



Desenvolvimento da Aplicação

Siga estas instruções durante o processo:

1. Baixar o repositório adequado
2. Ler as instruções e segui-las adequadamente. Contatar equipe '**Dev.Five**' antes de prosseguir caso não esteja claro o que deve ser feito.
3. Ao finalizar suas tarefas, fazer upload dos arquivos que foram modificados ou adicionados ao projeto dentro da pasta apropriada com seu nome.

Repositório do Projeto:

<https://github.com/mary0077/escola>

Instalar todas as dependências através do terminal

npm install

Para aplicar migrações no BD usando o prisma, execute:

npx prisma migrate dev

Instale as dependências do Swagger:

npm install swagger-jsdoc swagger-ui-express

Rodar o projeto:

node app.js

Arquivo com todas as APIs e seus testes criadas neste projeto (importável no Postman ou swagger):

testes/testedevfive.json

Demonstração do Site:

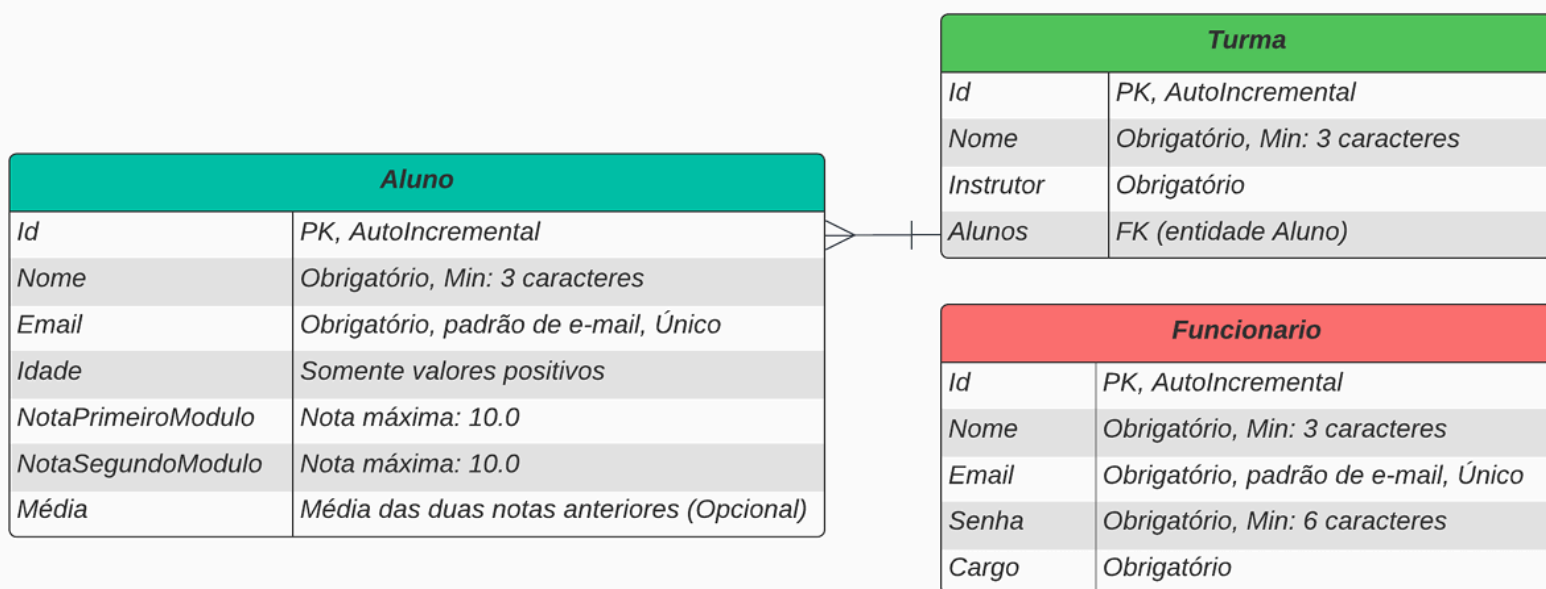
<https://genapiescola.onrender.com/auth/login>

Utilização/Documentação Swagger:

<https://genapiescola.onrender.com/api-docs/>



Modelo Físico do Banco de Dados (usar como referência)



Este banco de dados relacional possui três tabelas principais: **Aluno**, **Turma** e **Funcionário**.



Inicializando o Servidor do Site Corretamente

1. Banco de Dados PostgreSQL:

A aplicação está conectada ao banco de dados PostgreSQL hospedado no Render.

A URL de conexão do banco de dados é definida através da variável de ambiente `DATABASE_URL`, que está especificada no arquivo `.env` da aplicação.

`JWT_SECRET`: Chave secreta utilizada para a geração e verificação de tokens JWT (JSON Web Token) para autenticação de usuários

.

Configuração da Porta:

A aplicação será iniciada na porta `3000`, definida pela variável de ambiente `PORT` ou utilizando a porta `3000` como padrão.

Node:

Com as configurações acima feitas corretamente, basta abrir o terminal e rodar o comando **`node app.js`**

Servidor Local:

<http://localhost:3000>

Acessando Tela Principal:

<http://localhost:3000/auth/login>

Documentação Swagger:

<http://localhost:3000/api-docs>



Inicializando o Servidor do Site ou acessando link hospedado no Render

Tela inicial (login):

Aqui para se registrar!'. At the bottom of the form is a yellow 'Logar' button."/>

DEV.FIVE

Faça seu login aqui!

Email:

Senha:

Novo por aqui ?
Clique [Aqui](#) para se registrar!

Logar

OBS: Caso não tenha um cadastro, clique “[Aqui](#)” e realize o seu cadastro.

Tela de Cadastro:

DEV.FIVE

Registre-se aqui!

Nome:

Cargo:

Email:

Senha:

Registrar



Exemplos de teste no POSTMAN

Método HTTP GET - Só utiliza a barra de endereço e serve para definir o URL da rota que você deseja acessar e buscar dados do servidor.

Método HTTP POST: Usado para criar novos recursos ou enviar dados ao servidor, como informações de um formulário ou criação de um novo registro no banco de dados.

Exemplo testado do envio de novos dados ao banco de dados:

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/alunos`. The request body is a JSON object with the following fields:

```
{  "nome": "João Silva",  "email": "joao.silva@example.com",  "idade": 15,  "nota_primeiro_semestre": 8.5,  "nota_segundo_semestre": 9.0}
```

The response is also in JSON format, showing the created record with an assigned ID and timestamps:

```
{  "id": 1,  "nome": "João Silva",  "email": "joao.silva@example.com",  "idade": 15,  "NotaPrimeiroModulo": 8.5,  "NotaSegundoModulo": 9,  "Media": null,  "updatedAt": "2024-10-21T14:40:36.634Z",  "createdAt": "2024-10-21T14:40:36.634Z"}
```



Exemplo testado do envio de novos dados ao banco de dados:

Import POST http://localhost:3000/turmas

HTTP http://localhost:3000/turmas

POST http://localhost:3000/turmas

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1 {
2   "nome": "Turma A",
3   "instructor": 1,
4   "Alunos": 30
5 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "nome": "Turma A",
4   "instructor": 1,
5   "Alunos": 30,
6   "updatedAt": "2024-10-22T11:52:13.060Z",
7   "createdAt": "2024-10-22T11:52:13.060Z"
8 }
```

are



OBS: Email do aluno e funcionário são tidos como únicos, ou seja, será impossível a realização de um novo cadastro com o email igual.

Exemplo testado:

Search Postman

You are using the Lightweight API Client, sign in or create an account to work with collections

POST http://localhost:3000/fui POST http://localhost:3000/alunos + ...

HTTP http://localhost:3000/alunos

POST http://localhost:3000/alunos

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▾

```
1 {
2   "nome": "João Gomes",
3   "email": "joao.silva@example.com",
4   "idade": 20,
5   "nota_primeiro_semestre": 8.5,
6   "nota_segundo_semestre": 9.0
7 }
8
9
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "error": "E-mail já cadastrado"
3 }
```



Atenção!

Se você encontrar algum problema ou tiver sugestões, sinta-se à vontade para entrar em contato conosco.