Guia Definitivo para Instalação

Augusto Emerson FP Junior

5 de setembro de 2023

1 Contexto

Uma loja vende instrumentos musicais mágicos. Cada instrumento tem um nome, tipo (corda, sopro, percussão), preço e uma propriedade mágica (ex: flauta que faz o ouvinte dormir). A loja quer uma API RESTful para gerenciar seu inventário.

2 Requisitos

- Implementar uma API RESTful com endpoints para adicionar, remover, atualizar e listar instrumentos.
- Armazenar informações dos instrumentos em um banco de dados de sua escolha.
- Criar uma funcionalidade que calcule o preço total de todos os instrumentos de um tipo específico (ex: todos os instrumentos de corda).
- Criar uma funcionalidade de busca textual (texto exato) para as propriedades mágicas (ex: ao buscar "dormir", deve retornar todos os instrumentos com poderes que contenham "dormir")
- O código deve ser bem organizado, comentado e seguir padrões de desenvolvimento, como os princípios SOLID.
- Implementar uma camada de autenticação básica para acessar os endpoints da API.
- O projeto deve ser feito em .NET 6.0 ou 7.0 e deve executar em windows/linux.
- Deve ser utilizado o Entity Framework Core como ORM.

3 Entrega

- Código fonte da aplicação (pode ser um repositório GIT público)
- Documentação em Markdown ou PDF explicando como executar o código, incluindo as rotas da API.
- Indicação da versão e tipo do banco de dados utilizado.
- Instruções para criar o banco de dados e tabelas necessárias (pode ser um script SQL ou um README específico).

4 Configuração do Ambiente

Para a configuração do ambiente é necessário fazer a instalação de alguns requisitos.

4.1 Requisitos

- Git versão +2.41.0
- .NET 7.0
- Docker +24.0.5
- SQL Server 2022

4.2 Sistema testado

- Windows 10
- Git version 2.41.0.windows.2
- .NET 7.0.400
- Docker 24.0.5, build ced0996
- SQL Server 2022 based on Ubuntu 22.04

5 Instalação

Primeiro de tudo, é necessário clonar o repositório da aplicação no seu computador. O repositório você pode encontrar apertando **aqui**, ou, se preferir acessando o url *https://github.com/aejunior/SonsMagicos*.

Para fazer o clone usando a ferramenta Git, execute no seu terminal o comando:

```
$ git clone https://github.com/aejunior/SonsMagicos.git
```

Após feita a clonagem, você terá o código fonte da aplicação na sua máquina. Para executá-lo, é necessário fazer a instalação dos requisitos na seção de Configuração de Ambiente.

5.1 Banco de Dados

O primeiro passo, é configurar seu **Banco de Dados**, qual será o **SQL Server 2022**. Para evitar configuração no instalador, deixei preparado um arquivo *Compose*, chamado *docker-compose-db.yml* da ferramenta **Docker** pré-configurado no projeto. O conteúdo de configuração do ambiente pré-configurado.

```
version: '3.4'

services:
    db:
        image: mcr.microsoft.com/mssql/server:2022-latest
        environment:
          - ACCEPT_EULA=Y
          - MSSQL_SA_PASSWORD=MyPass@word
        ports:
            - "1433:1433"
```

Este arquivo Docker Compose configurará um contêiner **SQL Server 2022** com uma senha "SA" definida como "MyPass@word" e exporá a porta 1433 para acesso ao banco de dados.

No terminal, navegue até o diretório do arquivo *docker-compose-db.yml* e execute o seguinte comando para iniciar o contêiner do ***SQL Server 2022***:

```
$ docker-compose -f docker-compose-db.yml up -d
```

O banco de dados ***SQL Server 2022*** agora está configurado e em execução em um contêiner.

Para restaurar o esquema de tabelas do **Banco de Dados**, no diretório do projeto há um arquivo nomeado '**scheme.sql**' contendo os comandos de esquema das tabelas e criação do banco. Para visualizar o arquivo clique ***aqui***.

5.2 Aplicação

Antes de executar a aplicação .NET, você precisa configurar o ambiente. Siga estas etapas:

Agora, é hora de configurar o ambiente .NET e executar a aplicação Sons Mágicos. Certifique-se de ter o ambiente instalado no seu sistema, para saber se você tem instalado, no seu terminal execute o comando:

```
$ dotnet --version
7.0.400
```

Navegue até o diretório raiz da solução **SonsMagicos**, onde você clonou o repositório e execute os seguintes passos para configurar e executar a aplicação:

Para editar as configurações de conexão do **Banco de Dados** acesse o projeto **SonsMagicos.Api** e edite o arquivo ***appsettings.json*** no atributo "DefaultConnection".

```
{
    "ConnectionStrings": {
        "DefaultConnection":
        "Data Source=localhost;Initial Catalog=DevDB;
            Persist Security Info=True;
            TrustServerCertificate=True;User ID=SA;
```

```
            Password=MyPass@word"
    },
    "Logging": {
        "LogLevel": {
        "Default": "Information",
        "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*"
}
```

Para executar a solução, você deve fazer os seguintes passos

1. Restaurar as dependências do projeto:

   ```
   $ dotnet restore .\SonsMagicos.sln
   ```

2. Compilar o projeto:

   ```
   $ dotnet build .\SonsMagicos.sln
   ```

3. Opcional* - Executar as migrações do banco de dados para criar as tabelas necessárias:

   ```
   $ dotnet ef database update
   ```

4. Antes de executar o comando de iniciar a aplicação certifique-se que o **Banco de Dados** esteja executando. Para executar a aplicação:

   ```
   $ dotnet run --launch-profile 'http' --project
       .\SonsMagicos.Api\SonsMagicos.Api.csproj
   ```

Quando a aplicação é iniciada, um processo de geração automática cria dois usuários padrões no sistema. Os usuários criados são:

```
Nome de usuário: usuario@localhost
Senha: Senha#dific1l

Nome de usuário: admin@localhost
Senha: Senha#dific1l
```

Esses usuários são especialmente designados para o propósito de fornecer acesso a áreas restritas da aplicação que exigem ***autenticação de acesso básico***. Caso você não saiba o que é autenticação de acesso básico, **clique aqui**.

Exemplo de consumo usando autenticação básica no **curl**:

```
    $ curl --location 'http://localhost:5266/api/Instrumentos' \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header 'Authorization: Basic dXN1YXJpb0Bsb2NhbGhvc3Q6U2VuaGEjZGlmaWMxbA==' \
    --data '{
      "nome": "<string>",
      "preco": 12312,
      "propriedade": "<string>",
      "tipo": 3
}'
```

Com os passos feitos, sua aplicação estará executando. Para consumir, basta acessar os ***endpoints*** no URL base: `http://<hostname>:5266`.

6 Consumo da API

6.1 Listagem de itens

GET **/api/Instrumentos/**
Obtém todos os Instrumentos

Parameter

propriedade	Filtra por propriedade os Instrumentos. Por exemplo: dormir

Response application/json

200 OK: Listagem de Instrumentos

```
[
  {
    "id": 0,
    "nome": "string",
    "tipo": 1,
    "preco": 0,
    "propriedade": "string"
  },
  ...
]
```

500 INTERNAL_SERVER_ERROR: Erro interno do servidor

6.2 Criação de item

POST **/api/Instrumentos/**

Cria novo Instrumento

Parameter

Sem parâmetros

Body application/json

```
{
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```

Response application/json

201 CREATED: Instrumento criado

```
{
  "id": 0,
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```

400 BAD_REQUEST: Requisição feita incorretamente

```
{
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "errors": "object"
}
```

401 UNAUTHORIZED: Credenciais inválidas

500 INTERNAL_SERVER_ERROR: Erro interno do servidor

6.3 Atualização de item

PUT **/api/Instrumentos/**

Atualiza um Instrumento

Parameter

id ID do Instrumento

Body application/json

```
{
  "id": 0,
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```

Response application/json

202 ACCEPTED: Instrumento atualizado

```
{
  "id": 0,
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```

400 BAD_REQUEST: Requisição feita incorretamente

```
{
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "errors": "object"
}
```

401 UNAUTHORIZED: Credenciais inválidas

404 NOT_FOUND: Instrumento não encontrado

500 INTERNAL_SERVER_ERROR: Erro interno do servidor

6.4 Obter um item

GET **/api/Instrumentos/{id}**

Obtém o Instrumento do ID especificado

Parameter

id ID do Instrumento

Response application/json

200 OK: Instrumento do ID


```
{
  "id": 0,
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```


400 BAD_REQUEST: Requisição feita incorretamente


```
{
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "errors": "object"
}
```


404 NOT_FOUND: Instrumento não encontrado

500 INTERNAL_SERVER_ERROR: Erro interno do servidor

6.5 Exclusão de item

DELETE **/api/Instrumentos/{id}**

Exclui um Instrumento

Parameter

id ID do Instrumento

Response application/json

202 ACCEPTED: Instrumento excluído


```
{
  "id": 0,
  "nome": "string",
  "tipo": 1,
  "preco": 0,
  "propriedade": "string"
}
```


400 BAD_REQUEST: Requisição feita incorretamente


```
{
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "errors": "object"
}
```


401 UNAUTHORIZED: Credenciais inválidas

404 NOT_FOUND: Instrumento não encontrado

500 INTERNAL_SERVER_ERROR: Erro interno do servidor

6.6 Valor total

GET **/api/Instrumentos/ValorPorTipo**

Obtém a soma dos preços de um tipo de Instrumento

Parameter

tipo Tipo de Instrumento

1 - CORDA

2 - SOPRO

3 - PERCUSSAO

Response application/json

200 OK: Total retornado

```
0
```

400 BAD_REQUEST: Requisição feita incorretamente

```
{
  "type": "string",
  "title": "string",
  "status": 0,
  "detail": "string",
  "instance": "string",
  "errors": "object"
}
```

500 INTERNAL_SERVER_ERROR: Erro interno do servidor