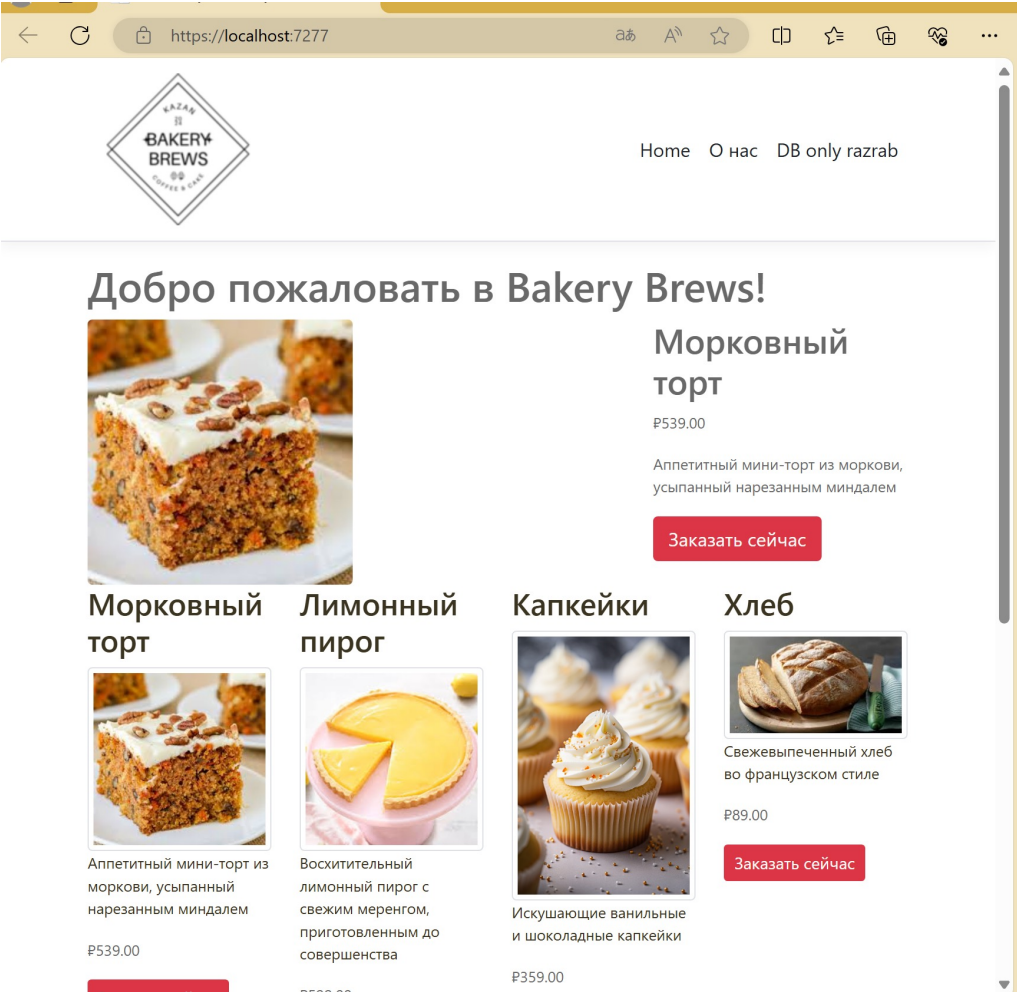


Сайт Bakery Brews предоставляет систему онлайн-заказа, которая позволяет пользователям размещать заказы на различные кондитерские изделия.

С технической точки зрения сайт представляет:

- использование файловой базы данных для хранения информации о продукте
- базовый доступ к данным
- создание и обработка форм



Этот сайт создан с использованием ASP.NET Core 2.2 и [Entity Framework Core](#), разработанного Microsoft.

Работа с данными

Создание БД

- Добавление ядра Entity Framework

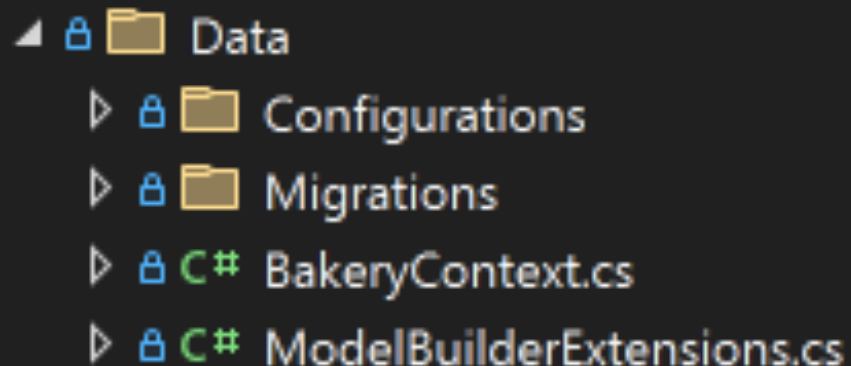
EF Core включает в себя ряд *поставщиков* - компонентов, которые работают с конкретными базами данных. Многие провайдеры поддерживаются третьими лицами. Команда EF Core поддерживает поставщиков для SQL Server, а SQLite - кросс-платформенная файловая база данных, которая будет использоваться в моем сайте.

- Чтобы его установить я использовал следующую команду:

```
dotnet add package Microsoft.EntityFrameworkCore.Sqlite
```

- Контекст

Основным компонентом EF Core, который вы будете использовать для связи с базой данных, является класс, который происходит от `DbContext`, известного как **контекст**. Контекст представляет собой сеанс с базой данных и предоставляет API для связи с ней.



Код BakeryContext.cs

```
using Bakery.Models;
using Microsoft.EntityFrameworkCore;
namespace Bakery.Data
{
    public class BakeryContext : DbContext
    {
        public BakeryContext(DbContextOptions<BakeryContext> options) : base(options) { }
        public DbSet<Product> Products { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite(@"Data source=Bakery.db");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.ApplyConfiguration(new ProductConfiguration()).Seed();
        }
    }
}
```

Контекст имеет свойство - DbSet с именем Products. Класс DbSet сопоставляется с таблицей в базе данных.

Данные и создание миграции

Перед тем как работать с данными необходимо создать файл конфигурации. Файл конфигурации в Entity Framework Core используется для того, чтобы точно указать, как данные вашего приложения должны быть сохранены в базе данных. Его можно было указать в методе `OnModelCreating` в коде `BakeryContext`, но я решил создать отдельный класс.

```
using Bakery.Models;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
namespace Bakery.Data
{
    public class ProductConfiguration : IEntityTypeConfiguration<Product>
    {
        public void Configure(EntityTypeBuilder<Product> builder)
        {
            builder.Property(p => p.ImageName).HasColumnName("ImageFileName");
        }
    }
}
```

Класс реализует интерфейс `IEntityTypeConfiguration<TEntity>`, который имеет один метод: `Configure`. Конфигурации определяются в этом методе. В этом случае свойство `ImageName` сопоставлено со столбцом с именем `"ImageFileName"`. Поведение по умолчанию заключается в сопоставлении столбцов с тем же именем, что и свойство.

Данные и создание миграции

Заполним нашу БД:

Код `ModelBuilderExtensions.cs`

```
using Bakery.Models;
using Microsoft.EntityFrameworkCore;

namespace Bakery.Data
{
    public static class ModelBuilderExtensions
    {
        public static ModelBuilder Seed(this ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Product>().HasData(
                new Product
                {
                    Id = 1,
                    Name = "Морковный торт",
                    Description = "Аппетитный мини-торт из моркови, усыпанный нарезанным миндалем",
                    Price = 539m, // ≈ $8.99 * 60
                    ImageName = "carrot_cake.jpg"
                },
                new Product
                {
                    Id = 2,
                    Name = "Лимонный пирог",
                    Description = "Восхитительный лимонный пирог с свежим меренгом, приготовленным до совершенства",
                    Price = 599m, // ≈ $9.99 * 60
                    ImageName = "lemon_tart.jpg"
                },
                .....Остальные товары.....
            );
            return modelBuilder;
        }
    }
}
```

Тело метода использует метод `HasData`, представленный в EF Core, для настройки указанного объекта для начальных данных. Значения, предоставленные для каждой сущности, отражают значения из исходного шаблона и включают ключевые значения.

Данные и создание миграции

Функция миграции Entity Framework Core позволяет вносить изменения в модель, а затем распространять эти изменения в схеме базы данных.

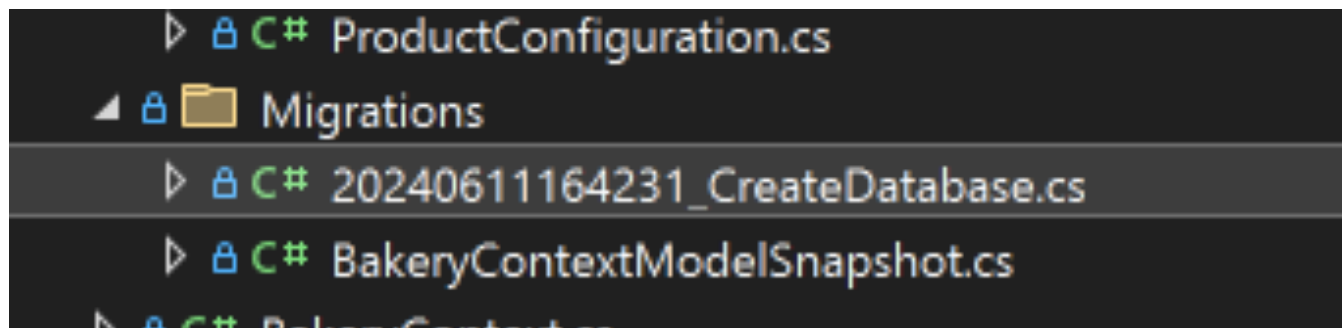
Функция миграции также может быть использована для создания базы данных, если она не существует.

Но прежде чем вы это сделаете, нужно установить инструменты, которые запускают код миграции.

```
dotnet add package Microsoft.EntityFrameworkCore.Design
```

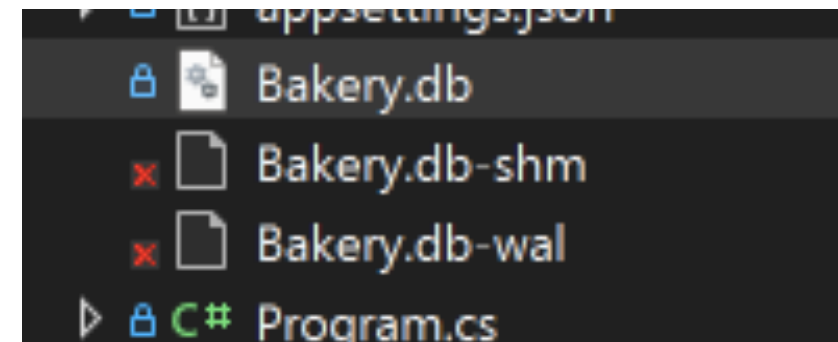
А теперь создаем миграцию, я добавлю ее в директорию Data/Migrations, чтобы не нарушать порядок в коде

```
dotnet ef migrations add CreateDatabase --output-dir Data/Migrations
```



Выполним миграцию:

```
dotnet ef database update
```



Для наглядности открою БД

Table: Products					
	Id	Name	Description	Price	ImageFileName
	Filter	Filter	Filter	Filter	Filter
1	1	Морковный торт	Аппетитный мини-то...	539.0	carrot_cake.jpg
2	2	Лимонный пирог	Восхитительный ...	599.0	lemon_tart.jpg
3	3	Капкейки	Искушающие ...	359.0	cupcakes.jpg
4	4	Хлеб	Свежевыпеченный ...	89.0	bread.jpg
5	5	Грушевый пирог	Грушевый пирог с ...	359.0	pear_tart.jpg
6	6	Шоколадный торт	Пышный шоколадны...	539.0	chocolate_cake.jpg
7	10	Кофе	Описание кофе	290.0	coffee.jpg

Можем видеть что данные успешно добавлены в БД

Отображение данных на главной странице

Обновим страницу

Все продукты отображаются вместе с их описаниями, изображением и ценой, при этом один из продуктов выбран случайным образом, чтобы появиться в качестве рекомендуемого продукта в верхней части страницы.


Свойства Products и FeaturedProduct:

- Products — это список всех продуктов, которые будут отображаться на странице.
- FeaturedProduct — это один продукт, который выбран случайным образом, из метода OnGetAsync, для отображения в качестве рекомендуемого продукта.

Код Index.cshtml.cs

```
namespace Bakery.Pages
{
    public class IndexModel : PageModel
    {
        private readonly BakeryContext db;
        public IndexModel(BakeryContext db) => this.db = db;
        public List<Product> Products { get; set; } = new
        List<Product>();
        public Product FeaturedProduct { get; set; }
        public async Task OnGetAsync()
        {
            Products = await db.Products.ToListAsync();
            FeaturedProduct = Products.ElementAt(new
            Random().Next(Products.Count));
        }
    }
}
```

Добро пожаловать в Bakery Brews!




Хлеб

Р89.00

Свежевыпеченный хлеб во французском стиле

Заказать сейчас

Морковный торт




Аппетитный мини-торт из моркови, усыпанный нарезанным миндалем

Р539.00

Заказать сейчас

Лимонный пирог




Восхитительный лимонный пирог с свежим меренгом, приготовленным до совершенства

Р599.00

Заказать сейчас

Капкейки




Искушающие ванильные и шоколадные капкейки

Р359.00

Заказать сейчас

Хлеб




Свежевыпеченный хлеб во французском стиле

Р89.00

Заказать сейчас

Грушевый пирог




Грушевый пирог с глазурью, посыпанный нарезанным миндалем и щепоткой корицы

Р359.00

Заказать сейчас

Шоколадный торт




Пышный шоколадный крем покрывает этот торт, созданный для любителей шоколада

Р290.00

Заказать сейчас

Кофе




Описание кофе

Р290.00

Заказать сейчас

Добро пожаловать в Bakery Brews!




Грушевый пирог

Р359.00

Грушевый пирог с глазурью, посыпанный нарезанным миндалем и щепоткой корицы

Заказать сейчас

Морковный торт




Аппетитный мини-торт из моркови, усыпанный нарезанным миндалем

Р539.00

Заказать сейчас

Лимонный пирог




Восхитительный лимонный пирог с свежим меренгом, приготовленным до совершенства

Р599.00

Заказать сейчас

Капкейки




Искушающие ванильные и шоколадные капкейки

Р359.00

Заказать сейчас

Хлеб




Свежевыпеченный хлеб во французском стиле

Р89.00

Заказать сейчас

Грушевый пирог




Грушевый пирог с глазурью, посыпанный нарезанным миндалем и щепоткой корицы

Р359.00

Заказать сейчас

Шоколадный торт




Пышный шоколадный крем покрывает этот торт, созданный для любителей шоколада

Р290.00

Заказать сейчас

Кофе



Описание кофе

Р290.00

Заказать сейчас

В верхней части показан рекомендуемый продукт. Нижняя секция просматривает все продукты и отображает их миниатюрное изображение. Каждый продукт включает в себя гиперссылку, стилизованную как кнопка (с использованием стиля Bootstrap). Гиперссылка генерируется помощником якорного тега, который включает в себя атрибут asp-route. Этот атрибут используется для передачи данных в виде значений маршрута на целевую страницу.

Добро пожаловать в Bakery Brews!



Лимонный пирог

Р599.00

Восхитительный лимонный пирог с свежим меренгом, приготовленным до совершенства

Заказать сейчас

Морковный торт



Аппетитный мини-торт из моркови, усыпанный нарезанным миндалем

Р539.00

Заказать сейчас

Лимонный пирог



Восхитительный лимонный пирог с свежим меренгом, приготовленным до совершенства

Р599.00

Заказать сейчас

Капкейки



Искушающие ванильные и шоколадные капкейки

Р359.00

Заказать сейчас

Хлеб



Свежевыпеченный хлеб во французском стиле

Р89.00

Заказать сейчас

Грушевый пирог



Грушевый пирог с глазурью, посыпанный нарезанным миндалем и щепоткой корицы

Р359.00

Заказать сейчас

Шоколадный торт



Пышный шоколадный крем покрывает этот торт, созданный для любителей шоколада

Р539.00

Заказать сейчас

Кофе



Описание кофе

Р290.00

Заказать сейчас

Код Index.cshtml

```
<h1>Добро пожаловать в Bakery Brews!</h1>
<div id="featuredProduct" class="row">
  <div class="col-sm-8">
    
  </div>
  <div id="featuredProductInfo" class="col-sm-4">
    <div id="productInfo">
      <h2>@Model.FeaturedProduct.Name</h2>
      <p class="price">P@string.Format("{0:f}", Model.FeaturedProduct.Price)</p>
      <p class="description">@Model.FeaturedProduct.Description</p>
    </div>
    <div id="callToAction">
      <a class="btn btn-danger order-button" asp-page="/order" asp-route-id="@Model.FeaturedProduct.Id" title="Заказать"
        @Model.FeaturedProduct.Name">Заказать сейчас</a>
    </div>
  </div>
</div>
<div id="productsWrapper" class="row">
  @foreach (var product in Model.Products)
  {
    <div class="col-sm-3">
      <a asp-page="/order" asp-route-id="@product.Id" title="Заказать" @product.Name">
        <div class="productInfo">
          <h3>@product.Name</h3>
          
          <p class="description">@product.Description</p>
        </div>
      </a>
      <div class="action">
        <p class="price float-left">P@string.Format("{0:f}", product.Price)</p>
        <a class="btn btn-sm btn-danger order-button float-right" asp-page="/order" asp-route-id="@product.Id" title="Заказать"
          @product.Name">Заказать сейчас</a>
      </div>
    </div>
  }
</div>
```

