**Crud 1-N - .NET Com Postgres**

**Passo 1: Criar o Projeto MVC no Visual Studio**

1. Abra o **Visual Studio 2022**

Tipo: ASP.NET Core Web App (Model-View-Controller)

De o nome de CrudProdutoFornecedor

**Passo 2: Instalar os pacotes necessários**

No menu do Visual Studio, vá em:

Ferramentas > Gerenciador de Pacotes NuGet > Console do Gerenciador de Pacotes

Instale os pacotes:

Install-Package Npgsql.EntityFrameworkCore.PostgreSQL

Install-Package Microsoft.EntityFrameworkCore.Tools

**Passo 3: Criar o Banco no PostgreSQL**

Abra o pgAdmin e crie um novo banco de dados chamado CrudFornecedorProduto.

**Passo 4: Configurar conexão com PostgreSQL**

Edite o arquivo appsettings.json:

{

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Database=CrudFornecedorProduto;Username=postgres;Password=suaSenhaAqui"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Passo 5: Criar as classes (Models)**

Em Models, crie:

**Fornecedor.cs**

namespace CrudProdutoFornecedor.Models;

public class Fornecedor

{

public int FornecedorId { get; set; }

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

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

public ICollection<Produto>? Produtos { get; set; }

}

**Produto.cs**

**Passo 6: Criar a classe Context**

Crie uma pasta chamada Data e nela crie a classe ApplicationDbContext.cs:

using CrudProdutoFornecedor.Models;

using Microsoft.EntityFrameworkCore;

namespace CrudProdutoFornecedor.Data;

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Fornecedor> Fornecedores { get; set; }

public DbSet<Produto> Produtos { get; set; }

}

**Passo 7: Registrar Context no Program.cs**

Edite o arquivo Program.cs adicionando:

using CrudProdutoFornecedor.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

if (!app.Environment.IsDevelopment())

{

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

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

**Passo 8: Gerar as Migrations e criar o Banco**

Execute no Console do Gerenciador de Pacotes:

Add-Migration Inicial

Update-Database

**Passo 9: Criar Controllers e Views (Scaffold)**

Vá em:

Clique direito em Controllers > Adicionar > Novo Scaffolded Item

Escolha:

Controlador MVC com exibições usando EF

Repita o processo para:

* Produto
* Fornecedor

Exemplo configuração Produto:

* Classe do modelo: **Produto**
* Classe do contexto de dados: **ApplicationDbContext**
* Clique em Adicionar.

Repita para **Fornecedor**.

**Passo 10: Executar o projeto**

* Pressione F5 para executar o projeto.
* Navegue para /Produtos e /Fornecedores.

**Adicionando Interface**

1. Editar o arquivo Views/Home/Index.cshtml

Apague o conteúdo atual do Index.cshtml e substitua por:

@{

ViewData["Title"] = "Página Principal";

}

<div class="container mt-5">

<div class="text-center">

<h1>Bem-vindo ao Sistema de Gestão</h1>

<p class="lead">Gerencie seus produtos e fornecedores de forma prática!</p>

</div>

<div class="row justify-content-center mt-5">

<div class="col-md-4 text-center">

<div class="card shadow-lg p-3 mb-5 bg-white rounded">

<div class="card-body">

<h5 class="card-title">Cadastro de Fornecedores</h5>

<p class="card-text">Gerencie informações dos fornecedores cadastrados.</p>

<a href="/Fornecedores" class="btn btn-primary">Acessar</a>

</div>

</div>

</div>

<div class="col-md-4 text-center">

<div class="card shadow-lg p-3 mb-5 bg-white rounded">

<div class="card-body">

<h5 class="card-title">Cadastro de Produtos</h5>

<p class="card-text">Gerencie os produtos disponíveis para venda.</p>

<a href="/Produtos" class="btn btn-success">Acessar</a>

</div>

</div>

</div>

</div>

</div>

**2. Melhorar o visual (CSS extra)**

Você pode adicionar um pequeno CSS para dar mais vida à página.

No Views/Shared/\_Layout.cshtml, entre as <head> tags, adicione depois do Bootstrap:

<style>

body {

background-color: #f8f9fa;

}

.card {

transition: transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

</style>

**3. Melhorando Layout das páginas de cadastro:**

Melhorando produtos

Views/Produtos/Index.cshtml

Onde aparece isso

<a asp-action="Edit" asp-route-id="@item.ProdutoId">Editar</a> |

<a asp-action="Details" asp-route-id="@item.ProdutoId">Detalhes</a> |

<a asp-action="Delete" asp-route-id="@item.ProdutoId">Excluir</a>,

Trocar por isso:

<a asp-action="Edit" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-warning">Editar</a>

<a asp-action="Details" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-info">Detalhes</a>

<a asp-action="Delete" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-danger">Excluir</a>

Código completo para Produtos

@model IEnumerable<CrudProdutoFornecedor.Models.Produto>

@{

ViewData["Title"] = "Produtos";

}

<div class="container mt-4">

<h1>Lista de Produtos</h1>

<p>

<a asp-action="Create" class="btn btn-success">Novo Produto</a>

</p>

<table class="table table-striped table-hover">

<thead class="table-dark">

<tr>

<th>Nome</th>

<th>Preço</th>

<th>Fornecedor</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Nome</td>

<td>R$ @item.Preco.ToString("F2")</td>

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

<td>

<a asp-action="Edit" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-warning">Editar</a>

<a asp-action="Details" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-info">Detalhes</a>

<a asp-action="Delete" asp-route-id="@item.ProdutoId" class="btn btn-sm btn-danger">Excluir</a>

</td>

</tr>

}

</tbody>

</table>

</div>

Melhorando Fornecedores

Código completo para Fornecedores

@model IEnumerable<CrudProdutoFornecedor.Models.Fornecedor>

@{

ViewData["Title"] = "Fornecedores";

}

<div class="container mt-4">

<h1>Lista de Fornecedores</h1>

<p>

<a asp-action="Create" class="btn btn-success">Novo Fornecedor</a>

</p>

<table class="table table-striped table-hover">

<thead class="table-dark">

<tr>

<th>Nome</th>

<th>Cidade</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Nome</td>

<td>@item.Cidade</td>

<td>

<a asp-action="Edit" asp-route-id="@item.FornecedorId" class="btn btn-sm btn-warning">Editar</a>

<a asp-action="Details" asp-route-id="@item.FornecedorId" class="btn btn-sm btn-info">Detalhes</a>

<a asp-action="Delete" asp-route-id="@item.FornecedorId" class="btn btn-sm btn-danger">Excluir</a>

</td>

</tr>

}

</tbody>

</table>

</div>

**Mostrando nome dos fornecedores no cadastro de produto**

**1. Alterar o Controller de Produto (GET Create)**

No ProdutosController.cs, altere o método Create (GET) para carregar a lista de fornecedores:

public IActionResult Create()

{

ViewData["FornecedorId"] = new SelectList(\_context.Fornecedores, "FornecedorId", "Nome");

return View();

}

**2. Alterar a View Create.cshtml**

Agora, no Views/Produtos/Create.cshtml, substitua isso:

<div class="form-group">

<label asp-for="FornecedorId" class="control-label"></label>

<input asp-for="FornecedorId" class="form-control" />

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

</div>

Por isso:

<div class="form-group">

<label asp-for="FornecedorId" class="control-label"></label>

<select asp-for="FornecedorId" class="form-control" asp-items="@(ViewData["FornecedorId"] as SelectList)">

<option value="">-- Selecione um fornecedor --</option>

</select>

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

</div>

**Alterando na opção de detalhes**

**Ajustar o ProdutosController.cs para incluir o Fornecedor**

No método Details do seu ProdutosController.cs, altere de:

var produto = await \_context.Produtos.FindAsync(id);

**Para:**

var produto = await \_context.Produtos

.Include(p => p.Fornecedor)

.FirstOrDefaultAsync(p => p.ProdutoId == id);

**No Editar a View Details.cshtml**

No arquivo Views/Produtos/Details.cshtml, localize o trecho que mostra FornecedorId:

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

@Html.DisplayNameFor(model => model.FornecedorId)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.FornecedorId)

</dd>

**Para:**

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

Fornecedor

</dt>

<dd class="col-sm-10">

@Model.Fornecedor?.Nome

</dd>

Fazer o mesmo para o **Delete**

**Resolvendo problemas das casas decimais – cadastro de produto**

Esse problema que você está enfrentando — **valor 10.50 sendo salvo como 1050,00** no banco — é **muito comum** em sistemas ASP.NET Core, principalmente quando se usa **decimal** + **cultura diferente (pt-BR)**.

🔎 Resumind**o:**

* **No Brasil, usamos vírgula como separador decimal (ex: 10,50)**
* **No banco PostgreSQL e internamente no .NET (padrão** en-US), o separador é ponto (ex: 10.50)
* Quando você digita 10.50, o sistema interpreta como 1050.

**Solução 1: Configurar a Cultura Global no Program.cs**

**No seu Program.cs, antes de app.Run(); adicione isto:**

var cultureInfo = new CultureInfo("pt-BR");

CultureInfo.DefaultThreadCurrentCulture = cultureInfo;

CultureInfo.DefaultThreadCurrentUICulture = cultureInfo;

**Solução 2: Alterar o campo no formulário para aceitar ponto corretamente**

Se você não quiser alterar a cultura global, outra abordagem (um pouco mais manual) seria:

No seu formulário de cadastro (Create.cshtml), onde tem o campo de preço:

<div class="form-group">

<label asp-for="Preco" class="control-label"></label>

<input asp-for="Preco" class="form-control" />

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

</div>

**Garanta que no navegador o campo de preço seja tratado como text**, e explique para o usuário que ele deve usar vírgula (10,50) ao invés de ponto.

Ou, force no HTML5:

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

**O que resolveu no meu projeto**

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