trilha-net-explorando-desafio

Sistema de hospedagem

│

|── 📁Comprovantes

│ └── .txt

|

|── 📁Contexto

│ └── SessaoAtual.cs – Espacos, Clientes, ClienteSelecionado, ClientesSelecionados, Reservas, ReservaSelecionada

| SanitizarDados, VErificarSessaoGlobal,

|

|── 📁 Controladoras

| |

│ ├── ClienteControlador.cs – void: MenuDoCadastroDeCliente, CadastrarCliente, EditarCadastroCliente,

| | ListarCadastroClientes

| |

│ ├── CriarReservaControladoar.cs – void: ContinuarCriacao, IniciarComEspacoVinculado,

| |

│ ├── EditarResrevaControladora.cs – voidContinuarEdicao,

| |

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

| | ExcluirCadastroEspaco, BuscarEspacoPorFiltro

| |

│ └── ReservaControladora:class ReservaControladora: void ObterStatusEspaco

|

|

|── 📁 Menus

│ ├── MenuPrincipal.cs – void: ExibirPrincipal, //RodapeGlobalLoop,

| |

│ ├── MenuClientes.cs – Resera?ExibirMenuClientes, 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

| |

│ ├── MenuReservas.cs – ExibirMenuReservas, CruiarReserva, ExibirSubMenuClientes, VincularClientesAReserva,

| | ListarCadastroReservas, CancelarReserva, EncerrarReserva, ExcluirCadastroReserva,

| |

│ └── MenuSobreSistema.cs - Informações

│

├── 📁 Models

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

| |

| └── EspacoHospedagem.cs.cs – tipoEspaco, id, tipoAbreviado, nomeFantasia, capacidade, valorDiaria,

| | quantidadeQuartos, quantidadeSuites, descricao

| |

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

| | |\_\_HorasUltrapassadas, FinalizarReserva, ObterQuantidadeClientes, Reserva, AtualizarClientesDaReserva,

| | VincularClientes, VincularEspaco, TotalDias, CalcularValorSemDesconto, CalcularDesconto,

| | CalcularValorComDesconto, CalcularTaxaAcrescimo, CalcularValorTotal, CalcularResumoFinanceiro

| |

| └── ResumoFinanceiro.cs – ValorDiaria, ValorSemDesconto, ValorDoDesconto, ValorComDesconto, ValorTaxaExtra,

| | ValorTotalEstimado

|

├── 📁 Servicos

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

|

├── 📁 Utils

| |\_\_ CentralDeNavegacao.cs - //DigitouZeroRetornaMenu, EncerrarSistema, ReiniciarEntrada, VoltarAoMenu

| |

| |\_\_CentralEntrada

| | |\_\_ class CentralEntrada: void – //LerTextoComCancelamento, LerTextoObrigatorio (MenuClientes,),

| | | //LerValorMonetarioComEscape, //LerDataObrigatoriaComEscape, //LerTextoComEscape,

| | | //LerNumeroInteiroComEscape, ExibirPerguntaOpcao, OperacaoCanceladaException,

| | |

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

| | |

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

| | //LerInteiroComCancelamento, LerInteiroMinMax, LerInteiroMinMaxSN, LerTextoComSN,

| | LerEmailComSN, LerNumeroComoString, LerNumeroInteiro, SimNao,

| |

| |\_\_CentralSaida

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

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

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

| | LimparTela (ReservaControladora,), LimparTelaMesmoMenu, LimparTelaRetornoMenus,

| | LinhaSeparadora (MenuEspacos,), MensagemPadraoSucesso, MensagemCancelamento, Submenu, Subtitulo,

| |

│ ├── EntradaHelper.cs: bool ConfirmarCancelamento, void – ExibirMensagemEntradaNumerada,

| | ExibirMensagemEntrada, ExibirMensagemRetorno (ReservaControladora.cs),

| | |

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

| | EscreverComAnimacao, EditarTexto, ExibirTituloComData

| |

│ ├── MenuRodape.cs – ExibirRodape (string nomeSubmenu, Action? aoVoltarParaSubmenu = null,

| | Action? aoVoltarMenuPrincipal = null)

| |

│ ├── MenuRodapeGlobal.cs – ExibirRodapeGlobal (string nomesubmenu)

│

|── DesafioProjetoHospedagem.csproj

|

|── Program.cs

│

|── README.md

│

|── trilha-net-explorando-desafio.sl

Links:

Cadastro e edição de Espaços

MenuPrincipal

└── MenuEspacos

└── EspacoControlador.BuscarEspacoPorFiltro()

└── SessaoAtual.EspacoSelecionado

└── CriarReservaControladora.IniciarComEspacoVinculado()

└── SessaoAtual.ReservaSelecionada

└── ReservaControladora.ObterStatusEspaco()

Cadastro e edição de Clientes

MenuClientes.ExibirMenuClientes()

└── ClienteControlador.CadastrarCliente()

└── SessaoAtual.Clientes.Add()

└── ClienteControlador.EditarCadastroCliente()

└── SessaoAtual.ClienteSelecionado

Criação de Reservas

MenuReservas.ExibirMenuReservas()

└── CriarReservaControladora.IniciarComEspacoVinculado()

└── EspacoControlador.BuscarEspacoPorFiltro()

└── SessaoAtual.EspacoSelecionado

└── new Reserva(clientesSelecionados, espacoSelecionado, dias)

└── SessaoAtual.ReservaSelecionada

└── CriarReservaControladora.ContinuarCriacao()

└── VincularClientes(SessaoAtual.ClientesSelecionados)

└── FinalizarReserva()

└── SessaoAtual.Reservas.Add(reservaFinal)

Vinculação de Clientes à Reserva

MenuReservas.ExibirSubMenuClientes()

└── ClienteControlador.ListarCadastroClientes()

└── Seleção manual → SessaoAtual.ClientesSelecionados

└── Reserva.VincularClientes(SessaoAtual.ClientesSelecionados)

Emissão de Comprovante de Reserva

MenuComprovantes.EmitirComprovante()

└── SessaoAtual.ReservaSelecionada

└── Reserva.CalcularResumoFinanceiro()

└── ResumoFinanceiro (ValorTotal, Desconto, TaxaExtra)

└── Geração visual do comprovante

Cancelamento ou Encerramento de Reserva

MenuReservas.CancelarReserva()

└── SessaoAtual.ReservaSelecionada

└── Reserva.Status = "Cancelada"

└── Atualização na lista SessaoAtual.Reservas

MenuReservas.EncerrarReserva()

└── SessaoAtual.ReservaSelecionada

└── Reserva.Status = "Encerrada"

└── Reserva.DataCheckout = hoje

Relatórios e Listagens

MenuRelatorios.Exibir()

└── SessaoAtual.Reservas → agrupadas por status

└── SessaoAtual.Clientes → hospedados vs não hospedados

└── SessaoAtual.Espacos → agrupados por tipo

Rodapé e navegação global

MenuRodape.ExibirRodape(submenu, aoVoltarParaSubmenu, aoVoltarMenuPrincipal)

MenuRodapeGlobal.ExibirRodapeGlobal("SubmenuAtual")

CentralDeNavegacao → NavegacaoAtual.Voltar()

→ DigitouZeroRetornaMenu()

## 🏛️ PAINEL TÉCNICO DO SISTEMA DE HOSPEDAGEM

### 🔹 1. Módulo de Espaços

| **Menu** | **Controladora** | **Sessão** | **Ações Principais** |
| --- | --- | --- | --- |
| MenuEspacos | EspacoControlador | SessaoAtual.Espacos | Cadastro, Edição, Listagem por Tipo |
|  | BuscarEspacoPorFiltro() ✅ | EspacoSelecionado | Localiza espaço por ID ou nome |

**Dependências:** → Espaço precisa estar cadastrado para reserva. → Espaço selecionado é salvo globalmente para uso em reserva.

### 🔹 2. Módulo de Clientes

| **Menu** | **Controladora** | **Sessão** | **Ações Principais** |
| --- | --- | --- | --- |
| MenuClientes | ClienteControlador | SessaoAtual.Clientes | Cadastro, Edição, Listagem |
|  |  | ClienteSelecionado | Cliente ativo para edição |
|  |  | ClientesSelecionados | Lista de clientes vinculados a reserva |

**Dependências:** → Reserva exige pelo menos um cliente disponível. → Vinculação é feita após seleção manual dos clientes.

### 🔹 3. Módulo de Reservas

| **Menu** | **Controladora** | **Sessão** | **Ações Principais** |
| --- | --- | --- | --- |
| MenuReservas | CriarReservaControladora | ReservaSelecionada | Criação da reserva, vinculação de clientes |
|  |  | Reservas | Lista geral de reservas criadas |
|  | ContinuarCriacao() | ClientesSelecionados | Finaliza reserva e vincula clientes |
|  |  | EspacoSelecionado | O espaço vinculado à reserva |

## --------------------------------------------------------------------------------------------------------------------------------------

## Ações principais do sistema

| **Ação** | **Finalidade** | **Origem esperada** |
| --- | --- | --- |
| **Cadastrar** | Criar novo cliente ou espaço | MenuClientes / MenuEspacos |
| **Editar** | Alterar dados já existentes (cliente ou reserva) | Submenus de edição |
| **Cancelar** | Cancelar uma reserva ativa antes da saída | MenuReservas / SubmenuCancelar |
| **Excluir** | Remover completamente um cadastro (reserva ou espaço) | SubmenuExcluir |
| **Criar Reserva** | Iniciar reserva nova associando cliente e espaço | MenuReservas |
| **Vincular** | Associar cliente ou espaço a uma reserva em andamento | CriarReserva / VincularCliente |
| **Listar** | Visualizar dados existentes (clientes, reservas, espaços) | Qualquer menu |

## 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** |

## Interpretação dos ENTERs — Navegação gentil e inteligente

| **N° de ENTERs consecutivos** | **Interpretação do sistema** | **Ação realizada** |
| --- | --- | --- |
| 1º ENTER | **Confirmação de entrada** | Segue o fluxo normalmente — o valor digitado é processado |
| 2º ENTER | **Alerta interno ao sistema** | Sistema percebe hesitação, mas não reage — aguarda próxima ação do usuário |
| 3º ENTER | **Decisão implícita de saída ou dúvida forte** | Sistema exibe mensagem: “Deseja sair desta página? (S/N)” e chama MiniMenu |

// ENTERs consecutivos ativam lógica emocional de saída:

// 1º = entrada normal

// 2º = hesitação detectada (sem ação)

// 3º = pergunta: "Deseja sair desta página? (S/N)" → MenuRodape ou navegação reversa, onde for indicada

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

**----------------------------------------------------------------------------------------------------------------------------------**

**Fluxo padrão:**

1. Buscar espaço
2. Criar reserva com dias e check-in
3. Vincular clientes
4. Finalizar reserva
5. Salvar reserva global

### 🔹 4. Módulo de Comprovantes

| **Menu** | **Controladora (implícita)** | **Sessão** | **Ações Principais** |
| --- | --- | --- | --- |
| MenuComprovantes | - | ReservaSelecionada | Exibe resumo financeiro, gera comprovante |

**Dependência:** → Reserva deve estar criada antes da emissão. → Utiliza cálculo financeiro baseado em espaço e dias.

### 🔹 5. Módulo Rodapé & Navegação

| **Componente** | **Ações** |
| --- | --- |
| MenuRodape | Exibe opções, volta ao menu principal ou submenu |
| MenuRodapeGlobal | Rodapé padronizado por contexto |
| NavegacaoAtual.Voltar() | Retorna ao ponto anterior sem quebra de fluxo |

**Regras de navegação:** ❌ Nada de ESC, nulo ou vazios como saída ✅ Fluxo sempre guiado por menus e rodapés

### 🔹 6. Sanitização & Validação

| **Local** | **Método** | **Objetivo** |
| --- | --- | --- |
| SessaoAtual | SanitizarDados() ✅ | Padroniza dados para evitar duplicações |
|  | VerificarSessaoGlobal() ✅ | Valida se há dados essenciais antes de navegar |

### 🔹 7. Relatórios (em construção ou planejado)

| **Menu** | **Sessão** | **Objetivo** |
| --- | --- | --- |
| MenuRelatorios | Reservas, Clientes, Espacos | Agrupamento por tipo, status, presença |

## 🧭 Fluxos Visuais (Resumo)

plaintext

MenuPrincipal

├── MenuEspacos

│ └── BuscarEspacoPorFiltro → SessaoAtual.EspacoSelecionado

├── MenuClientes

│ └── CadastrarCliente → SessaoAtual.Clientes

├── MenuReservas

│ └── IniciarComEspacoVinculado → CriarReserva → ContinuarCriacao

│ └── Reserva.Finalizar → SessaoAtual.Reservas.Add()

├── MenuComprovantes

│ └── Emitir → SessaoAtual.ReservaSelecionada → GerarResumoFinanceiro

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

## Como seguir para não duplicar ou bagunçar o codigo:

1. **Sempre olhar o mapa antes de sugerir algo novo** 👉 Se já existe, a gente refatora ou adapta!
2. **Identificar o método mais parecido existente** 👉 Às vezes ele faz 80% do que precisamos — só falta aquele ajuste fino.
3. **Criar apenas se não houver nenhuma versão próxima** 👉 E mesmo assim, vamos checar se algum método pode ser estendido ou sobrecarregado.
4. **Documentar o que foi modificado ou consolidado** 👉 Assim você sempre sabe o que evoluiu e pode acompanhar melhor o crescimento do sistema.

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

## 13jul2025

## ✅ Lista de Tarefas para Estabilizar o Sistema

### 🔧 1. Sanitizar os dados em SessaoAtual - ok

* Padronizar TipoEspaco, Status, NomeFantasia com .Trim().ToUpper()
* Evitar duplicidade nos agrupamentos da listagem global de espaços

### 🛡️ 2. Criar método VerificarSessaoGlobal()

Colocar no iniciode ExibirMenuresrevas, ExibirMenuPrincipal, ExibirMenuClientes - feito

* Validar se existem espaços, clientes e reservas antes de usar menus
* Exibir mensagens realistas sobre o que está ou não cadastrado

### 🧠 3. Corrigir mensagens incoerentes - ok

* Exibir corretamente se há clientes vinculados na reserva
* Evitar frases como "clientes vinculados" quando não há nenhum

### 🔄 4. Tornar métodos como BuscarEspacoPorFiltro() totalmente modulares -ok

* Remover ações internas como redirecionamento
* Deixar apenas a seleção do espaço e retorno — menu decide o que fazer

### 🧭 5. Mapear todos os fluxos entre menus e controladoras - ok

* Criar visualização de quem chama quem
* Identificar duplicações, sobreposições e chamadas desnecessárias

### 🎯 6. Criar rotina de validação antes de criar reserva

* Verificar se há pelo menos um cliente antes de prosseguir
* Sugerir cadastro ou redirecionamento elegante

## Trilha técnica de verificação — começando do topo

### 🎯 Etapa 1: MenuPrincipal.cs

* **Verifica:**
  + CabecalhoMenus() está presente no início da exibição?
  + Após cada escolha, chama rodapé via MenuRodape.ExibirRodapeGlobal()
  + Fluxo jamais termina em return ou Console.ReadLine() sem contexto
* **Linkagem esperada:**
  + Direciona para:
    - MenuClientes.ExibirMenuClientes()
    - MenuEspacos.ExibirMenuEspacos()
    - MenuReservas.ExibirMenuReservas()
    - MenuComprovantes.EmitirComprovante()
    - MenuRelatorios.Exibir()
    - MenuSobreSistema.Exibir()

### 🎯 Etapa 2: MenuEspacos.cs

* **Chama corretamente:**
  + EspacoControlador.CadastrarEspaco()
  + EspacoControlador.EditarCadastroEspaco()
  + EspacoControlador.BuscarEspacoPorFiltro()
* **Verifica:**
  + Está atualizando SessaoAtual.EspacoSelecionado após seleção?
  + Rodapé aparece no fim de cada operação
  + Nenhum null retornado na seleção de espaço

### 🎯 Etapa 3: EspacoControlador.cs

* **Verifica:**
  + Métodos têm mensagens via CentralSaida (sem Console.WriteLine)
  + BuscarEspacoPorFiltro() usa loop e nunca retorna null
  + Após cadastro, retorno ao menu é feito com rodapé claro

### 🎯 Etapa 4: MenuClientes.cs

* **Linkagem:**
  + ClienteControlador.CadastrarCliente()
  + ClienteControlador.EditarCadastroCliente()
  + ClienteControlador.ListarCadastroClientes()
* **Sessão:**
  + Atualiza corretamente SessaoAtual.ClienteSelecionado e ClientesSelecionados
  + Navegação guiada com MenuRodape ou retorno claro

### 🎯 Etapa 5: MenuReservas.cs

* **Chama:**
  + CriarReservaControladora.IniciarComEspacoVinculado()
  + CriarReservaControladora.ContinuarCriacao()
  + ReservaControladora.ObterStatusEspaco()
* **Verifica:**
  + Reserva criada com SessaoAtual.ReservaSelecionada
  + Sempre mostra rodapé após criação, cancelamento ou encerramento
  + Nunca existe return bruto sem orientação visual

### 🎯 Etapa 6: MenuComprovantes.cs

* **Usa:**
  + SessaoAtual.ReservaSelecionada
  + Fluxo segue com emissão → resumo financeiro → geração visual
* **Verifica:**
  + Mensagem clara com CentralSaida
  + Rodapé presente após emissão ou erro

### 🎯 Etapa 7: Menus auxiliares (MenuSobreSistema, MenuRelatorios)

* **Verifica:**
  + Todos têm CabecalhoMenus() + encerramento com RodapeGlobal
  + Informações exibidas com visual consistente via CentralSaida

## ✅ Proposta de Auditoria

Posso agora gerar um **quadro técnico por menu**, marcando:

| **Menu** | **Cabecalho ok** | **Rodapé ok** | **SessaoAtual ok** | **Linkagem ok** | **null retornado?** | **Observações** |
| --- | --- | --- | --- | --- | --- | --- |

### 🧹 7. Corrigir visualização da listagem de espaços

* Garantir que o título do grupo (ex: APARTAMENTO) apareça uma vez
* Verificar GroupBy() com .Trim().ToUpper() no TipoEspaco

### 🪶 8. Ajustar rodapés e menus que aparecem duplicados

* Evitar ExibirPrincipal() ser chamado duas vezes
* Usar navegação inteligente com NavegacaoAtual.Voltar() e flags de retorno

## Informações que devem aparecer na ListarEspacosGlobal()

Aqui está um checklist de tudo o que precisa estar visível para cada espaço listado:

Com carinho, Solange, aqui está o método ListarEspacosGlobal() completo, atualizado e alinhado com tudo que você pediu — com:

* 🟦 Cabeçalho com título, data e hora (CabecalhoMenus)
* 🟦 Agrupamento por tipo de espaço com separadores visuais
* 🟩 Dados completos do espaço: código, nome fantasia, capacidade, diária, quartos, suítes, descrição
* 🟥 Bloco "DETALHES" com status colorido
* 🟥 Se houver reserva ativa:
  + Check-in
  + Previsão de checkout
  + Resumo financeiro (ResumoFinanceiro)
  + Hóspedes numerados ou mensagem emocional
* 🔁 Navegação reversa via NavegacaoAtual e retorno ao fluxo original
* 🎯 E se veio de forma neutra ("Padrao"), oferece um mini-menu com ações

### 🔹

### Identidade visual:

* TipoEspaco como título (em **Cyan**, centralizado)
* Linha separadora em **DarkGray**, ocupando toda a largura

### 🔹 Dados do espaço:

* TipoAbreviado + ID formatado (ex: CHA 001) – em **Cyan**
* NomeFantasia
* Capacidade: quantidade de pessoas
* ValorDiaria: formatado com 2 casas decimais (R$ 400,00)
* QuantidadeQuartos e QuantidadeSuites, com destaque em **Cyan** para valores
* Descrição: exibida com prefixo
* Status: "Livre" ou "Ocupado" — com cor verde ou vermelha

### 🔹 Dados da reserva vinculada (se houver):

* Check-in: data de entrada real
* Previsão de checkout
* Estimativa financeira: total acumulado até o momento
* Hospedes: lista de nomes vinculados

## coisas que temos que fazer no documento inteiro. R

## 1-Eevisar cada metodo usado no meio do codigo para não haver escapes ou null (ver qual metodo se encaixa e modificar os metodos usados para que se encaixem),

2-Modificar métodos antiquados

## 3-implementar navegacao reversa emt udo

## 4- implementar rodape independente em todas as paginas,

## 5-tirando todos os desnecessários, inclusive do meio do projeto, pois o independente sempre estará fixado na tela.

## Arquivos com navegação reversa

// ➤ Registrar para navegação reversa: NavegacaoAtual.Registrar(() => MenuPrincipal.ExibirPrincipal());

Navegacao reversa: Após abertura da chave:

NavegacaoAtual.Registrar(() => MenuReservas.ExibirMenuReservas());

Nofinal:  MenuRodapeGlobal.ExibirRodapeGlobal("Menu Principal");

ExibirMenuReserva

ExibirMenuComprovantes

Verificar todos os arquivos: navegação reversa, menurodape, menurodapeGlobal, se ainda tem métodos com cancelamento ou antigos. Usados: CentralEntradaControladora.SimNao, LerInteiroMinMaxComSN, LerInteiroMinMax,

,

,

1. Menucomprovantes.cs – ok

### Salvamento do arquivo .txt

* Verifica se salvarEmArquivo é true
* Usa UTF8Encoding sem BOM → ótimo pra compatibilidade
* Cria o nome do arquivo com DateTime.Now → único e rastreável
* Salva com File.WriteAllText(...)
* Exibe confirmação com mensagem amigável

🟢 Tudo certo e bem padronizado!

## Etapas

1. Terminar de colocar navegacaoAtual e erficar rodapés no aprojeto todo
2. Verificar arquivos repetidos em menureservas.cs e resrvas.cs

### Navegação e retorno emocional

* Garantir que NavegacaoAtual.Registrar(...) esteja presente nos fluxos certos
* Validar o comportamento do Voltar() em todos os rodapés

### 🎨 Telas e integração

* Finalizar a criação ou validação de TelaClientes.SelecionarMultiplos()
* Montar tela ou método para visualização de resumo financeiro da reserva

### 🧹 Organização interna

* Revisar classes que têm métodos soltos fora de escopo
* Consolidar chamadas duplicadas ao rodapé
* Eventualmente extrair NavegacaoAtual para arquivo separado (se quiser modularizar mais)



## Estrutura Atual vs. Visual Refinado Proposto

| **Elemento** | **Atual** | **Proposto (Refinado)** |
| --- | --- | --- |
| Divisória horizontal | ─ simples | ─ em toda a largura (Console.WindowWidth) e cor DarkGray |
| Título do tipoEspaco | Texto plano | Nome do tipo em DarkCyan, centralizado, letras maiúsculas |
| Divisória horizontal | ─ simples | ─ em toda a largura (Console.WindowWidth) e cor DarkGray |
| **ID** | Sequência contínua global | Reiniciada por grupo (começa sempre em 1) ✅ cor normal |
| **tipoEspaco** | Texto plano | Cyan, letras maiúsculas |
| **nomeFantasia** | Exibido junto | ID em Gray, tipoEspaco em Cyan, nomeFantasia em cor padrão |
| **Capacidade** | Texto plano | Em DarkGray, com plural inteligente |
| **Detalhes (quartos/suítes)** | Básico | Texto estruturado: X quartos, sendo Y suítes |
| **Valor da Diária** | Texto direto | Em DarkGray + valor formatado :C |
| **Status** | Padrão | DarkGray seguido por Livre (verde)/Ocupado (azul) |
| Separador entre espaços | Linha simples | Duplo espaçamento visual |
| Id, tipoEspaco e nomeFantasia na mesma linha Exemplo: 1- APARTAMENTO Temático Portugal | | |
| ────────────────────────────────────────────────────────────  CABANA  ────────────────────────────────────────────────────────────  1 - CABANA Pardal (ID: 07)  Capacidade: 2 pessoas  1 quartos, sendo 1 suíte  Valor da Diária: R$ 190,00  Status: Livre  📌 O ID: 07 aqui é o **identificador interno único do sistema**, enquanto o 1 - CABANA Pardal é a **exibição ordenada** para o usuário dentro do grupo CABANA |  |  |

──────────────────────────────────────────────────────────── (cor DarkGray)

APARTAMENTO (tipoEspaco DarkCyan)

──────────────────────────────────────────────────────────── (cor DarkGray)

1 - (ID Gray) APARTAMENTO (tipoEspaco Cyan) do Sol(nomeFantasia cor normal)

Capacidade: (DarkGray) 5 pessoas

2 quartos, sendo {quantidadeSuites} 1suítes (cor normal)

Valor da Diária: (DarkGray) R$ 300,00(cor normal)

Status: (DarkGray) Livre (cor normal)

Espaço

espaço

──────────────────────────────────────────────────────────── (cor DarkGray)

CASA (tipoEspaco DarkCyan)

──────────────────────────────────────────────────────────── (cor DarkGray)

1 - (ID Gray) CASA (tipoEspaco Cyan) Americana(nomeFantasia cor normal)

Capacidade: (DarkGray) 3 pessoas

1 quartos, sendo {quantidadeSuites} 0 suítes (cor normal)

Valor da Diária: (DarkGray) R$ 200,00(cor normal)

Status: (DarkGray) Livre (cor normal)

2 - (ID Gray) CASA (tipoEspaco Cyan) italiana(nomeFantasia cor normal)

Capacidade: (DarkGray) 5 pessoas

3 quartos, sendo {quantidadeSuites} 2 suítes (cor normal)

Valor da Diária: (DarkGray) R$ 1.600,89(cor normal)

Status: (DarkGray) Livre (cor normal)

Lembrando que todo tipoEspaco começa a contar a id por 1 (nunca por 0 ou por contagem de espaço

Assim que entra na tela:

cabecalhoMenus

espaço

1 - Criar

2 - Listar

3 - Cancelar

4 – Excluir

espaço

Escolha sua opção da lista vertical: Entrada do usuario (Cyan)

espaço

\_\_\_\_\_\_\_\_\_\_ (toda tela reta continua bem fina quase da cor da tela, darkGray centralizado na tela)\_\_\_\_\_

espaço

Digite aqui sua opção do rodapé (8, 9 ou 0 ?) Entrada do usuario (Cyan centralizado na tela)

Digite aqui sua opção. 8.Voltar 9.Menu principal 0.Encerrar (Gray centralizado na tela)

Espaço

Assim que entra na tela:

cabecalhoMenus

espaço

📋 Listando todas as reservas cadastradas...

(Conteúdo da listagem aqui)

Escolha sua opção: Entrada do usuário(Cyan)

espaço

\_\_\_\_\_\_\_\_\_\_ (toda tela reta continua bem fina quase da cor da tela, darkGray?)\_\_\_\_\_

Digite aqui sua opção. 8, 9 ou 0 ? Entrada do usuario (darkGray)

espaço

8.Voltar 9.Menu principal 0.Encerrar (cyan)

Espaço

### NavegacaoAtual

* Registra a tela anterior com Registrar(() => ...)
* Permite retornar ao ponto exato com Voltar()
* Limpa a pilha para início novo com Limpar()

📌 Ideal pra métodos como:

* SelecionarEspaco()
* SelecionarMultiplosClientes()
* ListarEspacosGlobal()
* CentralResumoFinanceiro

Sem cabecalho, fora de padrão

CRIAR RESERVA com fluxo interativo

1-CRIAR RESERVA

- O sistema busca os espaços disponíveis por listagem completa de espaços, ou por id, ou tipoEspaco, ou nomeFantasia, ou números de quartos

- - Vincula espaço (espaço é obrigatório ter no sistema para criarReserva)

- não limpa tela

-- Retorna para CriarReserva

- Perguntar se deseja cadastrar clientes no momento

Se Sim: pede nome completo e telefone

Tenta localizar cliente no sistema:

Se encontrado: vincula

Se não: leva ao CadastrarCliente(), cadastra e vincula

Retorna automaticamente para CriarReserva() com dados atualizados e segue fluxo

Se não: exibe mensagem: Ok, vincule os hóspedes depois

Segue o fluxo de criarResereva

- apresenta a tela com os dados inseridos até o momento

- pede quantidade de dias estimados para hospedagem (dias de reserva podem ser prolongados, caso em algum momento o hospede queira ficar mais, ou pode ser antecipada a saída, caso hospede queira sair antes, então a conta financeira é feita a cada consulta com preço estimado até o cliente fazer checout)

- Faz o calculo financeiro estimado

- conclui criação da reserva

- limpa tela

- exibe o resumo da reserva com todos os dados, hora checkin, espaço, dias estimados calculo financeiro estimado até o momento, hospedes (caso tenham sido incluídos o momento da reserva, se não, também exibir a mensagem: vincule os hóspedes depois)

- exibe mensagem: Sucesso !!! Reserva criada

- exibe menurodape

Obs: lembrando das regras do sistema para visual e experiencia do usuário, como: cabecalhoMenus, menurodape no final, e durante o processo todo, null não é aceitável, permitir o usuário digitar novamente, 3enters, valores inválidos com mensagem amigável par digitar novamente (ver método adequados para o momento)

* Onde usar NavegacaoAtual.Registrar(...) (quando há navegação reversa)
* ✅ Onde usar MenuRodapeGlobal.ExibirRodapeGlobal(...) (após métodos globais)
* ✅ Onde manter MenuRodape.ExibirRodape(...) (após operações locais)

## Objetivo

## Recap: quando usar Registrar(...) em cada lugar

| **Local** | **Onde registrar** |
| --- | --- |
| ExibirMenuReservas() | Registrar volta ao principal |
| CriarReserva() | Registrar volta ao reservas |
| CancelarReserva() | Registrar volta ao reservas |
| ExcluirCadastroReserva() | Registrar volta ao reservas |
| EditarReserva() | Registrar volta ao reservas |

Você criou o método ListarEspacosGlobal() que tem rodapé global e permite **voltar** para o ponto que o chamou — como CriarReserva(). Logo, sempre que CriarReserva() **invoca um método com rodapé global**, ela deve **registrar-se como tela anterior** via:

csharp

NavegacaoAtual.Registrar(() => CriarReserva());

Cria a reserva com o espaço e dias Precisa decidir para onde ir: Se veio de criarReserva, vincula e volta para criarReserva. Se veio de outra pagina e quer vincular reserva é porque quer criar uma nova, então seleciona o espaço, vai para CriarReserva iniciar a criação já com a reservaSelecionada. Mas pode ter vindo de qualquer submenu: MenuEspacos, MenuReservas, SubmenuClientes etc., se veiod e criarReserva, precisa selecionar e retornar para continuar a reserva. se veio de cancelar, editar reserva ou excluir reserva, também precisa selecionar e retornar. se veio de outras paginas e selecionou, precisa escolher para onde ir, ou vai para criarReserva, ou para editar, ou cancelar, ou excluir

## Ações principais do sistema

| **Ação** | **Finalidade** | **Origem esperada** |
| --- | --- | --- |
| **Cadastrar** | Criar novo cliente ou espaço | MenuClientes / MenuEspacos |
| **Editar** | Alterar dados já existentes (cliente ou reserva) | Submenus de edição |
| **Cancelar** | Cancelar uma reserva ativa antes da saída | MenuReservas / SubmenuCancelar |
| **Excluir** | Remover completamente um cadastro (reserva ou espaço) | SubmenuExcluir |
| **Criar Reserva** | Iniciar reserva nova associando cliente e espaço | MenuReservas |
| **Vincular** | Associar cliente ou espaço a uma reserva em andamento | CriarReserva / VincularCliente |
| **Listar** | Visualizar dados existentes (clientes, reservas, espaços) | Qualquer menu |

## Todas essas ações devem:

* **Registrar sua origem com** NavegacaoAtual.Registrar("Origem")
* **Chamar os métodos/telas passando** "origemChamada"
* **Finalizar com rodapé inteligente:** MenuRodape.ExibirRodape(...) **ou** NavegacaoAtual.Voltar()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Arquivo analisado** | **Conteúdo principal** | | --- | --- | | EspacoControlador.cs | Controlador de lógica para espaços | | ReservaControladora.cs | Controle de verificação de status de espaços | | Cliente.cs | Modelo completo com derivação e construtores | | EspacoHospedagem.cs | Modelo de espaço com lógica e status padrão | | Reserva.cs | Modelo da reserva, vínculo e cálculo financeiro | | ServicoClientes.cs | Salvamento de clientes em arquivo .txt | | ServicoEspacos.cs (a excluir) | Exibição antiga dos espaços | | CentralDeNavegacao.cs | Classe parcial para reinício ou encerramento | | NavegacaoAtual.cs | Controle de histórico de telas (confirmado em fala) | | ListarEspacosGlobal(...) | Tela dinâmica que consome espaço + origem | |

## Caminhos possíveis até ListarEspacosGlobal(...)

| **Origem (origemChamada)** | | **Finalidade do fluxo** | **Destino após escolher espaço** | |
| --- | --- | --- | --- | --- |
| "CriarReserva" | | Criar reserva já iniciada, continuar após escolher espaço | CriarReservaControlador.ContinuarCriacao(...) | |
| "MenuReservas" | | Usuário quer iniciar nova reserva | CriarReservaControlador.IniciarComEspacoVinculado(...) | |
| "MenuEspacos" | | Veio explorar espaços, pode decidir criar reserva | CriarReservaControlador.IniciarComEspacoVinculado(...) | |
| "SubmenuClientes" | | Veio de vínculo cliente → espaço → reserva | CriarReservaControlador.IniciarComEspacoVinculado(...) | |
| "EditarReserva" | | Está substituindo o espaço de uma reserva existente | EditarReservaControlador.ContinuarEdicao(...) | |
| "CancelarReserva" | | Verifica o espaço antes de cancelar reserva | CancelarReservaControlador.ContinuarCancelamento(...) | |
| "ExcluirReserva" | | Confirma o espaço da reserva antes de excluir | ExcluirReservaControlador.ContinuarExclusao(...) | |
| "ListarCadastroReservas" | | Vindo da listagem e quer manipular reserva vinculada | Redireciona conforme ação desejada (editar/cancelar etc.) | |
| "MenuPrincipal" | | Navegação genérica ou via rodapé | CriarReservaControlador.IniciarComEspacoVincul | |
| **Classe** | **Foco principal** | | |
| ReservaControladora | Gerenciar reservas já feitas (listar, excluir, status) | | |
| CriarReservaControlador | Guiar o fluxo de criação de reserva (iniciar ou continuar) | | |

CASA

1. CASA Colinas
2. CASA Jardins
3. CASA Torres

APARTAMENTO

1. APARTAMENTO Esperança
2. APARTAMENTO Boa ventura

CHALÉ

1. CHALÉ Horizonte
2. CHALÉ Pescaria

Erros nos testes

1. em status espaços, podemos dividir com uma linha fina e pular uma linha entre o titulo e os espaços.
2. menu reservas sem cabecalho e fora do padrão.
3. em cadastro de reserva, lsitagemd e espaços fora de padrão e tods umerados e não estão numerados por tipoEspaco, então manter o padrão utilizao em status.
4. em cadastrar hospedea reserva, estava digitando e derepente a tela sumiu e voltou para menu, sem dar continuidade, logo depois de telefone, na segunda tentativa não errei na digitação e cliente cadastrado com suceso, mas não voltou para cadastro de reserva para continuar o cadastrod a reserva, e em listagem de reserva, nenhum cadastro e nehum hospede vinculado

## **Fluxo ideal da reserva**

| **Etapa** | **Ação Recomendada** |
| --- | --- |
| Cabeçalho | CabecalhoMenus("Cadastro de Reservas")  Informações: 10% de desconto a partir de 10 dias  Ultrapassado o tempo da diária, taxa de acréscimo de R$ 5,00 por hora |
| Escolha do espaço | Mostrar e manter dados visíveis do espaço |
| Tempo de hospedagem | Entrada com controle emocional |
| Cliente(s) | Exibir lista, buscar por nome, se existir vincular a reserva e se não existir → cadastrar, voltar para Cadastro da reserva e vincular |
| Check-in e duração | Validar datas, / duração pode ser estipulada, mas pode ser prolonga, calcular check-out |
| Confirmação | Mostrar resumo: espaço, clientes, data de checkin e data estimada para saída que pode ser alterada, |
| Descontos | Mensagem: valores atuais: valor da diária , ValorSemDesconto , DescontoAplicado, taxaDeAcrescimo, ValorComDesconto |
| Finalização | "Reserva criada com sucesso" + MenuRodape |
| Rodapé | Navegação clara após criar ou cancelar |

Cadastrar cliente: o que precisamos mudar e ou melhorar. vamos a sequencia de cadastar cliente: impar tela, ter cabecalhomenus, campos usuario incluir os dados (unico obrigatorio nome completo, restante opcionais) se apertar enter passa para o campo seguinte. usar metodo para experiencia usuario, se digitar enter 3 vezes, pegunta se quer sair (S/N), exibir menu vertical: 1- Criar reserva, 2- reservasAtivas 3- lista de clientes 4- Vicular reserva e também o menurodape

1. cadastro de reserva:

- menucbecalho

- informações sobre >10 dias e acrecimo 5,00

- exibir lista de espaços

- perguntar dias da hospedagem

- perguntar se quer cadastrar hospedes agora

- Se S,

-pedir nome completo do hospede

-limpar tela, buscar no cadastro de clientes:

se tiver, vincular a reserva, exibir mensagem clientea dicionado, limpar tela, e voltar para a pagina de cadastro de reserva.

- Se não, limpar tela, ir para cadastrarCliente, cadastrar e vincular a reserva, e ao finalizar exibir mensagem cliente cadastrado com sucesso e voltar para o cadastro de reserva

- limpar tela, exibir informações da reserva: data checkin, valor da diária, valorSemDesconto, valorDoSesconto(0 se não aplicado), valorComDesconto até o momento, nome dos hospedes( se já tiverem sido cadstrados, se não, mensagem: cadastre os clientes depois), após finalizar, exibir menurodape

| **ção** | **Atalho** |
| --- | --- |
| Criar múltiplos cursores | Alt + Clique (com o mouse) |
| Selecionar ocorrências | Ctrl + D |
| Selecionar todas ocorrências | Ctrl + Shift + L |
| Mover cursor abaixo/acima | Ctrl + Alt + ↓ ou ↑ |

public class ResumoFinanceiro

{

public decimal ValorDiaria { get; set; }

public decimal ValorSemDesconto { get; set; }

public decimal ValorDoDesconto { get; set; }

public decimal ValorComDesconto { get; set; }

public decimal ValorTaxaExtra { get; set; }

public decimal ValorTotalEstimado { get; set; }

}

## Sugestões de alinhamento extra

* 📦 **Todos os métodos** devem receber os parâmetros necessários (espacos, clientes, reservas, reservaSelecionada) para manter fluidez no fluxo entre menus e ações.
* ✨ MiniMenus ao final de cada operação — seja sucesso, falha ou cancelamento.
* 🚫 Nenhum método deve usar ref ou try/catch desnecessário — apenas onde a lógica emocional exige.
* 🧾 Mensagens com contexto: "Menu – Espaços de Hospedagem" em todas as saídas e retornos.

## Estrutura atual do projeto

| **Pasta** | **Conteúdo principal** | **Namespace ideal sugerido** |
| --- | --- | --- |
| Comprovantes | Geração de arquivos e comprovantes | DesafioProjetoHospedagem.Comprovantes |
| Controladoras | Lógica de controle de entidades (Cliente, Reserva...) | DesafioProjetoHospedagem.Controladoras |
| Menus | Interfaces de navegação e exibição | DesafioProjetoHospedagem.Menus |
| Models | Classes de dados (Cliente, Reserva...) | DesafioProjetoHospedagem.Models |
| Servicos | Serviços auxiliares ou externos | DesafioProjetoHospedagem.Servicos |
| Utils | Helpers e utilitários gerais | DesafioProjetoHospedagem.Utils |

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

## 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

|  |  |
| --- | --- |
| 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?

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 |

BuscarEspacoPorFiltro: esse bloco a finalidade é encontrar espaços livres com determinadas caracteristicas para vincular a reserva. o que é preciso ter no bloco,para que isso aconteça? busca espaços livres com determinadas caracteristicas, se encontrar, vincula a reserva e voltar para criarReserva (já em andamento, pois a primeira coisa é vincular espaço), se não encontrar, exibe mensagem, deseja visualizar lista completa de espaços? se sim, exibe lista com espaços livres e abaixo a lista de espaços ocupados com todas as informações checkin, previsão de checkout. e abaixo o menurodape, usando metodos novos, nunca com pressionar enter para retorno nem voltar abruptament,sempre usar metodo amigável

## Mapa estratégico do projeto - onde fazer chamadas com :

## NavegacaoAtual.Registrar(() => Exibir nomedometodo());

Exemplo:

// no menu de reservas

public static void ExibirCriarReserva()

{

// ... lógica de reserva

NavegacaoAtual.Registrar(() => ExibirCriarReserva());

ListarEspacosGlobal();

}

| **Módulo** | **Método ou Tela** | **Chamado por...** | **Aplicar MenuRodapeGlobal?** |
| --- | --- | --- | --- |
| **Espaços** | ListarEspacosGlobal() | Cadastro, Reserva, Filtro, Relatório... | ✅ Sim |
|  | BuscarEspacoPorFiltro() | Reserva, Relatório, Filtros | ✅ Sim |
| **Clientes** | ListarClientesGlobal() | Reserva, Relatórios, Vínculo | ✅ Sim |
|  | BuscarClientePorFiltro() | Reserva, Relatório | ✅ Sim |
| **Relatórios** | ExibirRelatorioPorPeriodo() | Relatórios gerais | ✅ Sim |
|  | ExibirResumoFinanceiro() | Menu financeiro | ✅ Sim |
| **Comprovantes** | ExibirMenuComprovantes() | Pós-reserva, histórico | ✅ Sim |
| **Reservas** | EscolherEspacoPorBusca() | Criar reserva | ✅ Sim |
|  | EscolherClientePorBusca() | Criar reserva | ✅ Sim |
| **Filtros Avançados** | ExibirFiltrosCustomizados() | Diversas telas | ✅ Sim |

## O que você NÃO precisa mexer

* MenuPrincipal
* MenuReservas.ExibirMenuReservas()
* MenuEspacos.ExibirMenuEspacos()
* MenuClientes.ExibirMenuClientes()
* Qualquer submenu que já usa MenuRodape com os Action fixos

talvez a resposta tenha que ser diferente:

busca feita: tipos espaço: apartamento, quantidade mínima de quartos: 2, quantidade mínima de suítes 1

Exemplo de saída: lista com todos os tipoEspaco com no mínimo (>=) das características pedidas

Exemplo da lista exibida com todos os tipoEspaco que contem >= das características pedidas:

CHALÉS

3-CHALÉ Fantasia: informações completas, e tem de 2 quartos pra mais e tem 1 suite pra mais

6-CHALÉ Oclos: todas as informações, e tem de 2 quartos pra mais e tem 1 suite pra mais

CABANAS

1-CABANA Angra: todas as informações, e tem de 2 quartos pra mais e tem 1 suite pra mais

13- CABANA Pontes: todas as informações, e tem de 2 quartos pra mais e tem 1 suite pra mais

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;

                }

            }

        }

    }

}

\*/