### Estrutura completa sugerida para seu Program.cs

1. **Usings + Configurações iniciais**
2. **Listas e variáveis globais**
3. **Método** MenuPrincipal()
4. **Todos os submenus:**
   * MenuReservas()
   * MenuSuites()
   * MenuHospedes()
   * MenuComprovantes()
   * MenuRelatorios()
5. **Métodos utilitários e de ação:**
   * Suítes: AdicionarSuite, ListarSuites, ExcluirSuite
   * Hóspedes: AdicionarHospede, ListarHospedes, ExcluirHospede, ListarHospedesComFiltro, BuscarHospedePorNome, BuscarHospedePorSuiteId, BuscarHospedePorData
   * Reservas: AdicionarReserva, ListarReservas, EncerrarReserva, ExcluirReserva
   * Comprovantes: EmitirComprovanteReserva
6. **Informações do sistema:**
   * MostrarSobreSistema()

### Estrutura planejada para o MenuReservas.cs

Ele vai reunir os seguintes pontos:

#### 📌 **Menu principal de reservas:**

* Criar nova reserva
* Visualizar reservas ativas
* Consultar reservas por espaço ou hóspede
* Encerrar ou cancelar reserva

#### 👥 **Hospedagem dinâmica:**

* AdicionarHospede(hospedesGerais, hospedesDaReserva)
* SubMenuHospedes(reservaSelecionada, hospedesGerais)

#### 🎯 **Funções auxiliares já identificadas:**

* Calcular valor total
* Mostrar previsão de check-out
* Histórico de reservas (opcional)
* Cancelamento de reservas com aviso visu

#### Parte 1 – Menu principal de reservas

* Exibe o menu e captura as opções do usuário
* Chama os métodos de criação, visualização e controle

#### 🧾 Parte 2 – Criar nova reserva

* Gera nova reserva com espaço, dias e hóspedes
* Usa AdicionarHospede(...)

#### 👥 Parte 3 – Submenu de hóspedes da reserva

* Permite adicionar ou consultar hóspedes dentro de uma reserva
* Usa SubMenuHospedes(...)

#### 🗂️ Parte 4 – Visualização, cancelamento e encerramento

* Listar reservas ativas
* Cancelar reserva
* Encerrar e calcular valor total

1-Adicionar

void AdicionarSuite()

{

Console.Clear();

Console.WriteLine($"Data: {DateTime.Now:dd/MM/yyyy} | Hora: {DateTime.Now:HH:mm:ss}");

Console.WriteLine("==================================");

Console.WriteLine(" Tipo | Capacidade | Diária ");

Console.WriteLine("---------------------------------");

Console.WriteLine("Padrão | 2 pessoas | R$ 80,00 ");

Console.WriteLine("---------------------------------");

Console.WriteLine("Premium | 3 pessoas | R$ 100,00 ");

Console.WriteLine("---------------------------------\n");

Console.WriteLine("- Estadia a partir de 10 dias: 10% de desconto");

Console.WriteLine("- Ultrapassado o tempo da diária: acréscimo de R$5,00 por hora\n");

Console.WriteLine("=== RESERVA DE SUÍTE ===\n");

int dias;

do

{

Console.Write("Para quantos dias deseja reservar? ");

string? entrada = Console.ReadLine();

if (!int.TryParse(entrada, out dias) || dias <= 0)

{

Console.WriteLine("Ops. Digite o número de dias para reservar.\n");

}

} while (dias <= 0);

int pessoas;

do

{

Console.Write("Para quantas pessoas? ");

string? entrada = Console.ReadLine();

if (!int.TryParse(entrada, out pessoas) || pessoas <= 0)

{

Console.WriteLine("Ops. Digite o número de pessoas.\n");

}

} while (pessoas <= 0);

if (pessoas > 3)

{

Console.WriteLine("\nA suíte Padrão comporta até 2 pessoas e a suíte Premium até 3 pessoas.");

Console.WriteLine("Deseja fazer essa reserva agora e, concluindo, criar outra para os excedentes? (S/N)\n");

string? continuar;

do

{

Console.Write("Sua resposta: ");

continuar = Console.ReadLine()?.Trim().ToUpper();

if (continuar != "S" && continuar != "N")

{

Console.WriteLine("Ops. Digite 'S' para sim ou 'N' para não.");

}

} while (continuar != "S" && continuar != "N");

if (continuar == "N")

{

Console.WriteLine("\nTudo bem! Pressione qualquer tecla para retornar ao menu.");

Console.ReadKey();

return;

}

do

{

Console.Write("\nQuantas pessoas deseja incluir nesta primeira reserva? (máx. 3): ");

string? entrada = Console.ReadLine();

if (!int.TryParse(entrada, out pessoas) || pessoas <= 0 || pessoas > 3)

{

Console.WriteLine("Ops. Digite um valor entre 1 e 3.");

}

else

{

break;

}

} while (true);

}

Console.WriteLine("\nSuítes disponíveis:");

if (pessoas <= 2)

{

Console.WriteLine("1 - Suíte Padrão (até 2 pessoas, R$80,00/diária)");

Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

}

else if (pessoas == 3)

{

Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

}

Console.Write("\nDigite o número da suíte desejada: ");

string? tipoEscolhido = Console.ReadLine();

Suite? suite = null;

if (tipoEscolhido == "1" && pessoas <= 2)

{

suite = new Suite("Padrão", 2, 80);

}

else if (tipoEscolhido == "2")

{

suite = new Suite("Premium", 3, 100);

}

else

{

Console.WriteLine("\nOpção inválida ou incompatível com a quantidade de hóspedes.");

Console.ReadKey();

return;

}

// Verificação final da capacidade

if (pessoas > suite.Capacidade)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"\n❌ A suíte {suite.TipoSuite} comporta no máximo {suite.Capacidade} pessoa(s).");

Console.ResetColor();

Console.WriteLine("\nTudo bem. Pressione qualquer tecla para retornar ao menu...");

Console.ReadKey();

return;

}

suite.Id = suites.Count + 1;

suites.Add(suite);

Reserva novaReserva = new Reserva(hospedes, suite, dias);

novaReserva.CadastrarSuite(suite);

// Perguntar se deseja cadastrar os hóspedes agora

string? resposta;

do

{

Console.Write("\nDeseja cadastrar os nomes dos hóspedes agora? (S/N): ");

resposta = Console.ReadLine()?.Trim().ToUpper();

if (resposta != "S" && resposta != "N")

{

Console.WriteLine("Ops. Digite 'S' para sim ou 'N' para não.");

}

} while (resposta != "S" && resposta != "N");

if (resposta == "S")

{

List<Pessoa> listaTemporaria = new();

for (int i = 1; i <= pessoas && i <= suite.Capacidade; i++)

{

Console.Write($"\nDigite o nome do hóspede {i}: ");

string? nome = Console.ReadLine()?.Trim();

Console.Write("Digite o sobrenome do hóspede: ");

string? sobrenome = Console.ReadLine()?.Trim();

if (string.IsNullOrWhiteSpace(nome) || string.IsNullOrWhiteSpace(sobrenome))

{

Console.WriteLine("Nome e sobrenome são obrigatórios. Tente novamente.");

i--;

continue;

}

listaTemporaria.Add(new Pessoa(nome, sobrenome));

if (listaTemporaria.Count == suite.Capacidade)

{

Console.WriteLine($"\n⚠️ A suíte atingiu sua capacidade máxima de {suite.Capacidade} hóspede(s).");

break;

}

}

try

{

novaReserva.CadastrarHospedes(listaTemporaria);

hospedes.AddRange(listaTemporaria);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\nHóspedes cadastrados com sucesso!");

Console.ResetColor();

Thread.Sleep(1000);

}

catch (Exception ex)

{

Console.WriteLine($"\nErro ao cadastrar hóspedes: {ex.Message}");

}

}

else

{

Console.WriteLine("Hóspedes não cadastrados. Você poderá adicioná-los depois.");

}

reservas.Add(novaReserva);

Console.Clear();

Console.WriteLine("==================================");

Console.WriteLine($"Data: {DateTime.Now:dd/MM/yyyy} | Hora: {DateTime.Now:HH:mm:ss}");

Console.WriteLine("==================================");

Console.WriteLine("\n=== RESUMO DA RESERVA ===");

Console.WriteLine($"Suíte {suite.TipoSuite}. Cadastro {suite.Id}");

Console.WriteLine($"Valor da diária: {suite.ValorDiaria.ToString("C")}");

Console.WriteLine($"Dias reservados: {dias}");

Console.WriteLine($"Quantidade de hóspedes: {novaReserva.Hospedes.Count}");

Console.WriteLine("----------------------------------------");

decimal valorSemDesconto = novaReserva.DiasReservados \* novaReserva.Suite.ValorDiaria;

decimal valorEstimado = novaReserva.CalcularValorDiaria();

decimal descontoAplicado = valorSemDesconto - valorEstimado;

Console.WriteLine($"Subtotal (sem desconto): {valorSemDesconto.ToString("C")}");

Console.WriteLine($"Desconto aplicado: {descontoAplicado.ToString("C")}");

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Total atual com desconto: {valorEstimado.ToString("C")}");

Console.ResetColor();

Console.WriteLine("----------------------------------------");

if (novaReserva.Hospedes == null || !novaReserva.Hospedes.Any())

{

Console.WriteLine("Nenhum hóspede foi cadastrado nesta reserva.");

Console.WriteLine("Acesse a opção 4 do MENU para cadastrar depois.");

}

else

{

Console.WriteLine($"\nQuantidade de hóspedes: {novaReserva.Hospedes.Count}");

Console.WriteLine("Lista de hóspedes:");

foreach (var h in novaReserva.Hospedes)

{

Console.WriteLine($"- {h.NomeCompleto}");

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\nReserva realizada com sucesso !!!");

Console.ResetColor();

Thread.Sleep(1000);

Console.WriteLine("\nPressione qualquer tecla para retornar ao menu principal...");

Console.ReadKey();

}

Pasta raiz: trilha-net-explorando-desafio: dentro tem pasta Models e os arquivos

DesafioPprojetoHospedagem.csproj, Program.cs, trilha-net-explorando-desafio.sln e IniciarSistema.bat

Dentro da pasta Models tem: Pessoa.cs, Reserva.cs, Hospede e EspacoHospedagem.cs,

**Trilha-net-explorando-desafio**

### |

### ├── Menus/

### │ ├── MenuHospedagem.cs

### │ ├── MenuComprovante.cs

### │ ├── MenuEmitirComprovanteReserva.cs

### │ ├── MenuEspaco.cs

### │ └── MenuHospedes.cs

### │ └── MenuPrincipal

### │ ├── MenuRelatorios.cs

### │ ├── MenuReserva.cs

### │ └── MostrarSobreSistemas.cs

### │

### ├── Models/

### │ ├── Hospede.cs

### │ ├── Pessoa.cs

### │ └── Reserva.cs

### |

### |------ Utils/

### | |\_\_\_ EntradaHelper.cs

### │

### ├── Program.cs

### ├── DesafioProjetoHospedagem.cs

### └── trilha-net-explorando-desafio.sln

### Models/

Modelos de dados: só propriedades e lógica interna.

* Hospede.cs → classe Hospede
* Pessoa.cs → se existir, como base
* Reserva.cs → classe Reserva
* EspacoHospedagem.cs → classe EspacoHospedagem

### Menus/

Responsáveis pelos fluxos e opções do usuário (as “telinhas” do terminal).

* MenuPrincipal.cs → método Exibir(...)
* MenuHospedes.cs → Exibir(), AdicionarHospede(), etc.
* MenuEspacos.cs → Exibir(...), CadastrarNovoEspaco(...)
* MenuReservas.cs → Exibir(...), AdicionarReserva(), etc.
* MenuRelatorios.cs → Exibir(...)
* MenuComprovantes.cs → Exibir(...), com chamadas para Emitir(...)
* MostrarSobreSistemas.cs → Exibir() com informações sobre o sistema

### 📁 Utils/

Funções reutilizáveis e utilitários gerais.

* EntradaHelper.cs → métodos como LerTexto, LerDecimal, LerInteiro
* (opcional futuramente: Validacoes.cs, Formatador.cs, etc.)

### Program.cs

* Ponto de entrada do sistema, criação das listas (hospedes, reservas, etc.), chamada ao MenuPrincipal.Exibir(...)

📌 **Depois de posicionar tudo**:

* Garanta que **cada classe esteja no** namespace **correto**
* Remova duplicatas nos using (ex: using System; repetido)
* No final, usamos o Shift + Alt + F pra deixar tudo bem alinhado no Visual Studio Code

| **Método** | **Deletar?** | **Aproveitamento** |
| --- | --- | --- |
| MenuSuites() | ✅ Sim | Já substituído por MenuEspacos.Exibir(...) |
| CadastrarNovaSuite() | ✅ Sim | Substituído por CadastrarNovoEspaco(...) |
| AdicionarSuite() | ✅ Sim | Lógica absorvida por AdicionarReserva(...) |
| ListarSuites() | ❗Adaptar | Transformar em listagem com status de espaço |
| ExcluirSuite() | ❗Adaptar | Reescrever como ExcluirEspaco(...) com verificação de reserva ativa |

## ****Método:**** ListarEspacosComStatus(...)

### ✅ O que ele faz:

Exibe todos os espaços cadastrados, e para cada um informa:

* Se está **disponível** ou **ocupado**
* Se ocupado, mostra data de check-in, duração e hóspedes

### 📍 Onde colocar:

Dentro do arquivo MenuEspacos.cs, como método **privado**, ao lado de ListarSuites(...) (ou substituindo ele, se preferir).

**O que falta para torná-lo totalmente funcional:**

1. **Receber a lista de hóspedes no método** Exibir(...)**:**

Atualmente o menu não tem acesso à lista real de hóspedes. Basta ajustar assim:

csharp

public static void Exibir(List<Hospede> hospedes)

E passar a lista ao chamar o menu no Program.cs ou outro menu.

1. **Implementar os métodos auxiliares:**
   * AdicionarHospedeCadastrado()
   * ListarHospedes()
   * ExcluirHospede()
   * ListarHospedesComFiltro()

Eles precisam receber a lista hospedes por parâmetro ou acessá-la diretamente caso vire um campo de classe (dependendo da arquitetura).

Se quiser, posso preparar agora um MenuHospedes.cs novinho e funcional como fizemos com MenuEspacos.cs — com entrada validada, listagem organizada, exclusão com confirmação e busca com filtro (por nome, por exemplo).

Quer que eu monte? 😄📋✨

### O que ****já está atendendo bem****:

* **Tipo do espaço é livre:** O campo TipoEspaco da classe EspacoHospedagem aceita qualquer string — excelente.
* **Capacidade, valor, número de quartos, suíte interna:** Esses campos já estão no construtor da classe e são solicitados no momento do cadastro — ótimo.
* **Descrição personalizável:** O campo descricao permite que o funcionário escreva o que quiser sobre o espaço — outro ponto forte!

### O que já está certo:

* using bem organizados no topo — ✔️
* Console.OutputEncoding, Clear(), Thread.Sleep() — ✔️
* Listas principais criadas **antes** da chamada do menu — ✔️
* Nenhum código está solto fora de função — ✔️
* MenuPrincipal() bem estruturado e começando com do { ... } while — ✔️

Resultado: Você vai cadastrar, por exemplo:

TipoEspaco: "Chalé"

NomeEspaco: "Maresia"

idEspaco: 01

e em características:

capacidadeHospedes: (exemplo) até 4

precoDiaria: (exemplo) RS 80,00

E o sistema já sabe que foi o 4º chalé cadastrado, por exemplo.

Testes:

1-Adicionar suíte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Adicionar suite | | | | | | | |  |
|  | Quant  dias | Quanti  pessoas | Cadastrar hóspedes | Digitar | | | | resumo |
| nome | | sobrenome | |
| suite padrão | 2 | 2 | sim | Certo |  | certo |  | Não apareceu sucesso !! |
| 2 | 2 | não | errado | Passou para sobrenome | Errado |  |  |
|  |  |  | nada |  | nada |  |  |
|  |  |  | errado | |  | |  |
| Suíte premium |  |  | sim |  | |  | |  |
|  |  | não |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |

Atualizado em 22jun 20:07h Program.cs

using System;

using System.Text;

using System.IO;

using System.Threading;

using DesafioProjetoHospedagem.Models;

using DesafioProjetoHospedagem.Models;

Console.OutputEncoding = Encoding.UTF8;

// Listas principais

List<Suite> suites = new();

List<Pessoa> hospedes = new();

List<Reserva> reservas = new();

Reserva? novaReserva = null;

bool sistemaLigado = true;

// ===========================

// FUNÇÃO: AdicionarHospede()

// ===========================

void AdicionarHospede()

{

    Console.Clear();

    Console.WriteLine("===CADASTRO DE HÓSPEDE===");

    Console.Write("Informe o primeiro nome: ");

    string? nome = Console.ReadLine()?.Trim();

    Console.Write("Informe o sobrenome: ");

    string? sobrenome = Console.ReadLine()?.Trim();

    if (string.IsNullOrWhiteSpace(nome) || string.IsNullOrWhiteSpace(sobrenome))

    {

        Console.WriteLine("Nome e sobrenome são obrigatórios. Retornando ao MENU.");

        Console.ReadKey();

        return;

    }

    // Monta o nome completo para checar duplicidade

    string nomeCompleto = $"{nome} {sobrenome}".ToUpper();

    if (hospedes.Any(h => h.NomeCompleto.ToUpper() == nomeCompleto))

    {

        Console.WriteLine("\nJá existe um hóspede com esse nome completo.");

        Console.ReadKey();

        return;

    }

    Pessoa novoHospede = new Pessoa(nome, sobrenome);

    hospedes.Add(novoHospede);

    Console.WriteLine($"\nHóspede: {novoHospede.NomeCompleto} cadastrado com sucesso!");

    Console.WriteLine("Pressione qualquer tecla para retornar ao MENU");

    Console.ReadKey();

}

// ===========================

// FUNÇÃO: ListarHospedes()

// ===========================

void ListarHospedes()

{

    Console.Clear();

    Console.WriteLine("===LISTA DE HÓSPEDES===");

    if (hospedes.Count == 0)

    {

        Console.WriteLine("Nenhum hóspede cadastrado.");

        Console.ReadKey();

        return;

    }

    Console.Write("Deseja buscar por nome, sobrenome ou nome completo? (Deixe em branco para listar todos): ");

    string? busca = Console.ReadLine()?.Trim().ToUpper();

    var resultados = string.IsNullOrEmpty(busca)

        ? hospedes

        : hospedes.Where(p =>

            p.Nome.ToUpper().Contains(busca) ||

            p.Sobrenome.ToUpper().Contains(busca) ||

            p.NomeCompleto.ToUpper().Contains(busca)).ToList();

    if (resultados.Count == 0)

    {

        Console.WriteLine("Nenhum hóspede encontrado com esse nome.");

    }

    else

    {

        Console.WriteLine("\nHóspedes encontrados:");

        foreach (var h in resultados)

        {

            Console.WriteLine($"- {h.NomeCompleto}");

        }

    }

    Console.WriteLine("\nPressione qualquer tecla para voltar ao MENU");

    Console.ReadKey();

}

// ===========================

// FUNÇÃO: EmitirComprovanteReserva()

// ===========================

void EmitirComprovanteReserva()

{

    Console.Clear();

    Console.WriteLine("=== EMISSÃO DO COMPROVANTE ===");

    // Aqui, você pode selecionar a reserva ativa (se só houver uma no sistema)

    // ou futuramente permitir escolher por ID. Vamos supor que já temos a reserva ativa:

    if (novaReserva != null)

    {

        // Garante que a pasta exista

        Directory.CreateDirectory("Comprovantes");

        // Caminho completo do arquivo

        string nomeArquivo = Path.Combine("Comprovantes", $"ComprovanteReserva\_{DateTime.Now:yyyyMMdd\_HHmmss}.txt");

        using (StreamWriter writer = new StreamWriter(nomeArquivo))

        {

            writer.WriteLine("========= COMPROVANTE DE RESERVA =========");

            writer.WriteLine($"Data de emissão: {DateTime.Now:dd/MM/yyyy} às {DateTime.Now:HH:mm:ss}");

            writer.WriteLine();

            writer.WriteLine($"Suíte {novaReserva.Suite?.TipoSuite}; Diária R$ {novaReserva.Suite?.ValorDiaria:C}");

            writer.WriteLine($"Dias reservados: {novaReserva.DiasReservados}");

            Console.ForegroundColor = ConsoleColor.Green;

            writer.WriteLine($"Valor estimado até o momento: {novaReserva.CalcularValorDiaria():C}");

            Console.ResetColor();

            Thread.Sleep(1000);

            writer.WriteLine();

            writer.WriteLine("Hóspedes:");

            if (novaReserva.Hospedes != null && novaReserva.Hospedes.Count > 0)

            {

                foreach (var hospede in novaReserva.Hospedes)

                {

                    writer.WriteLine($"- {hospede.NomeCompleto}");

                }

                Console.ForegroundColor = ConsoleColor.Green;

                Console.WriteLine("\nReserva realizada com sucesso !!!");

                Console.ResetColor(); // volta pra cor padrão

                Thread.Sleep(1000);

            }

            else

            {

                writer.WriteLine("- Nenhum hóspede cadastrado.");

            }

            writer.WriteLine("\nObs: No check-out, os valores podem ser diferentes em caso de consumo adicional ou horas extras.");

        }

        Console.WriteLine($"\nComprovante gerado com sucesso: {Path.GetFullPath(nomeArquivo)}");

    }

    else

    {

        Console.WriteLine("Nenhuma reserva ativa encontrada.");

    }

    Console.WriteLine("\nPressione qualquer tecla para retornar ao menu...");

    Console.ReadKey();

    /\*

    //antigo sem a pasta comprovantes

    if (novaReserva != null)

    {

        string nomeArquivo = $"ComprovanteReserva\_{DateTime.Now:yyyyMMdd\_HHmmss}.txt";

        using (StreamWriter writer = new StreamWriter(nomeArquivo))

        {

            writer.WriteLine("========= COMPROVANTE DE RESERVA =========");

            writer.WriteLine($"Data de emissão: {DateTime.Now:dd/MM/yyyy} às {DateTime.Now:HH:mm:ss}");

            writer.WriteLine();

            writer.WriteLine($"Suíte {novaReserva.Suite?.TipoSuite};Diária R$ {novaReserva.Suite?.ValorDiaria:C}");

            writer.WriteLine($"Dias reservados: {novaReserva.DiasReservados}");

            writer.WriteLine($"Valor estimado até o momento, em {DateTime.Now:dd/MM/yyyy} às {DateTime.Now:HH:mm:ss}: {novaReserva.CalcularValorDiaria():C}");

            writer.WriteLine();

            writer.WriteLine("Hóspedes:");

            foreach (var hospede in novaReserva.Hospedes)

            {

                writer.WriteLine($"- {hospede.NomeCompleto}");

            }

            writer.WriteLine("\nObs: No check out, os valores podem ser diferentes, caso haja gasto ou horas extras na saída.");

        }

        Console.WriteLine($"\nComprovante gerado com sucesso: {nomeArquivo}");

    }

    else

    {

        Console.WriteLine("Nenhuma reserva ativa encontrada.");

    }

    Console.WriteLine("\nPressione qualquer tecla para retornar ao menu...");

    Console.ReadKey();

    \*/

}

while (sistemaLigado)

{

    Console.Clear();

    Console.WriteLine("======================================");

    Console.WriteLine("\nSISTEMA DE HOSPEDAGEM - HOTEL FICTÍCIO");

    Console.WriteLine("======================================");

    Console.WriteLine($"Data: {DateTime.Now:dd/MM/yyyy} | Hora: {DateTime.Now:HH:mm:ss}");

    Console.WriteLine("======================================");

    Console.WriteLine("\n----- SUÍTES - INFORMAÇÕES -----\n");

    Console.WriteLine("  Tipo  | Capacidade |   Diária  ");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("Padrão  | 2 pessoas  | R$  80,00 ");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("Premium | 3 pessoas  | R$ 100,00 ");

    Console.WriteLine("---------------------------------\n");

    Console.WriteLine("- Estadia a partir de 10 dias: 10% de desconto");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("- Ultrapassado o tempo da diária:");

    Console.WriteLine("  Taxa de acréscimo por hora - R$ 5,00");

    //Console.Clear();

    Console.WriteLine("=======================================\n");

    Console.WriteLine("  MENU   | Adicionar | Listar | Excluir|");

    Console.WriteLine("----------------------------------------");

    Console.WriteLine("Suite    |     1     |   2    |   3    |");

    Console.WriteLine("----------------------------------------");

    Console.WriteLine("Hóspedes |     4     |   5    |   6    |");

    Console.WriteLine("=======================================");

    Console.WriteLine("Emitir comprovante   |   7    |");

    Console.WriteLine("-------------------------------");

    Console.WriteLine("Encerrar o sistema   |   8    |");

    Console.WriteLine("==============================\n");

    Console.Write("Escolha uma opção: ");

    string? opcao = Console.ReadLine();

    // Controle de opções

    switch (opcao)

    {

        case "1":

            AdicionarSuite();

            break;

        /\*case "2":

            ListarSuites();

            break;\*/

        /\*case "3":

            ExcluirSuite();

            break;\*/

        case "4":

            AdicionarHospede();

            break;

        /\*case "5":

        ListarHospedes();

        break;\*/

        /\*case "6":

            ExcluirHospede();

            break;\*/

        case "7":

            EmitirComprovanteReserva();

            break;

        case "8":

            sistemaLigado = false;

            break;

        default:

            Console.WriteLine("Opção inválida. Pressione qualquer tecla para voltar ao menu.");

            Console.ReadKey();

            break;

    }

} //final while

void AdicionarSuite()

{

    Console.Clear();

    Console.WriteLine($"Data: {DateTime.Now:dd/MM/yyyy} | Hora: {DateTime.Now:HH:mm:ss}");

    Console.WriteLine("==================================");

    Console.WriteLine("  Tipo  | Capacidade |   Diária  ");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("Padrão  | 2 pessoas  | R$  80,00 ");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("Premium | 3 pessoas  | R$ 100,00 ");

    Console.WriteLine("---------------------------------\n");

    Console.WriteLine("- Estadia a partir de 10 dias: 10% de desconto");

    Console.WriteLine("---------------------------------");

    Console.WriteLine("- Ultrapassado o tempo da diária:");

    Console.WriteLine("  Taxa de acréscimo por hora - R$ 5,00");

    Console.WriteLine("--------------------------------------");

    Console.WriteLine("\n=== RESERVA DE SUÍTE ===\n");

    int dias;

    do

    {

        Console.Write("Para quantos dias deseja reservar? ");

        string? entrada = Console.ReadLine();

        if (!int.TryParse(entrada, out dias) || dias <= 0)

        {

            Console.WriteLine("Ops. Digite o número de dias para reservar.\n");

        }

    } while (dias <= 0);

    int pessoas;

    do

    {

        Console.Write("Para quantas pessoas? ");

        string? entrada = Console.ReadLine();

        if (!int.TryParse(entrada, out pessoas) || pessoas <= 0)

        {

            Console.WriteLine("Ops. Digite o número de pessoas.\n");

        }

    } while (pessoas <= 0);

    if (pessoas > 3)

    {

        Console.WriteLine("\nA suíte Padrão comporta até 2 pessoas e a suíte Premium até 3 pessoas.");

        Console.WriteLine("Deseja fazer essa reserva agora e, concluindo, criar outra para os excedentes? (S/N)\n");

        string? continuar;

        do

        {

            Console.Write("Sua resposta: ");

            continuar = Console.ReadLine()?.Trim().ToUpper();

            if (continuar != "S" && continuar != "N")

            {

                Console.WriteLine("Ops. Digite 'S' para sim ou 'N' para não.");

            }

        } while (continuar != "S" && continuar != "N");

        if (continuar == "N")

        {

            Console.WriteLine("\nTudo bem ! Pressione qualquer tecla para retornar ao menu.");

            Console.ReadKey();

            return;

        }

        do

        {

            Console.Write("\nQuantas pessoas deseja incluir nesta primeira reserva? (máx. 3): ");

            string? entrada = Console.ReadLine();

            if (!int.TryParse(entrada, out pessoas) || pessoas <= 0 || pessoas > 3)

            {

                Console.WriteLine("Ops. Digite um valor entre 1 e 3.");

            }

            else

            {

                break; // entrada válida, sai do laço

            }

        } while (true);

        if (pessoas <= 2)

        {

            Console.WriteLine("1 - Suíte Padrão (até 2 pessoas, R$80,00/diária)");

            Console.WriteLine("------------------------------------------");

            Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

        }

        else if (pessoas == 3)

        {

            Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

        }

    }

    else

    {

        // Para até 3 pessoas: exibe normalmente as opções

        Console.WriteLine("\nTipos disponíveis:");

        if (pessoas <= 2)

        {

            Console.WriteLine("1 - Suíte Padrão (até 2 pessoas, R$80,00/diária)");

            Console.WriteLine("------------------------------------------");

            Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

        }

        else if (pessoas == 3)

        {

            Console.WriteLine("2 - Suíte Premium (até 3 pessoas, R$100,00/diária)");

        }

    }

    Console.Write("Digite o número da suíte desejada: ");

    string? tipoEscolhido = Console.ReadLine();

    Suite suite;

    if (tipoEscolhido == "1" && pessoas <= 2)

    {

        suite = new Suite("Padrão", 2, 80);

    }

    else if (tipoEscolhido == "2")

    {

        suite = new Suite("Premium", 3, 100);

    }

    else

    {

        Console.WriteLine("O número de pessoas ultrapassa a capacidade da suíte.");

        Console.ReadKey();

        return;

    }

    suite.Id = suites.Count + 1;

    suites.Add(suite);

    novaReserva = new Reserva(hospedes, suite, dias);

    novaReserva.CadastrarSuite(suite);

    // Perguntar se deseja cadastrar os hóspedes agora

    string? resposta;

    do

    {

        Console.Write("\nDeseja cadastrar os nomes dos hóspedes agora? (S/N): ");

        resposta = Console.ReadLine()?.Trim().ToUpper();

        if (resposta != "S" && resposta != "N")

        {

            Console.WriteLine("Ops. Digite 'S' para sim ou 'N' para não.");

        }

    } while (resposta != "S" && resposta != "N");

    if (resposta == "S")

    {

        List<Pessoa> listaTemporaria = new();

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

        {

            Console.Write($"\nDigite o nome do hóspede {i}: ");

            string? nome = Console.ReadLine()?.Trim();

            Console.Write("Digite o sobrenome do hóspede: ");

            string? sobrenome = Console.ReadLine()?.Trim();

            if (string.IsNullOrWhiteSpace(nome) || string.IsNullOrWhiteSpace(sobrenome))

            {

                Console.WriteLine("Nome e sobrenome são obrigatórios. Tente novamente.");

                i--;

                continue;

            }

            listaTemporaria.Add(new Pessoa(nome, sobrenome));

        }

        try

        {

            novaReserva.CadastrarHospedes(listaTemporaria);

            hospedes.AddRange(listaTemporaria);

            Console.ForegroundColor = ConsoleColor.Green;

            Console.WriteLine("\nHóspedes cadastrados com sucesso!");

            Console.ResetColor();

            Thread.Sleep(1000);

        }

        catch (Exception ex)

        {

            Console.WriteLine($"\nErro: {ex.Message}");

        }

    }

    else

    {

        Console.WriteLine("Hóspedes não cadastrados. Você poderá adicioná-los depois.");

    }

    novaReserva = new Reserva(hospedes, suite, dias);

    reservas.Add(novaReserva);

    suites.Add(suite);

    Console.Clear();

    Console.WriteLine("==================================");

    Console.WriteLine($"Data: {DateTime.Now:dd/MM/yyyy} | Hora: {DateTime.Now:HH:mm:ss}");

    Console.WriteLine("==================================");

    Console.WriteLine("\n=== RESUMO DA RESERVA ===");

    Console.WriteLine($"Suíte {suite.TipoSuite}. Cadastro {suite.Id}");

    Console.WriteLine($"Valor da diária: {suite.ValorDiaria.ToString("C")}");

    Console.WriteLine($"Dias reservados: {dias}");

    Console.WriteLine($"Quantidade de hóspedes: {pessoas}");

    Console.WriteLine("----------------------------------------");

    decimal valorSemDesconto = novaReserva.DiasReservados \* novaReserva.Suite.ValorDiaria;

    decimal valorEstimado = novaReserva.CalcularValorDiaria();

    decimal descontoAplicado = valorSemDesconto - valorEstimado;

    Console.WriteLine($"Subtotal (sem desconto): {valorSemDesconto.ToString("C")}");

    Console.WriteLine($"Desconto aplicado: {descontoAplicado.ToString("C")}");

    Console.ForegroundColor = ConsoleColor.Green;

    Console.WriteLine($"Total atual com desconto: {valorEstimado.ToString("C")}");

    Console.ResetColor();

    Console.WriteLine("----------------------------------------");

    if (novaReserva.Hospedes == null || !novaReserva.Hospedes.Any())

    {

        Console.WriteLine("Nenhum hóspede foi cadastrado nesta reserva.");

        Console.WriteLine("Acesse a opção 4 do MENU para cadastrar depois.");

        Console.ForegroundColor = ConsoleColor.Green;

        Console.WriteLine("\nReserva realizada com sucesso !!!");

        Console.ResetColor(); // volta pra cor padrão

        Thread.Sleep(1000);

    }

    else

    {

        Console.WriteLine($"\nQuantidade de hóspedes: {novaReserva.Hospedes.Count}");

        Console.WriteLine("Lista de hóspedes:");

        foreach (var h in novaReserva.Hospedes)

        {

            Console.WriteLine($"- {h.NomeCompleto}");

        }

        Console.ForegroundColor = ConsoleColor.Green;

        Console.WriteLine("\nReserva realizada com sucesso !!!");

        Console.ResetColor(); // volta pra cor padrão

        Thread.Sleep(1000); // precisa do using System.Threading;

    }

    Console.WriteLine("\nPressione qualquer tecla para retornar ao menu principal...");

    Console.ReadKey();

}

**Etapas principais do projeto (com base nos TODOs e nas melhorias sugeridas)**

**1. Completar os métodos da classe Reserva**

Esses métodos têm marcações TODO, então são prioridade para seguir as instruções do professor.

**Arquivo:** Reserva.cs (dentro da pasta Models) **O que falta:**

* [x] CadastrarHospedes — verificar se o número de hóspedes ultrapassa a capacidade
* [x] ObterQuantidadeHospedes — retornar a quantidade de hóspedes
* [x] CalcularValorDiaria — calcular valor com desconto (se for o caso)

🛠 **Status**: Já completamos esses! ✔️

## Objetivo desse passo:

* Garantir que **todo hóspede tenha nome e sobrenome** ao ser cadastrado
* Permitir **buscar hóspedes por nome, sobrenome ou nome completo**
* Permitir **listar hóspedes com visualização organizada e personalizável**

**2. Criar um menu interativo no Program.cs**

**Arquivo:** Program.cs **Objetivo:** Criar um sistema de menu com opções numeradas que oriente o usuário e permita:

* Adicionar/excluir suíte
* Adicionar/excluir hóspede
* Reservar suíte
* Calcular valores
* Exibir dados

Vamos fazer isso dentro de um while (true) com switch/case. Também vamos capturar exceções com try/catch para que erros não derrubem o sistema.

**3. Criar listas globais para gerenciar dados durante o uso**

**Arquivo:** Program.cs Vamos precisar guardar em memória (por enquanto) as:

* Suítes adicionadas
* Hóspedes cadastrados
* Reservas realizadas

Isso será feito com List<Suite>, List<Pessoa> e List<Reserva>.

**4. Criar funções auxiliares para cada opção do menu**

Exemplos:

* AdicionarSuite()
* AdicionarHospede()
* ExcluirSuite()
* ExcluirHospede()

**Arquivo:** Program.cs, fora do método Main, mas dentro da classe Program (ou como métodos estáticos se estivermos num arquivo procedural).

🔄 Essas funções organizam o código, facilitam testes e reaproveitamento, e deixam o Main() limpo.

**5. Aplicar regras extras:**

* [ ] Calcular valor adicional por hora excedente (ex: +R$25/hora)
* [ ] Aplicar confirmação de reserva e cadastro de hóspedes passo a passo
* [ ] Impedir cadastro inválido com mensagens amigáveis e retorno ao menu

Essas partes não estão sinalizadas com TODO, mas fazem parte da sua visão mais completa do sistema. Vamos inseri-las como melhorias nas funções auxiliares.

**✅ Próximo passo ideal:**

**Vamos começar pelo menu principal com opções básicas funcionando (etapas 2 e 3 acima).** Assim, você verá o sistema ganhando vida! Com base nisso, o plano é:

1. Criar listas para gerenciar dados no Program.cs
2. Montar o menu principal com Console.WriteLine + switch
3. Criar a função AdicionarSuite() completa, com validação e tratamento de erro
4. Depois seguimos para AdicionarHospede(), reservas, exclusões, e assim por diante

---------------------------------------------------------------------------------------------------------------------------------------------------

Pasta raiz: trilha-net-explorando-desafio, que tem a pasta Models e os arquivos gitgnore, DesafioProjetoHospedagem.csproj, Program.cs, readme.md e trilha-net-exporando-desafio.sln /Dentro da pasta Models tem Pessoa.cs, Reserva.cs e Suite.cs

Pessoa.cs

Abaixo, objetivo de aprimoramento

Descrição detalhada, com todas as lógicas e exceções que um sistema de hospedagem real precisaria!

Mini sistema comercial com fluxo de menus, tratamento de erros e interação amigável.

Sistema de HOSPEDAGEM

Menu principal

Hotel fictício - Sistema de hospedagem

Informações

- Suíte padrão: Capacidade para 2 pessoas, diária R$ 80,00

- Suíte premium: capacidade para 3 pessoas, diária R$ 100,00

- Reserva >= 10 dias, 10% desconto

- horarioDiaria ultrapassada, + R$ 25,00 por hora

-------------------------------------------------------------------------------------------------------

Menu principal

1. Adicionar Suíte
2. Adicionar hóspede
3. Excluir suíte
4. Excluir hóspede
5. Listar suítes
6. Listar hóspedes
7. Retornar ao Menu
8. Encerrar

Pressionar qualquer tecla para retornar

-------------------------------------------------------------------------------------------------------

1. ADICIONAR SUITE

Menu

Informações

- Suíte padrão: Capacidade 2 pessoas, diária R$ 80,00

- Suíte premium: capacidade 3 pessoas, diária R$ 100,00

- Mais de 10 dias, 10% desconto

- Horário da diária ultrapassada, será cobrado adicional de R$ 25,00 por hora

-------------------------------------------------------------------------------------------------------

Saída: Reserva para quantos dias? Ler

Saída: Qual o número de pessoas? Ler

Se numeroPessoas >0 && <=2

Saída: Digite 1 para reservar suíte padrão ou 2 para reservar suíte premium

Entrada: se 1, reservar suitePadrao{numero suíte Padrao}, saída com detalhamento da reserva: suíte reservada, valor diária, valor sem desconto, valor do desconto, valor total, quantidade de pessoas. E perguntar se quer cadastrar os hospedes agora e cadastrar apenas a quantidade de hópedes aceitáveis, senão, cadastrar o numero aceitável e informar para reservar outra suíte para os hóspedes excedentes

se 2 reservar {numero suíte Premium}, saída com detalhamento da reserva: suíte reservada, valor diária, valor sem desconto, valor do desconto, valor total, quantidade de pessoas. E perguntar se quer cadastrar os hospedes agora e cadastrar apenas a quantidade de hópedes aceitáveis, senão, cadastrar o numero aceitável e informar para reservar outra suíte para os hóspedes excedentes

se for outro valor, limpar tela, voltar e saída: Informe uma opção válida

Se entrada for válida: faça a ação

Se não, limpar tela e saída: Digite 1 para reservar suíte padrão, 2 para reservar suíte premium, 7 para retornar ao menu principal

Se numeroPessoas ==3

Saída: A suíte com capacidade para 3 pessoas é a suíte premium. Deseja confirmar a reserva?

Entrada: se Y reservar

se N , saída: Digite 1 para escolher novamente (vai para Menu adicionarSuite) ou 7 para retornar ao menu principal

Se reserva na suíte padrão ou na suíte premium for feita, saída: Quer adicionar nome dos hóspedes agora?

Se sim, adicionar nome hospede e retornar todas as informações referentes a hospedagem com nome dos hospedes

Se não, retornar todas as informações referentes a hospedagem com quantidade de hóspedes

Para opções:

Adicionar Suíte

Adicionar hóspede

Excluir suíte

Excluir hóspede

Listar suítes

Listar hóspedes

Retornar ao Menu

Sempre que a entrada estiver errada, retornar uma exceção, enviar mensagem explicando invalido e avisar que está retornando ao menu referente a opção em questão, limpar tela e voltar para o inicio da opção em questão