Crud N:N

Crie um projeto MVC Core chamado VendasMvc e uma Base de dados no Postgres.

Adicione os pacotes necessários

Instale o provedor Npgsql, ferramentas EF e o scaffolder de controllers/views.

Usando o console de gerenciador de pacotes execute:

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet add package Microsoft.EntityFrameworkCore.Tools

dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design

dotnet tool install -g dotnet-aspnet-codegenerator

**Configure a *connection string***

Edite **appsettings.json** (na raiz do projeto):

{

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Database=CRUDNN;Username=postgres;Password=5432;Include Error Detail=true"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Configure o EF Core no Program.cs**

Edite **Program.cs**:

using Microsoft.EntityFrameworkCore;

using System;

using VendasMvc.Data;

AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var connStr = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<AppDbContext>(opt =>

opt.UseNpgsql(connStr));

// (Opcional) mapear DateTime antigo do Npgsql se você migrar projetos antigos:

// AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);

var app = builder.Build();

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

**7) Crie as Models (com N:N entre Venda e Produto)**

Crie a pasta **Models** e os arquivos:

**Cliente.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.Models;

public class Cliente

{

public int Id { get; set; }

[Required, StringLength(120)]

public string Nome { get; set; } = string.Empty;

[EmailAddress, StringLength(120)]

public string? Email { get; set; }

[StringLength(20)]

public string? Telefone { get; set; }

public ICollection<Venda>? Vendas { get; set; }

}

**Produto.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.Models;

public class Produto

{

public int Id { get; set; }

[Required, StringLength(120)]

public string Nome { get; set; } = string.Empty;

[Range(0, 999999999)]

public decimal Preco { get; set; }

[Range(0, int.MaxValue)]

public int Estoque { get; set; }

public ICollection<VendaProduto>? Vendas { get; set; }

}

**Venda.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.Models;

public class Venda

{

public int Id { get; set; }

// UTC por padrão

public DateTime Data { get; set; } = DateTime.UtcNow;

public int ClienteId { get; set; }

public Cliente? Cliente { get; set; }

public ICollection<VendaProduto> Itens { get; set; } = new List<VendaProduto>();

public decimal Total => Itens?.Sum(i => i.PrecoUnitario \* i.Quantidade) ?? 0m;

}

**VendaProduto.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.Models;

public class VendaProduto

{

public int VendaId { get; set; }

public Venda? Venda { get; set; }

public int ProdutoId { get; set; }

public Produto? Produto { get; set; }

[Range(1, 999999)]

public int Quantidade { get; set; }

[Range(0, 999999999)]

public decimal PrecoUnitario { get; set; } // captura o preço na data da venda

}

**Crie o DbContext com o mapeamento fluente**

Crie Data/AppDbContext.cs:

using Microsoft.EntityFrameworkCore;

using VendasMvc.Models;

namespace VendasMvc.Data;

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<Cliente> Clientes => Set<Cliente>();

public DbSet<Produto> Produtos => Set<Produto>();

public DbSet<Venda> Vendas => Set<Venda>();

public DbSet<VendaProduto> VendasProdutos => Set<VendaProduto>();

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

// Chave composta da tabela de junção

modelBuilder.Entity<VendaProduto>()

.HasKey(vp => new { vp.VendaId, vp.ProdutoId });

modelBuilder.Entity<VendaProduto>()

.HasOne(vp => vp.Venda)

.WithMany(v => v.Itens)

.HasForeignKey(vp => vp.VendaId)

.OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<VendaProduto>()

.HasOne(vp => vp.Produto)

.WithMany(p => p.Vendas)

.HasForeignKey(vp => vp.ProdutoId)

.OnDelete(DeleteBehavior.Restrict); // evitar deletar produto se estiver em vendas

// Precisão de decimais para preços

modelBuilder.Entity<Produto>()

.Property(p => p.Preco)

.HasPrecision(18, 2);

modelBuilder.Entity<VendaProduto>()

.Property(vp => vp.PrecoUnitario)

.HasPrecision(18, 2);

}

}

**Migrations e criação do banco**

Add-Migration InitialCreate

Update-Database

**Scaffold de CRUD para Cliente e Produto**

Você pode gerar Controllers + Razor Views automáticos.

**Via Visual Studio (UI)**

* **Clique direito** na pasta **Controllers** → **Add** → **New Scaffolded Item…** → **MVC Controller with views, using EF**.
  + **Model class:** Cliente
  + **Data context class:** AppDbContext
  + **Controller name:** ClientesController

Repita para **Produto** → ProdutosController.

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

**Venda: ViewModel + Controller (N:N com carga)**

Como Venda tem itens (N:N) com **quantidade e preço por item**, é melhor usar um **ViewModel** para *Create/Edit*.

**Crie os ViewModels**

Crie a pasta **ViewModels**:

**ViewModels/VendaItemVM.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.ViewModels;

public class VendaItemVM

{

public int ProdutoId { get; set; }

public string ProdutoNome { get; set; } = string.Empty;

public decimal PrecoAtual { get; set; }

[Range(0, 999999)]

public int Quantidade { get; set; } // 0 = não incluir

}

**Crie ViewModels/VendaFormVM.cs**

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.ViewModels;

public class VendaFormVM

{

[Required]

public int ClienteId { get; set; }

public List<VendaItemVM> Itens { get; set; } = new();

}

**OBS:**

Por que ViewModels? Eles servem para agrupar dados da tela (select de cliente, lista de produtos com quantidades) sem expor diretamente suas entidades de domínio. Também evitam *overposting* e simplificam a validação.

**Crie o VendasController**

Crie **Controllers/VendasController.cs**:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using VendasMvc.Data;

using VendasMvc.Models;

using VendasMvc.ViewModels;

namespace VendasMvc.Controllers;

public class VendasController : Controller

{

private readonly AppDbContext \_context;

public VendasController(AppDbContext context)

{

\_context = context;

}

// GET: Vendas

public async Task<IActionResult> Index()

{

var vendas = await \_context.Vendas

.Include(v => v.Cliente)

.Include(v => v.Itens).ThenInclude(i => i.Produto)

.AsNoTracking()

.OrderByDescending(v => v.Data)

.ToListAsync();

return View(vendas);

}

// GET: Vendas/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null) return NotFound();

var venda = await \_context.Vendas

.Include(v => v.Cliente)

.Include(v => v.Itens).ThenInclude(i => i.Produto)

.FirstOrDefaultAsync(v => v.Id == id);

if (venda == null) return NotFound();

return View(venda);

}

// GET: Vendas/Create

public async Task<IActionResult> Create()

{

ViewBag.Clientes = await \_context.Clientes

.AsNoTracking()

.OrderBy(c => c.Nome)

.ToListAsync();

var produtos = await \_context.Produtos

.AsNoTracking()

.OrderBy(p => p.Nome)

.ToListAsync();

var vm = new VendaFormVM

{

Itens = produtos.Select(p => new VendaItemVM

{

ProdutoId = p.Id,

ProdutoNome = p.Nome,

PrecoAtual = p.Preco,

Quantidade = 0

}).ToList()

};

return View(vm);

}

// POST: Vendas/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(VendaFormVM vm)

{

if (!ModelState.IsValid)

{

ViewBag.Clientes = await \_context.Clientes.AsNoTracking().OrderBy(c => c.Nome).ToListAsync();

return View(vm);

}

// Carrega preços atuais dos produtos selecionados

var idsSelecionados = vm.Itens.Where(i => i.Quantidade > 0).Select(i => i.ProdutoId).ToList();

var produtos = await \_context.Produtos

.Where(p => idsSelecionados.Contains(p.Id))

.ToDictionaryAsync(p => p.Id);

if (!idsSelecionados.Any())

{

ModelState.AddModelError("", "Selecione pelo menos um produto com quantidade > 0.");

ViewBag.Clientes = await \_context.Clientes.AsNoTracking().OrderBy(c => c.Nome).ToListAsync();

return View(vm);

}

var venda = new Venda

{

ClienteId = vm.ClienteId,

Data = DateTime.Now

};

foreach (var item in vm.Itens.Where(i => i.Quantidade > 0))

{

var produto = produtos[item.ProdutoId];

venda.Itens.Add(new VendaProduto

{

ProdutoId = produto.Id,

Quantidade = item.Quantidade,

PrecoUnitario = produto.Preco // captura preço na venda

});

// (Opcional) baixa de estoque:

// produto.Estoque -= item.Quantidade;

}

\_context.Vendas.Add(venda);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

// GET: Vendas/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null) return NotFound();

var venda = await \_context.Vendas

.Include(v => v.Itens)

.FirstOrDefaultAsync(v => v.Id == id);

if (venda == null) return NotFound();

ViewBag.Clientes = await \_context.Clientes.AsNoTracking().OrderBy(c => c.Nome).ToListAsync();

var produtos = await \_context.Produtos.AsNoTracking().OrderBy(p => p.Nome).ToListAsync();

var vm = new VendaFormVM

{

ClienteId = venda.ClienteId,

Itens = produtos.Select(p =>

{

var existente = venda.Itens.FirstOrDefault(i => i.ProdutoId == p.Id);

return new VendaItemVM

{

ProdutoId = p.Id,

ProdutoNome = p.Nome,

PrecoAtual = p.Preco,

Quantidade = existente?.Quantidade ?? 0

};

}).ToList()

};

ViewBag.VendaId = venda.Id;

return View(vm);

}

// POST: Vendas/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, VendaFormVM vm)

{

var venda = await \_context.Vendas

.Include(v => v.Itens)

.FirstOrDefaultAsync(v => v.Id == id);

if (venda == null) return NotFound();

if (!ModelState.IsValid)

{

ViewBag.Clientes = await \_context.Clientes.AsNoTracking().OrderBy(c => c.Nome).ToListAsync();

ViewBag.VendaId = id;

return View(vm);

}

venda.ClienteId = vm.ClienteId;

// Remove todos os itens e recria (estratégia simples e segura)

\_context.VendasProdutos.RemoveRange(venda.Itens);

venda.Itens.Clear();

var idsSelecionados = vm.Itens.Where(i => i.Quantidade > 0).Select(i => i.ProdutoId).ToList();

var produtos = await \_context.Produtos

.Where(p => idsSelecionados.Contains(p.Id))

.ToDictionaryAsync(p => p.Id);

foreach (var item in vm.Itens.Where(i => i.Quantidade > 0))

{

var produto = produtos[item.ProdutoId];

venda.Itens.Add(new VendaProduto

{

VendaId = venda.Id,

ProdutoId = produto.Id,

Quantidade = item.Quantidade,

PrecoUnitario = produto.Preco

});

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

// GET: Vendas/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var venda = await \_context.Vendas

.Include(v => v.Cliente)

.Include(v => v.Itens).ThenInclude(i => i.Produto)

.FirstOrDefaultAsync(m => m.Id == id);

if (venda == null) return NotFound();

return View(venda);

}

// POST: Vendas/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var venda = await \_context.Vendas

.Include(v => v.Itens)

.FirstOrDefaultAsync(v => v.Id == id);

if (venda != null)

{

\_context.VendasProdutos.RemoveRange(venda.Itens);

\_context.Vendas.Remove(venda);

await \_context.SaveChangesAsync();

}

return RedirectToAction(nameof(Index));

}

}

**Views de Venda (Razor)**

Crie a pasta **Views/Vendas** com:

**Views/Vendas/Index.cshtml** adicione o código:

@model IEnumerable<VendasMvc.Models.Venda>

@{

ViewData["Title"] = "Vendas";

}

<h1>Vendas</h1>

<p>

<**a** **asp-action**="Create" class="btn btn-primary">Nova Venda</**a**>

</p>

<table class="table table-striped">

<thead>

<tr>

<th>Id</th>

<th>Data</th>

<th>Cliente</th>

<th>Total</th>

<th>Itens</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var v in Model)

{

<tr>

<td>@v.Id</td>

<td>@v.Data.ToString("dd/MM/yyyy HH:mm")</td>

<td>@v.Cliente?.Nome</td>

<td>@v.Total.ToString("C")</td>

<td>@v.Itens.Count()</td>

<td class="text-end">

<**a** **asp-action**="Details" **asp-route-id**="@v.Id" class="btn btn-sm btn-info">Detalhes</**a**>

<**a** **asp-action**="Edit" **asp-route-id**="@v.Id" class="btn btn-sm btn-warning">Editar</**a**>

<**a** **asp-action**="Delete" **asp-route-id**="@v.Id" class="btn btn-sm btn-danger">Excluir</**a**>

</td>

</tr>

}

</tbody>

</table>

**Crie Views/Vendas/Create.cshtml**

@model VendasMvc.ViewModels.VendaFormVM

@using VendasMvc.Models

@{

ViewData["Title"] = "Nova Venda";

var clientes = (IEnumerable<Cliente>)ViewBag.Clientes;

}

<h1>Nova Venda</h1>

<**form** **asp-action**="Create" method="post">

<div class="mb-3">

<label class="form-label">Cliente</label>

<**select** **asp-for**="ClienteId" class="form-select" required>

<**option** **value**="">-- selecione --</**option**>

@foreach (var c in clientes)

{

<**option** **value**="@c.Id">@c.Nome</**option**>

}

</**select**>

<**span** **asp-validation-for**="ClienteId" class="text-danger"></**span**>

</div>

<h4>Produtos</h4>

<p class="text-muted">Informe a quantidade desejada (0 = não incluir)</p>

<table class="table table-bordered align-middle">

<thead>

<tr>

<th>Produto</th>

<th class="text-end">Preço Atual</th>

<th class="text-end">Quantidade</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < Model.Itens.Count; i++)

{

<tr>

<td>

@Model.Itens[i].ProdutoNome

<**input** **type**="hidden" **asp-for**="Itens[@i].ProdutoId" />

<**input** **type**="hidden" **asp-for**="Itens[@i].ProdutoNome" />

<**input** **type**="hidden" **asp-for**="Itens[@i].PrecoAtual" />

</td>

<td class="text-end">@Model.Itens[i].PrecoAtual.ToString("C")</td>

<td class="text-end" style="max-width:140px;">

<**input** **asp-for**="Itens[@i].Quantidade" class="form-control text-end" min="0" />

<**span** **asp-validation-for**="Itens[@i].Quantidade" class="text-danger"></**span**>

</td>

</tr>

}

</tbody>

</table>

<button type="submit" class="btn btn-primary">Salvar</button>

<**a** **asp-action**="Index" class="btn btn-secondary">Cancelar</**a**>

</**form**>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**Crie Views/Vendas/Edit.cshtml**

@model VendasMvc.ViewModels.VendaFormVM

@using VendasMvc.Models

@{

ViewData["Title"] = "Editar Venda";

var clientes = (IEnumerable<Cliente>)ViewBag.Clientes;

var vendaId = (int)ViewBag.VendaId;

}

<h1>Editar Venda #@vendaId</h1>

<form asp-action="Edit" method="post" asp-route-id="@vendaId">

<div class="mb-3">

<label class="form-label">Cliente</label>

<select asp-for="ClienteId" class="form-select" required>

<option value="">-- selecione --</option>

@foreach (var c in clientes)

{

<option value="@c.Id" selected="@(c.Id == Model.ClienteId)">

@c.Nome

</option>

}

</select>

<span asp-validation-for="ClienteId" class="text-danger"></span>

</div>

<h4>Produtos</h4>

<p class="text-muted">Ajuste as quantidades (0 = remover)</p>

<table class="table table-bordered align-middle">

<thead>

<tr>

<th>Produto</th>

<th class="text-end">Preço Atual</th>

<th class="text-end">Quantidade</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < Model.Itens.Count; i++)

{

<tr>

<td>

@Model.Itens[i].ProdutoNome

<input type="hidden" asp-for="Itens[@i].ProdutoId" />

<input type="hidden" asp-for="Itens[@i].ProdutoNome" />

<input type="hidden" asp-for="Itens[@i].PrecoAtual" />

</td>

<td class="text-end">@Model.Itens[i].PrecoAtual.ToString("C")</td>

<td class="text-end" style="max-width:140px;">

<input asp-for="Itens[@i].Quantidade" class="form-control text-end" min="0" />

<span asp-validation-for="Itens[@i].Quantidade" class="text-danger"></span>

</td>

</tr>

}

</tbody>

</table>

<button type="submit" class="btn btn-primary">Salvar alterações</button>

<a asp-action="Index" class="btn btn-secondary">Cancelar</a>

</form>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**Crie Views/Vendas/Delete.cshtml**

@model VendasMvc.Models.Venda

@{

ViewData["Title"] = "Excluir Venda";

}

<h1>Excluir Venda</h1>

<h3>Você tem certeza que deseja excluir esta venda?</h3>

<div>

<h4>Venda #@Model.Id</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">Cliente</dt>

<dd class="col-sm-10">@Model.Cliente?.Nome</dd>

<dt class="col-sm-2">Data</dt>

<dd class="col-sm-10">@Model.Data.ToString("dd/MM/yyyy HH:mm")</dd>

<dt class="col-sm-2">Total</dt>

<dd class="col-sm-10">@Model.Total.ToString("C")</dd>

</dl>

<h4>Itens da Venda</h4>

<table class="table table-striped">

<thead>

<tr>

<th>Produto</th>

<th class="text-end">Preço Unit.</th>

<th class="text-end">Qtd</th>

<th class="text-end">Subtotal</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.Itens)

{

<tr>

<td>@item.Produto?.Nome</td>

<td class="text-end">@item.PrecoUnitario.ToString("C")</td>

<td class="text-end">@item.Quantidade</td>

<td class="text-end">@((item.PrecoUnitario \* item.Quantidade).ToString("C"))</td>

</tr>

}

</tbody>

</table>

</div>

<**form** **asp-action**="Delete" method="post">

<**input** **type**="hidden" **asp-for**="Id" />

<button type="submit" class="btn btn-danger">Excluir</button>

<**a** **asp-action**="Index" class="btn btn-secondary">Cancelar</**a**>

</**form**>

Se ao executar o projeto der erro altere **Controller garantindo carregamento dos itens**

Para que essa view funcione, o GET: Delete no VendasController precisa carregar **Cliente** e **Itens com Produto**:

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var venda = await \_context.Vendas

.Include(v => v.Cliente)

.Include(v => v.Itens).ThenInclude(i => i.Produto)

.FirstOrDefaultAsync(m => m.Id == id);

if (venda == null) return NotFound();

return View(venda);

}

**Criando classes definitivas para corrigir o problema com os pontos**

Adicione um Model Binder que aceita ponto e vírgula

Crie Infrastructure/InvariantDecimalModelBinder.cs:

using Microsoft.AspNetCore.Mvc.ModelBinding;

using System.Globalization;

namespace VendasMvc.Infrastructure;

public class InvariantDecimalModelBinder : IModelBinder

{

public Task BindModelAsync(ModelBindingContext ctx)

{

var valueResult = ctx.ValueProvider.GetValue(ctx.ModelName);

if (valueResult == ValueProviderResult.None) return Task.CompletedTask;

ctx.ModelState.SetModelValue(ctx.ModelName, valueResult);

var raw = valueResult.FirstValue?.Trim();

if (string.IsNullOrEmpty(raw))

return Task.CompletedTask;

// Regra robusta: escolhe o último separador como decimal e remove o outro como milhar

int lastComma = raw.LastIndexOf(',');

int lastDot = raw.LastIndexOf('.');

if (lastComma > lastDot)

raw = raw.Replace(".", ""); // ponto = milhar

else if (lastDot > lastComma)

raw = raw.Replace(",", ""); // vírgula = milhar

raw = raw.Replace(',', '.'); // decimal final = ponto

if (decimal.TryParse(raw, NumberStyles.Number, CultureInfo.InvariantCulture, out var dec))

{

ctx.Result = ModelBindingResult.Success(dec);

}

else

{

ctx.ModelState.TryAddModelError(ctx.ModelName, "Valor numérico inválido.");

}

return Task.CompletedTask;

}

}

Crie **Infrastructure/InvariantDecimalModelBinderProvider.cs**:

using Microsoft.AspNetCore.Mvc.ModelBinding;

namespace VendasMvc.Infrastructure;

public class InvariantDecimalModelBinderProvider : IModelBinderProvider

{

public IModelBinder? GetBinder(ModelBinderProviderContext context)

{

var t = context.Metadata.UnderlyingOrModelType;

if (t == typeof(decimal) || t == typeof(decimal?))

return new InvariantDecimalModelBinder();

return null;

}

}

**Registre o binder (garanta que este AddControllersWithViews é o único com options)**

No Program.cs:

using Microsoft.AspNetCore.Localization;

using Microsoft.EntityFrameworkCore;

using System;

using System.Globalization;

using VendasMvc.Data;

using VendasMvc.Infrastructure;

AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews(options =>

{

// nosso binder vem primeiro

options.ModelBinderProviders.Insert(0, new InvariantDecimalModelBinderProvider());

});

var connStr = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<AppDbContext>(opt =>

opt.UseNpgsql(connStr));

// (Opcional) mapear DateTime antigo do Npgsql se você migrar projetos antigos:

// AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);

var supportedCultures = new[] { new CultureInfo("pt-BR") };

builder.Services.Configure<RequestLocalizationOptions>(opts =>

{

opts.DefaultRequestCulture = new RequestCulture("pt-BR");

opts.SupportedCultures = supportedCultures;

opts.SupportedUICultures = supportedCultures;

});

var app = builder.Build();

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

**Garanta que os campos de preço usam o binder e a view não atrapalha**

**3.1 Modelo Produto**

Confirme que é decimal e (opcional, mas útil) anote com o binder:

public class Produto

{

public int Id { get; set; }

[Required, StringLength(120)]

public string Nome { get; set; } = string.Empty;

[ModelBinder(BinderType = typeof(InvariantDecimalModelBinder))]

[Range(0, 999999999)]

public decimal Preco { get; set; }

[Range(0, int.MaxValue)]

public int Estoque { get; set; }

public ICollection<VendaProduto>? Vendas { get; set; }

}

**Inputs nas Views (Create/Edit de Produto)**

Troque para **type="text"** (evita comportamento estranho de input type="number" com localizações):

<input asp-for="Preco" class="form-control text-end" type="text" inputmode="decimal" />

<span asp-validation-for="Preco" class="text-danger"></span>

**Alterando o Layout da aplicação**

Alterando a interface inicial:

Substitua Views/Shared/\_Layout.cshtml

@{

var title = ViewData["Title"] as string ?? "VendasMvc";

}

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@title - VendasMvc</title>

<**environment** **include**="Development">

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

</**environment**>

<**environment** **exclude**="Development">

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

</**environment**>

<!-- Ícones (Bootstrap Icons) -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css">

</head>

<body class="app-body">

<div class="app-shell">

<!-- Sidebar -->

<aside class="app-sidebar">

<div class="brand">

<i class="bi bi-cart-check"></i>

<span>VendasMvc</span>

</div>

<nav class="menu">

<**a** **asp-controller**="Home" **asp-action**="Index" class="menu-item">

<i class="bi bi-speedometer2"></i> <span>Dashboard</span>

</**a**>

<div class="menu-section">Cadastros</div>

<**a** **asp-controller**="Clientes" **asp-action**="Index" class="menu-item">

<i class="bi bi-people"></i> <span>Clientes</span>

</**a**>

<**a** **asp-controller**="Produtos" **asp-action**="Index" class="menu-item">

<i class="bi bi-box-seam"></i> <span>Produtos</span>

</**a**>

<div class="menu-section">Operações</div>

<**a** **asp-controller**="Vendas" **asp-action**="Index" class="menu-item">

<i class="bi bi-receipt-cutoff"></i> <span>Vendas</span>

</**a**>

</nav>

<div class="sidebar-footer">

<**a** **asp-controller**="Home" **asp-action**="Privacy" class="menu-item small">

<i class="bi bi-shield-lock"></i> <span>Privacidade</span>

</**a**>

</div>

</aside>

<!-- Área principal -->

<div class="app-main">

<!-- Topbar -->

<header class="topbar">

<button class="burger" id="burger" title="Menu">

<i class="bi bi-list"></i>

</button>

<div class="page-title">@title</div>

<div class="topbar-right">

<div class="search">

<i class="bi bi-search"></i>

<input type="text" placeholder="Pesquisar..." />

</div>

<div class="avatar" title="Usuário">

<i class="bi bi-person-circle"></i>

</div>

</div>

</header>

<!-- Conteúdo -->

<main class="content container-fluid py-4">

@RenderBody()

</main>

<footer class="footer text-center">

<small>&copy; @DateTime.Now.Year — VendasMvc</small>

</footer>

</div>

</div>

<**environment** **include**="Development">

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

</**environment**>

<**environment** **exclude**="Development">

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

</**environment**>

@RenderSection("Scripts", required: false)

<script>

// Toggle da sidebar (mobile e desktop)

document.getElementById('burger')?.addEventListener('click', function () {

document.body.classList.toggle('sidebar-open');

});

</script>

</body>

</html>

**Estilize em wwwroot/css/site.css**

Acrescente ao **final** do seu site.css (ou substitua se preferir):

/\* ==============================

Tema do App de Vendas

============================== \*/

:root {

--bg: #0f172a; /\* slate-900 \*/

--surface: #111827; /\* gray-900 \*/

--surface-2: #0b1220; /\* custom \*/

--card: #111827;

--text: #e5e7eb; /\* gray-200 \*/

--muted: #9ca3af; /\* gray-400 \*/

--primary: #22c55e; /\* green-500 \*/

--primary-600: #16a34a;

--border: #1f2937; /\* gray-800 \*/

--shadow: 0 8px 24px rgba(0,0,0,.35);

}

html, body { height: 100%; }

body.app-body {

background: linear-gradient(180deg, var(--bg) 0%, #0b1220 100%);

color: var(--text);

}

/\* Shell \*/

.app-shell { display: grid; grid-template-columns: 260px 1fr; min-height: 100vh; }

/\* Sidebar \*/

.app-sidebar {

background: linear-gradient(180deg, var(--surface) 0%, var(--surface-2) 100%);

border-right: 1px solid var(--border);

padding: 20px 14px;

position: sticky;

top: 0;

height: 100vh;

box-shadow: var(--shadow);

}

.brand {

display: flex; align-items: center; gap: 10px;

font-weight: 700; letter-spacing: .4px; font-size: 1.25rem;

color: var(--text); margin-bottom: 18px;

}

.brand i { color: var(--primary); font-size: 1.4rem; }

.menu { display: flex; flex-direction: column; gap: 6px; }

.menu-section {

color: var(--muted); font-size: .75rem; text-transform: uppercase;

padding: 14px 10px 6px; letter-spacing: .08em;

}

.menu-item {

display: flex; align-items: center; gap: 10px;

padding: 10px 12px; border-radius: 10px; color: var(--text);

text-decoration: none; transition: .15s ease;

}

.menu-item:hover { background: rgba(255,255,255,.06); }

.menu-item i { color: var(--muted); width: 22px; text-align: center; }

.menu-item.active, .menu-item:focus { background: rgba(34,197,94,.15); }

.menu-item.small { font-size: .9rem; color: var(--muted); }

.sidebar-footer { margin-top: auto; padding-top: 8px; border-top: 1px dashed var(--border); }

/\* Main \*/

.app-main { display: flex; flex-direction: column; min-width: 0; }

/\* Topbar \*/

.topbar {

display: flex; align-items: center; justify-content: space-between;

padding: 12px 18px; border-bottom: 1px solid var(--border);

background: rgba(255,255,255,.02); backdrop-filter: blur(6px);

position: sticky; top: 0; z-index: 10;

}

.burger {

border: none; background: transparent; color: var(--text); font-size: 1.6rem;

padding: 6px 10px; border-radius: 8px;

}

.burger:hover { background: rgba(255,255,255,.06); }

.page-title { font-weight: 600; }

.topbar-right { display: flex; align-items: center; gap: 12px; }

.search {

display: flex; align-items: center; gap: 8px; color: var(--muted);

background: rgba(255,255,255,.06); border: 1px solid var(--border);

padding: 6px 10px; border-radius: 10px; min-width: 240px;

}

.search input {

background: transparent; border: none; outline: none; color: var(--text);

}

.avatar { font-size: 1.6rem; color: var(--text); }

.content .card {

background: var(--card); color: var(--text); border: 1px solid var(--border);

border-radius: 16px; box-shadow: var(--shadow);

}

.card .card-title { color: var(--muted); font-size: .9rem; text-transform: uppercase; letter-spacing: .08em; }

.card .value { font-size: 1.6rem; font-weight: 700; }

.btn-primary {

background-color: var(--primary); border-color: var(--primary);

}

.btn-primary:hover { background-color: var(--primary-600); border-color: var(--primary-600); }

.table { color: var(--text); }

.table thead th { color: var(--muted); border-color: var(--border); }

.table td, .table th { border-color: var(--border); }

/\* Footer \*/

.footer { color: var(--muted); padding: 12px 0 30px; }

/\* Responsivo (sidebar offcanvas) \*/

@media (max-width: 992px) {

.app-shell { grid-template-columns: 0 1fr; }

.app-sidebar { position: fixed; left: -260px; width: 260px; transition: left .2s ease; z-index: 50; }

body.sidebar-open .app-sidebar { left: 0; }

body.sidebar-open { overflow: hidden; }

}

**Melhorando formulário de Clientes**

Em Models/Cliente.cs

using System.ComponentModel.DataAnnotations;

namespace VendasMvc.Models;

public class Cliente

{

public int Id { get; set; }

[Display(Name = "Nome completo")]

[Required(ErrorMessage = "Informe o nome.")]

[StringLength(120)]

public string Nome { get; set; } = string.Empty;

[Display(Name = "E-mail")]

[EmailAddress(ErrorMessage = "E-mail inválido.")]

[StringLength(120)]

public string? Email { get; set; }

[Display(Name = "Telefone")]

[StringLength(20)]

[RegularExpression(@"^\(?\d{2}\)?\s?\d{4,5}-?\d{4}$",

ErrorMessage = "Use o formato (99) 99999-9999")]

public string? Telefone { get; set; }

public ICollection<Venda>? Vendas { get; set; }

}

**Controller — Controllers/ClientesController.cs** adicione **busca** no Index, e **mensagens** com TempData.

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using VendasMvc.Data;

using VendasMvc.Models;

namespace VendasMvc.Controllers

{

public class ClientesController : Controller

{

private readonly AppDbContext \_context;

public ClientesController(AppDbContext context)

{

\_context = context;

}

// helper de mensagens

private void Success(string m) => TempData["Success"] = m;

private void Error(string m) => TempData["Error"] = m;

// GET: Clientes

public async Task<IActionResult> Index(string? q)

{

ViewData["Title"] = "Clientes";

ViewData["q"] = q;

var query = \_context.Clientes.AsNoTracking();

if (!string.IsNullOrWhiteSpace(q))

{

// Postgres: ILIKE faz case-insensitive

query = query.Where(c =>

EF.Functions.ILike(c.Nome, $"%{q}%") ||

EF.Functions.ILike(c.Email ?? "", $"%{q}%") ||

EF.Functions.ILike(c.Telefone ?? "", $"%{q}%"));

}

var list = await query.OrderBy(c => c.Nome).ToListAsync();

return View(list);

}

// GET: Clientes/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null) return NotFound();

var cliente = await \_context.Clientes.FirstOrDefaultAsync(m => m.Id == id);

if (cliente == null) return NotFound();

ViewData["Title"] = "Detalhes do Cliente";

return View(cliente);

}

// GET: Clientes/Create

public IActionResult Create()

{

ViewData["Title"] = "Novo Cliente";

return View(new Cliente());

}

// POST: Clientes/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Cliente cliente)

{

if (!ModelState.IsValid) return View(cliente);

\_context.Add(cliente);

await \_context.SaveChangesAsync();

Success("Cliente cadastrado com sucesso!");

return RedirectToAction(nameof(Index));

}

// GET: Clientes/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null) return NotFound();

var cliente = await \_context.Clientes.FindAsync(id);

if (cliente == null) return NotFound();

ViewData["Title"] = "Editar Cliente";

return View(cliente);

}

// POST: Clientes/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, Cliente cliente)

{

if (id != cliente.Id) return NotFound();

if (!ModelState.IsValid) return View(cliente);

try

{

\_context.Update(cliente);

await \_context.SaveChangesAsync();

Success("Cliente atualizado com sucesso!");

return RedirectToAction(nameof(Index));

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.Clientes.Any(e => e.Id == cliente.Id)) return NotFound();

Error("Não foi possível salvar as alterações.");

throw;

}

}

// GET: Clientes/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var cliente = await \_context.Clientes.FirstOrDefaultAsync(m => m.Id == id);

if (cliente == null) return NotFound();

ViewData["Title"] = "Excluir Cliente";

return View(cliente);

}

// POST: Clientes/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var cliente = await \_context.Clientes.FindAsync(id);

if (cliente != null)

{

\_context.Clientes.Remove(cliente);

await \_context.SaveChangesAsync();

Success("Cliente excluído com sucesso!");

}

return RedirectToAction(nameof(Index));

}

}

}

**Alterar Partial de Alertas — Views/Shared/\_Alerts.cshtml**

@{

var success = TempData["Success"] as string;

var error = TempData["Error"] as string;

}

@if (!string.IsNullOrEmpty(success))

{

<div class="alert alert-success alert-dismissible fade show" role="alert">

<i class="bi bi-check-circle me-2"></i>@success

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Fechar"></button>

</div>

}

@if (!string.IsNullOrEmpty(error))

{

<div class="alert alert-danger alert-dismissible fade show" role="alert">

<i class="bi bi-x-circle me-2"></i>@error

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Fechar"></button>

</div>

}

**Partial do Formulário — Views/Clientes/\_Form.cshtml**

usado tanto no **Create** quanto no **Edit**

@model VendasMvc.Models.Cliente

<div class="card p-3">

<div asp-validation-summary="ModelOnly" class="text-danger mb-2"></div>

<div class="row g-3">

<div class="col-12 col-md-6">

<label asp-for="Nome" class="form-label"></label>

<input asp-for="Nome" class="form-control" placeholder="Ex.: Maria Joana" />

<span asp-validation-for="Nome" class="text-danger"></span>

</div>

<div class="col-12 col-md-6">

<label asp-for="Email" class="form-label"></label>

<input asp-for="Email" class="form-control" placeholder="email@dominio.com" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

<div class="col-12 col-md-4">

<label asp-for="Telefone" class="form-label"></label>

<input asp-for="Telefone" class="form-control" placeholder="(18) 99999-9999" data-mask="phone" inputmode="numeric" />

<span asp-validation-for="Telefone" class="text-danger"></span>

</div>

</div>

</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

<script>

// máscara leve para telefone BR (sem libs)

function maskPhone(el) {

let v = el.value.replace(/\D/g, '').slice(0, 11);

if (v.length >= 1) v = '(' + v;

if (v.length >= 3) v = v.slice(0, 3) + ') ' + v.slice(3);

if (v.length >= 10) v = v.slice(0, 10) + '-' + v.slice(10);

el.value = v;

}

document.querySelectorAll('input[data-mask="phone"]').forEach(i => {

i.addEventListener('input', e => maskPhone(e.target));

maskPhone(i);

});

</script>

}

**5) Views**

**5.1 Views/Clientes/Index.cshtml**

@model IEnumerable<VendasMvc.Models.Cliente>

@{

ViewData["Title"] = "Clientes";

var q = ViewData["q"] as string ?? "";

}

@await Html.PartialAsync("\_Alerts")

<div class="d-flex justify-content-between align-items-center mb-3">

<h1 class="m-0">Clientes</h1>

<a asp-action="Create" class="btn btn-primary">

<i class="bi bi-person-plus"></i> Novo Cliente

</a>

</div>

<form method="get" class="mb-3">

<div class="input-group">

<span class="input-group-text"><i class="bi bi-search"></i></span>

<input type="text" class="form-control" name="q" value="@q" placeholder="Buscar por nome, e-mail ou telefone..." />

<button class="btn btn-outline-light" type="submit">Buscar</button>

@if (!string.IsNullOrWhiteSpace(q))

{

<a asp-action="Index" class="btn btn-outline-secondary">Limpar</a>

}

</div>

</form>

<div class="card p-0">

<div class="table-responsive">

<table class="table align-middle mb-0">

<thead>

<tr>

<th>Nome</th>

<th>E-mail</th>

<th>Telefone</th>

<th class="text-end">Ações</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Nome</td>

<td>

@if (!string.IsNullOrWhiteSpace(item.Email))

{

<a href="mailto:@item.Email" class="link-light">@item.Email</a>

}

</td>

<td>@item.Telefone</td>

<td class="text-end">

<div class="btn-group">

<a asp-action="Details" asp-route-id="@item.Id" class="btn btn-sm btn-outline-light">

<i class="bi bi-eye"></i>

</a>

<a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-sm btn-outline-warning">

<i class="bi bi-pencil"></i>

</a>

<a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-outline-danger">

<i class="bi bi-trash"></i>

</a>

</div>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

**Views/Clientes/Create.cshtml**

@model VendasMvc.Models.Cliente

@{

ViewData["Title"] = "Novo Cliente";

}

@await Html.PartialAsync("\_Alerts")

<h1 class="mb-3">Novo Cliente</h1>

<form asp-action="Create" method="post">

<partial name="\_Form" />

<div class="mt-3 d-flex gap-2">

<button type="submit" class="btn btn-primary">

<i class="bi bi-check2"></i> Salvar

</button>

<a asp-action="Index" class="btn btn-secondary">Cancelar</a>

</div>

</form>

**Views/Clientes/Edit.cshtml**

@model VendasMvc.Models.Cliente

@{

ViewData["Title"] = "Editar Cliente";

}

@await Html.PartialAsync("\_Alerts")

<h1 class="mb-3">Editar Cliente</h1>

<form asp-action="Edit" method="post">

<input type="hidden" asp-for="Id" />

<partial name="\_Form" />

<div class="mt-3 d-flex gap-2">

<button type="submit" class="btn btn-primary">

<i class="bi bi-check2"></i> Salvar alterações

</button>

<a asp-action="Index" class="btn btn-secondary">Cancelar</a>

</div>

</form>

**Views/Clientes/Details.cshtml**

@model VendasMvc.Models.Cliente

@{

ViewData["Title"] = "Detalhes do Cliente";

}

<h1 class="mb-3">Detalhes do Cliente</h1>

<div class="card p-3 mb-3">

<dl class="row mb-0">

<dt class="col-sm-3">Nome</dt>

<dd class="col-sm-9">@Model.Nome</dd>

<dt class="col-sm-3">E-mail</dt>

<dd class="col-sm-9">

@if (!string.IsNullOrWhiteSpace(Model.Email))

{

<a href="mailto:@Model.Email" class="link-light">@Model.Email</a>

}

</dd>

<dt class="col-sm-3">Telefone</dt>

<dd class="col-sm-9">@Model.Telefone</dd>

</dl>

</div>

<div class="d-flex gap-2">

<a asp-action="Edit" asp-route-id="@Model.Id" class="btn btn-warning">

<i class="bi bi-pencil"></i> Editar

</a>

<a asp-action="Index" class="btn btn-secondary">Voltar</a>

</div>