

Manual de Rotinas e de Execução

Desafio Prático FullStack (.NET Core + Angular + Banco de Dados)

Desenvolvido por Lucas Dejane

Data: 16/07/2025

Objetivo do documento

Este documento tem como propósito apresentar registros visuais reais da aplicação em funcionamento, por meio de imagens (“prints”) de telas e exemplos práticos de uso. Serão demonstradas também as movimentações realizadas diretamente no banco de dados, evidenciando a persistência e integridade das informações.

Todos os dados utilizados são fictícios, criados exclusivamente para fins de teste e validação da aplicação.

Além disso, serão incluídos cenários com erros simulados, com o objetivo de ilustrar o comportamento da aplicação frente às validações e regras de negócio definidas nos requisitos funcionais.

1. Configuração e Execução do Ambiente

Para a execução deste projeto FullStack, é necessário ter o seguinte ambiente configurado:

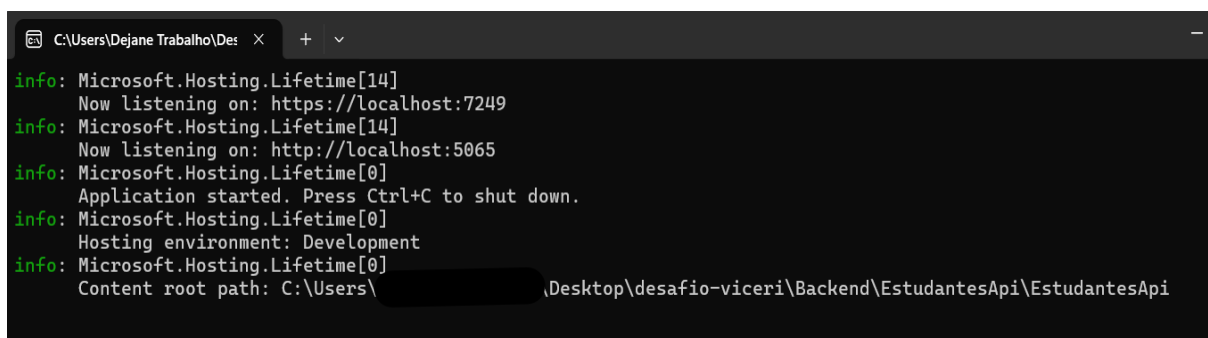
- **Node.js e npm:** Ambiente de execução JavaScript e seu gerenciador de pacotes.
- **Angular CLI:** Ferramenta de linha de comando para projetos Angular.
- **.NET SDK:** Kit de desenvolvimento de software para aplicações .NET (versão 8.0 LTS utilizada).
- **Visual Studio 2022 Community:** IDE utilizada para o desenvolvimento do Back-End ASP.NET Core.
- **SQL Server Express:** Sistema de Gerenciamento de Banco de Dados Relacional (SGBD) para persistência de dados.

1.1. Como Rodar a Aplicação

Para executar o sistema completo (Back-End e Front-End), siga os passos abaixo:

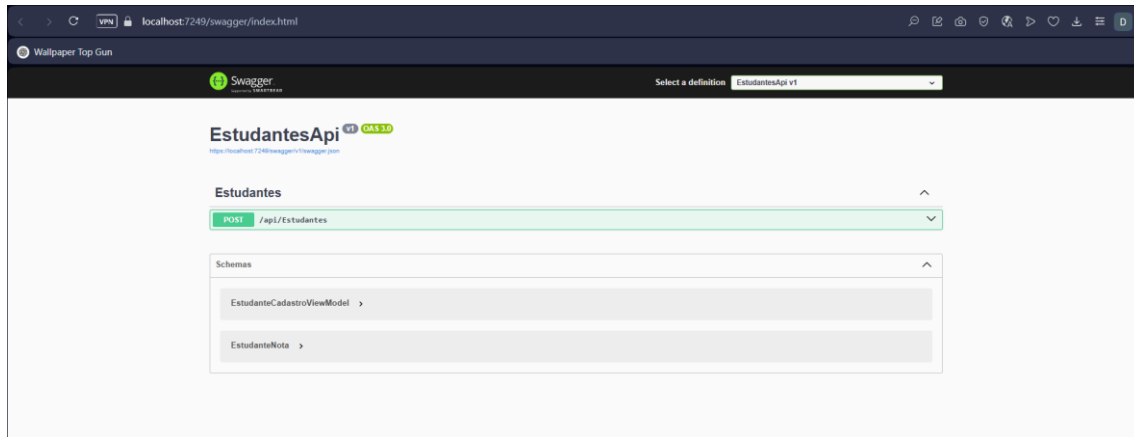
1.1.1. Iniciar o Back-End (API ASP.NET Core)

1. Abra a solução Backend/EstudantesApi/EstudantesApi.sln no Visual Studio 2022.
2. Pressione F5 ou clique no botão 'Play' (Iniciar) no Visual Studio para compilar e iniciar a API.
3. O console do Visual Studio (ou janela separada) exibirá as portas onde a API está ativa (ex: <https://localhost:7249>).



```
C:\Users\Dejane Trabalho\Des... X + v
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7249
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5065
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\...Desktop\desafio-viceri\Backend\EstudantesApi\EstudantesApi
```

4. Uma aba no navegador deverá abrir automaticamente na interface do Swagger UI (ex: <https://localhost:7249/swagger>), que serve para documentar e testar os endpoints da API.



1.1.2. Iniciar o Front-End (Aplicativo Angular)

1. Abra um terminal (preferencialmente Git Bash/MINGW64) e navegue até a pasta raiz do projeto Angular:
C:\Users\[SeuUsuario]\Desktop\desafio-viceri\Frontend\estudantes-app.
2. Execute o comando `ng serve --open` para compilar e iniciar o servidor de desenvolvimento Angular. Uma nova aba no navegador deverá abrir automaticamente.

```
$ ng serve --open
Initial chunk files | Names | Raw size
main.js | main | 18.48 kB
polyfills.js | polyfills | 95 bytes
styles.css | styles | 95 bytes
Initial total | 18.68 kB

Application bundle generation complete. [3.400 seconds]

▲ [WARNING] TypeScript compiler option 'isolatedModules' may prevent the 'emitDecoratorMetadata' option from emitting all metadata. [plugin angular-compiler]
The 'emitDecoratorMetadata' option is not required by Angular and can be removed if not explicitly required by the project.

▲ [WARNING] NG8113: RouterOutlet is not used within the template of AppComponent [plugin angular-compiler]
src/app/app.component.ts:12:4
12 | RouterOutlet,
   | ~~~~~

watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/
→ press h + enter to show help
```

3. Acesse <http://localhost:4200> no navegador.

2. Fluxo Padrão de Cadastro de Estudante

Esta seção demonstra o fluxo de cadastro de um novo estudante com suas notas, resultando em sucesso.

2.1. Tela Inicial do Formulário

Ao acessar o Front-End, o usuário se depara com o formulário de cadastro, onde pode inserir as informações do estudante e suas notas.

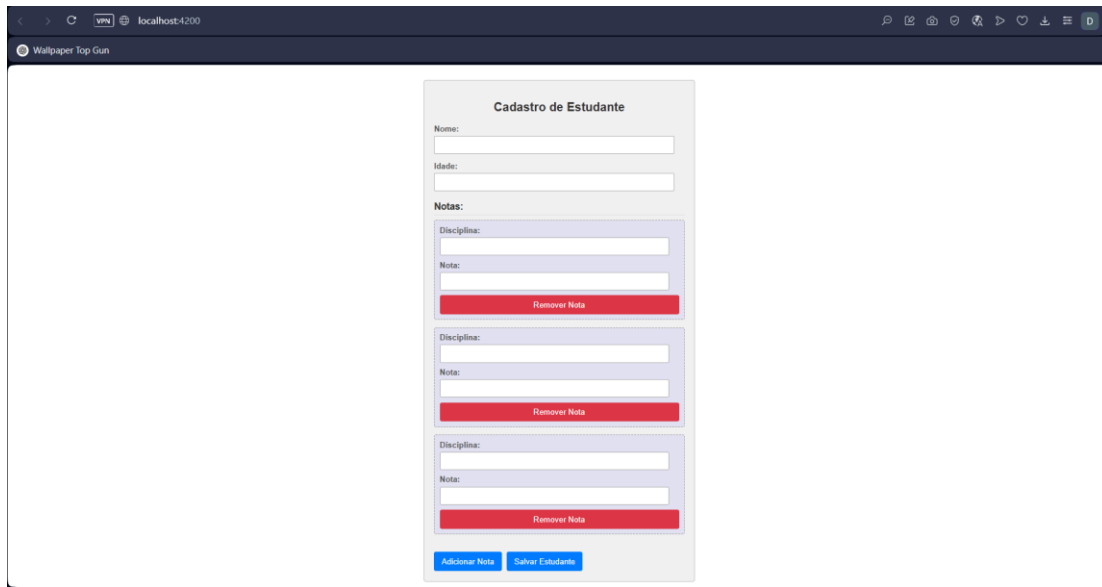


The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The page content is a form titled 'Cadastro de Estudante'. The form includes the following fields and controls:

- Nome:** A text input field.
- Idade:** A text input field.
- Notas:** A section containing:
 - Disciplina:** A text input field.
 - Nota:** A text input field.
 - Remover Nota:** A red button located below the 'Nota' field.
- Adicionar Nota:** A blue button located at the bottom left of the form.
- Salvar Estudante:** A blue button located at the bottom right of the form.

2.2. Adicionando Múltiplas Notas

O formulário permite adicionar dinamicamente novas linhas para inserir várias disciplinas e suas respectivas notas para um único estudante, utilizando o botão "Adicionar Nota".

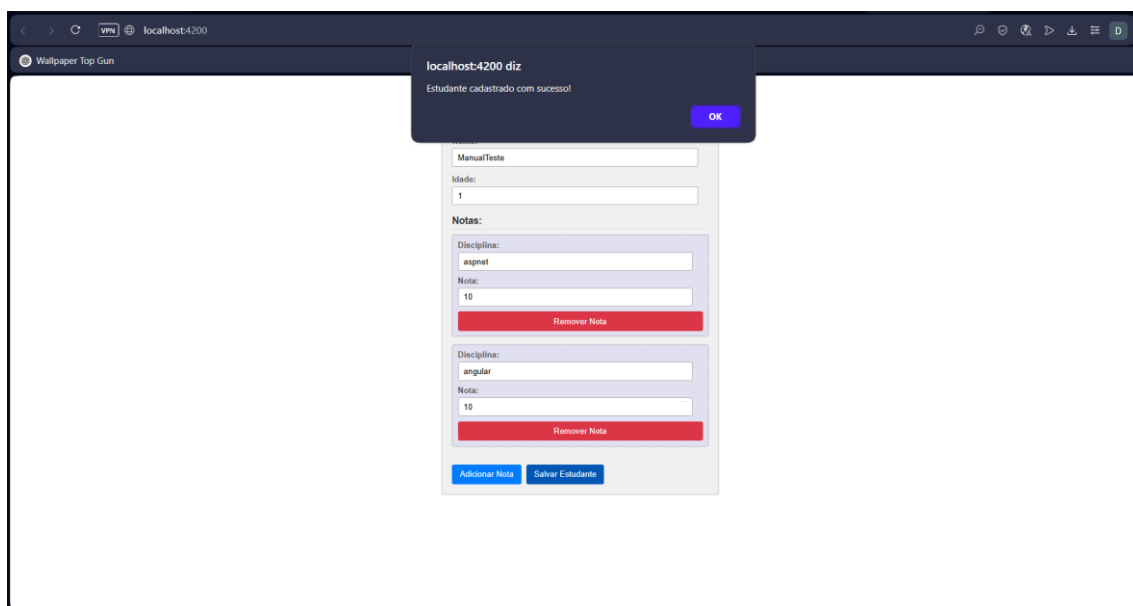


The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The page title is 'Wallpaper Top Gun'. The main content is a form titled 'Cadastro de Estudante'. The form has the following fields and buttons:

- Nome:** A text input field.
- Idade:** A text input field.
- Notas:** A section containing three identical rows for adding notes.
 - Each row has a **Disciplina:** text input field.
 - Each row has a **Nota:** text input field.
 - Each row has a red **Remover Nota** button.
- At the bottom of the form are two blue buttons: **Adicionar Nota** and **Salvar Estudante**.

2.3. Cadastro Realizado com Sucesso

Após preencher todos os dados válidos do estudante e suas notas, ao clicar no botão "Salvar Estudante", a aplicação realiza a chamada à API Back-End e, em caso de sucesso, exibe uma mensagem de confirmação.



The screenshot shows the same 'Cadastro de Estudante' form as in the previous image, but with a success message overlay. The message box is titled 'localhost:4200 diz' and contains the text 'Estudante cadastrado com sucesso!'. It has a purple **OK** button. The form fields are now populated with the following data:

- Nome:** ManualTeste
- Idade:** 1
- Notas:** Two rows are filled.
 - Row 1: **Disciplina:** aspmet, **Nota:** 10
 - Row 2: **Disciplina:** angular, **Nota:** 10
- Buttons: **Adicionar Nota** and **Salvar Estudante** are still visible at the bottom.

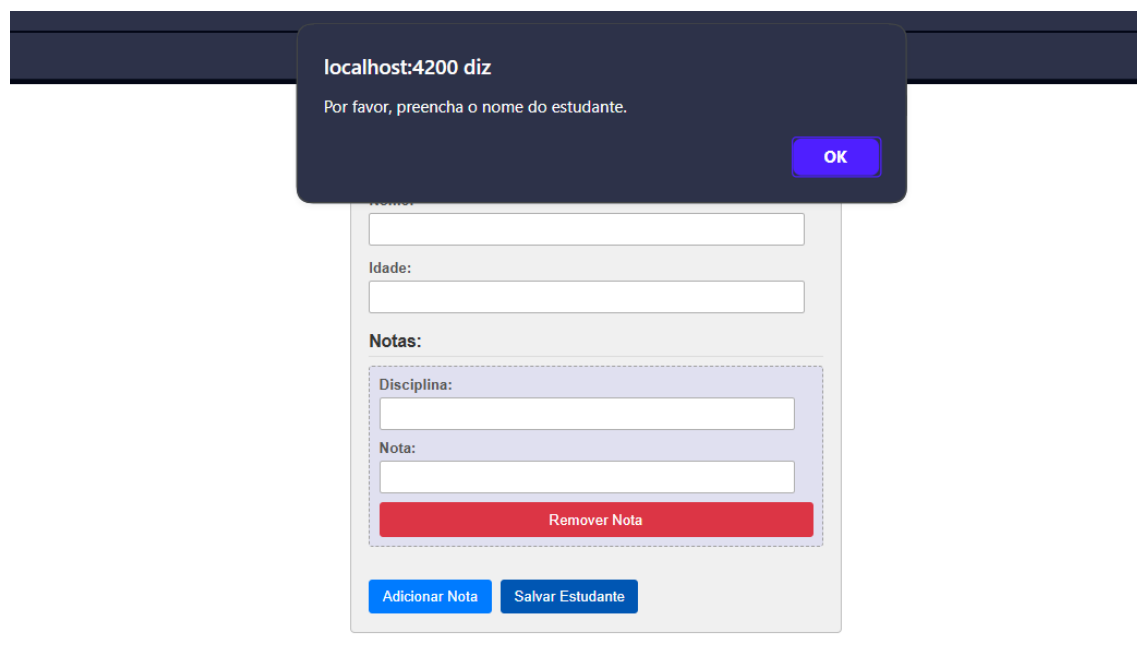
Ao clicar em “OK” o formulário deve ser limpo e ficar pronto para um novo cadastro, dessa forma facilita a inserção de dados de forma mais rápida em casos em que existam muitos alunos para serem cadastrados.

3. Validações e Cenários de Erro

A aplicação possui regras de negócio implementadas no Back-End para garantir a integridade dos dados. Esta seção demonstra o comportamento da aplicação ao tentar cadastrar dados inválidos.

3.1. Validação: Nome do Estudante Vazio

Ao tentar cadastrar um estudante com o campo 'Nome' vazio, a API rejeita a requisição e retorna uma mensagem de erro.



The screenshot displays a web application interface. At the top, a dark blue header bar is visible. Below it, a dark blue modal box contains the text "localhost:4200 diz" and "Por favor, preencha o nome do estudante." with an "OK" button. In the background, a light gray form is partially visible. The form has a "Nome:" label followed by an empty text input field. Below this is an "Idade:" label followed by an empty text input field. Further down is a "Notas:" section, which is a dashed border containing a "Disciplina:" label with an empty text input field, a "Nota:" label with an empty text input field, and a red "Remover Nota" button. At the bottom of the form are two blue buttons: "Adicionar Nota" and "Salvar Estudante".

3.2. Validação: Idade do Estudante Inválida (≤ 0)

A API valida que a idade do estudante deve ser um valor maior que zero.

The screenshot shows a web application interface with a dark blue header bar. A modal dialog box is open, displaying the text "localhost:4200 diz" and "Por favor, insira uma idade válida (maior que 0)." with an "OK" button. Below the dialog, the main form is visible. It has a text input field labeled "testelidadeMenor" containing the value "0". Below this is a label "Idade:" followed by a text input field containing "0". Underneath is a section titled "Notas:" which contains a sub-form with a "Disciplina:" label and an empty text input field, a "Nota:" label and an empty text input field, and a red "Remover Nota" button. At the bottom of the main form are two blue buttons: "Adicionar Nota" and "Salvar Estudante".

3.3. Validação: Nota Fora do Intervalo (0-10)

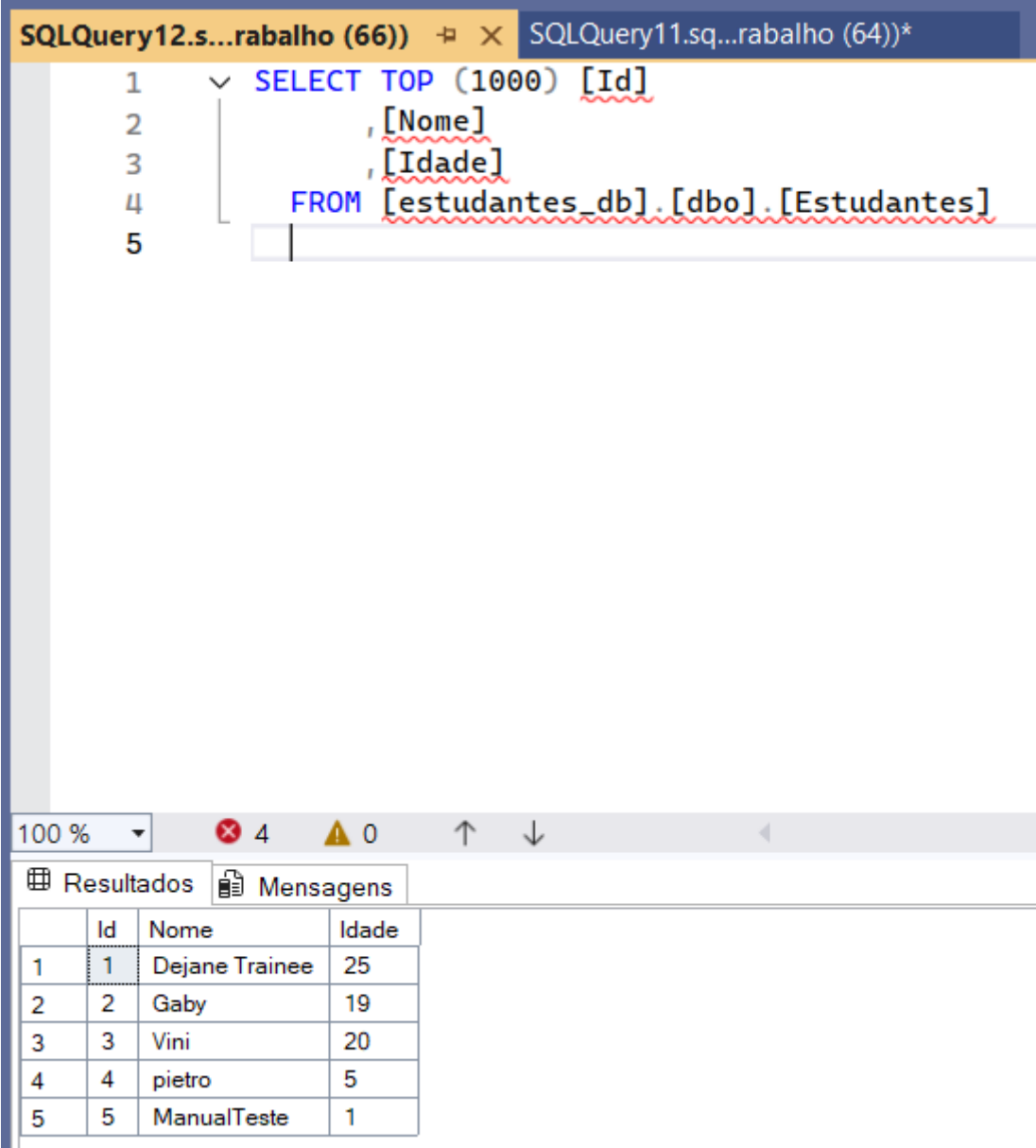
A API garante que as notas inseridas estejam dentro do intervalo de 0 a 10.

The screenshot shows the same web application interface. The modal dialog box now displays the text "localhost:4200 diz" and "Por favor, insira uma nota válida (entre 0 e 10) para todas as disciplinas." with an "OK" button. In the main form, the "testelidadeMenor" field still contains "0". The "Idade:" field now contains "10". In the "Notas:" section, the "Disciplina:" field contains "TesteNotaFora" and the "Nota:" field contains "11". The "Remover Nota" button is still present. The "Adicionar Nota" and "Salvar Estudante" buttons remain at the bottom.

4. Persistência no Banco de Dados

Esta seção comprova que os dados cadastrados através do Front-End foram devidamente persistidos no banco de dados SQL Server.

Tabela Estudantes:

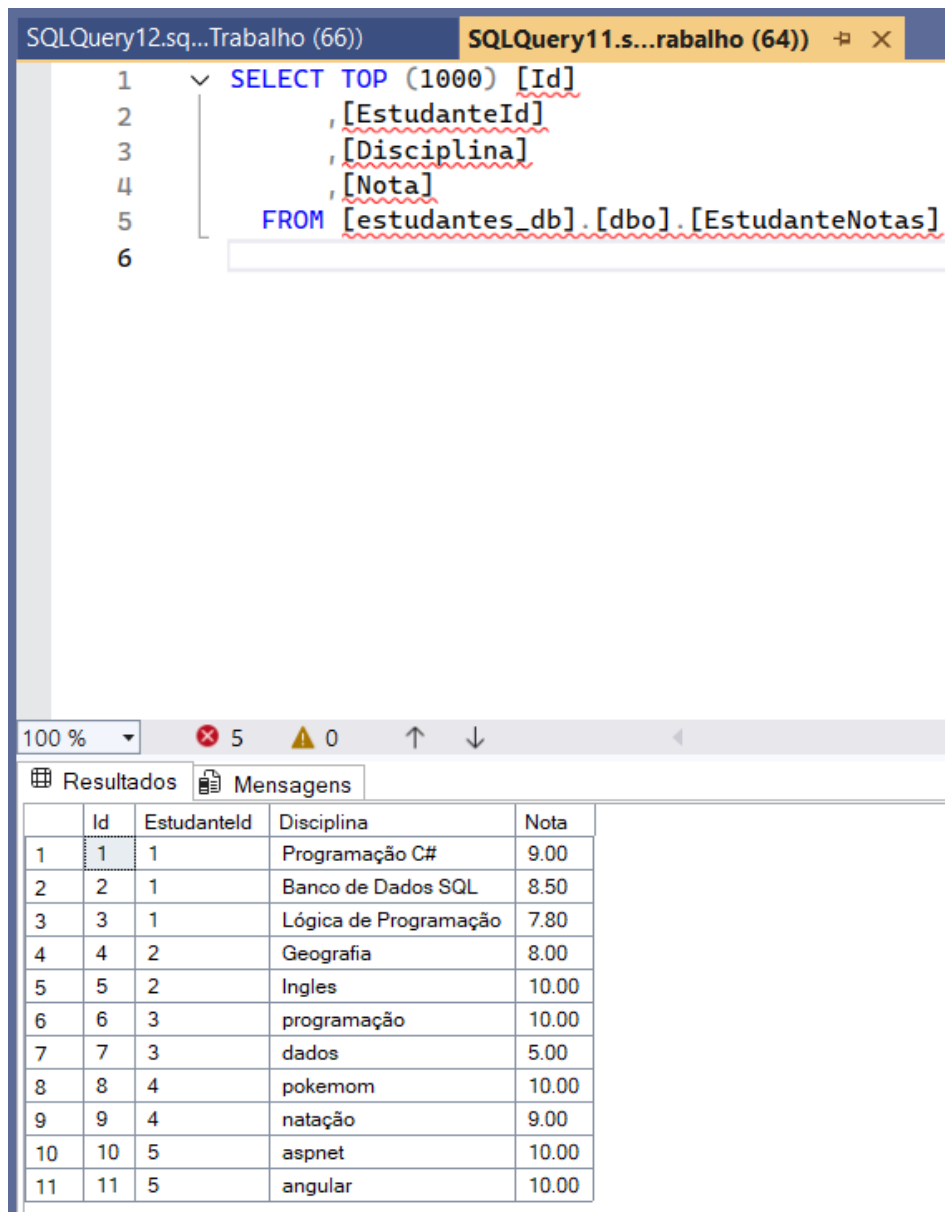


The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in the 'SQLQuery11.sq...rabalho (64))*' window. The query is a SELECT statement that retrieves the top 1000 records from the 'Estudantes' table in the 'estudantes_db' database, specifically selecting the 'Id', 'Nome', and 'Idade' columns. The bottom pane shows the 'Resultados' (Results) tab, which displays the query results in a table format. The table has four columns: 'Id', 'Nome', and 'Idade'. The results show five rows of data, with the first row having an 'Id' of 1, 'Nome' of 'Dejane Trainee', and 'Idade' of 25. The second row has an 'Id' of 2, 'Nome' of 'Gaby', and 'Idade' of 19. The third row has an 'Id' of 3, 'Nome' of 'Vini', and 'Idade' of 20. The fourth row has an 'Id' of 4, 'Nome' of 'pietro', and 'Idade' of 5. The fifth row has an 'Id' of 5, 'Nome' of 'ManualTeste', and 'Idade' of 1.

```
1 SELECT TOP (1000) [Id]
2     , [Nome]
3     , [Idade]
4 FROM [estudantes_db].[dbo].[Estudantes]
```

	Id	Nome	Idade
1	1	Dejane Trainee	25
2	2	Gaby	19
3	3	Vini	20
4	4	pietro	5
5	5	ManualTeste	1

Tabela EstudanteNota:



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in the 'SQLQuery11.s...rabalho (64)' window. The query is a SELECT statement with a TOP clause, selecting the first 1000 rows from the 'EstudanteNotas' table in the 'estudantes_db' database. The columns selected are Id, EstudanteId, Disciplina, and Nota. The bottom pane shows the 'Resultados' (Results) tab, which displays the query results in a table format. The table has 5 columns: Id, EstudanteId, Disciplina, and Nota. The results show 11 rows of data, with the first row having Id 1, EstudanteId 1, Disciplina 'Programação C#', and Nota 9.00. The last row has Id 11, EstudanteId 5, Disciplina 'angular', and Nota 10.00.

```
1 SELECT TOP (1000) [Id]
2     , [EstudanteId]
3     , [Disciplina]
4     , [Nota]
5 FROM [estudantes_db].[dbo].[EstudanteNotas]
```

	Id	Estudanteld	Disciplina	Nota
1	1	1	Programação C#	9.00
2	2	1	Banco de Dados SQL	8.50
3	3	1	Lógica de Programação	7.80
4	4	2	Geografia	8.00
5	5	2	Ingles	10.00
6	6	3	programação	10.00
7	7	3	dados	5.00
8	8	4	pokemom	10.00
9	9	4	natação	9.00
10	10	5	aspnet	10.00
11	11	5	angular	10.00

Finalizando, a aplicação foi apresentada em funcionamento real, com telas, testes, e movimentações no banco de dados. As validações foram simuladas e mostraram que as regras de negócio estão sendo respeitadas.

O sistema funciona como esperado e está pronto para evoluir. Documento encerrado com tudo o que precisava ser comprovado.