Teste prático para Dev Backend .NET C# 2021

Caíque Fortunato

OBJETIVO

Desenvolver uma API REST HTTP utilizando C# Asp NET para desenvolvimento de cadastro de alunos e responsáveis de uma escola.

REQUISITOS

Cadastrar e listar informações de Alunos e Responsáveis seguindo a estrutura definida pelo o *endpoint* da documentação com implementação de autenticação. Além disso, seguir os seguintes requisitos funcionais:

- Os alunos podem ser do ensino fundamental ou infantil.
- O e-mail é obrigatório para alunos do ensino fundamental.
- Cada aluno pode ter N responsáveis (Mãe, Pai, Tio, Avó).
- O e-mail é obrigatório para os responsáveis

IMPLEMENTAÇÃO

A implementação baseou-se nas seguintes tecnologias: Framework ASP Net Core na versão 3.1 e banco de dados MySQL. Demais tecnologias foram derivadas das citadas anteriormente como exemplo do NUnit para realização de testes unitários, AspNetCore Authentication e Entity Framework Core, por exemplo para criação da segurança e persistência, respectivamente.

BANCO DE DADOS

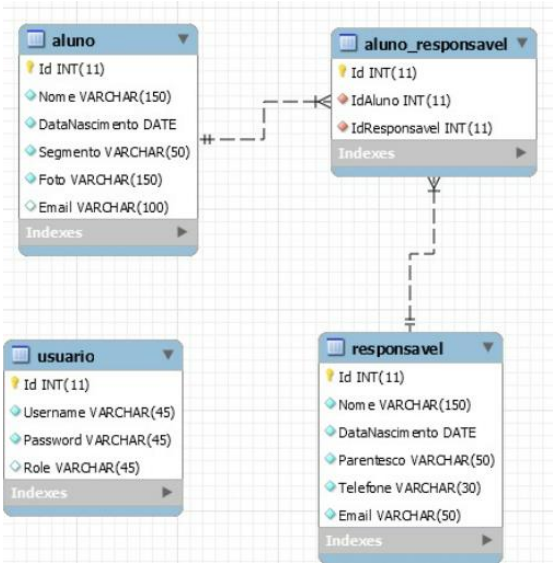
A modelagem do banco consistiu-se na criação de quatro tabelas: “alunos”, “usuario”, “aluno_responsável” e “responsavel” com os atributos conforme definição dos *endpoitns* da especificação e de estratégia para programação.

Sendo assim, as tabelas que refletem os *endpoints* solicitados são:

- Aluno → id, Nome, DataNascimento, Segmento, Foto e E-mail;
- Responsável → id, Nome, DataNascimento, Parentesco, Telefone e E-mail;

Já as outras tabelas utilizadas para auxílio são:

- AlunoResponsável → id, idAluno e idResponsável
    - Importante para o armazenamento da relação de n .. n entre as duas entidades.
- Usuário → id, Username, Password e Role
    - Criação de usuário para acesso ao sistema, devido a autenticação.



ESTRUTURA DO CÓDIGO

O código segue a estrutura de um projeto básico de Web Api do .NET tendo os seguintes diretórios principais: Controllers, DTO, Entities e Services.

Em *Controllers* há três controladores: Acesso, Aluno e Responsável sendo que o primeiro é utilizado para a requisição de acesso do usuário e os demais representam as ações dos *endpoints* solicitados.

A utilização do *Design Pattern* Data Transfer Object (DTO) foi utilizada para transportar dados entre os diferentes componentes do sistema, além de possuir algumas verificações como obrigatoriedade de e-mail de responsáveis e alunos da educação fundamental além dos tipos de escolaridade dos alunos, por exemplo.

Em *Entities* há definição dos objetos conforme banco de dados cujo ações são persistidas através da classe DBContext, que utiliza o framework de Entity Framework.

Por fim, em *Services* há uma classe chamada *TokenService* que gera Tokens de acesso para autenticação de usuários para realização de chamadas post e get.

ENDPOINTS E ESTRATÉGIAS DE IMPLEMENTAÇÃO

A API possui os *endpoins* solicitados que estão distribuídos nos três controladores de forma que:

1) **Responsável**
    a. Métodos GET:
        i. GetResponsaveis
    b. Métodos POST:
        i. InsereResponsavel {Id, Nome, DataNascimento, Telefone, Parentesco, Emai}
2) **Aluno**
    a. Métodos GET:
        i. GetAlunos
        ii. GetAlunoByNome {Nome}
        iii. GetAlunoBySegmento {Segmento}
        iv. GetAlunoByResponsavel {Responsavel}
    b. Métodos POST:
        i. InsereAluno {Id, DataNascimento, Segmento, Foto (*), Email, {Lista de ids dos responsáveis} }

    (*) Para o desenvolvimento da API foi considerado de que a foto já existe no servidor de forma que uma outra requisição específica para o envio do arquivo já foi executada com sucesso retornando para o "front" o caminho da imagem inserida. Com o caminho definido ele é enviado na requisição implementada neste trabalho através do parâmetro Foto.
3) **Acesso ***
    a. Métodos POST:
        i. Login {Username, Password, Role**}

    (*) O cadastro inicial do usuário foi realizado diretamente no banco de dados. Não foi implementado o cadastro inicial de um usuário por fugir da proposta do trabalho e pela limitação do tempo.

    (**) A Role é um item opcional que poderia ser utilizada em uma evolução como parâmetros de perfis de usuários para acessos específicos.

Importante: Os *endpoints* funcionam no formato JSON para envio e recebimento de informações.

AUTENTICAÇÃO / SEGURANÇA

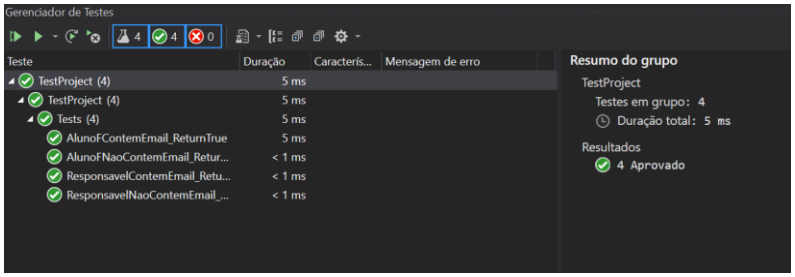
A API requer autenticação para todos os *endpoints* listados acima. Conforme citado anteriormente, foi utilizado o Bearer e JWT além da autenticação do usuário através de usuário e senha recuperando cadastro registrado no banco de dados (tabela usuário). A chave utilizada está na classe Settings já criptografada pelo padrão MD5.

Na pasta Services já o serviço de Token que vai gerar um token válido de duas horas para o usuário que se autenticar através do *controller* acesso. Nessa classe algumas tecnologias e conceitos como ClaimsIdentity e JwtSecurity são utilizados para geração do código que será retornado para o usuário.

TESTES AUTOMATIZADOS

Foi utilizado no projeto testes automatizados unitários através do framework NUnit. Os testes verificam a obrigatoriedade do e-mail para responsáveis e alunos do segmento fundamental.

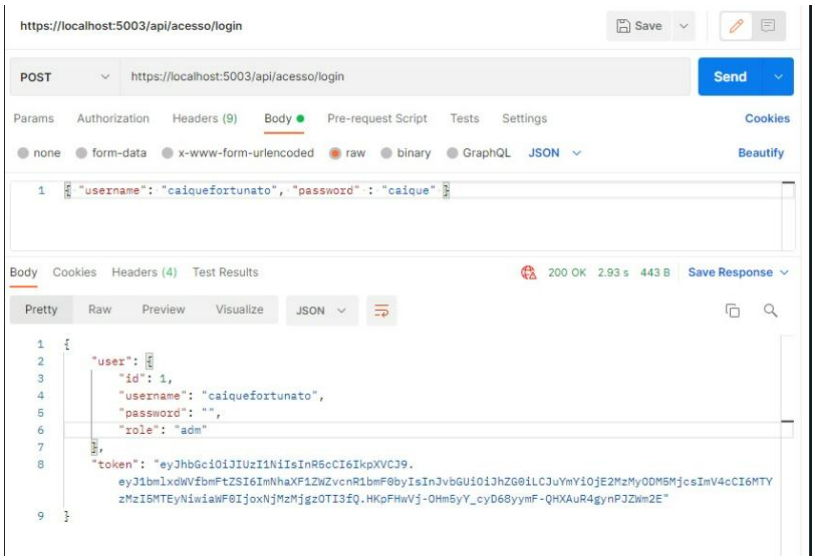
Todos os arquivos de testes estão na solução “TestProject” na classe UnitTest1.cs



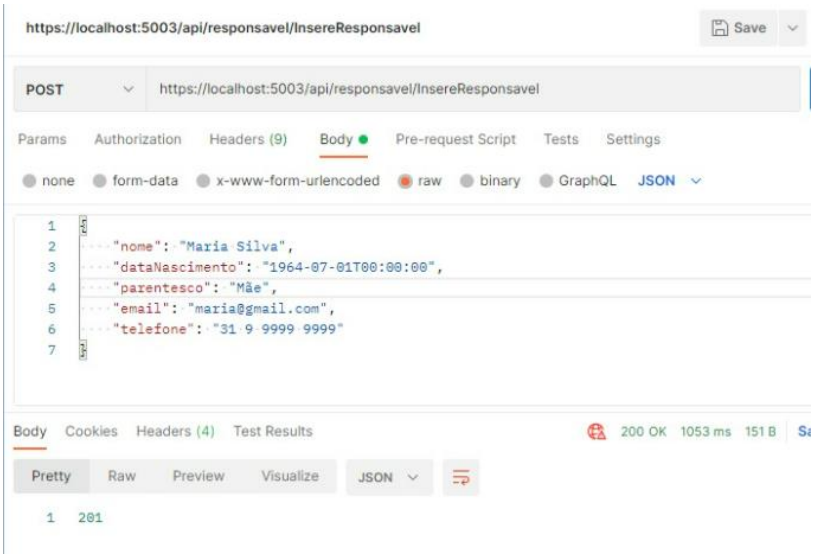
EXEMPLO DE FUNCIONAMENTO

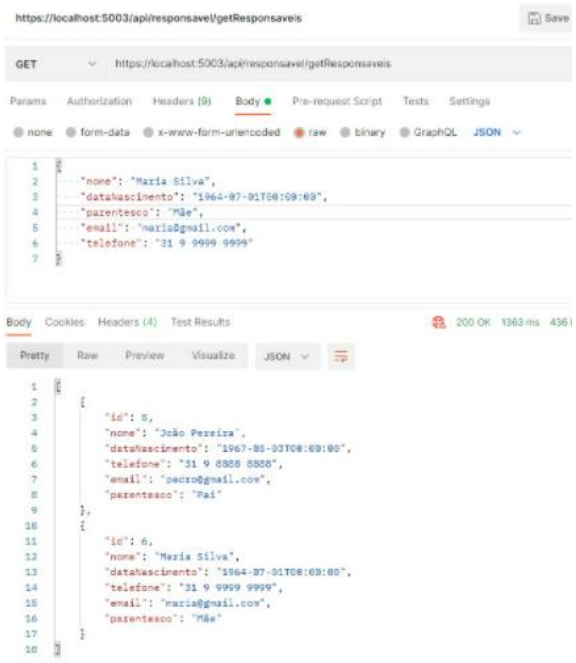
AUTENTICAÇÃO

A autenticação é realizada com usuário e senha cadastrados manualmente no banco, conforme especificado anteriormente na presente documentação. Para testes, foi criado o usuário de username: caiquefortunato e password: caique

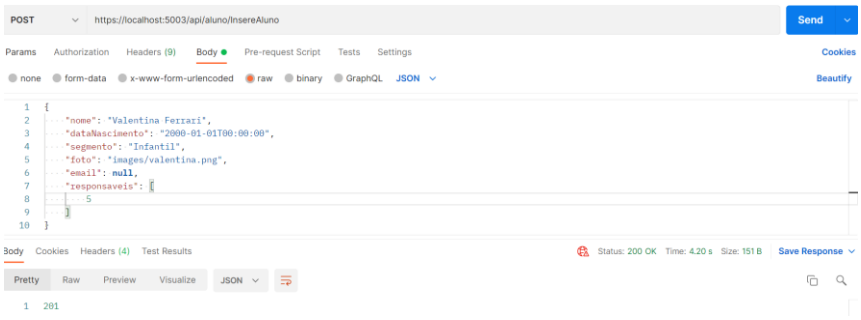


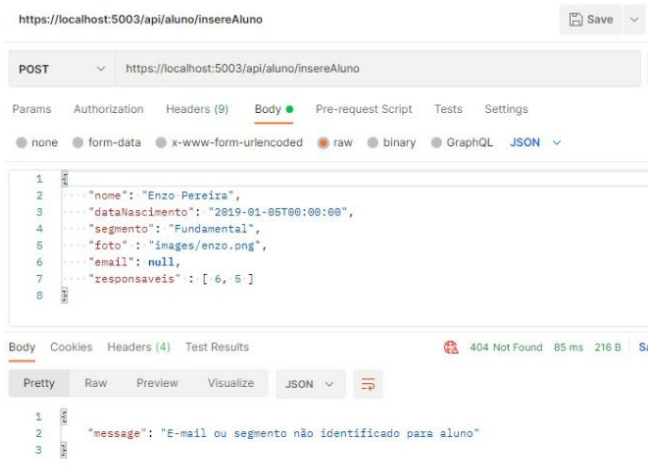
RESPONSAVEL

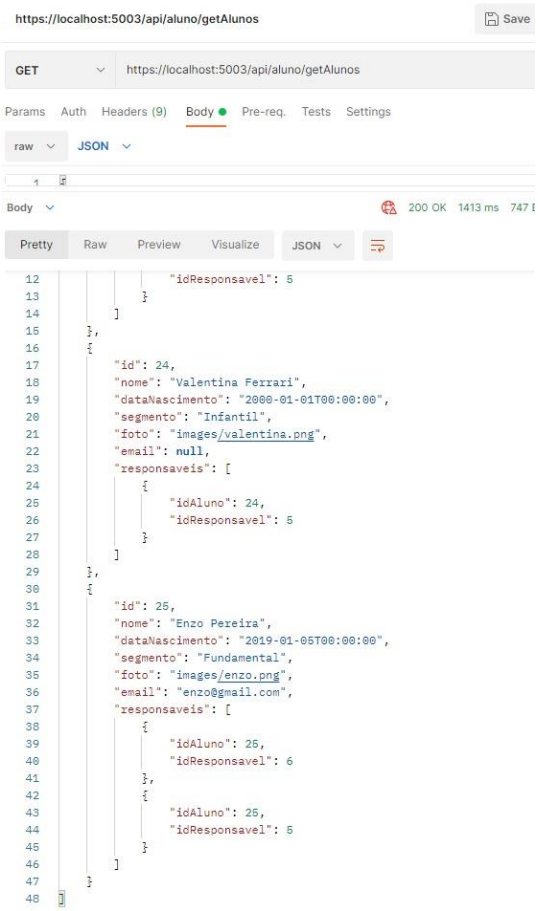



https://localhost:5003/api/responsavel/getResponsaveis
Save
GET
https://localhost:5003/api/responsavel/getResponsaveis
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"nome": "Maria Silva",
"dataNascimento": "1964-07-01T00:00:00",
"parentesco": "Mãe",
"email": "maria@gmail.com",
"telefone": "31 9 9999 9999"
}
Body Cookies Headers (4) Test Results
200 OK 1363 ms 436 B
Pretty Raw Preview Visualize JSON
[
{
"id": 5,
"nome": "João Pereira",
"dataNascimento": "1967-05-03T00:00:00",
"telefone": "31 9 8888 8888",
"email": "pedro@gmail.com",
"parentesco": "Pai"
},
{
"id": 6,
"nome": "Maria Silva",
"dataNascimento": "1964-07-01T00:00:00",
"telefone": "31 9 9999 9999",
"email": "maria@gmail.com",
"parentesco": "Mãe"
}
]

ALUNO


POST
https://localhost:5003/api/aluno/InsereAluno
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"nome": "Valentina Ferrari",
"dataNascimento": "2000-01-01T00:00:00",
"segmento": "Infantil",
"foto": "images/valentina.png",
"email": null,
"responsaveis": [
5
]
}
Body Cookies Headers (4) Test Results
Status: 200 OK Time: 4.20 s Size: 151 B
Save Response
Pretty Raw Preview Visualize JSON
201


https://localhost:5003/api/aluno/insereAluno
Save
POST
https://localhost:5003/api/aluno/insereAluno
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"nome": "Enzo Pereira",
"dataNascimento": "2019-01-05T00:00:00",
"segmento": "Fundamental",
"foto" : "images/enzo.png",
"email": null,
"responsaveis" : [6, 5]
}
Body Cookies Headers (4) Test Results
404 Not Found 85 ms 216 B
Pretty Raw Preview Visualize JSON
{
"message": "E-mail ou segmento não identificado para aluno"
}


https://localhost:5003/api/aluno/getAlunos
Save
GET
https://localhost:5003/api/aluno/getAlunos
Params Auth Headers (9) Body Pre-req. Tests Settings
raw JSON
Body
200 OK 1413 ms 747 B
Pretty Raw Preview Visualize JSON
"idResponsavel": 5
}
]
},
{
"id": 24,
"nome": "Valentina Ferrari",
"dataNascimento": "2000-01-01T00:00:00",
"segmento": "Infantil",
"foto": "images/valentina.png",
"email": null,
"responsaveis": [
{
"idAluno": 24,
"idResponsavel": 5
}
]
},
{
"id": 25,
"nome": "Enzo Pereira",
"dataNascimento": "2019-01-05T00:00:00",
"segmento": "Fundamental",
"foto": "images/enzo.png",
"email": "enzo@gmail.com",
"responsaveis": [
{
"idAluno": 25,
"idResponsavel": 6
},
{
"idAluno": 25,
"idResponsavel": 5
}
]
}
]