Arquitetura do Projeto: E-commerce de Venda de Licor

Modelo de Arquitetura: Arquitetura em Camadas

1. Camada de Apresentação (Web)

Tecnologias:

- **ASP.NET Core MVC**: Framework para construção de aplicações web baseadas em padrões Model-View-Controller.
- **Razor Pages**: Para renderização de páginas dinâmicas.
- **HTML/CSS/JavaScript**: Para a construção da interface do usuário.

Classes:

- **Controladores (Controllers):**
    - ProdutoController: Gerencia as requisições relacionadas a produtos (exibição do catálogo, detalhes do produto).
    - CarrinhoController: Gerencia o carrinho de compras e o processo de checkout.
    - UsuarioController: Gerencia o registro, login e visualização da conta do usuário.
    - EnderecoController: Gerencia a busca e validação de endereços.
    - AdminController: Para o painel administrativo, gerencia produtos, pedidos e usuários.
- **Views:**
    - Views/Produto/Index.cshtml: Página inicial com produtos em destaque e categorias.
    - Views/Produto/Detalhes.cshtml: Página com detalhes do produto específico.
    - Views/Carrinho/Index.cshtml: Página do carrinho de compras.
    - Views/Checkout/Index.cshtml: Página para finalização da compra.
    - Views/Conta/Perfil.cshtml: Página para gerenciamento da conta do usuário.
    - Views/Conta/HistoricoPedidos.cshtml: Página para visualização do histórico de pedidos.
    - Views/Admin/Produtos.cshtml: Página para gerenciar produtos.
    - Views/Admin/Pedidos.cshtml: Página para gerenciar pedidos.
    - Views/Admin/Usuarios.cshtml: Página para gerenciar usuários.
- **Modelos de Visualização (ViewModels):**
    - ProdutoViewModel
    - CarrinhoViewModel
    - CheckoutViewModel
    - LoginViewModel
    - RegistroViewModel
    - EnderecoViewModel

Responsabilidades:

- Exibir dados e páginas dinâmicas.
- Gerenciar a interação com o usuário e coletar dados de entrada.
- Validar dados do lado do cliente.

2. Camada de Aplicação (Service)

Tecnologias:

- **C#**: Linguagem de programação principal para lógica de aplicação.
- **ASP.NET Core**: Framework para construção de serviços e APIs.
- **AutoMapper**: Para mapeamento entre DTOs e entidades de domínio.
- **FluentValidation**: Para validação de dados de entrada.

Classes:

- **Serviços de Aplicação:**
    - ProdutoService: Lógica para buscar e filtrar produtos.
    - CarrinhoService: Lógica para gerenciar o carrinho de compras e calcular totais.
    - CheckoutService: Lógica para processar o checkout e validar informações.
    - UsuarioService: Gerencia a lógica de registro, autenticação e recuperação de senhas.
    - EnderecoService: Interage com APIs para validação de CEP e outros dados de endereço.
    - AdminService: Lógica para administração de produtos, pedidos e usuários.
- **DTOs (Data Transfer Objects):**
    - ProdutoDto
    - CarrinhoDto
    - CheckoutDto
    - LoginDto
    - RegistroDto
    - EnderecoDto

Responsabilidades:

- Encapsular a lógica de negócios e regras específicas.
- Facilitar a comunicação entre a camada de apresentação e a camada de domínio.

3. Camada de Domínio (Model)

Tecnologias:

- **C#**: Linguagem de programação para modelagem de dados e regras de negócios.
- **Dapper**: ORM para mapeamento objeto-relacional.

Classes:

- **Entidades de Domínio:**
    - Produto
    - Carrinho
    - Pedido
    - ItemPedido
    - Usuario
    - Endereco
- **Repositórios de Domínio:**
    - IProdutoRepository
    - ICarrinhoRepository
    - IPedidoRepository
    - IUsuarioRepository
    - IEnderecoRepository
- **Serviços de Domínio:**
    - AutenticacaoService: Para gerenciamento de tokens e autenticação.
    - CpfValidationService: Para validação de CPF usando uma API externa.

Responsabilidades:

- Gerenciar a lógica de negócios central e a estrutura de dados.
- Implementar as regras de negócios e operações complexas.

4. Camada de Infraestrutura (Repositório)

Tecnologias:

- **Dapper**: Micro-ORM para acesso rápido e eficiente ao banco de dados.
- **SQL Server**: Sistema de gerenciamento de banco de dados relacional.
- **Serilog**: Biblioteca de logging para registrar eventos e erros.
- **AutoMapper**: Para mapeamento entre entidades e DTOs.
- **HttpClient**: Para comunicação com APIs externas (validação de CPF e CEP).

Classes:

- **Repositórios de Dados:**
    - ProdutoRepository: Implementa IProdutoRepository para operações CRUD em produtos.
    - CarrinhoRepository: Implementa ICarrinhoRepository para operações CRUD em carrinhos.
    - PedidoRepository: Implementa IPedidoRepository para operações CRUD em pedidos.

    - UsuarioRepository: Implementa IUsuarioRepository para operações CRUD em usuários.
    - EnderecoRepository: Implementa IEnderecoRepository para operações CRUD em endereços.
- **Serviços de Comunicação:**
    - CepValidationService: Implementa a lógica para consultar a API de CEP.
    - CpfValidationService: Implementa a lógica para consultar a API de CPF.
- **Configuração de Segurança:**
    - SenhaService: Implementa criptografia e descriptação de senhas.
    - TokenService: Gera e valida tokens de segurança.

Responsabilidades:

- Implementar o acesso a dados e interações com APIs externas.
- Gerenciar segurança, criptografia e configurações de serviços básicos.

Exemplo de Funcionamento: Processamento de Pedido

Fluxo do Pedido:

1. **Entrada do Cliente:** ProdutoController exibe detalhes do produto e CarrinhoController adiciona o produto ao carrinho.
2. **Processamento na Camada de Aplicação:** CarrinhoService atualiza o carrinho e CheckoutService prepara o pedido.
3. **Lógica de Negócio na Camada de Domínio:** Pedido é criado e ItemPedido é adicionado ao pedido.
4. **Persistência na Camada de Infraestrutura:** PedidoRepository salva o pedido no banco de dados e SenhaService gerencia a segurança.
5. **Resposta ao Cliente:** CarrinhoController exibe a confirmação do pedido e CheckoutService finaliza a compra.