

Escola de Tecnologia Blue

Projeto Coding Girls

Reviane Cristina Lopes

Julho de 2022

Introdução

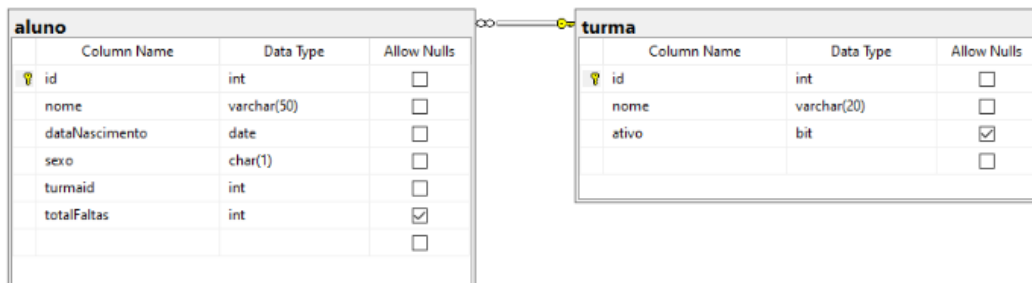
Neste projeto aplicaremos os conhecimentos adquiridos ao longo do curso intensivo de *back-end* em linguagem *C#* ministrado pela **Escola de Tecnologia Blue**, que em parceria com a **RDI**, empresa de *software* do grupo **Capgemini**, criaram o **Programa Coding Girls**.

O presente projeto visa a construção de uma *Web API*, utilizando *.NET*, um *framework* da *Microsoft*. O objetivo desde projeto visa desenvolver uma *Web API* que permita gerenciar os alunos de uma instituição de ensino, e que tenha as seguintes funcionalidades:

| <i>Endpoints</i> | Requisitos |
|---|--|
| <ul style="list-style-type: none">• Consultar todas as turmas;• Consultar turma pelo ID;• Consultar todos os alunos;• Consultar aluno pelo ID;• Incluir turmas;• Incluir alunos;• Excluir turmas;• Excluir alunos;• Atualizar turmas;• Atualizar alunos. | <ul style="list-style-type: none">• Um aluno não pode ser incluído sem uma turma;• Uma turma só pode ser excluída se não tiverem alunos cadastrados nela;• Um aluno pode ser movido de turma;• A consulta por turmas e alunos deve obedecer uma regra que é: só retornar alunos cuja condição é ativa(o); |

Diagrama EER (entidade-relacionamento estendido)

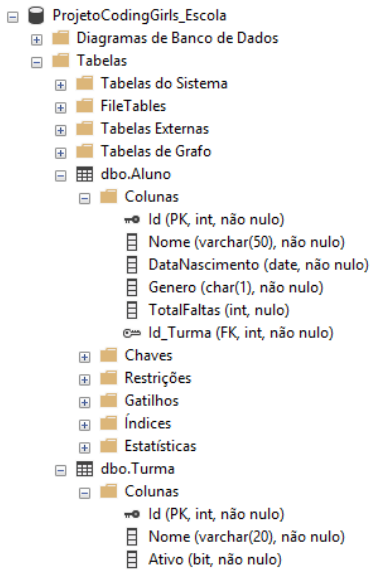
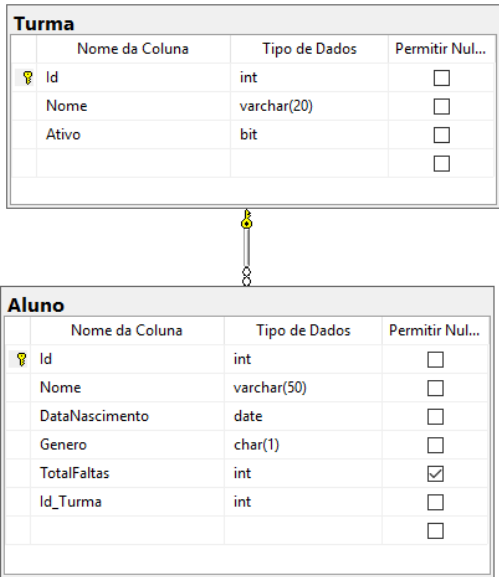
- Nome do banco: escola
- Tabelas existentes: aluno e turma.
- Descrição do relacionamento: uma turma pode conter vários alunos, porém, um aluno só pode estar vinculado a uma turma.



Desenvolvimento

Aqui teremos as especificações técnicas como os programas utilizados, e suas respectivas versões, algumas linhas de comando, principalmente as utilizadas para montar o banco de dados, e algumas configurações utilizadas para o desenvolvimento do projeto.

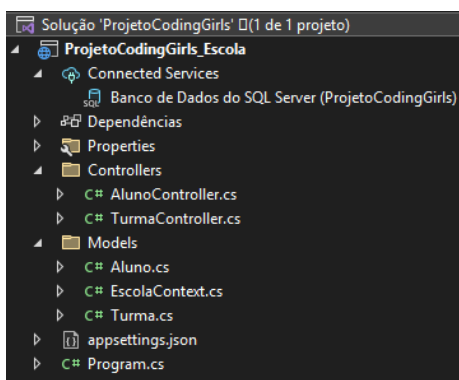
Em um primeiro momento, foi criado um bando de dados local, com o servidor *MSSQLLocalDB*, pelo programa da *Microsoft SQL Server Management Studio 18*. Abaixo temos as linhas de códigos, estrutura e o diagrama EER utilizadas para tal fim:

| Linhas de código: | |
|--|--|
| <pre>-- CRIANDO O BANDO DE DADOS CREATE DATABASE ProjetoCodingGirls_Escola; -- CRIANDO TABELA TURMA CREATE TABLE Turma (Id INT IDENTITY (1,1) PRIMARY KEY, Nome VARCHAR(20) NOT NULL, Ativo BIT NOT NULL,); -- CRIANDO TABELA ALUNO CREATE TABLE Aluno (Id INT IDENTITY (1,1) PRIMARY KEY, Nome VARCHAR(50) NOT NULL, DataNascimento DATE NOT NULL, Genero CHAR(1) NOT NULL, TotalFaltas INT NULL, Id_Turma INT FOREIGN KEY REFERENCES Turma(Id) NOT NULL,);</pre> | |
| Estrutura | Diagrama EER |
|  |  |

O programa foi desenvolvido no *Visual Studio 2022*, usando as seguintes configurações:

- Modelo: *API Web ASP.NET Core* com *HTTP*;
- Estrutura: *Framework .NET 6.0*;
- Controladores: *API Controller with actions using Entity framework v1.0.0.0*;
- Classe para contextualizar o banco de dados: *DbContext*;
- Classes: *Models* Turma e Alunos.

Abaixo temos a estrutura hierárquica do programa, para detalhes sobre o código vide o projeto no *GitHub*: https://github.com/RvnLps/ProjetoCodingGirls_Final.git.



As tabelas do banco de dados local foram conectadas com as variáveis do projeto, como pode ser vistas nas respectivas classes da pasta *Models*, *Aluno* e *Turma*, e a relação entre estas também foram efetuadas em *EscolaContext*, presente na mesma pasta.

Após teste e validação das funcionalidades, tanto pelo *Swagger* quanto pelo *Postman*, a próxima etapa, foi criar um banco de dados online na *Azure* e vinculá-lo ao projeto. Este banco foi criado da mesma forma que o local, utilizando as mesmas linhas de códigos, sua estrutura pode ser vista abaixo. As conexões foram realizadas e validadas aplicando os mesmos testes utilizados com o banco de dados local. Posteriormente, o projeto também foi publicado na *Azure*, para se tornar uma *Web API* efetivamente online, como demonstrado abaixo.

| Banco de dados online | Grupo de recursos na Azure | | | | | | | | | | |
|-------------------------------------|--|------|------|--------------------|-------------------|-------------------------------------|--------------------|-----------------------------|---------------------------------|----------------------|------------|
| | <table><thead><tr><th>Nome</th><th>Tipo</th></tr></thead><tbody><tr><td>ProjetoCodingGirls</td><td>Grupo de recursos</td></tr><tr><td>PCGRoot (projetcodinggirls/PCGRoot)</td><td>Banco de dados SQL</td></tr><tr><td>ProjetoCodingGirlsEscolaAPI</td><td>Serviço de Gerenciamento de API</td></tr><tr><td>Azure subscription 1</td><td>Assinatura</td></tr></tbody></table> | Nome | Tipo | ProjetoCodingGirls | Grupo de recursos | PCGRoot (projetcodinggirls/PCGRoot) | Banco de dados SQL | ProjetoCodingGirlsEscolaAPI | Serviço de Gerenciamento de API | Azure subscription 1 | Assinatura |
| Nome | Tipo | | | | | | | | | | |
| ProjetoCodingGirls | Grupo de recursos | | | | | | | | | | |
| PCGRoot (projetcodinggirls/PCGRoot) | Banco de dados SQL | | | | | | | | | | |
| ProjetoCodingGirlsEscolaAPI | Serviço de Gerenciamento de API | | | | | | | | | | |
| Azure subscription 1 | Assinatura | | | | | | | | | | |

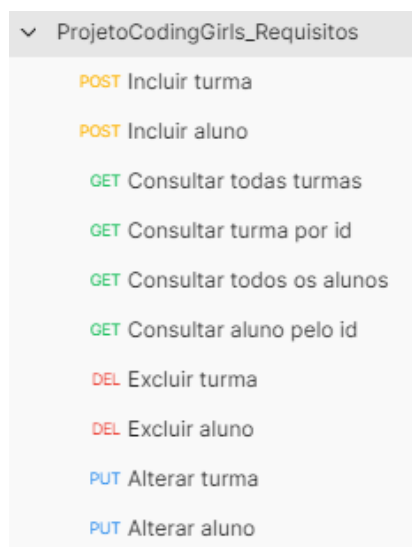
Resultado

Os testes de requisitos foram realizados mantendo o padrão de exibição, ou seja pela ferramenta *Swagger* para visualização da *Web API*, abaixo temos sua interface, os resultados dos testes apresentados foram realizados com o *Postman*. Estes testes não foram realizados diretamente pela *Web API* da *Azure*, pois sua apresentação é simplista, permitindo apenas visualizar, de forma não estruturada, os dados do banco de dados, e não permitir o uso de comando como *Post/Put/Delete*.

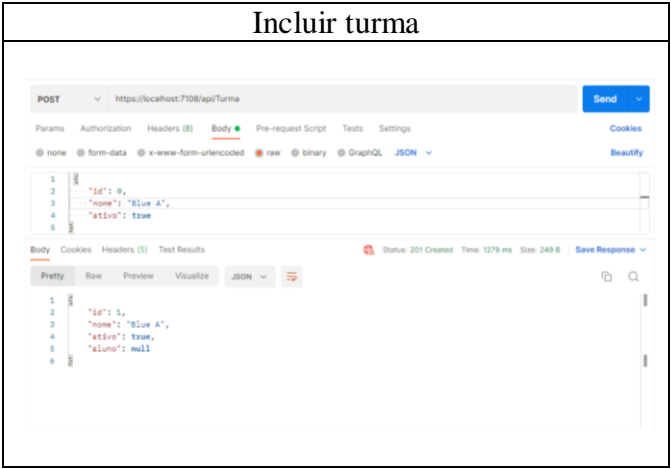
Abaixo temos a interface apresentada pelo *Swagger*, vale ressaltar que esses testes foram realizados utilizando-se o banco de dados online.



Os testes dos requisitos solicitados foram efetivamente realizados usando o programa *Postman*, o roteiro de teste se encontra na pasta do projeto no *GitHub*, abaixo temos as funções aplicáveis seguidas dos resultados dos testes:



Os testes foram iniciados incluindo uma turma, preenchendo os campos solicitados:

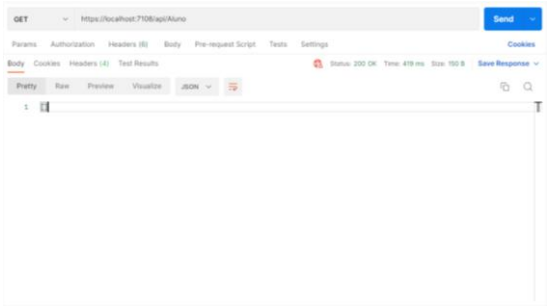
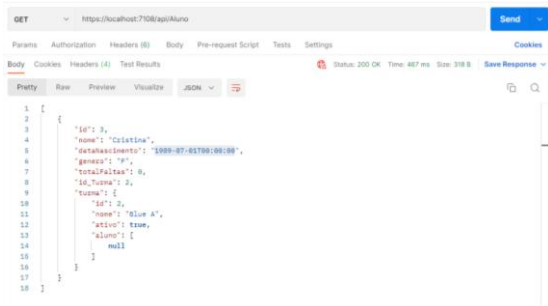


Ao tentar incluir um aluno, e preencher o id da turma com uma que ainda não foi cadastrada, aparece um erro correspondente a id inexistentes; se houver uma turma com id cadastrado, aparecerá os dados de cadastro

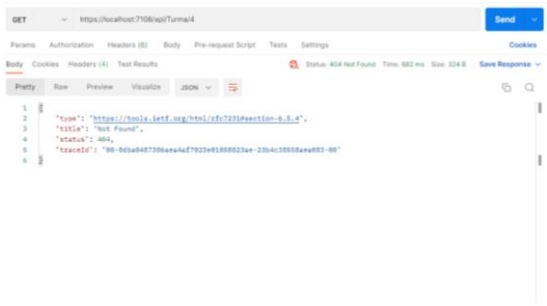
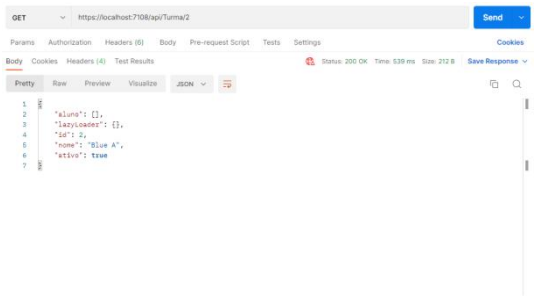
| Incluir aluno | |
|----------------------|------------------|
| | |
| Turma não cadastrada | Turma cadastrada |

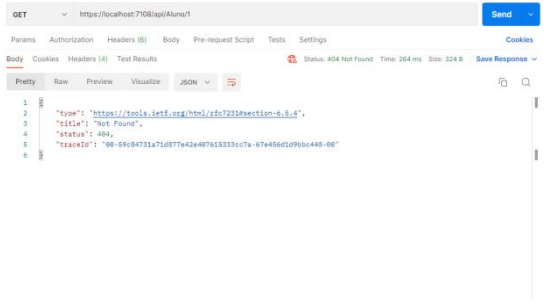
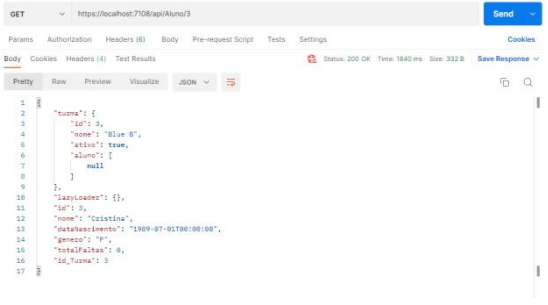
As consultas aos dados gerais das tabelas são apresentados em forma de cascada; no caso da tabela turma, apresenta os alunos cadastrados nesta; no caso dos alunos, apresenta tanto a turma quanto os outros alunos cadastrados na mesma turma.

| Consultar todas as turmas | |
|---------------------------|-------------------------------|
| | |
| Banco de dados vazio | Retorno com dados cadastrados |

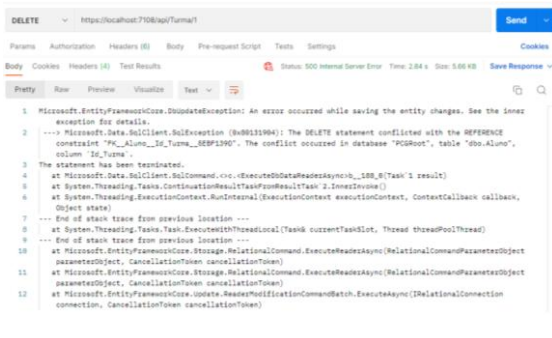
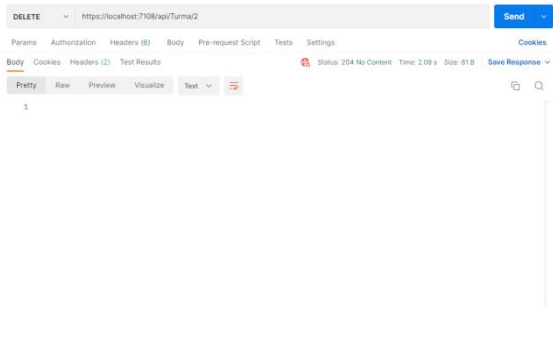
| Consultar todos os alunos | |
|---|--|
|  |  |
| Banco de dados vazio | Retorno do dado cadastrado |

As consultas por id, tanto da tabela turma quanto da tabela aluno, apresentam seus respectivos dados, e aqui também apresentam uma cascata de informações vinculadas.

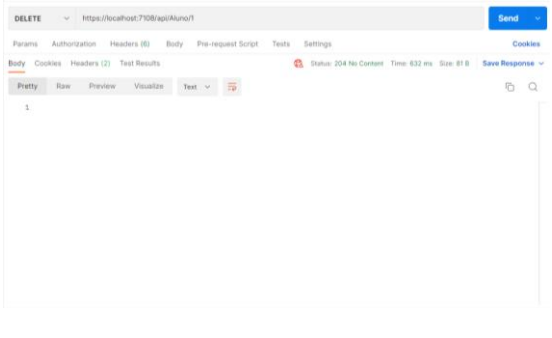
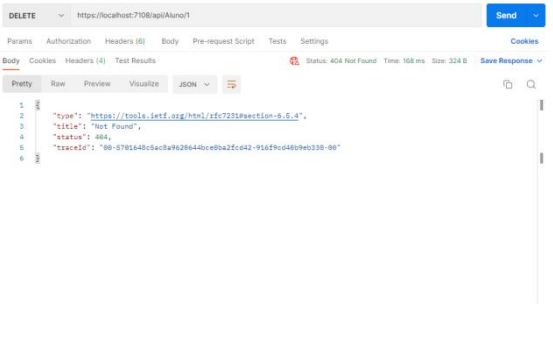
| Consultar turmas por id | |
|--|---|
|  |  |
| Id que não corresponde a um cadastro | Id que corresponde a um cadastro |

| Consultar aluno por id | |
|---|--|
|  |  |
| Id que não corresponde a um cadastro | Id que corresponde a um cadastro |

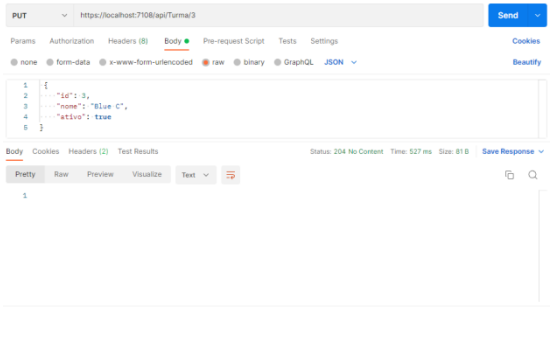
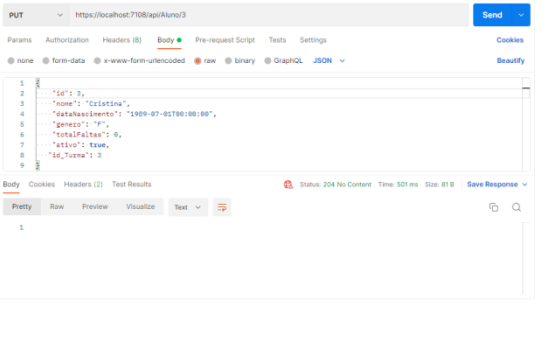
Ao tentar excluir uma turma, aparecerá um erro devido ao fato da classe aluno estar vinculada a ela, com isso, só será possível excluir uma turma quando esta estiver vazia, ou seja, será necessário excluir todos os alunos desta ou transferi-los para outra turma.

| Excluir turma | |
|---|--|
|  |  |
| Excluir turma com alunos cadastrados | Excluir turma vazia |

A exclusão de um aluno apresentará erro caso o id não possua aluno cadastrado.

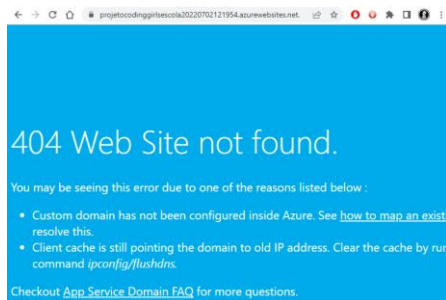
| Excluir aluno | |
|--|---|
|  |  |
| Excluir aluno cadastrado | Excluir id sem cadastro de aluno |

A alteração de informações de uma turma ou aluno acontece utilizando seu id, e por esse método, pode-se alterar um aluno de turma

| Alterar turma | Alterar aluno |
|---|--|
|  |  |
| Excluir aluno cadastrado | Excluir turma vazia |

Como mencionado anteriormente, este projeto se encontra online, e pode ser acessado por: <https://projetcodinggirlsescola20220702121954.azurewebsites.net>. No entanto, ao acessar este *link* diretamente, será apresentado o "erro 404", pois é necessário definir o caminho desejado. Para isso, deve-se adicionar ao fim do *link*: *"/api/Aluno"*, para visualizar dos dados contidos na tabela aluno; ou *"/api/Turma"*, para visualizar os dados da tabela turma; ou, também pode-se adicionar "/" e o número de um *id* em ambos as situações caso desejar, mas irá apresentar erro se não houver dados no cadastro correspondente. Abaixo temos as interfaces mencionadas:

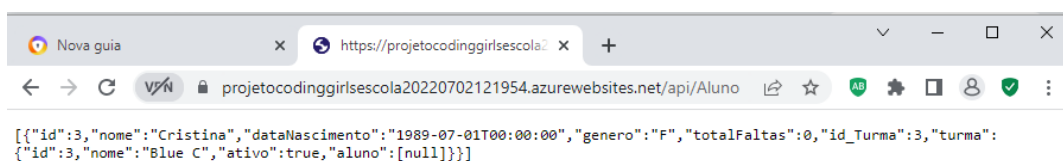
Acesso direto ao *link*



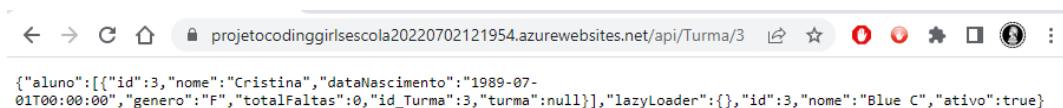
Acesso adicionando */api/Turma* ao *link*



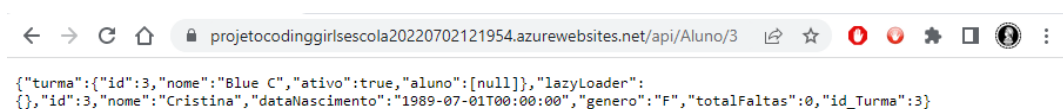
Acesso adicionando */api/Aluno* ao *link*



Acesso adicionando */api/Turma/3* ao *link*



Acesso adicionando */api/Aluno/3* ao *link*



Conclusões

Até o momento o projeto se encontra funcional, e atende os pré-requisitos de funcionalidade. Todavia, muitas melhorias podem ser feitas, como: melhor apresentação dos dados; mensagens de erros específicas; abrir o projeto online apresentando uma interface que possibilite a manipulação de dados, ou seja, melhorar o *front-end* do projeto.