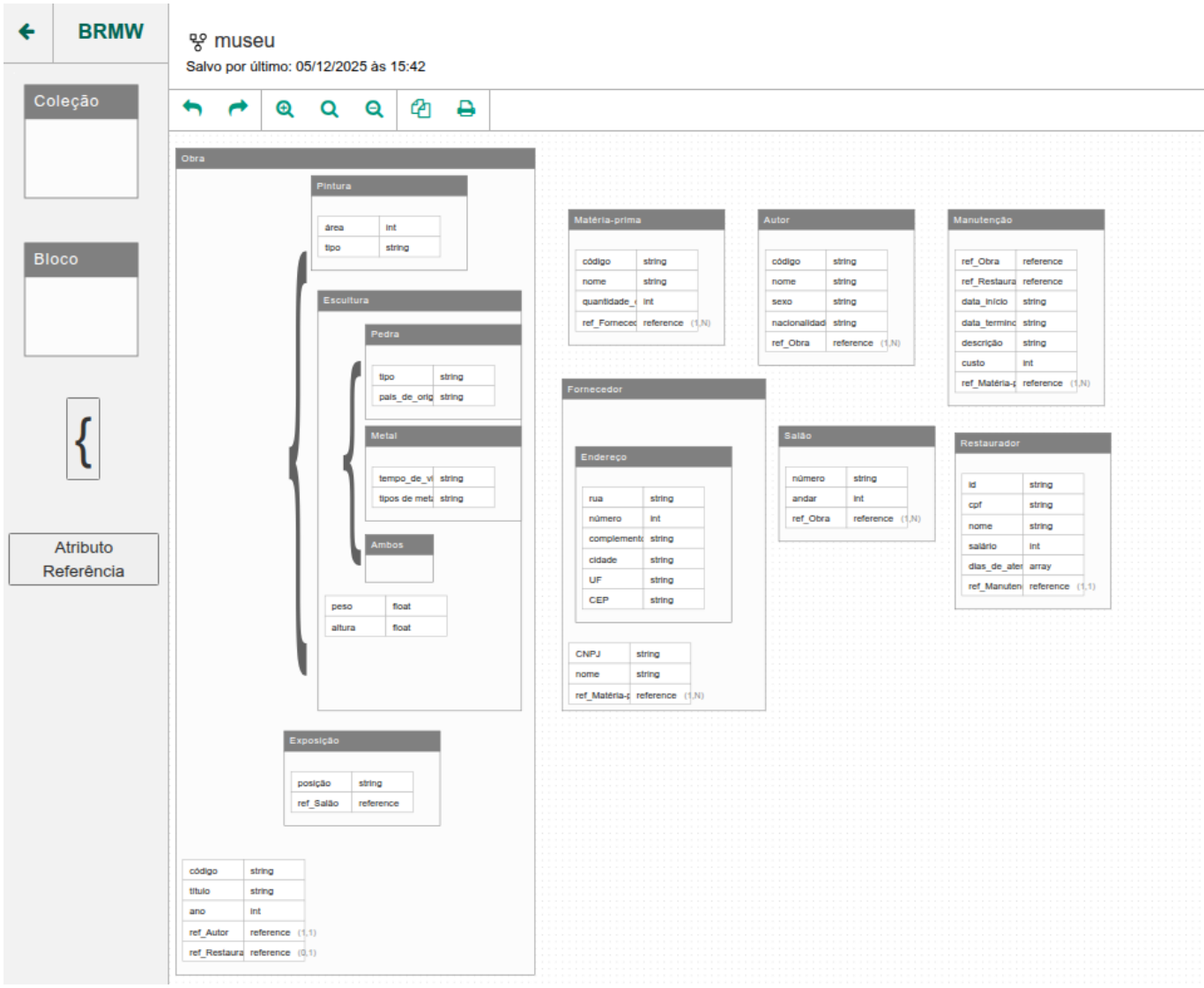

Figura 20 – Estudo de caso: definição dos atributos. Fonte: Autor.

6 Conclusão

Este trabalho propõe e implementa um módulo para a ferramenta *brModeloWeb* voltado à construção de esquema de dados segundo o modelo de dados de agregados. O objetivo principal — proporcionar meios gráficos e interativos para a construção de modelos agregados que suportem diferentes paradigmas *NoSQL* (documentos, família de colunas e chave-valor) — foi atendido por meio da implementação de um conjunto de funcionalidades que ampliam as capacidades da *brModeloWeb*. Foram realizados um levantamento bibliográfico e análise de trabalhos relacionados para embasar os requisitos e o projeto da solução. O estudo de caso (domínio Museu) confirmou que o protótipo funciona adequadamente em cenários reais, validando sua aplicabilidade prática.

Algumas atividades imediatas relacionadas a este trabalho devem ser consideradas em trabalhos futuros, como por exemplo:

- Realizar avaliação de usuário (usabilidade) para medir eficácia, eficiência e satisfação do módulo proposto, incluindo a realização de tarefas de modelagem e o preenchimento de um questionário pós-avaliação;
- Proceder eventuais correções de erros detectados e melhorias de usabilidade no módulo a partir da avaliação dos usuários;
- Produzir uma documentação de uso do módulo no *GitHub* da ferramenta *brModeloWeb*;
- Conectar este módulo a outros módulos da ferramenta responsáveis por outras etapas do projeto de bancos de dados *NoSQL*.
- Receber diagramas ER e convertê-los em modelagens de agregados.
- Garantir a conversão dos modelos de agregados para os três tipos de bancos *NoSQL*: orientado a documentos, chave-valor e família de colunas.

Referências

BARBOSA, M. S. BANCO DE DADOS NoSQL: uma alternativa para grandes empresas. In: *Anais do Simpósio Brasileiro de Banco de Dados (SBBD)*. [S.l.: s.n.], 2013.

CÂNDIDO, C. H. *Aprendizagem em banco de dados: implementação de ferramenta de modelagem E.R.* Monografia — Pós-Graduação em Banco de Dados, Universidade Federal de Santa Catarina - UFSC, Universidade de Várzea Grande - UNIVAG, Várzea Grande-MT, 2005. Especialização em Banco de Dados.

CATTELL, R. Scalable sql and nosql data stores. *Acm Sigmod Record*, ACM New York, NY, USA, v. 39, n. 4, p. 12–27, 2011.

EVANS, E. *Domain-driven design: tackling complexity in the heart of software*. [S.l.]: Addison-Wesley Professional, 2004.

HAN, J. et al. Survey on nosql database. In: IEEE. *2011 6th international conference on pervasive computing and applications*. [S.l.], 2011. p. 363–366.

HASHEM, H.; RANC, D. Evaluating nosql document oriented data model. In: IEEE. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. [S.l.], 2016. p. 51–56.

HEUSER, C. *Projeto de Banco de Dados: Volume 4*. [S.l.]: Bookman, 2009.

INDRAWAN-SANTIAGO, M. Database research: Are we at a crossroad? reflection on nosql. In: IEEE. *2012 15th International Conference on Network-Based Information Systems*. [S.l.], 2012. p. 45–51.

LIMA, C. de; MELLO, R. dos S. A workload-driven logical design approach for nosql document databases. In: *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*. [S.l.: s.n.], 2015. p. 1–10.

MENNA, O. S.; RAMOS, L. A.; MELLO, R. D. S. BrModeloNext: Nova Versão de uma Ferramenta para Modelagem de Bancos de Dados Relacionais. In: *Anais do 16º Simpósio Brasileiro de Banco de Dados (SBBD)*. Florianópolis: S.e., 2011. p. 1–7.

RUIZ, D. S.; MORALES, S. F.; MOLINA, J. G. Inferring versioned schemas from nosql databases and its applications. In: SPRINGER. *Conceptual Modeling: 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings 34*. [S.l.], 2015. p. 467–480.

SADALAGE, P. J.; FOWLER, M. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. [S.l.]: Pearson Education, 2013.

TUDORICA, B. G.; BUCUR, C. A comparison between several nosql databases with comments and notes. In: IEEE. *2011 RoEduNet international conference 10th edition: Networking in education and research*. [S.l.], 2011. p. 1–5.

Apêndice A – Artigo Resumido

Chapter 1

Incorporação do Modelo de Agregados na ferramenta brModeloWeb

Felipe Cruz Luiz

Abstract

This paper presents the development of a module for brModeloWeb that enables interactive modeling of aggregate schemas for NoSQL databases. It summarizes motivation, architecture and a museum-domain case study validating the prototype. The module supports collections, nested blocks, attributes (including references), cardinalities, disjunctions and model persistence.

Resumo

Este artigo apresenta o desenvolvimento de um módulo para a brModeloWeb que permite a modelagem interativa de esquemas de agregados para bancos de dados NoSQL. Sintetiza a motivação, a arquitetura e um estudo de caso no domínio de museu que valida o protótipo. O módulo suporta coleções, blocos aninhados, atributos (incluindo referências), cardinalidades, disjunções e persistência dos modelos.

1.1. Introdução

Bancos de dados *NoSQL* oferecem flexibilidade e escalabilidade, mas frequentemente carecem de representações formais e visuais de esquema que facilitem o projeto colaborativo e a manutenção. A modelagem por agregados organiza dados complexos em unidades lógicas (coleções e blocos aninhados) e é um um modelo intermediário para modelos orientados a documentos, família de colunas e chave-valor. Discussões sobre o ecossistema *NoSQL* e *trade-offs* entre consistência e disponibilidade podem ser encontradas em trabalhos clássicos [Cattell 2011, Sadalage and Fowler 2013], enquanto notações gráficas e formalizações para agregados são abordadas em trabalhos mais recentes [de Lima and dos Santos Mello 2015].

1.2. Metodologia

Adotou-se um ciclo incremental: levantamento bibliográfico, análise de ferramentas relacionadas *(Moon Modeler, Hackolade, DbSchema)*, especificação arquitetural e implementação *(frontend AngularJS + JointJS, backend Node.js;* e persistência em *MongoDB*). O protótipo foi validado com um estudo de caso no domínio de um museu.

1.3. Contribuições

O trabalho entrega:

- extensão da *brModeloWeb* para modelagem por agregados;
- interface gráfica para criação/edição de coleções, blocos e atributos (incluindo atributos de referência);
- suporte a cardinalidades, disjunções e aninhamentos;
- persistência dos modelos em um banco de dados MongoDB.

1.4. Estudo de Caso

No domínio Museu foi modelado coleções como Obra, Autor, Salão, Restaurador, Matéria-Prima, Fornecedor e Manutenção, incluindo especializações Pintura/Escultura, blocos Metal/Pedra e atributos de referência. O protótipo permitiu a modelagem visual e o salvamento do modelo na plataforma.

1.5. Conclusão

A extensão amplia a *brModeloWeb* para suportar práticas de modelagem *NoSQL* em ambiente *web e open-source*. Trabalhos futuros incluem avaliação de usabilidade, documentação pública e conectar este módulo a outros módulos da ferramenta responsáveis por outras etapas do projeto de bancos de dados *NoSQL*.

1.6. Referências

Cattell, R. (2011). Scalable sql and nosql data stores. *Acm Sigmod Record*, 39(4):12–27.

de Lima, C. and dos Santos Mello, R. (2015). A workload-driven logical design approach for nosql document databases. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, pages 1–10.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education.

Apêndice B – Código da Aplicação

Repositório com o código da aplicação:

https://codigos.ufsc.br/felipe.cl/brmodelo-app

Descrição: repositório contendo frontend, backend, instruções de instalação e exemplos de execução.