



Estácio

Centro Universitário Estácio do Ceará

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Igor Gomes de Moisés

**ESTRATÉGIA DE SOFTWARE NA TECNOLOGIA JAVA:
AUTOMAÇÃO DE TRANSAÇÕES EM SGBD'S RELACIONAIS**

FORTALEZA
2017

Centro Universitário Estácio do Ceará

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

**ESTRATÉGIA DE SOFTWARE NA TECNOLOGIA JAVA:
AUTOMAÇÃO DE TRANSAÇÕES EM SGBD'S RELACIONAIS**

Trabalho de Conclusão de Curso (TCC)
apresentado ao Centro Universitário
Estácio do Ceará como requisito parcial
para a conclusão do curso de Bacharelado
em Sistemas de Informação.

Orientador: Francisco Alves Carneiro

Coorientadora: Profa. Dra. Leticia Adriana
Pires Ferreira dos Santos

FORTALEZA

2017

IGOR GOMES DE MOISÉS

**ESTRATÉGIA DE SOFTWARE NA TECNOLOGIA JAVA: AUTOMAÇÃO DE
TRANSAÇÕES EM SGBD'S RELACIONAIS**

Trabalho de Conclusão de Curso (TCC)
apresentado ao Centro Universitário
Estácio do Ceará como requisito parcial
para a conclusão do curso de Bacharelado
em Sistemas de Informação.

Orientador: Francisco Alves Carneiro

Coorientadora: Profa. Dra. Leticia Adriana
Pires Ferreira dos Santos

Aprovada em ____/____/____.

BANCA EXAMINADORA

Prof Francisco Alves Carneiro (Orientador)

Centro Universitário Estácio do Ceará (Estácio - FIC)

Profa Dra Leticia Adriana Pires Ferreira dos Santos(Coorientadora)

Centro Universitário Estácio do Ceará (Estácio - FIC)

Terceiro professor da banca

Centro Universitário Estácio do Ceará (Estácio - FIC)

Aos meu pais João H lio e Silmia.

A minha av  Maria Neri Silva.

AGRADECIMENTOS

Agradeço, primeiramente, a minha falecida avó Maria Neri Silva, que com o ar de sua bondade me objetivou para que eu estudasse e escolhesse esta área de atuação profissional.

Aos meu falecidos avos José Moisés e Marcelina, a minha falecida tia e mãe por consideração Celi Melo, que sempre me apoiaram e me incentivaram.

Agradeço aos meus pais, João Hélio e Silmia, que me apoiam todos os dias, seja na vida pessoal e ou profissional, fazendo com que eu galgue sempre novos desafios.

A minha namorada, Joselane Lima, que me apoiou e me incentivou na criação deste trabalho.

Aos meus colegas de faculdade e profissão que participaram da pesquisa.

Ao meu amigo Fábio Rodrigues Franco Júnior que me auxiliou no desenvolvimento da ferramenta apresentada neste documento.

Ao meu orientador Prof. Francisco Alves Carneiro, pelo suporte analítico prestado, culminando em correções e incentivos

A minha coorientadora Profa. Dra. Leticia Adriana Pires Ferreira dos Santos, que por meio da relevância do seu trabalho científico, possibilitou-me no pouco tempo que lhe coube, a realização deste trabalho.

Aos professoras que, gentilmente, aceitaram participar da banca de defesa do TCC.

E a todos que fizeram parte da minha formação, direta ou indiretamente, deixo aqui os meus mais sinceros votos de agradecimento.

“Especificar requisitos detalhadamente de modo que uma máquina possa executá-los é *programar* – e tal especificação é o *código*” (*Robert C. Martin, 2008, p.2*)

RESUMO

Bancos de dados representam a base de armazenamento de informações de todo sistema. Estruturados logicamente são conjuntos de dados que, quando agrupados, definem um escopo único de caráter especializado e eficiente para consultas a informações. Para o gerenciamento dessa tecnologia foram criados os SGBD, Sistemas Gerenciadores de Bancos de Dados. Estes são softwares responsáveis pelo controle do banco de dados, assim como pela atuação de interface de entrada e saída de demandas. A tecnologia Java, possui um conjunto de interfaces únicas para a realização da integração com SGBD's (AUTOR, ANO). Uma das etapas do trabalho de desenvolvimento de todo *software* é a montagem das estruturas de integração e realização de transações da tecnologia com o banco de dados, a qual termina por demandar uma certa quantidade de tempo do trabalho de programação. O objetivo deste trabalho é, portanto, apresentar a ferramenta SigmaDB. Um novo Framework capaz de realizar a integração da tecnologia JAVA com qualquer SGBD relacional, buscando trazer agilidade de desenvolvimento, centralização da estrutura de códigos, boas práticas de desenvolvimento, assim como o controle e manutenção das tabelas do banco de dados. Utilizando-se dos conceitos avançados da tecnologia Java, a ferramenta foi desenvolvida baseada nas estruturas de *generics*, que lhe permite a manipulação de qualquer tipo de objeto, e da estrutura de reflexão, que lhe dá o poder de quebrar a regra de encapsulamento da linguagem de programação. Desta forma, ao receber qualquer objeto, se este seguir a convenção de *getters* e *setters*, a ferramenta será capaz de manipulá-lo da forma como ela achar necessário. Diante disto, a SigmaDB é capaz de converter o paradigma de Orientação a Objetos em consultas a bancos de dados, trazendo os benefícios citados.

Palavras-chave: Banco de dados. Frameworks. JAVA. Tuplas.

ABSTRACT

Databases represent the basis for storing system-wide information. Logically structured are datasets that, when grouped together, define a unique, specialized and efficient scope for querying information. For the management of this technology, DBMSs were created, Database Management Systems. These are software that is responsible for the control of the database, as well as for the interaction of input and output interface of demands (AUTHOR, YEAR). Java technology has a set of unique interfaces for integrating with DBMS (AUTHOR, YEAR). One of the steps in the development of all software is the assembly of the integration and transaction structures of the technology with the database, which ends up requiring a certain amount of programming time. The objective of this work is, therefore, to present the SigmaDB tool. A new framework capable of integrating JAVA technology with any relational DBMS, aiming to bring development agility, centralization of code structure, good development practices, as well as control and maintenance of database tables. Using the advanced concepts of Java technology, the tool was developed based on the structures of generics, which allows it to manipulate any type of object, and the reflection structure, which gives it the power to break the language encapsulation rule Of programming. In this way, upon receiving any object, if it follows the convention of getters and setters, the tool will be able to manipulate it in whatever way it deems necessary. In view of this, SigmaDB is able to convert the Object Orientation paradigm into database queries, bringing the mentioned benefits.

Keywords: Database, Frameworks, Java, Tuples

Sumário

1 INTRODUÇÃO.....	2
2 PROPOSTA DE TRABALHO.....	4
2.1 Metodologia de Trabalho.....	6
2.2 Precisão de Alocação de Recursos.....	6
2.3. Cronograma de trabalho.....	7
3 A EMPRESA E O NEGÓCIO.....	7
3.1 Histórico da empresa.....	7
3.2 Atividades da empresa.....	8
3.3 Organograma.....	8
3.4 Mercado consumidor.....	8
3.5 Concorrência.....	9
3.6 Premissas e Restrições ao projeto.....	9
4 O SISTEMA ATUAL.....	12
4.1 Justificativa de Escolha do Sistema.....	13
4.1.1 O Sistema.....	13
4.1.2 Funcionamento do Sistema.....	13
4.1.2.1 Configurando o Framework.....	13
4.1.2.2 Definição de padrão de utilização.....	14
4.1.2.3 Tratamento de tabelas.....	15
4.1.2.4 Consultando e persistindo dados.....	16
4.1.3 O Ambiente do Sistema.....	17
4.1.4 A definição do escopo.....	17
4.2 Motivação para o novo sistema.....	18
4.3 Situação desejada.....	18
4.4 Problemas no sistema atual.....	18
5. O Sistema Proposto.....	18

5.1 Lista de Requisitos do Sistema.....	18
5.1.1 Requisitos Funcionais.....	18
5.1.2 Requisitos Não Funcionais.....	20
5.2 Diagrama de Caso de Uso.....	21
5.3 Especificações dos casos de uso.....	22
5.3.1 Configura ferramenta.....	22
5.3.2 Persistência de dados.....	22
5.3.3 Pesquisa de dados.....	23
5.3.4 Persistência de logs.....	25
5.3.5 Pesquisa de logs.....	26
5.4 Modelo Conceitual de Classes.....	27
5.5 Modelo Conceitual de Dados.....	27
6. Conclusões.....	27
7. Referências Bibliográficas.....	28

1 INTRODUÇÃO

“Um sistema de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas que permitem aos usuários acessar e modificar esses dados”. (SILBERSCHATZ, 2006, p.2).

Além dos conjuntos de programas que compõe um banco de dados, esse também pode ser mantido por meio de outras aplicações, denominadas de SGBD (Sistemas Gerenciadores de Bancos de dados).

“Um sistema gerenciador de banco de dados (SGBD – *Database Management System*) é uma coleção de programas que permite aos usuários criar e manter um banco de dados”. (ELMASRE, 2011, p.3).

Segundo Deitel (2010), um sistema de gerenciamento de banco de dados fornece mecanismos para armazenar, organizar, pesquisar e alterar informações para muitos usuários. Os SGBD's disponibilizam acesso de dados sem que a representação interna desses sejam expostas para os usuários.

Elmasre (2011) aponta que bancos de dados desempenham um papel crítico em quase todas as áreas que utilizam-se de computadores para o controle de suas necessidades. Os softwares, também em sua maioria, necessitam do banco de dados para que possam passar a armazenar de forma eficiente e segura todas as informações que lhe são pertinentes.

O desenvolvimento de um *software* se dá por meio da escrita de códigos fonte numa determinada linguagem de programação. Uma das linguagens de programação mais difundidas e utilizadas atualmente no mercado de trabalho é o Java. Segundo o site *Business Insider*, a linguagem de programação Java no ano de 2016 ficou com a segunda colocação de tecnologia de desenvolvimento mais utilizada no mundo, perdendo apenas para o JavaScript.

Segundo sua documentação, o Java possui uma base bem estruturada para o desenvolvimento de praticamente todos os tipos de aplicações em rede, assim como é o padrão global para o desenvolvimento e distribuição de aplicações móveis, jogos, conteúdo baseado na Web e softwares corporativos. Isso deve-se ao fato da sua arquitetura ser mantida por uma máquina virtual chamada de JVM (*Java Virtual Machine*) e muitas APIs de controle de desenvolvimento.

A JVM traz uma característica única ao Java que é sua portabilidade sistêmica. Ela é um programa que carrega e executa aplicações Java, traduzindo *bytecodes* em código executável. Popek e Goldberg (1974) nos diz que uma máquina virtual é uma cópia isolada de uma máquina real. Exatamente por conta desta característica o Java torna-se escalável suficiente para ser executado em qualquer sistema operacional.

No desenvolvimento de aplicações corporativas, existe a necessidade de se realizar a integração da tecnologia utilizada para o desenvolvimento com algum SGBD, tendo em vista que esse é a forma mais segura e otimizada para o armazenamento das informações do sistema a ser trabalhado.

Tomando como base a tecnologia Java, existem dois protocolos distintos que servem para a realizar a integração do Java com SGBDs. Estes protocolos são ODBC (*Open Database Connectivity*), um padrão de acesso para várias linguagens de programação por meio de *drivers* que se transforma numa camada de tradução entre a linguagem de programação e o SGBD, e o JDBC (*Java Database Connectivity*), que é uma API Java que permite que aplicações conectem-se a um banco de dados, sendo esse o protocolo o mais utilizado pela comunidade para essa finalidade.

Por meio de ambos os protocolos, a tecnologia Java possui estruturas de códigos, denominadas de classes e *interfaces*, que são disponibilizadas em suas APIs nativas, a saber o pacote `java.sql.*`, que são responsáveis pela comunicação da linguagem com um banco de dados.

O conhecimento adquirido durante o tempo de oito anos de trabalho profissional com a tecnologia Java, me permitiu projetar e desenvolver uma ferramenta capaz de automatizar transações da tecnologia Java com o banco de dados, assim como centralizar todas as conexões num único ponto de acesso.

Desenvolvida inicialmente para atender demandas de uma startup de minha propriedade, essa ferramenta foi desenvolvida tomando como base os conceitos de reflexão, que tem o papel de realizar a quebra de encapsulamento da linguagem de programação e da estrutura de *generics* da API Java, com o objetivo de conseguir tratar qualquer conteúdo a ser trabalhado pelo programador. Dessa forma, fui capaz de desenvolver uma estrutura de centralização de código, criando uma área de alta coesão que pode ser reutilizada em qualquer projeto.

Do ponto de vista do trabalho de programação, a parte inicial de um projeto demanda uma determinada quantidade de tempo para a construção das estruturas de integração da aplicação que está sendo trabalhada com sua respectiva base de dados. O software apresentado neste documento tem como objetivo fazer com que esta etapa do projeto esteja automaticamente concluída, uma vez que a ferramenta proposta é uma ponte que interliga o Java a qualquer banco de dados. Possui também o intuito de buscar trazer coesão aos códigos do projeto, uma vez que a ferramenta será a controladora de qualquer transação com o banco.

Este trabalho inicia-se com uma breve apresentação sobre seu escopo por meio de uma introdução, localizada no capítulo um deste trabalho. Essa tem como objetivo principal dar iniciação ao objetivo principal que é a ligação de bancos de dados com a tecnologia Java.

Dada a introdução, o próximo ponto é a narrativa de apresentação da ferramenta, visando apontar sua finalidade, assim como seus objetivos. Encontraremos também a forma pela qual todo o conteúdo aqui apresentado foi trabalhado, incluindo toda a alocação de recursos necessárias e os métodos de pesquisa utilizados para a elaboração deste documento. Todas essas características serão apresentadas por meio da proposta de trabalho, que poderá ser encontrada no capítulo dois do documento.

Conforme já mencionado, a ferramenta foi criada com o propósito de suprir a necessidade de uma startup criada pelo autor deste trabalho. No terceiro capítulo será apresentada uma visão da empresa detentora da startup, apresentado seu histórico, mercado, listagem de serviços, estrutura funcional e como esta ferramenta se encaixa e traz melhorias para os objetivos da organização.

O quarto capítulo destina-se a apresentar o funcionamento da ferramenta. Apresenta a parte técnica do documento, mostrando como de fato o programador deve realizar seu trabalho com a ferramenta.

No quinto capítulo encontra-se como o sistema foi modelado. Apresenta toda a análise de projeto realizada para a construção do produto.

2 PROPOSTA DE TRABALHO

Para o desenvolvimento de qualquer aplicação se faz necessário a existência de uma camada de armazenamento de dados em disco. Atualmente, a área da computação considera que a utilização de bancos de dados é a maneira mais segura para a realização do armazenamento de informações.

Segundo Elmasre (2010), banco de dados é uma cadeia de informações relacionadas. Essas informações são fatos conhecidos que podem ser registrados e possuem significado implícito. Para o trabalho com banco de dados existem os SGBDs (Sistemas gerenciadores de bancos de dados), que são *softwares* que possuem recursos capazes de manipular as informações do banco de dados e interagir com usuários. Assim sendo, podemos citar como exemplo de SGBDs os *softwares* Oracle, PostgreSQL, MySQL, SQL Server, MongoDB, entre outros.

Bancos de dados possuem modelos lógicos de implementação. O mais difundido no mercado atualmente é o modelo relacional, inventado por Edgar Fran Codd (1970) e mantido por Chris Date e HugDarwen como um modelo geral de dados. Esse modelo é considerado o mais seguro e mais rápido, respectivamente, ao fato de que o modelo pode ser estendido com características de orientação a objeto e por se trabalhar, baseado num modelo matemático.

Este modelo é baseado no conceito de tabelas, onde cada tabela possui suas respectivas linhas, denominadas como tuplas, e colunas, denominadas como atributos. Todo Sistema Gerenciador de Banco de Dados, que implementa este modelo, deve possuir uma estrutura capaz de aceitar determinados tipos de comandos para o gerenciamento do banco de dados. Esses comandos são definidos por meio de uma linguagem denominada de SQL e segue agrupamentos definidos como *DDL (Data Definition Language)*, *DML (Data Manipulation Language)*, *DCL (Data Control Language)* e *TCL (Transaction Control Language)*.

Navathe (2010) afirma que o nome SQL é uma sigla para *Structured Query Language* (Linguagem de Consulta Estruturada) e que esta pode ser considerada um dos principais motivos para o sucesso dos bancos de dados relacionais comerciais.

Toda tecnologia de linguagem de programação de alto nível possui sua interface de conexão com bancos de dados, mesmo que esta tenha sua implementação auxiliada pelo próprio sistema operacional.

Tratamos, em específico, neste trabalho sobre a tecnologia JAVA, tendo em vista que a ferramenta de trabalho aqui proposta foi modelada e implementada nesta plataforma.

Para a realização da conexão do SGBD com a tecnologia JAVA, existem dois protocolos distintos de padrões de acesso que podem ser seguidos. O profissional da área de programação pode optar pelo protocolo ODBC (*Open Database Connectivity*), um padrão de acesso para várias linguagens de programação por meio de *drivers* que se transforma numa camada de tradução entre a linguagem de programação e o SGBD. Em paralelo, o profissional da área de programação também pode optar pelo protocolo JDBC, um conjunto de códigos JAVA responsável por realizar a comunicação da linguagem com qualquer SGBD que tenha o modelo lógico Relacional.

A linguagem de programação JAVA necessita de classes especiais para a realização da conexão com o SGBD. Essas classes especiais encontram-se acopladas no pacote `java.sql.*` de sua API pública. O desenvolvimento lógico para tratar da conexão da tecnologia com o SGBD geralmente envolve uma longa estrutura de código, o que sempre acaba demandando uma determinada quantidade de tempo nos projetos.

Com o intuito de evitar o retrabalho dos desenvolvedores, de tornar mais rápidas as implementações de código, de buscar trazer melhor entendimento do conceito lógico do sistema, de unificar soluções e de dar coesão aos códigos fontes, grupos de desenvolvimento de código fonte livre passaram a implementar ferramentas que automatizam conexões e comandos SQL ao SGBD. A essas ferramentas, generalizamos, chamando-as de *Frameworks* de bancos de dados. Podemos citar, como exemplo, a ferramenta Hibernate e JPA, que são ferramentas mundialmente difundidas para integração da tecnologia JAVA com banco de dados.

A ideia deste trabalho é, assim, apresentar uma ferramenta desenvolvida pelo autor, que tem o papel de realizar a conexão da linguagem de programação JAVA com qualquer banco de dados, realizando a execução de qualquer requisição ao banco, sem que seja necessário a escrita de comandos SQL.

Por meio do paradigma de Orientação a Objeto, a ferramenta é capaz de interpretar qualquer tipo de requisição embutidas em objetos e destiná-la em forma de uma transação ao SGBD, o qual irá processar a respectiva requisição e respondê-la a ferramenta, que por consequente entregará o retorno ao solicitante. Não somente isto, a ferramenta também é capaz de executar comandos SQL, obtendo-se o mesmo comportamento descrito anteriormente.

Tomando como base a aplicação desta ferramenta numa empresa, podemos observar os seguintes benefícios promovidos por uma única estrutura :

- **Conexão automática com qualquer banco de dados** – A ferramenta é capaz de realizar a conexão da linguagem JAVA com qualquer banco de dados, por intermédio das bibliotecas configuradas no projeto, e por meio de um arquivo de propriedades que deve ser entregue a ela. Neste arquivo deverá encontrar-se as informações necessárias para conexão com o banco. A saber, login, senha, driver e url do banco;
- **Único ponto de entrada e saída e remoção de dados** – A estrutura da ferramenta oferece uma estrutura única para a realização de comandos de persistência de

dados no Banco, assim como também, respectivamente, para pesquisa de dados do banco. Isso faz com que não exista redundância de código na implementação de consultas e persistências;

- **Não necessita de SQL** – Através de uma estrutura genérica da O.O, baseada na metodologia de reflexão para quebra de encapsulamento de objetos, a ferramenta é capaz de interpretar objetos e gerar o respectivo código SQL e executá-lo internamente, obtendo o resultado para o usuário;
- **Executa SQL** - Caso ainda seja necessário a realização da execução de um código SQL, a ferramenta é capaz de executar qualquer sintaxe SQL, retornando o resultado da mesma maneira;
- **Velocidade de desenvolvimento** – Para a realização da integração da ferramenta com um Banco de dados relacional, basta apenas que o programador configure, por meio de um arquivo xml, alguns parâmetros indicando a url do banco, driver, usuário e senha. A partir daí, por meio da utilização do paradigma da O.O, já se é possível a realização de qualquer transação por intermédio da ferramenta.

2.1 Metodologia de Trabalho

Este trabalho foi elaborado nos seguintes métodos:

- Pesquisa bibliográfica em livros relacionados a linguagem de programação JAVA, em livros relacionados a padrões de projetos e em livros relacionados a boas práticas de programação;
- Pesquisa em sites na internet;
- Ajustes de código para otimização da ferramenta, baseado nas consultas realizadas.

A pesquisa bibliográfica é definida por Koche (2006) como sendo a que se desenvolve procurando explicar um problema, utilizando o conhecimento disponível, a partir das teorias evidenciadas em livros ou obras congêneres. Na pesquisa bibliográfica, o investigador levanta o conhecimento disponível na área, identificando as teorias produzidas, analisando-as e avaliando a contribuição delas para auxiliar a compreender ou explicar o problema objeto da investigação.

2.2 Precisão de Alocação de Recursos

Para o desenvolvimento do Framework, foi necessário que os recursos humanos e materiais fosse respectivamente alocados da seguinte maneira:

- **Recursos Humanos** – Autor do projeto;
- **Recursos Materiais** – Computador MacBook Pro, Hospedagem de serviço de banco de dados em nuvem, Hospedagem de sistema web, softwares Eclipse e PostgreSQL.

Segundo Kotler (2000), o planejamento estratégico é, dessa forma, definido como um processo gerencial de desenvolver e manter um ajuste viável entre os objetivos, as habilidades, os recursos de uma organização e as oportunidades de um mercado em constante mudança.

Recursos de Hardware e Software		
Quantidade	Recurso	Custo
1	Computador MacBook Pro	R\$ 6000,00
1	Hospedagem de servidor de banco de dados	R\$ 240,00
1	Hospedagem de servidor com container de aplicação Tomcat	R\$ 240,00
1	Softwares para desenvolvimento de sistemas Navicat	R\$ 400,00
1	Software Eclipse e PostgreSQL	R\$ 0,00

Tabela 1: Recursos de Hardware e Software

Recursos Humanos		
Quantidade	Recurso	Custo
1	Analista de Sistemas Sênior com habilidade em programação JAVA – 2600 horas	R\$ 15.000,00

Tabela 2: Recursos Humanos

2.3. Cronograma de trabalho

O planejamento das atividades no tempo adequado para alcançar os objetivos pode ser verificado na Tabela 3:

	Janeiro	Fevereiro	Março
Proposta do Trabalho	X		
Estabelecimento dos Objetivos	X		
Leitura da bibliografia	X	X	X
Parte 1 do trabalho, documentação parcial do trabalho. (Itens 1, 2 e 3)	X	X	X
Trabalho final com documentação completa do trabalho. (Itens 1, 2 e 3 revisados e 4 e 5)			X

Tabela 3: Cronograma de Trabalho

3 A EMPRESA E O NEGÓCIO

A Cientz Publicidade e Tecnologia é uma empresa que atua há 10 anos no mercado de publicidade e propaganda na cidade de Fortaleza no estado do Ceará. Atualmente é detentora de startups na área de inovação de projetos em tecnologia da informação.

3.1 Histórico da empresa

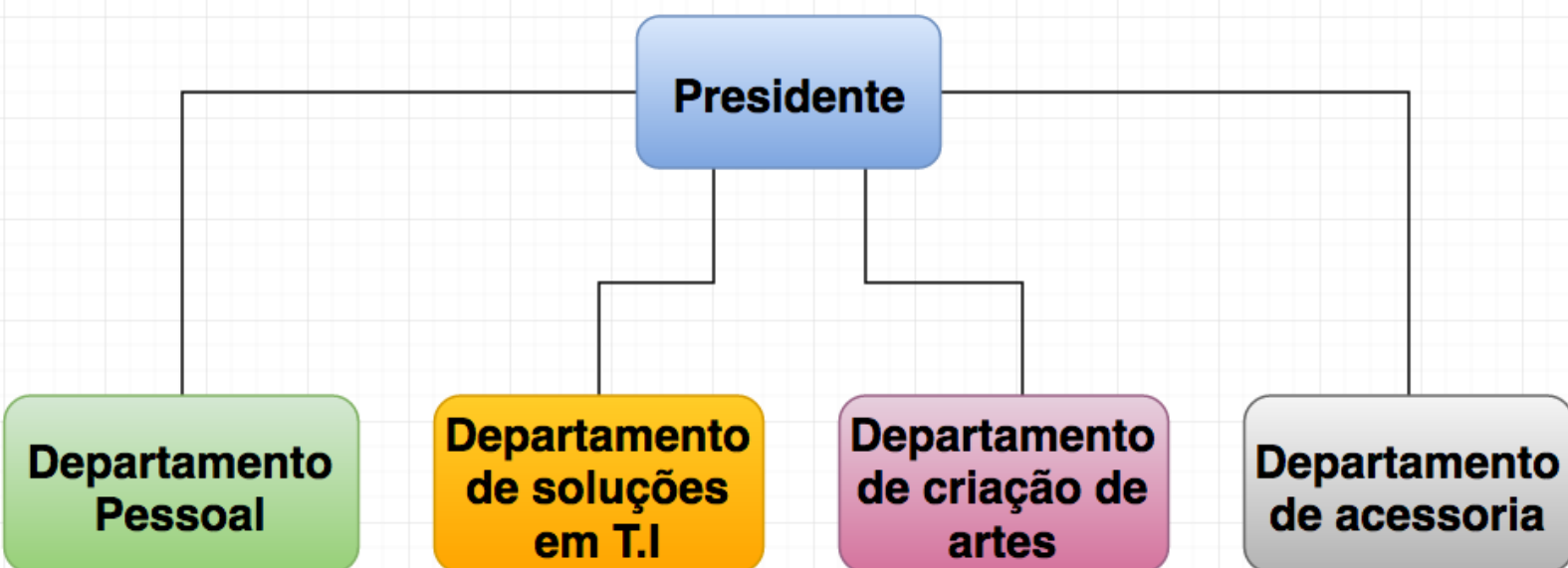
Com o intuito de entrar no neste novo mercado, a Cientz Publicidade e Tecnologia visa mesclar aos seus sistemas de informação a área da publicidade, aproximando ainda mais fornecedores e consumidores, de maneira que as oportunidades de mercado sejam ainda mais difundidas.

Sua proposta é que com um atendimento diferenciado, a empresa se coloque-se o mais próximo possível de seus clientes, de maneira que dessa forma o entendimento e atendimento de suas necessidades são supridos o mais rápido possível.

3.2 Atividades da empresa

A Cientz Publicidade atua nas seguintes vertentes de serviço:

- Assessoria de Imprensa;
- Lançamento de empresa;
- Lançamento ou Relançamento de produto;
- Lançamento Imobiliário;
- Criação de Briefing;
- Desenvolvimento de logotipo e identidade visual;
- Produção Digital;
- Monitoramento e criação de conteúdo para mídias sociais;
- Campanha incentivo e/ou relacionamento;
- Ação e campanha Promocional;
- Portfólio;
- Plano de mídia;
- Levantamento de dados sobre cliente e/ou produtos;
- Desenvolvimento de estratégias de comunicação publicitária;
- Criação de material promocional/Ponto de venda;
- Apresentação em Feiras, Eventos e Convenções;
- Rotulagem e etiquetas;
- Criação de catálogos, anúncios em jornal, adesivos, blimp, busdoor, display, front ligh, lumisoso, outdoor, painéis, placas backdrop, adesivação de veículos e outros;



3.4 Mercado consumidor

A Cientz Publicidade não possui um público alvo a ser atingido, de maneira que qualquer empresa ou pessoa física que sentir a necessidade de um de seus serviços ofertados pode vir a consumi-los. Entretanto dentre seus maiores clientes estão as empresas Grupo Nova Distribuidora e Vent7 inovações, para os quais a empresa possui algumas startups na área de tecnologia da informação em desenvolvimento.

3.5 Concorrência

A Cientz Publicidade, pelo fato de ser uma empresa de pequeno /médio porte, não tem em específico concorrentes diretos. Segundo seu CEO Edison Araújo, assim como a Cientz Publicidade, existem inúmeras outras empresas do mesmo tamanho, ou profissionais informais trabalhando na área publicitária. Por justa causa a Cientz trata como concorrente toda e qualquer empresa de publicidade do estado do Ceará.

3.6 Premissas e Restrições ao projeto

Atualmente a Cientz trabalha com as tecnologias de desenvolvimento PHP e Java. Como a ferramenta SigmaDB é desenvolvida exclusivamente para Java, os projetos da empresa, implementados nesta tecnologia, serão todos adaptados para que contemplem esta nova estrutura.

Por se tratar de um Framework de transação de dados, a ferramenta pode ser utilizada para o desenvolvimento de aplicações de ambientes Desktops e Web.

Para a utilização da ferramenta, se fará necessário que os projetos da empresa sigam a arquitetura apresentada abaixo:

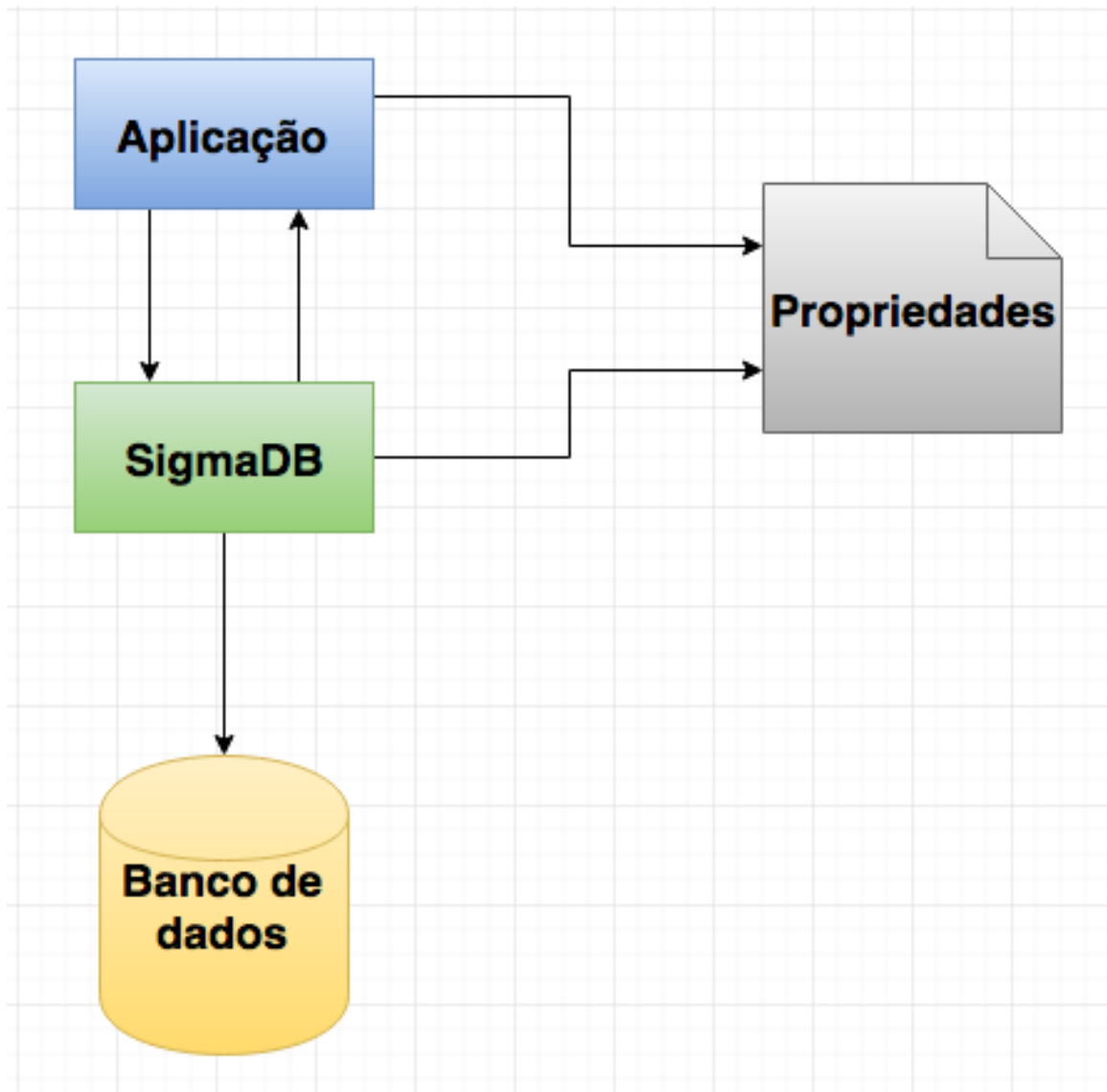


Figura 3: Arquitetura do Framework

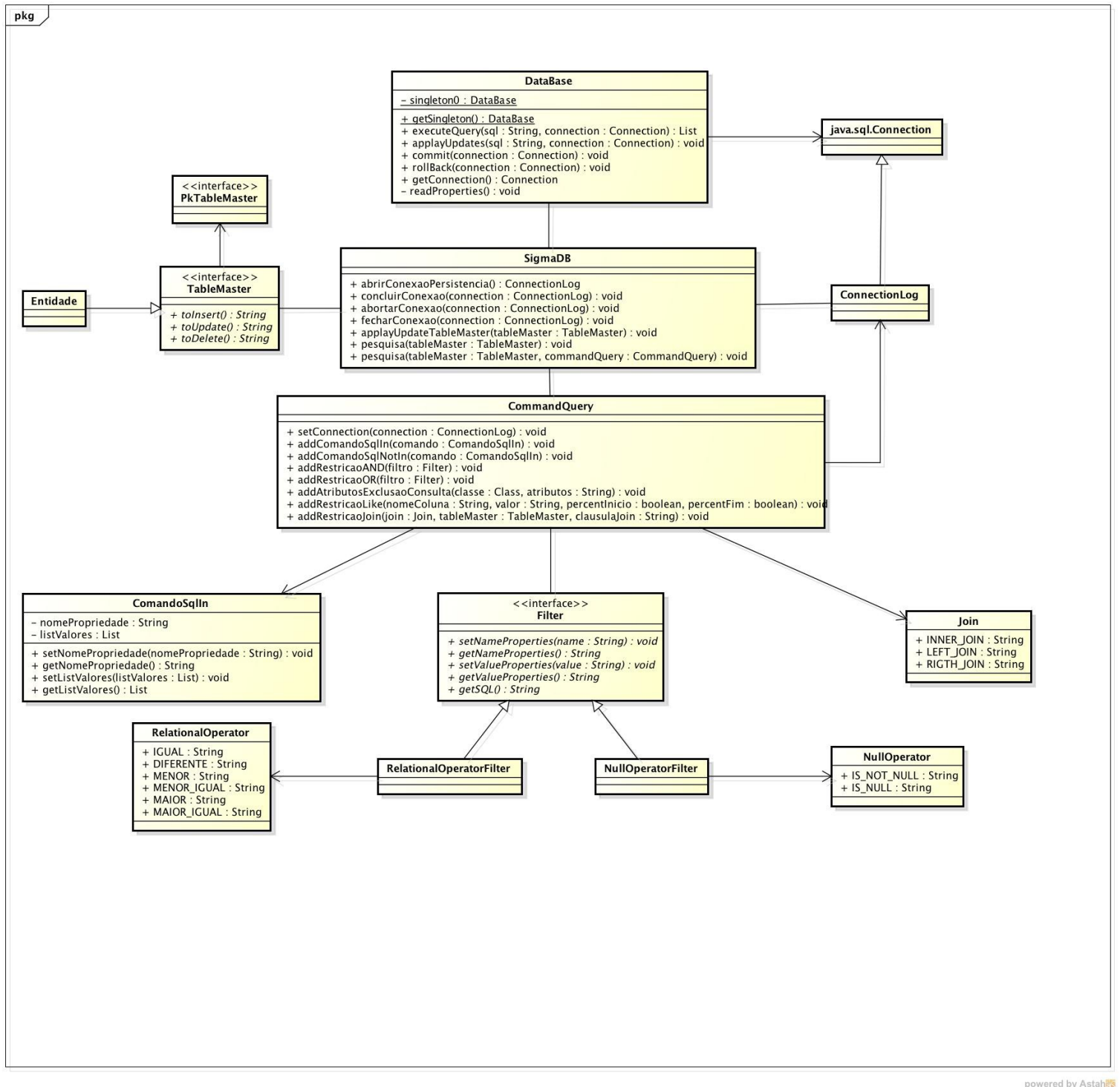


Figura 4: Diagrama de classes e relacionamentos do Framework

Podemos observar que o framework é uma ponte direta que liga a comunicação da aplicação a ser desenvolvida com sua base de dados. Entre eles, além do relacionamento entre classes JAVA, existe também um arquivo de propriedades, responsável por armazenar as informações pertinentes a conexão com o banco de dados. Este arquivo de propriedades deve

estar configurado no projeto da aplicação de acordo com as especificações do schema XSD da API da ferramenta.

Segundo ELMASRE (2010), uma entidade é equivalente a um objeto ou conceito do mundo real. ELMASRE (2010) complementa dizendo que um atributo representa alguma propriedade de interesse que compõe melhor uma entidade. Assim sendo, podemos concluir como entidade toda e qualquer tabela do banco de dados, assim como atributo as colunas destas tabelas.

Para o funcionamento da ferramenta, se faz necessário a existência de entidades do banco em formato de classe Java. Ou seja, se necessita de que exista um espelho da tabela em forma de um arquivo java. Para cada tabela que o usuário programador necessite manter no banco, deverá haver uma classe Java com o mesmo nome da entidade, mesmos nomes de atributos e respectivos tipos de dados. Essa classe que representa uma entidade, obrigatoriamente deverá herdar a classe TableMaster da ferramenta, conforme podemos observar na figura 4. O atributo que representa uma Primary Key, deverá obrigatoriamente receber uma *annotation* PkTableMaster. Será por meio desta interface que a ferramenta será capaz de identificar qual dos atributos da classe representarão a chave primária da entidade e desta forma montar as devidas restrições das transações.

Toda entidade tem um ponto único de entrada e saída de dados, que será a classe SigmaDB. Esta classe é capaz de realizar inserts, updates, deletes e selects. Os valores dos atributos da classe servirão como restrições para realização de consultas e persistência de dados.

Caso seja necessário a realização de consultas que envolva a junção de duas ou mais tabelas, o usuário deverá informar uma classe que delega as informações das entidades solicitadas, de maneira que a própria ferramenta se responsabilizará pela montagem da estrutura SQL das entidades contidas na classe, retornando uma lista de instâncias da classe informada contendo o resultado da consulta.

4 O SISTEMA ATUAL

Atualmente a empresa não possui nenhum Framework de trabalho para tratar da comunicação da tecnologia Java com banco de dados. Toda a estrutura de conexão com banco é feita por meio de conexões JDBC e por meio de classes proprietárias aleatórias da própria empresa. Isto tem causado grande dificuldade de manutenção de código, devido a que a falta de administração destas estruturas acabaram por gerar códigos redundantes e estruturas completamente desarticuladas.

4.1 Justificativa de Escolha do Sistema

A escolha por elaborar e implantar o Framework de transações com banco de dados, se dá pela necessidade de uma ferramenta mais robusta, veloz, centralizadora, que seja capaz de realizar qualquer comando SQL, sem que se faça necessário a escrita dos mesmos. Ou seja, uma ferramenta capaz de automatizar a conversão do paradigma O.O em estrutura SQL.

Existe a necessidade de se conseguir códigos coesos, desacoplados e não redundantes, de maneira que o profissional da área de desenvolvimento de software consiga agilidade e qualidade no seu trabalho.

4.1.1 O Sistema

A SigmaDB é uma ferramenta estruturada sobre a tecnologia Java capaz de se integrar a qualquer SGBD e realizar qualquer tipo de comando SQL dos tipos DML e DDL.

Além desta capacidade, ela traz recursos essenciais tais como abertura de transações, podendo desta maneira o usuário optar por realizar o commit ou rollback de operações com o banco, a seu critério.

A ferramenta também é capaz de agregar ao software que o utiliza, uma estrutura de auditoria de segurança de tuplas, de maneira que um administrador identifique quem, quando e aonde um usuário realizou determinada persistência na base de dados. Além deste benefício, o administrador terá em mãos o versionamento de tuplas, graças a estrutura de auditoria gerada pelo Framework. Desta forma o usuário será capaz de realizar o rollback de toda uma transação, independentemente de qual data esta tenha ocorrido. Esta é uma estrutura adicional da ferramenta, cuja qual pode ser ativada ou desativada por meio do arquivo de configuração da ferramenta.

Com o intuito de simplificar ao máximo a utilização do framework, a configuração do mesmo dar-se apenas pela criação de um arquivo do tipo XML no projeto onde ele será utilizado. Definidas todas as propriedades obrigatórias neste arquivo, a ferramenta já encontra-se funcional. Ou seja, toda a conexão com SGBD já encontra-se montada.

4.1.2 Funcionamento do Sistema

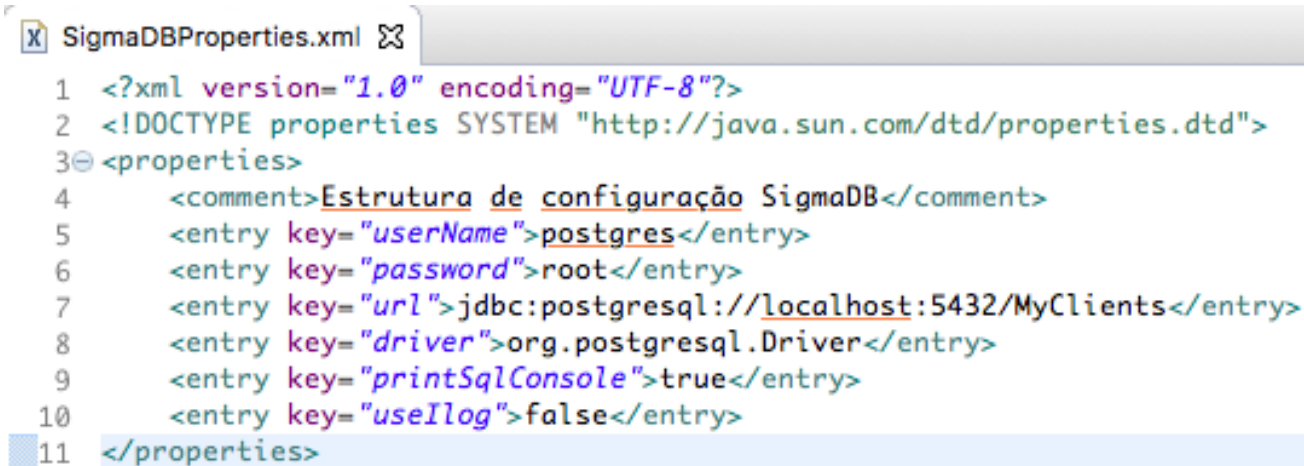
4.1.2.1 Configurando o Framework

O Framework deve ser configurado no projeto, incluindo a biblioteca de nome no SigmaDB.jar no classpath do projeto. Acompanhada desta biblioteca deverá ser incluídos os drivers do banco de dados que será utilizado na aplicação.

No projeto deverá ser criado um pacote de nome main.resources e dentro deste pacote deverá ser criado um arquivo de extensão .properties. Neste arquivo existirão todas as informações que são mutáveis a ferramenta. Ele guardará o nome do usuário, senha, descrição

de driver, url de conexão e verificação de permissão de utilização do console da aplicação para printar comandos sql executados.

Figura 5: Visão do arquivo SigmaDBProperties.xml



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3  <properties>
4      <comment>Estrutura de configuração SigmaDB</comment>
5      <entry key="userName">postgres</entry>
6      <entry key="password">root</entry>
7      <entry key="url">jdbc:postgresql://localhost:5432/MyClients</entry>
8      <entry key="driver">org.postgresql.Driver</entry>
9      <entry key="printSqlConsole">true</entry>
10     <entry key="useIlog">false</entry>
11 </properties>

```

4.1.2.2 Definição de padrão de utilização

Visando excelência no desenvolvimento, a ferramenta foi construída baseada em padrões de projeto. Segundo Christopher Alexander (1978), um padrão descreve um problema que pode ocorrer inúmeras vezes em determinado contexto, e descreve ainda a solução para esse problema, de maneira que essa solução possa ser utilizada sistematicamente em situações semelhantes.

A ferramenta baseia-se num padrão muito conhecido e difundido de nome DAO (*Data Access Object*). Este padrão visa definir uma interface única para a realização de processos que envolvam ligações diretas com um banco de dados. Desta maneira torna-se viável a separação da regra de negócio das aplicações com as regras de acesso a banco de dados.

Por consequência, toda classe do sistema que necessitar de uma interface com o banco de dados, deverá obrigatoriamente criar uma instância da classe SigmaDB que estará embutida dentro do Framework. A figura 6 apresenta um exemplo da chamada a classe e a listagem das principais funcionalidades disponibilizadas por ela. Podemos notar que a classe SigmaDB é uma interface centralizadora para o banco, de maneira que ela controla toda e qualquer requisição feita ao mesmo.

```

// Instância da classe

SigmaDB sigma = new SigmaDB();

-----

// Principais funcionalidades

public ConnectionLog abrirConexaoPersistencia() throws Exception;

```

```

public void concluirConexao(ConnectionLog connectionLog);

public void abortarConexao(ConnectionLog connectionLog);

public void applyUpdateTableMaster(TableMaster tableMaster, ConnectionLog connectionLog, DBOperation operation)

public <E extends TableMaster> List<E extends TableMaster> pesquisaTabela(E extends TableMaster bean) ;

```

Figura 6: Principais funcionalidades da classe SigmaDB

4.1.2.3 Tratamento de tabelas

Assim como utilizado em Frameworks conhecidos no mercado, como Hibernate e JPA, toda tabela do banco de dados deve possuir uma classe Java que seja sua cópia exata. Ou seja, uma classe Java em que cada atributo da classe corresponda a nomenclatura e tipo de dados definidos na entidade de banco de dados.

A ferramenta reconhece que uma determinada classe Java representa uma tabela do banco de dados, desde que esta implemente uma interface de nome TableMaster. Uma interface TableMaster dará a uma determinada classe Java toda a estrutura necessária para a geração de códigos SQL dinâmicos. Ou seja, é graças a esta estrutura que o programador poderá optar por não escrever SQL no desenvolvimento, fazendo desta maneira com que a ferramenta faça isto por ele.

Tabelas no banco de dados podem ou não possuir chaves primárias. Assim sendo, toda TableMaster possui uma Annotation de nome @PKTableMaster, que deverá ser aplicada para algum atributo da classe. A ferramenta identificará que o atributo que possuir esta Annotation na definição da classe, será respectivo a chave primária da tabela. A figura 7 apresenta um exemplo de uma classe que representa uma tabela para a ferramenta, assim como a definição de uma chave primária nela.


```

/**
 * Classe que representa a tabela de nome Pessoa no banco de dados.
 * @author Igor Moisés
 * @since 01/11/2016
 */
public class Pessoa extends TableMaster{

    @PKTableMaster
    private int pess_id;

    private String pess_nome;

    private String pess_login;

    private String pess_senha;

    private String pess_status;

    private String pess_cpf;

    private String pess_rg;

    private int pess_tipo;

    private Timestamp pess_nascimento;

    private String pess_sexo;

    ...
}

```

Figura 7: Exemplo de definição de classe para tabela

4.1.2.4 Consultando e persistindo dados

Podemos observar por meio das figuras 6 e 7 que os métodos que realizam consultas e persistências da interface SigmaDB, estão sempre esperando uma classe TableMaster. Ou seja, eles estarão sempre esperando uma classe que representa uma tabela do banco de dados.

Se um objeto TableMaster for informado ao método de consulta da classe SigmaDB, este por sua vez utilizará os valores das propriedades do objeto repassado como restrições para a consulta. Já se um objeto TableMaster for informado para o método de persistência, todos os valores contidos nas propriedades do objeto serão persistidos no banco. O método de persistência exige que o usuário informe se ele precisa Realizar um INSERT, UPDATE ou DELETE. Esta informação é repassada por parâmetro, conforme podemos observar na figura 8.

Ao persistir qualquer tabela, a Interface IntegracaoDAO, gera automaticamente toda a estrutura de auditoria e versionamento de dados. Ao analisarmos a figura 8, podemos ver que sempre que algo é solicitado ao banco, é exigido uma conexão ativa. Esta conexão que ele solicita tem a função de servir de mapeamento para transação. Ou seja, é através dela que a

ferramenta consegue desfazer o commit de um único registro, assim como o commit de uma transação de N de registros.

```
// Exemplo de persistência de dados

SigmaDB sigma = new SigmaDB();

Pessoa pessoa = new Pessoa();

pessoa.setPess_id(100);

pessoa.setPess_tipo(1);

pessoa.setPess_nome("José Martins Vasconcelos");

ConnectionLog connection = sigma.abrirConexaoPersistencia();

sigma.applyUpdateTableMaster(pessoa, connection, DBOperation.INSERT);

sigma.concluirConexao(connection);

-----

// Exemplo de consulta de dados

Pessoa pessoa = new Pessoa();

pessoa.setPess_tipo(1);

//Supondo que a classe Estabelecimento extenda TableMaster

Estabelecimento estabelecimento = new Estabelecimento();

estabelecimento.setEstab_tipo(2);

List<Pessoa> retornoConsulta = sigma.pesquisaTabela(pessoa);

List<Estabelecimento> retornoConsulta = sigma.pesquisaTabela(estabelecimento);
```

Figura 8: Exemplo de consulta e persistência no Framework

4.1.3 O Ambiente do Sistema

Por se tratar de uma ferramenta para o auxílio no desenvolvimento de sistemas, o Framework deverá ser utilizado por meio de IDEs de desenvolvimento para a tecnologia Java que proporcionem a utilização e importação de arquivos do tipo jar, que nada mais é do que um binário compactado de classes Java.

A ferramenta atuará nas estruturas de código definidas pelo programador, cujas quais atendam todos os requisitos explicados anteriormente nos itens 3.1.2.1, 3.1.2.2 e 3.1.2.3.

4.1.4 A definição do escopo

O objetivo deste projeto é apresentar uma ferramenta desenvolvida para que possibilite dar agilidade, melhoria de estruturação de código e centralização de comandos relacionados a Banco de dados.

A ferramenta deverá ser flexível a qualquer ambiente de execução. Ou seja, deve ser independente do sistema operacional utilizado pela máquina em que ele estiver sendo executado.

O sistema contará com as áreas descritas abaixo:

- Realização de consultas: A ferramenta deverá ser capaz de realizar qualquer consulta a qualquer tabela do sistema, de maneira que se um SQL não for informado para a consulta, a própria ferramenta deverá ser autônoma para gerar seu próprio SQL;
- Realização da persistência de dados: A ferramenta deverá ser capaz, por meio de uma estrutura única, salvar registros em qualquer tabela do banco;
- Realização de auditoria e versionamento de dados: A ferramenta deverá ser capaz de registrar todo commit feito por ela no banco em forma de log, guardando data, usuário e o que foi alterado, inserido e/ou removido;

4.2 Motivação para o novo sistema

Conforme descrito anteriormente, a empresa utiliza-se de uma estrutura de classes de propriedade própria para a realização de conexão com seu banco de dados. Essa estrutura de classes causava certos transtornos devido a sua dificuldade de manutenção, necessidade de se conseguir códigos coesos e não redundantes, de maneira que o profissional da área de desenvolvimento de software conseguisse agilidade em seu trabalho.

A empresa tem em sua necessidade principal fazer com que seus programadores não percam tempo desenvolvendo instruções de banco para serem executadas. A sua metodologia será a de se preocupar com a lógica da regra de negócio de suas aplicações ao invés de se além de se preocupar com esta lógica de negócio, também se preocupar com o banco de dados.

4.3 Situação desejada

A ferramenta desenvolvida deverá atender as expectativas dos programadores que a utilizarão, de maneira que estes por intermédio dela consigam construir códigos menos complexos, mais coesos, acoplados e principalmente ganhem uma grande velocidade de trabalho, não perdendo mais tempo com desenvolvimento de funcionalidades SQL.

4.4 Problemas no sistema atual

Conforme já comentado, a empresa não possui nenhuma ferramenta do gênero para este tipo de trabalho proposto pelo Framework aqui apresentado.

A empresa utiliza uma estrutura própria de conexão com banco de dados, a qual gera transtornos pelo fato de não trazer a agilidade necessária no desenvolvimento, assim como a geração de códigos de difícil manutenção.

Com a utilização da ferramenta aqui proposta, todas limitações descritas serão solucionadas.

5. O Sistema Proposto

5.1 Lista de Requisitos do Sistema

5.1.1 Requisitos Funcionais

Os requisitos funcionais definem como o sistema se comporta, através das declarações das funções que o sistema deve oferecer, de como o sistema deve reagir a entradas específicas e de como deve se comportar em determinadas situações, ou ainda define restrições explícitas do que o sistema não deve fazer(SOMMERVILLE, 2003).

Código	Descrição
RF_01	<p>Gerenciamento de conexão com bancos</p> <p>A ferramenta deve ser capaz de realizar conexão com qualquer banco de dados do tipo relacional.</p> <p>Para tanto, deverá dispor de uma estrutura genérica que seja capaz de ser configurada externamente no projeto que irá encapsular toda a API do Framework.</p> <p>No projeto externo, deverá ser criado um pacote de nome main.resources e dentro deste pacote deverá ser criado um arquivo de nome escolhido pelo usuário, porém com a extensão “.properties”.</p> <p>Neste arquivo deverá ser incluído os parâmetros abaixo:</p> <ul style="list-style-type: none"> - userName (Nome do usuário do banco de dados); - password (Senha do usuário do banco de dados); - url(URL de conexão com o banco de dados); - driver(Nome do driver de conexão com o banco de dados); - printSql(Boleano que indica se qualquer estrutura sql montada pela ferramenta, deva ser escrita no console da aplicação)
RF_02	<p>Mapeamento de tabelas</p> <p>A ferramenta deverá ser capaz de mapear classes e tabelas.</p> <p>Para tanto deverá ser desenvolvida uma interface que indique que uma determinada classe represente uma tabela do banco de dados.</p> <p>Esta interface deverá ser chamada de TableMaster e será responsável por realizar o mapeamento de tudo o que vem a ser uma tabela para a ferramenta. Esta estrutura também deverá possuir uma Annotation capaz de identificar qual atributo da classe representará uma primare key.</p>
RF_03	<p>Gerenciamento de Logs</p> <p>Toda tupla inserida no banco, independentemente de tabela, deverá ser auditada por meio de uma estrutura que armazene:</p> <p>data/hora do ocorrido, identificador de quem requisitou a ação, local de onde partiu a requisição, uma numeração única para a transação que será chamada de versão.</p> <ul style="list-style-type: none"> - Identificador de registro(Identificador único para o registro de log); - Data/hora(Data e hora em que ocorreu o evento de persistência de um dado); - Identificador de usuário (Alguma informação que identifique um usuário que realizou a persistência); - Local(Local de onde partiu a requisição da persistência); - Identificador de transação (Identificador único para a transação) - Conteúdo existente antes da alteração
RF_04	<p>Gerenciamento da estrutura de consultas sem sql</p> <p>Para utilizar as estruturas de pesquisa, a</p>

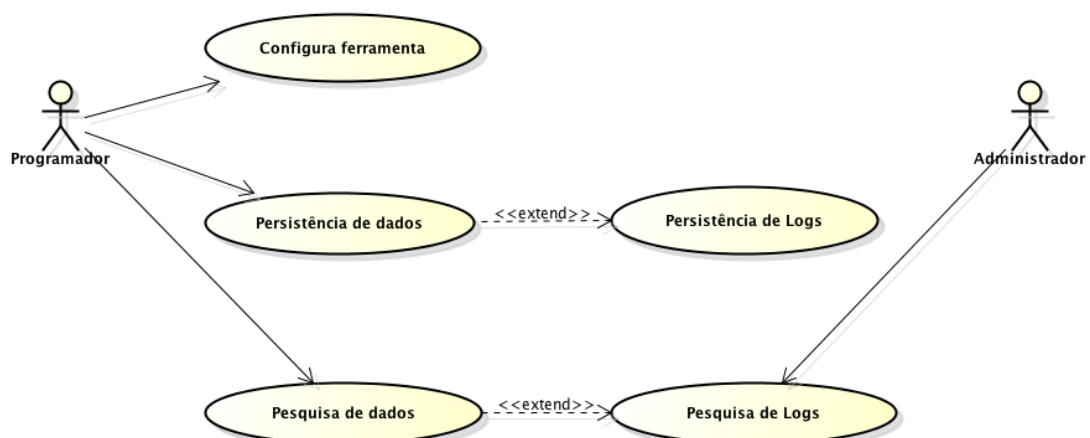
	<p>ferramenta deverá disponibilizar interfaces para que se possa realizar as devidas ações.</p> <p>Para tanto deverá ser disponibilizada uma interface que seja capaz de receber qualquer instância de classe e realizar uma consulta a base de dados.</p> <p>A ferramenta deverá ser capaz de identificar se a instância representa uma ou mais tabelas e desta forma realizar a respectiva consulta, tomando como restrições da consulta todos os valores das propriedades da instância informada.</p> <p>O retorno da consulta deverá ser uma lista de instâncias, do mesmo tipo da informada a ferramenta, contendo o resultado da consulta.</p>
RF_05	<p>Gerenciamento da estrutura de consultas com sql</p> <p>Para utilizar as estruturas de pesquisa, a ferramenta deverá disponibilizar interfaces para que se possa realizar as devidas ações.</p> <p>Para tanto deverá ser disponibilizada uma interface que seja capaz de receber qualquer instância de classe, uma consulta no formato sql e realizar uma consulta a base de dados, tomando como base o sql informado pelo usuário.</p> <p>O retorno da consulta deverá ser uma lista de instâncias, do mesmo tipo da informada a ferramenta, contendo o resultado da consulta.</p>
RF_06	<p>Gerenciamento da estrutura de persistência</p> <p>Para utilizar as estruturas de persistência de dados, a ferramenta deverá disponibilizar interfaces para que se possa realizar as devidas ações.</p> <p>Para tanto, deverá ser disponibilizada uma interface que se permita realizar qualquer tipo de persistência de dados. Será um único ponto de acesso onde se possa realizar as ações de Inserts, Updates e Deletes.</p> <p>Esta estrutura deverá receber uma instância de classe que represente uma tabela do banco de dados, com seu respectivo tipo de ação solicitado, seja um Insert, Update ou Delete.</p> <p>Baseado nos valores das propriedades da instância informada, a ferramenta deverá ser capaz de realizar a ação solicitada.</p> <p>Como retorno da ação, a ferramenta deverá sempre retornar o id(primary key) do registro trabalhado, se houver.</p>
RF_07	<p>Realização do rollback de dados baseado em versão.</p> <p>A ferramenta deverá ser capaz de realizar o roolback de dados do banco de dados. Para tanto ela deverá bazear-se nas informações guardadas através do #RF_03.</p> <p>Para tanto a ferramenta deverá disponibilizar uma interface que receba uma numeração de versão de transação e substitua as propriedades de todas as tuplas referenciadas pela versão, pelo valor da versão anterior.</p> <p>A saída desta ação deverá ser a visualização das tuplas com os valores anteriores a ultima alteração.</p>

5.1.2 Requisitos Não Funcionais

Código	Descrição
--------	-----------

RNF_01	<p>Estrutura complementar de consultas</p> <p>A ferramenta deverá disponibilizar para o usuário uma estrutura auxiliar para o desenvolvimento de consultas. Essa estrutura deverá ser chamada de ComplementoConsulta e será uma interface capaz de realizar as ações descritas abaixo:</p> <ul style="list-style-type: none"> - Permitir realizar restrições do tipo IN e Not IN em lista de objetos; - Permitir realizar restrições dentro do período de datas e tempos; - Permitir realizar restrições fora do período de datas e tempos; - Permitir realizar verificações elementos nulos em atributos de tuplas, tais como os comandos sql isnull e is not null; - Permitir realizar joins com qualquer tabela, fazendo com que consultas que apontem para uma única tabela possa realizar joins com qualquer tabela baseado na vontade e necessidade do usuário; - Permitir incluir novas restrições a consulta, além dos valores informados nas propriedades da instância de objetos informados para a interface de captação de consultas da ferramenta. Estas restrições podem ser cláusula AND e cláusulas OR;
RNF_02	<p>Desenvolvimento de consultas</p> <p>Baseado em RNF_01, a ferramenta deverá disponibilizar uma estrutura que seja capaz de realizar o desenvolvimento de qualquer consulta por meio de código JAVA, sem que seja necessário o usuário escrever nenhum comando SQL.</p> <p>Para tanto, na estrutura de RNF_01, deverá ser criada uma maneira para que seja inserida um nome de tabela e através da estrutura de joins e o restante das restrições que estarão preparadas, possa-se realizar qualquer tipo de consulta, onde o retorno desta será o mesmo retorno descritos em RNF_04.</p>

5.2 Diagrama de Caso de Uso



5.3 Especificações dos casos de uso

5.3.1 Configura ferramenta

O caso de uso Configura Ferramenta possibilita que o usuário programador configure a ferramenta dentro do seu projeto de desenvolvimento.

Atores:

Programador

Pré-condições:

O usuário programador deverá possuir instalado em seu computador uma IDE de desenvolvimento da tecnologia Java e o arquivo SigmaDB.jar, respectivo ao framework.

Pós-condições:

Possuir a ferramenta pronta para o auxílio do desenvolvimento da aplicação.

Fluxo Principal

Passo	Descrição
1	O fluxo principal inicia quando o usuário programador cria um novo projeto e inclui a biblioteca SigmaDB.jar no seu classpath.
2	<p>O programador deverá criar no seu projeto um pacote de nome main.resources e dentro dele criar um arquivo, de nome a sua escolha, com a extensão “.properties”.</p> <p>Neste arquivo deverá haver os atributos descritos abaixo, preenchidos com seus respectivos valores.</p> <ul style="list-style-type: none"> - userName (Nome do usuário do banco de dados); - password (Senha do usuário do banco de dados); - url(Url de conexão com o banco de dados); - driver(Nome do driver de conexão com o banco de dados); - printSql(Boleano que indica se qualquer estrutura sql montada pela ferramenta, deva ser escrita no console da aplicação)

5.3.2 Persistência de dados

O caso de uso Persistência de dados possibilita que o usuário programador configure a ferramenta para que ela consiga persistir informações na base de dados.

Atores:

Programador

Pré-condições:

O usuário programador deverá ter sua base de dados já formulada e implementada.

Pós-condições:

Dados persistidos na base de dados

Fluxo Principal

Passo	Descrição
1	O Fluxo principal inicia com o usuário programador definindo classes JAVA que representem cada entidade do seu banco de dados. Cada classe que representa uma entidade do banco de dados deve possuir atributos que sejam respectivos aos nomes de cada atributo da entidade do banco de dados, assim como seu tipo.
2	O usuário programador deverá fazer com que cada classe que represente uma entidade do seu banco de dados, estenda a classe de nome TableMaster que estará dentro da biblioteca SigmaDB, incluída no classpath do projeto.
3	Se a entidade do banco de dados possuir uma Primare Key, o atributo respectivo da classe Java deverá receber a annotation @PkTableMaster
4	O usuário programador deverá abrir uma conexão com o banco chamando o método abreConexao(), que retorna uma instância da classe Connection, da classe IntegracaoDAO, que é um singleton.
5	De posse de uma conexão aberta o usuário poderá repassar qualquer TableMaster para o método persiste da classe IntegracaoDAO, repassando também a conexão que ele possui aberta. Ao realizar a chamada ao método persiste, o usuário programador deve informar também qual o tipo de persistência ele necessitará, através de um enumerador de Ações. Este enumerador listará o Insert, Update ou Delete.
6	O usuário programador, após o término de todas as suas persistências, chamará o método gravaConexao(), da classe IntegracaoDAO, informando como parâmetro a conexão aberta, ou a conexão que ele deseja que seja comitada na base de dados.

5.3.3 Pesquisa de dados

O caso de uso Pesquisa de dados possibilita que o usuário programador configure a ferramenta para que ela consiga consultar informações na base de dados.

Atores:

Programador

Pré-condições:

O usuário programador deverá ter sua base de dados já formulada e implementada.
O usuário precisa ter implementado em classes Java cópias idênticas de suas entidades do banco, tanto em nome, quanto em nomes de atributos com seus respectivos tipos.

Pós-condições:

Lista de dados contendo o resultado das consultas.

Fluxo Principal

Passo	Descrição
1	O fluxo principal inicia-se com o usuário programador criando uma instância de um TableMaster que represente alguma entidade do banco de dados.
2	O usuário programador deverá pegar a instância criada e preencher os atributos, de maneira que cada atributo será considerado como uma restrição para a consulta que será realizada.
3	Com os devidos atributos já preenchidos, o usuário programador chamará o método pesquisa da classe IntergracaoDAO, passando como parâmetro a instância de TableMaster criada.
4	Como retorno, o usuário programador, receberá uma lista de instâncias do tipo que foi enviada para a consulta contendo o resultado da consulta.

Fluxos Alternativos

Passo	Descrição
	Consulta com SQL
1	O usuário programador deverá criar uma nova classe que estenda a classe IntegracaoDAO.
2	O usuário programador deverá montar métodos de pesquisa que recebam como parâmetro as restrições da consulta.
3	Neste método o programador deverá escrever uma String contendo uma sintaxe SQL de consulta. Com a estrutura da pesquisa SQL montada, o programador deverá informar uma instância dos objetos que serão retornados pela consulta, e através do método pesquisa de acessibilidade protected da classe IntegracaoDAO é possível realizar uma consulta informando a instância e o sql.
4	Como retorno, o usuário programador, receberá uma lista de instâncias do tipo que foi enviada para a consulta contendo o resultado da consulta.

Passo	Descrição
	Consultas Dinâmicas
1	O usuário programador deverá criar uma instância de uma estrutura auxiliar chamada de ComplementoConsulta. O ComplementoConsulta é uma classe que servirá para

	desenhar consultas dinâmicas.
2	Com a instância de ComplementoConsulta, o usuário poderá preencher a propriedade "From" do objeto, assim como as propriedades "Join". Ambas as propriedades recebem como parâmetro Objetos do tipo Class, que devem obrigatoriamente representar um TableMaster, ou seja um espelho da entidade do banco de dados.
3	O usuário Programador poderá definir todas as restrições da consulta no objeto ComplementoConsulta, que conta com uma estrutura de todos os tipos de restrições sql para consulta.
4	O método pesquisa da classe IntegracaoDAO está sobrecarregado com uma chamada que aceita como parâmetro uma instância de ComplementoConsulta. Uma instância dos objetos que deverão ser retornados como resultado da consulta e uma instância de ComplementoConsulta deverão ser repassados ao método "pesquisa".
5	Como retorno, o usuário programador, receberá uma lista de instâncias do tipo que foi enviada para a consulta contendo o resultado da consulta.

5.3.4 Persistência de logs

O caso de uso Persistência de logs possibilita que o usuário programador configure a ferramenta para que ela consiga persistir informações de auditoria de dados na base de dados.

Atores:

Programador

Pré-condições:

O usuário programador deverá ter sua base de dados já formulada e implementada.
O usuário precisa ter implementado em classes Java cópias idênticas de suas entidades do banco, tanto em nome, quanto em nomes de atributos com seus respectivos tipos.

Pós-condições:

Logs persistidos na base de dados

Fluxo Principal

Passo	Descrição
1	O fluxo principal inicia-se com o usuário programador criando uma nova entidade no banco de dados. Esta entidade deverá chamar-se de "log" e deverá conter os atributos abaixo. - Id : Integer - versao: Integer

	<ul style="list-style-type: none"> - tipo: Varchar - tabela – Varchar - pk_tabela – Integer - data_hora – Timestamp - origem – Varchar - usuario – Integer - valores - Varchar
2	Quando o usuário programador realizar qualquer persistência de dados, a ferramenta se responsabiliza de persistir as informações da entidade Log respectivas ao registro que está sendo persistido.

5.3.5 Pesquisa de logs

O caso de uso Pesquisa de logs possibilita que o usuário programador configure a ferramenta para que ela consiga consultar informações de auditoria de dados na base de dados.

Atores:

Programador, Administrador

Pré-condições:

O usuário programador deverá ter sua base de dados já formulada e implementada.
O usuário precisa ter implementado em classes Java cópias idênticas de suas entidades do banco, tanto em nome, quanto em nomes de atributos com seus respectivos tipos.
O usuário precisa possuir a entidade Log implementada.

Pós-condições:

Lista de dados contendo o resultado das consultas.

Fluxo Principal

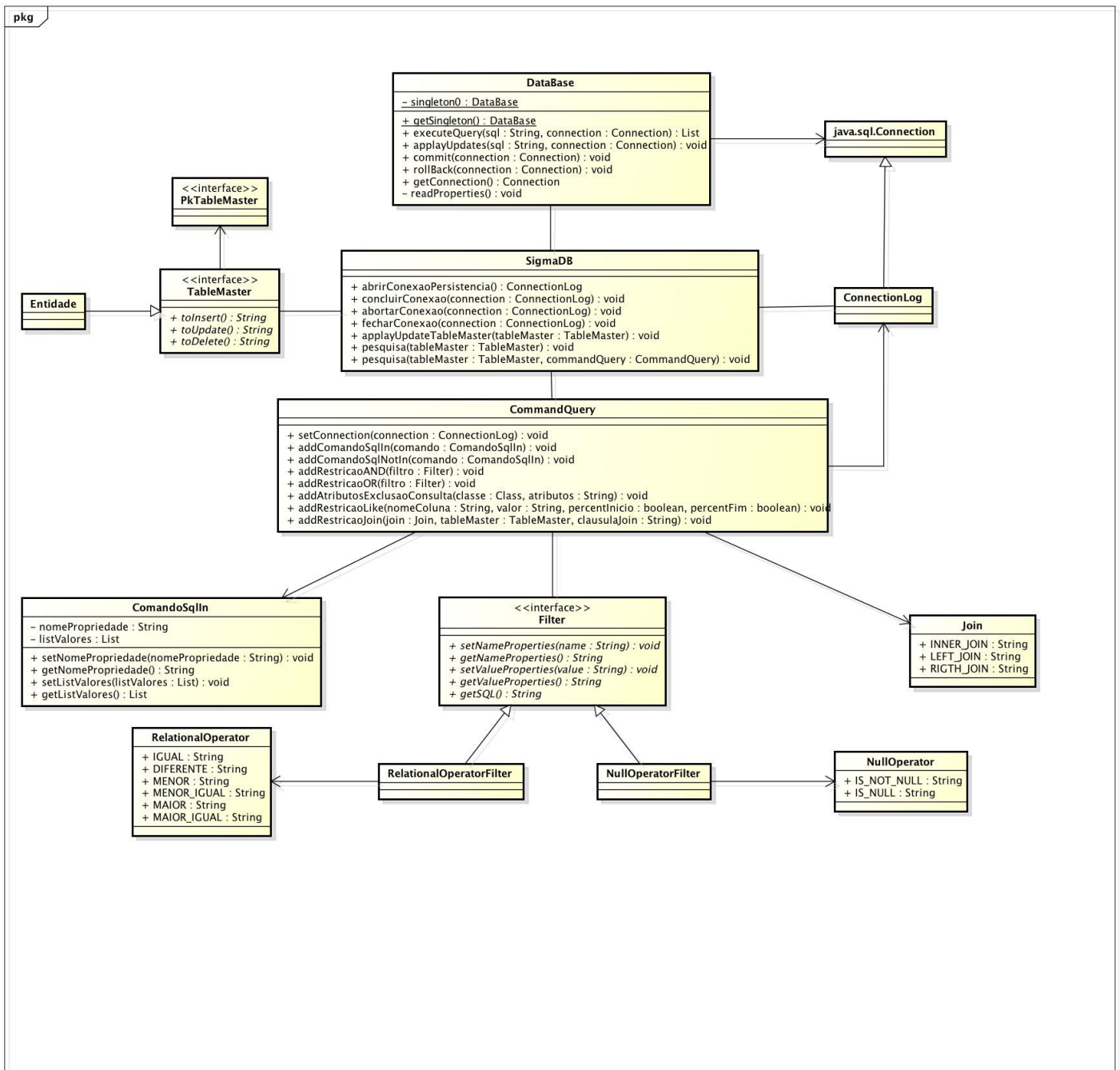
Passo	Descrição
1	O fluxo principal inicia-se com o usuário programador criando uma instância da classe Log.
2	Com a instância criada o usuário programador deverá preencher os atributos da instância, de maneira que cada atributo preenchido seja respectivo a uma restrição da consulta.
3	Feito o preenchimento dos atributos que servirão como restrições da consulta, o usuário deverá chamar o método de pesquisa da classe IntegracaoDAO, passando como parâmetro a instância da entidade Log.

4	Como resultado, o usuário programador receberá uma lista de instâncias da classe Log, contendo o resultado da consulta.
---	---

Fluxo Alternativo

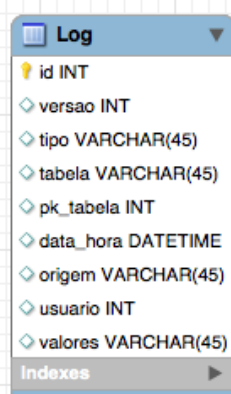
Passo	Descrição
	Utilização do Administrador
1	O fluxo alternativo inicia com o usuário Administrador acessando a base de dados.
2	Ao realizar acesso a entidade Log pela base de dados, o usuário administrador consegue ter a mesma visão do usuário Programador, consultando a estrutura de log da maneira como necessitar.
3	Como saída o usuário administrador obtém o resultado da consulta que lhe for necessária.

5.4 Modelo Conceitual de Classes



powered by Astah

5.5 Modelo Conceitual de Dados



6. Conclusões

Com o crescimento do mercado seguido por suas exigências, faz-se necessário a criação de um modelo de negócio que seja rápido, intuitivo e de fácil manuseio. A partir desta premissa, é que identifiquei a necessidade do investimento nesta ferramenta, de maneira que um programador possa, por consequência dela, trabalhar de maneira mais ágil.

Entretanto, sabemos que para elaborar e implementar qualquer tipo de sistema, faz-se necessário a realização de um levantamento das necessidades dos usuários, de maneira que se possa identificar quais são os pontos mais críticos no desenvolvimento do trabalho deste. Este estudo tomou como base os pontos de maior tempo gasto em desenvolvimento, tendo como intenção trabalhar em cima de melhorias que façam com que não seja necessário gastar mais o tempo inicial de trabalho.

Com a intenção de ser um diferencial dentre as ferramentas do gênero propostas atualmente no mercado, a aplicação desenvolvida neste estudo traz como novidades a possibilidade da escrita de consultas sql, assim como o versionamento automático do SGBD utilizado.

Diante das análises e levantamentos das funcionalidades, identificou-se a necessidade da criação e manutenção de artefatos para o projeto. A saber, casos de uso, elicitação e documentação de requisitos funcionais e não funcionais.

Diante do exposto, espera-se que a ferramenta possa oferecer aos seus usuários um serviço desenvolvimento ágil, robusto e inovador.

Concluí-se, desta maneira, que a implantação da ferramenta nos projetos de desenvolvimento trará um conjunto de benefícios ao modelo de negócio, independentemente de qual escopo ele esteja incluído, tendo em vista a agilidade de desenvolvimento proposta para o trabalho de programação, assim como a segurança dos dados.

7. Referências Bibliográficas

ELMASRI;NAVATHE, E. Sistemas de banco de dados, 6 ed. São Paulo: Pearson, 2011. 3p.
SOMMERVILLE, I. Engenharia de Software. Glasgow: Addison-Wesley, 2003.

KOCHE, José Carlos. Fundamentos de metodologia científica: teoria da ciência e iniciação à pesquisa. 28.ed. Petrópolis, RJ: Vozes, 2006.

KOTLER, Philip. Administração de marketing: análise, planejamento, implementação e controle. 2. ed. São Paulo: Editora Atlas, 1992.

Christopher Alexander : A Pattern Language, Oxford Press, Oxford, R. Unido, 1978.

FREEMAN, E. et al. Use a Cabeça: Padrões de Projetos. 2.ed. Jacaré, RJ Alta Books, 2007.

WIKIPEDIA. Disponível em : <https://pt.wikipedia.org/wiki/Edgar_Frank_Codd>.

WIKIPEDIA. Disponível em : <<https://pt.wikipedia.org/wiki/ODBC>>.

WIKIPEDIA. Disponível em : <<https://pt.wikipedia.org/wiki/JDBC>>.