ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

КУРСОВОЙ ПРОЕКТ

По дисциплине «Компьютерные технологии управления в технических системах»

Выполнили: ст. гр. ТУУ – 171

Дюдя В. А.

Капанин А. А.

Паньков Р. В.

Сесягина А. А.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

**Оглавление**

[Цель курсового проекта 2](#_Toc124767879)

[Предметная область 2](#_Toc124767880)

[Логическое проектирование реляционной БД 2](#_Toc124767881)

[Создание классов, описывающих предметную область 3](#_Toc124767882)

[Тестирование классов из проекта Domain 10](#_Toc124767883)

[Создание слоя для взаимодействия с данными 17](#_Toc124767884)

[Тестирование классов из проекта DataAccessLayer 25](#_Toc124767885)

[Проверка работоспособности 29](#_Toc124767886)

[Вывод 31](#_Toc124767887)

## **Цель курсового проекта**

Целью курсового проекта является построение ORM, закрепление навыков программирования.

## Предметная область

В качестве предметной области была выбрана доставка еды. База данных будет состоять из 4 таблиц(сущностей): покупатель, заказ, блюдо, запись блюда в заказе.

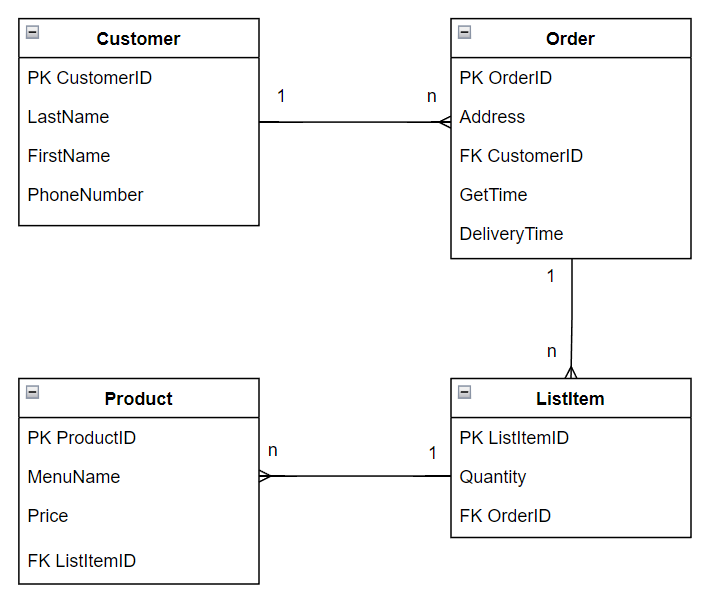
Покупатель: фамилия, имя, номер телефона.

Заказ: адрес, покупатель, время получения заказа, время доставки заказа.

Блюдо: название блюда, цена, запись блюда.

Запись блюда в заказе: количество, заказ.

## Логическое проектирование реляционной БД



***Рисунок 1****– ER диаграмма будущей БД*

## Создание классов, описывающих предметную область

Создадим проект FoodDelivery, а в нем библиотеки классов Domain и Demo.

Далее в Domain будем создавать классы по нашей предметной области.

Листинг файла Customer.cs:

// <copyright file="Customer.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain

{

/// <summary>

/// Покупатель.

/// </summary>

public class Customer : IEquatable<Customer>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Customer"/>.

/// </summary>

/// <param name="lastName"> Фамилия. </param>

/// <param name="firstName"> Имя. </param>

/// <param name="phoneNumber"> Номер телефона. </param>

public Customer(string lastName, string firstName, string phoneNumber)

{

if (string.IsNullOrWhiteSpace(firstName))

{

throw new ArgumentNullException(nameof(firstName));

}

if (string.IsNullOrWhiteSpace(lastName))

{

throw new ArgumentNullException(nameof(lastName));

}

if (string.IsNullOrWhiteSpace(phoneNumber))

{

throw new ArgumentNullException(nameof(phoneNumber));

}

this.Id = Guid.NewGuid();

this.FirstName = firstName;

this.LastName = lastName;

this.PhoneNumber = phoneNumber;

this.FullName =

string.Concat(this.LastName, " ", this.FirstName);

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Customer"/>.

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Customer()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Фамилия.

/// </summary>

public virtual string FirstName { get; protected set; }

/// <summary>

/// Имя.

/// </summary>

public virtual string LastName { get; protected set; }

/// <summary>

/// Полное имя.

/// </summary>

public virtual string FullName { get; protected set; }

/// <summary>

/// Номер телефона.

/// </summary>

public virtual string PhoneNumber { get; protected set; }

/// <summary>

/// Заказы.

/// </summary>

public virtual ISet<Order> Orders { get; set; } = new HashSet<Order>();

/// <summary>

/// Добавление заказов покупателю.

/// </summary>

/// <param name="order"> Заказ. </param>

public virtual void AddOrderToCustomer(Order order)

{

this.Orders.Add(order);

order.Customer = this;

}

/// <inheritdoc/>

public override string ToString()

=> this.FullName;

/// <inheritdoc/>

public virtual bool Equals(Customer? other)

{

return Equals(this.Id, other?.Id);

}

/// <inheritdoc/>

public override int GetHashCode()

=> this.Id.GetHashCode();

}

}

Листинг файла Order.cs:

// <copyright file="Order.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

using System;

namespace Domain

{

/// <summary>

/// Заказ.

/// </summary>

public class Order : IEquatable<Order>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Order"/>.

/// </summary>

/// <param name="address"> Адрес доставки. </param>

/// <param name="customer"> Покупатель. </param>

/// <param name="getTime"> Время получения заказа. </param>

/// <param name="deliveryTime"> Время доставки заказа. </param>

/// <exception cref="ArgumentNullException">Выброс ошибки.</exception>

public Order(string address, Customer customer, DateTime getTime, DateTime deliveryTime)

{

if (string.IsNullOrWhiteSpace(address))

{

throw new ArgumentNullException(nameof(address));

}

this.Id = Guid.NewGuid();

this.Address = address;

this.Customer = customer ?? throw new ArgumentNullException(nameof(customer));

this.GetTime = getTime;

this.DeliveryTime = deliveryTime;

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Order"/>.

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Order()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Адрес доставки заказа.

/// </summary>

public virtual string Address { get; protected set; }

/// <summary>

/// Время получения заказа.

/// </summary>

public virtual DateTime GetTime { get; protected set; }

/// <summary>

/// Время доставки заказа.

/// </summary>

public virtual DateTime DeliveryTime { get; protected set; }

/// <summary>

/// Покупатель.

/// </summary>

public virtual Customer Customer { get; set; }

/// <summary>

/// Записи блюд в заказе.

/// </summary>

public virtual ISet<ListItem> ListItems { get; set; } = new HashSet<ListItem>();

/// <summary>

/// Добавление записи блюда в заказ.

/// </summary>

/// <param name="listItem"> Запись блюда в заказе. </param>

public virtual void AddListItemToOrder(ListItem listItem)

{

this.ListItems.Add(listItem);

listItem.Order = this;

}

/// <inheritdoc/>

public virtual bool Equals(Order? other)

{

return Equals(this.Id, other?.Id);

}

/// <inheritdoc/>

public override int GetHashCode()

=> this.Id.GetHashCode();

}

}

Листинг файла Product.cs:

// <copyright file="Product.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain

{

/// <summary>

/// Блюдо.

/// </summary>

public class Product : IEquatable<Product>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Product"/>.

/// </summary>

/// <param name="menuName"> Название блюда. </param>

/// <param name="price"> Цена блюда. </param>

/// <exception cref="ArgumentNullException"></exception>

public Product(string menuName, decimal price)

{

if (string.IsNullOrWhiteSpace(menuName))

{

throw new ArgumentNullException(nameof(menuName));

}

if (price <= 0)

{

throw new ArgumentOutOfRangeException(nameof(price));

}

this.Id = Guid.NewGuid();

this.MenuName = menuName;

this.Price = price;

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Product"/>.

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Product()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Название блюда.

/// </summary>

public virtual string MenuName { get; protected set; }

/// <summary>

/// Цена блюда.

/// </summary>

public virtual decimal Price { get; protected set; }

/// <summary>

/// Запись блюда в заказе.

/// </summary>

public virtual ListItem ListItem { get; set; }

/// <inheritdoc/>

public override string ToString()

=> this.MenuName + ", " + this.Price;

/// <inheritdoc/>

public virtual bool Equals(Product? other)

{

return Equals(this.Id, other?.Id);

}

/// <inheritdoc/>

public override int GetHashCode()

=> this.Id.GetHashCode();

}

}

Листинг файла ListItem.cs:

// <copyright file="OrderProduct.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

using System.ComponentModel;

namespace Domain

{

/// <summary>

/// Запись блюда в заказе.

/// </summary>

public class ListItem : IEquatable<ListItem>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ListItem"/>.

/// </summary>

/// <param name="quantity"> Количество. </param>

/// <param name="order"> Заказ. </param>

public ListItem(int quantity, Product product)

{

this.Id = Guid.NewGuid();

this.Add(product, quantity);

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ListItem"/>.

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected ListItem()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Количество позиций одного блюда.

/// </summary>

public virtual int Quantity { get; protected set; }

/// <summary>

/// Заказ.

/// </summary>

public virtual Order Order { get; set; }

/// <summary>

/// Блюда.

/// </summary>

public virtual ISet<Product> Products { get; set; } = new HashSet<Product>();

/// <inheritdoc/>

public virtual bool Equals(ListItem? other)

{

return Equals(this.Id, other?.Id);

}

/// <inheritdoc/>

public override int GetHashCode()

=> this.Id.GetHashCode();

/// <summary>

///

/// </summary>

/// <param name="product"></param>

/// <param name="count"></param>

/// <exception cref="ArgumentNullException"></exception>

/// <exception cref="ArgumentOutOfRangeException"></exception>

public virtual void Add(Product product, int count)

{

if (product == null)

{

throw new ArgumentNullException(nameof(product));

}

if (count <= 0)

{

throw new ArgumentOutOfRangeException(nameof(count));

}

this.Quantity = count;

this.Products.Add(product);

product.ListItem = this;

}

}

}

## Тестирование классов из проекта Domain

Создадим проект Domain.Tests и дальнейшую работу будем проводить в нём.

Листинг файла CustomerTests.cs:

// <copyright file="CustomerTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain.Tests

{

using Domain;

/// <summary>

/// Модульные тесты для класса Customer.

/// </summary>

[TestFixture]

public class CustomerTests

{

/// <summary>

/// Тест на конструктор с правильным ФИ.

/// </summary>

/// <param name="middleName"> Отчество. </param>

[Test]

public void Ctor\_Valid\_DoesnotThrowException()

{

// Arrange

// Act

// Assert

Assert.DoesNotThrow(() => \_ = new Customer("Иванов", "Иван", "88888888888"));

}

/// <summary>

/// Тест на конструктор с неправильной фамилией.

/// </summary>

/// <param name="lastName"> Фамилия.</param>

[Test]

[TestCase(null)]

[TestCase("")]

public void Ctor\_WrongLastName\_ThrowException(string? lastName)

{

// Arrange

// Act

// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Customer(lastName, "Иван", "88888888888"));

}

/// <summary>

/// Тест на конструктор с неправильным именем.

/// </summary>

/// <param name="firstName"> Имя.</param>

[Test]

[TestCase(null)]

[TestCase("")]

public void Ctor\_WrongFirstName\_ThrowException(string? firstName)

{

// Arrange

// Act

// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Customer("Иванов", firstName, "88888888888"));

}

/// <summary>

/// Тест на конструктор с пустым номером телефона.

/// </summary>

/// <param name="phoneNumber"> Номер телефона.</param>

[Test]

[TestCase(null)]

[TestCase("")]

public void Ctor\_WrongPhoneNumber\_ThrowException(string? phoneNumber)

{

// Arrange

// Act

// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Customer("Иванов", "Иван", phoneNumber));

}

/// <summary>

/// Проверка на эквивиалентность двух одинаковых экземпляров.

/// </summary>

[Test]

public void AreEquals\_Success()

{

// Arrange

var customer = new Customer("Иванов", "Иван", "88888888888");

var customer2 = customer;

// Act & Assert

Assert.That(customer, Is.EqualTo(customer2));

}

/// <summary>

/// Проверка на преобразование в строку. Имеется отчество.

/// </summary>

[Test]

public void ToString\_ValidData\_Success()

{

// Arrange

var customer = new Customer("Иванов", "Иван", "88888888888");

var expected = "Иванов Иван";

// Act & Assert

Assert.That(customer.ToString(), Is.EqualTo(expected));

}

}

}

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

***Рисунок 2****– CustomerTests выполнены успешно*

Листинг файла OrderTests.cs:

// <copyright file="OrderTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain.Tests

{

using Domain;

/// <summary>

/// Модульные тесты для класса Order.

/// </summary>

[TestFixture]

public class OrderTests

{

/// <summary>

/// Тест на конструктор с правильным адресом.

/// </summary>

[Test]

public void Ctor\_Valid\_DoesnotThrowException()

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

// Act

// Assert

Assert.DoesNotThrow(() => \_ = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00)));

}

/// <summary>

/// Тест на конструктор с неправильным адресом.

/// </summary>

/// <param name="address"> Адрес.</param>

[Test]

[TestCase(null)]

[TestCase("")]

public void Ctor\_WrongAddress\_ThrowException(string? address)

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

// Act

// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Order(address, customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00)));

}

/// <summary>

/// Тест на конструктор с неправильным покупателем.

/// </summary>

/// <param name="customer"> Покупатель.</param>

[Test]

[TestCase(null)]

public void Ctor\_WrongCustomer\_ThrowException(Customer? customer)

{

// Arrange

// Act

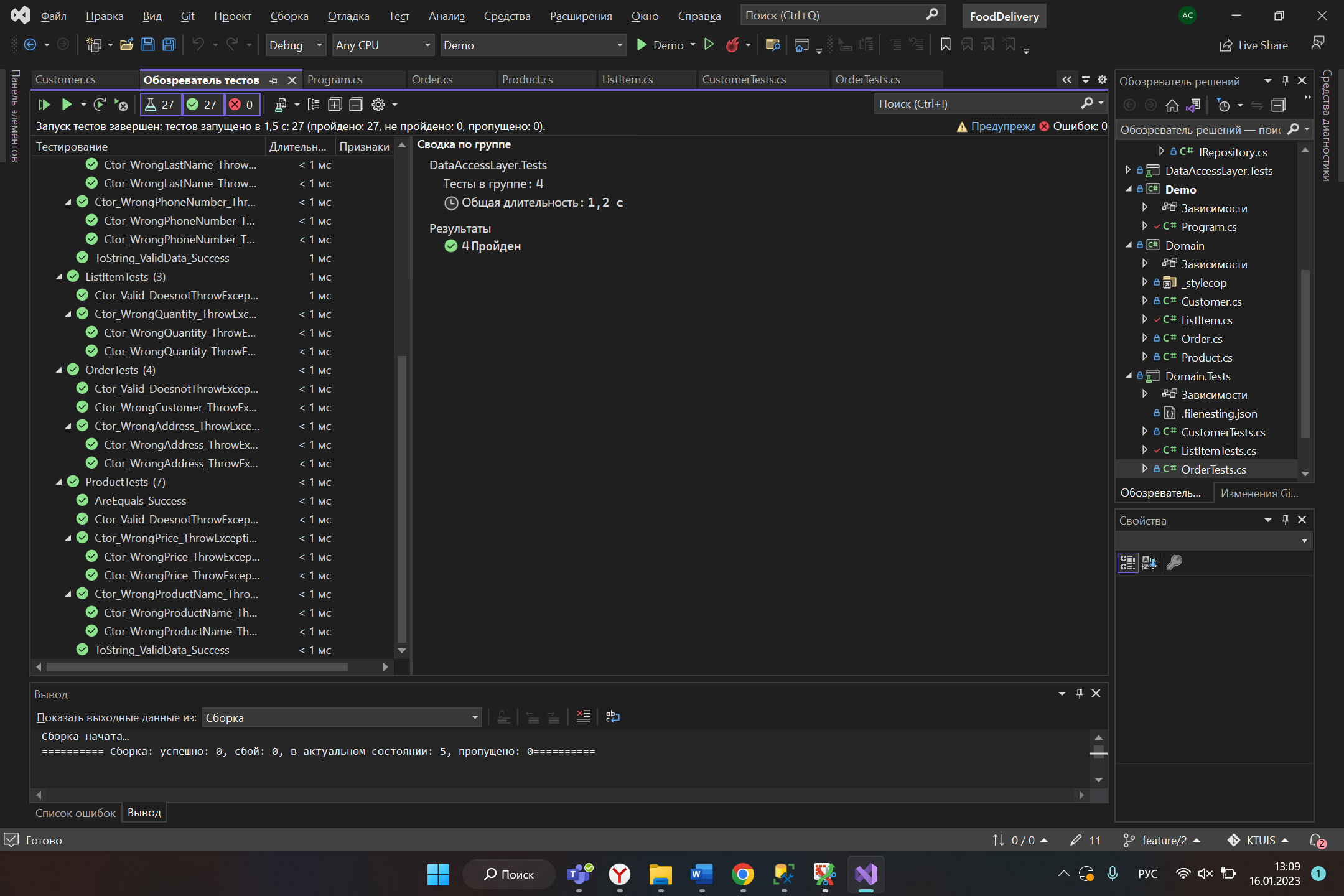
// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00)));

}

}

}



***Рисунок 3****– OrderTests выполнены успешно*

Листинг файла ProductTests.cs:

// <copyright file="ProductTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain.Tests

{

using Domain;

/// <summary>

/// Модульные тесты для класса Product.

/// </summary>

[TestFixture]

public class ProductTests

{

/// <summary>

/// Тест на конструктор с правильным названием блюда и ценой.

/// </summary>

[Test]

public void Ctor\_Valid\_DoesnotThrowException()

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

// Act

// Assert

Assert.DoesNotThrow(() => \_ = new Product("Пицца", 499));

}

/// <summary>

/// Тест на конструктор с неправильным названием блюда.

/// </summary>

/// <param name="menuName"> Название блюда.</param>

[Test]

[TestCase(null)]

[TestCase("")]

public void Ctor\_WrongProductName\_ThrowException(string? menuName)

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

// Act

// Assert

Assert.Throws<ArgumentNullException>(() => \_ = new Product(menuName, 499));

}

/// <summary>

/// Тест на конструктор с неправильной ценой.

/// </summary>

/// <param name="price"> Цена блюда.</param>

[Test]

[TestCase(-30)]

[TestCase(0)]

public void Ctor\_WrongPrice\_ThrowException(decimal price)

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

// Act

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Product("Пицца", price));

}

/// <summary>

/// Проверка на эквивиалентность двух одинаковых экземпляров.

/// </summary>

[Test]

public void AreEquals\_Success()

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

var product = new Product("Пицца", 499);

var product2 = product;

// Act & Assert

Assert.That(product, Is.EqualTo(product2));

}

/// <summary>

/// Проверка на преобразование в строку.

/// </summary>

[Test]

public void ToString\_ValidData\_Success()

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

var product = new Product("Пицца", 499);

var expected = "Пицца, 499";

// Act & Assert

Assert.That(product.ToString(), Is.EqualTo(expected));

}

}

}

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

***Рисунок 4****– ProductTests выполнены успешно*

Листинг файла ListItemTests.cs:

// <copyright file="ListItemTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Domain.Tests

{

using Domain;

/// <summary>

/// Модульные тесты для класса ListItem.

/// </summary>

[TestFixture]

public class ListItemTests

{

/// <summary>

/// Тест на конструктор с правильным количеством блюд.

/// </summary>

[Test]

public void Ctor\_Valid\_DoesnotThrowException()

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

Product product = new Product("Пицца", 499);

// Act

// Assert

Assert.DoesNotThrow(() => \_ = new ListItem(3, product));

}

/// <summary>

/// Тест на конструктор с неправильным количеством блюд.

/// </summary>

/// <param name="quantity"> Количество.</param>

[Test]

[TestCase(-30)]

[TestCase(0)]

public void Ctor\_WrongQuantity\_ThrowException(int quantity)

{

// Arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

Product product = new Product("Пицца", 499);

// Act

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new ListItem(quantity, product));

}

}

}

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

***Рисунок 5****– ListItemTests выполнены успешно*

Все тесты успешно закончили свою работу. Дальше разработаем классы для взаимодействия со слоем данных.

## Создание слоя для взаимодействия с данными

Создадим проект DataAccessLayer и дальнейшую работу будем проводить в нём.

Листинг файла Settings.cs, хранящего данные для подключения к БД:

// <copyright file="Settings.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer

{

/// <summary>

/// Настройки для СУБД.

/// </summary>

public sealed class Settings

{

private string dabaseServerName;

private string databaseName;

/// <summary>

/// Добавление сервера.

/// </summary>

/// <param name="serverName">Имя сервера.</param>

public void AddDabaseServer(string serverName)

{

this.dabaseServerName = serverName;

}

/// <summary>

/// Получение имени сервера.

/// </summary>

/// <returns>Имя сервера.</returns>

public string GetDatabaseServer()

{

return this.dabaseServerName;

}

/// <summary>

/// Добавление БД.

/// </summary>

/// <param name="databaseName">Название БД.</param>

public void AddDatabase(string databaseName)

{

this.databaseName = databaseName;

}

/// <summary>

/// Получение названия БД.

/// </summary>

/// <returns>Название БД.</returns>

public string GetDatabaseName()

{

return this.databaseName;

}

}

}

Листинг файла Configurator.cs, содержащий конфигуратор FluentNHibernate и фабрику сессий:

// <copyright file="Configurator.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Tool.hbm2ddl;

/// <summary>

/// FluentNHibernate конфигуратор.

/// </summary>

public static class Configurator

{

private static FluentConfiguration? fluentConfiguration;

/// <summary>

/// Создание фабрики сессий.

/// </summary>

/// <param name="settings"> Параметры СУБД. </param>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Фабрику сессий. </returns>

public static ISessionFactory GetSessionFactory(

Settings settings,

Assembly? assembly = null,

bool showSql = false)

{

return GetConfiguration(

settings, assembly ?? Assembly.GetExecutingAssembly(), showSql)

.BuildSessionFactory();

}

/// <summary>

/// Создание конфигурации FluentNHibernate.

/// </summary>

/// <param name="settings"> Параметры СУБД. </param>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Отображение генерируемого SQL-код. </param>

/// <returns> Конфигурация FluentNHibernate. </returns>

private static FluentConfiguration GetConfiguration(

Settings settings,

Assembly assembly,

bool showSql = false)

{

if (fluentConfiguration is null)

{

var databaseConfiguration =

MsSqlConfiguration.MsSql2012.ConnectionString(

x => x

.Server(settings.GetDatabaseServer())

.Database(settings.GetDatabaseName())

.TrustedConnection());

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

fluentConfiguration = Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly))

.ExposeConfiguration(BuildSchema);

}

return fluentConfiguration;

}

/// <summary>

/// Создание схемы по конфигурации.

/// </summary>

/// <param name="configuration"> Конфигурация FluentNHibernate. </param>

private static void BuildSchema(NHibernate.Cfg.Configuration configuration)

{

new SchemaExport(configuration).Execute(true, true, false);

}

}

}

Листинг файла ConfiguratorTests.cs, содержащий конфигуратор сессий для тестов и фабрику сессий:

// <copyright file="ConfiguratorTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Library.DataAccess

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

/// <summary>

/// Класс для настройки соединения для тестов.

/// </summary>

public class ConfiguratorTests

{

private static Configuration configuration;

/// <summary>

/// Создание фабрики сессий.

/// </summary>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Фабрику сессий. </returns>

public static ISessionFactory GetSessionFactory(Assembly assembly = null, bool showSql = false)

{

var databaseConfiguration = SQLiteConfiguration.Standard.InMemory();

if (showSql)

{

databaseConfiguration = databaseConfiguration.ShowSql().FormatSql();

}

return Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings

.AddFromAssembly(assembly ?? Assembly.GetExecutingAssembly()))

.ExposeConfiguration(c => configuration = c)

.BuildSessionFactory();

}

/// <summary>

/// Генерируется сессия для модульных тестов.

/// </summary>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Сессия подключения к тестовой БД. </returns>

public static ISession BuildSessionForTest(bool showSql = true)

{

var session = GetSessionFactory(showSql: showSql).OpenSession();

new SchemaExport(configuration)

.Execute(true, true, false, session.Connection, null);

return session;

}

}

}

Листинг файла CustomerMap.cs, содержащий маппинги класса Customer:

// <copyright file="CustomerMap.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using Domain;

using FluentNHibernate.Mapping;

/// <summary>

/// Класс отображения Покупателя.

/// </summary>

internal class CustomerMap : ClassMap<Customer>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="CustomerMap"/>.

/// </summary>

public CustomerMap()

{

this.Table("Customers");

this.Id(x => x.Id);

this.Map(x => x.LastName)

.Length(50)

.Not

.Nullable();

this.Map(x => x.FirstName)

.Length(50)

.Not

.Nullable();

this.Map(x => x.PhoneNumber)

.Length(11)

.Not

.Nullable();

this.HasMany(x => x.Orders)

.Not.Inverse();

}

}

}

Листинг файла OrderMap.cs, содержащий маппинги класса Order:

// <copyright file="OrderMap.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using Domain;

using FluentNHibernate.Mapping;

/// <summary>

/// Класс отображения Заказа.

/// </summary>

internal class OrderMap : ClassMap<Order>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="OrderMap"/>.

/// </summary>

public OrderMap()

{

this.Table("Orders");

this.Id(x => x.Id);

this.Map(x => x.Address)

.Length(50)

.Not

.Nullable();

this.References(x => x.Customer);

this.Map(x => x.GetTime)

.Not

.Nullable();

this.Map(x => x.DeliveryTime)

.Not

.Nullable();

this.HasMany(x => x.ListItems)

.Not

.Inverse();

}

}

}

Листинг файла ProductMap.cs, содержащий маппинги класса Product:

// <copyright file="ProductMap.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using Domain;

using FluentNHibernate.Mapping;

/// <summary>

/// Класс отображения Блюда.

/// </summary>

internal class ProductMap : ClassMap<Product>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ProductMap"/>.

/// </summary>

public ProductMap()

{

this.Table("Products");

this.Id(x => x.Id);

this.Map(x => x.MenuName)

.Length(50)

.Not

.Nullable();

this.Map(x => x.Price)

.Length(10)

.Not

.Nullable();

this.References(x => x.ListItem);

}

}

}

Листинг файла ListItemMap.cs, содержащий маппинги класса ListItem:

// <copyright file="ListItemMap.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using Domain;

using FluentNHibernate.Mapping;

/// <summary>

/// Класс отображения Записи блюда в заказе.

/// </summary>

internal class ListItemMap : ClassMap<ListItem>

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ListItemMap"/>.

/// </summary>

public ListItemMap()

{

this.Table("ListItems");

this.Id(x => x.Id);

this.Map(x => x.Quantity)

.Not

.Nullable();

this.References(x => x.Order);

this.HasMany(x => x.Products);

}

}

}

## Тестирование классов из проекта DataAccessLayer

Создадим проект DataAccessLayer.Tests и дальнейшую работу будем проводить в нём.

Листинг файла MapTests.cs, содержащий класс от которого будут наследовать все остальные классы:

// <copyright file="ListItemMapTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.Tests

{

using Library.DataAccess;

using NHibernate;

using NUnit.Framework;

public class MapTests

{

protected ISession Session { get; private set; }

[SetUp]

public void Setup()

{

this.Session = ConfiguratorTests.BuildSessionForTest();

}

[TearDown]

public void TearDown()

{

this.Session?.Dispose();

}

}

}

Листинг файла CustomerMapTests.cs, который содержит тестирование маппингов CustomerMap:

// <copyright file="CustomerMapTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.Tests

{

using FluentNHibernate.Testing;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты маппингов для класса Customer.

/// </summary>

[TestFixture]

internal class CustomerMapTests : MapTests

{

/// <summary>

/// Тест правильной работы маппинга.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// Arrange

var customer = new Customer("Иванов", "Иван", "88888888888");

// Act & Assert

new PersistenceSpecification<Customer>(this.Session)

.VerifyTheMappings(customer);

}

}

}

Листинг файла OrderMapTests.cs, который содержит тестирование маппингов OrderMap:

// <copyright file="OrderMapTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.Tests

{

using FluentNHibernate.Testing;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты маппингов для класса Order.

/// </summary>

[TestFixture]

internal class OrderMapTests : MapTests

{

/// <summary>

/// Тест правильной работы маппинга.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

this.Session.Save(customer);

var order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

// act & assert

new PersistenceSpecification<Order>(this.Session)

.VerifyTheMappings(order);

}

}

}

Листинг файла ProductMapTests.cs, который содержит тестирование маппингов ProductMap:

// <copyright file="ProductMap.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.Tests

{

using FluentNHibernate.Testing;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты маппингов для класса Album.

/// </summary>

[TestFixture]

internal class ProductMapTests : MapTests

{

/// <summary>

/// Тест правильной работы маппинга.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

this.Session.Save(customer);

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

this.Session.Save(order);

Product product = new Product("Пицца", 499);

ListItem listItem = new ListItem(3, product);

this.Session.Save(listItem);

// act & assert

new PersistenceSpecification<Product>(this.Session)

.VerifyTheMappings(product);

}

}

}

Листинг файла ListItemMapTests.cs, который содержит тестирование маппингов ListItemMap:

// <copyright file="ListItemMapTests.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace DataAccessLayer.Tests

{

using FluentNHibernate.Testing;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты маппингов для класса ListItem.

/// </summary>

[TestFixture]

internal class ListItemMapTests : MapTests

{

/// <summary>

/// Тест правильной работы маппинга.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// arrange

Customer customer = new Customer("Иванов", "Иван", "88888888888");

this.Session.Save(customer);

Order order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

this.Session.Save(order);

Product product = new Product("Пицца", 499);

var listItem = new ListItem(3, product);

// act & assert

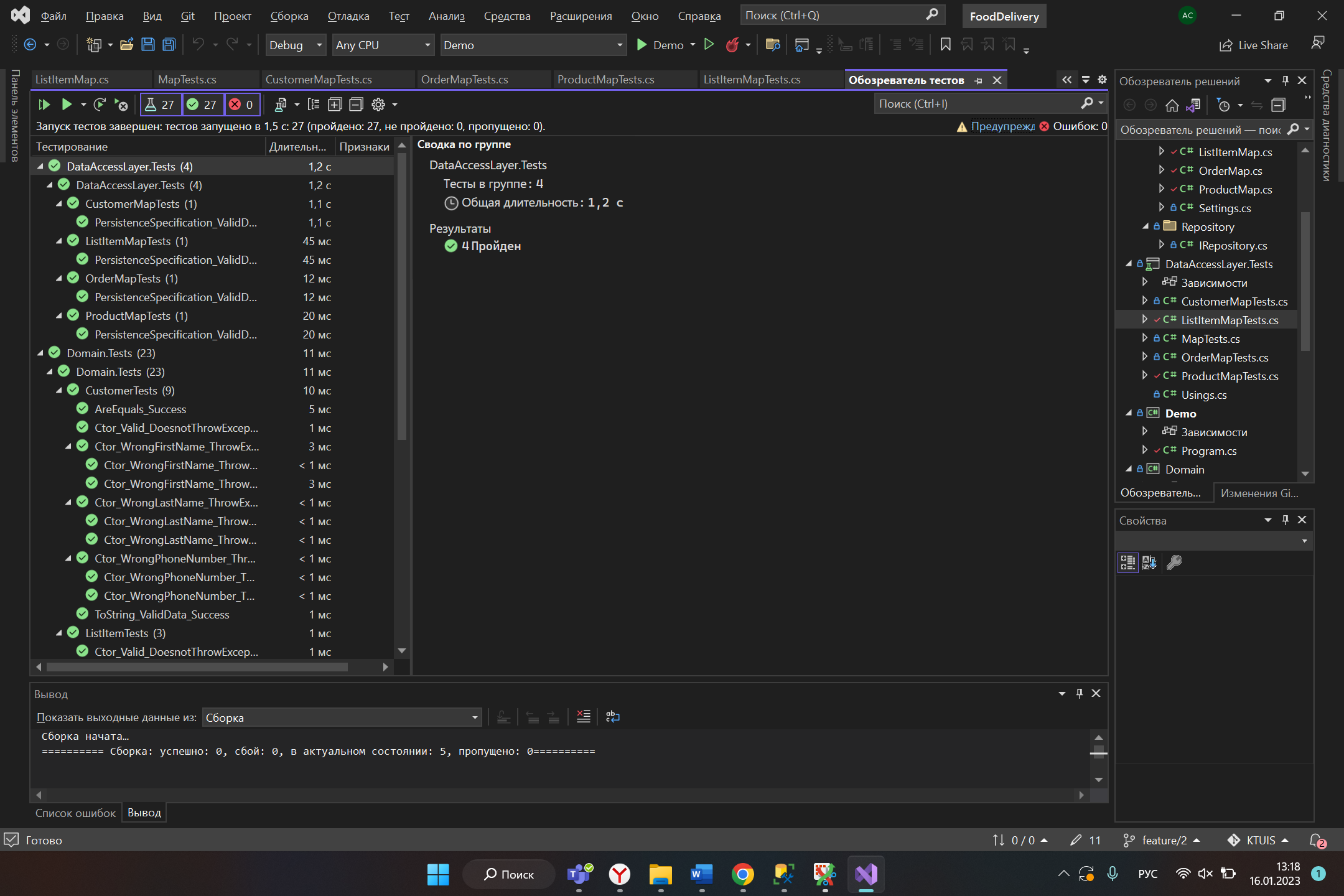
new PersistenceSpecification<Order>(this.Session)

.VerifyTheMappings(order);

}

}

}



***Рисунок 6****– Результат запуска тестов*

## Проверка работоспособности

Создадим проект Demo и дальнейшую работу будем проводить в нём.

Листинг файла Program.cs, в котором происходит взаимодействие с БД.

// <copyright file="Program.cs" company="ActiVia">

// Copyright (c) ActiVia 2022.

// </copyright>

namespace Demo

{

using System;

using DataAccessLayer;

using DataAccessLayer.ORM;

using Domain;

/// <summary>

/// Исполняемый файл.

/// </summary>

internal class Program

{

/// <summary>

/// Точка входа в программу.

/// </summary>

private static void Main()

{

var customer = new Customer("Иванов", "Иван", "88888888888");

var order = new Order("ул. Космонавтов, д. 11", customer, new DateTime(2022, 12, 20, 10, 00, 00), new DateTime(2022, 12, 20, 10, 50, 00));

var product = new Product("Пицца", 499);

var listItem = new ListItem(2, product);

customer.Orders.Add(order);

order.ListItems.Add(listItem);

listItem.Add(product, 2);

Console.WriteLine(customer);

Console.WriteLine(order);

Console.WriteLine(listItem);

Console.WriteLine(product);

var settings = new Settings();

settings.AddDabaseServer("STACEY");

settings.AddDatabase("FoodDelivery");

using var sessionFactory = Configurator.GetSessionFactory(settings, showSql: true);

using (var session = sessionFactory.OpenSession())

{

// Скорее всего порядок такой:

// т.к. Products --> ListItems --> Orders --> Customers

session.Save(customer); // 1

session.Save(order); // 2

session.Save(listItem); // 3

session.Save(product); // 4

session.Flush();

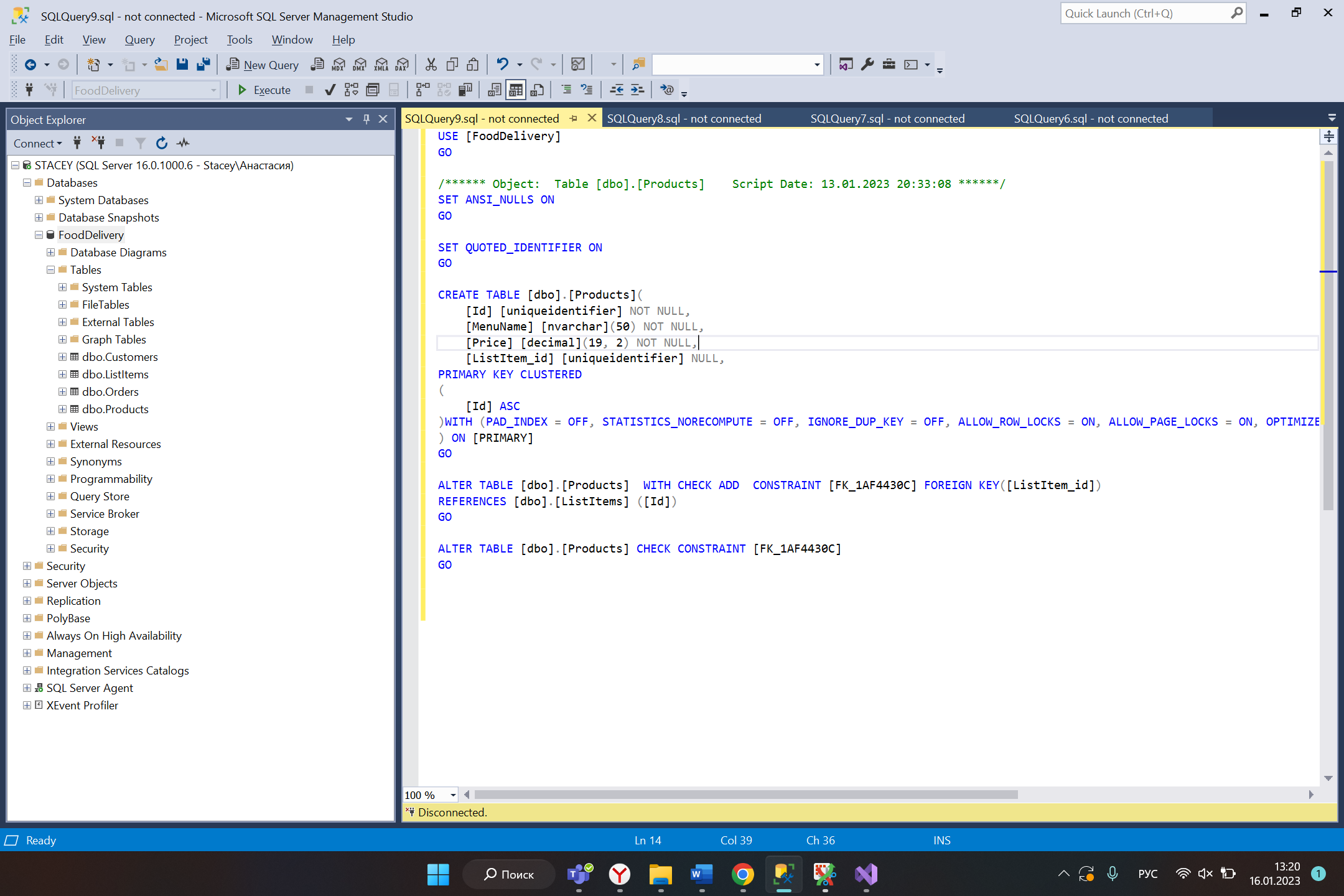
}

}

}

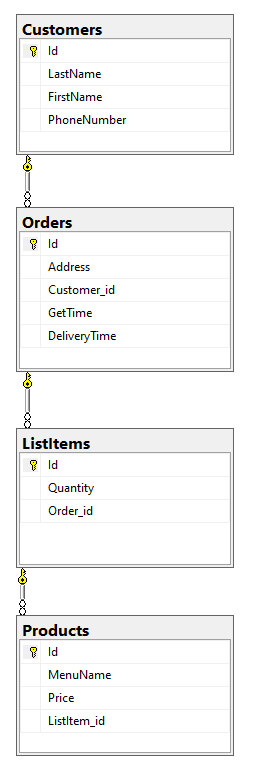
}

После запуска программы, в указанной БД были создана таблицы:



***Рисунок 7****– В БД FoodDelivery созданы таблицы*

Получили следующую диаграмму:



***Рисунок 8****– Диаграмма БД*

## Вывод

Была построена ORM, закреплены навыки программирования.