Roteiro de Instalação e Configuração .NET 8 LTS + ASP.NET Web API no VS Code

Do zero ao primeiro projeto Web API funcionando

📋 Índice

1. Pré-requisitos
2. Instalação do .NET 8 LTS
3. Instalação e Configuração do VS Code
4. Extensões Essenciais
5. Verificação da Instalação
6. Criando Primeiro Projeto Web API
7. Configuração do Ambiente
8. Testando a API
9. Solução de Problemas Comuns
10. Recursos Adicionais

---

1. Pré-requisitos

💻 Sistema Operacional

- ☑ Windows 10/11 (x64)
- ☑ macOS 10.15+
- ☑ Linux (Ubuntu 18.04+, CentOS, RHEL, etc.)

🔧 Requisitos Mínimos

- **RAM**: 4 GB (recomendado 8 GB+)
- **Espaço em Disco**: 2 GB livres
- **Processador**: x64 ou ARM64

---

2. Instalação do .NET 8 LTS

🪟 Windows

Opção 1: Download Oficial (Recomendado)

1. **Acesse o site oficial:**

```
https://dotnet.microsoft.com/download/dotnet/8.0
```

2. **Baixe o .NET 8 SDK:**
    - Clique em "Download .NET 8 SDK x64"
    - Execute o arquivo `.exe` baixado
    - Siga o wizard de instalação

Opção 2: Via Chocolatey

```
# Instalar Chocolatey (se não tiver)
Set-ExecutionPolicy Bypass -Scope Process -Force
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072
iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.
ps1'))

# Instalar .NET 8 SDK
choco install dotnet-8.0-sdk -y
```

Opção 3: Via WinGet

```
winget install Microsoft.DotNet.SDK.8
```

Linux (Ubuntu/Debian)

```
# Adicionar repositório Microsoft
wget https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-
prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb

# Atualizar pacotes
sudo apt-get update

# Instalar .NET 8 SDK
sudo apt-get install -y dotnet-sdk-8.0
```

---

3. Instalação e Configuração do VS Code

Download e Instalação

1. **Acesse:** `https://code.visualstudio.com/`
2. **Baixe** a versão para seu sistema operacional

3. **Execute** o installer e siga as instruções

⚙ Configuração Inicial

Durante a instalação no Windows, marque as opções:

- ☑ Add "Open with Code" action to Windows Explorer file context menu
- ☑ Add "Open with Code" action to Windows Explorer directory context menu
- ☑ Register Code as an editor for supported file types
- ☑ Add to PATH

---

4. Extensões Essenciais

🔌 Instalação das Extensões

Abra o VS Code e instale as seguintes extensões (Ctrl+Shift+X):

Essenciais para .NET

```
# Extensão oficial da Microsoft para C#
ms-dotnettools.csharp

# IntelliCode para sugestões inteligentes
VisualStudioExptTeam.vscodeintellicode

# NuGet Package Manager
jmrog.vscode-nuget-package-manager
```

Produtividade

```
# Bracket Pair Colorizer (para visualizar chaves)
CoenraadS.bracket-pair-colorizer-2

# GitLens (para Git avançado)
eamodio.gitlens

# REST Client (para testar APIs)
humao.rest-client

# Thunder Client (alternativa ao Postman)
rangav.vscode-thunder-client
```

Instalação Rápida via Comando

```

```

```
# Abra o terminal integrado (Ctrl+`) e execute:
code --install-extension ms-dotnettools.csharp
code --install-extension VisualStudioExptTeam.vscodeintellicode
code --install-extension jmrog.vscode-nuget-package-manager
code --install-extension humao.rest-client
code --install-extension rangav.vscode-thunder-client
```

---

5. Verificação da Instalação

☑ Comandos de Verificação

Abra o terminal (Windows: PowerShell, macOS/Linux: Terminal) e execute:

```
# Verificar versão do .NET
dotnet --version
# Resultado esperado: 8.0.xxx

# Verificar SDKs instalados
dotnet --list-sdks
# Deve mostrar o .NET 8.0.xxx

# Verificar runtimes
dotnet --list-runtimes
# Deve mostrar Microsoft.AspNetCore.App 8.0.xxx

# Verificar templates disponíveis
dotnet new list
# Deve mostrar templates incluindo 'webapi'
```

🎯 Resultado Esperado

```
$ dotnet --version
8.0.100

$ dotnet --list-sdks
8.0.100 [C:\Program Files\dotnet\sdk]

$ dotnet --list-runtimes
Microsoft.AspNetCore.App 8.0.0 [C:\Program
Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.NETCore.App 8.0.0 [C:\Program
Files\dotnet\shared\Microsoft.NETCore.App]
```

---

6. Criando Primeiro Projeto Web API

🚀 Passo a Passo

1. Criar Diretório do Projeto

```
# Criar pasta para projetos
mkdir C:\Dev\MeusProjetos
cd C:\Dev\MeusProjetos

# Ou no Linux/macOS
mkdir ~/Dev/MeusProjetos
cd ~/Dev/MeusProjetos
```

2. Criar Projeto Web API

```
# Criar novo projeto Web API
dotnet new webapi -n MinhaAPITeste --framework net8.0

# Navegar para o diretório
cd MinhaAPITeste

# Restaurar dependências
dotnet restore
```

3. Abrir no VS Code

```
# Abrir projeto no VS Code
code .
```

🗀 Estrutura do Projeto Criada

```
MinhaAPITeste/
├── Controllers/
│   └── WeatherForecastController.cs
├── Properties/
│   └── launchSettings.json
├── appsettings.json
├── appsettings.Development.json
├── MinhaAPITeste.csproj
├── Program.cs
└── WeatherForecast.cs
```

---

7. Configuração do Ambiente

🔧 Configurar HTTPS (Opcional)

Se quiser usar HTTPS em desenvolvimento:

```
# Criar certificado de desenvolvimento
dotnet dev-certs https --trust
```

📝 Configurar para HTTP Apenas (Recomendado para Início)

Edite o arquivo `Properties/launchSettings.json`:

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "weatherforecast",
      "applicationUrl": "http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

🛠️ Modificar Program.cs para HTTP

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```

```
// Comentar esta linha para não exigir HTTPS
// app.UseHttpsRedirection();

app.UseAuthorization();
app.MapControllers();

app.Run();
```

---

8. Testando a API

🏃 Executar o Projeto

No Terminal Integrado do VS Code (Ctrl+`)

```
# Executar o projeto
dotnet run

# Ou especificar o perfil HTTP
dotnet run --launch-profile http
```

Resultado Esperado

```
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shutdown.
```

🧪 Testar os Endpoints

1. No Navegador

Acesse: `http://localhost:5000/weatherforecast`

2. Com cURL

```
curl http://localhost:5000/weatherforecast
```

3. Com PowerShell

```
Invoke-RestMethod -Uri "http://localhost:5000/weatherforecast"
```

4. Com REST Client (VS Code)

Crie um arquivo `test.http` no projeto:

```
### Testar WeatherForecast
GET http://localhost:5000/weatherforecast
Accept: application/json
```

📊 Resultado Esperado

```
[
  {
    "date": "2025-10-01",
    "temperatureC": 32,
    "temperatureF": 89,
    "summary": "Hot"
  },
  {
    "date": "2025-10-02",
    "temperatureC": -13,
    "temperatureF": 9,
    "summary": "Bracing"
  }
]
```

---

9. Solução de Problemas Comuns

❌ Problema: "dotnet: command not found"

Solução:

```
# Windows: Reiniciar terminal ou VS Code
# Linux/macOS: Adicionar ao PATH
export PATH=$PATH:/usr/share/dotnet
```

❌ Problema: "Failed to determine the https port for redirect"

Solução:

1. Comentar `app.UseHttpsRedirection();` no `Program.cs`
2. Ou criar certificado: `dotnet dev-certs https --trust`

❌ Problema: "Port already in use"

Solução:

```
# Windows: Verificar qual processo está usando a porta
netstat -ano | findstr :5000
taskkill /F /PID [PID_NUMBER]

# Linux/macOS
lsof -ti:5000 | xargs kill -9
```

❌ Problema: VS Code não reconhece C#

Solução:

1. Instalar extensão: `ms-dotnettools.csharp`
2. Recarregar VS Code: `Ctrl+Shift+P` → "Developer: Reload Window"
3. Verificar se aparece "OmniSharp" na barra de status

❌ Problema: IntelliSense não funciona

Solução:

```
# Limpar cache e restaurar
dotnet clean
dotnet restore
```

No VS Code: `Ctrl+Shift+P` → "OmniSharp: Restart OmniSharp"

---

10. Recursos Adicionais

📚 Documentação Oficial

- **Microsoft Learn**: `https://learn.microsoft.com/aspnet/core`
- **.NET API Documentation**: `https://docs.microsoft.com/dotnet/api`
- **ASP.NET Core Tutorials**: `https://learn.microsoft.com/aspnet/core/tutorials`

🛠️ Ferramentas Úteis

Postman (Testar APIs)

```
https://www.postman.com/downloads/
```

Docker (Containerização)

```
https://www.docker.com/products/docker-desktop
```

Git (Controle de Versão)

```
https://git-scm.com/downloads
```

🎓 Cursos e Tutoriais

- **Microsoft Learn - Web API**: `https://learn.microsoft.com/training/paths/build-web-api-aspnet-core/`
- **C# Documentation**: `https://learn.microsoft.com/dotnet/csharp/`
- **Entity Framework Core**: `https://learn.microsoft.com/ef/core/`

🔧 Configurações Avançadas do VS Code

Crie um arquivo `.vscode/settings.json` no projeto:

```
{
    "dotnet.completion.showCompletionItemsFromUnimportedNamespaces": true,
    "omnisharp.enableEditorConfigSupport": true,
    "omnisharp.enableImportCompletion": true,
    "files.exclude": {
        "**/bin": true,
        "**/obj": true
    }
}
```

📦 Comandos Úteis para Desenvolvimento

```
# Criar diferentes tipos de projeto
dotnet new webapi -n MinhaAPI        # Web API
dotnet new mvc -n MinhaWebApp        # MVC Web App
dotnet new console -n MinhaConsoleApp # Console App

# Gerenciar pacotes NuGet
dotnet add package EntityFrameworkCore
dotnet remove package EntityFrameworkCore
dotnet list package

# Build e publicação
dotnet build                         # Compilar
dotnet publish -c Release           # Publicar para produção
dotnet watch run                    # Executar com hot reload
```

☑ Checklist Final

Antes de começar a desenvolver, verifique se:

- ☐ .NET 8 SDK instalado e funcionando (`dotnet --version`)
- ☐ VS Code instalado com extensões essenciais
- ☐ Projeto Web API criado e compilando
- ☐ API respondendo em `http://localhost:5000/weatherforecast`
- ☐ IntelliSense funcionando no VS Code
- ☐ Capaz de fazer debug (F5)

🎉 Parabéns!

Você agora tem um ambiente completo de desenvolvimento .NET 8 + ASP.NET Core Web API configurado no VS Code!

Próximos passos recomendados:

1. Explorar o controller `WeatherForecastController.cs`
2. Criar seus próprios endpoints
3. Adicionar Entity Framework Core para banco de dados
4. Implementar autenticação JWT
5. Criar testes unitários

Bom desenvolvimento! 🚀

Documento criado em setembro de 2025 - Versão 1.0
Para dúvidas ou atualizações, consulte a documentação oficial da Microsoft