Título do Trabalho:
Projeto de Sistema de Controle e Agendamento De Reservas Com .NET C#

Aluno: Felipe Rafael Barros da Silva
Disciplina: Lógica de Programação Algorítmica
Data : 10/06/2025
Instituição: UNINASSAU

1. INTRODUÇÃO

O desenvolvimento de um novo algoritmo representa um grande desafio. Para isso, é fundamental integrar uma lógica estrutural bem definida com uma escrita eficiente, a fim de criar um código cada vez mais legível, limpo e de fácil manutenção.

Neste relatório, aplicamos os conhecimentos adquiridos em lógica de programação para desenvolver um sistema de controle e agendamento de reservas. Utilizaremos a linguagem C# e abordaremos tanto conceitos básicos, como estruturas de repetição e condicionais, quanto recursos mais avançados, como programação orientada a objetos (POO), herança e polimorfismo.

O algoritmo desenvolvido funcionará em ambiente de terminal, permitindo criar novas reservas, atualizar cadastros existentes, excluir registros e consultar o total de reservas ativas no sistema. Além disso, será possível realizar consultas detalhadas, como verificar a disponibilidade de quartos e identificar quantas e quais pessoas estão programadas no sistema de reservas.

Como afirmam Deitel & Deitel (2016), dominar a lógica de programação é essencial para a construção de soluções computacionais eficientes e bem estruturadas, servindo de base para o desenvolvimento de qualquer sistema.

2. DESENVOLVIMENTO

2.1 Abordagem do problema

Em outrora, as reservas de hotéis e pousadas eram feitas completamente por interferência de manuais humanos. Essa abordagem aumentava o número nas solicitações de cadastro, atualização e até cancelamento de reservas visto que por um pequeno erro de atenção as reservas podem ser perdidas, duplicadas ou até escritas de forma equivocada. Pensando nesse sistema, buscamos aplicar uma lógica algorítmica para cuidar desse ecossistema de reservas, ajudando assim a gestão desses estabelecimentos e permitindo algo mais confiável e eficaz no controle dessas reservas.

2.2 Lógica do algoritmo

Em primeiro passo, o sistema será condicionado a priori dentro do terminal da nossa IDE(VS code), nesse terminal ao rodarmos o código poderemos ver a tela de menu principal do sistemas, como apresentado na figura 1,essa estrutura foi criada com o uso estrutura de repetição while e com auxílio de switch/case para rodar indefinidamente o sistema de tela de menu junto a sistema condiciona de opções. Dentro desse setor podemos ver todas as possíveis opções de atividades no sistema, abaixo explicaremos cada uma dessas opções de forma detalhada:

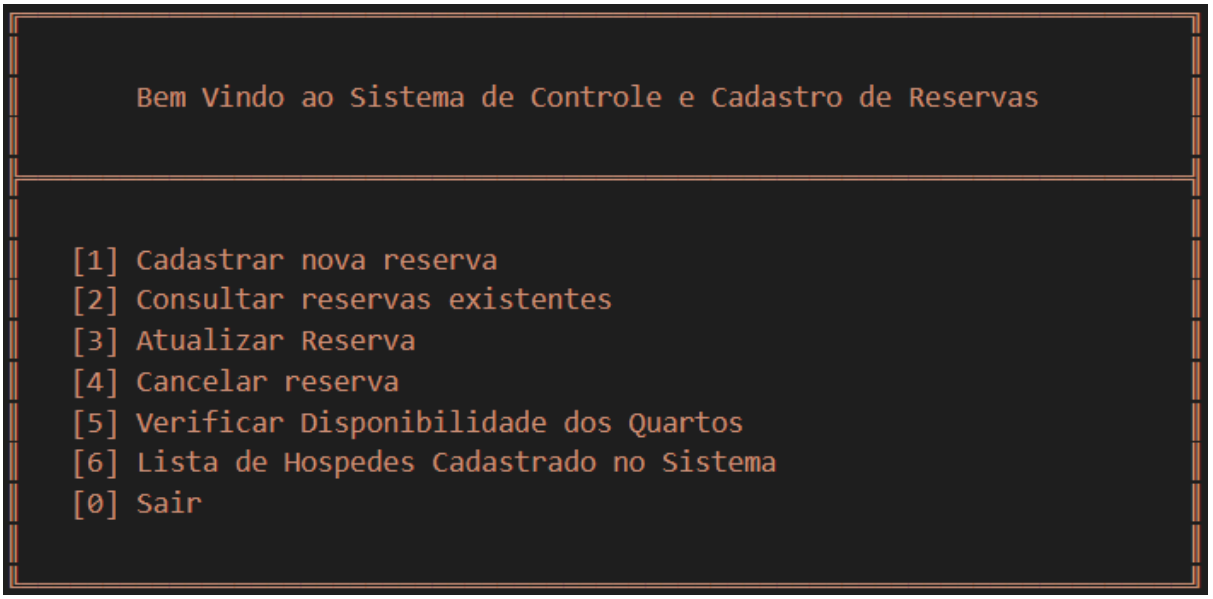

Figura 1 : Tela de menu do sistema

Opção 1: Cadastrar nova reserva
Explicação: Nesse setor, serão solicitadas todas as informações a respeito do número de pessoas que irão ocupar o quarto, o nome, cpf e idade de cada uma delas, ainda serão escolhidos os quartos que querem e a data inicial e final da reserva.

Opção 2: Consultar reservas existentes
Explicação: Podermos verificar o tutor de cada quarto(a primeira pessoa a inscrita na hora do cadastro de nova reserva), o tipo do quarto(luxo, médio, básico), o número do quarto, o preço total baseado na diária e os dias de entrada e saída e também o dia que serão feitas as entradas e saídas do quarto.

Opção 3: Atualizar reserva
Explicação: Sistema podemos escolher entre atualizar a data de saída e adicionar ou remover alguém do quarto.

Opção 4: Cancelar reserva
Explicação: Nesse segmento, você passará o número do quarto no qual quer cancelar a reserva e o sistema cuidará de cancelar a partir dele a reserva.
Opção 5: Verificar disponibilidade dos quartos
Explicação: Mostrará em tempo atual quais quartos estão ocupados, o tipo do quarto, o valor da diária, e status de ocupado ou disponível.

Opção 6: Listar hóspedes cadastrados no sistema
Explicação: Aqui mostraremos o número do quarto de todas as pessoas cadastradas no sistema, o nome da pessoa, CPF, Idade e data de quando ela foi inserida no sistema.

Opção 7: Sair
Explicação: Para encerrar o programa acione essa opção.

2.3 Código do algoritmo

Abaixo os códigos criados durante o desenvolvimento do projeto:

- **Program.cs (Código Principal do Projeto)**

O arquivo *Program.cs* representa o ponto de entrada principal do sistema. Nele, estruturamos os laços de repetição responsáveis pela interação contínua do usuário com o terminal, garantindo uma navegação fluida pelo menu de opções do sistema.

Além disso, esse módulo centraliza as principais operações do sistema, como **criar**, **atualizar**, **listar** e **cancelar** reservas, funcionando como o núcleo de controle da aplicação. A seguir, apresentamos o trecho principal do código:

```
using SistemaDeReservas.Models;

// Cria as instâncias uma vez no início
var GerenciadorPessoas = new GerenciadorPessoas();
var GerenciadorQuartos = new GerenciadorQuartos();
var GerenciadorReservas = new GerenciadorReservas(GerenciadorPessoas,
GerenciadorQuartos);

bool SistemaAtivo = true;

while (SistemaAtivo)
{
    Console.WriteLine(@"
╔═══════════════════════════════════════════════════════════════════╗
║                                                                   ║
║       Bem Vindo ao Sistema de Controle e Cadastro de Reservas     ║
║                                                                   ║
╚═══════════════════════════════════════════════════════════════════╝
╔═══════════════════════════════════════════════════════════════════╗
║                                                                   ║
║    [1] Cadastrar nova reserva                                     ║
║    [2] Consultar reservas existentes                              ║
║    [3] Atualizar Reserva                                          ║
║    [4] Cancelar reserva                                           ║
║    [5] Verificar Disponibilidade dos Quartos                      ║
║    [6] Lista de Hospedes Cadastrado no Sistema                    ║
║                                                                   ║
║    [0] Sair                                                       ║
║                                                                   ║
╚═══════════════════════════════════════════════════════════════════╝
");

    string Opção = Console.ReadLine();
```

```
    switch (Opção)
{
    case "1":
        GerenciadorReservas.CadastrarNovaReserva();
        break;
    case "2":
        Console.WriteLine(GerenciadorReservas.ListarReservas());
        Console.WriteLine("Clique em alguma tecla para voltar ao menu");
        Console.ReadLine();
        break;
    case "3":
        GerenciadorReservas.AtualizarReserva();
        break;
    case "4":
        Console.WriteLine(GerenciadorReservas.CancelarReserva());
        Console.WriteLine("Clique em alguma tecla para voltar ao menu");
        Console.ReadLine();
        break;
    case "5": //Lista Quartos cadastrados
        Console.WriteLine(GerenciadorReservas.ListarQuartos());
        Console.WriteLine("Clique em alguma tecla para voltar ao menu");
        Console.ReadLine();
        break;
    case "6":
        Console.WriteLine(GerenciadorReservas.ListarPessoasCadastradas());
        Console.WriteLine("Clique em alguma tecla para voltar ao menu");
        Console.ReadLine();
        break;
    case "0":
        SistemaAtivo = false;
        break;

    default:
        Console.WriteLine("Opção inválida! Tente novamente.");
        break;
}

}
```

- **Classes Utilizadas no Sistema**

Além da classe principal `Program.cs`, foram desenvolvidas três classes fundamentais que representam os principais objetos do sistema: `Pessoa.cs`, `Quarto.cs` e [`Reserva.cs`]().

Cada uma dessas classes é responsável por encapsular os **atributos essenciais** das entidades que modelam. A classe **`Pessoa`** armazena dados como nome, idade, CPF e Número que a pessoa está e data que foi cadastrada; a classe **`Quarto`** define propriedades como número, tipo, status de ocupação e preço da diária; por fim, a classe **`Reserva`** associa os hóspedes aos quartos reservados, considerando o período de permanência e custo total alocado.

Essas classes constituem a **base da estrutura orientada a objetos** do sistema, promovendo organização, modularidade e facilidade de manutenção. Além disso, elas são utilizadas para alimentar as listas de dados que sustentam as funcionalidades de controle e agendamento de reservas implementadas no programa.

A seguir, apresentamos os trechos principais dos arquivos `Pessoa.cs`, `Quarto.cs` e `Reserva.cs`, respectivamente:

- **Pessoa.cs - Classe de Modelo**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SistemaDeReservas.Models

{

    public class Pessoa

    {

        public string Nome { get; set; }

        public int Idade { get; set; }

        public String CPF { get; set; }

        public int NumeroQuarto { get; set; }

        public DateTime DataCadastro { get; set; }
```

```
        public Pessoa(string nome, int idade, String cpf, int
    numeroQuarto, DateTime dataCadastro)
        {
            Nome = nome;
            Idade = idade;
            CPF = cpf;
            NumeroQuarto = numeroQuarto;
            DataCadastro = dataCadastro;
        }
        public override string ToString()
        {
            return $"{Nome}";
        }
    }
    }
```

- **Quarto.cs - Classe de Modelo**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace SistemaDeReservas.Models
{
public class Quarto
{

public int Numero { get; set; } // Número do quarto
```

```
public String Tipo { get; set; } // Tipo, alto padrão, medio, baixo...
exemplo


public bool Disponivel { get; set; } = true; // Inicio com todos disponíveis

public int PrecoDiaria { get; set; }

public Quarto(int numero, string tipo, int precodiaria, bool disponivel)


            {

                Numero = numero;

                Tipo = tipo;

                PrecoDiaria = precodiaria;

                 Disponivel = disponivel;

            }}}
```

- **Reserva.cs - Classe de Modelo**

```
using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Threading.Tasks;


namespace SistemaDeReservas.Models

{

public class Reserva

{

 public DateTime DataEntrada { get; set; }

public DateTime DataSaida { get; set; }

public int PrecoDiaria { get; set; }

public int CustoTotal { get; private set; }
```

```
public int NumeroQuarto { get; set; }

public Reserva(int numeroQuarto,int precoDiaria, DateTime dataEntrada,
DateTime dataSaida)

{

NumeroQuarto = numeroQuarto;

DataEntrada = dataEntrada;

DataSaida = dataSaida;

PrecoDiaria = precoDiaria;

CalcularCustoTotal();

                    }

private void CalcularCustoTotal()

                    {

                     TimeSpan periodo = DataSaida - DataEntrada;

                     CustoTotal = PrecoDiaria * periodo.Days;

                     }}}
```

- **Classes Funcionais do Sistema**

Para complementar a estrutura orientada a objetos e garantir a organização do projeto, foram criadas classes funcionais específicas, responsáveis por gerenciar cada um dos segmentos principais do sistema. Essas classes incluem `GerenciadorPessoas.cs`, `GerenciadorQuartos.cs` e `GerenciadorReservas.cs`.

O objetivo dessas classes é encapsular as funcionalidades centrais utilizadas no programa principal (`Program.cs`), promovendo maior **separação de responsabilidades** e facilitando a manutenção do código. Elas realizam a ponte entre os dados armazenados nas entidades básicas (`Pessoa`, `Quarto` e `Reserva`) e as ações realizadas pelo usuário.

Cada classe funcional é responsável por operações específicas, como:

- Cadastrar, atualizar e excluir registros;
- Listar pessoas cadastradas;
- Verificar a disponibilidade dos quartos;
- Consultar e modificar reservas.

Além disso, essas classes fazem uso de herança e composição, promovendo reutilização de código e permitindo que os dados sejam tratados de forma mais eficiente e modular dentro da aplicação.

Abaixo, temos os códigos de GerenciadorPessoas.cs, GerenciadorQuartos.cs e GerenciadorReservas.cs respectivamente abaixo:

- **GerenciadorPessoas.cs - Classe Funcional**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Text;


namespace SistemaDeReservas.Models

{

    public class GerenciadorPessoas

    {

        private readonly List<Pessoa> _pessoas;


        public GerenciadorPessoas() //Iniciar a classe com uma lista de
pessoas

        {
```

```
            _pessoas = new List<Pessoa> { };

        }


        public String CadastrarPessoa(int NumeroQuarto, DateTime
DataCadastro)

        {

            Console.WriteLine("Qual a quantidade de pessoas no quarto?");

            int NumeroPessoas = int.Parse(Console.ReadLine()); // Ainda tem
que criar um sistema de verificaçõ se o que foi escrito é inteiro

            string Titular = "";



            for (int i = 1; i <= NumeroPessoas; i++)

            {

                Console.WriteLine("Qual o nome do do hospede " + i + " ?");

                String Nome = Console.ReadLine();

                Console.WriteLine("Qual o CPF do/da hospede " + Nome + "
?");

                String CPF = Console.ReadLine();

                Console.WriteLine("Qual o Idade do/a hospede " + Nome + "
?");

                int Idade = int.Parse(Console.ReadLine());



                _pessoas.Add(new Pessoa(Nome, Idade, CPF, NumeroQuarto,
DataCadastro));

                if (i == 1)

                {

                    Titular = Nome;

                }

            }

            return Titular;

        }
```

```
        public string ListarPessoasCadastradas()

        {

            if (!_pessoas.Any())

                return "Nenhuma pessoa cadastrada.";



            var tabela = new StringBuilder();



            // Cabeçalho com Número do Quarto

tabela.AppendLine("┌──────────────┬──────────────────┬──────────────────┬──
──────┬──────────────┐");

            tabela.AppendLine("│ N° Quarto    │ Nome             │ CPF
│ Idade  │ Data Cadastro │");

tabela.AppendLine("├──────────────┼──────────────────┼──────────────────┼──
──────┼──────────────┤");



            foreach (var pessoa in _pessoas.OrderBy(p =>
p.NumeroQuarto).ThenBy(p => p.Nome))

            {

                string cpfFormatado = pessoa.CPF;

                string nomeFormatado = pessoa.Nome.Length > 16 ?

                    pessoa.Nome.Substring(0, 13) + "..." :

                    pessoa.Nome.PadRight(16);



                tabela.AppendLine($"│ {pessoa.NumeroQuarto,-12} │
{nomeFormatado,-16} │ {cpfFormatado,-12}   │ {pessoa.Idade,6} │
{pessoa.DataCadastro:dd/MM/yyyy}    │");

            }



tabela.AppendLine("└──────────────┴──────────────────┴──────────────────┴──
──────┴──────────────┘");

            return tabela.ToString();

        }
```

```
        public void RemoverPessoasQuarto(int NumeroQuarto)



        {

            _pessoas.RemoveAll(p => p.NumeroQuarto == NumeroQuarto);

        }

        public void RemoverPessoaEspecificaQuarto(int NumeroQuarto)

        {

            Console.WriteLine("Quantas pessoas quer remover do quarto?");

            int NumeroPessoasRemover = int.Parse(Console.ReadLine());



            for (int i = 0; i < NumeroPessoasRemover; i++)// Repetir o
processo para o numero de pessoas que quer remover

            {

                for (int j = 0; j < _pessoas.Count; j++) // Avaliar quais
pessoas estão nesse quarto

                {

                    if (_pessoas[j].NumeroQuarto == NumeroQuarto)

                    {

                        Console.WriteLine("Pessoa " + j + ":" +
_pessoas[j]);

                    }







                }

                Console.WriteLine("qual o numero da pessoa que você quer
remover?");

                int Numero = int.Parse(Console.ReadLine());



                _pessoas.RemoveAt(Numero);

                Console.WriteLine("Pessoa Removida com sucesso!");

            }
```

```
        }
        public void AdicionarUmaPessoa()
        {
            try //Testando try - catch
            {
                Console.WriteLine("Digite o Nome da Nova Pessoa");
                string nome = Console.ReadLine();

                Console.WriteLine("Digite a Idade da Nova Pessoa");
                int idade = int.Parse(Console.ReadLine());

                Console.WriteLine("Digite o CPF da Nova Pessoa");
                string cpf = Console.ReadLine();

                Console.WriteLine("Digite o N° do quarto");
                int numeroQuarto = int.Parse(Console.ReadLine());


                DateTime dataCadastro = DateTime.Now;

                _pessoas.Add(new Pessoa(nome, idade, cpf, numeroQuarto,
dataCadastro));

                Console.WriteLine("Pessoa adicionada com sucesso!");
            }
            catch (FormatException)
            {
                Console.WriteLine("Erro: Formato inválido para idade ou
número do quarto.");
```

```
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Ocorreu um erro: {ex.Message}");
                }
            } } }
```

- **GerenciadorQuartos.cs - Classe Funcional**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Text;

namespace SistemaDeReservas.Models
{
    public class GerenciadorQuartos
    {
        private readonly List<Quarto> _quartos;

        public GerenciadorQuartos()
        {
            // Aqui o sistema inicia com um número de quartos X, mas n pode adicionar ainda durante a execução do codigo(Adicionar a funcionalidade)
            _quartos = new List<Quarto>
            {
                new Quarto(100, "luxo 1", 500, true),
                new Quarto(101, "luxo 2", 500, true),
                new Quarto(102, "luxo 3", 500, true),
```

```
                new Quarto(103, "luxo 4", 500, true),

                new Quarto(200, "medio 1", 250, true),
                new Quarto(201, "medio 2", 250, true),
                new Quarto(202, "medio 3", 250, true),
                new Quarto(203, "medio 4", 250, true),

                new Quarto(300, "básico 1", 100, true),
                new Quarto(301, "básico 2", 100, true),
                new Quarto(302, "básico 3", 100, true),
                new Quarto(303, "básico 4", 100, true),
            };
        }

        public string ListarQuartosFormatado()
        {
            if (!_quartos.Any())
                return "Nenhum quarto cadastrado.";

            var tabela = new StringBuilder();

            // Ajustei o tamanho da tabela para incluir a coluna de status
tabela.AppendLine("┌────────────┬──────────────────┬──────────────┬──────────────┐");
            tabela.AppendLine("│ Número     │ Tipo             │ Preço Diári │
Status     │");
tabela.AppendLine("├────────────┼──────────────────┼──────────────┼──────────────┤");
```

```
            foreach (var quarto in _quartos.OrderBy(q => q.Numero))

            {

                string status = quarto.Disponivel ? "Disponível" : "Ocupado
";

                tabela.AppendLine($"| {quarto.Numero,-8} | {quarto.Tipo,-12}
| R$ {quarto.PrecoDiaria,8} | {status} |");

            }



tabela.AppendLine("└──────────┴──────────────┴──────────────┴──────────────┘");

            return tabela.ToString();

        }



        public int AlterarDisponibilidade(bool Cadastro, int
NumeroDoQuartoRemove)

        {

            int auxiliar = 0;

            if (Cadastro == true)

            {

                int numero = 0;

                bool QuartoEncontrado = false;

                while (!QuartoEncontrado)

                {

                    Console.WriteLine("Por favor, selecione o n° do quarto
desejado:");

                    numero = int.Parse(Console.ReadLine());

                    for (int i = 0; i < _quartos.Count; i++)

                    {

                        if (_quartos[i].Numero == numero &&
_quartos[i].Disponivel == true)

                        {
```

```
                            _quartos[i].Disponivel = !_quartos[i].Disponivel;

                            QuartoEncontrado = true;

                            Console.WriteLine("O quarto " +
_quartos[i].Numero + " está disponível, por favor continue o cadastro abaixo.
");




                        }

                    }

                    if (QuartoEncontrado == false)

                    {

                        Console.WriteLine("Quarto não encontrado ou não está
disponível no momento, tente novamente.");

                    }




                }

                return numero;

            }

            else if (Cadastro == false)

            {

                for (int i = 0; i < _quartos.Count; i++)

                {

                    if (_quartos[i].Numero == NumeroDoQuartoRemove)

                    {

                        _quartos[i].Disponivel = !_quartos[i].Disponivel;

                        Console.WriteLine("O quarto " + _quartos[i].Numero +
" Agora está marcado como disponível com sucesso. ");
```

```
                }

            }

        }

        else

        {

            Console.WriteLine("Erro Do parametro passado na função
cadastro/remove");

        }

        return auxiliar;

    }



    public (int, string) DiariaDoQuarto(int NumeroQuarto)

    {

        int PrecoDiaria = 0;

        string Tipo = "";

        for (int i = 0; i < _quartos.Count; i++)

        {

            if (_quartos[i].Numero == NumeroQuarto)

            {

                PrecoDiaria = _quartos[i].PrecoDiaria;

                Tipo = _quartos[i].Tipo;

            }



        }

        return (PrecoDiaria, Tipo);

    }



    public int NumeroDeQuartosCadastrados()
```

```
        {
            int Numero = _quartos.Count;
            return Numero;
        } } }
```

- **GerenciadorReservas.cs - Classe Funcional**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Text;
using System.Diagnostics;

namespace SistemaDeReservas.Models
{
    public class GerenciadorReservas
    {
        private readonly List<Reserva> _reservas;
        private readonly GerenciadorPessoas _gerenciadorPessoas;

        private readonly GerenciadorQuartos _gerenciadorQuartos;

        private readonly List<string> TitularesQuartos;

        private readonly List<int> NumeroQuartos;
```

```
        private readonly List<string> TipoQuarto;




        public GerenciadorReservas(GerenciadorPessoas GerenciadorDePessoas,
GerenciadorQuartos GerenciadorDeQuartos)

        {

            //Fazendo cada reserva receber caracteristicas de pessoas e
quarto

            _reservas = new List<Reserva>();

            _gerenciadorPessoas = GerenciadorDePessoas;

            _gerenciadorQuartos = GerenciadorDeQuartos;



            TitularesQuartos = new List<string>();

            NumeroQuartos = new List<int>();

            TipoQuarto = new List<string>();



        }



        public String ListarPessoasCadastradas()

        {

            return _gerenciadorPessoas.ListarPessoasCadastradas();

        }

        public String ListarQuartos()

        {

            return _gerenciadorQuartos.ListarQuartosFormatado();

        }



        public String ListarReservas()
```

```
        {
            if (!_reservas.Any())
                return "Nenhuma reserva cadastrada.";

            var tabela = new StringBuilder();

tabela.AppendLine("┌──────────┬──────────────┬─────────────┬──────────────┬──────────┬──────────────┐");
            tabela.AppendLine("│ Titular  │    Tipo      │ N° Quarto   │ Preço Total │Data Entrada│ Data Saida │");
            tabela.AppendLine("│          │              │             │              │            │              │");
tabela.AppendLine("├──────────┼──────────────┼─────────────┼──────────────┼──────────┼──────────────┤");

            for (int i = 0; i < _reservas.Count; i++)
            {
                tabela.AppendLine($"│ {LimitString(TitularesQuartos[i], 7),-7}│ {LimitString(TipoQuarto[i], 12),-12} │ {LimitString(NumeroQuartos[i].ToString(), 11),-11} │R$ {LimitString(_reservas[i].CustoTotal.ToString(), 10),-10}│ {_reservas[i].DataEntrada:dd/MM/yyyy} │{_reservas[i].DataSaida:dd/MM/yyyy} │");

tabela.AppendLine("└──────────┴──────────────┴─────────────┴──────────────┴──────────┴──────────────┘");

            }
```

```
            return tabela.ToString();

        }

        private string LimitString(string input, int maxLength)

        {

            if (string.IsNullOrEmpty(input))

                return string.Empty.PadRight(maxLength);



            return input.Length <= maxLength ? input : input.Substring(0,
maxLength - 1) + "...";

        }



        public void CadastrarNovaReserva()

        {

            Console.WriteLine("Vamos iniciar o cadastro!");




            Console.WriteLine("Escolha o quarto dentre as opções
disponíveis");

            Console.WriteLine(ListarQuartos());



            int NumeroQuarto =
_gerenciadorQuartos.AlterarDisponibilidade(true, 0);



            (int PrecoDiaria, String TipoDoQuarto) =
_gerenciadorQuartos.DiariaDoQuarto(NumeroQuarto);



            Console.WriteLine("Data de Saida? formato: DD/MM/AAAA");
//Precisa verificar ainda as datas

            DateTime DataSaida = DateTime.Parse(Console.ReadLine());
```

```
            DateTime DataEntrada = DateTime.Now;



            string Titular =
_gerenciadorPessoas.CadastrarPessoa(NumeroQuarto, DataEntrada); //Cadastra as
pessoas e me retorna a primeira cadastrasda(Titular do Quarto)



            TitularesQuartos.Add(Titular);

            NumeroQuartos.Add(NumeroQuarto);

            TipoQuarto.Add(TipoDoQuarto);



            _reservas.Add(new Reserva(NumeroQuarto, PrecoDiaria, DataEntrada,
DataSaida));

        }



        public string CancelarReserva()

        {

            if (!_reservas.Any())

                return "Nenhuma reserva cadastrada.";




            int NumeroAlvo = 0;



            Console.WriteLine(ListarQuartos());

            Console.WriteLine("Qual o quarto que você quer cancelar a
reserva?");

            int NumeroDoQuarto = int.Parse(Console.ReadLine());



            for (int i = 0; i < _reservas.Count; i++)

            {

                if (NumeroDoQuarto == NumeroQuartos[i])
```

```
                {
                    NumeroAlvo = i;

                    _gerenciadorQuartos.AlterarDisponibilidade(false,
NumeroDoQuarto);
                    _gerenciadorPessoas.RemoverPessoasQuarto(NumeroDoQuarto);
                    _reservas.RemoveAll(r => r.NumeroQuarto ==
NumeroDoQuarto);
                    TitularesQuartos.RemoveAt(NumeroAlvo);
                    NumeroQuartos.RemoveAt(NumeroAlvo);
                    TipoQuarto.RemoveAt(NumeroAlvo);
                    return "Reserva do Quarto Cancelada";
                }
            }
            Console.WriteLine("Quarto Não encontrado ou Não está Ocupado,
tente novamente.");




            return "";
        }



        public void AtualizarReserva()
        {

            Console.WriteLine(@"
```

```
            ╠
╠
            ╠   [1] Deseja atualizar um nova data de saida
╠
            ╠   [2] Deseja adicionar ou remover Pessoas de um mesmo quarto
╠
            ╠   [ ] Pressione qualquer outra tecla voltar ao menu
╠
            ╠
╠
╚═══════════════════════════════════════════════════════════════════════════╝
            ");
            string Opcao = Console.ReadLine();


            switch (Opcao)
            {
                //Muda data de Saida
                case "1":
                    bool Ativo = true;
                    while (Ativo) //Rodará até que se acha o quarto válido
                    {
                        Console.WriteLine(ListarQuartos());
                        Console.WriteLine("Dentre os quartos ocupados, qual
você deseja modificiar a data limite de térmido da reserva");
                        int QuartoPedido = int.Parse(Console.ReadLine());



                        if (NumeroQuartos.Count > 0)
                        {
                            for (int i = 0; i < NumeroQuartos.Count; i++)
```

```
                            {
                                if (NumeroQuartos[i] == QuartoPedido)
                                {
                                    DateTime novaData;
                                    Console.WriteLine("Data de saida antiga:
" + _reservas[i].DataSaida);
                                    Console.WriteLine("Escolha nova data de
saida (dd/MM/yyyy):");
                                    while
(!DateTime.TryParse(Console.ReadLine(), out novaData))
                                    {
                                        Console.WriteLine("Formato inválido.
Digite novamente (dd/MM/yyyy):");
                                    }
                                    _reservas[i].DataSaida = novaData;
                                }
                            }
                        }
                        else
                        {
                            Console.WriteLine("Não há quartos cadastrados");
                            Ativo = false;
                        }
                    }
                    break;
                //Adicionar ou Remover Pessoa de um quarto
                case "2":
                    Console.WriteLine("Para adicionar pressione 1, para
remover pressione 2");
                    string Opcao2 = Console.ReadLine();
```

```
                    if (Opcao2 == "1")
                    {
                        Console.WriteLine("Quantas pessoas quer adicionar no
quarto?");
                        int NumeroPessoasAdd = int.Parse(Console.ReadLine());

                        for (int i = 0; i < NumeroPessoasAdd; i++)
                        {
                            _gerenciadorPessoas.AdicionarUmaPessoa();
                        }
                    }
                    else if (Opcao2 == "2")
                    {
                        Console.WriteLine("De qual quarto deseja remover as
pessoas");
                        int NumeroQuarto = int.Parse(Console.ReadLine());

_gerenciadorPessoas.RemoverPessoaEspecificaQuarto(NumeroQuarto);
                    }



                    else
                    {
                        Console.WriteLine("Opção inválida");
                    }

                    break;
                default:
```

```
            break;

    } } } }
```

3. CONCLUSÃO

Como primeira análise, o sistema se comportou conforme o esperado, entregando todas as funcionalidades propostas. Foi possível desenvolver uma aplicação funcional que atende à demanda de hotéis, pousadas e outros estabelecimentos similares que necessitam de um sistema confiável para controle e agendamento de reservas.

Apesar de eficiente, o algoritmo ainda apresenta pequenos pontos que podem ser otimizados, tanto em desempenho quanto na estrutura de código. No entanto, de maneira geral, o sistema apresenta uma lógica bem estruturada, escrita limpa e modularidade suficiente para aplicação em cenários reais.

Durante o desenvolvimento do projeto, não apenas os conhecimentos em programação com .NET e C# foram consolidados, mas também houve um grande avanço na capacidade de abstração algorítmica, raciocínio lógico e aplicação de conceitos como orientação a objetos, polimorfismo e herança. Esses elementos foram essenciais para a construção de uma solução escalável e de fácil manutenção.

A experiência permitiu compreender de forma mais clara a importância de práticas como o código limpo (clean code) e a separação de responsabilidades, que são fundamentais no desenvolvimento de sistemas de qualidade. Como afirma Robert C. Martin no seu livro Clean Code: A Handbook of Agile Software Craftsmanship, "códigos limpos são como poesia bem escrita: dizem o máximo com o mínimo de palavras".

Além disso, o projeto abriu portas para futuras melhorias, como integração com bancos de dados, criação de APIs e eventual publicação em plataformas web. Trata-se de uma base sólida para aplicações mais complexas no futuro.

Por fim, o repositório do projeto será disponibilizado no GitHub(**https://github.com/Felipe-Rafael-Barros/Projeto---Sistema-de-Controle-e-Agendamento-de-Reservas**) para contribuições, sugestões e eventuais evoluções. Esta etapa representa um marco importante na formação prática e

teórica em lógica de programação, reforçando o papel da disciplina na preparação para desafios do mundo real.

4. REFERÊNCIAS

[1] Martin, Robert C. *Código Limpo: Habilidades Práticas do Agile Software*. 1ª ed. Alta Books, 2009.

[2] Microsoft Docs - C# Programming Guide. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/. Acesso em: 08 jun. 2025.

[3] Digital Innovation One - https://www.dio.me/articles/programacao-orientada-a-objetos-com-c

[4] DEITEL, Paul; DEITEL, Harvey. Fundamentos de Programação: algoritmos, lógica e C/C++. 7. ed. São Paulo: Pearson, 2016.