

**UNIVERSIDADE PAULISTA – UNIP EaD**  
**PROJETO INTEGRADO MULTIDISCIPLINAR VIII**

**Curso Superior de Tecnologia em**  
**Análise e Desenvolvimento de sistemas**

**FELIPE ÍTALO DA PAIXÃO PEREIRA – 0621436**

**CODIFICAÇÃO EM C# DO MECANISMO DE ACESSO A UM TRECHO DE BANCO  
DE DADOS, COM PROTÓTIPOS DE INTERFACE GRÁFICA COM USUÁRIO  
ASP.NET E ANDROID**

**FELIPE ÍTALO DA PAIXÃO PEREIRA – 0621436**

Codificação em C# do mecanismo de acesso a um trecho de Banco de Dados, com  
protótipos de Interface Gráfica com Usuário ASP.NET e Android

Projeto Integrado Multidisciplinar em  
Análise e Desenvolvimento de Sistemas

Projeto Integrado Multidisciplinar para obtenção do título de tecnólogo em  
Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista –  
UNIP EaD.

Orientador: Professor Robson Batista Alves

## RESUMO

Este trabalho tem como objetivo apresentar o Projeto Multidisciplinar VIII, que possui como característica desenvolver um mecanismo de acesso a um trecho de banco de dados com o uso do MySQL, utilizando a linguagem C#. Juntamente, é proposto no projeto a utilização do framework ASP.NET para a interação do trecho de banco de dados. Também foi utilizado o Android Studio para a elaboração do protótipo de interface gráfica em Android, fornecendo ao usuário as funcionalidades CRUD para os dados relacionados no trecho de banco de dados. O banco de dados que foi criado no sistema manterá o cadastro de pessoas, e juntamente da interação de ASP.NET e Android a interface gráfica permitirá que o usuário efetive operações CRUD no banco de dados. O presente projeto teve como base os ensinamentos das disciplinas de Programação Orientada a Objetos II, Desenvolvimento de software para internet e Tópicos especiais de programação orientada a Objetos, concluindo assim toda a base para que o trabalho tivesse seu desenvolvimento.

**Palavras-chave:** Desenvolvimento de Software, Análise e Desenvolvimento de Sistemas, Banco de Dados, C#, ASP.NET, Android, Crud, Programação orientada.

## ABSTRACT

This paper aims to present the Multidisciplinary Project VIII, which is characterized by the development of a database access mechanism using MySQL and the C# language. Additionally, the project proposes the use of the ASP.NET framework for interacting with the database segment. The Android Studio was employed to create a graphical interface prototype for Android, providing the user with CRUD functionalities for the data related to the database segment. The system's database will maintain a record of individuals, and through the interaction of ASP.NET and Android, the graphical interface will enable the user to perform CRUD operations on the database. This project is grounded in the teachings of Object-Oriented Programming II, Software Development for the Internet, and Special Topics in Object-Oriented Programming courses, thus providing the foundational knowledge for its execution.

**Keywords:** Software Development, Analysis and System Development, Database, C#, ASP.NET, Android, CRUD, Object-Oriented Programming.

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>6</b>
<b>2 DESENVOLVIMENTO .....</b>	<b>7</b>
<b>3 Programação Orientada a Objetos II .....</b>	<b>7</b>
<b>3.1 Código do Diagrama Proposto .....</b>	<b>8</b>
<b>3.2 Implementação da classe CarrinhoRepository na conexão com o Banco de Dados. ....</b>	<b>11</b>
<b>3.3 Implementação de lógica da classe CarrinhoRepository com o acesso ao banco de dados usando Android Studio. ....</b>	<b>13</b>
<b>4 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET .....</b>	<b>14</b>
<b>4.1 Protótipo de interface gráfica .....</b>	<b>15</b>
<b>4.2 Código-fonte ASPX do layout das Telas de Desktop .....</b>	<b>17</b>
<b>5 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS.....</b>	<b>18</b>
<b>5.1 Protótipo de interface gráfica em Android .....</b>	<b>19</b>
<b>5.2 Telas no Android Studio .....</b>	<b>19</b>
<b>5.3 Código fonte XML do layout do Android .....</b>	<b>20</b>
<b>6 CONCLUSÃO .....</b>	<b>22</b>
<b>7 REFERÊNCIAS .....</b>	<b>23</b>

## 1 INTRODUÇÃO

No escopo do presente projeto, almeja-se a concepção e apresentação de um mecanismo de acesso a um segmento específico de banco de dados integrado a um sistema de gestão, cuja função preponderante reside no registro e administração de informações referentes a indivíduos.

A abordagem abraçada para a concretização deste empreendimento compreende a implementação integral da lógica do sistema web, destacando-se pela utilização das operações CRUD (Create, Read, Update, Delete), empregando a linguagem de programação C# e a estruturação das interfaces através do framework ASP.NET, notadamente explorando as nuances da arquitetura Model-View-Controller (MVC).

Neste contexto, o projeto espera fornecer uma compreensão abrangente das etapas de desenvolvimento, desde a concepção da lógica operacional até a prototipagem visual, evidenciando as boas práticas e diretrizes preconizadas no âmbito do ASP.NET, com especial ênfase na versatilidade proporcionada pelo Editor Visual Studio 2019.

No desdobramento da empreitada, destaca-se a incursão no domínio móvel, oferecendo uma perspectiva convergente para os usuários que demandam flexibilidade e acessibilidade. Nesse sentido, a materialização de uma versão móvel do sistema em apreço é delineada através da linguagem de programação Java, orquestrada pela plataforma Android Studio. A adesão a essa abordagem visa a realçar a importância de adaptação e responsividade, demonstrando, por meio de um emulador, a interface gráfica e a eficácia do sistema em dispositivos móveis.

A importância intrínseca deste projeto não apenas repousa na materialização de um sistema de cadastro de pessoas, mas, acima de tudo, reflete o compromisso em integrar conceitos de desenvolvimento de software, explorando os recursos avançados disponíveis nas plataformas C# e Java, delineando um paradigma interdisciplinar que abraça tanto o universo web quanto a esfera móvel.

Dessa forma, busca-se não apenas atender às demandas práticas inerentes à gestão de informações pessoais, mas também contribuir para o arcabouço teórico e prático do desenvolvimento de software contemporâneo, mediante uma abordagem coesa e progressista.

## **2 DESENVOLVIMENTO**

No âmbito do desenvolvimento do trabalho, delinearemos o processo de concepção e implementação do sistema, explorando as nuances fundamentais da Programação Orientada a Objetos II. A materialização das entidades do sistema, intrínsecas ao domínio de cadastro de pessoas, será abordada no tópico 3, onde se evidenciará o código das classes fundamentais concebidas com base no diagrama de classes preconizado.

O escopo desta seção também engloba a programação orientada a objetos II, situando a implementação em um contexto teórico sólido e contextualizando a aplicação prática dos princípios da orientação a objetos. Subsequentemente, o desenvolvimento adentrará a especificidade da classe DAO (Data Access Object), onde serão elucidados os aspectos do código-fonte respaldado pelo diagrama de classes proposto.

Em paralelo, será abordada a implementação da lógica da classe CarrinhoRepository, conferindo uma abordagem detalhada sobre a interação dessa classe com o banco de dados no ambiente Android Studio. Essa progressão no desenvolvimento proporcionará uma compreensão abrangente da estruturação das entidades, suas interações com o banco de dados e a coerente aplicação dos conceitos de Programação Orientada a Objetos II no contexto do sistema em questão.

### **3 Programação Orientada a Objetos II**

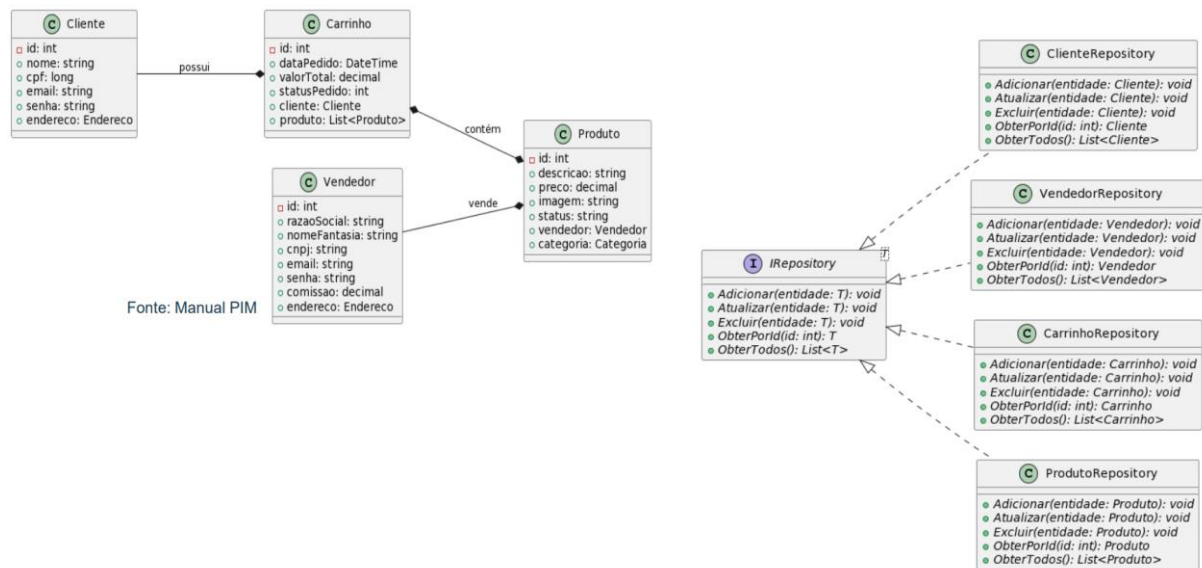
A Programação Orientada a Objetos II desempenha um papel fundamental no desenvolvimento deste projeto, oferecendo uma abordagem estruturada e modular para a concepção das entidades essenciais ao sistema de cadastro de pessoas. Este paradigma promove a organização do código por meio da encapsulação, herança e polimorfismo, possibilitando uma implementação mais coesa e flexível.

A classe de entidades, concebida a partir do diagrama de classes proposto, reflete a aplicação desses princípios, representando de maneira clara e eficiente os elementos centrais do domínio em questão. Cada atributo e método foi meticulosamente projetado para refletir a estrutura conceitual do sistema, proporcionando uma base sólida para a interação com o banco de dados e demonstrando o comprometimento com as melhores práticas da Programação Orientada a Objetos II.

### 3.1 Código do Diagrama Proposto

Seguindo as orientações do Manual do PIM VIII, foi proposto um Diagrama de classes apresentado a interação entre os formulários e a maneira a qual essas classes se relacionam, na imagem abaixo, podemos observar o Diagrama de Classes que foi proposto para utilização na criação desse Projeto.

Imagem 01 - Diagrama de Classes Proposto



Fonte: Manual do PIM

Podemos observar na imagem o relacionamento entre diversas classes, Cliente, Carrinho, Produto e Vendedor, ClienteRepository, VendedorRepository, CarrinhoRepository, ProdutoRepository. Na próxima imagem será apresentada o código criado para cada uma dessas diferentes classes. A seguir é mostrado a classe Carrinho, criado no Visual Studio.



Imagem 02 – Código da Classe Cliente

```

1  using System;
2  using System.ComponentModel.DataAnnotations;
3
4  namespace PIM_VIII.Models
5  {
6      public class Cliente
7      {
8          [Key]
9          public int Id { get; set; }
10
11          [Required(ErrorMessage = "O campo Nome é obrigatório.")]
12          [StringLength(100, MinimumLength = 2, ErrorMessage = "O Nome deve ter entre 2 e 100 caracteres.")]
13          public string Nome { get; set; }
14
15          [Required(ErrorMessage = "O campo CPF é obrigatório.")]
16          [StringLength(11, MinimumLength = 11, ErrorMessage = "O CPF deve ter 11 dígitos.")]
17          [RegularExpression(@"(0-9)+", ErrorMessage = "O CPF deve conter apenas números.")]
18          public string CPF { get; set; }
19
20          [Required(ErrorMessage = "O campo Email é obrigatório.")]
21          [EmailAddress(ErrorMessage = "O Email não é válido.")]
22          public string Email { get; set; }
23
24          [Required(ErrorMessage = "O campo Senha é obrigatório.")]
25          [StringLength(100, MinimumLength = 6, ErrorMessage = "A Senha deve ter pelo menos 6 caracteres.")]
26          public string Senha { get; set; }
27
28          [Required(ErrorMessage = "O campo EnderecoId é obrigatório.")]
29          public int? EnderecoId { get; set; }
30
31          public Endereco Endereco { get; set; }
32
33          public DateTime DataCadastro { get; set; } = DateTime.Now;
34
35          // Construtor para inicializar propriedades necessárias
36          public Cliente()
37          {
38              CPF = string.Empty;
39              Email = string.Empty;
40              Nome = string.Empty;
41              Senha = string.Empty;
42          }
43
44      }
45
46      public class Endereco
47      {
48          [Key]
49          public int Id { get; set; }
50
51      }
52
53  }

```

Fonte: Autor (2023)

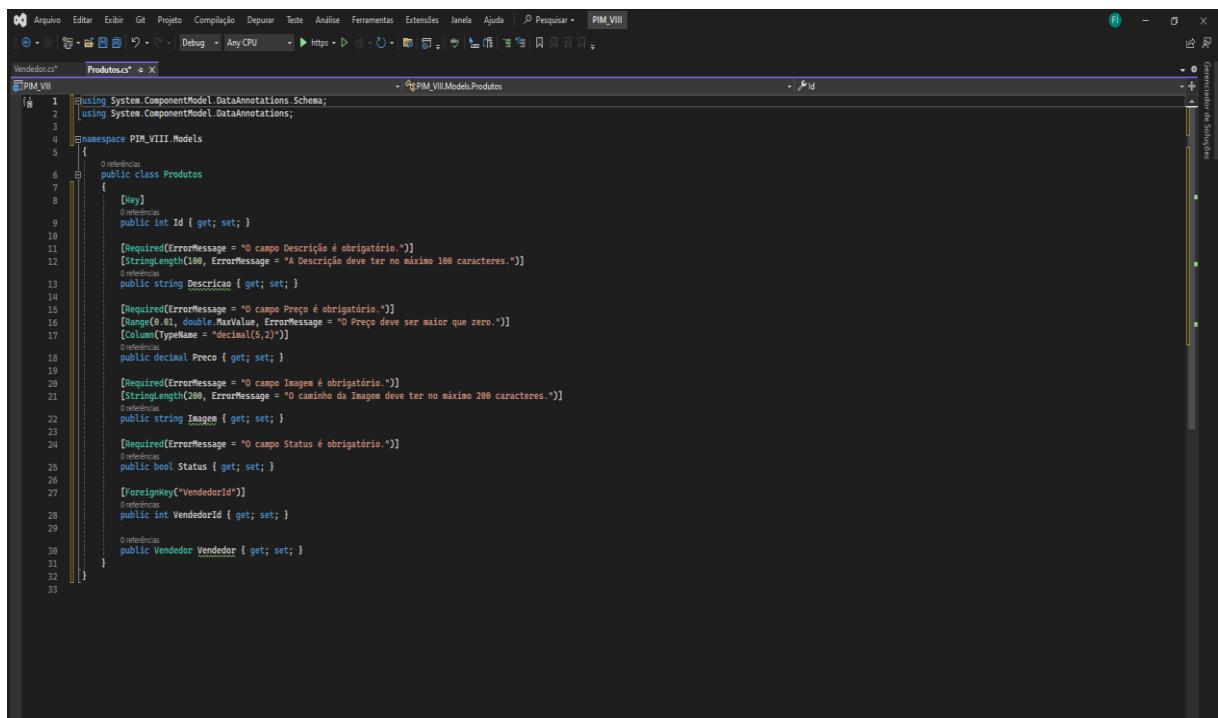
Imagem 03 – Código da Classe Carrinho

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations;
4  using System.ComponentModel.DataAnnotations.Schema;
5
6  namespace PIM_VIII.Models
7  {
8      public class Carrinho
9      {
10          [Key]
11          public int Id { get; set; }
12
13          [Required(ErrorMessage = "O campo DataPedido é obrigatório.")]
14          public DateTime DataPedido { get; set; }
15
16          [Required(ErrorMessage = "O campo ValorTotal é obrigatório.")]
17          [Column(TypeName = "decimal(7,2)")]
18          public decimal ValorTotal { get; set; }
19
20          [Required(ErrorMessage = "O campo StatusPedido é obrigatório.")]
21          public int StatusPedido { get; set; }
22
23          [ForeignKey("ClienteId")]
24          public int ClienteId { get; set; }
25
26          public Cliente Cliente { get; set; }
27
28          // Relação muitos para muitos com Produto
29          public List<ItemCarrinho> ItemCarrinho { get; set; } = new List<ItemCarrinho>();
30
31          // Construtor para inicializar propriedades necessárias
32          public Carrinho()
33          {
34              DataPedido = DateTime.Now;
35          }
36
37      }
38
39      public class ItemCarrinho
40      {
41          public int Id { get; set; }
42
43          public string Descricao { get; set; }
44
45      }
46
47  }

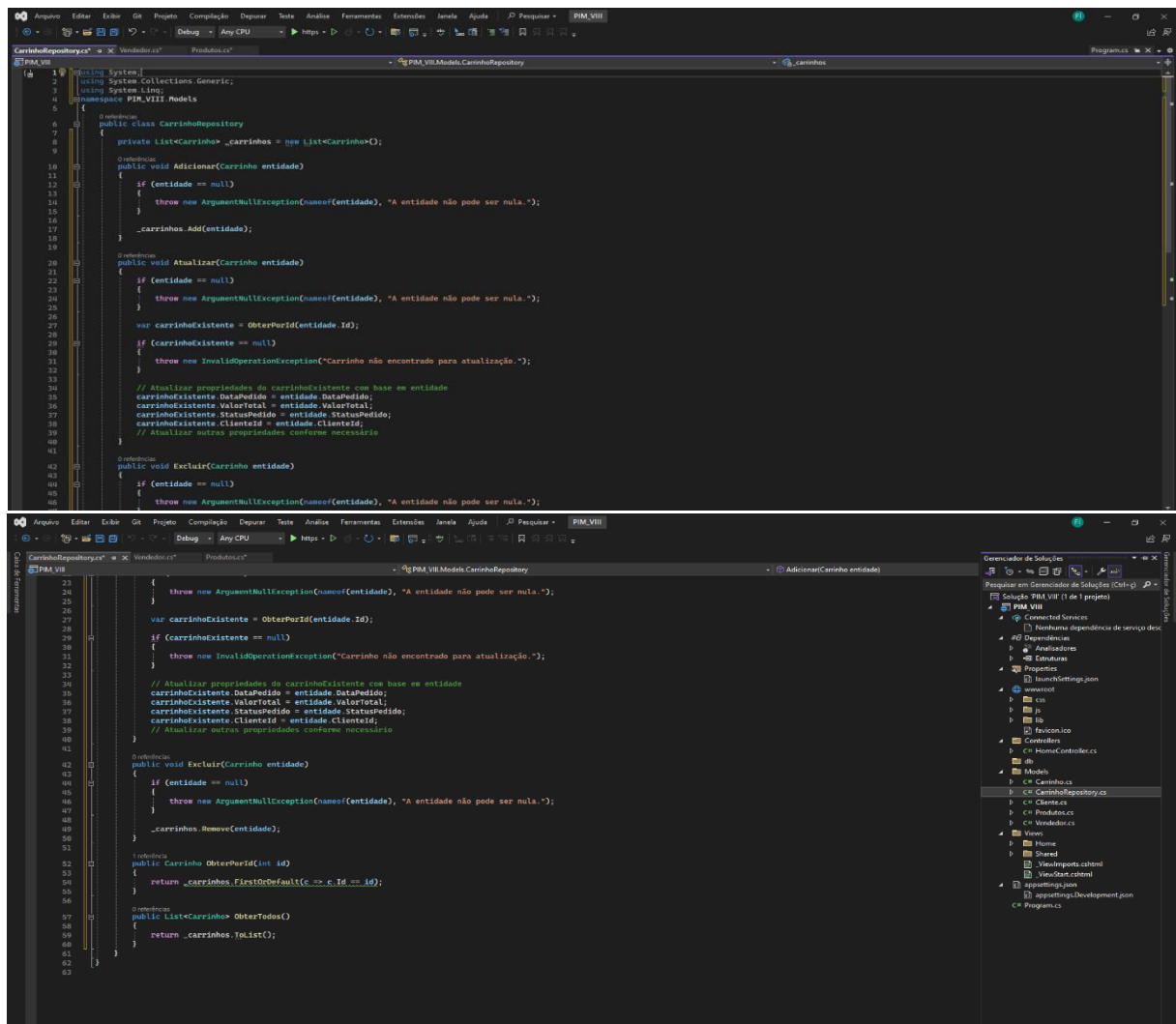
```

Imagem 04 – Código da Classe Produtos



Fonte: Autor (2023)

Imagem 05 – Código da Classe CarrinhoRepository



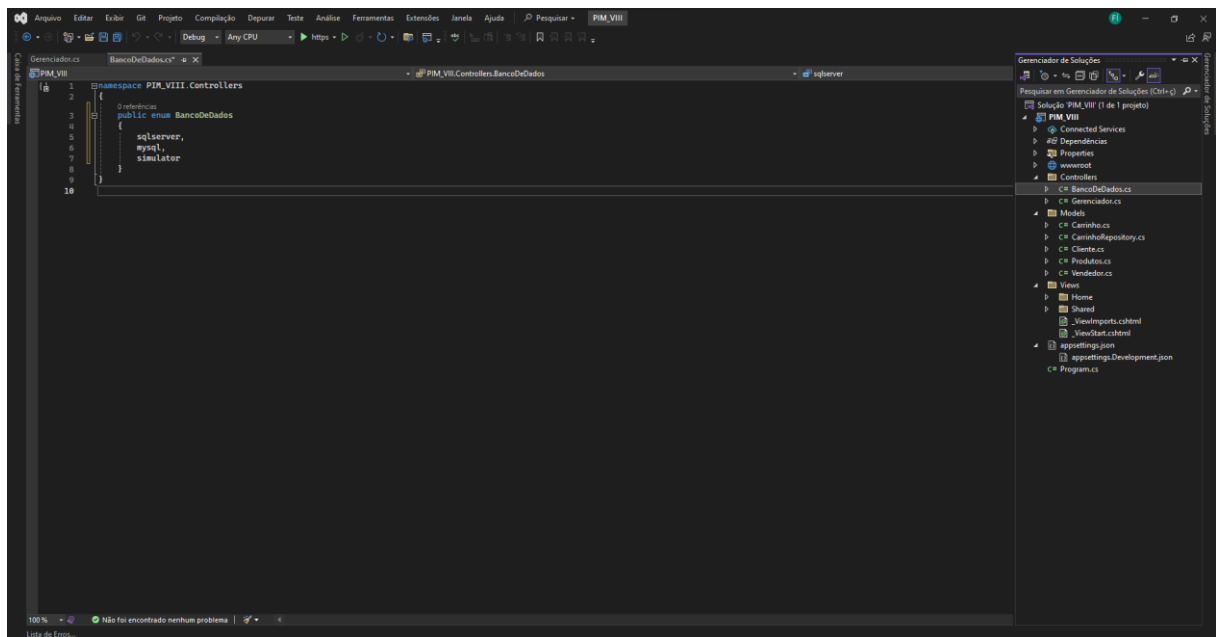
Fonte: Autor (2023)

### 3.2 Implementação da classe CarrinhoRepository na conexão com o Banco de Dados.

O desenvolvimento da implementação da lógica aplicada na classe CarrinhoRepository, responsável pela interação com o banco de dados, foi concebido no âmbito do pacote controllers, por meio das classes GerenciadorController e BancoDeDados.

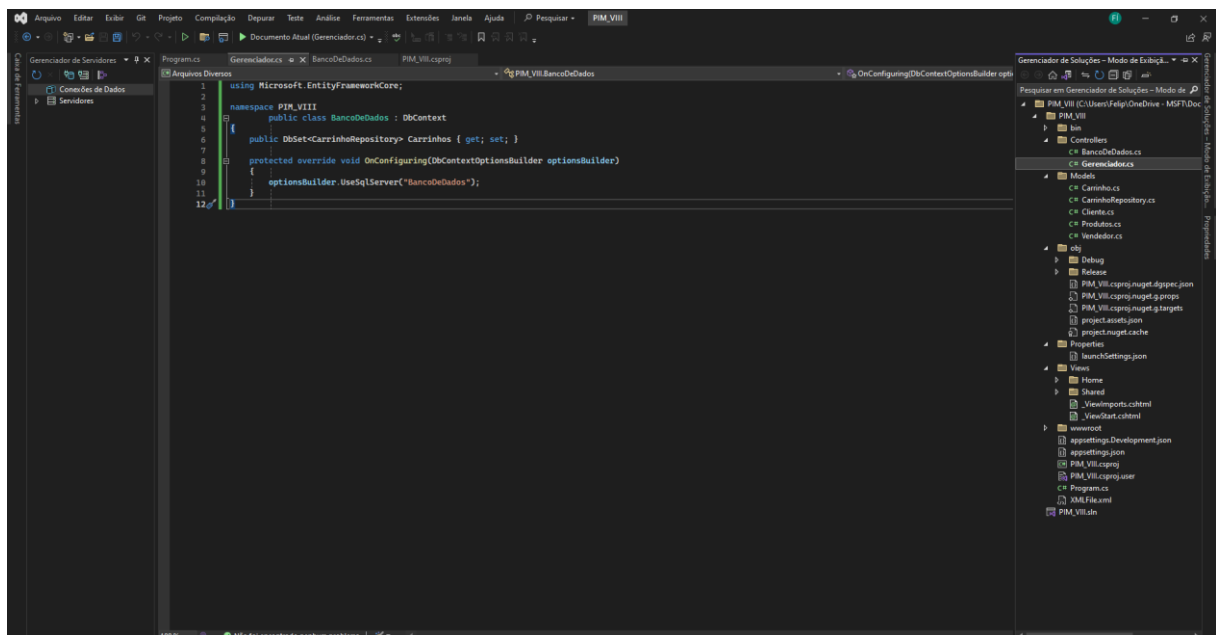
Dentro dessa classe e no Banco de Dados, foram estabelecidas as constantes representativas de cada tipo de banco de dados que será utilizado no projeto, a saber: SQL Server, MySQL e Simulador (uma implementação que simula as funções de um banco de dados). Estas constantes foram integradas à interface da CarrinhoRepository. A estrutura resultante é delineada na imagem abaixo:

Imagem 06 – Banco de Dados



Fonte: Autor (2023)

Imagem 07 – Gerenciador Banco de Dados



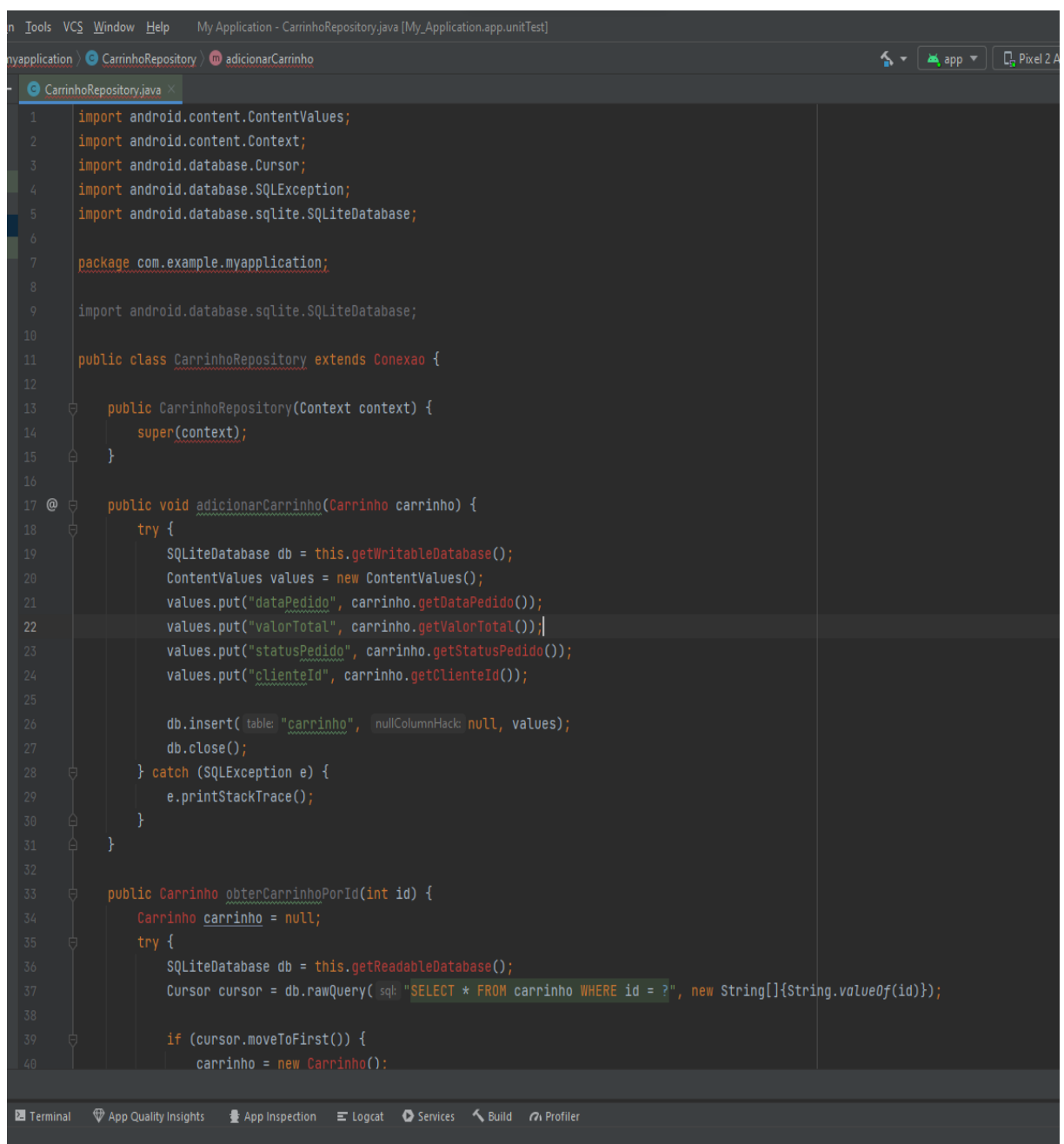
Fonte: Autor (2023)

No contexto do ASP.NET Core MVC, a conexão entre a classe CarrinhoRepository e o banco de dados foi estabelecida por meio da aplicação dos princípios de injeção de dependência. Na configuração inicial do serviço no arquivo Startup.cs, a classe BancoDeDados, que herda de DbContext e representa o contexto do banco de dados, foi registrada como um serviço, assim como a classe CarrinhoRepository. Essa prática permite que o ASP.NET Core MVC, durante o ciclo

de vida da aplicação, forneça automaticamente instâncias válidas do contexto e do repositório sempre que necessário. Essa abordagem não apenas promove uma gestão eficiente das dependências, mas também aprimora a modularidade e testabilidade do código, seguindo as melhores práticas de desenvolvimento no contexto do framework.

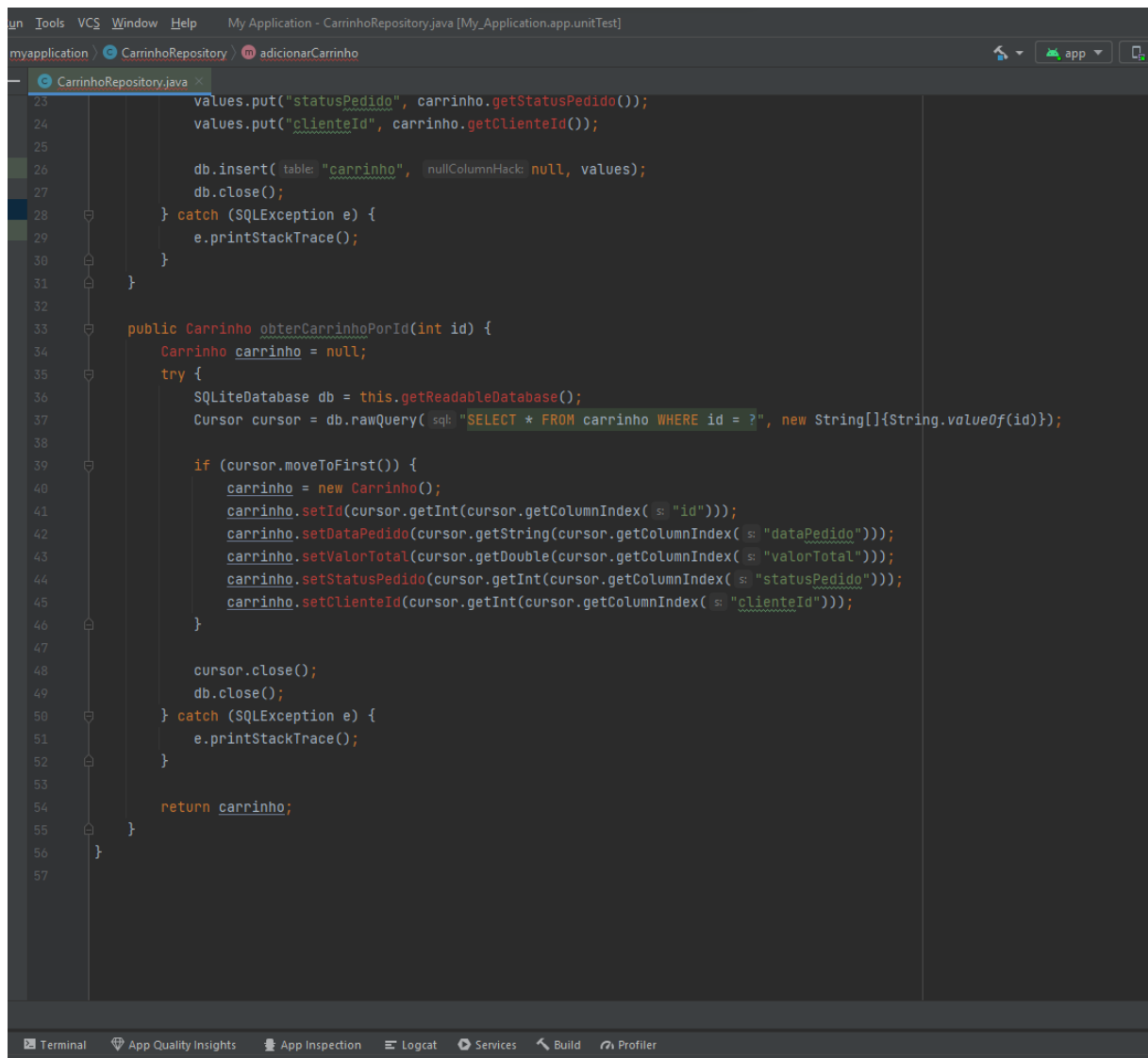
### 3.3 Implementação de lógica da classe CarrinhoRepository com o acesso ao banco de dados usando Android Studio.

Imagem 08 – CarrinhoRepository Android Studio



Fonte: Autor (2023)

Imagem 09 - CarrinhoRepository Android Studio



```

23         values.put("statusPedido", carrinho.getStatusPedido());
24         values.put("clienteId", carrinho.getClienteId());
25
26         db.insert( table: "carrinho", nullColumnHack: null, values);
27         db.close();
28     } catch (SQLException e) {
29         e.printStackTrace();
30     }
31 }
32
33 public Carrinho obterCarrinhoPorId(int id) {
34     Carrinho carrinho = null;
35     try {
36         SQLiteDatabase db = this.getReadableDatabase();
37         Cursor cursor = db.rawQuery( sql: "SELECT * FROM carrinho WHERE id = ?", new String[]{String.valueOf(id)});
38
39         if (cursor.moveToFirst()) {
40             carrinho = new Carrinho();
41             carrinho.setId(cursor.getInt(cursor.getColumnIndex( s: "id")));
42             carrinho.setDataPedido(cursor.getString(cursor.getColumnIndex( s: "dataPedido")));
43             carrinho.setValorTotal(cursor.getDouble(cursor.getColumnIndex( s: "valorTotal")));
44             carrinho.setStatusPedido(cursor.getInt(cursor.getColumnIndex( s: "statusPedido")));
45             carrinho.setClienteId(cursor.getInt(cursor.getColumnIndex( s: "clienteId")));
46         }
47
48         cursor.close();
49         db.close();
50     } catch (SQLException e) {
51         e.printStackTrace();
52     }
53
54     return carrinho;
55 }
56 }
57

```

Fonte: Autor (2023)

#### 4 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

O cenário contemporâneo revela uma marcante crescente na demanda por desenvolvimento de software para internet, representando uma tendência robusta e duradoura. A sociedade moderna está cada vez mais integrada à internet, utilizando-a para efetuar transações comerciais, realizar pagamentos, consultar extratos bancários, comunicar-se em redes sociais, entre outras atividades essenciais.

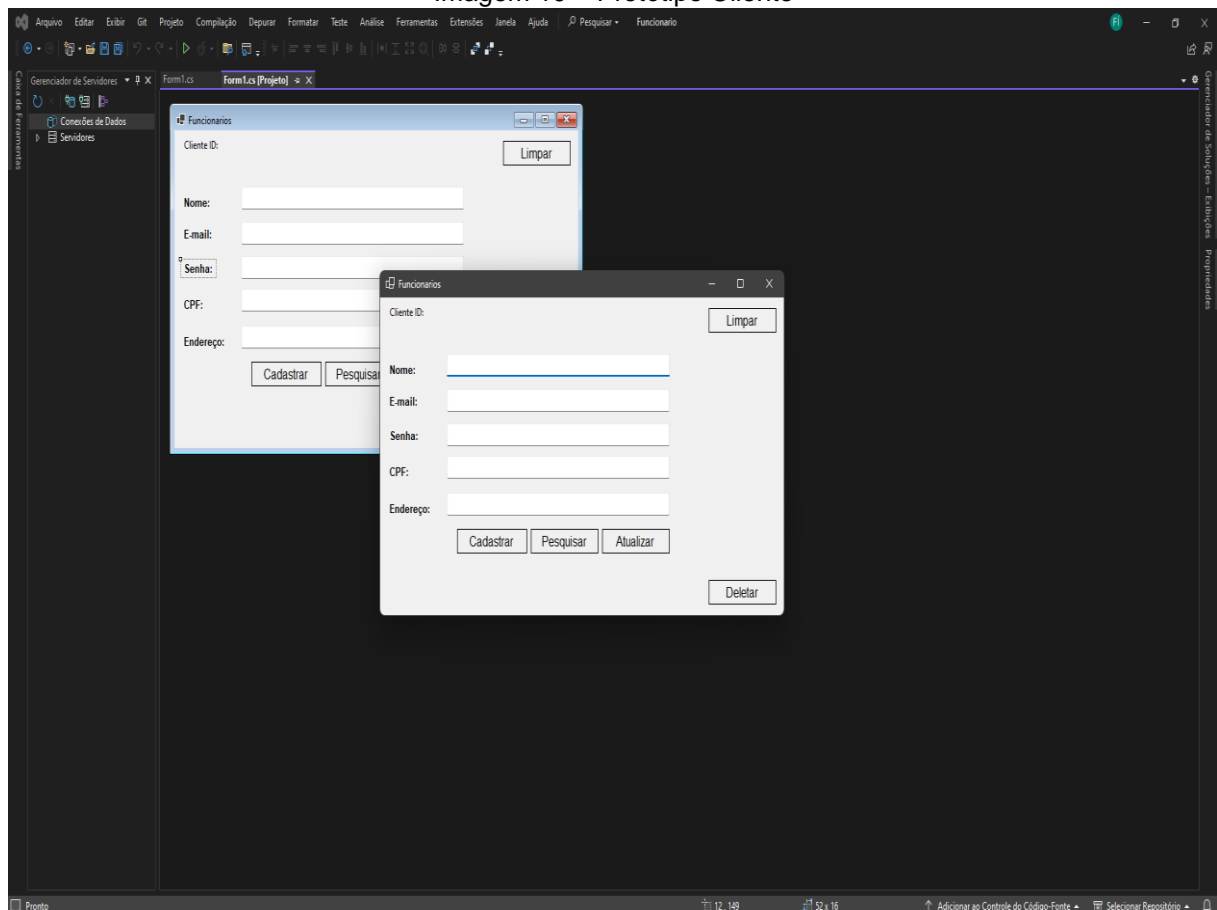
Nesse contexto, a arquitetura Modelo-Controle-Visão (MVC) emerge como uma das mais amplamente empregadas para o desenvolvimento de software para internet. Esta arquitetura facilita a interação entre a interface do usuário e os dados do banco, proporcionando respostas rápidas e dinâmicas. Sua subdivisão em três

camadas – Modelo, responsável pela implementação lógica e tratamento de dados; Controle, gerenciando as camadas Modelo e Visão; e Visão, encarregada da apresentação visual – estabelece uma estrutura eficiente para a criação de sistemas web.

No âmbito prático, foram elaborados protótipos de interface gráfica em ASP.Net com funcionalidades CRUD, empregando o navegador Google Chrome para a sua concepção e visualização. Esses protótipos representam uma materialização concreta dos princípios MVC, demonstrando a capacidade dessa arquitetura em viabilizar interações eficazes entre usuários e sistemas.

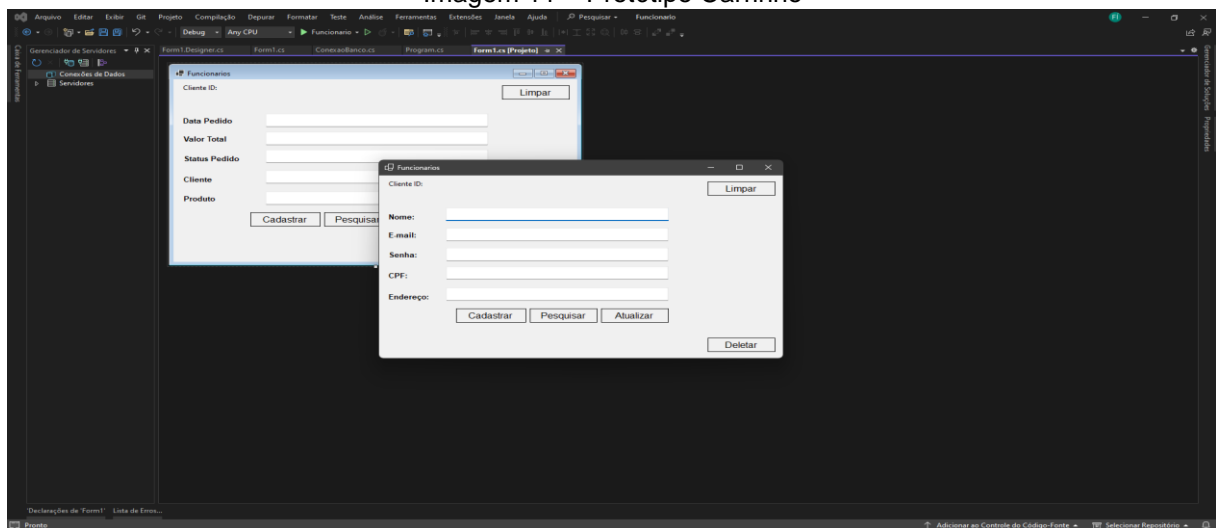
#### 4.1 Protótipo de interface gráfica

Imagem 10 – Protótipo Cliente



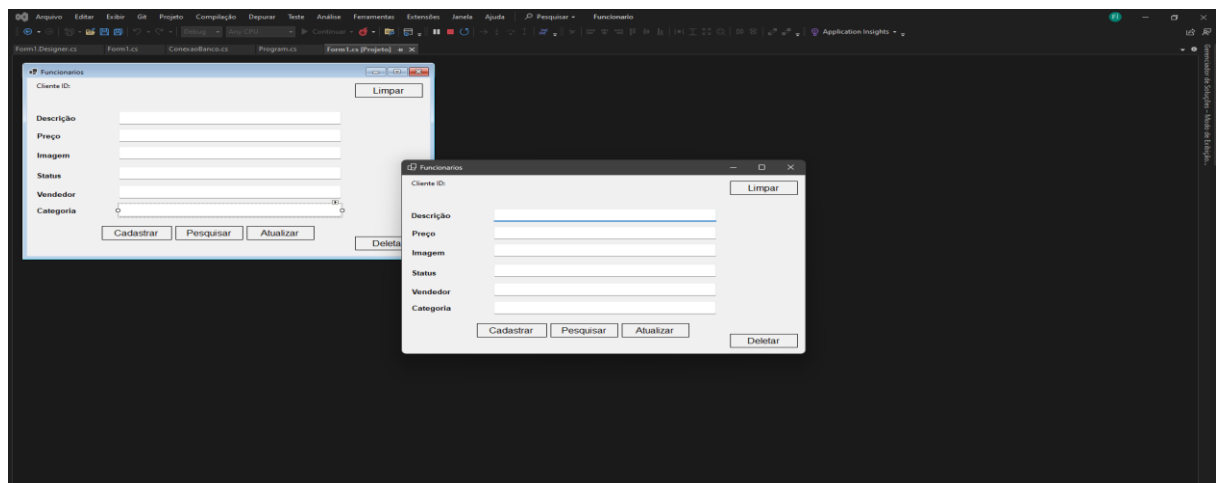
Fonte: Autor (2023)

Imagem 11 – Protótipo Carrinho



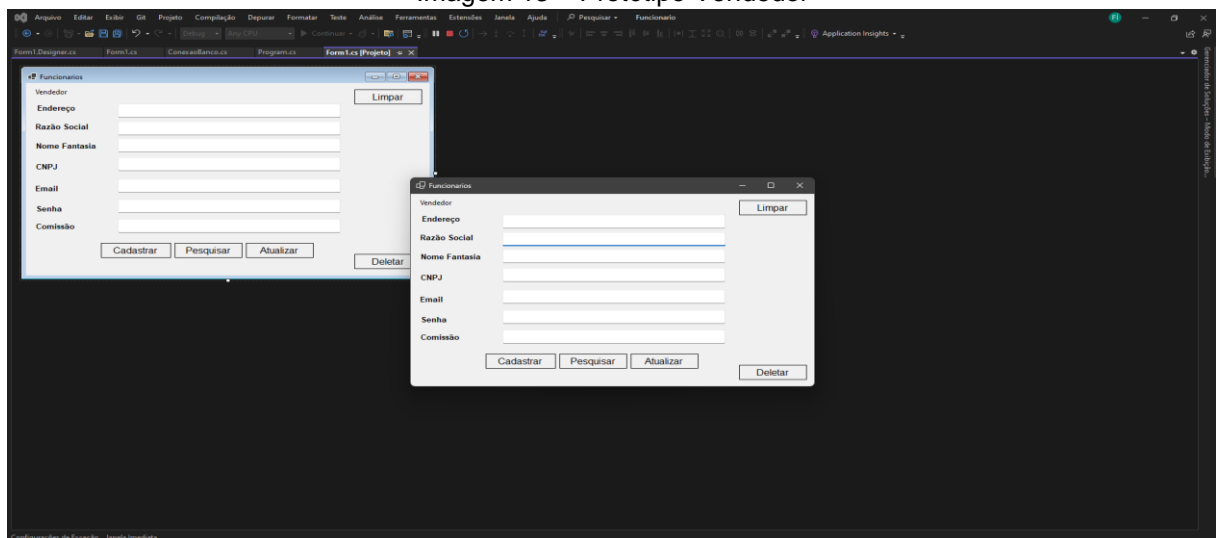
Fonte: Autor (2023)

Imagem 12 – Protótipo Produto



Fonte: Autor (2023)

Imagem 13 – Protótipo Vendedor



Fonte: Autor (2023)



## 4.2 Código-fonte ASPX do layout das Telas de Desktop

Imagem 14 – Código-fonte cadastro clientes

```

1 using Google.Protobuf.WellKnownTypes;
2 using MySql.Data.MySqlClient;
3 using Ong.BouncyCastle.Cms;
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace cliente
12 {
13     internal class cadastroCliente
14     {
15         private int id;
16         private string nome;
17         private string senha;
18         private string email;
19         private string cpf;
20         private string endereco;
21
22         2 referências
23         public int Id
24         {
25             get { return id; }
26             set { id = value; }
27         }
28
29         1 referência
30         public string Nome
31         {
32             get { return nome; }
33             set { nome = value; }
34         }
35
36         2 referências
37         public string Email
38         {
39             get { return email; }
40             set { email = value; }
41         }
42
43         1 referência
44         public string Cpf
45         {
46             get { return cpf; }
47             set { cpf = value; }
48         }
49
50         2 referências
51         public string Endereco
52         {
53             get { return endereco; }
54             set { endereco = value; }
55         }
56
57         2 referências
58         public string senha
59         {
60             get { return senha; }
61             set { senha = value; }
62         }
63
64         //método para cadastro de clientes no banco de dados.
65         0 referências
66         public bool cadastrarclientes()
67         {
68             try
69             {
70                 MySqlConnection MySqlConnection = new MySqlConnection(ConexaoBanco.bancoServidor);
71                 MySqlConnection.Open();
72
73                 string insert = $"insert into clientes (nome, email, cpf, endereco) values ('{Nome}','{Email}','{Cpf}','{Endereco}')";
74
75                 MySqlCommand comandoSql = MySqlConnection.CreateCommand();
76                 comandoSql.CommandText = insert;
77
78                 comandoSql.ExecuteNonQuery();
79                 return true;
80             }
81             catch (Exception ex)
82             {
83                 //mensagem de erro do banco de dados quando não for possível cadastrar usuários ou funcionários no banco
84                 //erro ligado ao banco de dados.
85                 MessageBox.Show("Erro no banco de dados - método cadastrarcliente: " + ex.Message);
86                 return false;
87             }
88         }
89
90         0 referências
91         public MySqlDataReader localizarcliente()
92         {
93             try
94             {
95                 MySqlConnection MySqlConnection = new MySqlConnection(ConexaoBanco.bancoServidor);
96                 MySqlConnection.Open();
97
98                 string select = $"select id, nome, email, cpf, endereco from clientes where cpf = '{cpf}'";
99                 MySqlCommand comandoSql = MySqlConnection.CreateCommand();
100                 comandoSql.CommandText = select;

```

```

78 public MySqlDataReader localizarcliente()
79 {
80     try
81     {
82         MySqlConnection MySqlConexaoBanco = new MySqlConnection(ConexaoBanco.bancoServidor);
83         MySqlConexaoBanco.Open();
84
85         string select = $"select id, nome, email, cpf, endereco from clientes where cpf = '{cpf}';";
86         MySqlCommand comandoSql = MySqlConexaoBanco.CreateCommand();
87         comandoSql.CommandText = select;
88
89         MySqlDataReader reader = comandoSql.ExecuteReader();
90         return reader;
91     }
92     catch (Exception ex)
93     {
94         MessageBox.Show("Erro no banco de dados - método localizarFuncionário: " + ex.Message);
95         return null;
96     }
97 }
98
99 0 referências
100 public bool atualizarcliente()
101 {
102     try
103     {
104         MySqlConnection MySqlConexaoBanco = new MySqlConnection(ConexaoBanco.bancoServidor);
105         MySqlConexaoBanco.Open();
106
107         string update = $"update clientes set email = '{Email}', endereco = '{Endereco}' where id = '{Id}';";
108         MySqlCommand comandoSql = MySqlConexaoBanco.CreateCommand();
109         comandoSql.CommandText = update;
110
111         comandoSql.ExecuteNonQuery();
112         return true;
113     }
114     catch (Exception ex)
115     {
116         MessageBox.Show("Erro no banco de dados - método atualizarcliente: " + ex.Message);
117         return false;
118     }
119 }
120 0 referências
121 public bool deletarcliente()
122 {
123     try
124     {
125         MySqlConnection MySqlConexaoBanco = new MySqlConnection(ConexaoBanco.bancoServidor);
126         MySqlConexaoBanco.Open();
127
128         string delete = $"delete from clientes where id = '{Id}';";
129         MySqlCommand comandoSql = MySqlConexaoBanco.CreateCommand();
130         comandoSql.CommandText = delete;
131
132         comandoSql.ExecuteNonQuery();
133         return true;
134     }
135     catch (Exception ex)
136     {
137         MessageBox.Show("Erro banco de dados - método deletarcliente: " + ex.Message);
138         return false;
139     }
140 }

```

Fonte: Autor (2023)

## 5 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Ao longo da história, a evolução tecnológica tem proporcionado ferramentas inovadoras para aprimorar a eficiência e a velocidade nas tarefas cotidianas. Analogamente à Revolução Industrial, que transformou a produção por meio da introdução de máquinas, o advento dos computadores digitais trouxe consigo a necessidade de linguagens de programação mais acessíveis.

Nesse contexto, a linguagem de programação Java se destacou como uma solução eficaz, oferecendo versatilidade para desenvolvimento multiplataforma,

permitindo que os computadores interpretem códigos de maneira compreensível para os humanos.

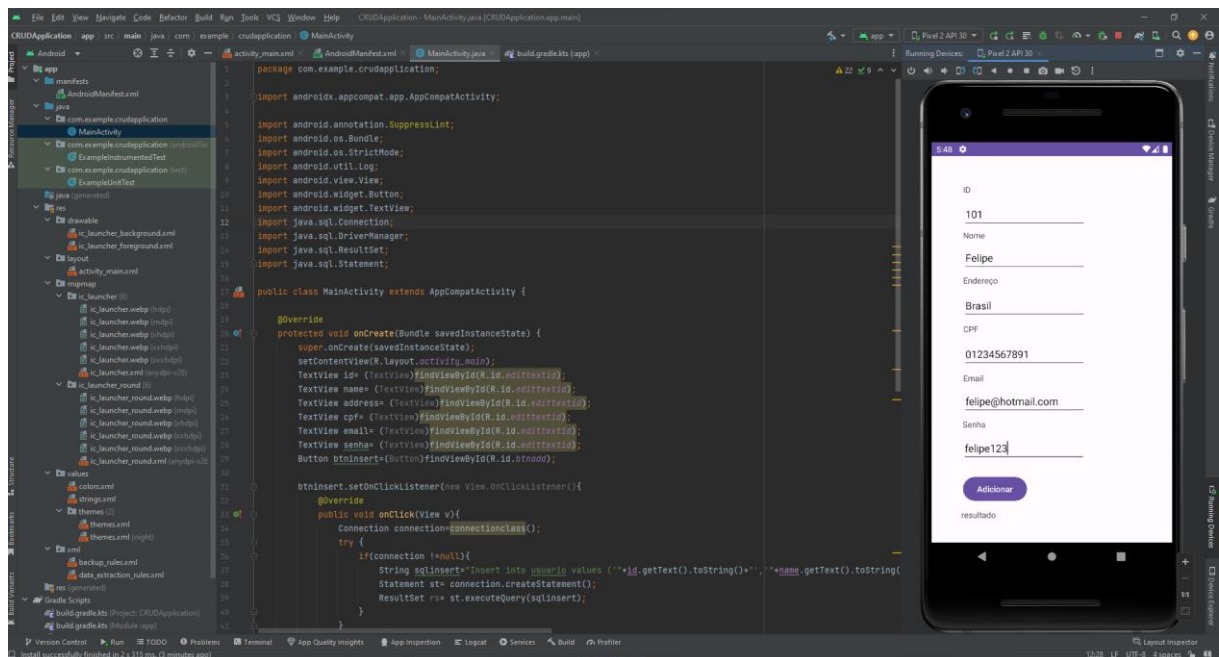
## 5.1 Protótipo de interface gráfica em Android

No âmbito deste projeto, foi desenvolvido um protótipo de interface gráfica para Android utilizando a poderosa ferramenta Android Studio. Essa etapa crucial envolveu a implementação das funcionalidades CRUD (Create, Read, Update, Delete) para proporcionar uma experiência interativa e completa aos usuários. A utilização do Android Studio, uma IDE (Integrated Development Environment) oficial para o desenvolvimento Android, permitiu uma integração fluida e eficiente na criação da interface.

As imagens subsequentes ilustram visualmente o resultado, destacando a eficácia e a usabilidade alcançadas por meio da implementação do protótipo. Esse processo não apenas valida o design conceitual, mas também estabelece as bases para a implementação posterior do aplicativo completo.

## 5.2 Telas no Android Studio

Imagem 15 – Tela inicial



Fonte: Autor (2023)



O desenvolvimento do aplicativo em Android se deu a partir da criação da classe principal denominada MainAcitiviy, a qual fazia a ligação com a classe Cadastro e ConexaoBanco para a conexão entre o app e o banco de dados do MySQL, assim, fazendo a aplicação ser rodada, conforme se mostra os códigos fonte das imagens abaixo:

```
crudapplication  MainActivity
activity_main.xml  AndroidManifest.xml  MainActivity.java  build.gradle.kts (app)
package com.example.crudapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.os.StrictMode;
import android.util.Log;

import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class MainActivity extends AppCompatActivity {

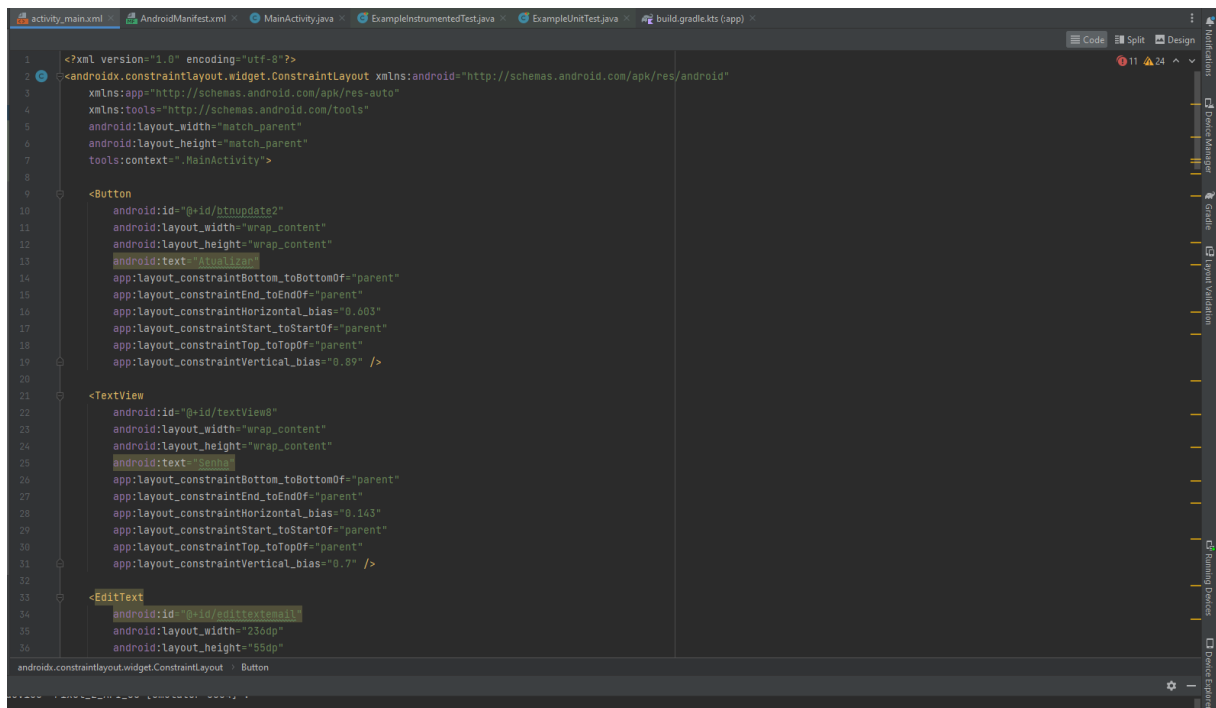
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView id= (TextView)findViewById(R.id.edittextid);
        TextView name= (TextView)findViewById(R.id.edittextid);
        TextView address= (TextView)findViewById(R.id.edittextid);
        TextView cpf= (TextView)findViewById(R.id.edittextid);
        TextView email= (TextView)findViewById(R.id.edittextid);
        TextView senha= (TextView)findViewById(R.id.edittextid);
        Button btninsert=(Button)findViewById(R.id.btnaad);
        Button btnupdate=(Button)findViewById(R.id.btnupdate);

        btninsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v){
                Connection connection=connectionClass();
                try {
                    if(connection !=null){
                        String sqlinsert=insert into usuario values ('"+id.getText().toString()+"','"+name.getText().toString()+"','"+address.getText().toString()+"','"+cpf.getText().toString()+"','"+senha.getText().toString()+"')";
                    }
                }
            }
        });
    }
}
```

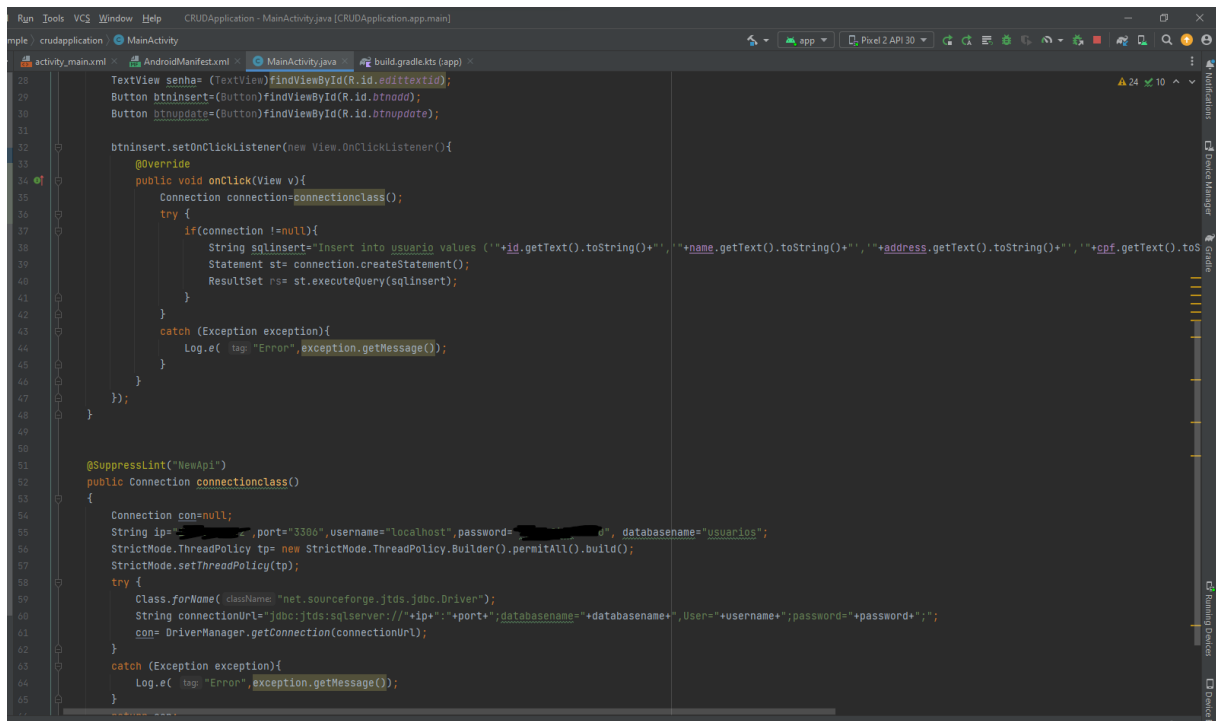
Fonte: Autor (2023)

Imagem 18 – AndroidManifest.xml



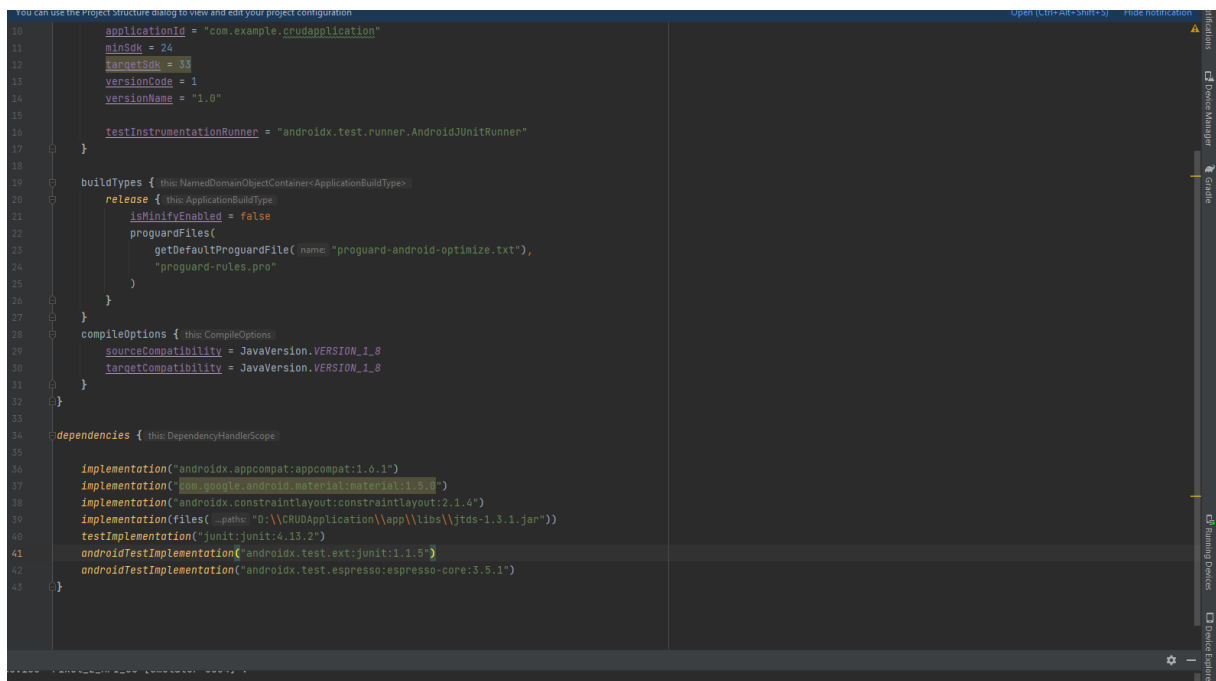
Fonte: Autor (2023)

Imagem 19 – MainActivity.java – Banco de dados



Fonte: Autor (2023)

Imagem 20 - Implementações



Fonte: Autor (2023)

## 6 CONCLUSÃO

O presente projeto foi elaborado em consonância com as diretrizes propostas neste bimestre, fundamentando-se em extensivas pesquisas sobre o tema em questão. A integração do conhecimento teórico adquirido nas disciplinas de Programação Orientada a Objetos II, Desenvolvimento de Software para Internet e Tópicos Especiais de Programação Orientada a Objetos permitiu uma aplicação prática consistente. Utilizando ferramentas como o Visual Studio 2019 e o Android Studio, foi desenvolvido protótipos e códigos, empregando as linguagens C# e Java.

No âmbito da Programação Orientada a Objetos II, foram apresentados os códigos-fontes das classes de entidades e do banco de dados conforme o diagrama fornecido no manual do PIM VIII. Destaca-se a implementação lógica entre as classes CarrinhoRepository e Banco de Dados, revelando a interconexão eficiente entre os componentes do sistema. Já no Desenvolvimento de Software para Internet, a adoção da arquitetura MVC se mostrou crucial, simplificando a troca de mensagens entre a interface do usuário e as demais camadas do sistema. Os protótipos de interface gráfica em ASP.NET para plataforma Web e as implementações das funcionalidades CRUD refletem a aplicação prática desses conceitos, evidenciando a estruturação lógica do projeto.

No contexto dos Tópicos Especiais de Programação Orientada a Objetos, exploramos o desenvolvimento do mecanismo CRUD na versão mobile utilizando o Android Studio. A criação do protótipo mobile, com a devida atenção à estruturação da tela, ressaltou a versatilidade da programação orientada a objetos na esfera mobile. Apesar da constante busca por aprimoramento, os conhecimentos adquiridos durante a elaboração desse projeto se revelaram de suma importância, proporcionando uma compreensão abrangente e eficaz das práticas de desenvolvimento de software.

## 7 REFERÊNCIAS

- Cockburn, A. **Escrevendo Casos de Uso Eficazes**, 2000.
- FOWLER, M. UML Essencial: **um breve guia para linguagem padrão**. 3 ed. Bookman, 2011.
- Lopes, P., Santos, P., & Carvalho, M. **Melhoria de processo de software com MPS.BR: estudo de caso em uma empresa de TI**. **Simpósio Brasileiro de Qualidade de Software**, 63-72, 2016.
- Marinho, S. **Desenvolvimento de Software para Internet**. São Paulo: Editora Sol, 2020
- Marinho, S. **Programação Orientada a Objetos II**. São Paulo: Editora Sol, 2020
- Osterwalder, A., & Pigneur, Y. **Business model generation: A handbook for visionaries, game changers, and challengers**. Wiley, 2010.
- Otler, I., & Armstrong, G. **Princípios de marketing**. Pearson, 2017.
- Timmons, J. A., & Spinelli, S. **New venture creation: Entrepreneurship for the 21st century**. McGraw-Hill Education, 2012.
- Van Horne, J. C., & Wachowicz, J. M. **Fundamentals of financial management**. Pearson, 2009.