**DOCUMENTAÇÃO DA FERRAMENTA**

**Criado por Edmar de Souza**

**Documento criado para a implementação de um projeto usando C# ASP.NET e Angular para a turma de formação da Cast Group.**

**OBJETIVO**

Criar um software web para atender algumas regras de negócio que foi estabelecido pela empresa.

**REQUISITOS**

**Regra:**

Deve ser acessado pelo menu principal **“Curso”** da página inicial;  
Deve permitir cadastrar, visualizar, alterar e excluir um curso.  
Devem estar disponíveis os seguintes campos para criação do curso:

**• Curso**  
o Descrição do assunto (obrigatório)  
o Data de início (obrigatório)  
o Data de término (obrigatório)  
o Quantidade de alunos por turma (opcional)  
o Categoria (lista em banco de dados)

**• Categorias existentes**  
o Multiplataforma  
o Banco de dados  
o Metodologia  
o Comportamento  
o Comunicação

-Não será permitida a inclusão de cursos dentro do mesmo período. O sistema deve identificar tal situação e retornar um código de erro e a mensagem:  
**“Existe(m) curso(s) planejados(s) dentro do período informado.”**

-Não será permitida a inclusão de cursos com a data de início menor que a data atual.  
-Deve ter um botão “**Adicionar**”, para a inclusão no curso na lista;  
-Deve apresentar mensagem de alerta caso o campo não for preenchido;  
-Deve ter um botão “**Voltar**”, para encerrar o acesso e retorno a tela principal;  
-Deve apresentar mensagem de alerta de dados gravados com sucesso.  
-Deve apresentar mensagem de alerta de curso já cadastrada;

-Deve permitir a pesquisa pelo nome do curso e período que ele ocorre;  
-Não deve permitir a exclusão de cursos já realizados;  
-Deve registrar em uma tabela de log a data da inclusão, data da última atualização e usuário responsável;

**Entregáveis (versionado no github):**• Código fonte  
• Scripts de banco de dados  
• Evidências de teste  
• Descrição técnica da solução implementada

**SOFTWARES UTILIZADOS NO DESENVOLVIMENTO**

* Angular 12.2.12
* NPM 6.14.15
* Node v14.18.1
* Sistema Operacional Windows 10 Pro
* Visual Studio Code 1.61.2
* Visual Studio Community 2019
* Firefox 94.0.1
* Git version 2.33.0.windows.2
* Github
* SQL Server 2019
* Bootstrap v5.0

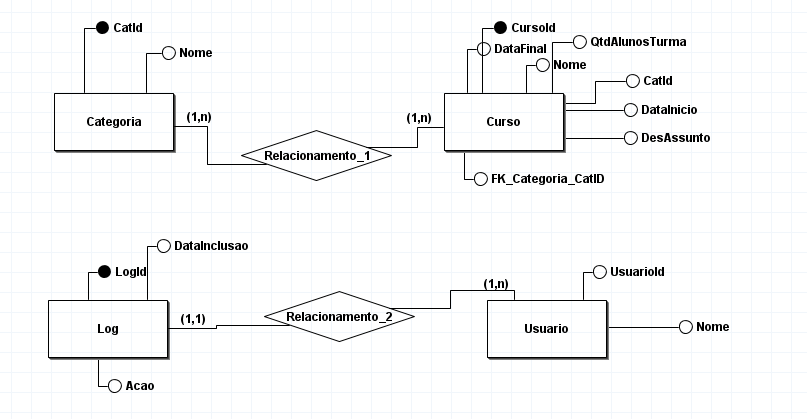
**LINGUAGENS UTILIZADAS NO PROJETO**

Abaixo há uma lista das linguagens utilizadas para criação, implementação e manutenção do código desse projeto:

* Typescript e Javascript para manter o SPA mais dinâmico e funcional.
* SQL ou T-SQL para manipulação, criação, exclusão e visualização de bases de dados.
* HTML, CSS e Bootstrap para estilização e personalização de cada elemento da página, permitindo até mesmo criar animações por meio dela.
* C# junto com ASP.NET e Entity Framework para criação e manutenção da API que será utilizada no Backend.

**BR Modelo**

Esquema de relacionamento entre as entidades do banco de dados:



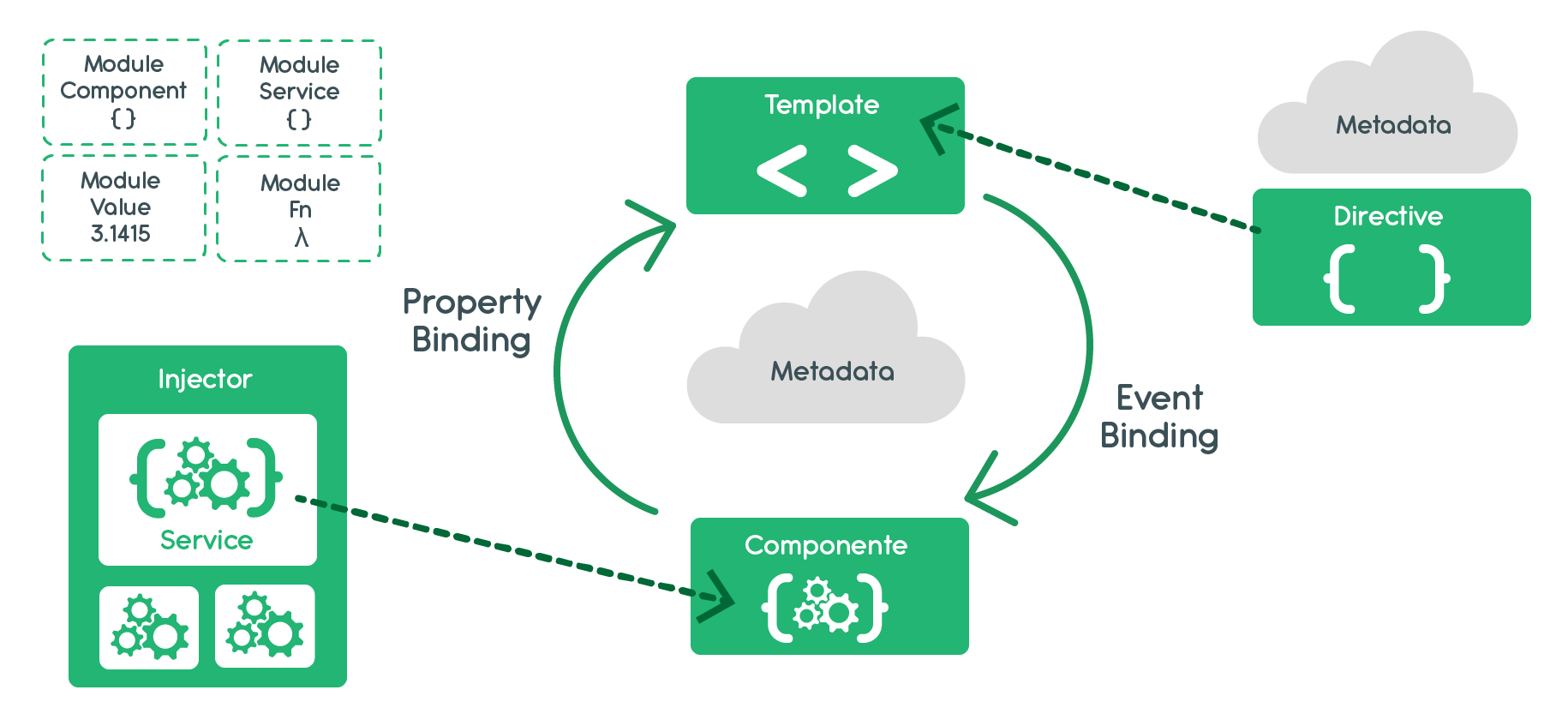
**ANGULAR**

Angular é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e, principalmente, JavaScript, criada pelos desenvolvedores da Google. O Angular permite a criação de Single Page Applications, ou seja, são carregadas somente uma única vez e utiliza de modularização para representa uma única página.

Os elementos básicos (*building blocks*) de uma aplicação Angular são:

* Módulos
* Componentes
* Templates
* Metadata
* Data binding
* Diretivas
* Serviços
* Injeção de dependências

Figura 1 – Entendendo os objetos do Angular



Fonte: Alga Works (2021)

**Módulos utilizados no Angular**

**NGX-Pagination** utilizado para criar uma paginação dos dados puxados do banco de dados.

**NGX-SearchPipe** utilizado para criar um campo de pesquisa de dados, seu ponto forte é uma implementação simples e funcional no Angular.

**HttpClient** permite utilizar requisições e pedidos via protocolo HTTP.

**Toastr** permite criar notificações e mensagens para os aplicativos angular.

**BrowserAnimation** é utilizado como dependência pelo Toastr para permitir as animações utilizada pelo Toastr.

**Boostrap** usado para otimizar e facilitar a configuração e personalização de páginas estáticas usando CSS para torná-las mais dinâmicas.

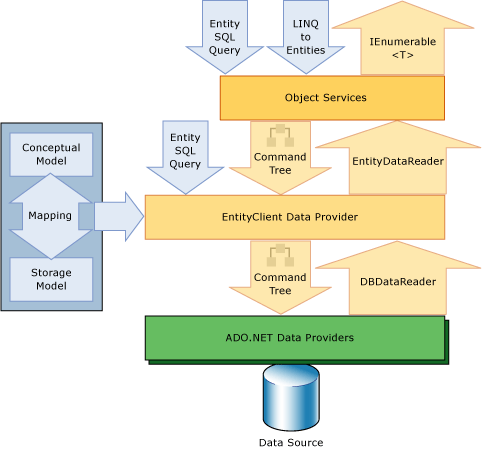
**API – Uso de C# ASP.NET Core e Entity Framework**

**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 2 – Funcionamento do Entity Framework**



Fonte: Docs Microsoft (2021)

**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).

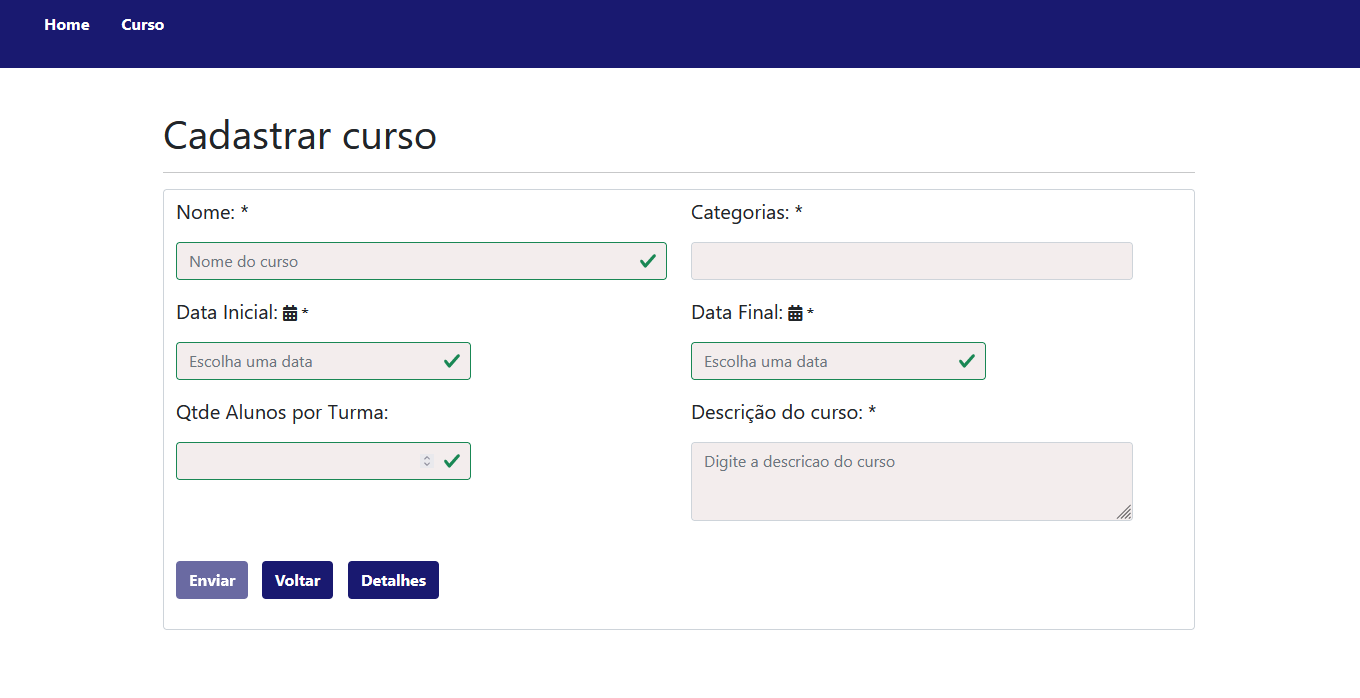
**EVIDÊNCIA DE TESTE**

**Figura 3 – Página Home**



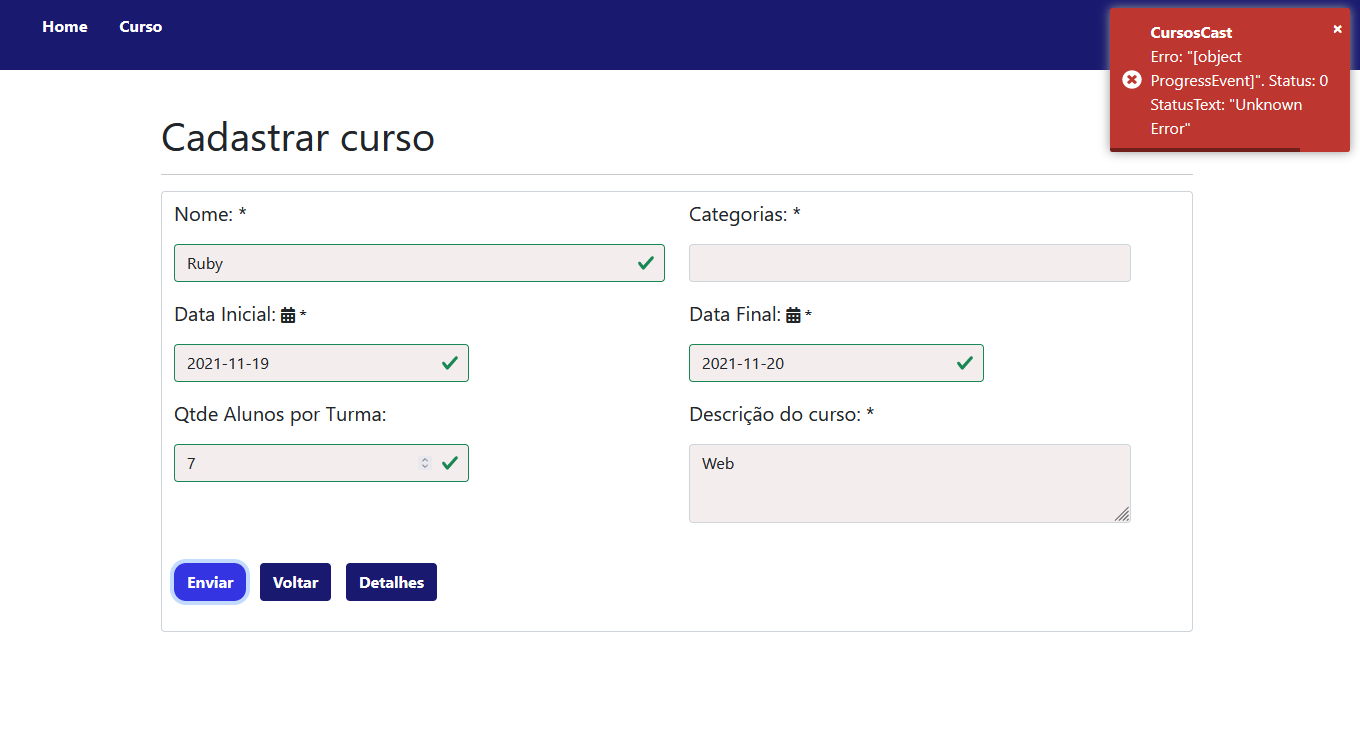
Fonte: Autor (2021)

**Figura 4 – Página de Cadastro de usuário**



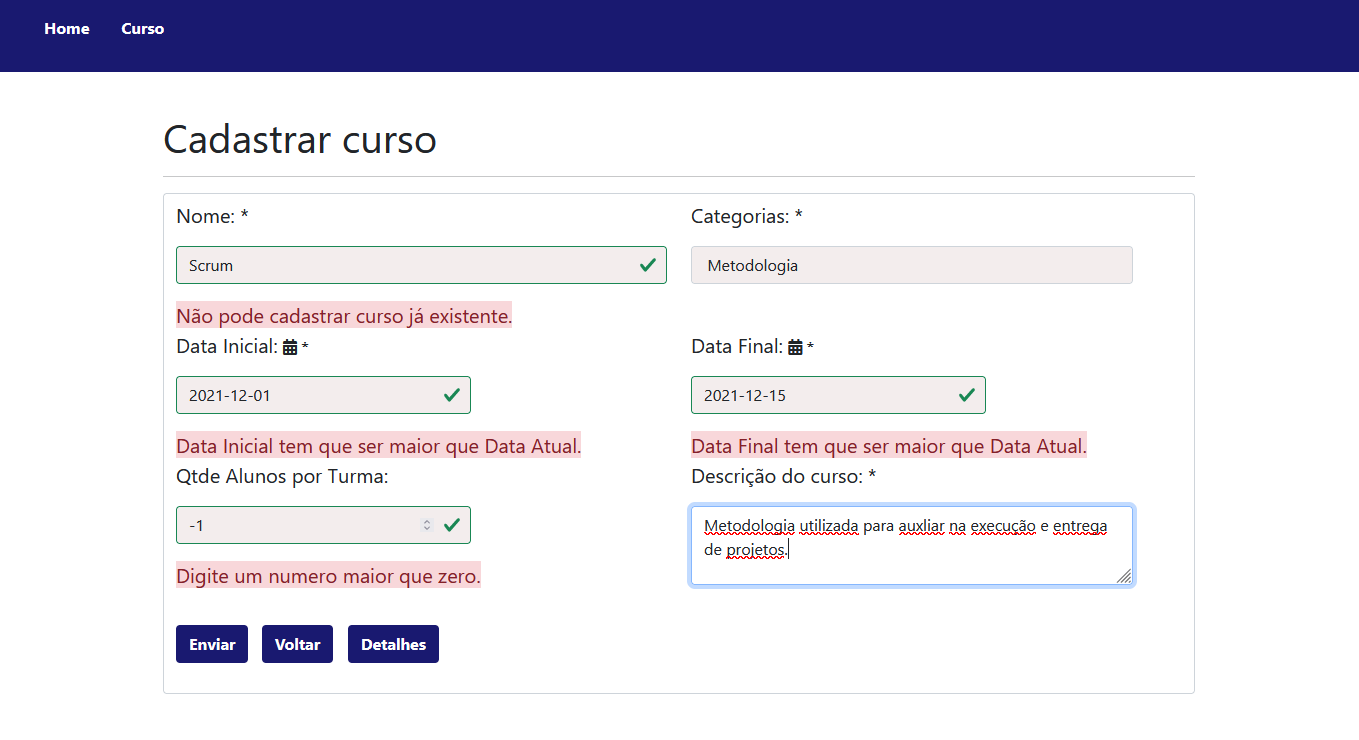
Fonte: Autor (2021)

**Figura 5 – Mensagem de erro na execução da aplicação**

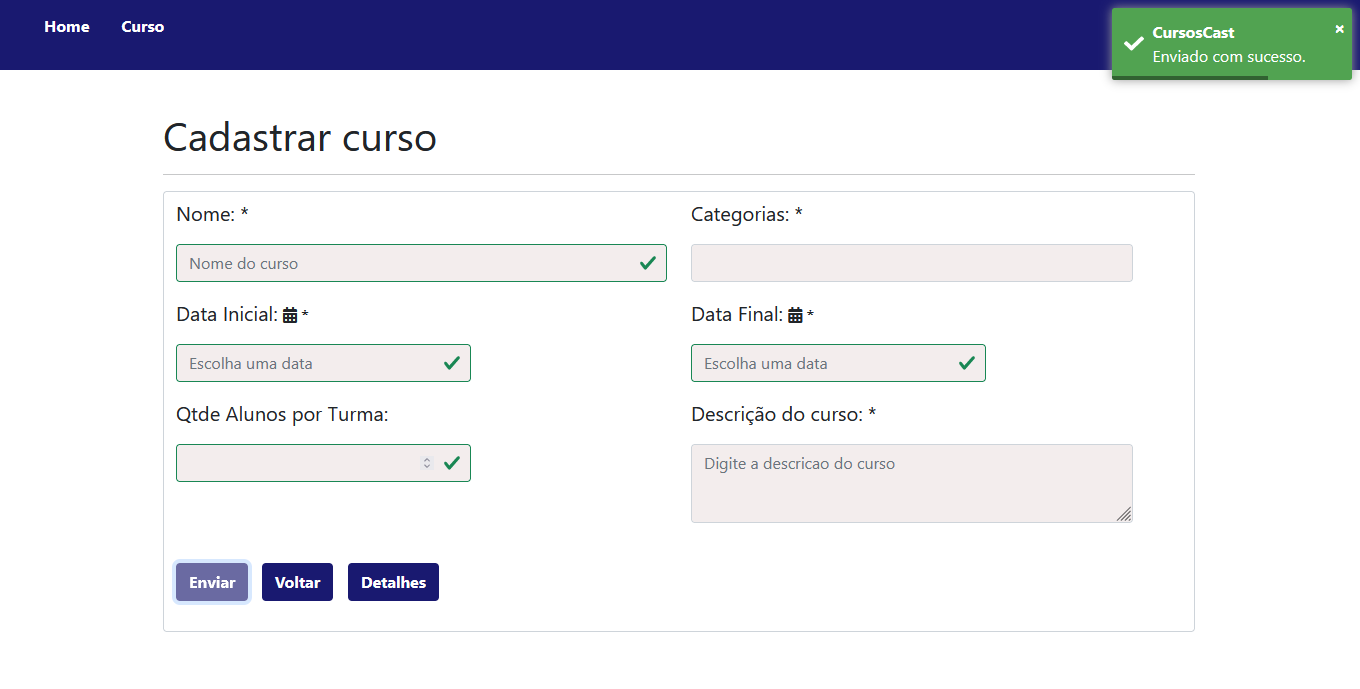


**Fonte: Autor (2021)**

**Figura 6 – Preenchimento e alerta para campos**

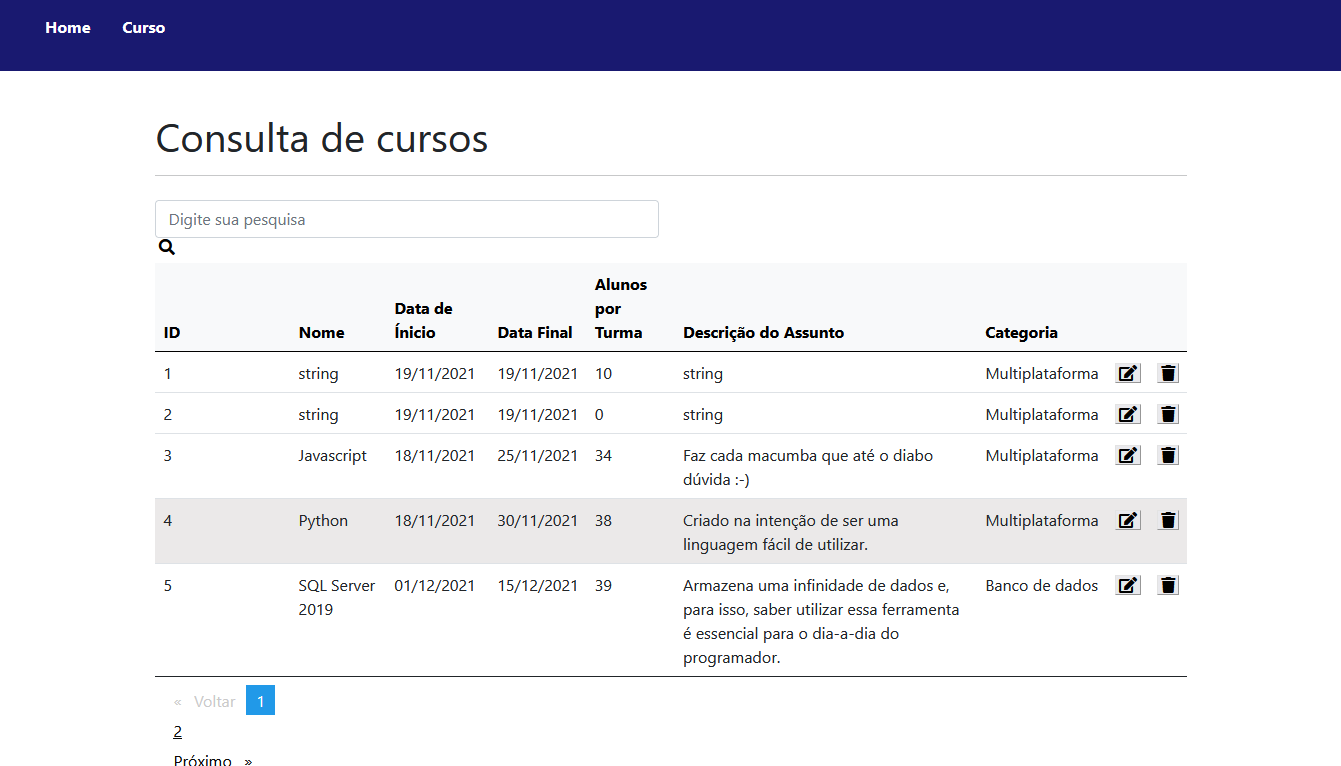


Fonte: Autor (2021)

**Figura 7 – Mensagem de sucesso no cadastramento**

Fonte: Autor (2021)

**Figura 8 – Relatório de cursos cadastrados no banco de dados**



Fonte: Autor (2021)

**CONCLUSÃO**

O projeto de integração de uma aplicação front-end e uma API envolve muitas tecnologias e pormenores que somente é vista no momento de execução do projeto. Isso permite ter uma visão melhor das comunicações e integrações entre softwares, linguagens e tecnologias diferentes. É possível perceber por meio desse projeto que o mundo web é infinitamente complexo, mas ao mesmo tempo animador, por proporcionar um alto nível de dinâmica entre as tecnologias aqui abordadas. Não foi somente possível aprender tecnologia na execução desse projeto, mas a conhecer e gerenciar meu tempo, minhas frustrações, o projeto e pedir ajuda sempre quando era necessário. Isso permite crescer em todos os aspectos, tanto em hard skill, quanto por soft skill.

**REFERÊNCIAS BILIOGRÁFICAS**

Google. Disponível em: <https://angular.io/>. Acessado em: 18 nov. 2021.

AFONSO, **Alexandre. O que é Angular?** Alga Works. Disponível em: <https://blog.algaworks.com/o-que-e-angular/>. Acessado em: 18 nov. 2021.

Microsoft. [S:l]: Docs Microsoft, 28 set. 2021. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ef/overview>. Acessado em: 18 nov. 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: 18 nov. 2021.

Mozilla. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>. Acessado em: 18 nov. 2021.