trilha-net-explorando-desafio

│

|── 📁Comprovantes

│ └── .txt

│

|── 📁 Controladoras

| |

│ ├── ClienteControlador.cs – void: CadastrarCliente, ExcluirCadastroCliente, ExcluirCadastroCliente,

| | ListarCadastroClientes

| |

│ ├── EspacoControlador.cs – void: CadastrarEspaco, EditarCadastroEspaco, ExibirResumoEspacoCadastrado,

| | ExcluirCadastroEspaco, ListarEspacos

| |

│ └── ReservaControladora:class ReservaControladora: void CriarReserva

|

|

|── 📁 Menus

│ ├── MenuPrincipal.cs – void: Exibir (espaços, clientes, reservas, resrvaSelecionada)

| |

│ ├── EspacoHospedagem.cs – EspacoHospedagem (tipoEspaco, nomeFantasia, capacidade, valorDiaria,

| | quantidadeQuartos, possuiSuite, quantidadeSuite, decricao)

| |

│ ├── MenuClientes.cs – Resera?ExibirMenuClientes (clientes, reservas, espaços, reservaSelecionada)

| | void: CadastrarCliente, EditarCadastroCliente, ExcluirCadastroCliente, ListarCadastroClientes,

| | ListarClientesHospedados, ListarClientesNaoHospedados,

| |

│ ├── MenuComprovantes.cs (submenu com emissão opcional com: Emitir comprovante de reserva)

| | excluir, voltar)

| |

│ ├── MenuEspacos.cs – Reserva? ExibirMenuEspacos (espaços, reservas, clientes, reservaSelecionada)

| |

│ ├── MenuRelatorios.cs – Exibir (espaços, reservas, clientes)

| |

│ ├── MenuReservas.cs – Exibir(reservas, clientes, espaços), void: CadastrarReserva, ExibirSubMenuClientes,

| | VincularClientesAReserva, ListarCadastroReservas, CancelarReserva, EncerrarReserva,

| | ExcluirCadastroReserva, VincularClientes, GerarComprovante,

| |

│ └── MenuSobreSistema.cs - Informações

│

├── 📁 Models

│ ├── Clientes.cs – DataNascimento, Nome, Sobrenome, Email, DataCadastro, Cpf

| |

| └── Reserva.cs – (Ativa, Clientes, Espaco, DiasReservados, DataCheckIn),

| |

| |\_\_ Construtor padrão - CalcularDiaria, FinalizarReserva, ObterQuantidadeClientes,

| |

| |\_\_ Construtor com parâmetros - Reserva, VincularClientes, VincularEspaco

│

├── 📁 Servicos

│ ├── ServicoClientes.cs – class ServicoCliente: CaminhoArquivo, void - SalvarClientes

|

├── 📁 Utils

| |\_\_ CentralDeNavegacao.cs

| | |\_\_ class CentralDeNavegacao: void - DigitouZeroRetornaMenu, EncerrarSistema, ReiniciarEntrada

| | |

| | |\_\_ class MiniMenuHelper: void - RodarMiniMenu

| |

| |\_\_CentralEntrada

| | |\_\_ class CentralEntrada: void – LerTextoObrigatorio, LerValorMonetario, ValorMonetarioComEscape,

| | | LerDataObrigatoriaComEscape, LerTextoComCancelamento, ExibirPerguntaOpcao,

| | |

| | ├── class CentralEntradaInteira: void - LerComCancelamento, LerDataComCancelamento, UmAteLimite,

| | |

| | ├── class CentralEntradaControladora: void - LerEmailComValidcao, LerDecimalcomCancelamento,

| | LerInteiroComCancelamento, LerInteiroMinMax, LerNumeroComoString, LerNumeroInteiro, SimNao,

| |

| |\_\_CentralSaida

| | |--- enum TipoMensagem: Erro, sucesso, instrucao, Informacao, Alerta

| | |--- class CentralSaida.cs: void – AguadarTeclaParaRetornar, Borracha, CabecalhoMenus,

| | CabecalhoMensagem, CentralizarTexto, Exibir, ExibirDespedidaSimples, ExibirDestaque, ErroIntervalo,

| | LimparTela, LimparTelaMesmoMenu, LimparTelaRetornoMenus, LinhaSeparadora,

| | MensagemPadraoSucesso, MensagemCancelamento, Submenu, Subtitulo,

| |

│ ├── EntradaHelper.cs: bool ConfirmarCancelamento, void - ExibirMensagemEntrada,

| | ExibirMensagemEntradaNumerada, ExibirMensagemRetorno,

| | |

│ ├── InterfaceHelper.cs: AnimacaoGiratoria, CentralizarTexto, AzulBranco, CentralSaidaMensagens,

| | EscreverComAnimacao,

| | EditarTexto, ExibirTituloComData

| |

│ ├── MiniMenu.cs -

| |

│ ├── MiniMenuHelper.cs – RodarMiniMenu

│

|── DesafioProjetoHospedagem.csproj

|

|── Program.cs

│

|── README.md

│

|── trilha-net-explorando-desafio.sl

|  |  |
| --- | --- |
| EspacoControlador.cs | linhas 116 e 436 |
| MenuRelatorios.cs | linha 48 |
| MostrarSobre.cs | linha 37 |
| MenuComprovantes.cs | linha 47 |
| MenuClientes.cs | linha 74 |

1. a tela ficou igual um arco iris, todo colorido, precisamos fazer com que apenas a pergunta abaixo das listas de cada menu fique em azul.
2. 2- o valor da diaria,a ceitou letras, precisa aceitar valors como reais e colcocar R$ antes dos valroes. ex valor R$ 1.500,00 .
3. 3- quantidade de quarros não recebeu a entrada do usuario na mesma linha.
4. 4- ponto e virgula está duplicado.
5. 5- precisamos de apenas uma mensagem de sucesso e a tela retornou duas mensagens. deixar apenas: Cadastro realizado com sucesso !!! que pode ser na cor verde

## Mapeamento final das cores no sistema

| **Elemento** | **Cor correta** | **Exemplo de uso** |
| --- | --- | --- |
| **Perguntas ao usuário** | ConsoleColor.White | Console.ForegroundColor = ConsoleColor.White; Console.Write("Nome do espaço: "); |
| **Pergunta abaixo da lista** | ConsoleColor.Blue | Console.Write("Escolha sua opção: "); → **somente** no rodapé após lista/menu |
| **Mensagem de sucesso** | ConsoleColor.Green | CentralSaida.Exibir(TipoMensagem.Sucesso, "Cadastro realizado com sucesso !!!"); |
| **Mensagem de erro** | ConsoleColor.Red | CentralSaida.Exibir(TipoMensagem.Erro, "Entrada inválida. Digite novamente."); |
| **Instruções e dicas** | ConsoleColor.Yellow | CentralSaida.Exibir(TipoMensagem.Instrucao, "Use o minimenu para navegar."); |
| **Informações neutras** | ConsoleColor.White | Exibir textos como título, ID gerado, etc. |

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

Metodo LerNumeroInteiro (CentralEntrada.cs – CentaralEntradaControladora

## O que esse método faz perfeitamente

* Exibe a pergunta com destaque azul 💙
* Lê somente valores numéricos
* Rejeita letras, símbolos ou campos em branco
* Exibe erro em vermelho 🟥
* Dá um tempo visual com Thread.Sleep(1200) para o usuário absorver o erro
* Repete até a entrada ser válida

## 🧠 Onde aplicar esse método com segurança

Esse método é ideal para perguntas como:

* Capacidade máxima para quantas pessoas
* Quantidade de quartos
* Quantidade de suítes
* Quantidade de hóspedes
* Número de dias da reserva
* Número de identificação (se for inteiro)

💡E você pode passar o texto da pergunta como quiser:

csharp

int capacidade = CentralEntradaControladora.LerNumeroInteiro("Capacidade máxima para quantas pessoas");

int quantidadeQuartos = CentralEntradaControladora.LerNumeroInteiro("Quantidade de quartos");

## 🧩 Dica extra — validação com faixa opcional

Se mais pra frente quiser reforçar que o número precisa estar dentro de uma faixa (ex: entre 1 e 10), você pode evoluir esse método com dois parâmetros extras:

csharp

public static int LerNumeroComIntervalo(string mensagem, int min, int max)

{

while (true)

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write($"\n{mensagem}: ");

Console.ResetColor();

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

if (int.TryParse(entrada, out int numero))

{

if (numero >= min && numero <= max)

return numero;

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"\nDigite um número entre {min} e {max}.");

Console.ResetColor();

Thread.Sleep(1200);

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("\nValor inválido! Digite um número inteiro válido.");

Console.ResetColor();

Thread.Sleep(1200);

}

}

}

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

use LerNumeroInteiro(...) para inteiros e LerValorDecimal(...) para valores monetários:

csharp

int capacidade = CentralEntradaInteira.LerNumeroInteiro("Capacidade máxima para quantas pessoas");

decimal valorDiaria = CentralEntradaControladora.LerValorDecimal("Valor da diária");

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

## Resumo bonitinho:

| **Campo** | **Tipo** | **O que permite** |
| --- | --- | --- |
| DataCheckIn | DateTime | Sempre precisa estar preenchido ✅ |
| DataCheckOut | DateTime? | Pode estar vazio (null) ❌ |

## Quando métodos com cancelamento são úteis

Eles são bons para cenários como:

* **Filtros de busca**: o usuário pode não querer preencher todos os campos
* **Cadastro opcional ou com etapas saltáveis**
* **Formulários longos com botão de “Cancelar” ou “Voltar”**
* **Assistentes interativos ou fluxos conversacionais**

Mas no seu sistema — que é **voltado para cadastros formais**, com validações rígidas e campos obrigatórios — **não faz sentido permitir que o usuário aperte Enter 3 vezes e siga sem preencher**.

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

Escolha sua opção : ler inteiro minmax:

Temos as seguintes situações:

Menu principal: opções de 0 a 6. se digitar errado, enquanto errar, mensagem: digite novamente (até digitar certo). Após 3 tentativas erradas perguntar se quer continuar na tela ( Se S, continua, se N, pedir para escolher uma opção no minimenu. Digitando certo, segue o fluxo.

Submenus: cada um tem uma quantidade de itens em sua lista, então para todos tem que ler de 1(menus não aceitam 0 e sempre iniciam em 1)

Minimenu: tem as opções 0, encerrar sistema, 9- 8- 7-

Como padrão em todo sistema, se digitar errado, aparece mensagem digite novamente (até acertar). Se digitar

Erros encontrados:

1.Cadastro de espaços:

Não aparece minimenu

Após cadastro aparecem duas linhas de sucesso. deletar linha: operação concluída com sucesso

Revisados: reserva, menuespacos, menureservas, menucomprovantes, menucomprovantereservas deletado, menuprincipal, menurelatorios, mostrar sistema, menu clientes

Obs: Duvidas:

1. SimNao não é mais uado mas tem o caod e se pressionar enter 3 vezes

## Regras Funcionais (oficiais do sistema)

| **Regra** | **Comportamento esperado** |
| --- | --- |
| **1. cadastros sempre iniciam em 1** | Nenhum item pode ter índice 0 ou negativo |
| **2. Limite superior baseado em dados reais** | O valor máximo permitido depende da quantidade já cadastrada (espacos.Count, suites.Count, etc). |
| **Respeitar items de cada menu** | Cada submenu pode ter um menu diferente do outro, com itens e numerações diferentes |
| **4. Nunca acessar índices além do que está cadastrado** | Caso ocorra, exibir mensagem: **"Ops. Digite algo válido !!!" digite novamente.** |
| **5. Três [ENTER]s seguidos sem digitar nada** | Exibir mensagem: **"Ops, você pressionou ENTER 3 vezes. Digite novamente ou use as opções do rodapé** |
| O que faz mais sentido no seu caso  * Campos **obrigatórios**, com repetição até serem preenchidos corretamente * Validações específicas para tipo, faixa e formato * Mensagens de erro padronizadas (ex: "Digite um número inteiro válido") * Sem tolerância para campos nulos em cadastros essenciais  **Mensagens e Perguntas**  * 📣 Perguntas devem aparecer com cor branca e estilo limpo * ❗ Respeitar espaçamento visual acima da pergunta * 💬 Pergunta e resposta devem estar na **mesma linha**  **4. Tratamento das Respostas**  * ✅ Sempre exigir entrada válida   + Se for número, valida faixa   + Se for texto, exige não vazio * 🚫 Se entrada inválida:   + Exibir mensagem: "Ops. Digite algo válido !!!"   + Repetir pergunta * ⏳ Se o usuário pressionar Enter 3 vezes seguidas ou digitar dados inválidos:   + Exibir: "Ops, você pressionou ENTER 3 vezes. Digite novamente ou use as opções do rodapé."   + Perguntar: "Deseja retornar ao menu principal? (S/N)"     - Se S: navega para menu principal     - Se N: continua e reinicia tentativas |  |

O que precisamos:

1. mensagens e perguntas para controle de ações, no menu principal e submenus com validações para não aceitar nulo ou invalidos e com opções que dê escolhas para usuário, e texto na cor branca.
2. Acima do minimenu a pergunta: Escolha sua opção: com texto na cor azul e tem função de navegação e ações de Baseado no que já temos, que métodos podemos adaptar para que tudo funcione conforme as regras e com validação para não afetar osistema?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Situação** | **Comportamento necessário** | **Método mais adequado** | | --- | --- | --- | | ✅ Seleção de opção em menus (1 a N) | Entrada **obrigatória**, valida faixa, não pode ser nula ou inválida | LerInteiroMinMax(...) com validação de intervalo | | ✅ Cadastro de itens (ID sempre começa em 1) | Geração interna (sem entrada), mas sempre positivo e incremental | int id = lista.Count + 1; + validação id > 0 | | ❌ Digitar 3 vezes Enter ou inválido seguidas | Exibir aviso e perguntar se deseja voltar ao menu principal | Lógica integrada ao loop de entrada (tentativas >= 3) | | 🎯 Entrada numérica opcional, mas ainda precisa ser válida | Permitir null mas validar o valor se digitado | LerInteiroComCancelamento(...) + verificação manual | | 🎯 Entrada numérica simples, sem faixa de valores | Aceitar apenas inteiros, obrigatoriamente válidos | LerInteiroComTentativas(...) (personalizado) | | 📝 Entrada de texto obrigatório | Não pode ser vazio, repetir até digitar algo válido | LerTextoComCancelamento(...) com validação de texto | | 💰 Valores decimais (como diárias, preço etc.) | Deve aceitar apenas número decimal válido, obrigatório | LerDecimalComCancelamento(...) adaptado com tentativas | | 📋 Escolha dentro de uma lista já cadastrada | Não acessar índices fora do lista.Count, repetir se inválido | LerIndiceValido(titulo, lista.Count) (personalizado) | | 🔄 Navegação por rodapé (MiniMenu) | Entradas como S, N, 0, X, com validação específica | LerComandoMiniMenu() com tratamento de letras/ações |  🧠 Observações importantes:  * Todos os métodos devem incorporar:   + **Validação do tipo de dado** (inteiro, decimal, texto)   + **Contador de tentativas inválidas**   + **Mensagem após 3 erros:** "Ops, você pressionou ENTER 3 vezes. Digite novamente ou use as opções do rodapé."   + **Pergunta:** "Deseja retornar ao menu principal? (S/N)" |

### Regras de listagem

| **Ação** | **Exibe quem?** |
| --- | --- |
| **Listar clientes** | Todos que estão **cadastrados no sistema**, independente de estarem hospedados ou não. Inclui quem **já foi hóspede** e quem **ainda nem hospedou**. |
| **Listar hóspedes** | Somente aqueles clientes com **reserva ativa no momento** — ou seja, **estão hospedados agora**. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Item** | **Status** | | --- | --- | | ExibirMenuEspacos(...) | Corrigido, com MiniMenu, switch, return reservaSelecionada ✅ | | Opção 1 - Cadastrar | EspacoControlador.CadastrarEspaco → MenuEspacos.CadastrarEspaco ✅ | | Opção 2 - Listar | EspacoControlador.ListarEspacos → MenuEspacos.ListarCadastroEspacos ✅ | | Opção 3 - Editar | EspacoControlador.EditarEspacos → MenuEspacos.EditarCadastroEspaco ✅ | | Opção 4 - Excluir | EspacoControlador.ExcluirEspacos → MenuEspacos.ExcluirCadastroEspaco ✅ | | Opção 0 - Voltar | Navegação funcional via CentralDeNavegacao.VoltarAoMenu(...) ✅ | | Atalhos 6,7,8,9 | Suporte completo via MiniMenu.Executar(...) ✅ | | Cabeçalho visual e pergunta azul | Estão no lugar certinho 💙 | |

Para navegação entre menus:

int? entrada = CentralEntradaControladora.LerInteiroComCancelamento("");

if (entrada != null &&

MiniMenu.Executar(entrada.Value, "Espaços",

() => ExibirMenuEspacos(...), espacos, hospedes, reservas, reservaSelecionada))

{

return reservaSelecionada; // retorna se o atalho foi reconhecido e executado

}

// Se não for atalho, segue para o menu vertical

//aqui vem o switch{}

| **Opção** | **Ação** | **Mensagem associada** |
| --- | --- | --- |
| 9 | Voltar ao menu anterior | “Retornando para Menu Reservas...” |
| 8 | Ir ao menu principal | “Retornando para Menu Principal...” |
| 7 | Limpar tela e reiniciar entrada | “Campos limpos. Vamos tentar de novo!” |
| 0 | Encerrar o sistema | “Encerrando o sistema... Obrigado por utilizar...” |
| ENTER × 3 | Interrupção por inércia | “Ops! Você pressionou ENTER 3 vezes seguidas...” |

Validação de faixa (1 a limite) Restrição de entrada ✅ Pertence à CentralEntradaControlada - ok

Digitação de 0 pode ser escrito pelo usuario para inserção de dados e leitura do sistema, não considerar, a menos que seja digitado como uma escolha no minimenu(0 encerra o sistema, 9 volta ao menu anterior e 8 volta para menuPrincipal e 7 limpa os dados digitados na tela atual para nova digitação) - Ação que leva ao miniMenu para que usuario decida para onde quer navegar- 🔄 Pertence à lógica da navegação, mas é acionada aqui

ENTER x3 com confirmação- Proteção contra inação do usuário - leva ao minimenu

Mensagens padronizadas - Coesão visual e UX consistente ✅ Já embutidas via **CentralSaidaMensagens**

Retorno null como sinal de saída Canal de retorno à lógica externa ✅ Mapeia diretamente para "cancelamento via entrada" => não entendi, pode me explicar com exemplo?

📌 Ou seja: o método é como um porteiro da interação navegável. Ele não decide para onde ir, mas te devolve um “sinal” (null, 0, etc.) que o sistema interpreta. => não entendi, pode me explicar com exemplo?

├── 📁 Regras

│ ├── GerenciadorEspacos.cs (Lógica de CRUD de espaços)

│ ├── GerenciadorReservas.cs (Lógica de CRUD de reservas)

│ ├── GerenciadorClientes.cs (Lógica de CRUD de clientes)

│ ├── GerenciadorRelatorios.cs (Geração de relatórios)

│ └── GeradorComprovante.cs ((opcional) Geração de comprovantes)

│

├── 📁 BancoDadosSimulado

│ ├── ListasGlobais.cs (Armazena listas de clientes, reservas, espaços, etc.)

│

├── 📁 Utilitarios

│ ├── DataHelper.cs (Operações com datas, períodos, formatações)

│ ├── TextoHelper.cs (Quebra de texto, acentuação, capitalização)

│

├── Program.cs (Método Main: ponto de entrada da aplicação))

├── README.md ((Documentação do projeto)

└──- gitignore (Regras de exclusão para o versionamento Git)

SISTEMA DE HOSPEDAGEM – HOTEL FICTÍCIO

│

├── MENU – PRINCIPAL

│ ├── 1 - ESPAÇOS

│ │ ├── 1 - Cadastrar Espaço

│ │ ├── 2 - Listar Espaços

│ │ ├── 3 - Editar Espaço

│ │ ├── 4 - Excluir Espaço

│ │ └── 0 - Voltar

│ │

│ ├── 2 – MENU RESERVAS

│ │ ├── 1 - Cadastrar Reserva

│ │ ├── 2 - Listar Reservas

│ │ ├── 3 - Editar Reserva

│ │ ├── 4 - Excluir Reserva

│ │ └── 0 - Voltar

│ │

│ ├── 3 – MENU CLIENTES

│ │ ├── 1 - Cadastrar Cliente

│ │ ├── 2 - Listar Clientes

│ │ ├── 3 - Editar Cliente

│ │ ├── 4 - Excluir Cliente

│ │ └── 0 - Voltar

│ │

│ ├── 4 - MENU RELATÓRIOS

│ │ ├── 1 - Ocupação por período

│ │ ├── 2 - Faturamento por espaço

│ │ └── 0 - Voltar

│ │

│ ├── 5 – MENU SOBRE O SISTEMA

│ │ └── Exibição das informações do sistema e autores

│ │

│ └── 0 - SAIR

Arquivos do sistema de hospedagem:

1 - Cadastrar

2 - Listar

3 - Editar

4 - Excluir

0 - Voltar

Entrada de opção via:

csharp

int? opcao = EntradaHelper.LerInteiroComCancelamento("→ Escolha uma opção");

Saída limpa com mensagem elegante:

csharp

case 0:

InterfaceHelper.ExibirMensagemRetorno("Menu – Principal");

continuar = false;

break;

✅ Comportamento padrão ao navegar:

Limpeza suave entre opções (LimparTelaMesmoMenu() ou RetornoMenus(...))

Cabeçalhos consistentes com ExibirCabecalhoMenu(...)

Ações guiadas por perguntas em azul com PerguntaAzul(...)

Confirmações e alertas com cores padronizadas

Fluxo sempre em loop while (continuar) para manter o menu ativo

LerOpcaoEntreUmELimite, como vimos antes, é usado para cadastrar espaços e para seguir a regra, fizemos este método, no qual o cadastro tem que iniciar em 1 (listagem de: CHALÉS : 1- Chalé Fantasia , 2- CHALÉ maresia, listagem de CABANAS: 1- CABANA Opala, 2- CABANA Mercedez ... até o numero de cabanas cadastradas, sendo cada tipoEspaco iniciado em 1)

LimparTelaSistema() Limpeza total (ex: início do sistema)

LimparTelaRetornaParaMenuAnterior() Volta a um menu anterior com título no topo

LimparTelaMesmoMenu() Atualiza visual mantendo o contexto do menu atual

ExibirCabecalhoMenu() Com titulo do sistema, data e hora e titulo do menu. Aparece centralizado,

com titulo sistema escrito em branco, data e hora azul, traços continuo fino em toda largura da

tela na cor branca após qualquer LimparTela... para retomar o título (ou seja) manter o cabeçalho

existente sem mudanças, a menso que o titulo do tipo de menu não esteja incluido aqui nesse helper

e se aidna não estiver incluso nesse helper, vamos incluir: Menu - {titulo Menu} (MENU - ESPAÇOS,

MENU - RESERVA, MENU - CLIENTES, MENU - COMPROVANTES, MENU - RELATÓRIOS, MENU - SOBRE SISTEMA)

O que temos pronto para melhorar experiencia do usuário. Verificar opções e decidir se está ok ou se existe possibilidade melhor:

1- EntreAteELimite(...) - Selecionar opções de item entre 1 e limite. Se digitar valor invalido manda mensagem: Ops, digite um valor válido. Já embutir TresVezesZero() ? Utiizado em inserção e dados.

2- AguardaZero(...). Aguarda o usuário digitar exclusivamente 0 (é útil para os menus não mudarem a tela rapidamente até o usuário digitar o 0? Estamos tendo esse problema nesse exato momento.)

3- DigitouZeroRetornaMenu() - É a regra do sistema para retornar ao menu anterior

4- SimNao()

5- Se digitou errado, usuário recebe mensagem do sistema pedindo para digitar um valor válido. Acho que já temos algum método helper feito para isso. Verificar qual

Regras do sistema:

1. UmAteLimite (InterfaceHelper) - Aceita somente números de 1 até limite indefinido para fins de cadastro: espaços, reservas, hospedes, funcionários, etc, sendo que cada tipo sempre inicia em 1. Exemplo:

CABANAS

1. CABANA Fantasia
2. CABANA Maravilha

CHALÉS

1. CHALÉ Outubro
2. CHALÉ Novembro
3. CHALÉ Primavera

CASAS

1. CASA Pipoca
2. CASA Algodão doce
3. CASA Verão
   1. Aplicar **Regra sequencial de mensagens de valores digitados**

**Regra sequencial de mensagens de valores digitados**:

1. Digitou o esperado, segue o fluxo
2. Digitou valor inválido (vai para EntradaHelper MensagemErro()). Mensagem: Ops. Digite um valor válido.
3. Em qualquer momento da digitação pressionou enter 3 vezes (vai para EntradaHelper EnterTresVezes). Mensagem: Pressionou enter 3 vezes. Digite um valor válido para continuar ou 0 para retornar ao menu anterior (vai para InterfaceHelper DigitouZeroRetornaMenu())
4. Se resposta tiver que ser S/N (vai para EntradaHelper SimNao())
5. Botão para: Resetar dados digitados na página (8), retornar para menu anterior (9) e encerar sistema (0). Até que botões estejam ativos, usar 8, 9 e 0 ou já podemos colocar selecionáveis?

## 🧩 Tabela das Centrais Funcionais

| **Central** | **Responsabilidade** | **Substitui / Absorve** |
| --- | --- | --- |
| CentralEntrada(...) | Entrada simples de dados (texto, número), com ou sem cancelamento | LerTexto, LerTextoComCancelamento, LerDecimal, LerNumeroInteiro |
| CentralEntradaControlada(...) | Entrada numérica validada com limite, tratamento de ENTERs, opções 0/8/9 | UmAteLimite, EnterTresVezes, DigitouZeroRetornaMenu |
| CentralSaidaMensagens(...) | Mensagens visuais padronizadas com cores e semântica (Erro, Info, Sucesso) | MensagemErroMenu, MensagemErro, ExibirMensagemRetorno, etc. |
| CentralDeNavegacao(...) | Menu pós-ação (continuar, voltar, resetar, menu principal, sair) | Qualquer sequência de “O que deseja fazer agora...” usada após cadastros ou ações |
| CentralSaidaVisual(...) | Encerramentos elegantes, com mensagens finais, retornos visuais, delays | AguardaZero, comandos manuais de Environment.Exit, prints soltos de encerramento |
| SimNao(...) | Pergunta binária com suporte a variações (S, SIM, N, NAO, NÃO) | Nenhuma duplicata no momento, mas padrão oficial a partir de agora |

Vamos garantir que não exista:

* ConfirmarSimNao(...)
* PerguntaSimNao(...)
* LerSimNao(...)

Se houver algum desses, precisamos:

* Ver se estão em uso
* Unificar tudo em SimNao(...) com suporte a S, SIM, N, NÃO, NAO
* Remover ou renomear os antigos para evitar confusão

## 🧠 Sobre a fusão EntradaComControle(...)

Esse método pode ser o **cérebro das entradas numéricas com regras do sistema**, incluindo:

* Validação entre 1 e limite
* Contagem de ENTERs vazios
* Exibição de ExibirMensagemErroPadrao(...)
* Chamada de DigitouZeroRetornaMenu(...) ou CentralDeNavegacao(...) se necessário

📌 Ele pode substituir chamadas diretas a UmAteLimite(...) em contextos mais complexos.

## ✨ Próximos passos sugeridos

1. **Criar** CentralDeNavegacao(...) **completo** com as 5 opções
2. **Expandir** SimNao(...) **com variações e remover duplicatas**
3. **Implementar** EntradaComControle(...) **com lógica unificada**
4. Atualizar o **mapa de helpers** com:
   * Nome
   * Função
   * Onde usar
   * Substitui quem (se aplicável)

## Etapa 1 – Mapeamento de métodos com prefixos comuns

### 🔸 Métodos com Ler...

| **Método** | **Função atual** | **Observação** |
| --- | --- | --- |
| LerTexto() | Lê texto simples | OK, mas pode ser fundido com cancelamento |
| LerTextoComCancelamento() | Lê texto e permite cancelar com ENTER | Pode ser parâmetro opcional em LerTexto(...) |
| LerDecimal() | Lê número decimal | OK, mas pode ser fundido com LerNumero() |
| LerNumeroInteiro() | Lê número inteiro | Similar ao acima |

✅ **Sugestão:** Criar um único método LerEntrada(...) com parâmetros opcionais:

csharp

public static string? LerEntrada(string tipo, bool permiteCancelar = false, bool decimalPermitido = false)

### 🔸 Métodos com Exibir...

| **Método** | **Função atual** | **Observação** |
| --- | --- | --- |
| ExibirMensagemErroPadrao() | Mensagem de erro genérica | ✅ Centralizado |
| ExibirMensagemErroPersonalizada() | Mensagem de erro com texto customizado | Pode ser unificado com o acima |
| ExibirMensagemRetorno() | Mensagem de retorno visual | OK |
| ExibirMensagemEntrada() | Mensagem antes de entrada | Pode ser absorvido por ExibirPerguntaOpcao() |
| ExibirMensagemEntradaNumerada() | Entrada com numeração | Verificar se é usada |
| ExibirMensagemIntervalo() | Entrada com faixa de valores | Pode ser fundida |
| ExibirPerguntaOpcao() | Mostra pergunta visual com cor | ✅ Essencial |

✅ **Sugestão:** Criar um método ExibirMensagem(...) com tipo de mensagem como enum:

csharp

public enum TipoMensagem { Erro, Sucesso, Info, Entrada, Intervalo }

public static void ExibirMensagem(string texto, TipoMensagem tipo)

### 🔸 Métodos com Mensagem...

| **Método** | **Função atual** | **Observação** |
| --- | --- | --- |
| MensagemErroMenu() | Mensagem de erro específica de menu | Pode ser absorvida por ExibirMensagem(...) |
| MensagemErro() | Genérica, mas nome vago | Substituir por ExibirMensagemErroPadrao() |

## 🧠 Estratégia de fusão

| **Ação** | **Resultado esperado** |
| --- | --- |
| Fundir LerTexto, LerDecimal, LerNumeroInteiro | Criar LerEntrada(...) com tipo e validação |
| Fundir mensagens visuais | Criar ExibirMensagem(...) com enum de tipo |
| Manter SimNao() | ✅ Já validado e único |
| Eliminar duplicatas com nomes vagos (MensagemErro()) | Substituir por nomes claros e padronizados |

## 📘 Próximo passo

Se quiser, posso gerar agora:

1. A versão unificada de LerEntrada(...)
2. O novo ExibirMensagem(...) com enum
3. Atualizar o mapa de helpers com os que serão fundidos, renomeados ou removidos

| **Método** | **Visibilidade** | **Finalidade** | **Cor** |
| --- | --- | --- | --- |
| ExibirMensagemEntradaNumerada | private | Exibe instrução com limite superior (ex: entre 1 e 5) | Azul |
| ExibirMensagemEntrada | private | Variante com "0 para voltar" e limite (ex: até 10) | Azul |
| ExibirMensagemErroIntervalo | public | Exibe mensagem de erro com faixa numérica | Vermelho |
| ExibirMensagemRetorno | public | Mensagem de sucesso ao concluir algo | Verde |

# Program.cs

using System;

using System.Text;

using System.Threading;

using System.Globalization;

using DesafioProjetoHospedagem.Models;

using DesafioProjetoHospedagem.Utils;

using Menus;

// Cultura e acentuação

Thread.CurrentThread.CurrentCulture = new CultureInfo("pt-BR");

Thread.CurrentThread.CurrentUICulture = new CultureInfo("pt-BR");

Console.OutputEncoding = Encoding.UTF8;

// Animação inicial

ExibirTelaDeCarregamento();

// Listas principais

List<EspacoHospedagem> espacos = new();

List<Hospede> hospedes = new();

List<Hospede> hospedesDaReserva = new();

List<Reserva> reservas = new();

Reserva? reservaSelecionada = null;

// Chamada principal do sistema

MenuPrincipal.Exibir(espacos, hospedes, reservas, ref reservaSelecionada);

// Método auxiliar

static void ExibirTelaDeCarregamento()

{

    Console.Clear();

    Console.ForegroundColor = ConsoleColor.Yellow;

    InterfaceHelper.EscreverComAnimacao("Iniciando sistema . . . ");

    Console.ResetColor();

    Thread.Sleep(300);

    Console.WriteLine("\n");

}

# InterfaceHelper.cs

using System;

using System.Threading;

using DesafioProjetoHospedagem.Utils;

namespace DesafioProjetoHospedagem.Utils

{

    public static class InterfaceHelper

    {

        public static void LimparTela(string titulo)

        {

            Thread.Sleep(500); // Pequeno delay para suavizar o "piscar" da troca de telas

            Console.Clear();

            Console.ForegroundColor = ConsoleColor.Yellow;

            Console.WriteLine($" {titulo.ToUpper()} \n");

            Console.ResetColor();

        }

        public static void ExibirAnimacaoGiratoria(string mensagem = "Retornando", int ciclos = 12, int delay = 80)

        {

            string[] simbolos = new[] { "⠋", "⠙", "⠹", "⠸", "⠼", "⠴", "⠦", "⠧", "⠇", "⠏" };

            Console.WriteLine(); // espaçamento

            Console.ForegroundColor = ConsoleColor.Green; // 💚 Cor da animação

            Console.Write($" {mensagem}: ");

            for (int i = 0; i < ciclos; i++)

            {

                Console.Write(simbolos[i % simbolos.Length]);

                Thread.Sleep(delay);

                Console.Write("\b"); // volta o cursor

            }

            Console.Write("✔");

            Console.ResetColor();

            Thread.Sleep(400); // pausa final

            Console.WriteLine();

        }

        /\*//limpar tela entre menus sem animação

        public static void LimparTelaEntreMenus(string titulo)

        {

            ExibirAnimacaoGiratoria("Trocando de menu");

            Console.Clear();

            Console.ForegroundColor = ConsoleColor.Yellow;

            Console.WriteLine($" {titulo.ToUpper()} \n");

            Console.ResetColor();

        }

        \*/

        public static int ExibirPerguntaOpcao(string mensagem)

        {

            Console.ForegroundColor = ConsoleColor.Blue;

            Console.Write($"\n{mensagem}: ");

            Console.ResetColor();

            string? entrada = Console.ReadLine();

            while (!int.TryParse(entrada, out int valor))

            {

                Console.ForegroundColor = ConsoleColor.Red;

                Console.Write("Ops. Digite novamente: ");

                Console.ResetColor();

                entrada = Console.ReadLine();

            }

            return int.Parse(entrada);

        }

        //chamar nos com InterfaceHelper.ExibirPerguntaOpcao(" ex: Menu - Espaços");

        public static void LinhaSeparadora(char caractere = '─', ConsoleColor? cor = null)

        {

            if (cor.HasValue)

                Console.ForegroundColor = cor.Value;

            Console.WriteLine(new string(caractere, Console.WindowWidth));

            if (cor.HasValue)

                Console.ResetColor();

        }

        public static void PerguntaAzul(string texto)

        {

            Console.ForegroundColor = ConsoleColor.Blue;

            Console.Write(texto);

            Console.ResetColor();

        }

        public static void LimparTelaRetornoMenus(string titulo)

        {

            Console.Clear();

            Console.ForegroundColor = ConsoleColor.Yellow;

            Console.WriteLine($" {titulo.ToUpper()} \n");

            Console.ResetColor();

        }

        public static void ExibirMensagemRetorno(string tituloRetorno = "Menu anterior")

        {

            // Transição visual integrada: efeito giratório + limpeza

            ExibirAnimacaoGiratoria("Retornando ao menu anterior");

            LimparTelaRetornoMenus(tituloRetorno);

        }

        public static void ExibirTituloComData(string titulo)

        {

            Console.Clear();

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

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

            Console.WriteLine($"     {titulo.ToUpper()}");

            Console.ForegroundColor = ConsoleColor.Yellow;

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

            Console.ResetColor();

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

        }

        public static void EscreverComAnimacao(string texto, int delay = 30)

        {

            foreach (char c in texto)

            {

                Console.Write(c);

                Thread.Sleep(delay);

            }

            Console.WriteLine();

        }

        public static void ExibirCabecalhoMenu(string nomeMenu)

        {

            Console.Clear();

            // Título do sistema (sem cor)

            CentralizarTexto("SISTEMA DE HOSPEDAGEM – HOTEL FICTÍCIO");

            // Data e hora (azul, sem rótulo)

            string dataHora = $"{DateTime.Now:dd/MM/yyyy}  {DateTime.Now:HH:mm:ss}";

            CentralizarTexto(dataHora, ConsoleColor.Blue);

            // Separador com caractere leve ─ (sem cor)

            InterfaceHelper.LinhaSeparadora('─');

            // Nome do menu (em azul)

            CentralizarTexto(nomeMenu.ToUpper(), ConsoleColor.Blue);

            Console.WriteLine();

        }

        public static void CentralizarTexto(string texto, ConsoleColor? cor = null)

        {

            int posicao = Math.Max((Console.WindowWidth - texto.Length) / 2, 0);

            Console.SetCursorPosition(posicao, Console.CursorTop);

            if (cor.HasValue)

                Console.ForegroundColor = cor.Value;

            Console.WriteLine(texto);

            if (cor.HasValue)

                Console.ResetColor();

        }

        public static int LerOpcaoEntreUmELimite(string mensagem, int limiteSuperior)

        {

            int opcao;

            string? entrada;

            do

            {

                Console.ForegroundColor = ConsoleColor.Blue;

                Console.Write($"\n{mensagem} (0 para voltar): ");

                Console.ResetColor();

                entrada = Console.ReadLine();

                bool entradaValida = int.TryParse(entrada, out opcao);

                if (!entradaValida || (opcao != 0 && (opcao < 1 || opcao > limiteSuperior)))

                {

                    Console.ForegroundColor = ConsoleColor.Red;

                    Console.WriteLine($"Opção inválida. Digite um número entre 1 e {limiteSuperior}, ou 0 para voltar.");

                    Console.ResetColor();

                }

            } while (opcao != 0 && (opcao < 1 || opcao > limiteSuperior));

            return opcao;

        }

    }

}

# EntradaHelper.cs

using System;

namespace DesafioProjetoHospedagem.Utils

{

    public static class EntradaHelper

    {

        // ======== MÉTODOS AUXILIARES INTERNOS =========

        private static bool ConfirmarCancelamento()

        {

            string? resposta;

            do

            {

                Console.ForegroundColor = ConsoleColor.Blue;

                Console.Write("\nVocê pressionou Enter várias vezes. Deseja cancelar e voltar ao menu atual? (S/N): ");

                Console.ResetColor();

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

                if (resposta == "S") return true;

                if (resposta == "N") return false;

                Console.ForegroundColor = ConsoleColor.Yellow;

                Console.WriteLine("Entrada inválida. Digite apenas S ou N.");

                Console.ResetColor();

            } while (true);

        }

        private static void MensagemErro(string texto = "Valor inválido. Tente novamente.")

        {

            Console.ForegroundColor = ConsoleColor.Red;

            Console.WriteLine(texto);

            Console.ResetColor();

        }

        // ======== ENTRADAS BÁSICAS =========

        public static string LerTexto(string mensagem)

        {

            string? entrada;

            do

            {

                Console.Write($"{mensagem}: ");

                entrada = Console.ReadLine();

            }

            while (string.IsNullOrWhiteSpace(entrada));

            return entrada.Trim();

        }

        public static int LerInteiro(string mensagem)

        {

            int valor;

            while (true)

            {

                Console.Write($"{mensagem}: ");

                if (int.TryParse(Console.ReadLine(), out valor))

                    return valor;

                MensagemErro();

            }

        }

        public static decimal LerDecimal(string mensagem)

        {

            while (true)

            {

                Console.Write($"{mensagem}: ");

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

                if (!string.IsNullOrWhiteSpace(entrada))

                {

                    entrada = entrada.Replace('.', ',');

                    if (decimal.TryParse(entrada, out decimal valor))

                        return valor;

                }

                MensagemErro();

            }

        }

        public static int LerNumeroInteiro(string mensagem)

        {

            int numero;

            while (true)

            {

                Console.Write($"{mensagem}: ");

                string? entrada = Console.ReadLine();

                if (!string.IsNullOrWhiteSpace(entrada) && int.TryParse(entrada, out numero))

                    return numero;

                MensagemErro();

            }

        }

        // ======== ENTRADAS COM CANCELAMENTO =========

        public static string? LerTextoComCancelamento(string mensagem, int maxVazios = 3)

        {

            int vazios = 0;

            while (true)

            {

                Console.Write($"{mensagem}: ");

                string? entrada = Console.ReadLine();

                if (!string.IsNullOrWhiteSpace(entrada))

                    return entrada.Trim();

                vazios++;

                if (vazios >= maxVazios && ConfirmarCancelamento())

                    return null;

            }

        }

        public static int? LerInteiroComCancelamento(string mensagem, int maxVazios = 3)

        {

            int vazios = 0;

            while (true)

            {

                Console.WriteLine();

                Console.ForegroundColor = ConsoleColor.Cyan;

                Console.Write("→ ");

                Console.Write($"{mensagem}: ");

                Console.ResetColor();

                string? entrada = Console.ReadLine();

                if (!string.IsNullOrWhiteSpace(entrada) && int.TryParse(entrada, out int valor))

                    return valor;

                if (string.IsNullOrWhiteSpace(entrada))

                {

                    vazios++;

                    if (vazios >= maxVazios && ConfirmarCancelamento())

                        return null;

                }

                else

                {

                    MensagemErro(); // Aqui você pode mostrar uma mensagem com cor e símbolo (⚠️)

                }

            }

        }

        public static decimal? LerDecimalComCancelamento(string mensagem, int maxVazios = 3)

        {

            int vazios = 0;

            while (true)

            {

                Console.Write($"{mensagem}: ");

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

                if (string.IsNullOrWhiteSpace(entrada))

                {

                    vazios++;

                    if (vazios >= maxVazios && ConfirmarCancelamento())

                        return null;

                    continue;

                }

                entrada = entrada.Replace('.', ',');

                if (decimal.TryParse(entrada, out decimal valor))

                    return valor;

                MensagemErro();

            }

        }

        public static bool LerConfirmacaoSimNao(string mensagem)

        {

            string? resposta;

            do

            {

                Console.ForegroundColor = ConsoleColor.Blue;

                Console.Write($"{mensagem} (S/N): ");

                Console.ResetColor();

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

                if (resposta == "S") return true;

                if (resposta == "N") return false;

                Console.ForegroundColor = ConsoleColor.Yellow;

                Console.WriteLine("Entrada inválida. Digite apenas S ou N.");

                Console.ResetColor();

            } while (true);

        }

        public static string? LerEmailComValidacao(string mensagem)

        {

            string? email;

            do

            {

                email = LerTextoComCancelamento(mensagem);

                if (email == null) return null;

                if (email.Contains("@") && email.Contains(".") && email.Length >= 5)

                    return email;

                Console.ForegroundColor = ConsoleColor.Yellow;

                Console.WriteLine("Formato de e-mail inválido. Tente novamente.");

                Console.ResetColor();

            } while (true);

        }

        public static int LerOpcaoEntreUmELimite(string mensagem, int limiteSuperior)

        {

            int opcao;

            string? entrada;

            do

            {

                Console.ForegroundColor = ConsoleColor.Blue;

                Console.Write($"\n{mensagem} (0 para voltar): ");

                Console.ResetColor();

                entrada = Console.ReadLine();

                bool entradaValida = int.TryParse(entrada, out opcao);

                if (!entradaValida || (opcao < 0 || opcao > limiteSuperior))

                {

                    MensagemErro($"Opção inválida. Digite um número entre 0 e {limiteSuperior}.");

                }

            } while (opcao < 0 || opcao > limiteSuperior);

            return opcao;

        }

    }

}

# MenuPrincipal.cs

using System;

using System.Collections.Generic;

using DesafioProjetoHospedagem.Models;

using DesafioProjetoHospedagem.Utils;

namespace Menus

{

    public static class MenuPrincipal

    {

        public static void Exibir(

        List<EspacoHospedagem> espacos,

        List<Hospede> hospedes,

        List<Reserva> reservas,

        ref Reserva? reservaSelecionada)

        {

            bool continuar = true;

            while (continuar)

            {

                Console.Clear();

                // Título do sistema

                InterfaceHelper.CentralizarTexto("SISTEMA DE HOSPEDAGEM – HOTEL FICTÍCIO");

                // Hora

                string dataHora = $" {DateTime.Now:dd/MM/yyyy HH:mm:ss}";

                InterfaceHelper.CentralizarTexto(dataHora, ConsoleColor.Blue);

                // Separador

                Console.WriteLine("".PadRight(Console.WindowWidth, '\_'));

                // Título do menu

                InterfaceHelper.CentralizarTexto("MENU PRINCIPAL", ConsoleColor.Blue);

                Console.WriteLine();

                // Opções

                Console.WriteLine("1 - Espaços");

                Console.WriteLine("2 - Clientes");

                Console.WriteLine("3 - Reservas");

                Console.WriteLine("4 - Emitir Comprovante de Reserva");

                Console.WriteLine("5 - Relatórios e Estatísticas");

                Console.WriteLine("6 - Sobre o Sistema");

                Console.WriteLine("0 - Sair\n");

                //Pergunta final em azul

                int opcao = InterfaceHelper.ExibirPerguntaOpcao("Escolha sua opção");

                switch (opcao)

                {

                    case 1:

                        InterfaceHelper.LimparTela("Menu de Espaços");

                        MenuEspacos.ExibirMenuEspacos(espacos, reservas);

                        break;

                    case 2:

                        InterfaceHelper.LimparTela("Menu de Clientes");

                        MenuClientes.ExibirMenuClientes(hospedes, reservas);

                        break;

                    case 3:

                        InterfaceHelper.LimparTela("Menu de Reservas");

                        MenuReservas.Exibir(reservas, hospedes, espacos);

                        break;

                    case 4:

                        InterfaceHelper.LimparTela("Comprovante de Reserva");

                        MenuEmitirComprovanteReserva.Exibir(reservas, ref reservaSelecionada);

                        break;

                    case 5:

                        InterfaceHelper.LimparTela("Relatórios e Estatísticas");

                        MenuRelatorios.Exibir(espacos, reservas, hospedes);

                        break;

                    case 6:

                        InterfaceHelper.LimparTela("Sobre o Sistema");

                        MostrarSobreSistema.Exibir();

                        break;

                    case 0:

                        continuar = false;

                        Console.ForegroundColor = ConsoleColor.Blue;

                        Console.WriteLine("\nEncerrando o sistema... Até logo!");

                        Console.ResetColor();

                        break;

                    default:

                        Console.ForegroundColor = ConsoleColor.Yellow;

                        Console.WriteLine("\nOpção inválida. Tente novamente.");

                        Console.ResetColor();

                        InterfaceHelper.ExibirMensagemRetorno();

                        break;

                }

            }

        }

    }

}

/\*antigo

using System;

using System.Collections.Generic;

using DesafioProjetoHospedagem.Models;

using DesafioProjetoHospedagem.Utils;

namespace Menus

{

    public static class MenuPrincipal

    {

        public static void Exibir(

        List<EspacoHospedagem> espacos,

        List<Hospede> hospedes,

        List<Reserva> reservas,

        ref Reserva? reservaSelecionada)

        {

            bool continuar = true;

            while (continuar)

            {

                Console.Clear();

                // Título do sistema

                InterfaceHelper.CentralizarTexto("SISTEMA DE HOSPEDAGEM – HOTEL FICTÍCIO");

                // Hora

                string dataHora = $" {DateTime.Now:dd/MM/yyyy HH:mm:ss}";

                InterfaceHelper.CentralizarTexto(dataHora, ConsoleColor.Blue);

                // Separador

                Console.WriteLine("".PadRight(Console.WindowWidth, '\_'));

                // Título do menu

                InterfaceHelper.CentralizarTexto("MENU PRINCIPAL", ConsoleColor.Blue);

                Console.WriteLine();

                // Opções

                Console.WriteLine("1 - Espaços");

                Console.WriteLine("2 - Clientes");

                Console.WriteLine("3 - Reservas");

                Console.WriteLine("4 - Emitir Comprovante de Reserva");

                Console.WriteLine("5 - Relatórios e Estatísticas");

                Console.WriteLine("6 - Sobre o Sistema");

                Console.WriteLine("0 - Sair\n");

                //Pergunta final em azul

                InterfaceHelper.PerguntaAzul("Escolha sua opção: ");

                InterfaceHelper.ExibirPerguntaOpcao("Escolha sua opção: ");

                string? entrada = Console.ReadLine();

                if (!int.TryParse(entrada, out int opcao))

                {

                    Console.ForegroundColor = ConsoleColor.Red;

                    Console.WriteLine("\nEntrada inválida. Digite um número entre 0 e 6.");

                    Console.ResetColor();

                    InterfaceHelper.ExibirMensagemRetorno();

                    continue;

                }

                switch (opcao)

                {

                    case 1:

                        InterfaceHelper.LimparTela("Menu de Espaços");

                        MenuEspacos.Exibir(espacos, reservas);

                        break;

                    case 2:

                        InterfaceHelper.LimparTela("Menu de Clientes");

                        MenuClientes.ExibirMenuClientes(hospedes, reservas);

                        break;

                    case 3:

                        InterfaceHelper.LimparTela("Menu de Reservas");

                        MenuReservas.Exibir(reservas, hospedes, espacos);

                        break;

                    case 4:

                        InterfaceHelper.LimparTela("Comprovante de Reserva");

                        MenuEmitirComprovanteReserva.Exibir(reservas, ref reservaSelecionada);

                        break;

                    case 5:

                        InterfaceHelper.LimparTela("Relatórios e Estatísticas");

                        MenuRelatorios.Exibir(espacos, reservas, hospedes);

                        break;

                    case 6:

                        InterfaceHelper.LimparTela("Sobre o Sistema");

                        MostrarSobreSistema.Exibir();

                        break;

                    case 0:

                        continuar = false;

                        Console.ForegroundColor = ConsoleColor.Blue;

                        Console.WriteLine("\nEncerrando o sistema... Até logo!");

                        Console.ResetColor();

                        break;

                    default:

                        Console.ForegroundColor = ConsoleColor.Yellow;

                        Console.WriteLine("\nOpção inválida. Tente novamente.");

                        Console.ResetColor();

                        InterfaceHelper.ExibirMensagemRetorno();

                        break;

                }

            }

        }

    }

}

\*/