

# **SOFTWARE PARA CONTROLE DE ROTINAS SQL EM SGBD MYSQL**

**Gabriel Pinheiro Bartholomeu<sup>1,2</sup>**

**Aldelir Fernando Luiz<sup>2</sup>**

<sup>1</sup>Philips Clinical Informatics – Blumenau (SC)

<sup>2</sup>Instituto Federal Catarinense – Campus Blumenau

`gabriel_bartholomeu@hotmail.com`

`aldelir.luiz@ifc.edu.br`

BLUMENAU-SC

2018

**GABRIEL PINHEIRO BARTHOLOMEU**

**PROJETO E DESENVOLVIMENTO DE SISTEMA:**

**SOFTWARE PARA CONTROLE DE ROTINAS SQL EM SGBD MYSQL**

Trabalho de conclusão do curso de Tecnologia em  
Análise e Desenvolvimento de Sistemas, apresentado  
como requisito parcial para obtenção do grau de  
Tecnólogo em Análise e Desenvolvimento de Sistemas  
do Instituto Federal Catarinense – Campos Blumenau

Orientador: Prof. Aldelir Fernando Luiz

**BLUMENAU-SC**

**2018**

**GABRIEL PINHEIRO BARTHOLOMEU**

**PROJETO E DESENVOLVIMENTO DE SISTEMA:**

**SISTEMA PARA CONTROLE DE ROTINAS SQL EM SGBD MYSQL**

**BANCA EXAMINADORA**

---

Orientador: Prof.

---

Prof.

BLUMENAU-SC

2018

## **AGRADECIMENTOS**

Ao prof. MSc. Aldelir Fernando Luiz pelo suporte e orientações para desenvolvimento e efetivação deste trabalho.

Ao prof. MSc. Ricardo de la Rocha Ladeira pelo suporte e incentivo na entrega e finalização do trabalho.

Ao meu colega Cristian Soares Queiroz por ter ficado várias noites ao meu lado me ajudando a testar e levantar todas as pendências de melhoria e correção.

À minha família, meus pais Marcelo Bartholomeu e Cibele Cristina Pinheiro Bartholomeu e minha irmã Mariana Pinheiro Bartholomeu, por terem me dado o suporte para erguer a cabeça e seguir em frente nos momentos mais tensos e difíceis.

A todos muito obrigado pelo carinho e ajuda.

## RESUMO

O trabalho desenvolvido tem por foco suprir uma necessidade encontrada nos desenvolvimentos de projetos pessoais e profissionais, onde durante o processo de desenvolvimento não há um versionamento adequado das rotinas SQL<sup>1</sup> do banco de dados, causando atrasos em correções de ambiente devido ao desalinhamento do banco com a documentação do sistema. Com isso, propõe-se o desenvolvimento de um sistema informático que visa o controle e documentações de rotinas SQL para SGBD<sup>2</sup> MySQL através de um módulo de dicionário, que armazenará todas as rotinas cadastradas e será responsável por controlar a criação e destruição das rotinas cadastradas no banco de dados. No sistema proposto, além da visualização das rotinas do sistema, será possível acessar os históricos de modificações, validar as rotinas e permitir a criação de novas rotinas com base em modelos básicos. O sistema desenvolvido cumpre a proposta no quesito de controle de rotinas SQL para o SGBD MySQL, porém é desejado que futuramente, o sistema seja capaz de gerenciar todos os objetos do ambiente do banco de dados, incluindo índices, chaves estrangeiras e primárias, *view*, dentre outros objetos.

Palavras-Chave: Banco de Dados. Documentação. Controle de objetos.

---

1 SQL (*Structured Query Language*) – é uma linguagem padrão universal para manipular banco de dados relacionais;

2 SGBD (Sistema de Gerenciamento de Banco de Dados) – é um conjunto de *softwares* responsáveis pelo gerenciamento do banco de dados, utilizando-se principalmente de uma interface gráfica.

## ABSTRACT

The work developed focuses in supplying a need found in personal and professional projects developing, while during the development there has not been an appropriate versioning to database SQL<sup>3</sup> routines, causing a delay in environment fixing due database and documentation misalignment. With that in mind, it is proposed the development of a computer system that aims the control and document MySQL DSMS<sup>4</sup> SQL routines, through a dictionary module, that will store all registered routines and will be responsible for managing the routines creation and destruction registered in database. In this proposed system, beyond object visualization, it will be possible to access modification histories, to validate the database objects and to allow the creation of new objects based in a basic template. This developed proposed system meets the proposal, but is desired in the future, this system will be capable of managing all database objects, including indexes, foreign and primary keys, views, among other objects.

Keywords: Database. Documentation. Object control.

---

3 SQL (*Structured Query Language*) – it's an universal default language for manipulating relational databases;

4 DSMS (*Data Base Management System*) – it's a set of softwares responsible for managing the database, mainly using graphical interface.

## LISTA DE FIGURAS

Figura 1. Ilustra o diagrama de casos de uso implementados nesse projeto.....	18
Figura 2. Fluxo de atividade do login do usuário no sistema.....	27
Figura 3. Fluxo do processo de inserção de um novo usuário no sistema.....	27
Figura 4. Fluxo da inativação de um objeto do banco.....	28
Figura 5. Fluxo do processo de consulta de usuários.....	28
Figura 6. Fluxo de alteração de senha do usuário.....	29
Figura 7. Fluxo do cadastro de um novo objeto.....	30
Figura 8. Fluxo de atualização de um objeto.....	31
Figura 9. Fluxo do processo de validação de um objeto.....	32
Figura 10. Diagrama do MER (Modelo Entidade Relacionamento) do banco.....	34
Figura 11. Primeiro protótipo da tela de login do sistema.....	35
Figura 12. Primeiro protótipo dos menus do sistema.....	35
Figura 13. Primeiro conceito da tabela de controle de objetos.....	36
Figura 14. Primeiro conceito da tabela de controle de objetos em modo de inserção.....	37
Figura 15. Primeiro protótipo do menu de relatórios.....	37
Figura 16. Tela para visualização de relatórios gerados.....	38
Figura 17. Tela para seleção dos filtros que seriam utilizados para gerar os relatórios.....	39
Figura 18. Protótipo da tela de login.....	40
Figura 19. Protótipo da tela de objetos.....	40
Figura 20. Protótipo da tela de históricos.....	41
Figura 21. Tela de usuários.....	41
Figura 22. Tela de cadastro de usuários.....	42
Figura 23: Tela de alteração de senha.....	42
Figura 24. Protótipo da tela de objetos em modo de detalhe.....	43
Figura 25: Modelo de configuração do ambiente do software.....	45

## **LISTA DE ABREVIATURAS**

IDE – *Integrated Development Environment*

SGBD – Sistema de Gerenciamento de Banco de Dados

DSMS – *Data Base Management System*

UML – *Unified Modeling Language*

SQL – *Structured Query Language*

ANSI – *American National Standards Institute*

MER – Modelo Entidade Relacionamento

DER – Diagrama Entidade Relacionamento

SSL – *Secure Socket Layer*



## **SUMÁRIO**

<b>1. INTRODUÇÃO.....</b>	<b>9</b>
1.1. DESCRIÇÃO DO PROBLEMA.....	10
1.2. OBJETIVO/SOLUÇÃO.....	10
1.3. ESCOPO.....	10
1.4. VIABILIDADE DO PROJETO.....	11
1.5. METODOLOGIA DE TRABALHO.....	11
<b>2. TRABALHOS CORRELATOS EXISTENTES.....</b>	<b>12</b>
<b>3. REQUISITOS.....</b>	<b>13</b>
3.1. REQUISITOS FUNCIONAIS.....	13
3.2. REQUISITOS NÃO FUNCIONAIS.....	14
<b>4. DIAGRAMAS UML.....</b>	<b>16</b>
4.1. CASOS DE USO.....	17
4.2. DIAGRAMAS DE ATIVIDADES.....	26
<b>5. MODELAGEM DE DADOS.....</b>	<b>33</b>
<b>6. PROJETO DE INTERFACE.....</b>	<b>35</b>
<b>7. IMPLEMENTAÇÃO.....</b>	<b>44</b>
7.1. DESENVOLVIMENTO.....	44
7.2. CONFIGURAÇÃO.....	44
<b>8. RESULTADOS OBTIDOS.....</b>	<b>47</b>
<b>9. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS.....</b>	<b>48</b>
<b>10. REFERÊNCIAS.....</b>	<b>49</b>

## 1. INTRODUÇÃO

Durante o desenvolvimento de um *software* dentro de uma equipe, quantas vezes é identificado que há um problema no ambiente do sistema e que a solução é demorada e ineficaz devido a uma má documentação? Quantas vezes é necessário realizar manutenções em um *software* que não foi devidamente analisado e documentado, causando dores de cabeça e mais problemas?

Com o crescimento das empresas e dos projetos, que necessitam de mais e mais colaboradores, não é difícil nos depararmos com as respostas questionadas acima: muitas vezes, principalmente em grandes projetos envolvendo muitos colaboradores. Números e mais números de entrega são exigidos e devido a toda essa cobrança muitas vezes passam despercebidas as alterações que são realizadas no ambiente para testes no ambiente de desenvolvimento, comprometendo o desempenho e futuras manutenções do projeto, principalmente quando envolve o ambiente e análise de clientes.

No capítulo 1 desse documento serão tratados o problema, os objetivos propostos para solução do problema, definiremos o escopo e metodologia utilizada para desenvolvimento do projeto. No capítulo 2, verificaremos os trabalhos correlatos existentes, seguindo o capítulo 3, onde serão tratados os requisitos levantados para o desenvolvimento do sistema. No capítulo 4, serão apresentados os diagramas UML<sup>5</sup> que foram utilizados durante o desenvolvimento do projeto junto das suas respectivas descrições. A seguir, no capítulo 5, será apresentada a modelagem de dados desenvolvida. Em seguida, no capítulo 6 serão apresentadas as interfaces desenvolvidas e de seus respectivos conceitos, desde primeira ideia do projeto até a sua conclusão. Finalizando, no capítulo 7 serão apresentadas as informações de implementação do *software*, desde o arquivo de configuração até a rotina para teste do ambiente e do sistema. No capítulo 8, serão analisados os resultados obtidos com o desenvolvimento do projeto, seguindo com o capítulo 9, onde serão apresentadas as considerações finais e fechando o com o capítulo 10, onde estão apresentadas as referências utilizadas para desenvolvimento do projeto.

---

5 UML (*Unified Modeling Language*) – é uma linguagem padrão, baseada no desenvolvimento de diagramas, para a elaboração da estrutura de projetos de software.

### 1.1. DESCRIÇÃO DO PROBLEMA

Como descrito inicialmente, com o crescimento das empresas e grupos de desenvolvimento nas empresas de tecnologia, a demanda por customizações sempre é uma prioridade para a evolução da empresa, porém acompanhada dessas evoluções, vem muitas vezes uma má gestão do ambiente de desenvolvimento.

Durante o desenvolvimento de customizações para o cliente, muitos desenvolvedores acabam por fazer testes em suas bases locais e esquecem de validar os objetos que serão liberados para os clientes, gerando dessa forma, muitos obstáculos para identificar o real problema, principalmente para desenvolvedores menos experientes.

### 1.2. OBJETIVO/SOLUÇÃO

Visando resolver esse problema de alinhamento de rotinas e um melhor controle na documentação do sistema, propõe-se um sistema informático que fará o armazenamento e validações das rotinas SQL entre a aplicação e banco de dados, fornecendo para o usuário um controle visual dos objetos existentes, de suas modificações, usuários de criação e modificação, data de criação modificação e também uma validação dos objetos para garantir que estão sendo gerados sem erros.

### 1.3. ESCOPO

A aplicação desenvolvida nesse projeto é acessada por validação de *login* onde o usuário administrador poderá realizar o cadastro de novos usuários, controlar todos os objetos, inserindo, modificando, inativando e ativando, além de poder validar os objetos do banco. Haverá também um tipo de usuário padrão, que poderá apenas inserir e atualizar os objetos, além de realizar as validações das rotinas no banco, porém não poderá realizar a inativação e a ativação de nenhum objeto e nem cadastrar novos usuários. Ambos os usuários terão permissão para alterar a própria senha.

A aplicação está sendo desenvolvida apenas para aplicação *desktop* e não há planos futuros para desenvolvimento para *mobile* ou *cloud*.

#### 1.4. VIABILIDADE DO PROJETO

Esse projeto foi desenvolvido pensando em desenvolvedores e empresas de pequeno/médio porte que trabalhem com SGBD MySQL e que desejam um software para organizarem e controlarem as rotinas de SQL que existem no banco de dados. Dessa forma, eles poderão ter uma visão clara das rotinas que estão sendo modificadas no ambiente de desenvolvimento e garantir o que está ativo ou não no ambiente.

#### 1.5. METODOLOGIA DE TRABALHO

O desenvolvimento do trabalho, inicialmente, estava amparado nas necessidades pessoais identificadas no ambiente de trabalho, onde a qualidade do trabalho entregue estava sendo comprometida pelo desenvolvimento de outros colaboradores. Neste sentido, num primeiro momento se buscou realizar o levantamento de informações do tema, como isso poderia melhorar as entregas e customizações e garantir uma maior segurança e qualidade de *software*.

Na etapa seguinte, buscou-se identificar as tecnologias e ferramentas mais adequadas a serem empregadas no desenvolvimento do sistema. Devido à facilidade de uso entre plataformas e da familiaridade com a linguagem foi-se definido que a linguagem de desenvolvimento seria em Java, utilizando a IDE<sup>6</sup> NetBeans, e para o banco de dados, seria utilizado o SGBD MySQL Workbench.

O ambiente de desenvolvimento utilizado foi o NetBeans IDE 8.1 que é livre e de código aberto. Essa IDE possibilita o desenvolvimento em várias linguagens de programação, entre elas: Java, PHP, C, C++, entre outras. (NetBeans, 2018)

O SGBD para desenvolvimento foi o MySQL por ser um *software open source*, que apesar de também ter uma licença comercial, o código livre foi projetado inicialmente para atender aplicações de pequeno e médio porte, mas hoje já atende aplicações de grande porte.

---

6 IDE (*Integrated Development Environment*) – são softwares que agrupam ferramentas para apoiarem os desenvolvedores no desenvolvimento de *software*.

## 2. TRABALHOS CORRELATOS EXISTENTES

Atualmente existem *softwares* e recursos para controle de rotinas e objetos de bancos SQL, dos quais podemos listar:

- Oracle Scheduler: É uma ferramenta disponibilizada pela Oracle para agendamento de tarefas. O Oracle Scheduler permite “executar desde códigos de banco de dados, como *stored procedures* ou *packages*, até programas executáveis no sistema operacional, como shell scripts (no UNIX) ou arquivos de lote (*batch*) e ainda arquivos executáveis no MS Windows (os famosos .exe). “(REZENDE, 2012);
- Sistemas de Backup: Existem disponíveis no mercado diversos *softwares* de backup de banco de dados, como por exemplo o vendido pela EaseUS;
- Modelo para gerenciamento de banco de dados SQL através de *Stored Procedures*, desenvolvido por SPECIALSKI, E. S. para o trabalho de mestrado em Ciências da Computação na Universidade Federal de Santa Catarina<sup>7</sup>.

O Oracle Sheduler é uma ferramenta versátil que permite, dentre vários fatores, a realização de *backups* do banco de dados, o que é um válido para controle dos objetos e rotinas do banco de dados, porém não oferece um processo de versionamento, além de exigir um conhecimento das ferramentas e SGBDs em uso pelo desenvolvedor. O mesmo situação pode ser descrita para outros *softwares* de *backup* como o vendido pela EaseUS.

O modelo desenvolvido por Specialski supre uma questão não apresentada e não trabalhada neste documento, que é a possibilidade de gerenciar o banco de dados independente do SGBD que o usuário está utilizando, porém não dispõe de nenhum recurso visual e nem de versionamento para que o usuário identifique as modificações nas rotinas e nos objetos do banco de dados.

A proposta do *software* desenvolvido nesse projeto é permitir que o usuário tenha um controle das rotinas SQL do banco de dados de forma simples e visual. O *software* proposto visa armazenar as modificações, usuários e datas dos cadastros, além de poder ativar e inativar as rotinas do banco de dados e utilizar os registros como um *backup* no caso do objeto ter sido criado de forma errada ou ser necessário reverter alguma customização ou correção.

---

<sup>7</sup> O trabalho desenvolvido se encontra listado nas referências.

### 3. REQUISITOS

A engenharia de requisitos é uma forte ferramenta para traçar exatamente o que o sistema deverá fazer, os serviços a serem oferecidos e as limitações do mesmo. Abaixo está descrito os requisitos utilizados nesse projeto.

Importante definirmos que há requisitos funcionais e não funcionais. Como Sommerville escreve em seu livro *Engenharia de Software*, 9ª edição de 2011:

Requisitos funcionais são declarações de como serão as reações do sistema de acordo com as entradas, dos serviços que serão fornecidos e de como será seu comportamento em determinadas situações. Para requisitos não funcionais, o autor os interpreta como definições que não estão diretamente relacionados com os serviços oferecidos pelo sistema, são requisitos que fazem referência a propriedades do sistema, como confiabilidade e tempo de resposta.

#### 3.1. REQUISITOS FUNCIONAIS

RF001 – O sistema deverá permitir que usuários administrativos possam realizar o cadastro de novos usuários;

RF002 – O sistema deverá identificar o nível de acesso dos usuários para controle de exibição dos itens;

RF003 – O sistema deverá permitir que o usuário logado altere a própria senha;

RF004 – O sistema deverá possibilitar que usuários cadastrem objetos dos tipos *PROCEDURE*, *FUNCTION* e *TRIGGER*;

RF005 – O sistema deverá permitir que os usuários realizem a edição dos objetos cadastrados;

RF006 – O sistema deverá salvar todas as alterações que os usuários fizerem nos objetos cadastrados;

RF007 – O sistema deverá salvar um histórico de todos os objetos inseridos por usuários pela aplicação;

RF008 – O sistema deverá salvar um histórico de todos os objetos modificados por usuários pela aplicação;

RF009 – O sistema não deverá permitir a exclusão de objetos;

RF010 – O sistema não deverá permitir a exclusão de históricos;

RF011 – O sistema deverá permitir que usuários realizem a validação de objetos pela aplicação;

RF012 – O sistema deverá permitir que usuários de nível administrativo possam inativar objetos;

RF013 – O sistema deverá permitir que os usuários filtrem os objetos que desejam ver, sendo as opções: Todos, Apenas ativos, Apenas inativos;

RF014 – O sistema deverá permitir que usuários criem novos objetos utilizando um modelo básico;

RF015 – O sistema deverá permitir que o usuário defina a base que vai ser utilizada através de um arquivo de configurações;

RF016 – O sistema deverá permitir que o usuário consulte os outros usuários cadastrados no sistema;

RF017 – O sistema deverá exibir no título da janela do dicionário o usuário logado.

### 3.2. REQUISITOS NÃO FUNCIONAIS

RNF001 – O sistema deverá possuir uma conexão com o banco MySQL;

RNF002 – O sistema deverá ser executado no sistema operacional Windows;

RNF003 – O sistema deverá ser executado em um ambiente com Java 8 ou posterior instalado e configurado;

RNF004 – O sistema deverá trazer cem registros em até 3 segundos ao trocar de filtro ou iniciar a aplicação;

RNF005 – Em casos de erros, o sistema deverá exibir uma mensagem de aviso;

RNF006 – O sistema deverá ser executado em uma máquina com pelo menos 4GB de memória RAM;

RNF007 – Não deverão ser compartilhados o usuário e senha entre usuários;

RNF008 – O sistema não deverá armazenar nenhum log contendo as informações manipuladas pelo usuário entre a aplicação e o servidor.



## 4. DIAGRAMAS UML

A UML em algumas literaturas é tratada como um padrão de diagramação ou uma linguagem gráfica para visualização, construção, documentação de um projeto entre outras denotações. Para Fowler (2005),

UML é uma linguagem gráfica de modelagem que auxilia no detalhamento de um projeto de software, especialmente quando construídos no estilo orientado a objetos, é uma linguagem utilizada como esboço de como será o projeto para nossa melhor compreensão sobre o projeto em si, pois as linguagens de programação não encontram-se em um nível de abstração alto o suficiente para auxiliar nos estudos sobre o projeto.

Como lista CETA no *All You Need to Know About UML Diagrams: Types and 5+ Examples*, os diagramas UML podem ser divididos em quatorze tipos de diagramas. Abaixo estão os quatorze diagramas listados por CETA (traduzidos e adaptados, BARTHOLOMEU) e uma breve explicação sobre suas utilizações:

- Diagrama de atividades: representa o fluxo de atividades que podem ser executadas pelo sistema ou pelos atores;
- Diagrama de caso de uso: documenta o que o sistema faz. Não visa focar no como;
- Diagrama de visão geral de interação: representa o envio ou recebimento de dados entre um ator e um caso de uso;
- Diagrama de tempo: apresenta o comportamento dos objetos e suas interações em uma escala de tempo, focalizando as condições que mudam no decorrer desse período;
- Diagrama de máquina de estados: também conhecido como diagrama de estados, esse diagrama é utilizado para acompanhar as mudanças sofridas nos estados de uma instância de uma determinada classe;
- Diagrama de sequência: busca representar uma perspectiva, orientada por tempo, da colaboração entre objetos;
- Diagrama de comunicação: também conhecido como diagrama de colaboração, representa uma coleção de objetos que, em conjunto, trabalham para atender algum comportamento do sistema;

- Diagrama de classes: busca representar as classes do sistema e seus inter-relacionamentos;
- Diagrama de objetos: representa os objetos do software como se estivessem em tempo de execução, ou seja, com seus devidos valores e relacionamentos;
- Diagrama de componentes: representa uma coleção de componentes de software;
- Diagrama de estrutura composta: utilizado para descrever os relacionamentos entre os elementos. Também utilizado para descrever a colaboração interna de classes, interfaces ou componentes para especificar uma especialidade;
- Diagrama de implementação: representa uma coleção de componentes e mostra como é sua distribuição em um ou vários nós de hardware;
- Diagrama de pacotes: busca representar o sistema através de pacotes de dados e seus inter-relacionamentos;
- Diagrama de perfil: destina-se a criar uma visão do relacionamento entre classes para atender determinado domínio.

Neste documento serão apresentados dois diagramas UML: o diagrama de casos de uso e o diagrama de atividades.

#### 4.1. CASOS DE USO

Os casos de uso são representações de funcionalidades do sistema que necessitam de atores e outros casos de uso para serem operacionalizados, já o diagrama de casos de uso permite a visualização dos relacionamentos entre casos de uso, e entre atores (Melo, 2010).

O diagrama de caso de uso foi escolhido para ser apresentado nesse documento devido à visão geral que ele fornece do programa e das interações que os atores possuem com cada caso de uso.

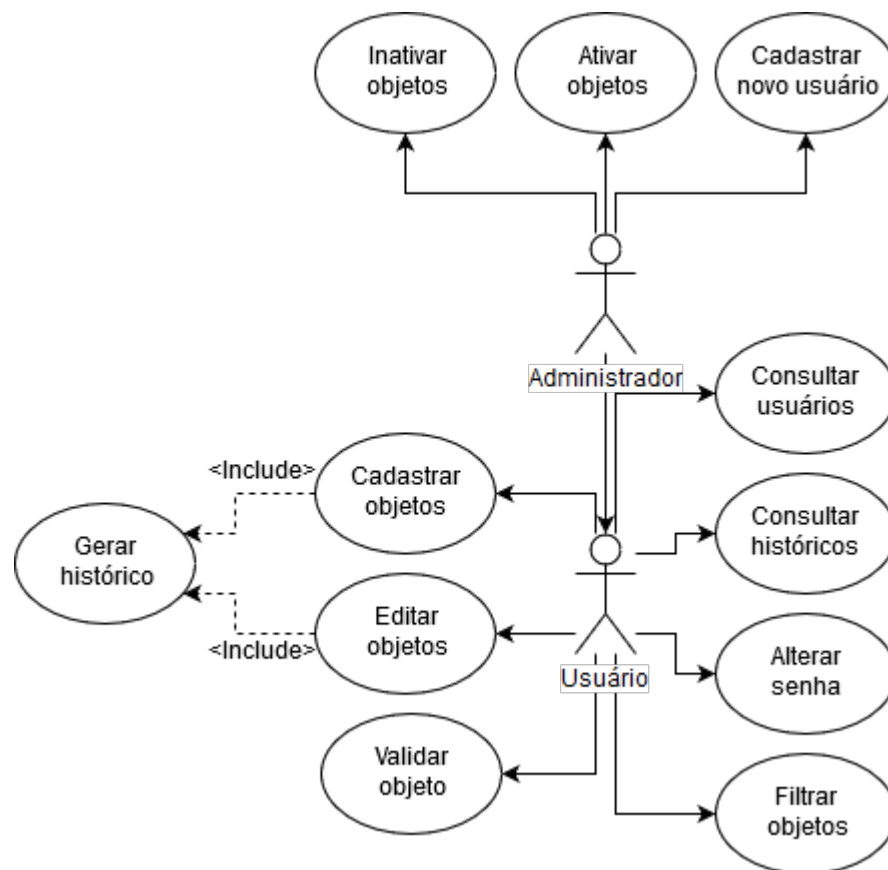


Figura 1. Ilustra o diagrama de casos de uso implementados nesse projeto.

Nome: **Inativar objetos**

Descrição: Inativar objeto cadastrado, removendo ele da base em uso.

Atores: Administrador.

Pré-condição: Estar conectado no sistema e possuir ao menos um objeto ativo no sistema.

Fluxo Normal:

1. O usuário conecta no sistema;
2. O usuário seleciona um registro ativo que deseja inativar;
3. Usuário clica no botão lateral “Inativar”;
4. O sistema exibe uma tela de confirmação. Caso positivo, segue com o processo;
5. O sistema apagará o objeto do banco de dados;
6. O sistema atualiza o identificador do objeto para inativo;
7. O sistema atualiza a tabela de objetos.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 2 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso;

Fluxo de exceção 3: Se no passo 5 o sistema der erro no processo de exclusão do objeto no banco, exibe uma mensagem de aviso;

Fluxo de exceção 4: Se no passo 6 o sistema der erro no processo de atualização do registro, exibe uma mensagem de aviso;

Fluxo de exceção 5: Se no passo 7 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso.

Nome: **Ativar objetos**

Descrição: Ativar objeto cadastrado, criando ele na base em uso.

Atores: Administrador.

Pré-condição: Estar conectado no sistema e possuir ao menos um objeto inativo no sistema.

Fluxo Normal:

1. O usuário conecta no sistema;
2. O usuário seleciona um registro inativo que deseja ativar;
3. Usuário clica no botão lateral “Ativar”;
4. O sistema exibe uma tela de confirmação. Caso positivo, segue com o processo;
5. O sistema criará o objeto no banco de dados do cliente;
6. O sistema atualiza o identificador do objeto para ativo;
7. O sistema atualiza a tabela de objetos.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 2 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso;

Fluxo de exceção 3: Se no passo 5 o sistema der erro no processo de criação do objeto no banco, exibe uma mensagem de aviso;

Fluxo de exceção 4: Se no passo 6 o sistema der erro no processo de atualização do registro, exibe uma mensagem de aviso;

Fluxo de exceção 5: Se no passo 7 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso.

Nome: **Cadastrar novo usuário**

Descrição: Realizar o cadastro de um novo usuário que utilizará o sistema.

Atores: Administrador.

Pré-condição: Possuir um usuário de carga inicial no sistema e estar conectado no sistema.

Fluxo Normal:

1. O usuário conecta no sistema;
2. O usuário deverá selecionar o menu “Usuários” no menu superior;
3. O usuário deverá selecionar o submenu “Listar usuários” para abrir a tela de usuários;
4. O usuário deverá, através do clique do botão direito, clicar no menu “Criar usuário”;
5. O sistema deverá abrir uma tela com os campos Usuário, Nome completo, Senha e Administrador;
6. O usuário deverá preencher todos os campos;
7. O usuário confirma a criação;
8. Caso todos os campos estejam fornecidos corretamente, o sistema inserirá um novo usuário no sistema com base nas informações fornecidas.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 3 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso;

Fluxo de exceção 3: Se no passo 8 o sistema der erro no processo de inserção do registros no banco, exibe uma mensagem de aviso.

Nome: **Consultar usuários**

Descrição: Acessar uma lista de todos os usuários cadastrados no sistema.

Atores: Administrador, Usuário.

Pré-condição: Estar conectado no sistema

Fluxo Normal:

1. O usuário conecta no sistema;
2. O usuário deverá selecionar o menu “Usuários” no menu superior;
3. O usuário deverá selecionar o submenu “Listar usuários” para abrir a tela de usuários;
4. O sistema deverá exibir uma tela com todos os usuários cadastrados.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 4 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso.

Nome: **Alterar senha**

Descrição: Alterar a senha do usuário logado.

Atores: Administrador, Usuário.

Pré-condição: Estar conectado no sistema.

Fluxo Normal:

1. O usuário conecta no sistema;
2. O usuário deverá selecionar o menu “Usuários” no menu superior;
3. O usuário deverá selecionar o submenu “Trocar senha” para abrir a tela de alteração de senha;
4. O sistema abrirá uma tela com a as os campos Senha atual e Nova senha;
5. O usuário deverá preencher os campos com sua senha atual e sua nova senha desejada respectivamente;
6. O usuário confirma que deseja alterar a senha;
7. Caso todos os campos estejam preenchidos, a senha atual esteja de acordo com a do sistema e caso a nova senha seja diferente da senha atual, o sistema atualizará a senha do usuário conforme informado.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 7 o sistema der erro no processo de inserção do registros no banco, exibe uma mensagem de aviso.

**Nome: Cadastrar objetos**

**Descrição:** Inserir novas rotinas no sistema.

**Atores:** Administrador, Usuário.

**Pré-condição:** Estar conectado no sistema.

**Fluxo Normal:**

1. Usuário conecta no sistema;
2. Usuário acessa a aba “Objetos”;
3. Usuário clica no botão lateral “Novo”;
4. O usuário preenche todas as informações sendo elas: Nome objeto, Tipo objeto e preenche o SQL;
5. Usuário clica em “Salvar”;
6. O sistema valida o objeto em inserção;
7. Sistema salva um histórico da inserção;
8. Atualiza a tabela de objetos.

**Fluxo de exceção:**

**Fluxo de exceção 1:** Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

**Fluxo de exceção 2:** Se no passo 4 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso;

**Fluxo de exceção 3:** Se no passo 6 o sistema der erro no processo de criação do objeto no banco, exibe uma mensagem de aviso e não salva registro na tabela;

**Fluxo de exceção 4:** Se no passo 8 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso.

**Nome: Editar objetos**

**Descrição:** Editar o conteúdo de rotinas já existentes no sistema.



Atores: Administrador, Usuário.

Pré-condição: Estar conectado no sistema e possuir ao menos um registro ativo para edição.

Fluxo Normal:

1. Usuário conecta no sistema;
2. O usuário seleciona o objeto que deseja editar;
3. Usuário clica no botão lateral “Editar”;
4. Usuário edita o conteúdo do objeto;
5. Usuário clica no botão lateral “Salvar”;
6. O sistema valida o objeto em edição;
7. O sistema salva um histórico da modificação.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 6 o sistema der erro no processo de validação do objeto, exibe uma mensagem de aviso. O erro pode ser no processo de exclusão do objeto do banco de dados ou no de criação.

Nome: **Gerar histórico**

Descrição: Gerar um histórico no sistema quando houver inserção ou modificação de objetos no sistema.

Atores: Sistema.

Pré-condição: Criar ou editar um objeto do sistema.

Fluxo Normal:

1. Usuário conecta no sistema;
2. Usuário clica no botão lateral “Novo” ou “Editar”;
3. Usuário cria/edita um objeto do banco;

4. Usuário clica no botão lateral “Salvar”;
5. Sistema valida o objeto em criação/edição;
6. Sistema salva um histórico da modificação.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 5 o sistema der erro no processo de validação do objeto, exibe uma mensagem de aviso. O erro pode ser no processo de exclusão do objeto do banco de dados ou no de criação.

Nome: **Validar objetos**

Descrição: Insere o objeto na base caso ele seja válido.

Atores: Administrador, Usuário.

Pré-condição: Estar conectado no sistema e possuir ao menos um registro cadastrado e selecionado.

Fluxo Normal:

1. Usuário conecta no sistema;
2. Usuário seleciona o objeto que deseja validar;
3. Usuário clica no botão lateral “Detalhe”;
4. Usuário através, do clique do botão direito, seleciona o menu “Validar objeto”;
5. O sistema valida o objeto apagando-o da base e criando-o novamente;
6. Exibe uma mensagem de aviso caso a validação tenha sido realizada com sucesso.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 5 o sistema der erro no processo de validação do objeto, exibe uma mensagem de aviso. O erro pode ser no processo de exclusão do objeto do banco de dados ou no de criação.

Nome: **Filtrar objetos**

Descrição: Filtrar os objetos que estão sendo exibidos na tabela de objetos.

Atores: Administrador, Usuário.

Pré-condição: Estar conectado no sistema.

Fluxo Normal:

1. Usuário conecta no sistema;
2. Usuário, através do clique do botão direito, seleciona o menu “Filtros”;
3. No menu “Filtros” o usuário deverá selecionar o submenu de qual filtro deseja usar, sendo eles: Todos, Apenas ativos, Apenas inativos;
4. O sistema recarrega a tabela de objetos com base no filtro selecionado pelo usuário.

Fluxo de exceção:

Fluxo de exceção 1: Se no passo 1 o sistema não conseguir conectar no banco de dados, exibe uma mensagem de aviso;

Fluxo de exceção 2: Se no passo 4 o sistema der erro no processo obtenção dos registros da tabela de objetos, exibe uma mensagem de aviso.

## 4.2. DIAGRAMAS DE ATIVIDADES

O diagrama de atividades apresenta o fluxo de uma atividade para outra, é uma visualização dinâmica do sistema e dão ênfase ao fluxo de controle na execução de um comportamento (Booch & Rumbaugh & Jacobson, 2005).

O diagrama de atividades foi selecionado para apresentação nesse documento, pois permite uma visão mais específica do comportamento de cada processo do sistema.

A figura 2 exibe como é feita a validação de *login* do usuário no sistema.

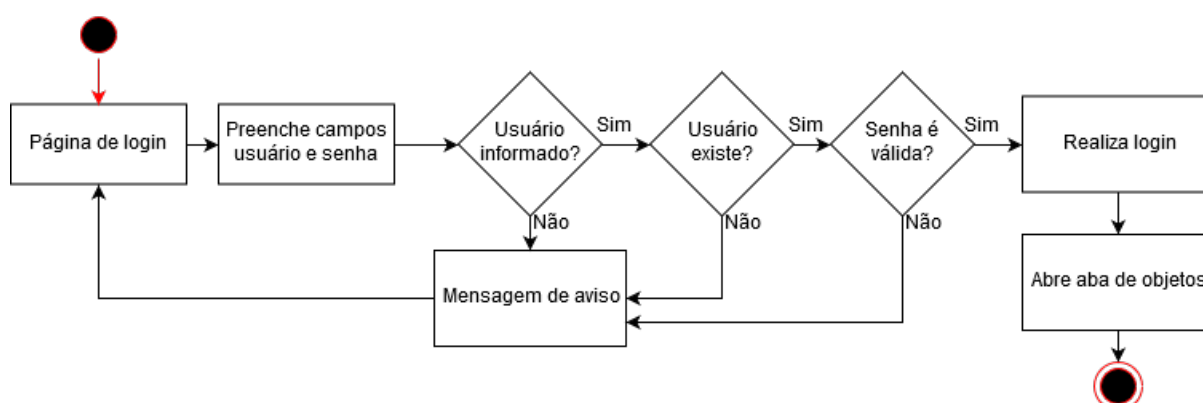


Figura 2. Fluxo de atividade do login do usuário no sistema.

O cadastro de usuários, representado na figura 3, solicita ao usuário logado as informações básicas como nome de usuário, nome completo e senha para login. Não será permitido o cadastro de dois usuários com o mesmo nome de usuário, além de ser obrigatório preencher todos os campos.

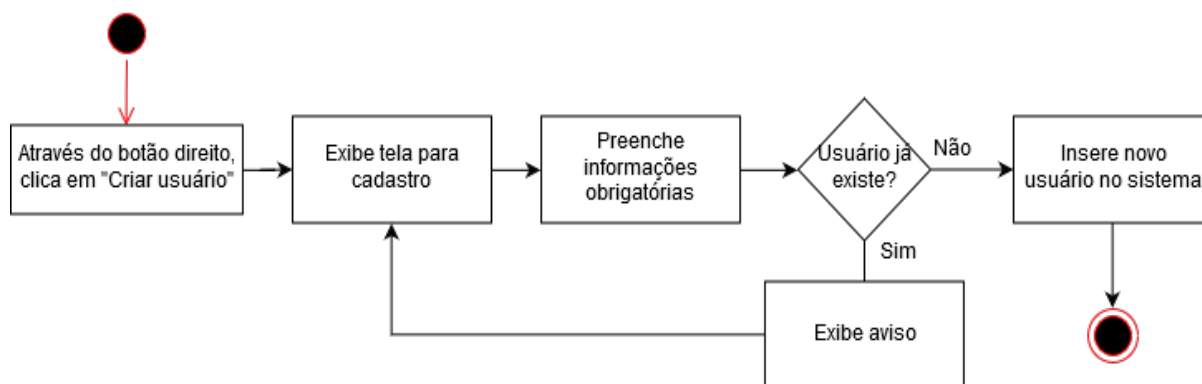


Figura 3. Fluxo do processo de inserção de um novo usuário no sistema.

O usuário administrador pode querer inativar objetos do sistema. Não foi desenvolvido uma exclusão de objetos devido ao controle de documentações. A figura 4 mostra o fluxo de inativação de um objeto.

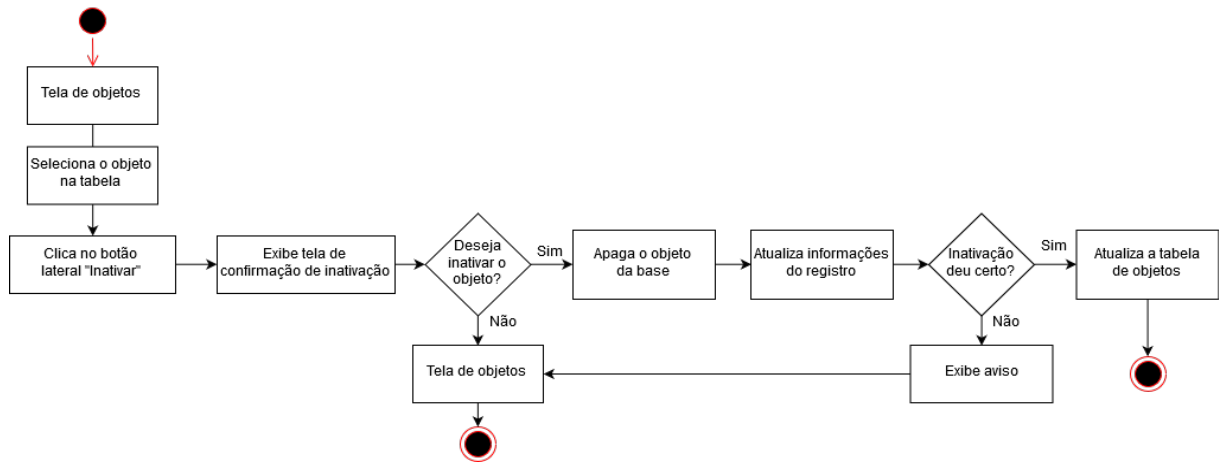


Figura 4. Fluxo da inativação de um objeto do banco.

A figura 5 mostra o processo de consulta de usuários no sistema, não demandando nenhum processamento pesado do sistema.

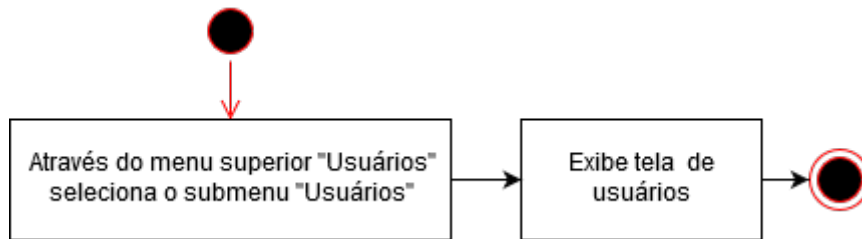


Figura 5. Fluxo do processo de consulta de usuários.

A

figura 6 mostra o fluxo de alteração de senha do usuário. O sistema não permitirá que o usuário tente modificar a senha do sistema para a senha atual, além de obrigar o preenchimento de todos os campos.

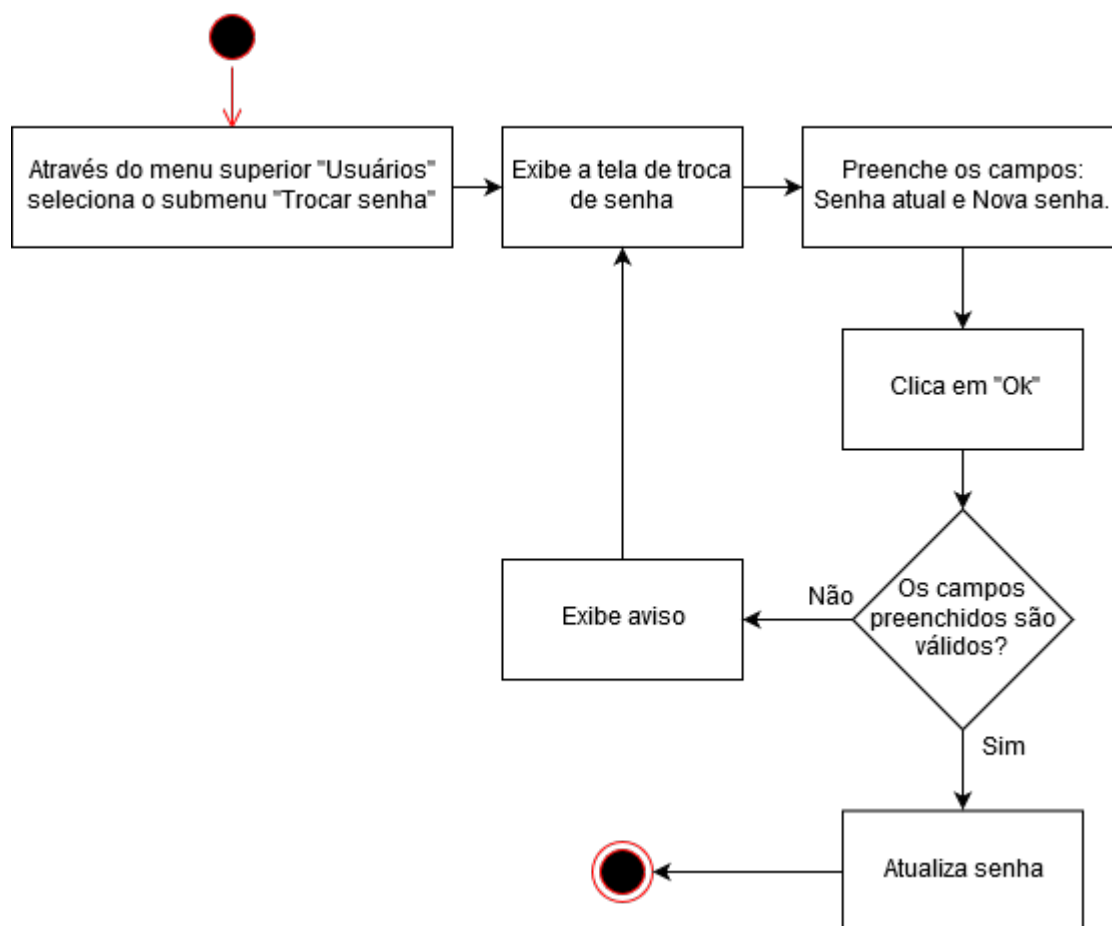


Figura 6. Fluxo de alteração de senha do usuário.

A figura 7 mostra o processo para salvar um novo objeto no sistema. O usuário pode escolher fazer um objeto do zero ou escolher um modelo padrão para inserir ele no sistema. Após salvar, o sistema deverá validar se o objeto em criação foi criado no banco de dados ou não.

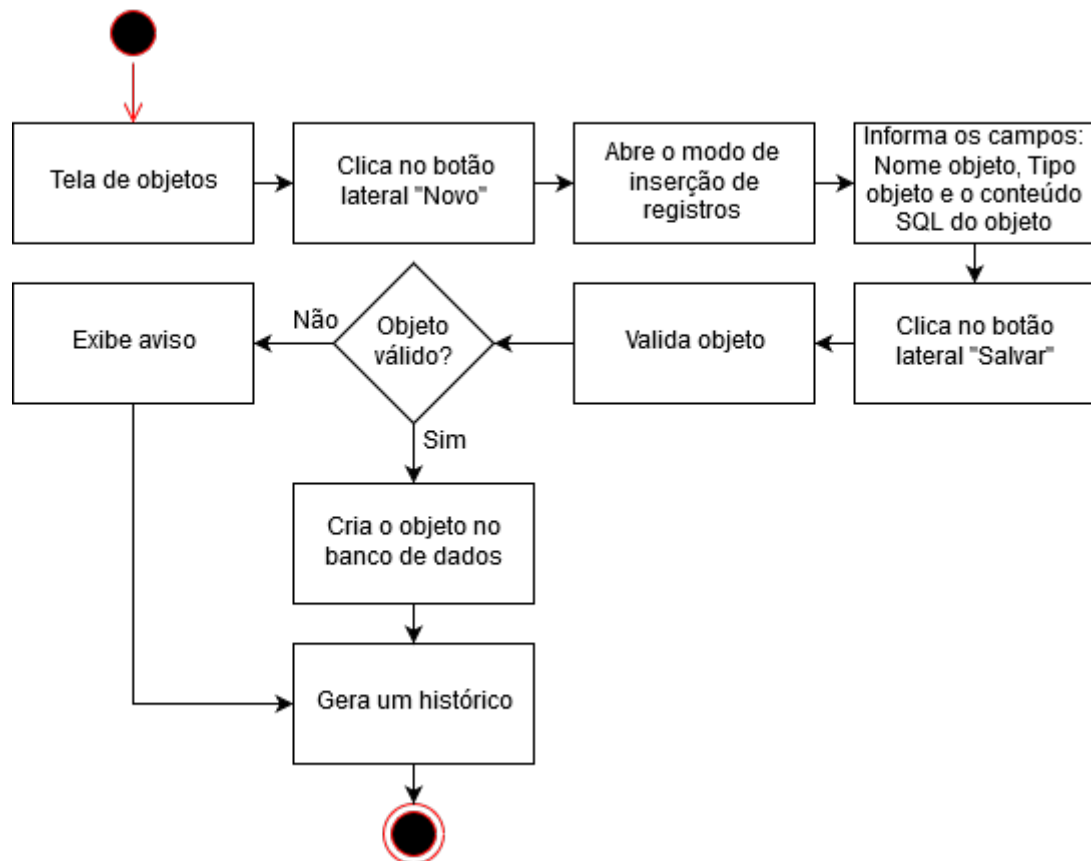


Figura 7. Fluxo do cadastro de um novo objeto.

A figura 8 apresenta o processo de edição dos objetos, que segue o mesmo princípio do processo de inserção, exceto pelo fator de o objeto já existir no sistema.

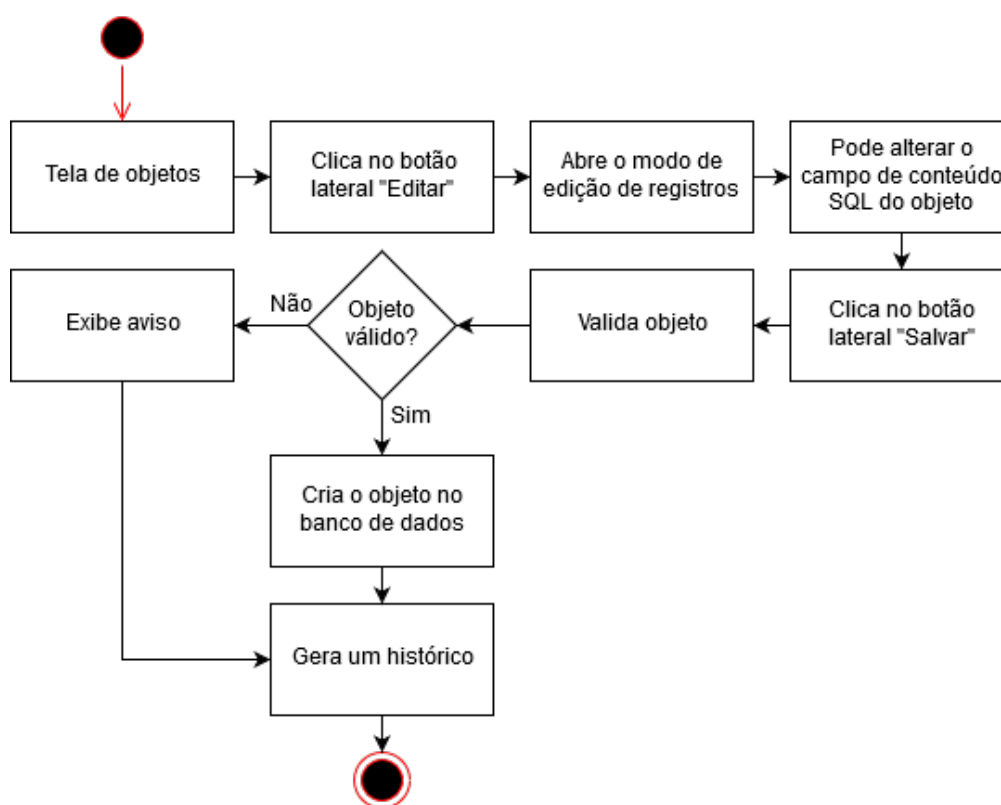


Figura 8. Fluxo de atualização de um objeto.

O processo de validação, exibido na figura 9, consiste na etapa final do processo de salvamento de um objeto, que é a validação do mesmo e a exibição de uma mensagem de erro caso o objeto esteja não tenha sido excluído ou salvo corretamente do banco de dados.



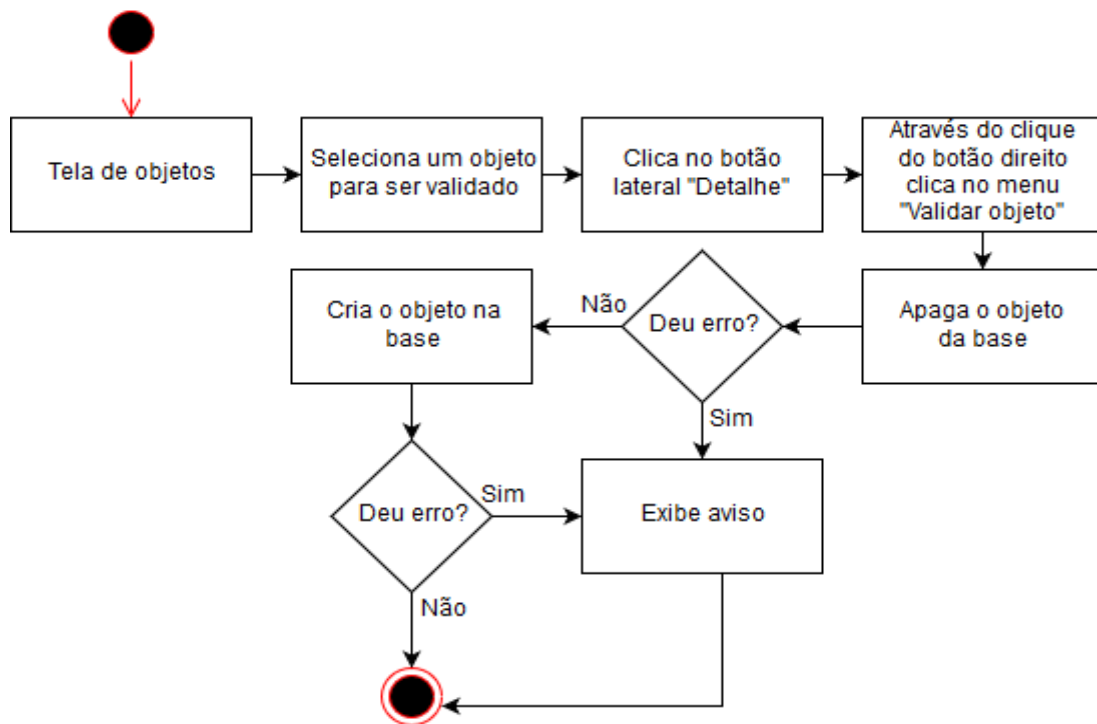


Figura 9. Fluxo do processo de validação de um objeto.

## 5. MODELAGEM DE DADOS

Para iniciarmos essa fase, consideramos importante a apresentação de conceitos básicos, mas importantes para compreensão dos termos.

Segundo Ramakrishnan e Gehrke(2008), “um banco de dados é uma coleção de dados que, tipicamente, descreve as atividades de uma ou mais organizações relacionadas”. O mesmo autor define que um sistema de gerenciamento de banco de dados, ou SGBD, é um *software* projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados.

Como Xavier e Pereira descrevem no livro *Conceitos às Consultas Complexas*, de 2009, “para banco de dados relacional existe a *Structured Query Language* (SQL), ou Linguagem de Consulta Estruturada”. É uma linguagem de pesquisa declarativa inicialmente criada pela IBM. Em 1986, a *American National Standards Institute* (ANSI) trabalhou para que fosse criado e adaptado um padrão para a linguagem, a ISO também colaborou em 1987. Apesar de padronizada possui variações e extensões produzidas por fabricantes de SGBDs, entretanto, normalmente pode migrar entre plataformas sem precisar alterar suas principais estruturas. Os recursos disponibilizados por essa linguagem são: consultas (*SELECT*), atualizações (*UPDATE*), filtros (*WHERE*) e ordenações (*ORDER BY*) (Milani, 2016).

Esse projeto consiste no gerenciamento dos objetos do banco do usuário, assim como na validação e controle dos mesmos, ou seja, uma vasta gama de dados para podermos realizar operações a fim de processar as informações pertinentes para garantirmos o sucesso no alcance de nossos objetivos. Dessa forma, há um conjunto de dados a ser trabalhado, portanto, precisamos de um banco de dados.

Para o gerenciamento do banco de dados do projeto, foi utilizado o SGBD MySQL e a modelagem foi realizada a através da ferramenta MySQL Workbench 6.3, que fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes para configuração do servidor (MySQL, 2018).

As definições de quais tabelas seriam necessárias, quais atributos deveriam constar na base de dados que se deu com base em análises em todos os requisitos de sistemas. A figura 10 mostra o DER do sistema.

As tabelas são:

APPLICATION\_USER: Armazena todos os dados dos usuários do sistema;

DICTIONARY: Armazena o conteúdo de todos os objetos do banco;

DICTIONARY\_HISTORY: Armazena os históricos de inserção e alteração de todos os objetos do banco;

OBJECTS\_TYPES: Armazena os tipos de objeto que o sistema tratará.

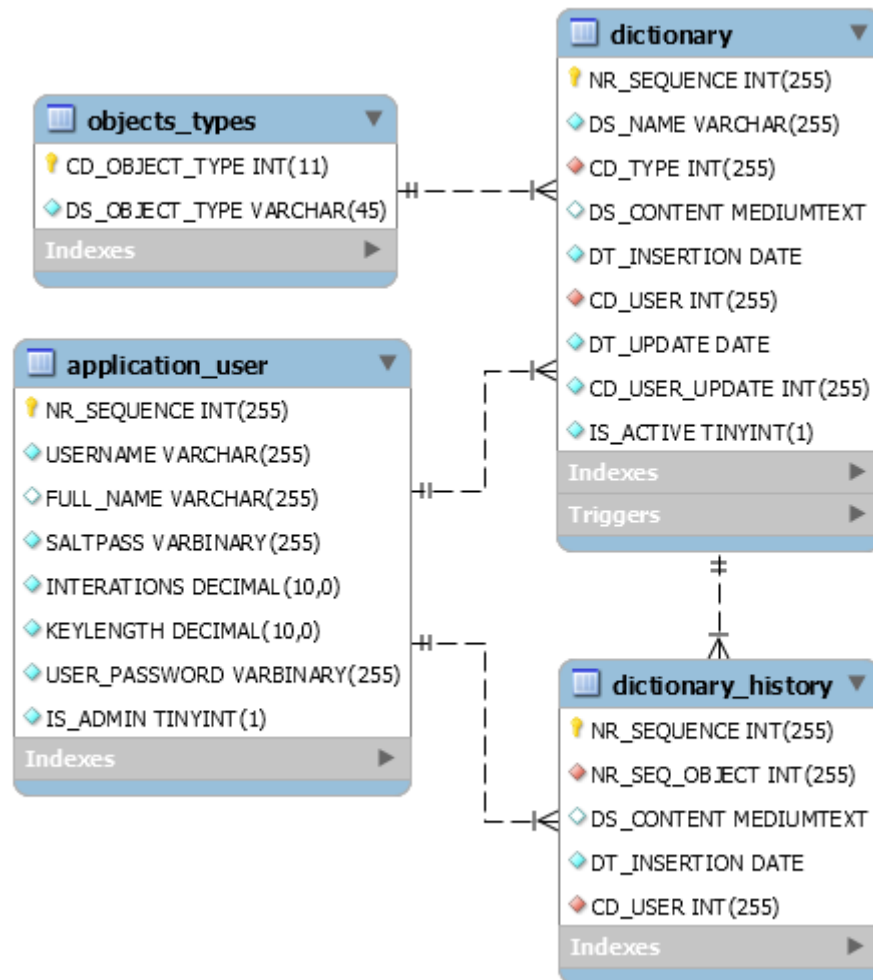


Figura 10. Diagrama do MER (Modelo Entidade Relacionamento) do banco.

## 6. PROJETO DE INTERFACE

Em um primeiro momento, o sistema havia sido projetado para ser focado apenas na geração de relatórios das modificações dos objetos no sistema, devido a isso o sistema foi conceituado para ser de uso intuitivo e claro, para que o usuário não tivesse problemas com botões e telas muito complexas e pesadas.

Abaixo estão os primeiros protótipos desenvolvimentos no *software* Axure RP 8, um *software* especializado em interfaceamento de projetos que apesar de ser pago por assinatura, disponibiliza de uma versão *trial* por 30 dias.

A figura 11 representa a primeira tela de login desenvolvida.

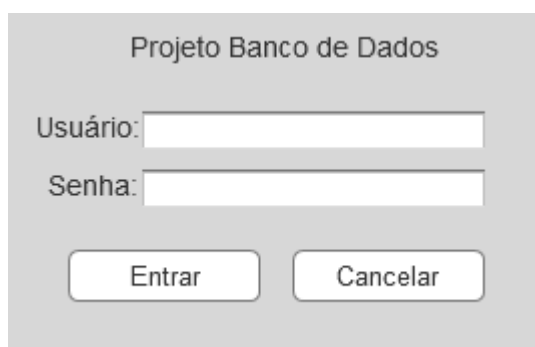
A imagem mostra um protótipo de uma tela de login. No topo, há o título "Projeto Banco de Dados". Abaixo dele, há dois campos de entrada: "Usuário:" seguido de um campo de texto e "Senha:" seguido de um campo de texto. Na base da tela, há dois botões: "Entrar" e "Cancelar".

Figura 11. Primeiro protótipo da tela de login do sistema.

A figura 12 apresenta o primeiro modelo de menus do sistema. O primeiro menu foi conceituado para dividir o dois principais módulos do sistema.

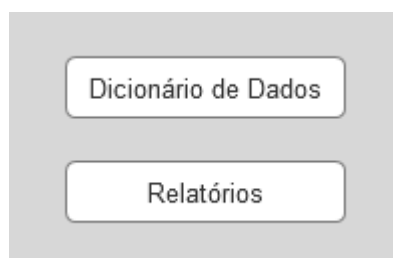
A imagem mostra um protótipo de uma tela com dois botões de menu. O botão superior é rotulado "Dicionário de Dados" e o botão inferior é rotulado "Relatórios".

Figura 12. Primeiro protótipo dos menus do sistema.

A figura 13 é o primeiro modelo de dicionário conceito. Ele foi desenvolvido de uma forma a exibir apenas as informações mais importantes para o sistema, sendo elas o nome do objeto cadastrado, o tipo do objeto, o usuário de criação do objeto, data de criação do objeto, último usuário a modificar o objeto e a data no qual foi modificado.

Filtro:

Nome	Tipo	Usuário Criação	Data Criação	Usuário Modificação	Data modificação
CONSISTIR_OBJETOS	PROCEDURE	gpbarholomeu	16/05/2017	gpbarholomeu	16/05/2017
VALIDAR_OBJETOS	PROCEDURE	csqueiroz	15/04/2017	gpbarholomeu	16/05/2017
OBTER_FRASES_SIS	FUNCTION	gpbarholomeu	01/01/2016	csqueiroz	02/01/2017

Detalhe

Novo

Salvar

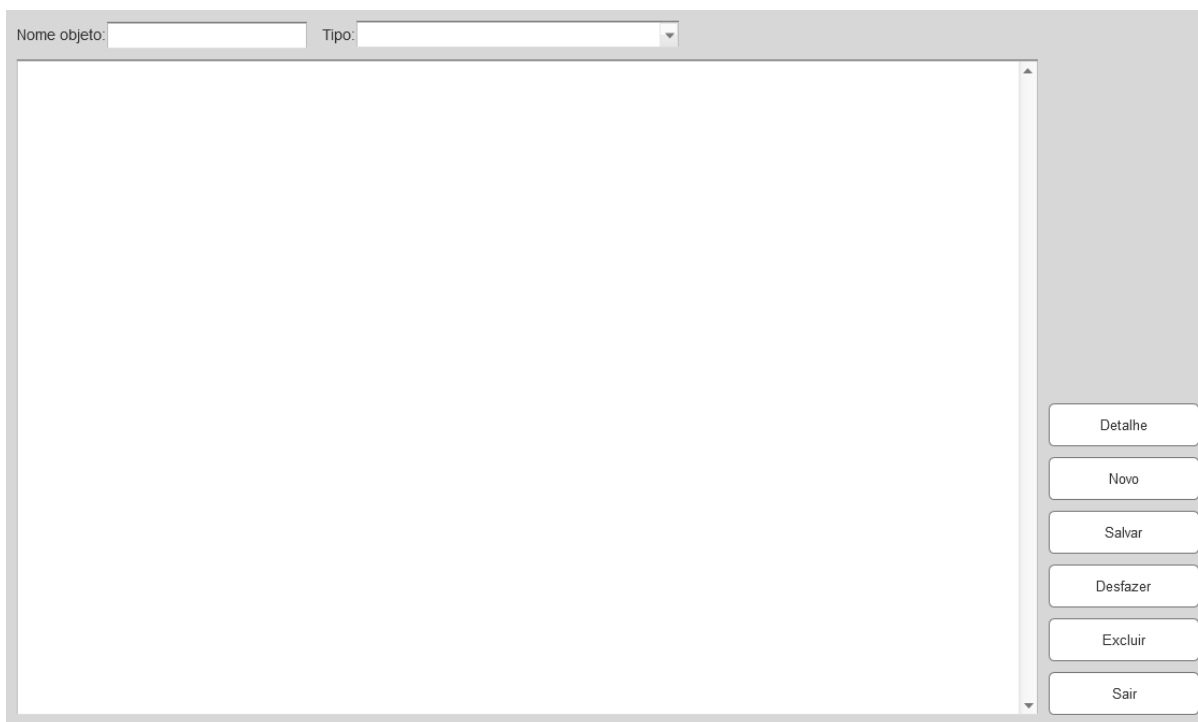
Desfazer

Excluir

Sair

Figura 13. Primeiro conceito da tabela de controle de objetos.

A figura 14 representa o sistema em modo de inserção de registros.



The image shows a software interface for object control. At the top, there are two input fields: 'Nome objeto:' followed by a text box, and 'Tipo:' followed by a dropdown menu. Below these is a large, empty rectangular area with a vertical scrollbar on the right side. To the right of this area is a vertical stack of six buttons: 'Detalhe', 'Novo', 'Salvar', 'Desfazer', 'Excluir', and 'Sair'.

Figura 14. Primeiro conceito da tabela de controle de objetos em modo de inserção.

A figura 15 mostra o submenu do módulo de relatórios, separando os dois principais usos: Listagem de relatórios gerados e gerar novos relatórios.

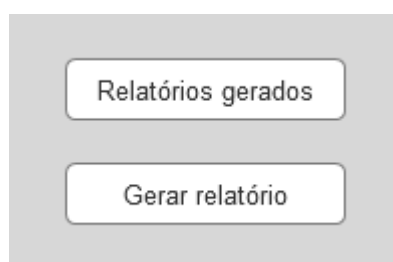


Figura 15. Primeiro protótipo do menu de relatórios.

A figura 16 mostra como deveria ser o módulo de relatórios gerados. Consiste no diretório no qual o relatório foi salvo, o usuário quem gerou o relatório e a data em que o relatório foi gerado.

Caminho arquivo	Usuário geração	Data geração
\\10.201.3.122\Relatórios\procedure_gpbartholomeu	gpbartholomeu	01/06/2017
\\10.201.3.122\Relatórios\procedure_csqueiroz	csqueiroz	01/06/2017
\\10.201.3.122\Relatórios\procedure_rfpadilha	gpbartholomeu	02/06/2017

Figura 16. Tela para visualização de relatórios gerados.

A figura 17 representa a tela de geração de relatórios. Nela o usuário deveria selecionar quais dos objetos listados realmente foram modificados por ele, definir um intervalo de busca para o sistema buscar os objetos modificados e adicionar uma descrição do motivo da modificação dos relatórios que seria agregado no relatório final.

**Procedures:**  
☐ CONSISTIR\_BASE  
☐ VALIDAR\_OBJETOS

**Scripts:**  
☐ 1021 - Função Laudos  
☐ 12920 - Função Sistemas

**Descrição:**  
De:  Para:

Figura 17. Tela para seleção dos filtros que seriam utilizados para gerar os relatórios.

Durante o desenvolvimento do projeto principal, foi-se descontinuado o módulo de geração de relatórios devido ao trabalho necessário para criação das telas em Java Swing, vínculo com a biblioteca IText e vínculo com o *software* JasperReport. O conceito do módulo de relatório foi salvo para desenvolvimentos futuros.

Mantendo a proposta de um sistema amigável, intuitivo e de fácil uso para todos os tipos de usuário, os protótipos a seguir foram desenvolvimentos na plataforma Netbeans durante o desenvolvimento do *software*, buscando uma visão limpa e clara dos objetos, dos menus e botões do sistema.

A figura 18 apresenta o novo protótipo da tela de *login*, se mantendo fiel ao primeiro conceito apresentado, na figura 11.



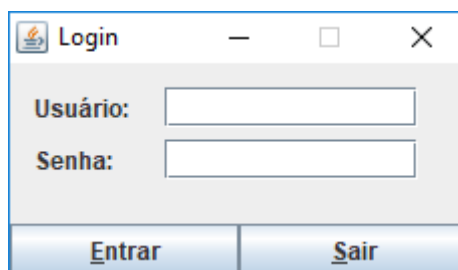

 A imagem mostra um protótipo de uma janela de login. No topo, há uma barra de título com o ícone de uma chave e o texto "Login", seguido por botões de minimizar, maximizar e fechar. Abaixo, há dois campos de entrada: "Usuário:" e "Senha:". Na base da janela, há dois botões: "Entrar" e "Sair".

Figura 18. Protótipo da tela de login.

A tela de objetos, apresentada na figura 19, foi modificada para se adequar ao componente Jtable (componente de tabela utilizado para exibir os registros inseridos no banco de dados), porém mantendo a essência dos itens a serem visualizados, através das colunas: Sequência, Nome do objeto, Tipo do objeto, Data inserção, Usuário inserção, Data atualização, Usuário atualização e Ativo.

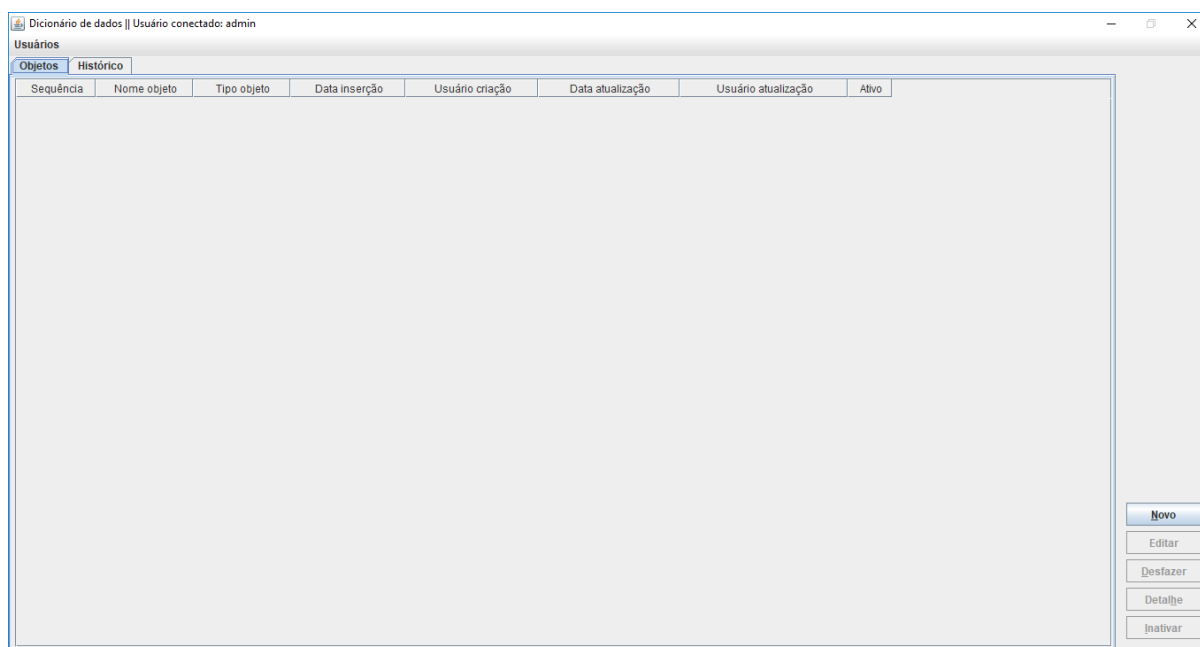

 A imagem mostra um protótipo de uma janela de gerenciamento de objetos. No topo, há uma barra de título com o texto "Dicionário de dados | Usuário conectado: admin". Abaixo, há uma barra de menu com "Objetos" e "Histórico". A principal área da janela é ocupada por uma tabela com as seguintes colunas: "Sequência", "Nome objeto", "Tipo objeto", "Data inserção", "Usuário criação", "Data atualização", "Usuário atualização" e "Ativo". Na parte inferior direita, há uma barra de ferramentas com os botões "Novo", "Editar", "Desfazer", "Detalhe" e "Inativar".

Figura 19. Protótipo da tela de objetos.

Na figura 20 é exibida a aba de históricos, que foi desenvolvida como forma de consulta e controle. A aba de históricos exibirá a sequência do histórico, data do histórico, o usuário responsável pela inserção/modificação e o conteúdo SQL da modificação que ocorreu naquele objeto.

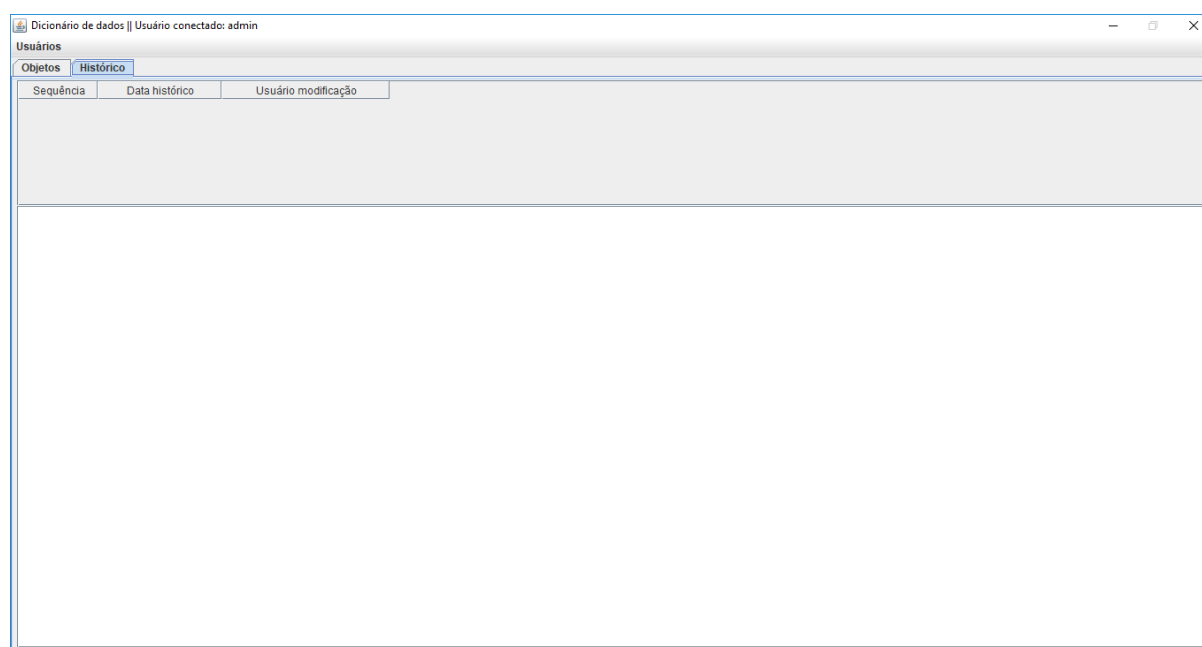


Figura 20. Protótipo da tela de históricos.

A figura 21 mostra a tela de usuários, a qual não havia sido conceituada no primeiro modelo do projeto. A tela de usuários não tem nenhum propósito de controle avançado, sendo utilizada para listar todos os usuários cadastrados no sistema e seus respectivos níveis de acesso. Através da tela de usuários também é possível cadastrar novos usuários.

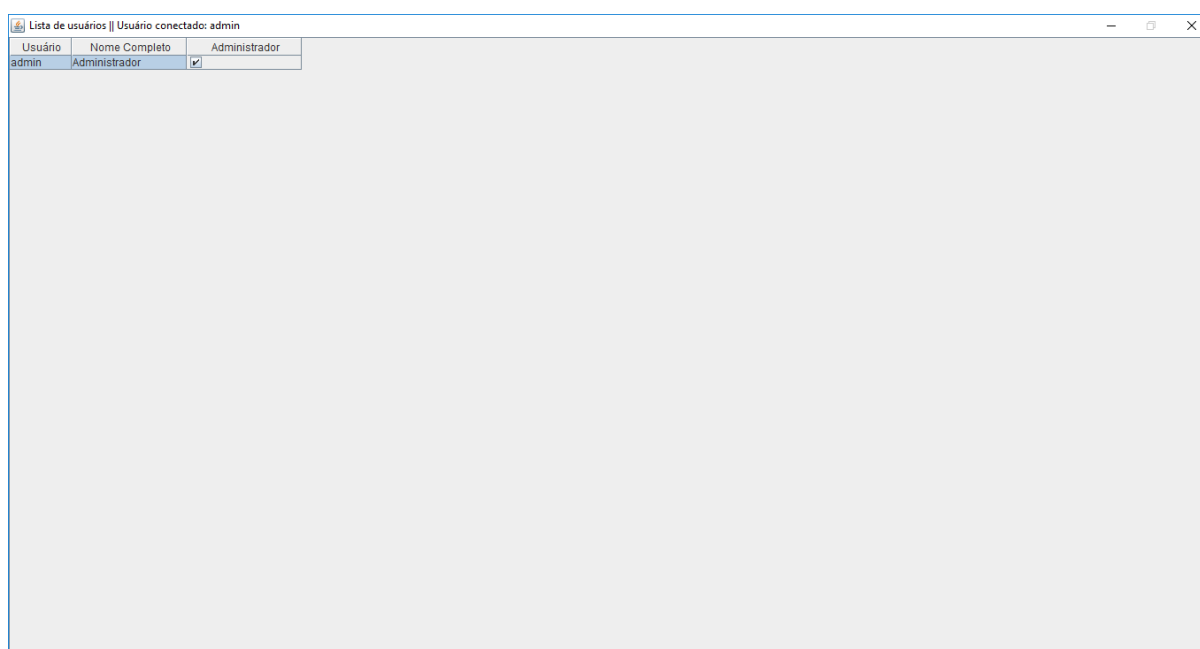


Figura 21. Tela de usuários.

Na figura 22 é apresentada a tela de cadastro de novos usuários.

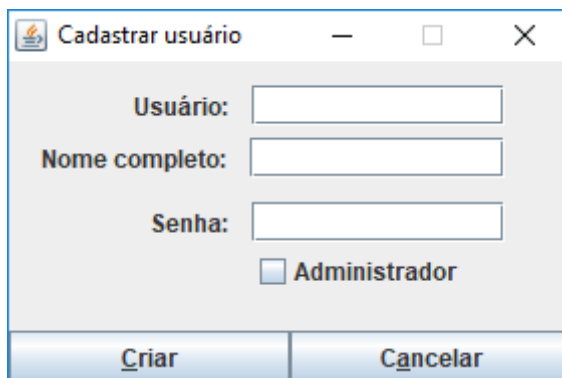
A imagem mostra uma janela de software intitulada "Cadastrar usuário". Ela possui uma barra de título com ícone, texto e botões de minimizar, maximizar e fechar. O corpo da janela contém três campos de texto rotulados "Usuário:", "Nome completo:" e "Senha:". Abaixo do campo "Senha:" há uma caixa de seleção com o rótulo "Administrador". Na base da janela, há dois botões: "Criar" e "Cancelar".

Figura 22. Tela de cadastro de usuários.

A figura 23 apresenta a tela de alteração de senhas, o qual é acessada pelo menu principal "Usuários".

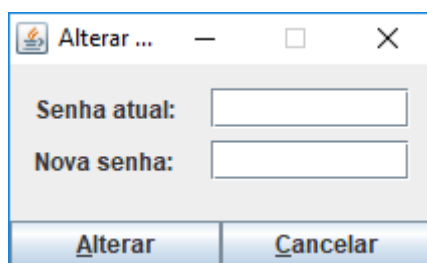
A imagem mostra uma janela de software intitulada "Alterar ...". Ela possui uma barra de título com ícone, texto e botões de minimizar, maximizar e fechar. O corpo da janela contém dois campos de texto rotulados "Senha atual:" e "Nova senha:". Na base da janela, há dois botões: "Alterar" e "Cancelar".

Figura 23: Tela de alteração de senha.

A figura 24 apresenta o sistema no modo de visualização de um objeto já registrado em modo de detalhe.

The screenshot shows a web application window titled "Dicionário de dados | Usuário conectado: admin". The window has a tabbed interface with "Objetos" and "Histórico" tabs. The "Objetos" tab is active, displaying a form for editing a database object. The form includes fields for "Nome objeto:" (PrimeiroObjeto), "Tipo objeto:" (PROCEDURE), "Data inserção:" (01/09/2018 17:07:42), and "Usuário criação:" (admin). Below these fields is a large text area containing SQL code: 

```
CREATE PROCEDURE 'PrimeiroObjeto' ()  
  
BEGIN  
  
SELECT 1 FROM DUAL;  
  
END;
```

 On the right side of the window, there is a vertical toolbar with five buttons: "Novo", "Editar", "Desfazer", "Grid", and "Inativar".

Figura 24. Protótipo da tela de objetos em modo de detalhe.

## 7. IMPLEMENTAÇÃO

### 7.1. DESENVOLVIMENTO

O *software* do projeto foi todo desenvolvido em Java, utilizando a interface Java Swing para desenvolvimento das telas. O banco de dados foi desenvolvido para o SGBD MySQL.

Para o desenvolvimento da interface, foram utilizados os componentes do Java Swing: JFrame, JPanel, JLabel, JTextField, JPasswordField, JTable, JTextArea, JButton, JMenuBar, JMenu, JTabbedPane, JScrollPane, JOptionPane e JPopupMenu. Todas as telas e janelas foram desenvolvidas utilizando a IDE Netbeans, que conta com a geração de arquivos .form (responsáveis por salvar as configurações e propriedades visuais das telas).

Para permitir que a aplicação seja aberta em modo de tela cheia porém sem sobrepor a barra de tarefas do sistema operacional, foi utilizada a biblioteca **SunGraphicsEnvironment**, responsável pelo ambiente gráfico da aplicação. Utilizando essa biblioteca foi possível obter a área “disponível” na tela do usuário, para poder definir o tamanho da janela da aplicação para preencher a tela todo sem sobrepor a barra de tarefas

A aplicação se comunica com o servidor através do conector **mysql-connector-java-8.0.11.jar**, disponibilizado pela própria Oracle no site do MySQL. É necessário vincular a aplicação ao conector, e durante o processo de abertura da conexão, é inicializado o *driver* do MySQL em uma nova instância.

Depois do *driver* ter sido carregado pela aplicação, são passadas as informações do banco de dados com o qual a aplicação se conectará. É necessário passar pelo menos o ip do servidor, a porta do ip, o *schema* do banco de dados que será utilizado, o usuário com o qual a aplicação se conectará no banco e a senha que será utilizada. É possível passar outras informações como a localização (fuso horário) e se a comunicação aplicação-servidor utilizará o certificado SSL<sup>8</sup>.

### 7.2. CONFIGURAÇÃO

A configuração do *software* deve seguir o seguinte processo:

---

8 SSL (*Secure Socket Layer*) é certificado que permitirá que a aplicação se comunique com o servidor utilizando criptografia, codificando os dados que são enviados e recebidos pelo servidor.

- Configuração do ambiente Java;
- Configuração do banco de dados em SGBD MySQL;
- Configuração do arquivo de banco de dados;

A configuração do arquivo de acesso do *software* deverá ser criado no diretório conforme o exemplo: \$DIRETORIO\_APLICACAO\$/src/main/java/Database. O arquivo deverá ser um arquivo chamado Config.dmdat (é um arquivo de texto com a extensão modificada manualmente, devido a isso pode ser criado um arquivo .txt e depois alterar a extensão para .dmdat). A figura 25 apresenta como deve ser preenchido o arquivo de configuração do *software*.

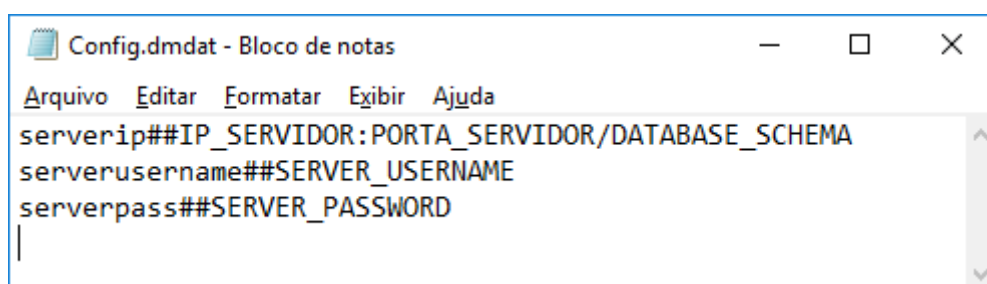


Figura 25: Modelo de configuração do ambiente do software

As variáveis:

- IP\_SERVIDOR: Deverá ser trocada pelo ip do servidor que será utilizado pelo *software*;
- PORTA\_SERVIDOR: Deverá ser trocada pela porta do ip que do servidor que será utilizado pelo *software*;
- DATABASE\_SCHEMA: Deverá ser trocada pelo *schema* que será utilizado como base pelo *software*;
- SERVER\_USERNAME: Deverá ser trocada pelo usuário com o qual o *software* conectará no banco de dados;
- SERVER\_PASSWORD: Deverá ser trocada pela senha do usuário com o qual o *software* conectará no banco de dados.

Uma vez configurado o arquivo Config.dmdat, o usuário deverá executar o arquivo InitialLoad.sql disponibilizado junto do arquivo executável, que será responsável por realizar criar os objetos de controle no banco de dados do usuário, além de inserir o primeiro usuário no banco de dados, o usuário **admin**.

Depois do arquivo InitialLoad.sql ter sido executado, o usuário já poderá utilizar o *software* normalmente, realizando o login utilizando o usuário admin, senha admin.

O *software* não foi desenvolvido pensando na instalação, sendo necessário verificar futuramente melhores formas de otimizar a configuração do ambiente e possibilitar que o usuário possa modificar o arquivo de configuração dinamicamente.

## **8. RESULTADOS OBTIDOS**

O sistema, que tem por objetivo atender a necessidade de controlar os registros e históricos das rotinas do banco de dados em SGBD MySQL do usuário, foi desenvolvido de forma que atendeu a todos os requisitos mensurados na fase de análise do projeto.

O acesso dos usuários está funcional, permitindo registrar corretamente quais usuários estão inserindo e modificando os objetos no banco.

Todos os cadastros de objetos estão de acordo com o previsto, permitindo fazer a inserção, edição e inativação dos objetos, além de validar se os objetos estão com a sintaxe de SQL correta.



## 9. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O projeto desenvolvido atende as necessidades para o qual foi desenvolvido, permitindo o controle e visualização das alterações realizadas nas rotinas SQL em SGBD MySQL.

O sistema está funcional, entretanto há detalhes que pode ser aprimorados. As próximas etapas são definir uma melhor estrutura de segurança na comunicação aplicação x banco de dados através de um protocolo SSL, aprimorar também o Look and Feel (estilo da tela e componentes) da aplicação para deixá-la mais leve e agradável e em tons mais fortes.

Conforme apresentado anteriormente no documento, há a intenção de implementar um módulo de relatórios de sistema, que deverá gerar, com base em um modelo configurado pelo cliente, um relatório de inserções e modificações de objetos com base em filtros de data e tipos de objetos.

Tendo como alvo os desenvolvedores iniciantes, desenvolvedores *freelancers* ou até mesmo pequenas empresas, o *software* desenvolvido nesse trabalho poderá ser de grande ajuda para aqueles usuários que buscam um versionamento e controle das rotinas SQL em SGBD MySQL que são desenvolvidas em seus projetos, além de permitir que o usuário volte o objeto em casos de problemas devido ao recurso de histórico desenvolvido.

## 10. REFERÊNCIAS

BOOCH, G.; RUMBAUGH, J.; JACOBSON, L. *UML: guia de usuário*, Elsevier, 2005.

MYSQL. *MySQL Documentation*. Disponível em: <<https://dev.mysql.com/doc/>>. Acesso em: Jan-Jul, 2018.

MELO, A. C. *Desenvolvendo aplicações com UML 2.2: do conceitual à implementação*, 3ª Ed. Rio de Janeiro: Brasport, 2010.

XAVIER, F. S. V.; PEREIRA, L. B. R. *SQL dos Conceitos às Consultas Complexas*. Rio de Janeiro: Editora Ciência Moderna, 2009.

VENTURA, P.: *O que é um requisito não-funcional*. Disponível em:<<https://www.ateomomento.com.br/o-que-e-um-requisito-nao-funcional/>>. Acesso em: Abril, 2018.

NETBEANS. *NetBeans IDE Features*. Disponível em: <<https://netbeans.org/features/index.html>>. Acesso em: Dez, 2017.

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE, *UML - Casos de Uso*. Disponível em:<<http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>>. Acesso em: Jun, 2018.

FOWLER, M. *UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos*, 3ª Ed. Porto Alegre: Bookman, 2005.

FONSECA, G.: *Os principais diagramas da UML – Resumo rápido*. Disponível em:<<https://www.profissionaisti.com.br/2011/07/os-principais-diagramas-da-uml-resumo-rapido/>>. Acesso em: Jul, 2018.

ORACLE, *Java Platform Standard Edition 7 Documentation*. Disponível em:<<https://docs.oracle.com/javase/7/docs/index.html>>. Acesso em: Nov, 2017 – Jul, 2018.

REZENDE, R. *Oracle Scheduler: Um guia completo*, Disponível em:<<https://www.devmedia.com.br/oracle-scheduler-um-guia-completo-parte-1/26323>>. Acesso em: Ago, 2018.

SPECIALSKI, E. S., *Um modelo para o gerenciamento de banco de dados SQL através de Stored Procedures*. Disponível em: <<https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/84209/191454.pdf?sequence=1>>. Acesso em: Set, 2018.

CETA, N. *All you need to know about UML Diagrams: Types and 5+ examples*. Disponível em: <<https://tallyfy.com/uml-diagram/>>. Acesso em: Set, 2018.