Relatório – Atividade 3   
  
Para dar início a esta atividade, a primeira coisa que devemos fazer é abrir uma nova branch no git, usando o comando *“git checkout -b”* o nome da Branch, conforme solicita o enunciado deverá ser *“melhorias”,* o comando completo deve ser digitado da seguinte forma: *“git checkout -b melhorias”*.   
  
Como podemos ver no print abaixo, tirado do terminal da IDE VS Code, o código funcionou perfeitamente:

Texto

Descrição gerada automaticamente  
Após digitar o código “git checkout -b melhorias” o terminal retornou a mensagem: *“Switched to new Branch ‘melhorias’”.*

Feito esta alteração, podemos iniciar a implementação das novas funcionalidades, solicitadas pelo enunciado do exercício.  
  
A primeira a ser implementada ser á a “venderProduto”, na classe ProdutosDAO.   
- Na classe ‘ProdutosDAO’, vamos criar um método chamado “venderProduto”. Este método receberá o update da informação ‘status’ da tabela e colocará “vendido” no lugar de qualquer outro termo que esteja lá. Vide abaixo o print do código deste método, inserido após os getters and setters da classe ‘ProdutosDAO’:  
*Texto

Descrição gerada automaticamente*O código utilizado foi este:   
  
*public void venderProduto(int produtoId) {*

*String sql = "UPDATE produtos SET status = 'Vendido' WHERE id = ?";*

*try (Connection conn = this.connect();*

*PreparedStatement pstmt = conn.prepareStatement(sql)) {*

*pstmt.setInt(1, produtoId);*

*pstmt.executeUpdate();*

*} catch (SQLException e) {*

*System.out.println(e.getMessage());*

*}*

*}*

Agora que já é possível alterar o status do produto, vamos listar todos os produtos que já foram vendidos. Para isto, vamos criar um novo método, também na classe ProdutosDAO, onde serão mostrados todos os produtos que foram vendidos. Veja no print abaixo como ficou o comando inserido na classe ‘ProdutosDAO’:  
Texto

Descrição gerada automaticamente  
O comando utilizado para a criação deste método foi:   
  
*public List<Produto> listarProdutosVendidos() {*

*List<Produto> produtosVendidos = new ArrayList<>();*

*String sql = "SELECT \* FROM produtos WHERE status = 'Vendido'";*

*try (Connection conn = this.connect();*

*Statement stmt = conn.createStatement();*

*ResultSet rs = stmt.executeQuery(sql)) {*

*while (rs.next()) {*

*Produto produto = new Produto();*

*while (rs.next()) {*

*Produto produto = new Produto();*

*produto.setId(rs.getInt("id"));*

*produto.setNome(rs.getString("nome"));*

*produto.setPreco(rs.getDouble("preco"));*

*produto.setStatus(rs.getString("status"));*

*produtosVendidos.add(produto);*

*}*

*produtosVendidos.add(produto);*

*}*

*} catch (SQLException e) {*

*System.out.println(e.getMessage());*

*}*

*return produtosVendidos;*

*}*

Agora que já criamos os comandos que listarão e atualização o status dos produtos para ‘vendido’, vamos criar a tela onde estas informações serão mostradas. Veja no print abaixo, como ficou nossa nova classe:   
  
Texto

Descrição gerada automaticamente  
O código ficou desta forma:   
  
*import javax.swing.\*;*

*import javax.swing.table.DefaultTableModel;*

*import java.awt.\*;*

*import java.util.List;*

*public class TelaVendas extends JFrame {*

*private JTable tabelaVendas;*

*private DefaultTableModel modeloTabela;*

*public TelaVendas() {*

*setTitle("Tela de Vendas");*

*setSize(600, 400);*

*setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);*

*setLocationRelativeTo(null);*

*setLayout(new BorderLayout());*

*modeloTabela = new DefaultTableModel();*

*modeloTabela.addColumn("ID");*

*modeloTabela.addColumn("Nome");*

*modeloTabela.addColumn("Preço");*

*modeloTabela.addColumn("Status");*

*tabelaVendas = new JTable(modeloTabela);*

*JScrollPane scrollPane = new JScrollPane(tabelaVendas);*

*add(scrollPane, BorderLayout.CENTER);*

*carregarDados();*

*}*

*private void carregarDados() {*

*ProdutosDAO dao = new ProdutosDAO();*

*List<Produto> produtosVendidos = dao.listarProdutosVendidos();*

*modeloTabela.setRowCount(0);*

*for (Produto produto : produtosVendidos) {*

*modeloTabela.addRow(new Object[]{*

*produto.getId(),*

*produto.getNome(),*

*produto.getPreco(),*

*produto.getStatus()*

*});*

*}*

*}*

*public static void main(String[] args) {*

*SwingUtilities.invokeLater(() -> {*

*TelaVendas telaVendas = new TelaVendas();*

*telaVendas.setVisible(true);*

*});*

*}*

*}*Agora, temos que ir na classe ListagemView e fazer com que o botão Consultar Vendas, encontre a classe TelaVendas e mostre (ou consulte) todas as vendas. O print agora não é do VS Code e sim do Apache Net Beans, vamos utilizar ele para configurar o front end, pois é bem prático de criar e modificar botões e janelas com o JFrame Form do Apache Net Beans.   
  
Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Repare que o botão já estava criado desde a última atividade, ele apenas não funcionava. Agora inserimos o seguinte comando neste pedaço do código.   
  
*“public void actionPerformed(ActionEvent e) {*

*TelaVendas telaVendas = new TelaVendas();*

*telaVendas.setVisible(true); }”*