

ACESSO A DADOS RELACIONAIS UTILIZANDO JAVA: CRIANDO UMA APLICAÇÃO-PROTÓTIPO PARA UMA LANCHONETE FICTÍCIA

Acadêmico: Tatiana Medeiros

RESUMO

O objetivo deste projeto é acessar dados dispostos em tabelas no banco de dados relacional MySQL a partir de código escrito em Java, possibilitando a busca por correspondências entre dados e a manipulação de linhas para exibir conteúdos ou acrescentar cadastros de clientes fictícios. Foram utilizados o software MySQL Workbench, a linguagem Java, as bibliotecas Java API (como o JDBC) e o ambiente de desenvolvimento IntelliJ. A presença de um banco de dados mostra a importância que o armazenamento de informações, assim como o devido gerenciamento, tem para um adequado aproveitamento e eficiência de uma aplicação. As ferramentas tecnológicas estão em constante evolução, portanto, o que fazia sentido no passado pode não fazer mais no presente. Porém, conceitos básicos devidamente aprendidos podem contribuir para a redução do tempo de aprendizagem de novas tecnologias e tornar os profissionais mais aptos a lidar com problemas inevitáveis em sistemas.

Palavras-chave: MySQL, Java, JDBC.

INTRODUÇÃO

A comunicação entre sistemas deve ser eficiente e devidamente planejada, a escolha das tecnologias utilizadas no desenvolvimento de um projeto requer não somente o conhecimento das ferramentas de forma individual, mas também em como atuam em conjunto. A consolidação da linguagem Java no mundo corporativo alinhada à agilidade, facilidade de uso e bom desempenho do MySQL resultam em sistemas ágeis e escaláveis. A facilidade quanto à integração de ambas as ferramentas fomenta o uso em conjunto e permite que a flexibilidade de Java seja beneficiada pela robustez do MySQL.

Este trabalho visa o desenvolvimento de um programa simplificado, isto é, sem as complexidades de um sistema utilizado em um cenário real e sem o uso de frameworks e interfaces “amigáveis”. O objetivo é mostrar exemplos de como uma aplicação Java pode ler, alterar e inserir dados em tabelas criadas no MySQL. A lógica aplicada ao longo das chamadas via JDBC pode servir de protótipo para projetos que de fato gerem benefícios no mundo real acarretando na automação de tarefas, embora este não seja o principal uso da linguagem.

Dante do exposto, será apresentado uma aplicação de uma lanchonete fictícia em que o cliente precisa comunicar-se com um sistema para realizar cadastro, fazer login e escolher itens disponibilizados em cardápios. Todas as ações citadas utilizam código em Java e a importação de classes contidas em pacotes disponibilizados pela

própria linguagem acarretando em ganho de performance e ampliação de funções sem necessariamente aumentar a quantidade de linhas de código. As entradas do usuário, assim como as saídas enviadas para este, ocorrem via console do IntelliJ IDEA, visto que o projeto é simplificado e não possui o foco em interfaces gráficas, mas sim em trazer conhecimentos básicos sobre banco de dados e programação.

FUNDAMENTAÇÃO TEÓRICA

Os bancos de dados tornaram-se ferramentas tecnológicas de suma importância em aplicações, negócios e, até mesmo, são utilizados em análises preditivas. Dentre os tipos mais conhecidos encontram-se os relacionais e os não-relacionais. Assim como linguagens de programação podem ser estruturadas em paradigmas, o armazenamento de informações também engloba lógicas de organização.

O MySQL é robusto, rápido, possui características de um Sistema Gerenciador de Banco de Dados (SGBD), disponibiliza vários tipos de tabelas para armazenamento de dados, oferece multiacesso a dados, permite o gerenciamento e a concorrência de acesso, entre outros (Milani, 2007).

Geralmente os chamados “databases” não são usados sozinhos, mas sim em parceria com outros sistemas em que linguagens computacionais são utilizadas, tais como Java, Python e PHP. Segundo Luckow e Melo (2010), os bancos de dados permitem a persistência de dados em que uma comunicação adequada deve ser estabelecida para que operações sejam realizadas. Acrescentam ainda que uma das formas de concretizar o acesso a dados usando Java dá-se pelo driver Java Database Connectivity (JDBC).

O conceito resumido de banco de dados é “um local no qual é possível armazenar informações para consulta ou utilização, quando necessário” (Carvalho, 2015). O armazenamento e o acesso adequado e eficiente de informações podem ser utilizados para a confirmação de acesso de um usuário a uma determinada camada do sistema, o uso de registros, a captação de padrões através de esquematizações mais avançadas e a flexibilização geográfica, temporal e redução de custos com espaços físicos.

Os SGBD permitem uma melhor organização e facilidade de acesso às informações. Com o tempo, novas tecnologias surgem, sendo necessário que sistemas realizem cada vez mais funções para atender grandes volumes de dados e proporcionar alta disponibilidade (Costa, 2006). Um SGBD é um conjunto de software que permite o gerenciamento de uma base de dados, incluindo acesso e manipulação.

A escolha adequada de um banco de dados viabiliza o desenvolvimento de sistemas mais robustos, seguros e escaláveis. Questões como compatibilidade e usabilidade também devem ser avaliadas de modo a facilitar o manuseio de dados e informações. Comandos SQL podem ser enviados a partir de uma aplicação Java, por exemplo, através de interfaces do tipo Java-MySQL.

O JDBC é uma camada de abstração que permite a um programa Java acessar uma interface padrão para conectar-se a um banco relacional através da SQL (Ramon, 2001). O JDBC é uma biblioteca que permite a comunicação entre Java e um banco de dados. Encontra-se na API da própria linguagem e contém métodos prontos para que as consultas sejam feitas. Seu uso é voltado para banco de dados relacionais como o MySQL.

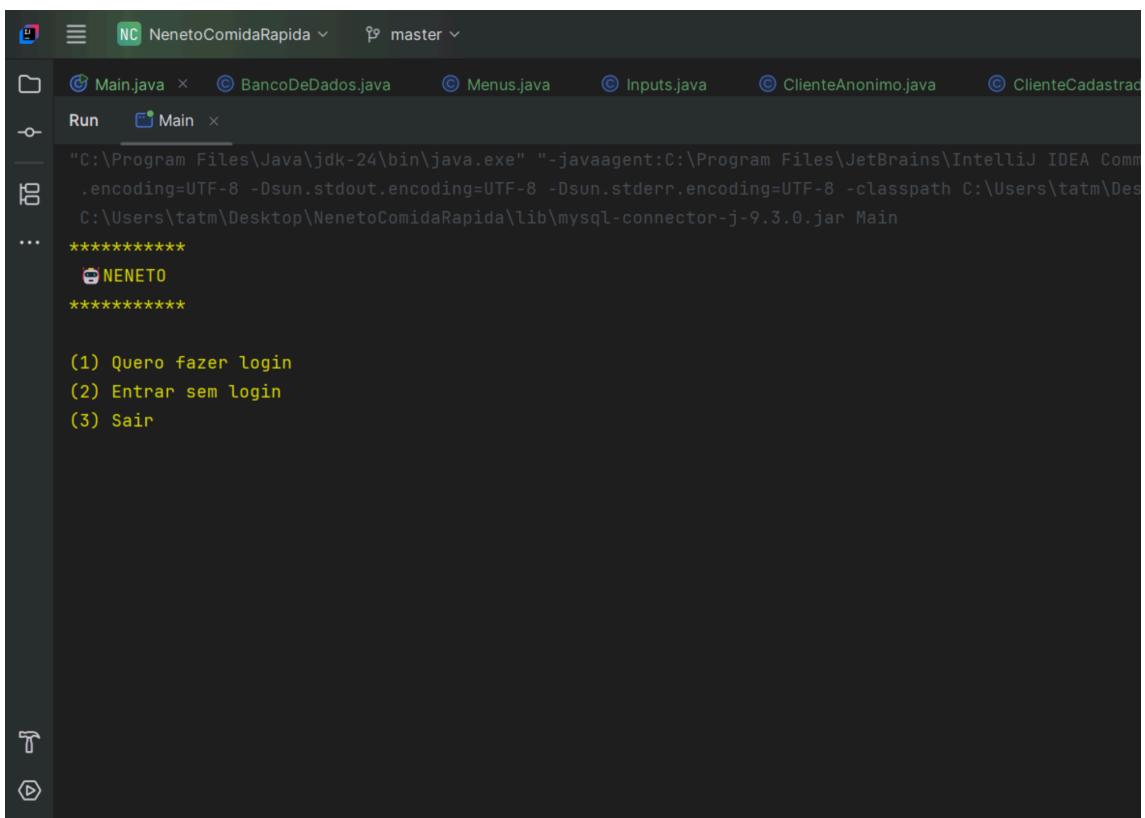
Java é uma linguagem de programação que permite tratar uma ampla gama de exceções que podem ocorrer ao longo do desenvolvimento e da execução de um programa, tal fato evita que processos sejam encerrados inesperadamente ou que não gerem retorno para uma melhor compreensão do ocorrido. Dentre os benefícios de Java estão a confiabilidade, portabilidade, legibilidade e reusabilidade.

METODOLOGIA

O trabalho envolveu o uso do IntelliJ IDEA que é um ambiente de desenvolvimento integrado, do Connector/J para a implementação da API do Java Database Connectivity, além de classes importadas do Java Development Kit e da instalação do MySQL que é um software de sistema gerenciador de banco de dados. Como o objetivo foi captar (leitura) e enviar (escrita) dados inseridos em tabelas, ou seja, o foco em uma aplicação simples, não houve a utilização de bibliotecas gráficas para a exposição de interfaces, mas sim o uso do console, Aplicação de console em Java, para entrada de dados e saída dos processamentos.

A tomada de decisões, a lógica das execuções, a interação com o banco de dados e as demais funcionalidades serão exibidas por meio de imagens com trechos de códigos. Também foi disponibilizado no início deste documento um link da publicação do projeto completo no Github. Conforme o programa é executado, o retorno ao código é feito para que a compreensão do todo seja mais fácil, rápida e dinâmica. Para tornar o trabalho mais lúdico, convido o leitor a imaginar uma lanchonete com cardápios digitais disponibilizados em que todo o processo de escolha dos lanches é feito por meio de um sistema. A seguir, a figura 1 exibe o início do programa a partir do momento em que o botão “run” do IntelliJ é acionado:

Figura 1 – Início da execução do programa



```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Comm
.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\tatm\Des
C:\Users\tatm\Desktop\NenetoComidaRapida\lib\mysql-connector-j-9.3.0.jar Main
*****
👤 NENETO
*****
(1) Quero fazer login
(2) Entrar sem login
(3) Sair
```

Fonte: o autor.

As opções disponibilizadas permitem que o usuário acesse uma conta já cadastrada ou crie uma conta digitando “1”; prossiga para a escolha do cardápio sem os passos da primeira opção digitando “2”; ou encerre o programa totalmente digitando “3”. A fim de desenvolver uma adequada explicação do projeto, a entrada fornecida foi o valor “1”. Após essa escolha, um outro menu com opções é exibido e novamente “1” é digitado. Dados de login, CPF e senha, são fornecidos corretamente e nesse processo são executados os métodos “autenticarClienteBancoDeDados” e “confirmarClienteBandoDeDados”. A autenticação dá-se através da checagem da tabela “clientes” nas colunas “cpf” e “senha”, enquanto que a confirmação envolve uma resposta ao cliente assegurando-o de que o login foi realizado.

Figura 2 – Autenticação do cliente no sistema

The screenshot shows a Java development environment with a terminal window displaying the output of a Java application. The terminal shows the following sequence of events:

- A menu is displayed:
 - (1) Já tenho cadastro
 - (2) Criar cadastro
 - (3) Sair
- The user selects option 1.
- The application proceeds with:
 - Acessando cadastro...
 - CPF:
12345678912
 - SENHA:
xuxu@21
 - Login realizado com sucesso!
 - Exibindo o cardápio...
- The application displays a list of food items with their descriptions and prices:
 - *****
✖ LANCHES

 - (1) Fatia de Pizza 1 unidade R\$ 3,99
 - (2) Happy Burguer 1 unidade R\$ 19,99
 - (3) Salada Fit 1 porção R\$ 4,99
 - (4) Big Esfiha 1 unidade R\$ 1,79
 - (5) Mini Salgadinhos 250 gramas R\$ 6,99
 - (6) Nepal Nuggets 250 gramas R\$ 6,19
 - (7) Batata Doce Assada 250 gramas R\$ 2,99

Fonte: o autor.

Figura 3 – Tabela “clientes”

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

SCHEMAS

nenetodb

- Tables: bebidas, clientes, lanches, sobremesas
- Views
- Stored Procedures
- Functions

sys

Query 1

1 select * from clientes;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wr

cliente_id	nome	cpf	senha	data_cadastro
1	Roberta Miranda	123456789-12	xuxu@21	2025-05-21
2	Calby Peixoto	987654321-01	calbyaqu1	2025-06-10
3	Roberto Carlos	987654320-11	@xuxuzinho	2025-06-14
HULL	HULL	HULL	HULL	HULL



The screenshot shows the MySQL Workbench interface with a query results window. The query `select * from clientes;` has been run, and the results are displayed in a grid. The grid has columns: cliente_id, nome, cpf, senha, and data_cadastro. There are four rows of data, plus a header row and a footer row with all cells containing 'HULL'. A purple arrow points to the first data row.

cliente_id	nome	cpf	senha	data_cadastro
1	Roberta Miranda	123456789-12	xuxu@21	2025-05-21
2	Calby Peixoto	987654321-01	calbyaqu1	2025-06-10
3	Roberto Carlos	987654320-11	@xuxuzinho	2025-06-14
HULL	HULL	HULL	HULL	HULL

Fonte: o autor.

Figura 4 – “autenticarClienteBancoDeDados”



```
public class BancoDeDados { 16 usages new *
    public static void autenticarClienteBandoDeDados(String cpf, String senha) { 1 usage new *
        String sql = "SELECT COUNT(*) FROM clientes WHERE cpf = ? AND senha = ?";

        try (Connection conn = DriverManager.getConnection(
            BancoDeDados.URL,
            BancoDeDados.USUARIO,
            BancoDeDados.SENHA
        );
            PreparedStatement stmt = conn.prepareStatement(sql)) {

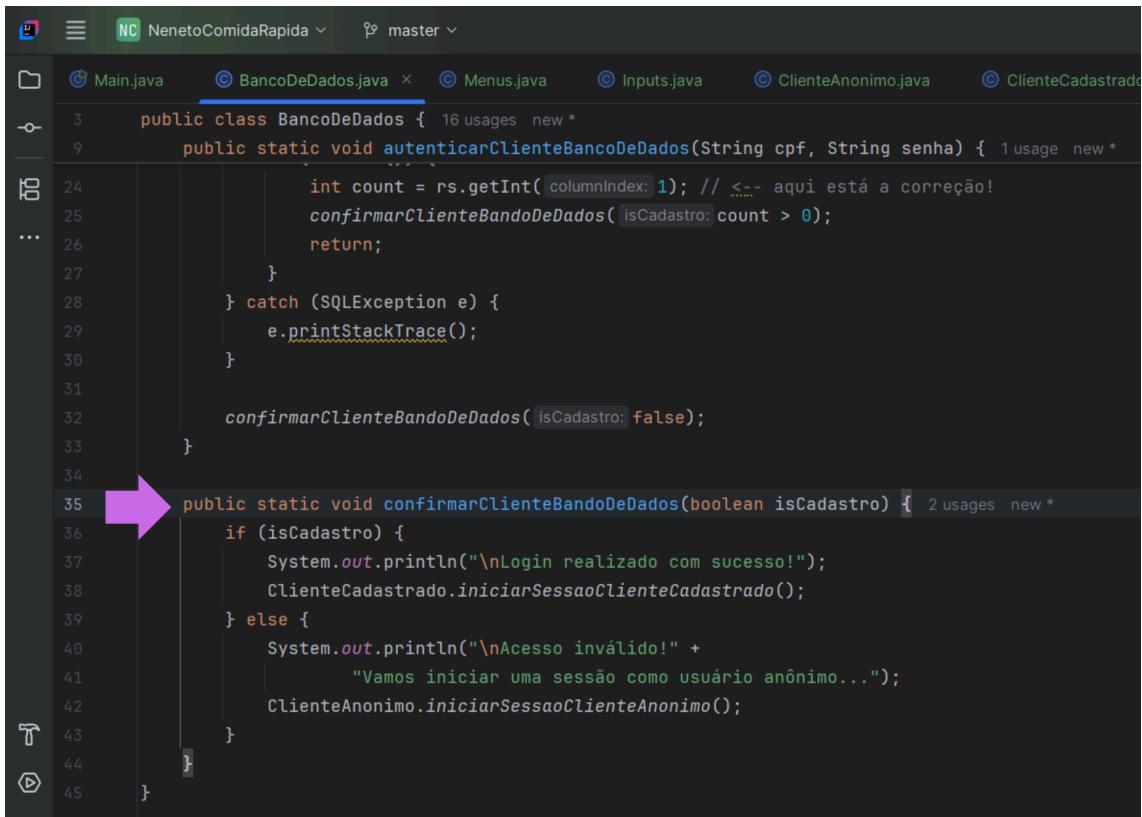
            stmt.setString(parameterIndex: 1, cpf);
            stmt.setString(parameterIndex: 2, senha);

            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                int count = rs.getInt(columnIndex: 1); // <-- aqui está a correção!
                confirmarClienteBandoDeDados(isCadastro: count > 0);
                return;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        confirmarClienteBandoDeDados(isCadastro: false);
    }
}
```

Fonte: o autor.

Figura 5 – “confirmarClienteBandoDeDados”



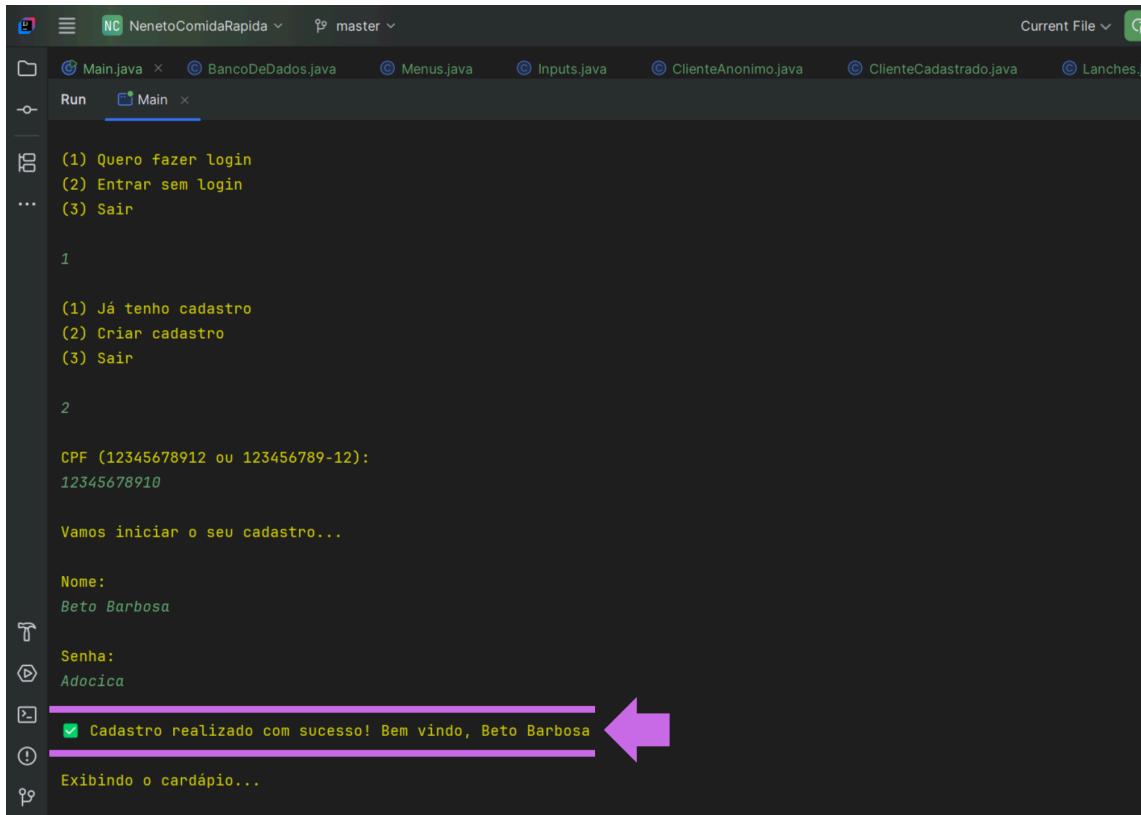
```
public class BancoDeDados {    public static void autenticarClienteBancoDeDados(String cpf, String senha) {        ...        int count = rs.getInt(columnIndex: 1); // <-- aqui está a correção!        confirmarClienteBandoDeDados(isCadastro: count > 0);        return;    }    } catch (SQLException e) {        e.printStackTrace();    }    }    }    confirmarClienteBandoDeDados(isCadastro: false);}35     public static void confirmarClienteBandoDeDados(boolean isCadastro) {    if (isCadastro) {        System.out.println("\nLogin realizado com sucesso!");        ClienteCadastrado.iniciarSessaoClienteCadastrado();    } else {        System.out.println("\nAcesso inválido!" +            "Vamos iniciar uma sessão como usuário anônimo...");        ClienteAnonimo.iniciarSessaoClienteAnonimo();    }    }45 }
```

Fonte: o autor.

Métodos de validação de entradas são também executados para checar se o usuário inseriu o tipo adequado de dado e se a quantidade de caracteres inseridos está em conformidade com o padrão de armazenamento definido no momento da criação das colunas no banco de dados. Deste modo é possível direcionar o programa em caso de inputs inadequados e exceções, inclusive gerando avisos e possibilitando novas tentativas de inserção de dados, evitando erros inesperados ou encerramentos inadequados.

Caso o usuário tenha optado por criar uma conta, uma nova linha (row) será criada na tabela “clientes”. A figura 7 mostra a tabela após a alteração sofrida, em que um novo cadastro é inserido. Pode-se dizer que a figura 3 representa o momento anterior e, a figura 7, o momento posterior aos comandos inseridos no MySQL pelo programa. Contudo, independente se o usuário optou ou não por fazer login, exceto se optou por “sair”, o cardápio será exibido ao mesmo tempo em que a classe “Scanner” é evocada para permitir que escolhas sejam feitas conforme as opções de lanches, bebidas e sobremesas exibidas sequencialmente.

Figura 6 – Inserção de um novo cadastro



```
(1) Quero fazer login
(2) Entrar sem login
(3) Sair

1

(1) Já tenho cadastro
(2) Criar cadastro
(3) Sair

2

CPF (12345678912 ou 123456789-12):
12345678910

Vamos iniciar o seu cadastro...

Nome:
Beto Barbosa

Senha:
Adocica

✓ Cadastro realizado com sucesso! Bem vindo, Beto Barbosa
Exibindo o cardápio...
```

Fonte: o autor.

Figura 7 – Tabela “clientes” atualizada

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help' are visible. Below the menu is a toolbar with various icons. The 'Navigator' pane on the left lists 'SCHEMAS' (with 'nenetodb' selected) and 'Tables' (which include 'bebidas', 'clientes', 'lanches', and 'sobreomesas'). The 'Query 1' tab contains the SQL query: 'select * from clientes;'. The 'Result Grid' pane displays the data from the 'lanches' table:

	cliente_id	nome	cpf	senha	data_cadastro
1	Roberta Miranda	123456789-12	xuxu@21	2025-05-21	
2	Calby Peixoto	987654321-01	calbyaqu1	2025-06-10	
3	Roberto Carlos	987654320-11	@xuxuzinho	2025-06-14	
5	Beto Barbosa	123456789-10	Adocica	2025-06-15	
	HULL	HULL	HULL	HULL	HULL

Fonte: o autor.

Após a escolha dos itens expositos no cardápio, o programa encerra exibindo o valor total de tudo que o cliente fictício escolheu para montar um lanche completo (ver figura 11). Porquanto os menus para a escolha dos itens seguem métodos parecidos, somente a tabela “lanches” e a classe “Lanches” serão exibidas em imagens de modo a facilitar a compreensão. As figuras a seguir exibem as telas do MySQL Workbench, do código Java responsável pela leitura das colunas da tabela “lanches” e do console exibindo o resultado da comunicação estabelecida, respectivamente.

Figura 8 – Tabela “lanches”

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1

SCHEMAS

nenetodb

- Tables
 - bebidas
 - clientes
 - lanches
 - sobremesas
- Views
- Stored Procedures
- Functions

sys

Query 1

```
1 select * from lanches;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

	lanche_id	nome	quantidade_preço	medida	preço
▶	1	Fatia de Pizza	1	unidade	3.99
	2	Happy Burguer	1	unidade	19.99
	3	Salada Fit	1	porção	4.99
	4	Big Esfiha	1	unidade	1.79
	5	Mini Salgadinhos	250	gramas	6.99
	6	Nepal Nuggets	250	gramas	6.19
	7	Batata Doce Assada	250	gramas	2.99
	8	Bolinhas de Queijo	250	gramas	4.99
	HULL	NULL	NULL	NULL	NULL

Fonte: o autor.

Figura 9 – comandos SQL via código em Java

```
public class Lanches { 1 usage new *
    public static String[] listarLanchesComoMenu() { 1 usage new *
        ArrayList<String> lanchesFormatados = new ArrayList<>();
        DecimalFormat formatarQuantidade = new DecimalFormat( pattern: "0.#");

        String sql = "SELECT nome, quantidade_preço, medida, preço FROM lanches"; ←

        try (Connection conn = DriverManager.getConnection(BancoDeDados.URL, BancoDeDados.USUARIO, BancoDeDados.SENHA);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql)) {

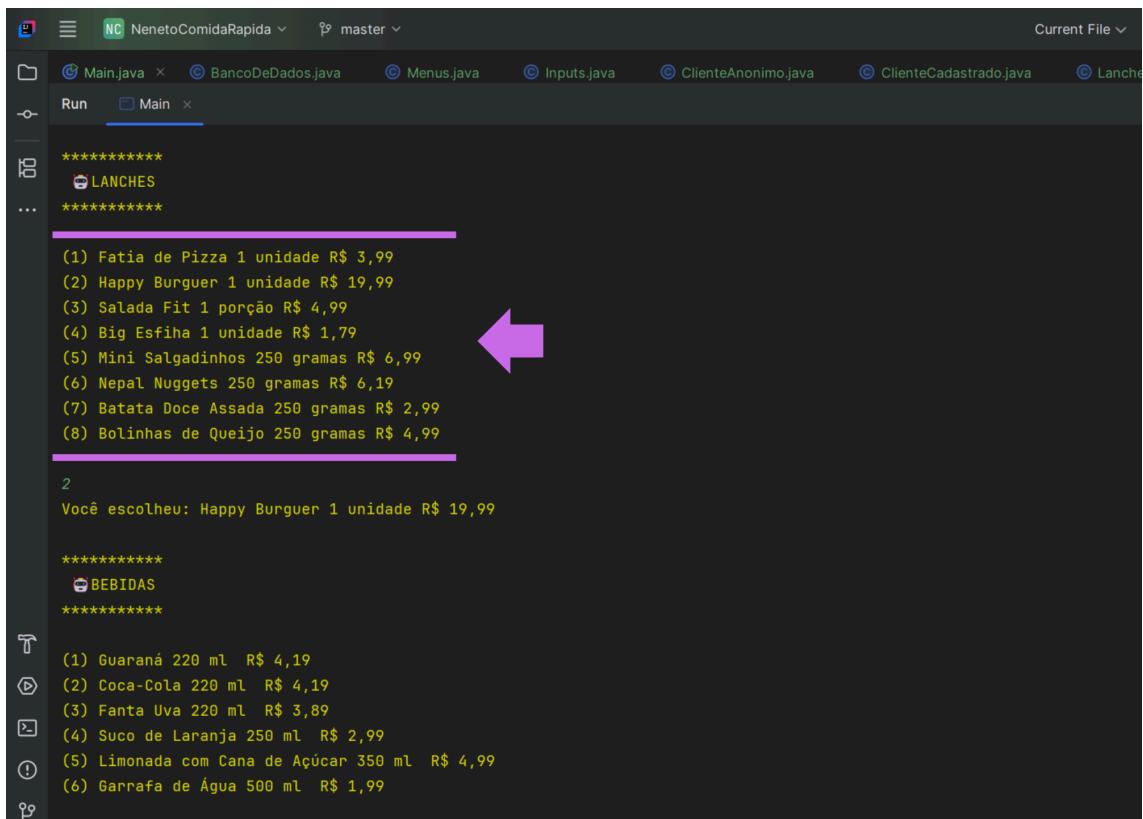
            while (rs.next()) {
                String nome = rs.getString( columnLabel: "nome");
                double qtd = rs.getDouble( columnLabel: "quantidade_preço");
                String medida = rs.getString( columnLabel: "medida");
                double preco = rs.getDouble( columnLabel: "preço");

                String linha = nome + " " + formatarQuantidade.format(qtd) + " " + medida + " R$ " + String.format("%.2f", preco);
                lanchesFormatados.add(linha);
            }

            } catch (SQLException e) {
                e.printStackTrace();
            }
        return lanchesFormatados.toArray(new String[0]);
    }
}
```

Fonte: o autor.

Figura 10 – Saída via console das linhas da tabela “lanches” em forma de lista/menu



```
Current File ▾
```

```
Main.java ✘ BancoDeDados.java ✘ Menus.java ✘ Inputs.java ✘ ClienteAnonimo.java ✘ ClienteCadastrado.java ✘ Lanches.java ✘
```

```
Run Main
```

```
*****  
✖ LANCHES  
...  
*****  
(1) Fatia de Pizza 1 unidade R$ 3,99  
(2) Happy Burguer 1 unidade R$ 19,99  
(3) Salada Fit 1 porção R$ 4,99  
(4) Big Esfiha 1 unidade R$ 1,79  
(5) Mini Salgadinhos 250 gramas R$ 6,99  
(6) Nepal Nuggets 250 gramas R$ 6,19  
(7) Batata Doce Assada 250 gramas R$ 2,99  
(8) Bolinhas de Queijo 250 gramas R$ 4,99  
*****  
2  
Você escolheu: Happy Burguer 1 unidade R$ 19,99  
*****  
✖ BEBIDAS  
*****  
(1) Guaraná 220 ml R$ 4,19  
(2) Coca-Cola 220 ml R$ 4,19  
(3) Fanta Uva 220 ml R$ 3,89  
(4) Suco de Laranja 250 ml R$ 2,99  
(5) Limonada com Cana de Açúcar 350 ml R$ 4,99  
(6) Garrafa de Água 500 ml R$ 1,99
```

Fonte: o autor.

Figura 11 – Encerramento do programa com a exibição do pedido e o seu valor total

```
NenetoComidaRapida master
Main.java BancoDeDados.java Menus.java Inputs.java ClienteAnonimo.java ClienteCadastrado.java Lanches.java
Run Main
(3) Fanta Uva 220 ml R$ 3,89
(4) Suco de Laranja 250 ml R$ 2,99
(5) Limonada com Cana de Açúcar 350 ml R$ 4,99
...
(6) Garrafa de Água 500 ml R$ 1,99

1
Você escolheu: Guaraná 220 ml R$ 4,19

*****
SOBREMESAS
*****

(1) Salada de Frutas Vermelhas 250 gramas R$ 9,99
(2) Brownie R$ 2,39
(3) Bolo Festa R$ 3,59
(4) Trufa de Chocolate com Maracujá R$ 2,99

3
Você escolheu: Bolo Festa R$ 3,59

VALOR TOTAL: R$ 27,77

CARDÁPIO FINAL:
• Lanche: Happy Burguer 1 unidade R$ 19,99
• Bebida: Guaraná 220 ml R$ 4,19
• Sobremesa: Bolo Festa R$ 3,59

Process finished with exit code 0
```

Fonte: o autor.

Os comandos SQL usados para a criação das tabelas e o preenchimento de linhas e colunas foram suprimidos para manter o trabalho conciso e para tornar a exposição do conteúdo mais enxuta. O propósito foi expor a comunicação entre um banco de dados e Java, mostrando a lógica envolvida em sistemas e como muitas estruturas precisam ser desenvolvidas para que ações, aparentemente simples, sejam executadas.

CONSIDERAÇÕES

Todo o conteúdo estudado para a realização deste projeto envolvendo Java e MySQL agregou conhecimento intelectual e profissional. Entender de forma mais aprofundada através da estruturação de camadas em um sistema permitiu uma melhor compreensão sobre como os banco de dados funcionam e a importância de ir além de digitar linhas de códigos, conhecendo ferramentas externas que podem e devem ser agregadas visando a ampliação de funcionalidades, agilidade, segurança e eficiência.

A linguagem Java está presente não somente em sistemas legados ou grandes sistemas empresariais, ainda é ativamente usada e atualizada, possuindo uma comunidade ativa e que torna o seu uso mais prático através do compartilhamento de

experiências e ensinamentos. Assim como o ecossistema Java, os SGBD constituem tecnologias com um leque de possibilidades quanto ao uso e estão consolidadas no mercado.

Aprender os conceitos que englobam a programação e a lógica por trás do gerenciamento de dados permite que pessoas e organizações possam usufruir de aplicações robustas, ágeis e flexíveis. A integração realizada na aplicação desenvolvida permitiu que em vez de escrever comandos SQL de forma direta no banco de dados, o código em Java ficasse responsável pelas leituras, checagens e inserções. Listas foram alimentadas com dados existentes entre linhas e colunas, permitindo a persistência de dados mesmo após o término da execução do programa.

REFERÊNCIAS BIBLIOGRÁFICAS

- MILANI, A. **MySQL - Guia do Programador**. São Paulo: Ed. Novatec, 2007.
- LUCKOW, D.H.; MELO, A. A. de. **Programação Java para a Web**. São Paulo: Ed. Novatec, 2010.
- CARVALHO, V. **MySQL: Comece com o principal banco de dados open source do mercado**. São Paulo: Ed. Casa do Código, 2015.
- COSTA, R. L. **SQL: Guia Prático**. São Paulo: Ed. Brasport, 2006.
- RAMON, F. **JDBC 2: acesso a banco de dados utilizando a linguagem Java**. São Paulo: Ed. Novatec, 2001.