Definição dos Modelos de Dados (Entidades) para o Site de Vendas de Comidas

Olá! Nesta etapa, vamos definir as estruturas de dados (Modelos ou Entidades) que seu aplicativo usará. Em um projeto .NET MVC, essas são classes C# que geralmente ficam na pasta `Models`. Elas representam as tabelas que teremos no nosso banco de dados MySQL.

Para o seu site de vendas de comidas, precisaremos das seguintes entidades principais:

1. `Usuario`: Para armazenar informações dos usuários que farão login com o Google.
2. `Produto`: Para representar os alimentos que você venderá (nome, descrição, preço, imagem).
3. `Agendamento`: Para guardar os detalhes dos pedidos agendados pelos usuários.
4. `ItemPedido`: Para detalhar quais produtos e em que quantidade compõem cada agendamento.

Vamos detalhar cada uma delas.

1. Modelo `Usuario`

Este modelo armazenará as informações básicas do usuário obtidas após o login com o Google.

```
 // Pasta: Models/Usuario.cs
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

public class Usuario
{
    [Key] // Indica que esta é a chave primária. Para login com Google, o ID fornecido por eles é uma boa chave.
    public string Id { get; set; } // Geralmente o "Subject ID" do Google.

    [Required(ErrorMessage = "O nome do usuário é obrigatório.")]
    [StringLength(100, ErrorMessage = "O nome deve ter no máximo 100 caracteres.")]
    public string Nome { get; set; }
```

```
    [Required(ErrorMessage = "O email do usuário é obrigatório.")]
    [EmailAddress(ErrorMessage = "Formato de email inválido.")]
    public string Email { get; set; }

    public string FotoUrl { get; set; } // URL da foto de perfil do Google (opcional)

    // Propriedade de navegação: Um usuário pode ter vários agendamentos
    public virtual ICollection<Agendamento> Agendamentos { get; set; }

    public Usuario()
    {
        Agendamentos = new HashSet<Agendamento>();
    }
}
```

Explicação: * `[Key]`: Define `Id` como a chave primária da tabela `Usuarios`. * `[Required]`: Indica que o campo é obrigatório. * `[StringLength]`: Define um tamanho máximo para a string. * `[EmailAddress]`: Valida se o formato do email é válido. * `ICollection<Agendamento> Agendamentos`: É uma propriedade de navegação. Isso significa que, a partir de um objeto `Usuario`, você poderá acessar todos os agendamentos feitos por ele. O Entity Framework Core (que veremos mais à frente) usa isso para entender o relacionamento entre as tabelas.

2. Modelo `Produto`

Este modelo representará cada item de comida disponível no seu cardápio.

```
// Pasta: Models/Produto.cs
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

public class Produto
{
    [Key] // Chave primária
    public int Id { get; set; } // ID autoincrementável

    [Required(ErrorMessage = "O nome do produto é obrigatório.")]

[StringLength(100, ErrorMessage = "O nome do produto deve ter no máximo 100 caracteres.")]
    public string Nome { get; set; }

    [Required(ErrorMessage = "A descrição do produto é obrigatória.")]
    [StringLength(500, ErrorMessage = "A descrição deve ter no máximo 500 caracteres.")]
    public string Descricao { get; set; }
```

```
    [Required(ErrorMessage = "O preço do produto é obrigatório.")]
    [Column(TypeName = "decimal(18,2)")] // Define o tipo de dado no banco para precisão monetária
    [Range(0.01, 10000.00, ErrorMessage = "O preço deve ser entre 0.01 e 10000.00")]
    public decimal Preco { get; set; }

    [Display(Name = "URL da Imagem")]
    [StringLength(2048)] // URL pode ser longa
    public string UrlImagem { get; set; }

    [StringLength(50)]
    public string Categoria { get; set; } // Ex: "Prato Principal", "Bebida", "Sobremesa"

    // Propriedade de navegação: Um produto pode estar em vários itens de pedido
    public virtual ICollection<ItemPedido> ItensPedido { get; set; }

    public Produto()
    {
        ItensPedido = new HashSet<ItemPedido>();
    }
}
```

Explicação: * `[Column(TypeName = "decimal(18,2)")]` : Especifica que no banco de dados MySQL, o campo `Preco` deve ser do tipo `DECIMAL` com 18 dígitos no total e 2 casas decimais. Isso é importante para evitar problemas de arredondamento com valores monetários. * `[Range]` : Define um valor mínimo e máximo para o preço. * `[Display(Name = "URL da Imagem")]` : Define como o nome do campo será exibido em algumas partes da interface (ex: labels de formulários gerados automaticamente).

3. Modelo `Agendamento`

Este modelo guardará as informações de um pedido agendado por um usuário.

```
// Pasta: Models/Agendamento.cs
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

public class Agendamento
{
    [Key]
    public int Id { get; set; }

    [Required]
```

```
    public string UsuarioId { get; set; } // Chave estrangeira para a tabela Usuario
    [ForeignKey("UsuarioId")]
    public virtual Usuario Usuario { get; set; } // Propriedade de navegação para o
Usuario

    [Required(ErrorMessage = "A data e hora do agendamento são obrigatórias.")]
    [Display(Name = "Data e Hora do Agendamento")]
    public DateTime DataHoraAgendamento { get; set; }

    [Display(Name = "Data e Hora da Criação do Pedido")]
    public DateTime DataHoraCriacao { get; set; }

    [Required(ErrorMessage = "O status do agendamento é obrigatório.")]
    [StringLength(50)]
    public string Status { get; set; } // Ex: "Pendente", "Confirmado", "Em Preparo",
"Pronto para Retirada/Entrega", "Entregue", "Cancelado"

    [StringLength(500, ErrorMessage = "As observações devem ter no máximo 500
caracteres.")]
    public string Observacoes { get; set; }

    [Column(TypeName = "decimal(18,2)")]
    public decimal ValorTotal { get; set; }

    // Propriedade de navegação: Um agendamento tem vários itens de pedido
    public virtual ICollection<ItemPedido> ItensPedido { get; set; }

    public Agendamento()
    {
        ItensPedido = new HashSet<ItemPedido>();
        DataHoraCriacao = DateTime.UtcNow; // Define a data de criação
automaticamente
        Status = "Pendente"; // Status inicial padrão
    }
}
```

Explicação: * `UsuarioId` e `Usuario`: Definem o relacionamento com a tabela `Usuario`. `UsuarioId` é a chave estrangeira, e `Usuario` é a propriedade de navegação que permite acessar o objeto `Usuario` relacionado. * `[ForeignKey("UsuarioId")]`: Especifica qual propriedade é a chave estrangeira. * `DataHoraCriacao = DateTime.UtcNow;`: No construtor, definimos que a data de criação do pedido será o momento atual em UTC.

4. Modelo `ItemPedido`

Este modelo é uma tabela de junção que representa os itens específicos dentro de um agendamento. Um agendamento pode ter vários produtos, e um produto pode estar em vários agendamentos. Esta tabela resolve esse relacionamento Muitos-para-Muitos, e

também armazena a quantidade de cada produto no pedido e o preço unitário no momento da compra (para histórico).

```
// Pasta: Models/ItemPedido.cs
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

public class ItemPedido
{
    [Key]
    public int Id { get; set; }

    [Required]
    public int AgendamentoId { get; set; } // Chave estrangeira para Agendamento
    [ForeignKey("AgendamentoId")]
    public virtual Agendamento Agendamento { get; set; } // Navegação para
Agendamento

    [Required]
    public int ProdutoId { get; set; } // Chave estrangeira para Produto
    [ForeignKey("ProdutoId")]
    public virtual Produto Produto { get; set; } // Navegação para Produto

    [Required(ErrorMessage = "A quantidade é obrigatória.")]
    [Range(1, 100, ErrorMessage = "A quantidade deve ser entre 1 e 100.")]
    public int Quantidade { get; set; }

    [Required(ErrorMessage = "O preço unitário é obrigatório.")]
    [Column(TypeName = "decimal(18,2)")]
    public decimal PrecoUnitario { get; set; } // Preço do produto no momento da
compra
}
```

Explicação: * Este modelo tem chaves estrangeiras para `Agendamento` e `Produto`, ligando um item específico de um pedido ao agendamento geral e ao produto do cardápio. * `PrecoUnitario`: É importante armazenar o preço do produto no momento em que o pedido foi feito, pois os preços no cardápio podem mudar com o tempo.

Próximos Passos (Resumo do que virá depois):

1. **DbContext:** Você precisará criar uma classe que herda de `DbContext` (do Entity Framework Core). Esta classe representa a sessão com o banco de dados e permite que você consulte e salve dados. Nela, você declarará `DbSet<T>` para cada um dos seus modelos (ex: `public DbSet<Usuario> Usuarios { get; set; }`).

2. **String de Conexão:** Você configurará a string de conexão no arquivo `appsettings.json` para que seu aplicativo saiba como se conectar ao seu servidor MySQL local.
3. **Registro do DbContext:** No arquivo `Program.cs` (ou `Startup.cs`), você registrará o seu `DbContext` e informará ao Entity Framework Core que você está usando MySQL.
4. **Migrations:** Você usará comandos do .NET CLI (como `dotnet ef migrations add NomeDaMigration` e `dotnet ef database update`) para criar o esquema do banco de dados (as tabelas) baseado nos seus modelos C#.

Este é um passo fundamental! Ter modelos bem definidos facilita muito o restante do desenvolvimento. Salve esses códigos em arquivos `.cs` dentro da pasta `Models` do seu projeto quando ele for criado.

Lembre-se que estes são os modelos iniciais. Conforme o projeto evolui, você pode precisar adicionar mais campos ou até mesmo novos modelos.