[Introdução ao Desenvolvimento ASP.NET Core]

Proposta de Projeto

1. Título do Projeto:

- Aplicação de Blog Simples com MVC e API RESTful

2. Objetivo:

- Desenvolver uma aplicação de blog que permita aos usuários criar, editar, visualizar e excluir posts e comentários, tanto através de uma interface web utilizando o padrão MVC, quanto através de uma API RESTful que expõe os mesmos recursos do blog.

3. Descrição Geral e Requisitos Funcionais:

- Este projeto consiste em criar uma aplicação web utilizando ASP.NET Core, que oferece funcionalidades de blog, incluindo criação, edição, visualização e exclusão de posts e comentários. A aplicação será dividida em duas partes:
    1. **Aplicação MVC:** Interface web onde os usuários podem interagir com o blog.
    2. **API RESTful:** Uma API que expõe os mesmos recursos do blog, permitindo a integração com outras aplicações ou o desenvolvimento de um front-end alternativo.
- A aplicação deve suportar autenticação e autorização, permitindo diferentes níveis de acesso (administradores e usuários comuns). Administradores podem gerenciar todos os posts e comentários, enquanto usuários comuns podem criar e gerenciar seus próprios posts e comentários.

- **Separação de Responsabilidades entre a Aplicação MVC e a API:**
    - A aplicação MVC não deve consumir a API, ambas devem oferecer o mesmo mecanismo de funcionalidades (busque uma maneira inteligente de não duplicar código).
- **Definição Estrutural do Blog:**
    - O blog deve ser estruturado em três entidades principais: Autor, Posts e Comentários.
- **Mecanismo de CRUD para Posts e Comentários:**
    - A aplicação deve permitir a criação, leitura, atualização e exclusão de Posts e Comentários. Este mecanismo deve ser robusto, com tratamento de erros adequado, como por ex: verificação de campos obrigatórios.
- **Exibição de Lista de Posts na Página Inicial:**
    - A home page deve exibir uma lista dos Posts com informações como título, descrição, data de publicação e Autor.
- **Acesso Anônimo aos Posts:**
    - Posts devem ser publicamente acessíveis a todos os usuários, independentemente de estarem autenticados. Isso inclui a exibição completa do conteúdo do Post e a visualização de Comentários associados.
- **Comentários Restritos a Usuários Logados:**
    - Somente usuários autenticados podem adicionar Comentários a um Post.
- **Associação de Posts a Autores:**
    - Cada Post deve estar associado a um único Autor, e esta associação deve ser persistida no banco de dados. O sistema deve garantir que apenas o Autor original ou o Admin pode modificar ou deletar seus os posts e comentários.

- **Identidade do Autor como Usuário Registrado:**
    - O sistema deve garantir que todos os Autores sejam usuários registrados, utilizando o ASP.NET Core Identity. Deve haver uma validação para garantir que apenas usuários autenticados possam criar conteúdo.
- **Privilégios de Administrador:**
    - O administrador deve ter total controle sobre todos os Posts e Comentários, incluindo a capacidade de editar e deletar qualquer conteúdo. Este controle inclui o gerenciamento de Comentários que possam ser inadequados ou spam.
- **Gerenciamento de Usuários:**
    - O sistema deve utilizar as funcionalidades padrão do ASP.NET Core Identity para o registro e autenticação de usuários. A gestão de papéis (como atribuir o papel de administrador) pode ser feita manualmente no banco de dados a criação de um usuário pode ser feita através da tela “Register” fornecida pelo Identity.
- **Validação e Tratamento de Erros:**
    - Deve haver uma validação robusta em todas as operações de CRUD para garantir que os dados inseridos sejam válidos e seguros. O sistema deve fornecer feedback de erro claro e informativo, tanto na aplicação MVC quanto na API.
- **Segurança da API:**
    - A API deve ser protegida com autenticação JWT, garantindo que apenas usuários autenticados possam acessar endpoints que modificam dados (criação, atualização, exclusão).
- **Recursos Extras**
    - Recursos adicionais podem render pontos extras, mas não são obrigatórios. Recomendo que você concentre seus esforços em entregar com excelência os requisitos obrigatórios já definidos. Adicionar complexidade desnecessária não é preciso neste momento (guarde suas energias para projetos mais avançados). Se desejar acrescentar algo a mais, mantenha o projeto simples e funcional. Um bom investimento de tempo seria melhorar a experiência do usuário (UX).

4. Requisitos Técnicos:

- **Linguagem de Programação:** C#
- **Frameworks:**
    - ASP.NET Core MVC para a interface web.
    - ASP.NET Core Web API para expor os recursos do blog.
- **Acesso a Dados:**
    - SQL Server para armazenar posts, comentários e informações dos usuários.
    - EF Core para mapear o BD e realizar operações de CRUD
- **Autenticação e Autorização:**
    - Implementar autenticação usando ASP.NET Core Identity.
    - Usar roles para diferenciar os privilégios entre administradores e usuários comuns.
- **Front-end:**
    - Views dentro do ASP.NET Core MVC.
    - Uso básico de HTML e CSS para estilização.
    - Use a criatividade ou templates prontos para desenvolver a interface
- **API:**
    - Implementar endpoints RESTful para CRUD (Create, Read, Update, Delete) de posts e comentários.
    - A API deve suportar autenticação/autorização via JWT.
- **Versionamento:**
    - Github para controle de versão, com o código sendo hospedado em um repositório publico e dentro dos padrões especificados em: https://github.com/desenvolvedor-io/template-repositorio-mba

5. Critérios de Sucesso:

- **Funcionalidade Completa:**
    - Implementação de CRUD para posts e comentários tanto na aplicação MVC quanto na API.
- **Qualidade do Código:**
    - Código claro e bem documentado, aderindo às práticas ensinadas no curso.
- **Segurança:**
    - Implementação correta de autenticação e autorização.
    - Proteção da API com autenticação JWT.
- **Documentação:**
    - Incluir um arquivo README.md detalhado no repositório GitHub, dentro dos padrões indicados e com instruções de instalação, configuração e uso se necessário.
    - Documentar a API usando Swagger ou outra ferramenta semelhante.
- **Configuração:**
    - O projeto deve rodar com a menor configuração de infra possível, para isso utilize a prática ensinada no vídeo a seguir: https://desenvolvedor.io/plus/criando-e-populando-automaticamente-qualquer-banco-de-dados

6. Prazos:

- **Início do Desenvolvimento:** 02/09/2024
- **Primeira entrega (avaliação):** 14/10/2024
- **Segunda entrega (final):** 04/11/2024
- **Apresentação:** 06/11/2024 à 12/11/2024

7. Entrega:

- **Repositório no GitHub:**
    - O código deve ser versionado e entregue através de um repositório público no Github.
- **Documentação:**
    - O README.md deve seguir as diretivas e padrões informados na documentação do projeto referência.
    - Incluir um arquivo FEEDBACK.md no repositório onde os feedbacks serão consolidados, o instrutor fará um PR no repositório atualizando este arquivo.

8. Matriz de avaliação:

- Os projetos serão avaliados e receberão uma nota de 0 até 10 considerando os critérios a seguir:

Critério	**Peso**	**Comentários**
Funcionalidade	30%	Avalie se o projeto atende a todos os requisitos funcionais definidos.
Qualidade do Código	20%	Considere clareza, organização, uso de padrões de codificação.
Eficiência	10%	Avalie o desempenho e a eficiência das soluções implementadas.
Inovação	10%	Considere a criatividade e inovação na solução proposta.
Documentação	10%	Verifique a qualidade e completude da documentação, incluindo README.md.
Apresentação	10%	Avalie a clareza, organização e impacto da apresentação ao vivo.
Resolução de Feedbacks	10%	Considere como o aluno ou grupo abordou os feedbacks da revisão de código.