



## Relatório - Trabalho 2

Célio Henrique Zeni, Erick Vinicius da Silva, Erick Antonio, Patrick Pasqualotto e Pedro Marangoni

Universidade Tecnológica Federal do Paraná – UTFPR

Campus Dois Vizinhos – DV

Coordenação de Engenharia de Software – COENS

Estrada para Boa Esperança do Iguaçu, km 04 – Zona Rural,

CEP 85660-000, Dois Vizinhos - PR - Brasil

e-mail: [patrickpasqualotto@alunos.utfpr.edu.br](mailto:patrickpasqualotto@alunos.utfpr.edu.br), [celio@alunos.utfpr.edu.br](mailto:celio@alunos.utfpr.edu.br),

[pedromarangoni@alunos.utfpr.edu.br](mailto:pedromarangoni@alunos.utfpr.edu.br)

**Resumo** – O objetivo deste relatório é abordar como foi montado e estruturado nosso sistema de vendas, desde a parte da interface com a construção de menus até a parte do backend onde existe uma integração com o Banco de Dados que vai armazenar todos os lançamentos feitos através da aplicação, além disso também relatar como foi o uso de índices para otimizar as consultas realizadas pelo sistema. Além disso, iremos falar sobre a função rollback que nos oferece integridade ao Banco de Dados e claro a parte de permissões de acesso que irá nos dar segurança.

## I INTRODUÇÃO

A finalidade deste trabalho é desenvolver uma **aplicação** que contenha uma **interface gráfica** e integrada a um **banco de dados**, o sistema deve possuir a operação de venda dos itens de forma completa, tendo uma **indexação** que contribua com uma busca mais rápida e eficiente dos dados, **processamento de transações e controle de concorrência**, **recuperação**(garantir consistência dos dados e ter uma rotina de backup) e garantir a **segurança** com criação de usuários e grupos, concessão de diferentes privilégios aos grupos, adicionar usuários aos grupos.

## II DESENVOLVIMENTO DA APLICAÇÃO

O primeiro passo foi a criação das tabelas que vão existir dentro do Banco de Dados, foi seguido a mesma lógica das tabelas fornecidas no pdf([link](#)).

Agora com as tabelas criadas focamos então em decidir como iríamos montar uma interface e fazer com que ela se comunicasse com o nosso Banco de Dados que deveria ser PostgreSQL, então optamos por utilizar Java pelo fato da familiaridade devido a utilização na matéria de “Programação Orientada a Objetos”, utilizamos o Java tanto na parte visual utilizando a biblioteca Swing que facilita na criação de Menus através de uma ferramenta dentro do NetBeans, onde se pode ver onde vai ser colocado cada caixa de texto, label, etc e já gera o código das marcações onde só cabe a nós criar as funções dentro das funções de ação de cada botão.

Primeiro criamos cada menu individualmente, então dividimos por MenuCadastroFuncionário, MenuCadastroFornecedor, MenuCadastroProduto, Menu CadastroLogin, após todos esses menus estruturados então criamos o MenuCadastroGeral, onde em cada botão tinha uma ação para chamar seu respectivo menu e por último foi criado o MenuLogin e feito uma validação para ver se o usuário que iria fazer o cadastro tinha acesso ou não ao sistema e claro se o login fosse efetuado com sucesso ele chamaria o MenuCadastroGeral.

### II.I - Menu Login

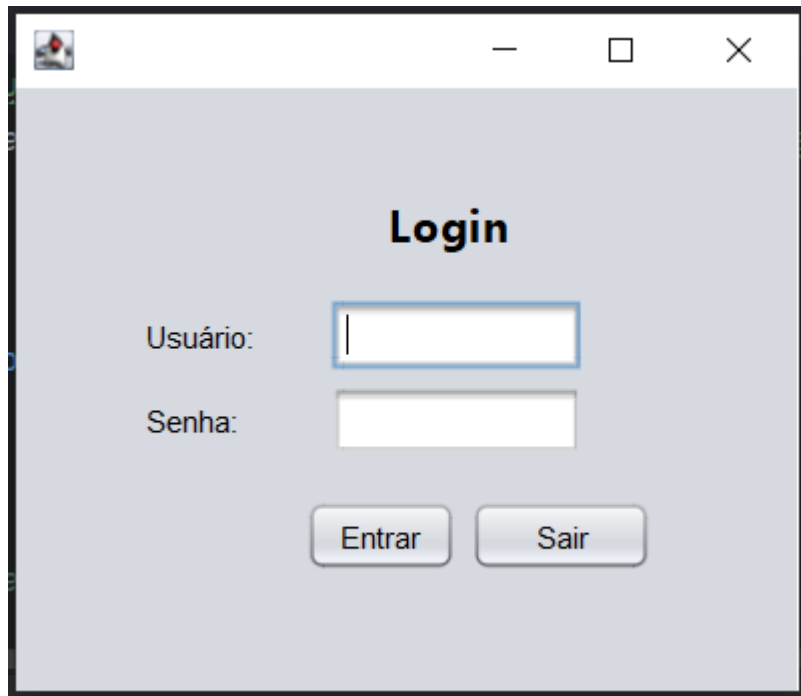


Imagem 01 - Tela de Login;

Fonte - Autoria Própria;

Onde foi inserimos duas caixas de texto para que o usuário pudesse inserir seus dados de login e os dois botões a baixo para caso ele “Entrar”/Efetuar o Login, onde existe uma validação que vai buscar lá no Banco de Dados e vai checar se esse usuário tá cadastrado e se as credenciais estão corretas, caso o

retorno lá no Banco seja positivo ele vai abrir o “MenuCadastroGeral” e caso a devolutiva seja negativa irá mostrar a seguinte mensagem:

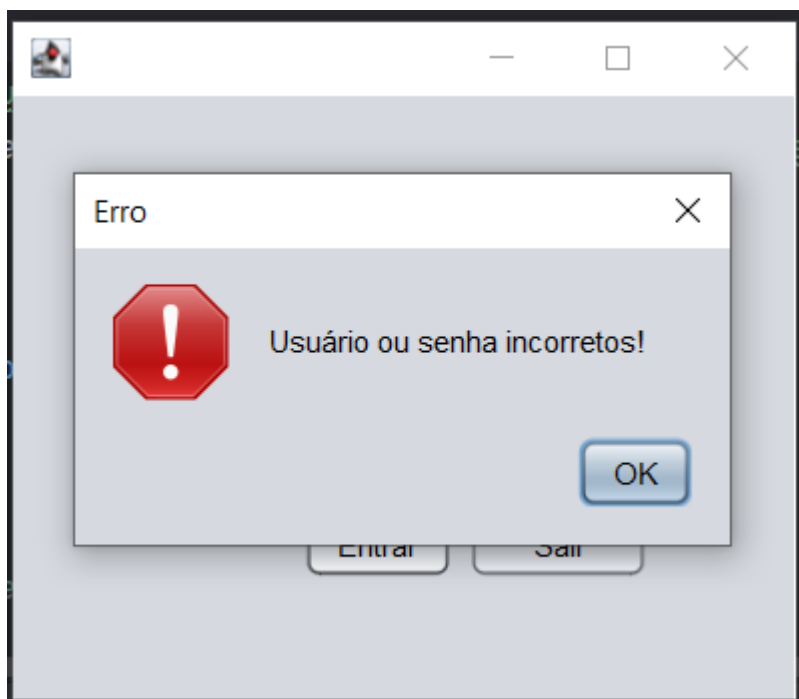


Imagem 02 - Mensagem de Erro na Tela de Login;  
Fonte - Autoria Própria;

Após essa validação, vamos ter então o:

## II.II - Menu Geral

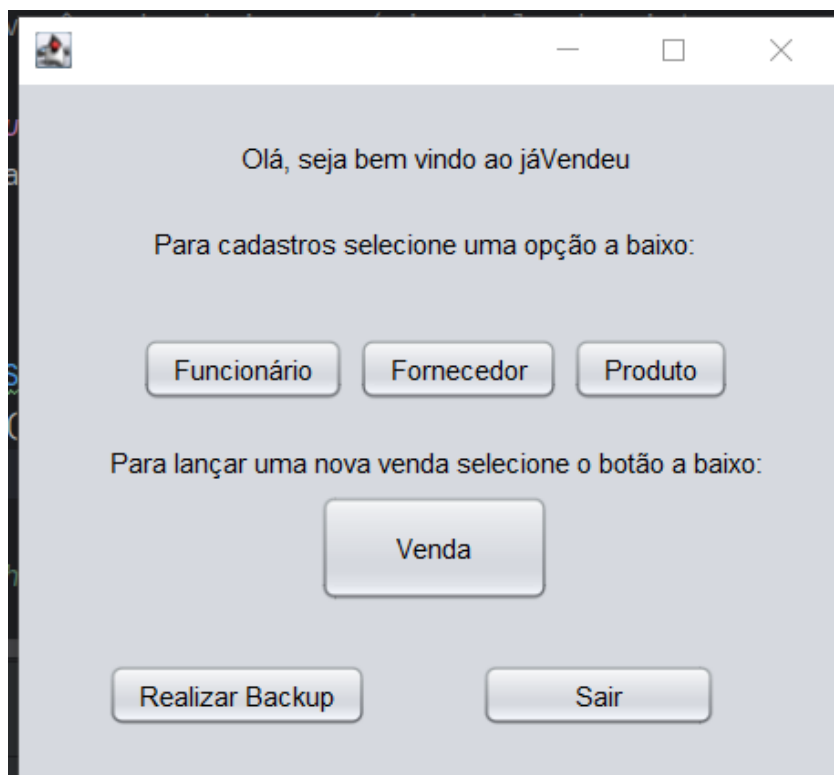


Imagem 03 - Menu Geral;  
Fonte - Autoria Própria;

Aqui então temos todas as funcionalidades do nosso sistema, desde o cadastro de um novo Funcionário, o cadastro de novos fornecedores/produtos e o mais importante e o foco deste trabalho é a Venda. Conseguimos também desenvolver a questão do Backup que quando o usuário logado desejar ele pode clicar em “Realizar Backup” e o sistema irá realizar o backup e restaurá-lo de forma automática. Foram criados esses 6 botões e como dito anteriormente ao clicar em cada um deles devido a ação do botão é acionada abrindo o respectivo menu.

Por exemplo se desejássemos cadastrar um novo Funcionário dentro do sistema iremos acessar o:

### II.III - Menu Funcionário

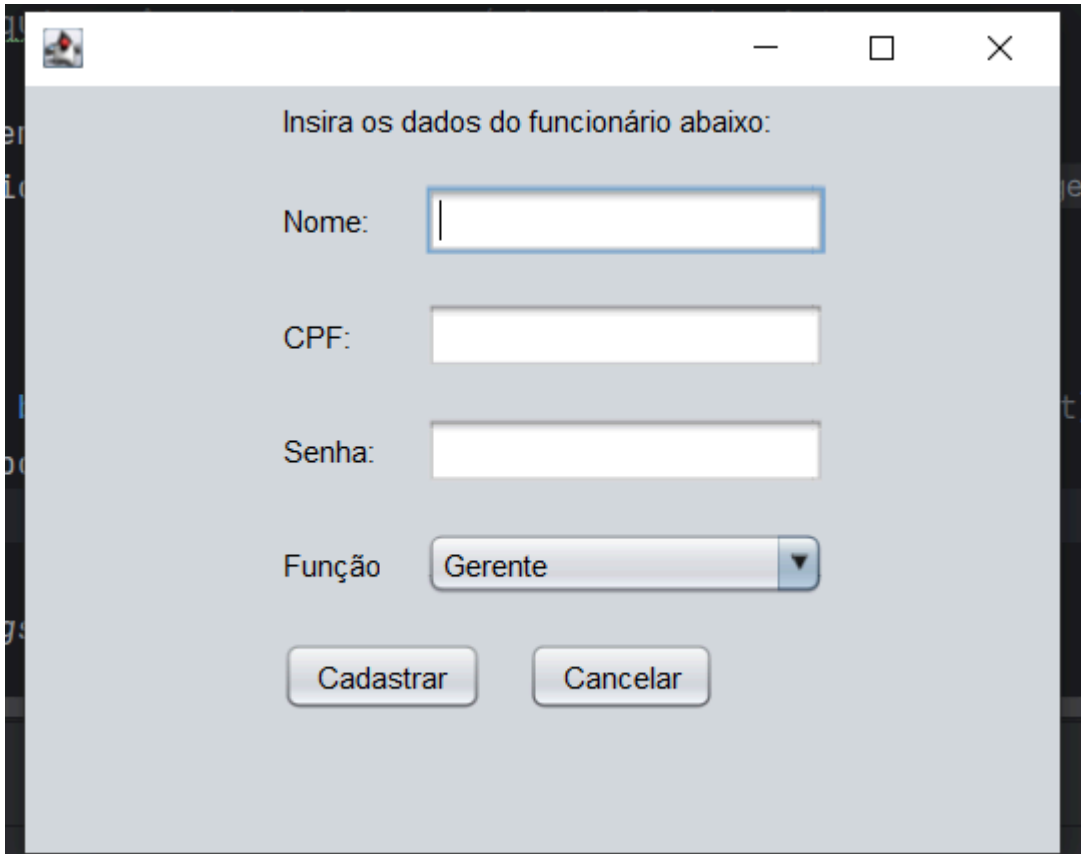
A imagem mostra uma janela de software com o título "Insira os dados do funcionário abaixo:". Dentro da janela, há quatro campos de entrada: "Nome:" com um campo de texto, "CPF:" com um campo de texto, "Senha:" com um campo de texto, e "Função:" com uma caixa de seleção (ComboBox) que atualmente exibe "Gerente". Abaixo dos campos, há dois botões: "Cadastrar" e "Cancelar". A janela possui uma barra de título com ícones de minimizar, maximizar e fechar.

Imagem 04- Tela de Cadastro de Funcionário;

Fonte - Autoria Própria;

Aqui então foi criado 4 caixas de textos que vão receber as informações referentes ao funcionário que temos que cadastrar dentro do banco de dados e também foi pré definido algumas funções e inserido uma ComboBox para estar exibindo as funções que o Funcionário pode estar exercendo, onde esses dados que serão inseridos aqui serão cadastrados instantaneamente no Banco de Dados ao clicar em “Cadastrar” e caso eu clique em “Cancelar” ele vai fechar a tela do Menu Funcionário e vai voltar para a anterior.

E caso o usuário tenta-se cadastrar um novo funcionário sem preencher todos os campos o sistema também irá me avisar com um erro na tela.

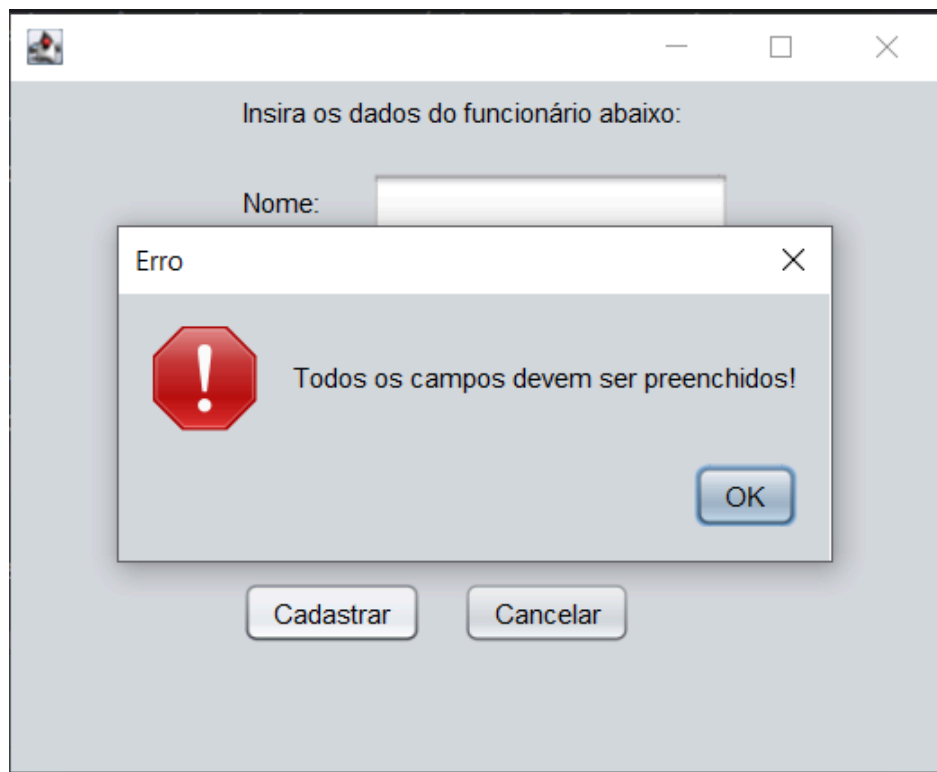


Imagem 04- Tela de Cadastro de Funcionário;

Fonte - Autoria Própria;

Após a conclusão do cadastro voltamos então para o Menu Principal e vamos abrir o:  
II.IV - Menu Fornecedor

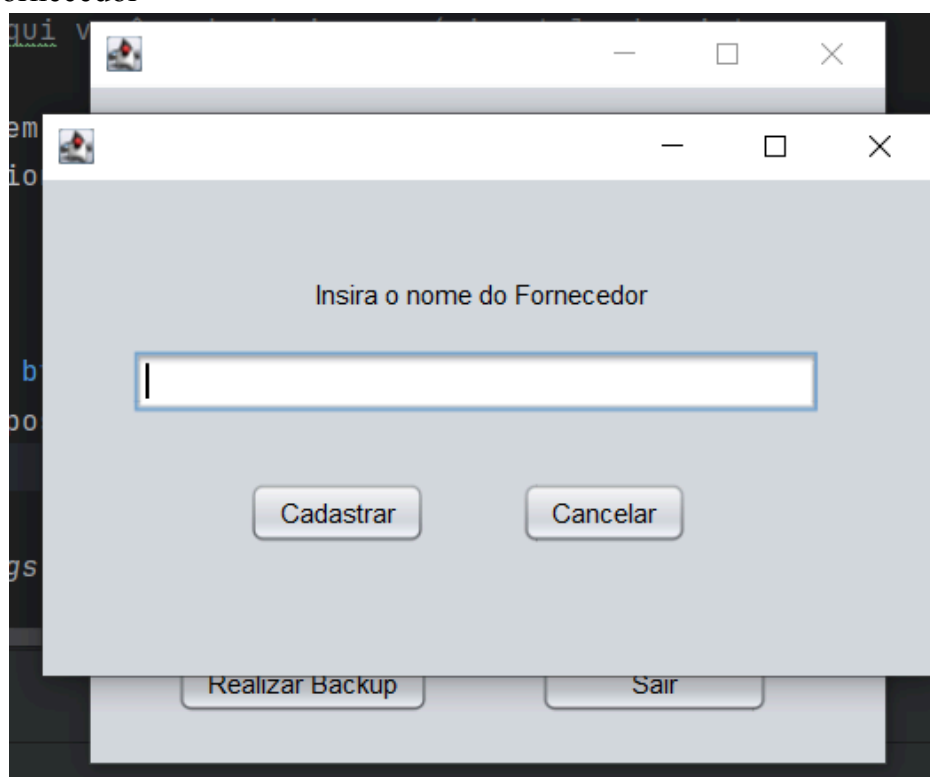
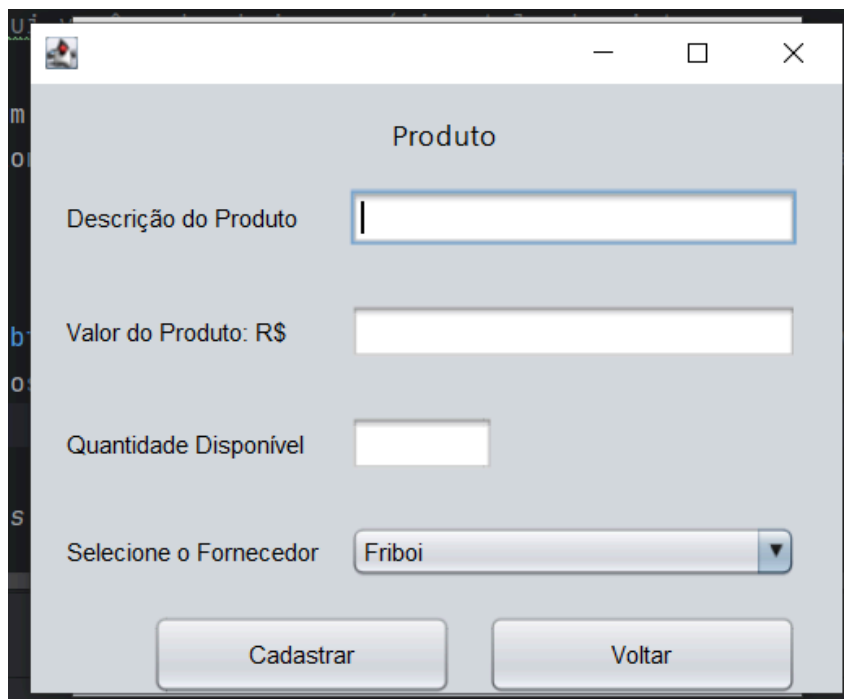


Imagem 05- Tela de Cadastro de Funcionário;

Fonte - Autoria Própria;

Aqui então podemos inserir as informações de um novo fornecedor que possa ser necessário o cadastro. E por último antes do Menu da Venda, vamos ter o:

## II.V - Menu Produto



A screenshot of a software window titled "Produto". It contains four input fields: "Descrição do Produto" (with a cursor), "Valor do Produto: R\$" (empty), "Quantidade Disponível" (empty), and "Selecione o Fornecedor" (a dropdown menu showing "Friboi"). At the bottom are two buttons: "Cadastrar" and "Voltar".

Imagem 06- Tela de Cadastro de Produto;  
Fonte - Autoria Própria;

E da mesma forma que temos as validações para não permitir que o usuário cadastre fornecedor ou funcionário sem preencher todos os campos, também temos a validação aqui no cadastro do produto.

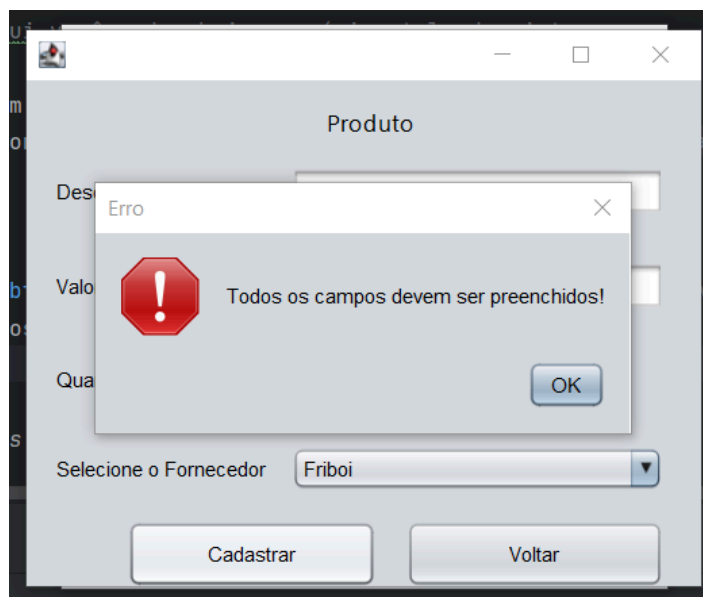


Imagem 07- Janela de Erro;  
Fonte - Autoria Própria;

E por último e claro o mais importante o:

## II.VI- Menu Venda

Seleção de Funcionário: Patrick Pasqualotto

Adicionar Remover

Código	Nome	Quantidade	Valor Unitário	Subtotal
--------	------	------------	----------------	----------

Valor Total: R\$ 0,00

Concluir Venda Cancelar Venda

Imagem 08- Menu de Venda;

Fonte - Autoria Própria;

Onde aqui conseguimos selecionar qual foi o funcionário que lançou essa venda, através de um ComboBox, temos 4 botões, 1 para adicionar novos produtos que vai me abrir uma nova janela, temos o botão de remover que irá remover os produtos que vão estar na tabela a baixo, temos a Tabela “Carrinho” onde vão ser inseridos os itens selecionados e vou ter o “Concluir Venda” que ao ser selecionado vai efetuar o lançamento da venda e o “Cancelar Venda” que ao ser acionado vai estar efetuando o cancelamento da venda.

Seleção de Funcionário: Patrick Pasqualotto

Adicionar Remover

Código

Adicionar Produto

Produto: Alcatra 1kg

Quantidade: 1

Confirmar

Valor Total: R\$ 0,00

Concluir Venda Cancelar Venda

Imagem 08- Menu de Venda;

Fonte - Autoria Própria;

Aqui então vamos selecionar os produtos que queremos adicionar ao “Carrinho” que vai ser a nossa tabela no menu anterior.



### III CRIAÇÃO DE ÍNDICES E ANÁLISE DE DESEMPENHO

Como solicitado criamos também alguns índices para algumas tabelas, mas como foi feito isso? Primeiro analisei quais eram as consultas que o sistema iria usar na sua comunicação com o banco para buscar otimizar ela, então vi que da maneira que desenvolvemos a aplicação vai ocorrer alguns “SELECT” para buscar os Funcionários, Fornecedores e seus respectivos códigos e atributos é claro, algum listando todos e outros por igualdade. Então decidimos usar o Hash por termos bastante busca por igualdade, também utilizamos o Brin que como vimos na CCH 1 é ótimo para tabelas ordenadas que possuem um grande volume e por último e não menos importante decidimos utilizar também o B-Tree por ser o mais genérico e ver como ele se comportaria nas consultas utilizadas.

Para analisar os índices que vamos criar utilizamos as seguintes consultas, levando em conta o cenário onde vamos estar acessando a tabela “tb\_funcionarios” neste primeiro momento.

CONSULTA 1

```
SELECT nom_funcionario FROM tb_funcionarios;
```

CONSULTA 2

```
SELECT cod_funcionario FROM tb_funcionarios WHERE nom_funcionario = 'Pai do Mateus';
```

E com base nessas consultas, criamos os seguintes índices:

```
-- B-Tree
CREATE INDEX idx_funcionarios ON tb_funcionarios (nom_funcionario);
DROP INDEX idx_funcionarios;

-- hash
CREATE INDEX idx_funcionarios ON tb_funcionarios USING hash (nom_funcionario);
DROP INDEX idx_funcionarios;

--B-Tree
CREATE INDEX idx_funcionarios ON tb_funcionarios USING BRIN(nom_funcionario);
DROP INDEX idx_funcionarios;
```

E assim foi obtido os seguintes resultados para cada consulta:

### Desempenho Dos Índices

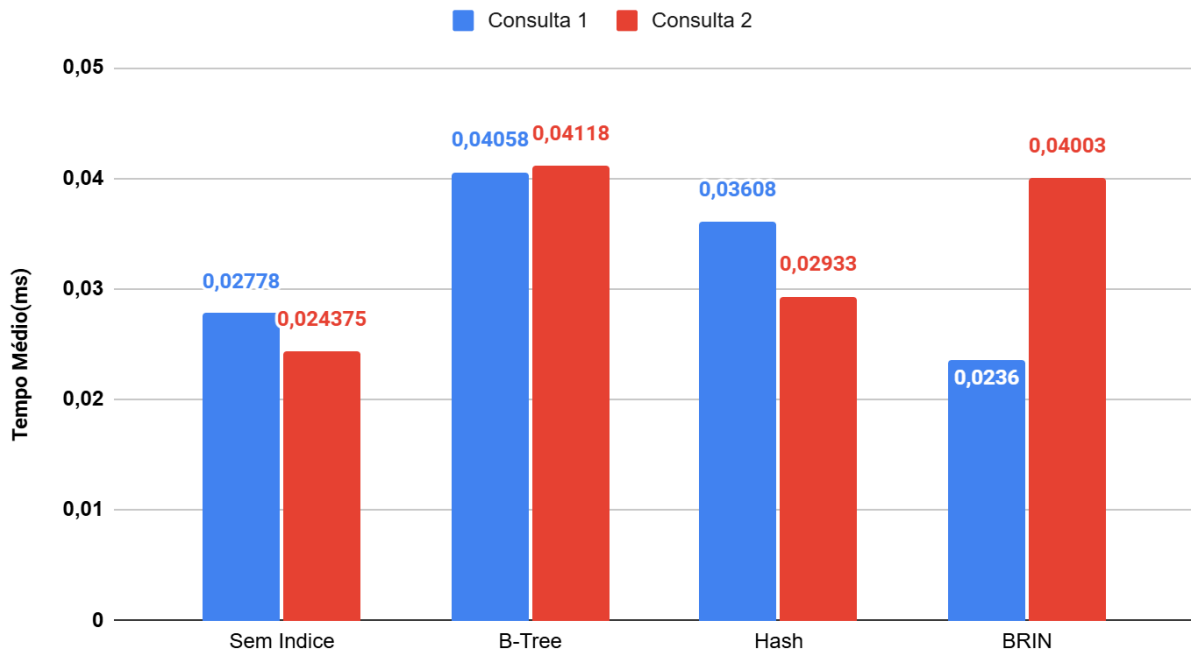


Imagem 09- Gráfico de Desempenho de Índices;  
Fonte - Autoria Própria;

Como podemos ver o único índice que de fato nos ajudou foi o BRIN mesmo ele sendo mais indicado para grande volumes de dados ordenados, mas olhando de uma maneira mais detalhista considerando os valores mínimos e máximos de tempo de cada consulta o B-Tree tem pontos a ser considerados, devido ao campo amostral dos dados ser baixo porque dentro do banco de dados não tínhamos um volume grande de dados não se pode tomar uma decisão baseada nisso, mas já dá pra ter uma base do que se esperar. Esperava um desempenho melhor utilizando o índice HASH por ser uma consulta por igualdade, mas acredito que isso se dá pelo que mencionei anteriormente sobre ser um volume pequeno de dados.

Mas queria ter algo mais concreto para me basear, então criei alguns índices para utilizar na tabela de Produto também, porque dentro da aplicação é “solicitado” alguns SELECT’s dessa tabela. As consultas utilizadas para esse segunda análise foram:

CONSULTA 1

```
SELECT cod_produto FROM tb_produtos WHERE des_produto = 'Heineken Caixa 6 Latas  
350ml';
```

CONSULTA 2

```
SELECT qte_produto FROM tb_produtos WHERE des_produto = 'Heineken Caixa 6 Latas
350ml';
```

Dentro da aplicação elas são utilizadas para buscar o código do produto em questão e para buscar a quantidade de estoque disponível, utilizando os mesmo índices que antes porém agora na coluna “des\_produto” da tb\_produtos. Os índices utilizados foram:

```
-- B-Tree
CREATE INDEX idx_produtos ON tb_produtos (des_produto);
DROP INDEX idx_produtos;

-- hash
CREATE INDEX idx_produtos ON tb_produtos USING hash (des_produto);
DROP INDEX idx_produtos;

--B-Tree
CREATE INDEX idx_produtos ON tb_produtos USING BRIN(des_produto);
DROP INDEX idx_produtos;
```

E com base nisso obtivemos os seguintes tempos:

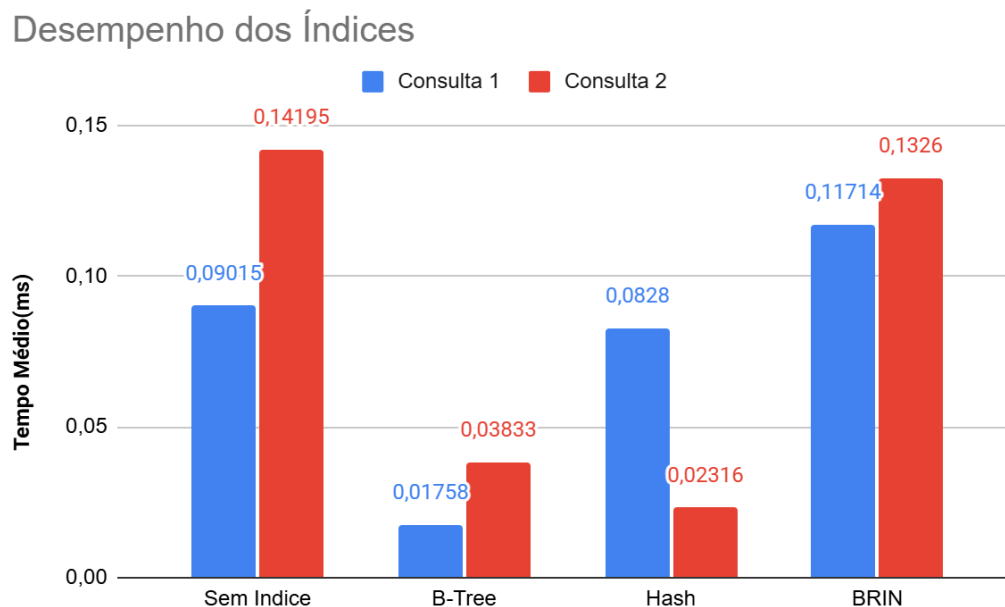


Imagem 10- Gráfico de Desempenho de Índices;  
Fonte - Autoria Própria;

O código utilizado para me ajudar a fazer a média de tempo de execução de cada consulta após ser executada cem vezes foi [este](#).

## IV RECUPERAÇÃO

A solução foi implementada em Java, utilizando a classe **ProcessBuilder** para executar um arquivo **BAT** localizado na pasta raiz do sistema, responsável por realizar o backup dos dados. O código foi desenvolvido dentro de um método chamado **realizarBackup()**.

```
@echo off
cd "c:\Program Files\PostgreSQL\16\bin"
set PGPASSWORD=masterkey
pg_dump -U postgres -f "C:\Users\*****\Desktop\"pasta do sistema"\backup.bkp" "nome banco"
exit
```

Dentro desse método, acessamos o arquivo no diretório do sistema e, utilizando um bloco **try**, realizamos o tratamento dos resultados da execução do backup. Se a execução for bem-sucedida, o sistema exibirá uma mensagem informando a conclusão e retornará à tela principal. Caso ocorra alguma falha, o sistema alertará que o backup não foi concluído devido a um erro e retornará normalmente à tela do sistema.

```
private void realizarBackup() { 1 usage
    try {
        // Caminho onde está o arquivo BAT
        String arquivoBat = "C:\\Users\\Erick\\Desktop\\sistemadevendasrestruturado\\executarbackup.bat";

        // Criando o processo para executar o BAT através do cmd
        ProcessBuilder processBuilder = new ProcessBuilder(...command: "cmd.exe", "/c", arquivoBat);
        processBuilder.inheritIO(); // Exibe saída no console

        // Inicia a execução do arquivo de backup
        Process process = processBuilder.start();
        int exitCode = process.waitFor(); // Aguarda a execução do backup

        System.out.println("Backup finalizado com código: " + exitCode);
    } catch (IOException | InterruptedException e) {
        System.err.println("Erro ao executar o backup: " + e.getMessage());
    }
}
```

Optamos por utilizar um arquivo externo para a execução do backup devido à facilidade de manipulação do arquivo **.BAT**, sem a necessidade de modificar ou retrabalhar o código em **Java**. Além disso, essa abordagem permite capturar possíveis erros durante a execução do backup sem comprometer o funcionamento do sistema ou causar falhas críticas.

Concluimos que essa solução possibilita a execução de backups de forma confiável e automática, contribuindo para a segurança dos dados do sistema. Melhorias futuras podem incluir a implementação de logs detalhados e a verificação automática da integridade do backup gerado.



## V SEGURANÇA

No requisito de segurança, foram criados os papéis de usuários, utilizando o comando ROLE. Após isso, foi concedido os privilégios para cada grupo e atribuído os usuários aos seus respectivos papéis.

Script da criação dos papéis:

```
CREATE ROLE administradores;  
CREATE ROLE operadores_caixa;  
CREATE ROLE repositores_estoque;  
CREATE ROLE financeiro;
```

Após a criação dos papéis, foi definido quais privilégios seriam atribuídos para cada um.

```
-- libera todas os privilegios para o administrador  
GRANT ALL PRIVILEGES ON tb_fornecedores, tb_funcionarios, tb_itens, tb_produtos,  
tb_vendas TO administradores;  
  
-- libera acesso apenas para ver e atualizar estoque para os repositores e para  
dar acesso e opção para eles cadastrarem novos fornecedores  
  
GRANT SELECT, UPDATE ON tb_produtos TO repositores_estoque;  
GRANT INSERT, SELECT ON tb_fornecedores TO repositores_estoque;  
  
-- Dando permissão para os caixas para lançarem as vendas e os itens  
  
GRANT SELECT, INSERT, UPDATE ON tb_vendas TO operadores_caixa;  
GRANT SELECT, INSERT, UPDATE ON tb_itens TO operadores_caixa;  
  
-- Libera acesso para o grupo do financeiro visualizar as vendas e funcionários  
  
GRANT SELECT ON tb_vendas, tb_funcionarios TO financeiro
```

Papéis	Privilégios	Tabelas
administradores	Todos os privilégios foram concedidos em todas as tabelas.	tb_fornecedores tb_funcionarios tb_itens tb_produtos tb_vendas
operadores_caixa	SELECT, INSERT, e UPDATE	tb_vendas tb_itens
repositores_estoque	SELECT, INSERT e UPDATE	tb_produtos tb_itens
financeiro	SELECT	tb_vendas tb_funcionarios

Por último, foi criado os usuários e vinculado com os seus respectivos grupos:

```
CREATE USER patrick WITH PASSWORD '1';
CREATE USER celio WITH PASSWORD '1';
CREATE USER erickribeiro WITH PASSWORD '1';
CREATE USER ericksilva WITH PASSWORD '1';
CREATE USER pedro WITH PASSWORD '1';

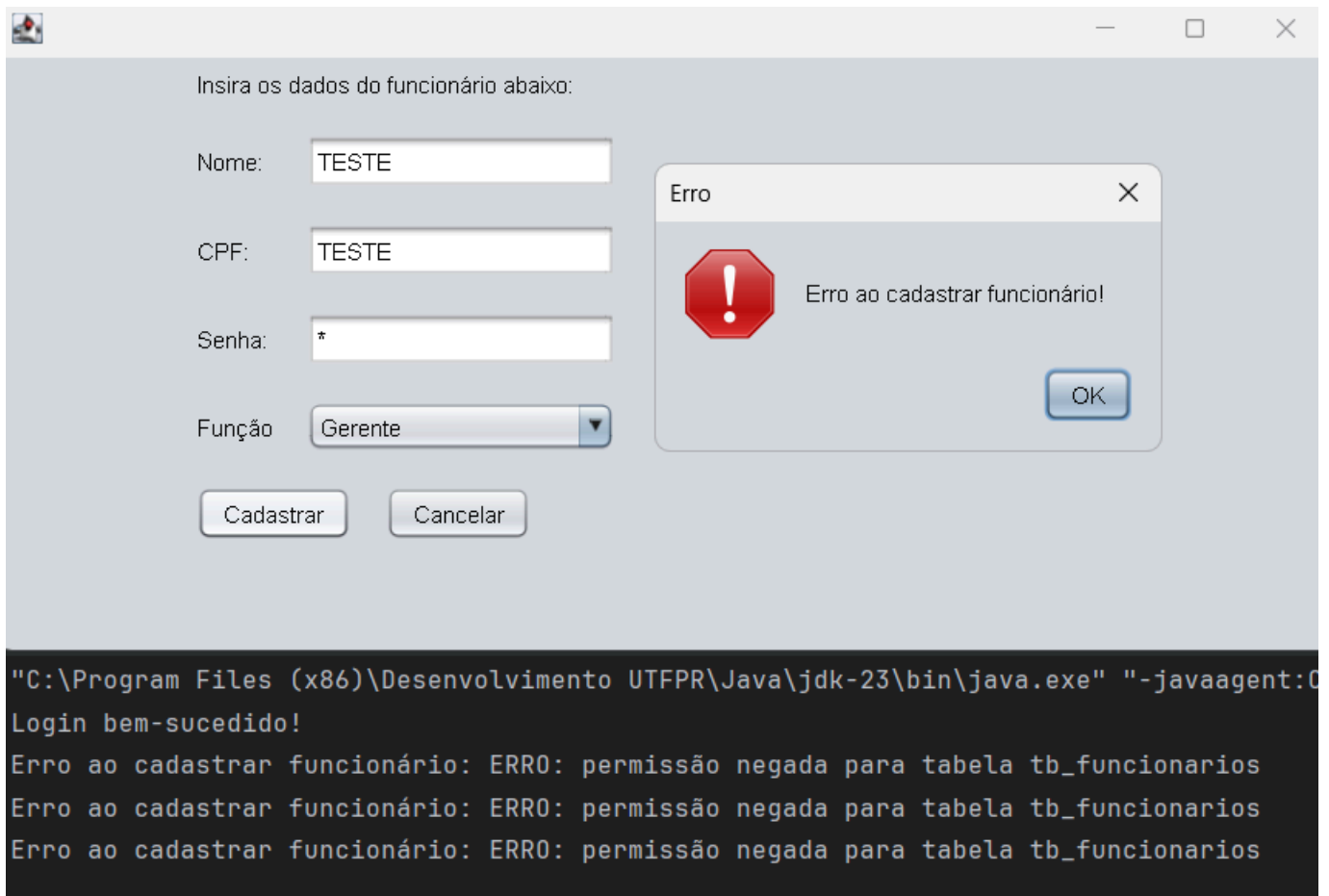
-- Adicionar os usuários aos grupos
GRANT administradores to patrick;
GRANT operadores_caixa to celio, erick;
GRANT repositores_estoque to ericksilva;
GRANT financeiro to pedro;
```

Foram realizados os seguintes testes:

Teste de cadastro de um funcionário utilizando um usuário pertencente ao grupo operadores\_caixa. Pela restrição, não deve permitir que usuários que possuem esse papel consigam cadastrar um funcionário.

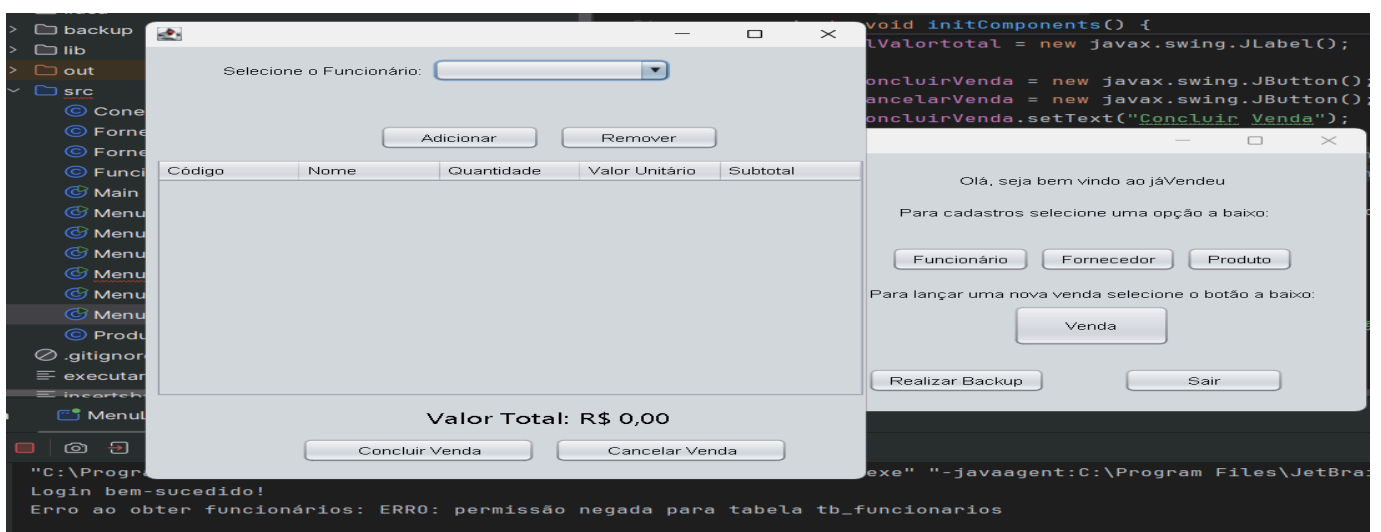
Foi realizado o login na aplicação utilizando o usuário celio.





Realizado a tentativa de venda, utilizando um usuário do grupo `repositores_estoque`. Já não permitirá ao usuário informar o fornecedor, pois ele não tem permissão para buscar esses dados da tabela `tb_fornecedores`.

O usuário utilizado foi o `ericksilva`.



Por fim, ajustamos as permissões do grupo `repositores_estoque` para permitir a realização de vendas. Essa modificação tem um propósito didático, demonstrando como é possível alterar as permissões de um grupo diretamente pelo banco de dados. Concedemos apenas o privilégio de `SELECT`, pois, com essa permissão, um usuário autenticado já seria capaz de consultar os funcionários necessários para viabilizar a venda.

```
GRANT INSERT ON tb_itens TO repositores_estoque;  
GRANT INSERT ON tb_vendas TO repositores_estoque;  
GRANT INSERT ON tb_itens TO repositores_estoque;
```

```
GRANT SELECT ON tb_funcionarios TO repositores_estoque;
```

Selecione o Funcionário:

Código	Nome	Quantidade	Valor Unitário	Subtotal
1	Hamburguer	1	10.0	10.0

Valor Total: R\$ 10,00



Selecione o Funcionário:

TESTE

Adicionar

Remover

Código	Nome	Quantidade	Valor Unitário	Subtotal
1	Hamburguer	1	10.0	10.0

Sucesso



Venda concluída com sucesso!

OK

Valor Total: R\$ 10,00

Concluir Venda

Cancelar Venda

## VI CONCLUSÃO

O desenvolvimento desta aplicação foi uma experiência de muito aprendizado, unindo interface gráfica, banco de dados e segurança. Conseguimos criar uma aplicação funcional, garantindo um bom desempenho com a utilização de índices e um controle de permissões.

Definir os papéis dos usuários e definir as permissões ajudou a manter um controle rigoroso sobre o que pode ser feito no sistema. Foram feitos testes para garantir que cada grupo de usuários tenha acesso apenas as funcionalidades permitidas.

O backup automático trouxe mais segurança aos dados, e os testes realizados confirmaram que as restrições de acesso funcionam.

No geral, o projeto atingiu seus objetivos e mostrou a importância de uma boa estruturação no desenvolvimento de sistemas.