**Objetivo:**

A partir dos conceitos e práticas sobre API apresentados em aula, usando a IDE VisualStudio 2022, crie um projeto ASP.NET Core API (9) no estilo NETFLIX.

Durante a prova, todo tipo de recurso para consulta estará disponível.

**Requisitos técnicos:**

1 –Autenticação por JWT – JSON WEB TOKEN:

* Proteger a API usando JWT.
* Prepare a API para apenas suportar tokens JWT gerados com a chave secreta da aplicação.
* A chave deverá estar armazenada no arquivo de configuração *appsettings.json* e ser recuperada quando necessário.

2 - Utilização de Classes de Serviço, DTOs, Repositórios e Entidades:

* A aplicação deve ser organizada com classes de serviço, DTOs e entidades para uma melhor estruturação do código. Analise com critério, quais recursos são necessários.

3 - Injeção de Dependência

* Analise sua implementação do projeto e utilize injeção de dependência sempre que possível para evitar o alto nível de acoplamento de código.

4 - Documente a API com o Swagger (TODOS OS ENDPOINTS).

**Requisitos da API:**

Além dos requisitos técnicos acima, a API deve implementar os seguintes endpoints:

**POST /usuarios/validar (requisição anônima)**

* Este endpoint tem como objetivo validar um usuário com base no e-mail e senha fornecidos.
* O corpo da requisição, é necessário fornecer conter o e-mail e a senha do usuário (em formato JSON).
* Se o corpo da requisição estiver vazio, a requisição deve ser rejeitada com uma mensagem apropriada.
* Caso as credenciais (e-mail + senha) não correspondam a um usuário existente, rejeite a requisição e retorne uma mensagem explicando o erro.
* Se o usuário for validado corretamente, **gere um token JWT** que deverá ser utilizado para autenticação em requisições autenticadas.
* O E-mail e Id do usuário deverão estar dentro do token.
* Este EP deve retornar o código HTTP 200 para indicar um usuário válido, caso contrário, o HTTP código para NotFound (404).
* Já existe um usuário cadastrado. Usuário: *fipp* e Senha: *123456*. Não há necessidade de cadastrar outros.

**GET /perfis (requisição autenticada)**

* O objetivo deste endpoint é retornar todos os perfis associados ao usuário.
* Cada perfil possui um **nome** e um **tipo** (normal ou infantil) que na implementação deverá ser um ENUM.
* O Id do usuário deverá ser extraído do token de autenticação.
* A resposta deve incluir todos os perfis vinculados ao usuário.

**POST /perfis (requisição autenticada)**

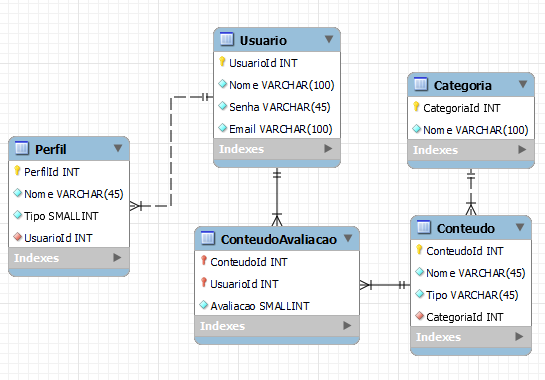
* O objetivo deste endpoint é salvar um perfil do usuário.
* No corpo da requisição, devem ser enviados o **nome do perfil** e seu **tipo** (normal ou infantil) (em formato JSON).
* Caso já exista um perfil com o mesmo nome, a requisição deverá ser rejeitada com uma mensagem informando o motivo da recusa.
* Outra regra é o limite de 4 perfis por usuário. Caso o perfil ultrapassar o limite a requisição deve ser rejeitada com uma mensagem informando o motivo da recusa.
* O Id do usuário deverá ser extraído do token de autenticação.
* O EP deve retornar o código HTTP 200 para indicar o sucesso, caso contrário, o HTTP código para BadRequest.

**POST /conteudo/{conteudoId}/avaliacao (requisição autenticada)**

* Este endpoint tem como objetivo registrar uma avaliação do conteúdo pelo o usuário.
* O corpo da requisição (em formato JSON) deve incluir o id da avaliação (1 - NÃO É PARA MIM, 2 - GOSTEI OU 3 – ADOREI). Na implementação, use um ENUM para representar os tipos de avaliação.
* Valide que a avaliação apenas seja um dos tipos.
* O Id do usuário deverá ser extraído do token de autenticação.
* O EP deve retornar o código HTTP 200 para indicar o sucesso, caso contrário, o HTTP código para BadRequest.

**GET /conteudo/recomendacao (requisição autenticada)**

* Este endpoint tem como objetivo obter conteúdos recomendados para um usuário.
* O Id do usuário deverá ser extraído do token de autenticação.
* O algoritmo de recomendação deve encontrar conteúdo da mesma categoria dos conteúdos classificados pelo próprio usuário como “ADOREI”. Caso nenhum item, então aplicar a mesma abordagem para conteúdos classificados como “GOSTEI”.
* O EP deve retornar o código HTTP 200 com a lista de conteúdo recomendados.

****

**Script para gerar o Banco de Dados**

CREATE TABLE `Usuario` (

`UsuarioId` int NOT NULL AUTO\_INCREMENT,

`Nome` varchar(100) NOT NULL,

`Senha` varchar(45) NOT NULL,

`Email` varchar(100) NOT NULL,

PRIMARY KEY (`UsuarioId`)

);

INSERT INTO `Usuario` VALUES (1,'fipp','123456', 'fipp@fipp.com.br');

CREATE TABLE `Categoria` (

`CategoriaId` int NOT NULL AUTO\_INCREMENT,

`Nome` varchar(100) NOT NULL,

PRIMARY KEY (`CategoriaId`)

);

INSERT INTO `Categoria` VALUES (1,'Terror'),(2,'Romance'),(3,'Comedia');

CREATE TABLE `Conteudo` (

`ConteudoId` int NOT NULL AUTO\_INCREMENT,

`Nome` varchar(45) NOT NULL,

`Tipo` varchar(45) NOT NULL,

`CategoriaId` int NOT NULL,

PRIMARY KEY (`ConteudoId`),

KEY `FK\_Conteudo\_Categoria\_idx` (`CategoriaId`),

CONSTRAINT `FK\_Conteudo\_Categoria` FOREIGN KEY (`CategoriaId`) REFERENCES `Categoria` (`CategoriaId`) ON DELETE CASCADE

);

INSERT INTO `Conteudo` VALUES (1,'Conteúdo 1','Filme',1),(2,'Conteúdo 2','Série',2),(3,'Conteúdo 3','Filme',3),(4,'Conteúdo 4','Filme',1);

CREATE TABLE `ConteudoAvaliacao` (

`ConteudoId` int NOT NULL,

`UsuarioId` int NOT NULL,

`Avaliacao` smallint NOT NULL,

PRIMARY KEY (`ConteudoId`,`UsuarioId`),

KEY `FK\_ConteudoAvaliacao\_Usuario\_idx` (`UsuarioId`),

CONSTRAINT `FK\_ConteudoAvaliacao\_Conteudo` FOREIGN KEY (`ConteudoId`) REFERENCES `Conteudo` (`ConteudoId`) ON DELETE CASCADE,

CONSTRAINT `FK\_ConteudoAvaliacao\_Usuario` FOREIGN KEY (`UsuarioId`) REFERENCES `Usuario` (`UsuarioId`) ON DELETE CASCADE

);

CREATE TABLE `Perfil` (

`PerfilId` int NOT NULL AUTO\_INCREMENT,

`Nome` varchar(45) NOT NULL,

`Tipo` smallint NOT NULL,

`UsuarioId` int NOT NULL,

PRIMARY KEY (`PerfilId`),

KEY `FK\_Perfil\_Usuario\_idx` (`UsuarioId`),

CONSTRAINT `FK\_Perfil\_Usuario` FOREIGN KEY (`UsuarioId`) REFERENCES `Usuario` (`UsuarioId`) ON DELETE CASCADE

);