# Book API

## Requisitos do WebService

O sistema de back-end desenvolvido deve ser capaz de oferecer acesso à um catálogo de livros, buscados de um arquivo JSON, disponibilizado no repositório do GitHub disponibilizado por e-mail.

Tendo isso em vista, toma-se o seguinte como requisitos:

- O sistema deve ser capaz de permitir acesso a informações de um livro específico, através da informação de seu id
- O sistema deve ser capaz de listar informações de todos os livros existentes ao usuário
- O sistema deve ser capaz de listar informações de livros mediante apresentação de queries de pesquisa. As queries devem permitir filtragens que utilizem como parâmetro  qualquer atributo de um livro, de forma singular ou em conjunto com outros atributos.

Não funcionais:

- O sistema deve possuir testes unitários, que sejam capazes de validar as funções utilizadas no web-service.
- O sistema deve buscar ser RESTfull

Deve-se, então, buscar organizar o código do software em um estilo arquitetural baseado em camadas (sem camada de apresentação), ou clean architecture. Tendo esse último uma preferência maior.

## Preparação

- [x] ~~Realizar um esboço de modelo conceitual (útil para definição correta das entities)~~
- [x] ~~Pesquisar sobre APIs REST em C# (.NET 3.1 >)~~
- [x] ~~Pesquisar sobre leitura de arquivos JSON em C# (.NET 3.1 >)~~
- [x] ~~Pesquisar sobre testes unitários em C# (.NET 3.1 >)~~
- [x] ~~Realizar um esboço de diagrama de pacotes, ou Components(C4) (se necessário)~~
- [x] ~~Planejar pacotes, classes e funcionalidades~~

## Codificação

- [x] ~~Compor classes de camada de API~~
- [x] ~~Compor testes de funcionalidades da API~~
- [x] ~~Compor classes de leitura e carregamento (pode ser in-memory mesmo) de arquivo JSON~~
    - [x] ~~Resolver problema de alternância entre List<string> e string~~
    - [x] ~~Pesquisar melhor personalização de Parser~~
    - [x] ~~Ver se há uma maneira fácil de carregar arquivos na inicialização do websersice~~
- [x] ~~Compor testes de funcionalidades de leitura e carregamento~~
- [x] ~~Compor classes e funcionalidades de pesquisa (query)~~
- [x] ~~Compor testes de funcionalidade de query~~
- [x] ~~Testar tudo e, se possível, teste de integração (se sobrar tempo)~~
- [x] ~~Pensar sobre utilizar Docker para facilitar deploy (se sobrar tempo)~~
    - [x] ~~.NET já tem Dockerfile fácil~~
- [x] ~~Entregar tudo~~

## Util para consulta:

**Doc C#** https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/

**Intro to Web API** https://www.youtube.com/watch?v=87oOF9Ve-KA

**API mais avançadinha (mas usando padrão minimal)** https://www.youtube.com/watch?v=b7OoeiG_BzU&t=2708s

**Parse de JSON (com Framework)** https://www.youtube.com/watch?v=Y14gG9IJ230

**.NET já tem JSON Deserializer** https://learn.microsoft.com/pt-br/dotnet/standard/serialization/system-text-json/deserialization

**Dá pra peresonalizar nomes, ou desativar case sensetive** https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/customize-properties?pivots=dotnet-8-0

**Dá pra personalizar o converter para resolver problema de alternância entre Array e String** https://learn.microsoft.com/pt-br/dotnet/standard/serialization/system-text-json/converters-how-to?pivots=dotnet-8-0#registration-sample---jsonconverter-on-a-type

**Velho Singleton** https://stackoverflow.com/questions/12316406/thread-safe-c-sharp-singleton-pattern

**Testes unitários com C#** https://learn.microsoft.com/pt-br/previous-versions/visualstudio/visual-studio-2017/test/unit-test-basics?view=vs-2017

**Método de pegar informações de arquivo de configuração de projeto** https://learn.microsoft.com/pt-br/dotnet/api/microsoft.extensions.configuration.configurationbuilder?view=dotnet-plat-ext-8.0