**SoadPizza Project Documentation**

**Overview**

The SoadPizza project is a console application designed to manage a pizza store. It allows users to perform CRUD operations on customers and products, and to view orders for specific customers. The application uses Entity Framework Core for data access and PostgreSQL as the database.

**Setup Instructions**

**Prerequisites**

* .NET SDK
* PostgreSQL
* Visual Studio or any other C# IDE

**Configuration**

1. **Database Configuration**:
   * Ensure PostgreSQL is installed and running.
   * Create a database named SoadPizzaDB.
   * Update the connection string in App.config with your PostgreSQL server details.

**<!-- App.config -->**

**<connectionStrings>**

**<add name="SoadPizzaDB" connectionString="Host=localhost;Database=SoadPizzaDB;Username=yourusername;Password=yourpassword" providerName="Npgsql" />**

**</connectionStrings>**

1. **Migrations**:
   1. Open the terminal in the project directory and run the following commands to apply migrations:

dotnet ef migrations add InitialCreate

dotnet ef database update

**Running the Application**

* Open the project in Visual Studio.
* Set MainApp as the startup project.
* Run the application.

**Project Structure**

Main Components

1. **MainApp.cs**
   * Entry point of the application.
   * Configures the database context and initializes the ConsoleHandler.

using System;

using System.Configuration;

using Microsoft.EntityFrameworkCore;

using SoadPizza.Services;

namespace SoadPizza.ConsoleApp

{

class MainApp

{

static void Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<SoadPizzaDBContext>();

var connectionString = ConfigurationManager.ConnectionStrings["SoadPizzaDB"].ConnectionString;

optionsBuilder.UseNpgsql(connectionString);

using (var context = new SoadPizzaDBContext(optionsBuilder.Options))

{

var pizzaService = new PizzaService(context);

var consoleHandler = new ConsoleHandler(pizzaService, context);

consoleHandler.Run();

}

}

}

}

**2. ConsoleHandler.cs**

* Handles user input and interacts with the PizzaService to perform CRUD operations

using System;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using SoadPizza.Model;

using SoadPizza.Services;

namespace SoadPizza.ConsoleApp

{

public class ConsoleHandler

{

private readonly PizzaService \_pizzaService;

private readonly SoadPizzaDBContext \_context;

public ConsoleHandler(PizzaService pizzaService, SoadPizzaDBContext context)

{

\_pizzaService = pizzaService;

\_context = context;

}

public void Run()

{

while (true)

{

Console.WriteLine("Choose an option:");

Console.WriteLine("1. Add Customer");

Console.WriteLine("2. Add Product");

Console.WriteLine("3. Show Customers");

Console.WriteLine("4. Show Products");

Console.WriteLine("5. Show Orders for a Customer");

Console.WriteLine("6. Show Products with Price Greater Than");

Console.WriteLine("7. Exit");

var choice = Console.ReadLine();

switch (choice)

{

case "1":

AddCustomer();

break;

case "2":

AddProduct();

break;

case "3":

ShowCustomers();

break;

case "4":

ShowProducts();

break;

case "5":

ShowOrdersForCustomer();

break;

case "6":

ShowProductsWithPriceGreaterThan();

break;

case "7":

return;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

}

private void AddCustomer()

{

Console.WriteLine("Enter First Name:");

var firstName = Console.ReadLine();

Console.WriteLine("Enter Last Name:");

var lastName = Console.ReadLine();

Console.WriteLine("Enter Address:");

var address = Console.ReadLine();

Console.WriteLine("Enter Phone:");

var phone = Console.ReadLine();

if (string.IsNullOrEmpty(firstName) || string.IsNullOrEmpty(lastName))

{

Console.WriteLine("First Name and Last Name are required.");

return;

}

var customer = new Customer

{

FirstName = firstName ?? string.Empty,

LastName = lastName ?? string.Empty,

Address = address,

Phone = phone

};

try

{

\_pizzaService.AddCustomer(customer);

Console.WriteLine("Customer added successfully.");

}

catch (Exception ex)

{

Console.WriteLine($"Error adding customer: {ex.Message}");

}

}

private void AddProduct()

{

Console.WriteLine("Enter Product Name:");

var name = Console.ReadLine();

Console.WriteLine("Enter Product Price:");

if (decimal.TryParse(Console.ReadLine(), out var price))

{

if (string.IsNullOrEmpty(name))

{

Console.WriteLine("Product Name is required.");

return;

}

var product = new Product

{

Name = name ?? string.Empty,

Price = price

};

try

{

\_pizzaService.AddProduct(product);

Console.WriteLine("Product added successfully.");

}

catch (Exception ex)

{

Console.WriteLine($"Error adding product: {ex.Message}");

}

}

else

{

Console.WriteLine("Invalid price. Please try again.");

}

}

private void ShowCustomers()

{

var customers = \_pizzaService.GetCustomers();

Console.WriteLine("Customers:");

foreach (var customer in customers)

{

Console.WriteLine($"ID: {customer.Id}, Name: {customer.FirstName} {customer.LastName}");

}

}

private void ShowProducts()

{

var products = \_pizzaService.GetProducts();

Console.WriteLine("Products:");

foreach (var product in products)

{

Console.WriteLine($"Product ID: {product.Id}, Name: {product.Name}, Price: {product.Price}");

}

}

private void ShowOrdersForCustomer()

{

Console.WriteLine("Enter Customer ID:");

if (int.TryParse(Console.ReadLine(), out var customerId))

{

var orders = \_context.Orders

.Where(o => o.CustomerId == customerId)

.Include(o => o.OrderDetails)

.ThenInclude(od => od.Product)

.ToList();

Console.WriteLine("\nOrders:");

foreach (var order in orders)

{

Console.WriteLine($"Order ID: {order.Id}, Order Placed: {order.OrderPlaced}");

foreach (var detail in order.OrderDetails)

{

Console.WriteLine($"Product: {detail.Product.Name}, Quantity: {detail.Quantity}");

}

}

}

else

{

Console.WriteLine("Invalid Customer ID. Please try again.");

}

}

private void ShowProductsWithPriceGreaterThan()

{

Console.WriteLine("Enter Minimum Price:");

if (decimal.TryParse(Console.ReadLine(), out var minPrice))

{

var products = \_context.Products

.Where(p => p.Price > minPrice)

.ToList();

Console.WriteLine("\nProducts:");

foreach (var product in products)

{

Console.WriteLine($"Product ID: {product.Id}, Name: {product.Name}, Price: {product.Price}");

}

}

else

{

Console.WriteLine("Invalid price. Please try again.");

}

}

}

}

1. **PizzaService.cs**
   1. Provides methods to interact with the database for CRUD operations on Customer and Product entities.
2. using System.Collections.Generic;
3. using System.Linq;
4. using SoadPizza.Model;
5. namespace SoadPizza.Services
6. {
7. public class PizzaService
8. {
9. private readonly SoadPizzaDBContext \_context;
10. public PizzaService(SoadPizzaDBContext context)
11. {
12. \_context = context;
13. }
14. // Create
15. public void AddCustomer(Customer customer)
16. {
17. \_context.Customers.Add(customer);
18. \_context.SaveChanges();
19. }
20. public void AddProduct(Product product)
21. {
22. \_context.Products.Add(product);
23. \_context.SaveChanges();
24. }
25. // Read
26. public List<Customer> GetCustomers()
27. {
28. return \_context.Customers.ToList();
29. }
30. public List<Product> GetProducts()
31. {
32. return \_context.Products.ToList();
33. }
34. // Update
35. public void UpdateCustomer(Customer customer)
36. {
37. \_context.Customers.Update(customer);
38. \_context.SaveChanges();
39. }
40. public void UpdateProduct(Product product)
41. {
42. \_context.Products.Update(product);
43. \_context.SaveChanges();
44. }
45. // Delete
46. public void DeleteCustomer(int customerId)
47. {
48. var customer = \_context.Customers.Find(customerId);
49. if (customer != null)
50. {
51. \_context.Customers.Remove(customer);
52. \_context.SaveChanges();
53. }
54. }
55. public void DeleteProduct(int productId)
56. {
57. var product = \_context.Products.Find(productId);
58. if (product != null)
59. {
60. \_context.Products.Remove(product);
61. \_context.SaveChanges();
62. }
63. }
64. }
65. }
66. **SoadPizzaDBContext.cs**
    * Defines the database context and the DbSet properties for the entities.

* using Microsoft.EntityFrameworkCore;
* using SoadPizza.Model;
* namespace SoadPizza.Services
* {
* public class SoadPizzaDBContext : DbContext
* {
* public SoadPizzaDBContext(DbContextOptions<SoadPizzaDBContext> options) : base(options) { }
* public DbSet<Customer> Customers { get; set; }
* public DbSet<Order> Orders { get; set; }
* public DbSet<OrderDetail> OrderDetails { get; set; }
* public DbSet<Product> Products { get; set; }
* protected override void OnModelCreating(ModelBuilder modelBuilder)
* {
* // Define relationships and constraints here
* }
* }
* }

1. **Model Classes**
   * Define the entities used in the application.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

namespace SoadPizza.Model

{

public class Customer

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string? Address { get; set; }

public string? Phone { get; set; }

public ICollection<Order> Orders { get; set; }

}

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

[Column(TypeName = "decimal(6,2)")]

public decimal Price { get; set; }

}

public class Order

{

public int Id { get; set; }

public DateTime OrderPlaced { get; set; }

public DateTime OrderFulfilled { get; set; }

public int CustomerId { get; set; }

public Customer Customer { get; set; }

public ICollection<OrderDetail> OrderDetails { get; set; }

}

public class OrderDetail

{

public int Id { get; set; }

public int OrderId { get; set; }

public int ProductId { get; set; }

public int Quantity { get; set; }

public Order Order { get; set; }

public Product Product { get; set; }

}

}

**LINQ Usage**

LINQ (Language Integrated Query) is used extensively in the ConsoleHandler class to query the database. Here are some examples:

* **ShowOrdersForCustomer**:
  + This method uses LINQ to filter orders by CustomerId and includes related OrderDetails and Product entities.

private void ShowOrdersForCustomer()

{

Console.WriteLine("Enter Customer ID:");

if (int.TryParse(Console.ReadLine(), out var customerId))

{

var orders = \_context.Orders

.Where(o => o.CustomerId == customerId)

.Include(o => o.OrderDetails)

.ThenInclude(od => od.Product)

.ToList();

Console.WriteLine("\nOrders:");

foreach (var order in orders)

{

Console.WriteLine($"Order ID: {order.Id}, Order Placed: {order.OrderPlaced}");

foreach (var detail in order.OrderDetails)

{

Console.WriteLine($"Product: {detail.Product.Name}, Quantity: {detail.Quantity}");

}

}

}

else

{

Console.WriteLine("Invalid Customer ID. Please try again.");

}

}

1. **ShowProductsWithPriceGreaterThan**:
   * This method uses LINQ to filter products by price.

private void ShowProductsWithPriceGreaterThan()

{

Console.WriteLine("Enter Minimum Price:");

if (decimal.TryParse(Console.ReadLine(), out var minPrice))

{

var products = \_context.Products

.Where(p => p.Price > minPrice)

.ToList();

Console.WriteLine("\nProducts:");

foreach (var product in products)

{

Console.WriteLine($"Product ID: {product.Id}, Name: {product.Name}, Price: {product.Price}");

}

}

else

{

Console.WriteLine("Invalid price. Please try again.");

}

}

**Fluent API Usage**

The Fluent API is used in the SoadPizzaDBContext class to configure the model and relationships between entities. Here is an example:

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Define relationships and constraints here

// Example: Configuring a one-to-many relationship between Customer and Order

modelBuilder.Entity<Order>()

.HasOne(o => o.Customer)

.WithMany(c => c.Orders)

.HasForeignKey(o => o.CustomerId);

// Example: Configuring a one-to-many relationship between Order and OrderDetail

modelBuilder.Entity<OrderDetail>()

.HasOne(od => od.Order)

.WithMany(o => o.OrderDetails)

.HasForeignKey(od => od.OrderId);

// Example: Configuring a one-to-many relationship between Product and OrderDetail

modelBuilder.Entity<OrderDetail>()

.HasOne(od => od.Product)

.WithMany()

.HasForeignKey(od => od.ProductId);

}

**Conclusion**

This documentation provides an overview of the SoadPizza project, setup instructions, and descriptions of the main components. It also highlights the usage of LINQ and Fluent API for querying and configuring the database. For further details, refer to the source code and comments within the code files.