**DOCUMENTAÇÃO DA FERRAMENTA**

**Criado por Edmar de Souza**

**Documento criado para a implementação de um projeto usando C# ASP.NET Web API e consumindo os dados com Windows Forms para a NextSoft.**

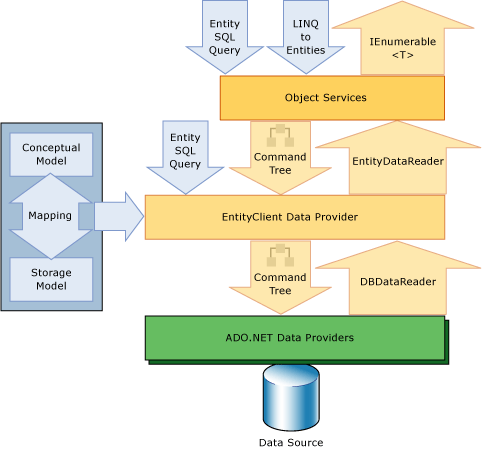
**API – Uso de C# ASP.NET Core e Entity Framework**

**Entity Framework**

o Entity Framework é um conjunto de tecnologias no ADO.NET que dão suporte ao desenvolvimento de aplicativos de software orientado a dados. O Entity Framework permite que os desenvolvedores trabalhem com dados na forma de objetos e propriedades específicos de domínio, como clientes e endereços de clientes, sem ter que se preocupar com as tabelas e colunas de banco de dados subjacentes em que esses data são armazenados. Com o Entity Framework, os desenvolvedores podem trabalhar em um nível mais alto de abstração ao lidar com dados e podem criar e manter aplicativos orientados a dados com menos códigos do que em aplicativos tradicionais. como o Entity Framework é um componente do .NET Framework, os aplicativos Entity Framework podem ser executados em qualquer computador no qual o .NET Framework a partir da versão 3,5 SP1 esteja instalado.

O diagrama a seguir ilustra a arquitetura de Entity Framework para acessar dados:

**Figura 1 - Funcionamento do Entity Framework**



Fonte: Docs Microsoft (2022)

**Os pacotes do NuGet para o Entity Framework**

Foram instalados os seguintes pacotes no projeto para uso do Entity Framework e suas ferramentas para modelar classes dentro do C#:

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.Design
* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools

Fora os pacotes do Entity Framework instalamos mais dois pacotes para uso no Visual Studio Community 2019:

* Microsoft.VisualStudio.Web.CodeGeneration.Design
* Swashbuckle.AspNetCore

**Model**

Foram criadas as models Categoria, Curso, Log, Usuario para ser usada como classe no C# e permitir acesso a manipulação dos campos no SQL Server.

**Categoria** -> É a categoria dos cursos que vão ser cadastrados.

**Curso** -> É as propriedades sobre os cursos que foram cadastrados.

**Log** -> É a entidade que guardará os dados referente aos logs que seriam os tipos de conexões HTTP na qual o dado foi incluído ou alterado no banco de dados.

**Usuario** -> É a entidade responsável em registrar o nome do usuário executou a ação que gerou o log.

**Controllers**

O controler é o responsável em fazer o meio de campo entre as entidades criadas pelo Entity Framework no C# e as requisições HTTP ao banco de dados. Para cada model foi criada um controller de mesmo nome.

**Migration**

O recurso de migrações no EF Core oferece uma maneira de atualizar de forma incremental o esquema de banco de dados para mantê-lo em sincronia com o modelo de dados do aplicativo, preservando os dados existentes no banco de dados.

Em um nível alto, as migrações funcionam da seguinte maneira:

* Quando uma alteração de modelo de dados é introduzida, o desenvolvedor usa as ferramentas do EF Core para adicionar uma migração correspondente que descreve as atualizações necessárias para manter o esquema de banco de dados em sincronia. O EF Core compara o modelo atual com um instantâneo do modelo antigo para determinar as diferenças e gera arquivos de origem de migração; os arquivos podem ser acompanhados no controle do código-fonte do projeto, como qualquer outro arquivo de origem.
* Depois que uma nova migração é gerada, é possível aplicá-la a um banco de dados de várias maneiras. O EF Core registra todas as migrações aplicadas em uma tabela de histórico especial, permitindo que ela saiba quais migrações foram ou não aplicadas.

**Protocolo HTTP**

HTTP é um protocolo baseado em texto sem conexão. Isso significa que as pessoas que acessam o site da sua empresa enviam solicitações a servidores que as exibem na forma do seu site em formato de texto, imagens, e outros tipos de mídia. Depois que a solicitação é atendida por um servidor, a conexão entre o usuário e o servidor é desconectada.

Uma nova conexão deve ser feita para cada solicitação, isto é, cada vez que alguém acessa o seu site. Em suma, quando alguém digita a URL do seu site em um navegador, é isto que acontece:

1. se a URL pertencer a um domínio próprio, o navegador primeiro se conecta a um servidor e recuperará o endereço IP correspondente ao servidor;
2. o navegador se conecta ao servidor e envia uma solicitação HTTP para a página da web desejada (que, neste exemplo, é o seu site);
3. o servidor recebe a solicitação e verifica a página desejada. Se a página existir, o servidor a mostrará. Se o servidor não conseguir encontrar a página solicitada, ele enviará uma [mensagem de erro HTTP 404](https://rockcontent.com/br/blog/erro-404/), ou seja, página não encontrada;
4. o navegador, então, recebe a página de volta e a conexão é fechada;
5. caso a página exista (e é isso que se espera), o navegador a analisa e procura outros elementos necessários para concluir a sua exibição, o que inclui seus textos, imagens e afins;
6. para cada um desses elementos, o navegador faz conexões adicionais e solicitações HTTP para o servidor para cada elemento;
7. quando o navegador terminar de carregar todos os elementos, a página será carregada na janela do navegador.

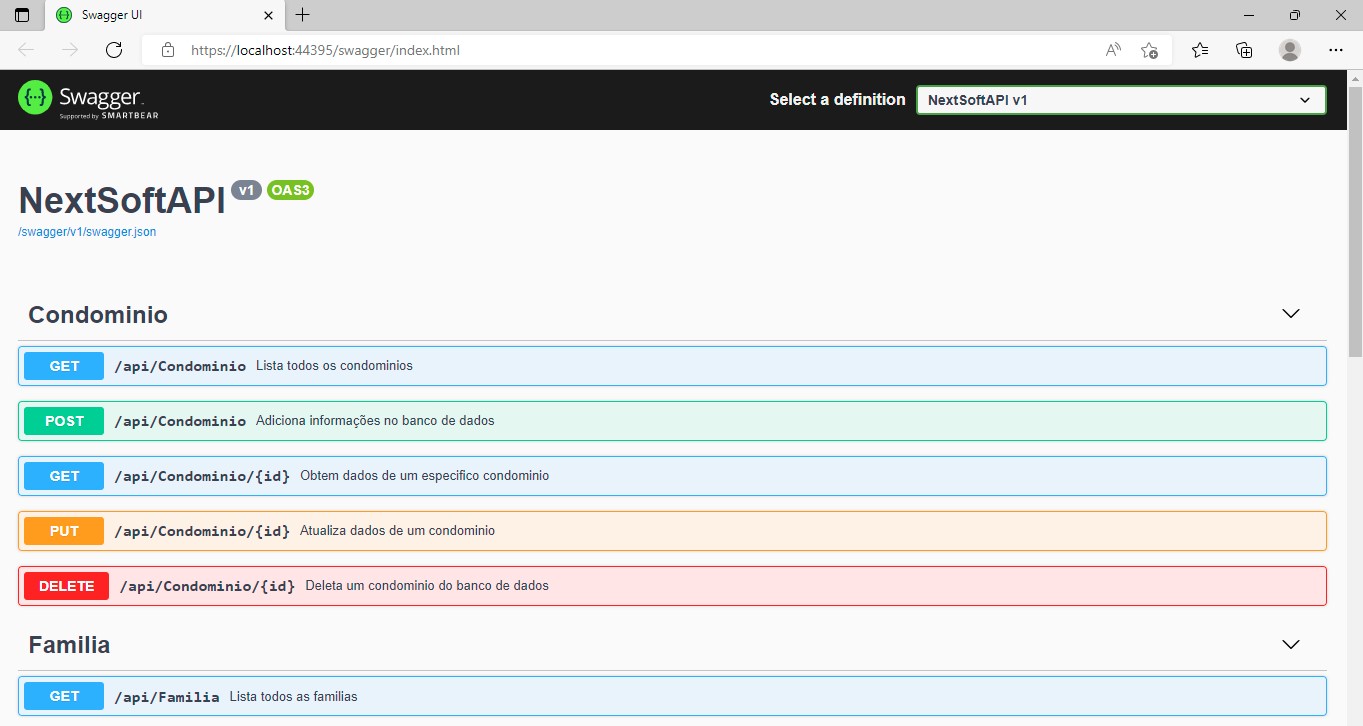
**Código de Status**

Os códigos de *status* das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

1. Respostas de informação (100-199),
2. Respostas de sucesso (200-299),
3. Redirecionamentos (300-399)
4. Erros do cliente (400-499)
5. Erros do servidor (500-599).

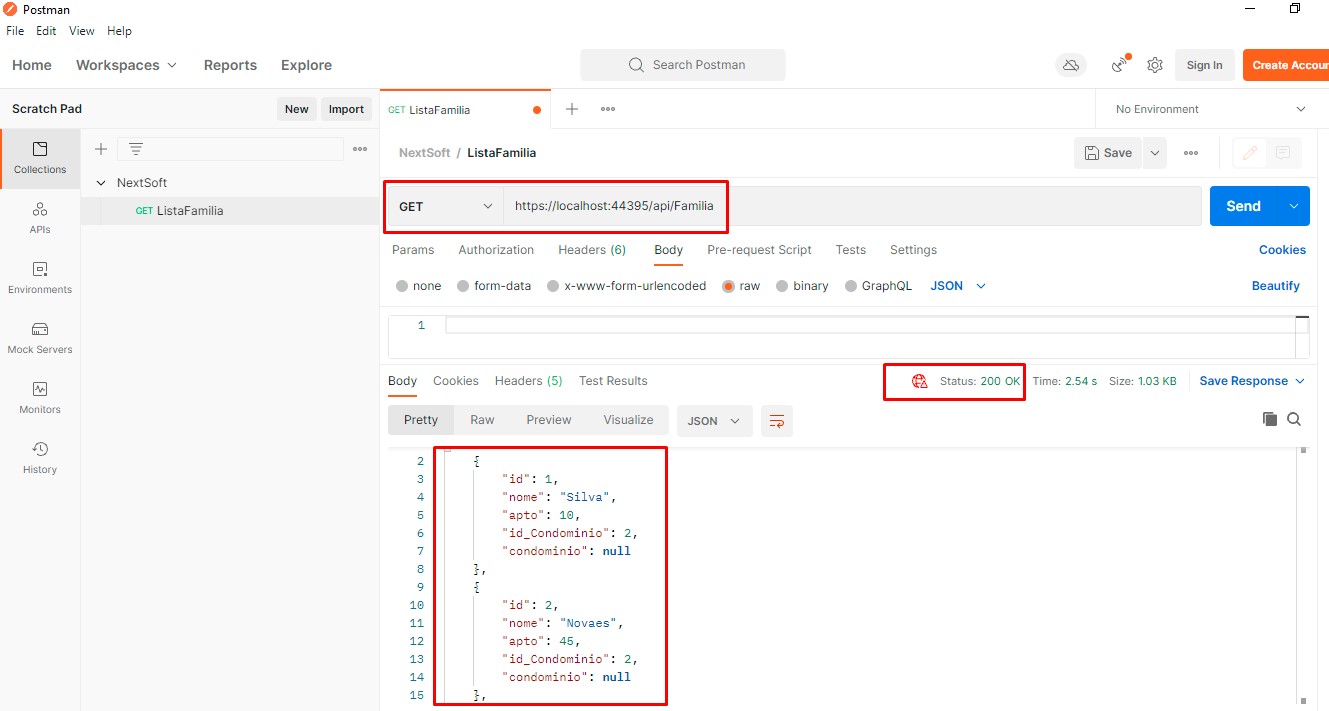
**EVIDÊNCIA DE TESTE**

**Figura 2 - Tela Principal da API**

****

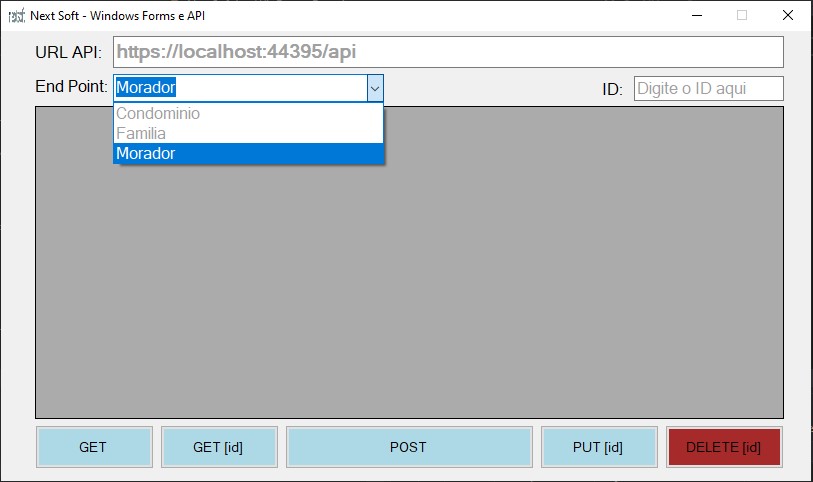
Fonte: Autor (2022)

**Figura 3 - Realizando requisição à API com PostMan**



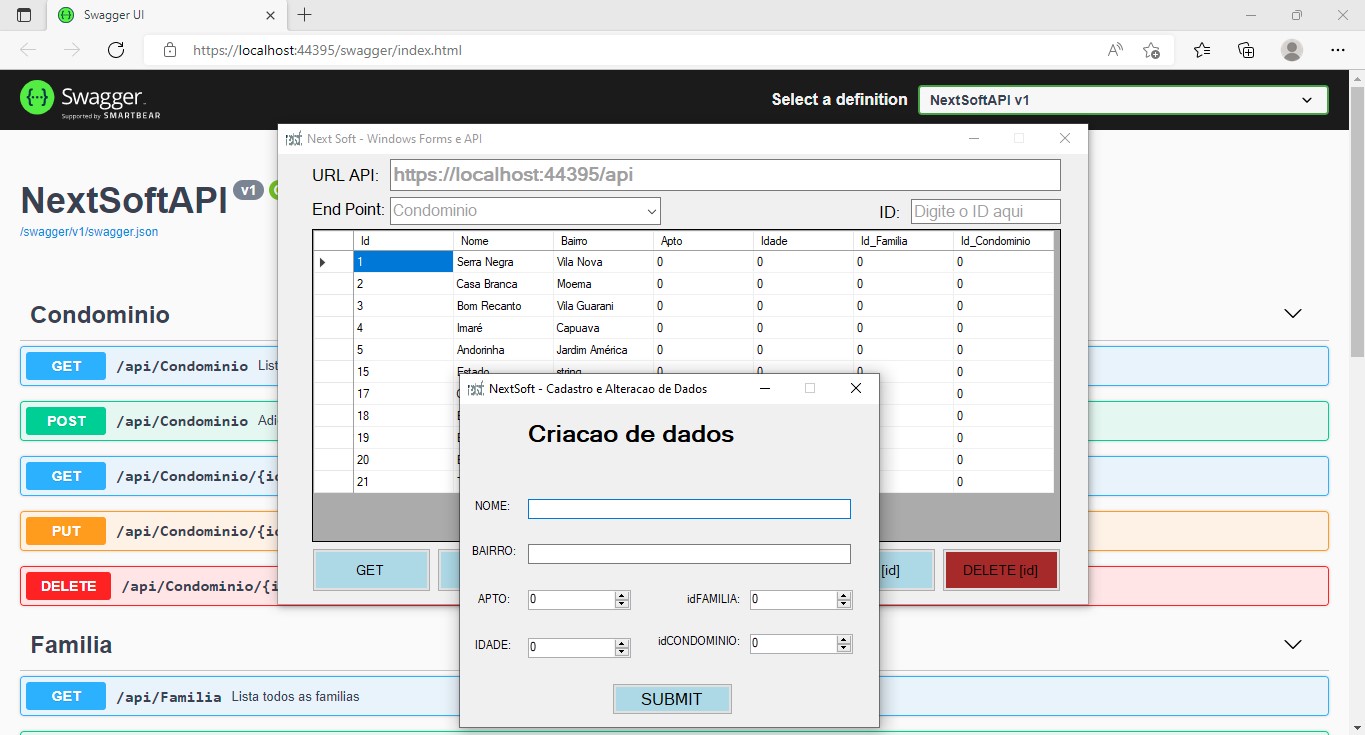
Fonte: Autor (2022)

**Figura 4 - Tela Inicial do APP que consome a API**



Fonte: Autor (2022)

**Figura 5 - Modal para POST e PUT do App**

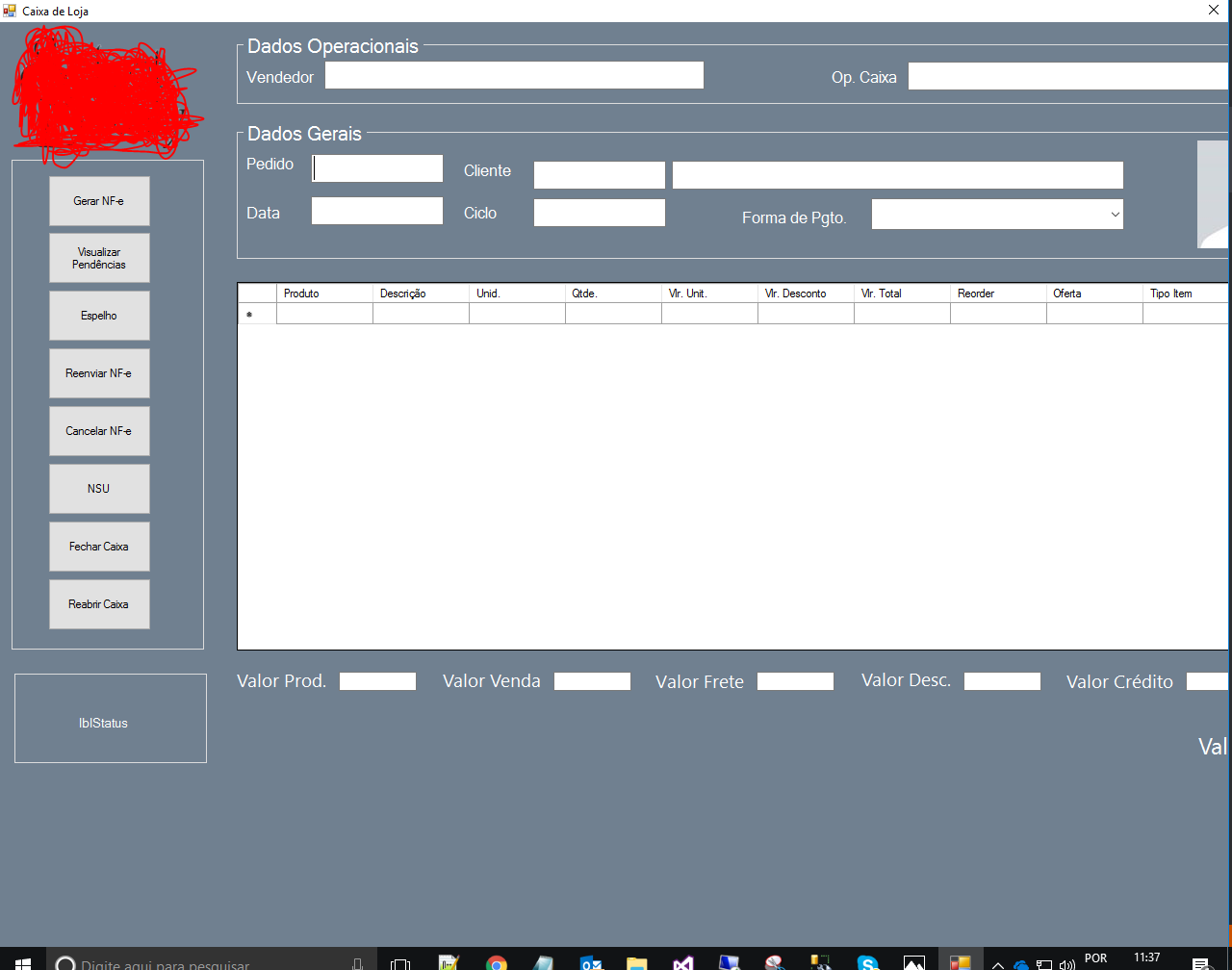


Fonte: Autor (2022)

**WINDOWS FORMS**

Windows Forms é uma estrutura de interface do usuário para criar Windows aplicativos de área de trabalho. Ele fornece uma das maneiras mais produtivas de criar aplicativos de área de trabalho com base no designer visual fornecido no Visual Studio. Funcionalidades como o posicionamento do tipo "arrastar e soltar" de controles visuais facilitam a criação de aplicativos da área de trabalho com o Windows Forms, você desenvolve aplicativos graficamente ricos que são fáceis de implantar, atualizar e trabalhar enquanto estiverem offline ou conectados à internet. Windows Forms aplicativos podem acessar o hardware local e o sistema de arquivos do computador em que o aplicativo está sendo executado. O Windows Forms faz parte do .NET Framework e está sendo descontinuado, apesar de existir muitas aplicações que ainda a utiliza.

**Figura 6 - Tela de uma aplicação Windows Forms**



Fonte: Stack (2022)

**FERRAMENTAS UTILIZADAS**

**Visual Studio 2019**

IDE para código em C#

**Entity Framework v5.0.17**

Mapea entidades do banco de dados para classes no C#

**Windows Forms .NET Framework 5**

Framework utilizado para criar aplicativo nativo para Windows

**ASP.NET Core Web API**

Framework utilizado para criação de API (API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.)

**SQL Server 2019**

Banco de dados da Microsoft

**PostMan v9.19.0**

O Postman é uma ferramenta que dá suporte à documentação das requisições feitas pela API.

**CONCLUSÃO**

A cada projeto executado percebo que mais conhecimento eu ganho das tecnologias envolvidas na interligação de uma API e o aplicativo que consome os dados. Entendo mais sobre os pormenores que antes não foi visto em outros projetos e a massa cinzenta fica mais lógica e melhor. Todo projeto me ajuda a entender sobre mim e sobre a carga profissional que tenho que buscar dia após dia cada vez mais. Humildade me define na busca dessa meta, não sei quanto tempo, só sei que vou conseguir. Isso já basta. Obrigado NextSoft pela oportunidade.

**REFERÊNCIAS BILIOGRÁFICAS**

Microsoft. [S:l]: Docs Microsoft, 28 set. 2021. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ef/overview>. Acessado em: 02 jun. 2021.

Microsoft. [S:l]: Docs Microsoft, 15 jul. 2021. Disponível em: <https://docs.microsoft.com/pt-br/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>. Acessado em: 02 jun. 2022.

Mozilla. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>. Acessado em: 02 jun. 2022.

Figura 6. Disponível em: <https://i.stack.imgur.com/PQUV8.png>. Acesso em 02 jun. 2022.