MBA DevXpert Full Stack .NET

[Introdução ao Desenvolvimento ASP.NET Core]

Requisitos do Projeto Módulo 01

1. Título do Projeto

Gestão de Mini Loja Virtual com Cadastro de Produtos e Categorias

2. Objetivo

Desenvolver uma aplicação web básica usando conceitos do Módulo 1 (C#, ASP.NET Core MVC, SQL, EF Core, APIs REST) para gestão simplificada de produtos e categorias em um formato tipo e-commerce marketplace.

3. Aspectos Gerais do Projeto

A aplicação consistirá em uma plataforma web básica com uma interface intuitiva, que possibilite ao usuário realizar registro/login de usuário, cadastrar, visualizar, atualizar e remover categorias e produtos, além de consultar esses dados através de uma API REST básica.

A solução será estruturada nos apectos principais:

- **Frontend (Interface de Usuário)**: Desenvolvido em ASP.NET Core MVC, permitindo interações claras e simples com o usuário.
  Este projeto tem como objetivo ser a área administrativa de um e-commerce.
- **Backend (API)**: Uma API REST simples, também em ASP.NET Core, responsável por expor funcionalidades básicas de autenticação, consultas e modificações.
- **Banco de Dados:** SQL Server/SQLite com EF Core para persistência e gerenciamento de dados.

A aplicação deve suportar autenticação e autorização, por se tratar de uma aplicação e-commerce tipo marketplace, cada usuário é um “vendedor”, portanto deve ter acesso apenas aos seus próprios produtos cadastrados.

- **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 Modelo:**
    - A aplicação deve possuir ao menos três entidades principais: Vendedor, Categoria e Produto.
- **Mecanismo de CRUD para Categorias e Produtos:**
    - A aplicação deve permitir a criação, leitura, atualização e exclusão de Categorias e Produtos. Este mecanismo deve ser robusto, com tratamento de erros adequado, como por ex: verificação de campos obrigatórios.
- **Exibição de Produtos na Página Inicial:**
    - A home page deve exibir uma lista dos Produtos com informações como título, descrição, valor, etc. Deve exbir a imagem do produto.
    - Além disto, acesso as telas de CRUD para Produtos e Categorias devem estar expostas no menu.
- **Acesso Anônimo aos Produtos:**
    - Produtos devem ser publicamente acessíveis a todos os usuários via API, independentemente de estarem autenticados. Isso inclui a exibição completa dos dados dos Produtos e suas Categorias.
- **Ações de Escrita Restrita a Usuários Logados:**
    - Somente usuários autenticados podem adicionar, editar e remover um produto ou categoria.
- **Associação de Produtos a Venderores:**
    - **Cada Produto deve estar associado a um único Vendedor**, e esta associação deve ser persistida no banco de dados. O sistema deve garantir que apenas o Vendedor proprietário pode consultar, visualizar e modificar dados do Produto.
- **Categorias são "Livres"**
    - **As categorias não estão restritas a um único vendedor**, todos podem ver as categorias e editar. Futuramente essa funcionalidade será tranferida a um perfil administrador.

- **Identidade do Vendedor como Usuário Registrado:**
    - O sistema deve garantir que todos os Venderores sejam usuários registrados, utilizando o ASP.NET Core Identity. Deve haver uma validação para garantir que apenas usuários autenticados possam registrar produtos.
- **Gerenciamento de Usuários:**
    - O sistema deve utilizar as funcionalidades padrão do ASP.NET Core Identity para o registro e autenticação de usuários.
    - **O user do Identity deve co-existir com a entidade Vendedor**, ambos compartilham o mesmo ID e devem ser criados no mesmo processo.
- **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 não são necessários.
    - **Recomendo que você concentre seus esforços em entregar com excelência** os requisitos obrigatórios já definidos. Adicionar complexidade desnecessária não é preciso neste momento (**guarde seus skills técnicos 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 Funcionais

1. Gerenciamento de Usuários

- Cadastro e Login de usuários
- Autenticação via ASPNET Identity
- Autenticação via Cookie (MVC) e JWT (API)

2. Gerenciamento de Categorias

- Cadastro de novas categorias, com informações básicas (Nome e descrição).
- Visualização de uma lista de categorias cadastradas.
- Edição e exclusão de categorias existentes.
- Impedir exclusão de categorias com produtos associados.

3. Gerenciamento de Produtos

- Cadastro de produtos contendo nome, descrição, imagem, preço, estoque e categoria associada.
- Edição, visualização e exclusão de produtos cadastrados.
- Validação dos campos obrigatórios com feedback visual claro ao usuário.

4. API REST para Produtos

- Expor endpoints anonimos para:
    - Login
    - Listagem completa de produtos.
    - Listagem de produtos filtrados por categoria.
- Endpoints com autenticação para:
    - CRUD completo das entidades

Regras de Negócio

- Não permitir preços ou estoques negativos.
- Categorias devem ser cadastradas previamente para associar produtos.

5. Requisitos Técnicos

- **Linguagem de Programação: C# .NET 8 ou superior**
- **Frameworks:**
    - ASP.NET Core MVC para a interface web.
    - ASP.NET Core Web API para expor Produtos/Categorias.
- **Acesso a Dados:**
    - SQL Server/SQLite – O projeto deve ser compatível com SQL Server, **porém deve ser entregue configurado para SQLite em modo "Development"** para que seja executado sem dependencias de infra. Use os environments do ASPNET.
    - EF Core para mapear o BD e realizar operações de CRUD
- **Autenticação e Autorização:**
    - Implementar autenticação usando ASP.NET Core Identity.
    - **Associar o ID do AspNetUser com o Vendedor** (não é necessário relacionamento) basta criá-los no mesmo momento e usar o ID do AspNetUser no ID do Vendedor.
- **Front-end:**
    - Views dentro do ASP.NET Core MVC.
    - Navegação e UX com 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

6. Casos de Uso

Caso de Uso 1: Cadastro e Autenticação de Usuários

- **Descrição:** O sistema permite que usuários se cadastrem e realizem login:
- Usuário acessa a página de cadastro/login.
- Preenche nome, e-mail e senha.
- Sistema valida os dados e gera conta ou autentica
- A autenticação por API deve ser feita com JWT

Caso de Uso 2: Cadastro de Categorias

- **Descrição:** Sistema permite ao usuário adicionar, editar, visualizar e excluir categorias.
- **Regras:** Não permitir exclusão se houver produtos associados.

Caso de Uso 3: Cadastro de Produtos

- **Descrição:** Cadastro detalhado de produtos com todas as informações obrigatórias validadas (nome, descrição, imagem, preço, estoque, categoria).
- **Regras:** Não permitir valores negativos ou categorias inexistentes.

Caso de Uso 4: Visualização de Produtos

- **Descrição:** Sistema exibe produtos com imagens e informações principais na home do sistema administrativo.
- Grid de produtos disponível na página inicial, em formato de vitrine de e-commerce.

7. Critérios de Sucesso

Funcionalidades Completas:

- Requisitos funcionando corretamente e sem erros ou falhas

Qualidade do Código:

- Código claro e bem documentado, aderindo às práticas ensinadas nos cursos.

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 com **suporte a autenticação JWT**.

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
- **Caso não seja possível rodar a aplicação por alguma dependência de infra ou setup necessário o projeto não será avaliado**

8. Prazos

- **Início do Desenvolvimento:** 18/03/2025
- **Primeira entrega (avaliação):** 07/04/2025 (20 dias)
- **Segunda entrega (final):** 05/05/2025 (28 dias)
- **Apresentação da avaliação final e notas:** 16/05/2025

9. 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.

10. 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 e Desempenho	20%	Avalie o desempenho e a eficiência das soluções implementadas.
Inovação e Diferenciais	10%	Considere a criatividade e inovação na solução proposta.
Documentação e Organização	10%	Verifique a qualidade e completude da documentação, incluindo README.md.
Resolução de Feedbacks	10%	Avalie a clareza, organização e impacto da apresentação ao vivo.