Министерство образования Саратовской области

Государственное автономное профессиональное образовательное учреждение Саратовской области

«Новоузенский агротехнологический техникум»

**Специальность: 09.02.07 Информационные системы и программирование**

**Курсовая работа**

**по МДК 01.01. «Разработка программных модулей»**

**на тему:**

**“Создание простого приложения для обработки данных”**

Выполнил: студент(ка) группы П-21:

Кушуков Тимур Ержанович

02.06.2023г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Руководитель: преподаватель

Бекбулатов Р.Ж.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество) (оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

г. Новоузенск 2023г.

Содержание

ГЛАВА 1.УСТАНОВКА И ПОДКЛЮЧЕНИЕ NORTHWIND

1.1.Настройка проекта

1.2.Создание модели

ГЛАВА 2. ПРИВЯЗКА ДАННЫХ МОДЕЛИ К СТРАНИЦЕ XAML.

2.1. Подробная привязка данных модели к станице

ГЛАВА 3. НАСТРОЙКА МАКЕТА СТРАНИЦЫ И ДОБАВЛЕНИЕ СЕТКИ

.3.1. Настройка макета страницы и добавление сетки для новых клиентов и заказов

ГЛАВА 4. ДОБАВЛЕНИЕ КНОПОК ДЛЯ НАВИГАЦИИ , ДОБАВЛЕНИЕ , ОБНОВЛЕНИЯ И УДАЛЕНИЯ

4.1. Добавления команд, привязок и кнопок в XAML

ГЛАВА 5. ДОБАВЛЕНИЕ ОБРАБОТЧИКОВ ДАННЫХ В КЛАСС MAINWINDOW.

5.1 Добавление обработчиков данных в класс MainWindow.xaml.cs

ГЛАВА 6. ВЫПОЛНЕНИЕ ПРИЛОЖЕНИЯ

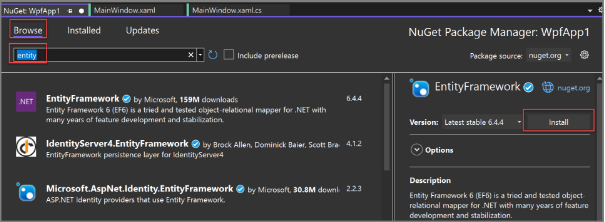
ЗАКЛЮЧЕНИЕ.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.

ГЛАВА 1.УСТАНОВКА И ПОДКЛЮЧЕНИЕ NORTHWIND

1.1.Настройка проекта

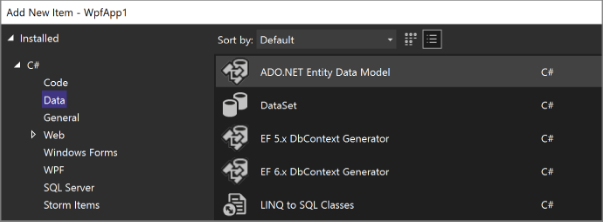
1. В Visual Studio создайте проект **приложения WPF на C# (платформа .NET Framework).**
2. Добавьте пакет NuGet для Entity Framework 6. В **Обозреватель решений** выберите узел проекта. В меню main выберите **Проект**>**Управление пакетами NuGet**.
3. В **диспетчере пакетов NuGet** щелкните ссылку **Обзор** . Entity Framework, вероятно, является верхним пакетом в списке. Нажмите кнопку **Установить** в области справа и следуйте инструкциям. В окне Выходные данные будет показано, когда установка будет завершена.



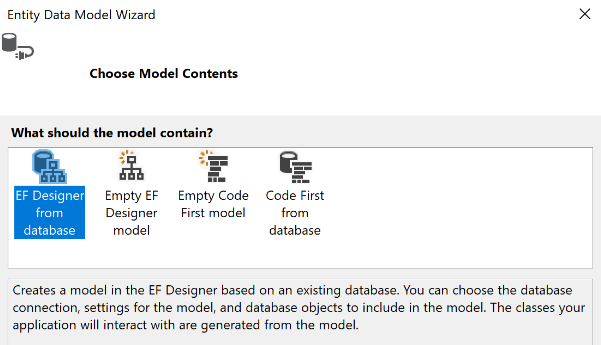
4.Теперь можно использовать Visual Studio для создания модели на основе базы данных Northwind.

1.2.Создание модели

1.Щелкните правой кнопкой мыши узел проекта в **обозревателе решений** и последовательно выберите **Добавить**>**Новый элемент**. В левой области в узле C# выберите **Данные** , а в средней области выберите **ADO.NET Entity Data Model**.



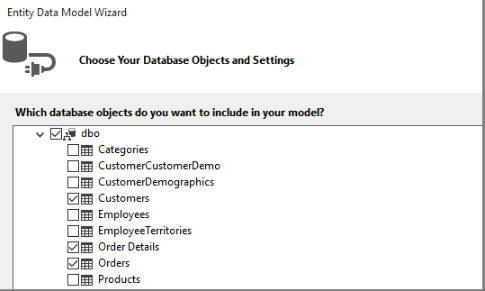
2. Вызовите модель Northwind\_model и нажмите кнопку **Добавить**. Откроется **мастер модели entity data.** Выберите **EF Designer из базы данных** и нажмите кнопку **Далее**.



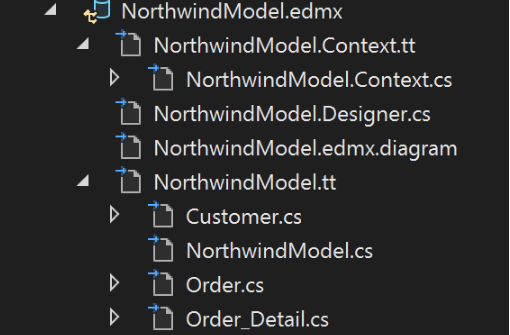
3.На следующем экране выберите подключение LocalDB Northwind (например, (localdb)\MSSQLLocalDB, укажите базу данных Northwind и нажмите кнопку **Далее**.

Если подключение не отображается, выберите **Новое подключение**, а затем в диалоговом окне **Выбор источника данных** выберите **Microsoft SQL Server**, **нажмите кнопку Продолжить** и в диалоговом окне **Свойства подключения** введите (localdb)\MSSQLLocalDB и в разделе **Выберите или введите имя базы данных**, выберите **Northwind** и нажмите **кнопку ОК**.

4.На следующей странице мастера выберите таблицы, хранимые процедуры и другие объекты базы данных для включения в модель Entity Framework. Разверните узел dbo в представлении в виде дерева и выберите **Клиенты**, **Заказы** и **Сведения о заказе**. Оставьте флажки по умолчанию и нажмите кнопку **Готово**.



5.Мастер создает классы C#, представляющие модель Entity Framework. Классы являются обычными старыми классами C#, и они являются тем, что мы привязали к пользовательскому интерфейсу WPF. EDMX-файл описывает связи и другие метаданные, которые связывают классы с объектами в базе данных. TT-файлы — это шаблоны T4, которые создают код, который работает с моделью и сохраняет изменения в базе данных. Все эти файлы можно просмотреть в **Обозреватель решений** в узле Northwind\_model:



6.TT-файлы общего назначения, и вам нужно настроить один из них для работы с привязкой данных WPF, для которой требуется ObservableCollections. В **Обозреватель решений** разверните узел Northwind\_model, пока не найдете Northwind\_model.tt. (Убедитесь, что вы не находитесь в файле .Context.tt , который находится непосредственно под EDMX-файлом .)

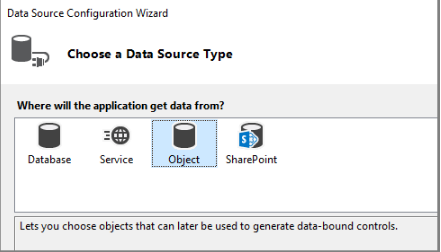
* + Замените два вхождения на [ICollection](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.icollection)[ObservableCollection<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.objectmodel.observablecollection-1).
  + Замените первое
  + вхождение [HashSet<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.hashset-1)[ObservableCollection<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.objectmodel.observablecollection-1) на около строки 51. Не заменяйте второе вхождение HashSet.
  + Замените единственное вхождение [System.Collections.Generic](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic) (около строки 431) на [System.Collections.ObjectModel](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.objectmodel).

1. Нажмите клавишу **F5** или **CTRL**+**F5** , чтобы выполнить сборку и запуск проекта. При первом запуске приложения классы модели видны мастеру источников данных.

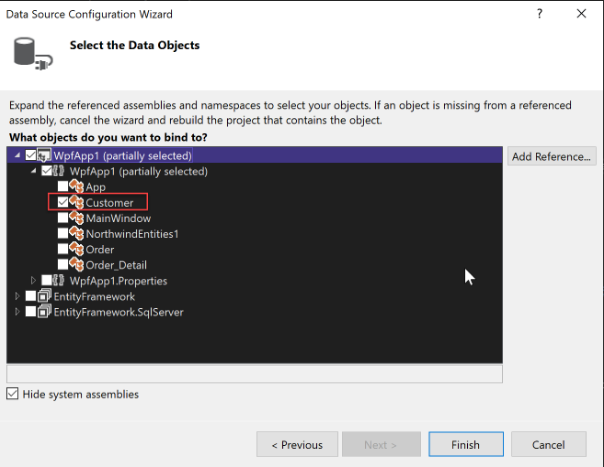
ГЛАВА 2. ПРИВЯЗКА ДАННЫХ МОДЕЛИ К СТРАНИЦЕ XAML.

2.1. Подробная привязка данных модели к станице

1. В меню main выберите **Проект**>**Добавить новый источник данных**, чтобы открыть **мастер настройки источника данных**. Выберите **Объект,** так как выполняется привязка к классам модели, а не к базе данных:



2. Разверните узел для проекта и выберите **Клиент**. (Источники для заказов автоматически создаются из свойства навигации Orders в Customer.)



3.Нажмите кнопку **Готово**.

4. В представлении кода перейдите к файлу MainWindow.xaml . Для целей этого примера мы сохраняем простой код XAML. Измените заголовок MainWindow на более описательное и увеличьте его высоту и ширину до 600 x 800. Вы всегда можете изменить его позже. Теперь добавьте эти три определения строк в сетку main, одну строку для кнопок навигации, одну для сведений о клиенте и другую для сетки, в которую отображаются его заказы:

**<Grid.RowDefinitions>**

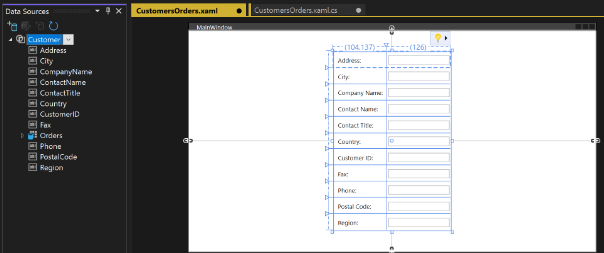
**<RowDefinition Height="auto"/>**

**<RowDefinition Height="auto"/>**

**<RowDefinition Height="\*"/>**

**</Grid.RowDefinitions>**

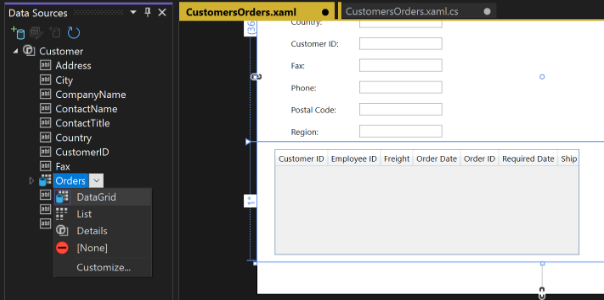
5.Теперь откройте файл MainWindow.xaml , чтобы просмотреть его в конструкторе. В результате окно **Источники данных** отображается в качестве параметра в поле окна Visual Studio рядом **с панелью элементов**. Щелкните вкладку, чтобы открыть окно, или нажмите **клавиши SHIFT**+**ALT**+**D** или выберите **Просмотреть**> другие**источники данныхWindows**>. Мы будем отображать каждое свойство в классе Customers в отдельном текстовом поле. Сначала щелкните стрелку в поле со списком **Клиенты** и выберите **Сведения**. Затем перетащите узел в среднюю часть области конструктора, чтобы конструктор знал, что вы хотите, чтобы он был в средней строке. В противном случае можно указать строку вручную позже в XAML (Grid.Row="1"). По умолчанию элементы управления размещаются по вертикали в элементе сетки, но на этом этапе их можно расположить как угодно в форме. Например, может быть целесообразно поместить текстовое поле **Имя** сверху над адресом. Пример приложения для этой статьи изменяет порядок полей и преобразует их в два столбца.

****

В представлении XAML теперь можно увидеть новый Grid элемент в строке 1 (средней строке) родительского элемента Grid. Родительский DataContext элемент Grid имеет атрибут , который ссылается на [CollectionViewSource](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.data.collectionviewsource) объект , добавленный в Windows.Resources элемент . Учитывая этот контекст данных, когда первое текстовое поле привязывается к **Address**, это имя сопоставляется со свойством Address в текущем Customer объекте CollectionViewSourceв .

**<Grid DataContext="{StaticResource customerViewSource}">**

6. Когда клиент отображается в верхней половине окна, вы хотите увидеть его заказы в нижней половине. Заказы отображаются в одном элементе управления представления сетки. Чтобы привязка данных master с подробными данными работала должным образом, важно привязаться к свойству Orders в классе Customers, а не к отдельному узлу Orders. Перетащите свойство Orders класса Customers в нижнюю часть формы, чтобы конструктор поместил его в строку 2:



7. Visual Studio создал весь код привязки, который соединяет элементы управления пользовательского интерфейса с событиями в модели. Все, что вам нужно сделать, чтобы просмотреть некоторые данные, — написать код для заполнения модели. Сначала перейдите к файлу MainWindow.xaml.cs и добавьте элемент данных в класс MainWindow для контекста данных. Этот объект, созданный автоматически, действует как элемент управления, который отслеживает изменения и события в модели. Вы также добавите элементы данных CollectionViewSource для клиентов и заказов, а также связанную логику инициализации конструктора в существующий конструктор MainWindow(). Верхняя часть класса должна выглядеть следующим образом:

**public partial class MainWindow : Window**

**{**

**NorthwindEntities context = new NorthwindEntities();**

**CollectionViewSource custViewSource;**

**CollectionViewSource ordViewSource;**

**public MainWindow()**

**{**

**InitializeComponent();**

**custViewSource = ((CollectionViewSource)(FindResource("customerViewSource")));**

**ordViewSource = ((CollectionViewSource)(FindResource("customerOrdersViewSource")));**

**DataContext = this;**

}

Если его еще нет, добавьте директиву using для System.Data.Entity, чтобы перенести Load метод расширения в область:

**using System.Data.Entity;**

Теперь прокрутите вниз и найдите Window\_Loaded обработчик событий. Обратите внимание, что Visual Studio добавил объект CollectionViewSource. Представляет объект NorthwindEntities, выбранный при создании модели. Вы уже добавили это, так что вам не нужно здесь. Давайте заменим код в Window\_Loaded , чтобы метод выглядел следующим образом:

**private void Window\_Loaded(object sender, RoutedEventArgs e)**

**{**

**// Load is an extension method on IQueryable,**

**// defined in the System.Data.Entity namespace.**

**// This method enumerates the results of the query,**

**// similar to ToList but without creating a list.**

**// When used with Linq to Entities, this method**

**// creates entity objects and adds them to the context.**

**context.Customers.Load();**

**// After the data is loaded, call the DbSet<T>.Local property**

**// to use the DbSet<T> as a binding source.**

**custViewSource.Source = context.Customers.Local;**

**}**

1. Нажмите клавишу **F5**. Вы должны увидеть сведения о первом клиенте, который был получен в CollectionViewSource. Вы также должны увидеть их заказы в сетке данных. Форматирование не очень удобно, поэтому давайте исправим это. Вы также можете создать способ просмотра других записей и выполнения основных операций CRUD.

ГЛАВА 3. НАСТРОЙКА МАКЕТА СТРАНИЦЫ И ДОБАВЛЕНИЕ СЕТКИ.

## 3.1. Настройка макета страницы и добавление сетки для новых клиентов и заказов

1.Расположение по умолчанию, созданное Visual Studio, не идеально подходит для вашего приложения, поэтому мы предоставим окончательный код XAML для копирования в код. Вам также потребуется несколько "форм" (которые фактически являются сетками), чтобы позволить пользователю добавить нового клиента или заказ. Чтобы иметь возможность добавить нового клиента и заказ, вам потребуется отдельный набор текстовых полей, которые не привязаны к CollectionViewSourceданным . Вы будете контролировать, какую сетку пользователь видит в любой момент времени, задав свойство Visible в методах обработчика. Наконец, вы добавляете кнопку Удалить в каждую строку в сетке Заказы, чтобы пользователь мог удалить отдельный заказ.

Сначала добавьте следующие стили в Windows.Resources элемент MainWindow.xaml:

**<Style x:Key="Label" TargetType="{x:Type Label}" BasedOn="{x:Null}">**

**<Setter Property="HorizontalAlignment" Value="Left"/>**

**<Setter Property="VerticalAlignment" Value="Center"/>**

**<Setter Property="Margin" Value="3"/>**

**<Setter Property="Height" Value="23"/>**

**</Style>**

**<Style x:Key="CustTextBox" TargetType="{x:Type TextBox}" BasedOn="{x:Null}">**

**<Setter Property="HorizontalAlignment" Value="Right"/>**

**<Setter Property="VerticalAlignment" Value="Center"/>**

**<Setter Property="Margin" Value="3"/>**

**<Setter Property="Height" Value="26"/>**

**<Setter Property="Width" Value="120"/>**

**</Style>**

Затем замените всю внешнюю сетку следующей разметкой:



ГЛАВА 4. ДОБАВЛЕНИЕ КНОПОК ДЛЯ НАВИГАЦИИ , ДОБАВЛЕНИЕ , ОБНОВЛЕНИЯ И УДАЛЕНИЯ

4.1. Добавления команд, привязок и кнопок в XAML

1. Сначала добавьте команды в файл MainWindow.xaml внутри Windows.Resources элемента :

**<RoutedUICommand x:Key="FirstCommand" Text="First"/>**

**<RoutedUICommand x:Key="LastCommand" Text="Last"/>**

**<RoutedUICommand x:Key="NextCommand" Text="Next"/>**

**<RoutedUICommand x:Key="PreviousCommand" Text="Previous"/>**

**<RoutedUICommand x:Key="DeleteCustomerCommand" Text="Delete Customer"/>**

**<RoutedUICommand x:Key="DeleteOrderCommand" Text="Delete Order"/>**

**<RoutedUICommand x:Key="UpdateCommand" Text="Update"/>**

**<RoutedUICommand x:Key="AddCommand" Text="Add"/>**

**<RoutedUICommand x:Key="CancelCommand" Text="Cancel"/>**

2.CommandBinding сопоставляет RoutedUICommand событие с методом в коде программной части. Добавьте этот CommandBindings элемент после закрывающего Windows.Resources тега:

**<Window.CommandBindings>**

**<CommandBinding Command="{StaticResource FirstCommand}" Executed="FirstCommandHandler"/>**

**<CommandBinding Command="{StaticResource LastCommand}" Executed="LastCommandHandler"/>**

**<CommandBinding Command="{StaticResource NextCommand}" Executed="NextCommandHandler"/>**

**<CommandBinding Command="{StaticResource PreviousCommand}" Executed="PreviousCommandHandler"/>**

**<CommandBinding Command="{StaticResource DeleteCustomerCommand}" Executed="DeleteCustomerCommandHandler"/>**

**<CommandBinding Command="{StaticResource DeleteOrderCommand}" Executed="DeleteOrderCommandHandler"/>**

**<CommandBinding Command="{StaticResource UpdateCommand}" Executed="UpdateCommandHandler"/>**

**<CommandBinding Command="{StaticResource AddCommand}" Executed="AddCommandHandler"/>**

**<CommandBinding Command="{StaticResource CancelCommand}" Executed="CancelCommandHandler"/>**

**</Window.CommandBindings>**

3.Теперь добавьте с StackPanel помощью кнопок навигации, добавить, удалить и обновить. Сначала добавьте этот стиль в Windows.Resources:

**<Style x:Key="NavButton" TargetType="{x:Type Button}" BasedOn="{x:Null}">**

**<Setter Property="FontSize" Value="24"/>**

**<Setter Property="FontFamily" Value="Segoe UI Symbol"/>**

**<Setter Property="Margin" Value="2,2,2,0"/>**

**<Setter Property="Width" Value="40"/>**

**<Setter Property="Height" Value="auto"/>**

**</Style>**

Во-вторых, вставьте этот код сразу после RowDefinitions для внешнего Grid элемента в верхней части страницы XAML:

**<StackPanel Orientation="Horizontal" Margin="2,2,2,0" Height="36" VerticalAlignment="Top" Background="Gainsboro" DataContext="{StaticResource customerViewSource}" d:LayoutOverrides="LeftMargin, RightMargin, TopMargin, BottomMargin">**

**<Button Name="btnFirst" Content="|◄" Command="{StaticResource FirstCommand}" Style="{StaticResource NavButton}"/>**

**<Button Name="btnPrev" Content="◄" Command="{StaticResource PreviousCommand}" Style="{StaticResource NavButton}"/>**

**<Button Name="btnNext" Content="►" Command="{StaticResource NextCommand}" Style="{StaticResource NavButton}"/>**

**<Button Name="btnLast" Content="►|" Command="{StaticResource LastCommand}" Style="{StaticResource NavButton}"/>**

**<Button Name="btnDelete" Content="Delete Customer" Command="{StaticResource DeleteCustomerCommand}" FontSize="11" Width="120" Style="{StaticResource NavButton}"/>**

**<Button Name="btnAdd" Content="New Customer" Command="{StaticResource AddCommand}" FontSize="11" Width="80" Style="{StaticResource NavButton}"/>**

**<Button Content="New Order" Name="btnNewOrder" FontSize="11" Width="80" Style="{StaticResource NavButton}" Click="NewOrder\_click"/>**

**<Button Name="btnUpdate" Content="Commit" Command="{StaticResource UpdateCommand}" FontSize="11" Width="80" Style="{StaticResource NavButton}"/>**

**<Button Content="Cancel" Name="btnCancel" Command="{StaticResource CancelCommand}" FontSize="11" Width="80" Style="{StaticResource NavButton}"/>**

**</StackPanel>**

ГЛАВА 5. ДОБАВЛЕНИЕ ОБРАБОТЧИКОВ ДАННЫХ В КЛАСС MAINWINDOW.

5.1 Добавление обработчиков данных в класс MainWindow.xaml.cs

Код программной части минимальный, за исключением методов add и delete. Навигация выполняется путем вызова методов для свойства View объекта CollectionViewSource. Показывает DeleteOrderCommandHandler , как выполнить каскадное удаление для заказа. Сначала необходимо удалить связанные с ним Order\_Details. Добавляет UpdateCommandHandler нового клиента или заказ в коллекцию или просто обновляет существующий клиент или заказ с учетом изменений, внесенных пользователем в текстовые поля.

Добавьте эти методы обработчика в класс MainWindow в файле MainWindow.xaml.cs. Если у коллекции CollectionViewSource для таблицы Customers другое имя, необходимо изменить имя в каждом из следующих методов:



ГЛАВА 6. ВЫПОЛНЕНИЕ ПРИЛОЖЕНИЯ

Чтобы начать отладку, нажмите клавишу **F5**. В сетке должны быть заполнены данные о клиентах и заказах, а кнопки навигации должны работать должным образом. Щелкните **Зафиксировать** , чтобы добавить нового клиента или заказ в модель после ввода данных. Нажмите кнопку **Отмена** , чтобы вернуться к новому клиенту или новой форме заказа без сохранения данных. Вы можете вносить изменения в существующие клиенты и заказы непосредственно в текстовых полях, и эти изменения автоматически записываются в модель.

СОСТАВ ТАБЛИЦ БД.

Рис.1 Customers



Рис.2 Orders

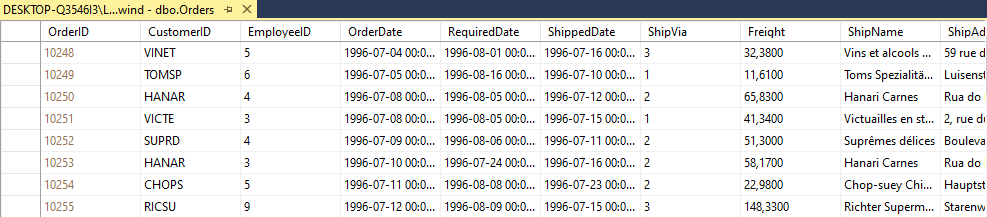
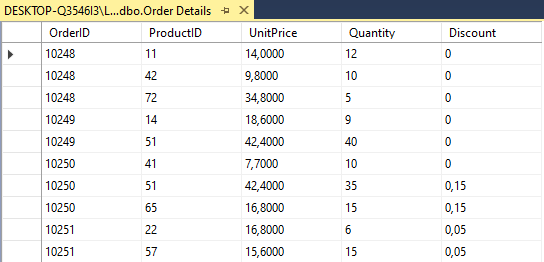
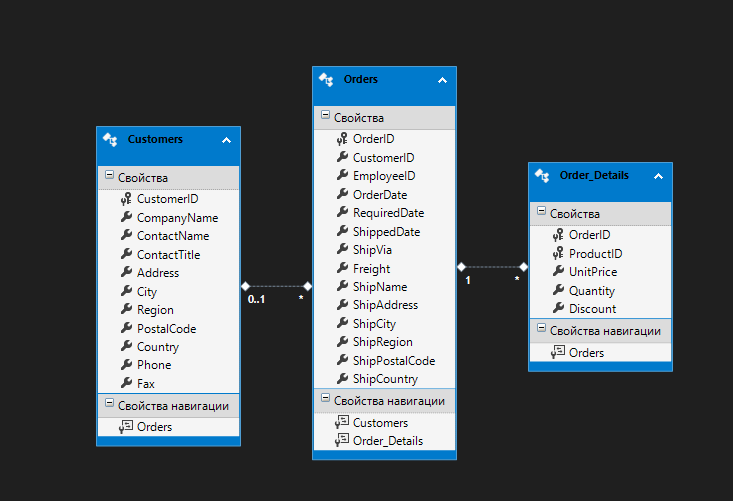


Рис.3 Order\_Details



ДИАГРАММЫ ТАБЛИЦ.



ЗАКЛЮЧЕНИЕ.

В рамках курсового проектирования было разработано «Программное средство для автоматизации работы кассира в кинотеатре».

Программное средство имеет ряд достоинств: простой и понятный интерфейс, небольшой объем памяти, занимаемый приложением на различных носителях информации, удобно для быстрой обработки информации.

В программе реализованы такие задачи, как: редактирование данных, фильтрация данных, формирование отчетов, продажа и бронирование билетов на сеансы.

Программа реализована в полном объеме и в соответствии с заданными требованиями. Полностью отлажена и протестирована. Поставленные задачи выполнены.

В программном средстве, благодаря среде разработки Microsoft Visual Studio 2010, был разработан удобный, понятный и простой в использовании интерфейс.

Программа реализована полностью в соответствии с поставленной задачей курсового проекта. Приложение было протестировано и отлажено.

Проект был реализован с помощью среды разработки Microsoft Visual Studio 2010, с использованием языка C# и системы управления базами данных Microsoft SQL Server 2008.

Программное средство готово к практическому использованию. Данное приложение может быть дополнено и модернизировано

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.

1 Багласова Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий».– Мн.: ТБП, 2003

2 Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов.- Мн.: ТБП, 2006

3 Кунец И.Е. Объектно – ориентированный анализ и проектирование. Работа в среде Rational Rose. Методическое пособие для учащихся ССУЗ по специальности 2-40 01 01 «Программное обеспечение информационных технологий».– Мн.: ТБП, 2006

4 МаккиА.И. VisualStudio 2010 для профессионалов. –СПб.: Питер, 2010

5 КарповР.Н. Microsoft Office 2007 – Мн.: Бином, 2007

6 Томров А.К. RationalRose для начинающих. – Мн.: Бином, 2005

7 Хернандес М. Вьесках. Дж. SQL. – СПб.: Питер, 2001

8 Петцольд Ч. Программирование для MicrosoftWindows на C# – М.: Издательско-торговый дом «Русская редакция», 2002

9 Холецкий В.И. Windows7операционная система. – СПб.: Питер, 2011

10 ГОСТ 19.301-2000 ЕСПД. Программа и методика испытаний

11 ГОСТ 19.401-2000 ЕСПД. Текст программы

12 ГОСТ 19.402-2000 ЕСПД. Описание программы

13 ГОСТ2.105-95 ЕСКД. Общие требования к текстовым документам