Universidad Fidélitas.

Programación Avanzada en Web

Practica Evaluada #2

Integrante: JOSUÉ ALEXANDER ZAMORA CONEJO

Profesor: ARIAS BRENES JOSE ANDRES

MAR 2025

Propuesta de Solución

Para la empresa La Frutica, se propone desarrollar una base de datos relacional utilizando Entity Framework Core con SQL Server para gestionar la base de datos. EF Core es un ORM (Object-Relational Mapping) que permite interactuar con la base de datos de manera eficiente mediante el uso de clases y objetos en C#.

Tecnologías

- ASP.NET Core Web App MVC: Para los modelos, vistas y controlador.
- Entity Framework Core: Para la inyección con la base de datos.
- SQL Server: Como base de datos.

Arquitectura

- Modelo: Definen los modelos de los datos y el DB Context.
- Vista: Visualiza el CRUD de los productos y las categorías.
- Controlador: Gestiona la manipulación de datos.

Diseño de los modelos

Para el diseño de la base de datos se requiere que el producto pueda tener varias categorías.

```
Tabla de productos:
public class Producto
{
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string Descripcion { get; set; }
    public decimal Precio { get; set; }
    public int Cantidad { get; set; }
    public ICollection<ProdCategoria>? ProdCategorias { get; set; }
}
```

```
Tabla de Categorías:
public class Categoria
 {
     public int Id { get; set; }
```

```
    public string Nombre { get; set; }
    public ICollection<ProdCategoria>? ProdCategorias { get; set; }
}
```

Tabla de producto-categorías para relacionar varias categorías a un producto:

```
public class ProdCategoria
{
    public int ProductoId { get; set; }
    public Producto? Producto { get; set; }
    public int CategoriaId { get; set; }
    public Categoria? Categoria { get; set; }
}
```

DB Context

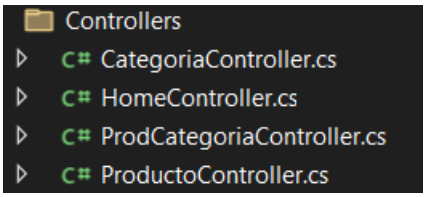
Se definen los DbSet y el método OnModelCreating para configurar las propiedades y relaciones de las tablas.

```
using Microsoft.EntityFrameworkCore;

namespace PracticaN2_JosueZamoraConejo.Models
{
    public class FructicaDBContext:DbContext
    {
        public DbSet<Producto> Productos { get; set; }
        public DbSet<Categoria> Categorias { get; set; }
        public DbSet<ProdCategoria> ProdCategorias { get; set; }

        public FructicaDBContext(DbContextOptions<FructicaDBContext> options) :
base(options) { }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Producto>(Producto =>
            {
                Producto.HasKey(e => e.Id);
                Producto.Property(n => n.Nombre).IsRequired().HasMaxLength(300);
                Producto.Property(n =>n.Descripcion).IsRequired().HasMaxLength(500);
                Producto.Property(n =>n.Precio).IsRequired().HasColumnType("float");
                Producto.Property(n =>n.Cantidad).IsRequired().HasColumnType("int");

            });
            modelBuilder.Entity<Categoria>(Categoria =>
            {
                Categoria.HasKey(e => e.Id);
                Categoria.Property(n => n.Nombre).IsRequired().HasMaxLength(300);

            });
            modelBuilder.Entity<ProdCategoria>()
                .HasKey(pc => new { pc.ProductoId, pc.CategoriaId });
```

```
            modelBuilder.Entity<ProdCategoria>()
                .HasOne(pc => pc.Producto)
                .WithMany(p => p.ProdCategorias)
                .HasForeignKey(pc => pc.ProductoId);

            modelBuilder.Entity<ProdCategoria>()
                .HasOne(pc => pc.Categoria)
                .WithMany(c => c.ProdCategorias)
                .HasForeignKey(pc => pc.CategoriaId);
        }
    }
}
```

Controlador

Se definen las acciones en los controladores para manejar el CRUD de los Productos, Categorías y Productos/Categorías.



Vistas

Por ultimo se crear las vistas para gestionar los productos.