

## Curso .NET

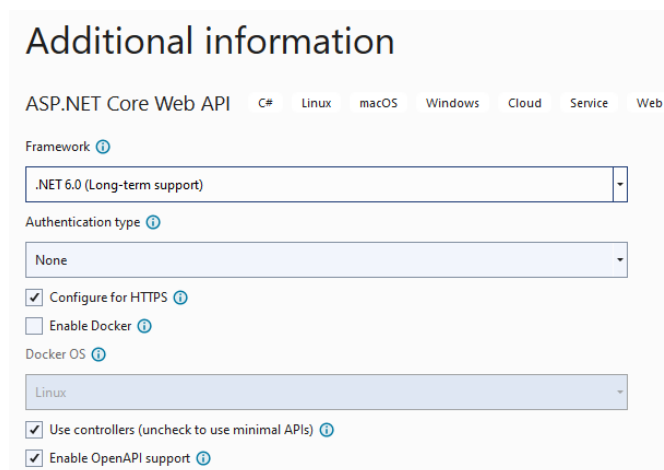
Unidade Curricular: Laboratório Web

Docente: David Jardim

### FICHA DE TRABALHO 12

## Exercícios:

1. Projeto (*API with controllers*)
  - a. Crie um projeto do tipo *Web API* denominado por Ficha12 com as seguintes definições:



Additional information

ASP.NET Core Web API C# Linux macOS Windows Cloud Service Web

Framework ⓘ

.NET 6.0 (Long-term support)

Authentication type ⓘ

None

☒ Configure for HTTPS ⓘ

☐ Enable Docker ⓘ

Docker OS ⓘ

Linux

☒ Use controllers (unchecked to use minimal APIs) ⓘ

☒ Enable OpenAPI support ⓘ

2. Modelos
  - a. Crie uma pasta denominada por Models
  - b. Crie uma classe *Book* para representar um livro e adicione as propriedades referidas no diagrama de classes (tenha em atenção os tipos) respeitando os nomes
  - c. Crie uma classe *Publisher* para representar uma editora e adicione as propriedades referidas no diagrama de classes (tenha em atenção os tipos) respeitando os nomes
  - d. Crie uma classe *LibraryContext* para criar a sessão com a base de dados e que será usada para realizar as operações necessárias à BD
    - i. Adicione uma propriedade do tipo DbSet para realizar queries à base de dados utilizando instâncias do tipo Book
    - ii. Adicione uma propriedade do tipo DbSet para realizar queries à base de dados utilizando instâncias do tipo Publisher
    - iii. Implemente o construtor por parâmetros (consulte o material disponibilizado)
    - iv. Sobreponha o método *OnConfiguring* tendo em conta a *connection string* do seu servidor de MySQL (consulte o material disponibilizado)
    - v. Sobreponha o método *OnModelCreating* tendo em conta as propriedades das colunas das tabelas na DB(consulte o material disponibilizado)

### 3. Serviços

- a. Crie uma pasta denominada por *Services*
- b. Crie uma interface denominada por *IBookService* e defina os seguintes métodos abstratos tendo em conta os seus parâmetros e tipo a devolver
  - i. *GetAll*
  - ii. *Create*
  - iii. *DeleteByISBN*
  - iv. *GetByISBN*
  - v. *Update*
  - vi. *GetByAuthor*
  - vii. *Download*
  - viii. *UpdatePublisher*
- c. Crie uma classe denominada por *BookService* que implementa a interface anterior
  - i. Implemente o construtor por parâmetros que receberá como parâmetro uma referência para a classe *LibraryContext*
  - ii. Implemente os métodos abstratos que utilizarão a referência para a classe *LibraryContext* de forma a realizar as operações de seleção e modificação na base de dados (consulte o material disponibilizado)

### 4. Controllers

- a. Tendo em conta a Tabela 1, na pasta *Controllers* crie um controller *BooksController*
- b. Adicione um atributo privado do tipo *IBookService*
- c. Implemente o construtor por parâmetros, onde o parâmetro é do tipo *IBookService*
- d. Implemente os seguintes endpoints tendo em conta as validações necessárias e respostas adequadas e utilize o serviço para :
  - i. Listar todos os funcionários existentes no ficheiro e devolver os mesmos na resposta
  - ii. Adicionar um novo livro, o ISBN do novo livro deve ser devolvido na resposta
  - iii. Apagar um livro pelo seu ISBN. O ISBN do livro removido deve ser devolvido na resposta
  - iv. Selecionar apenas um livro pelo seu ISBN e devolver esse mesmo livro na resposta
  - v. Alterar os detalhes de um determinado livro pelo seu ISBN e devolver esse mesmo livro atualizado na resposta
  - vi. Listar todos os livros por autor
  - vii. Efetuar download da lista atual de livros como um ficheiro .json
  - viii. Alterar a editora de um livro utilizando o ISBN como parâmetro da rota e o ID da editora como parâmetro da *query*

### 5. Dados

- a. Crie uma pasta denominada por *Data*
- b. Crie uma classe estática denominada por *LibraryDBInitializer*
- c. Implemente um método estático denominado por *InsertData*, este método deverá receber uma referência para a classe *LibraryContext*, e essa referência deverá ser utilizada para adicionar algum conteúdo inicial à base de dados, por exemplo, adicionar uma editora e alguns livros (consulte o material disponibilizado)
- d. Crie uma estática classe denominada por *LibraryExtension* que será usada como um middleware para invocar o método para inserir os dados na base de dados
- e. Na classe *LibraryExtension* implemente um método estático chamado *CreateDbIfNotExists* para criar a base de dados se não existir e inserir os dados (consulte o material disponibilizado)

## 6. Program

- Aplicando *dependency injection* adicione à lista de serviços da aplicação um contexto para a base de dados do tipo *LibraryContext* (consulte o material disponibilizado)
- Aplicando *dependency injection* adicione à lista de serviços da aplicação um serviço com tempo de vida scoped para a base de dados do tipo *IBookService* com implementação *BookService* (consulte o material disponibilizado)
- Invoque a extensão *CreateDbIfNotExists* através da instância para a sua aplicação

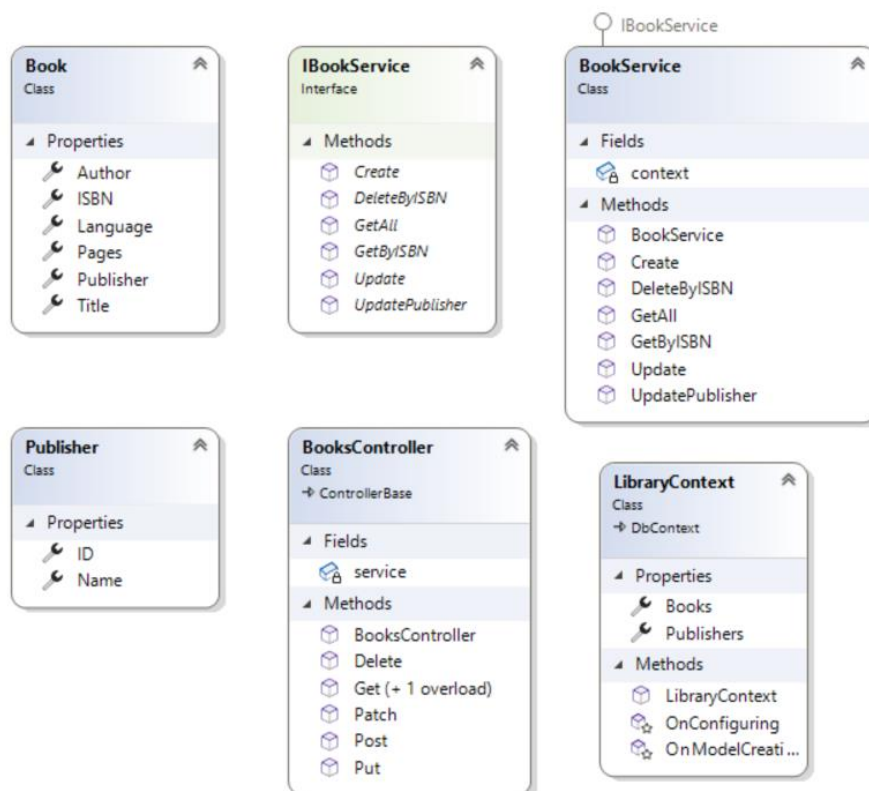


Figura 1 - Diagrama de Classes

URI	Método HTTP	Body do POST	Resultado
/books	GET	empty	List all books
/books	POST	JSON String	Add details of a new book
/books/{isbn}	DELETE	empty	Delete an existing book
/books/{isbn}	GET	empty	Show details of a book
/books/{isbn}	PUT	JSON String	Update details of a book
/books/{author}	GET	empty	Show books from an author
/books/download	GET	empty	Download list of books
/books/{isbn}/publisherId	PATCH	empty	Update the publisher of a book

Tabela 1 - Métodos a implementar