Task 6 – Proteger API

Antes de inicializar as atividades, definir quem irá desenvolver cada trecho de código de maneira que todos trabalhem.

Atividades:

1. Definir **enum** tipo de usuário (Mínimo, ADMIN e USER);

2. Modificar modelo tabela de usuário:
    a. Incluir **enum** tipo de usuário como atributo;
    b. Deletar banco no SQL Server, para poder pegar alterações ao recriar.

3. Criar serviço para gerar token JWT;

4. Criar **DTOs** do contexto de autenticação e autorização:
    a. Criar dto para efetuar Login no sistema;
    b. Criar dto para devolver autorização;

5. Criar **interface** para serviços de autenticação e autorização com as assinaturas:
    a. UserModel CreateUserNotDuplicated(UserRegisterDTO user);
        Esta assinatura sera utilizada para criar um usuario validando duplicado, para que o sistema não duplique usuario.

    b. AuthorizationDTO GetAuthorization(UserLoginDTO user);
        Esta assinatura será utilizada para pegar autorização do usuario para poder manipular a API de qualquer parte.

6. Implementar interface de serviços de autenticação e autorização:
    a. Criar um método para criptografar senha;
    b. Implementar método CreateUserNotDuplicated;
    c. Implementar método GetAuthorization.

7. Incluir serviço de autenticação no arquivo ***Startup.cs***, no método ***ConfigureServices***;

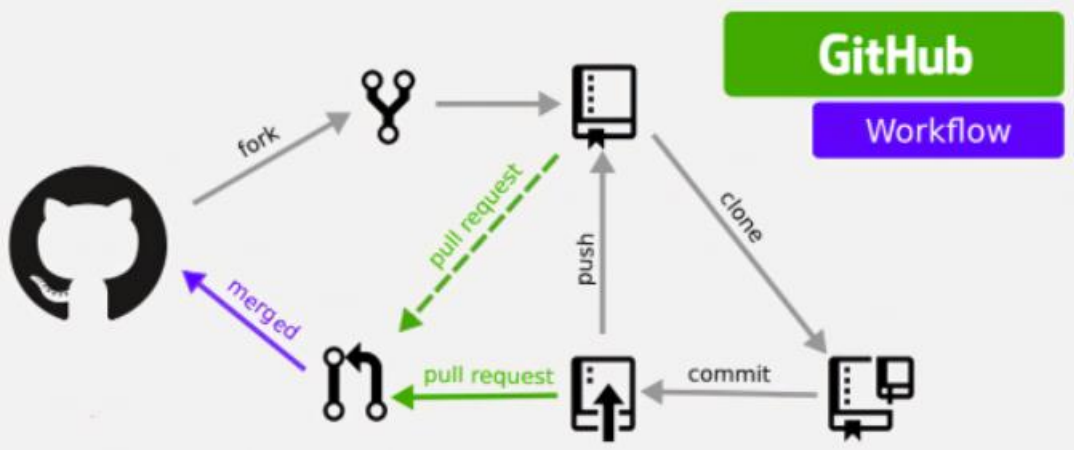
8. Incluir no documento ***Startup.cs***, no método ***Configure***, os parâmetros para usar autenticação e autorização;

9. Definir notações de autenticação para cada end-point nos controladores do sistema;

10. Testar controladores com **Portman** validando o acesso das rotas com credenciais validas e invalidas. Testar todos os end-points, não deixar retornar status 500.

Entrega

Para esta entrega é necessário que apenas uma pessoa esteja com o projeto criado no repositório, considerar este repositório como repositório principal do projeto. Os demais companheiros faram ***fork*** do projeto. O ***fork*** é uma forma de trazer o repositório de um destino para dentro de seu GitHub. Desta maneira cada um terá uma versão do projeto dentro de seu repositório. É importante que o repositório principal seja sempre atualizado com as informações do projeto. O papel de cada um dos participantes desenvolvedores ao terminar uma tarefa atualizar seus repositórios e o repositório principal. A seguir existe um fluxo que explica este processo:



Na figura acima o ícone do gato (GitHub) representa o repositório principal. Ao efetuar ***fork***, em seu repositório é possível visualizar o projeto. Tenha o projeto clonado em uma pasta bem definida em sua máquina. Sempre que for passado uma tarefa se preocupe de realizar um ***pull*** do repositório principal, para se manter atualizado. Ao realizar sua tarefa caso não tenha permissão para editar o repositório principal faça um ***push*** para seu repositório, e a partir dali criar um ***pull*** request para o repositório principal. Caso tenha permissão para editar o repositório principal faça um ***push*** diretamente de sua máquina para o repositório principal (Para isso é necessário ter o ***remote*** do repositório principal em sua máquina) e depois para seu repositório.

Dicas:

- Sempre deve manter atenção ao repositório principal, ele pode estar atualizado e o seu não;
- Assim que terminar sua tarefa, procure atualizar seu repositório e depois o repositório principal;
- Antes de começar a escrever uma tarefa, procure saber se seu projeto, em sua máquina e repositório, estão atualizados com relação ao repositório principal;
- Conflito não é erro, e sim um sinal que o GitHub não consegue decidir por você qual código está correto, então o mesmo encaminha isso como forma de conflito;

Cada um deve postar o link de seu repositório atualizado na plataforma!