БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине «Базы данных»

Тема: «Разработка приложения с базой данных: Автобаза – информационная система учета работы транспорта и водителей»

|  |  |
| --- | --- |
| Исполнитель: | студент 3 курса группа 10701116  Баран Дмитрий Александрович |
| Руководитель проекта: | доцент Куприянов Андрей Борисович |

Минск 2019

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Базы данных»

Тема: «Разработка приложения с базой данных: Автобаза – информационная система учета работы транспорта и водителей »

|  |  |
| --- | --- |
| Исполнитель: | студент 3 курса группа 10701116  Баран Дмитрий Александрович |
| Руководитель проекта: | доцент Куприянов Андрей Борисович |

Минск 2019

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc9196595)

[**1 Проектирование базы данных** 6](#_Toc9196596)

[1.1 Построение концептуальной модели 6](#_Toc9196597)

[1.2 Построение логической модели 7](#_Toc9196598)

[1.3 Построение физической модели 12](#_Toc9196599)

[**2 Разработка приложения для работы с базой данных** 13](#_Toc9196600)

[2.1 Назначение 13](#_Toc9196601)

[2.2 Структура приложения 13](#_Toc9196602)

[2.3 Описание приложения 14](#_Toc9196603)

[2.4 Тестирование приложения 16](#_Toc9196604)

[2.5 Руководство пользователя 17](#_Toc9196605)

[**ЗАКЛЮЧЕНИЕ** 22](#_Toc9196606)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 23](#_Toc9196607)

[**ПРИЛОЖЕНИЕ** 24](#_Toc9196608)

# ВВЕДЕНИЕ

В настоящее время многие процессы нужно автоматизировать, поэтому создаются приложения, которые работают с базой данных. Благодаря таким приложениям работа многих диспетчеров становится проще, так как в один клик можно посмотреть данные из таблиц, добавить, редактировать или удалить данные из база данных.

База данных служит для упрощенного взиамодействия между различными данными, для их редактирования и просмотра. Это очень полезное обеспечение, особенно, когда говорят о самых различных информационных системах.

Говоря о теме данного проекта, мы не можем не использовать базы данных в нём, иначе приложение будет неэффективным и небезопасным. При проектировании баз данных по теме «Автобаза – информационная система учета работы транспорта и водителей» в первую очередь нужно подумать про удобство использования и помощи для упрощения работы учета водителей.

В данном приложении должна быть спроектирована таблица журнала, где ведётся учет всех заказов и водителей, которые выполняют заказы. Такая таблица с соответствующей функциональностью значительно упрощает работу ведения учёта.

Таблицы и поля должны быть спроектированы таким образом, чтобы обеспечивалась целостность данных и их непротиворечивость.

# 1 Проектирование базы данных

Процесс проектирования включает в себя следующие этапы:

* концептуальное проектирование;
* логическое проектирование;
* физическое проектирование.

## 1.1 Построение концептуальной модели

Задачей концептуального проектирования является: выделение основных объектов предметной области, которые реализуются в виде сущностей, определение свойств сущностей и связей между ними.

Основные сущности с атрибутами:

* Водитель: уникальный идентификатор водителя, фамилия, имя, контактные данные.
* Услуга предоставления авто: уникальный идентификатор, марка, номерной знак.
* Место прибытия: уникальный идентификатор, название страны, название региона, название города.
* Клиент: уникальный идентификатор, фамилия, имя, контактные данные.
* Журнал заказов: уникальные идентификаторы заказа, города, услуги, водителя, клиента, дата выполения заказа, стоимость.

Для добавления записи в журнал заказов нужно будет указать дату выполнения заказа, стоимость, число мест в автомобиле, услугу, клиента, водителя, город. Затем, нужно сохранить изменения в базу данных. Также можно будет добавлять, редактировать и удалять во всех таблицах данные. Создавать отчёт о каждой поездке, а также можно фильтровать по времени и стоимости заказа. На рисунке 1 изображена ER-модель базы данных.

ER-модель - модель данных, позволяющая описывать концептуальные схемы предметной области. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

На ER-диаграмме (рисунок 1) сущности обозначены прямоугольниками, а их атрибуты овалами.

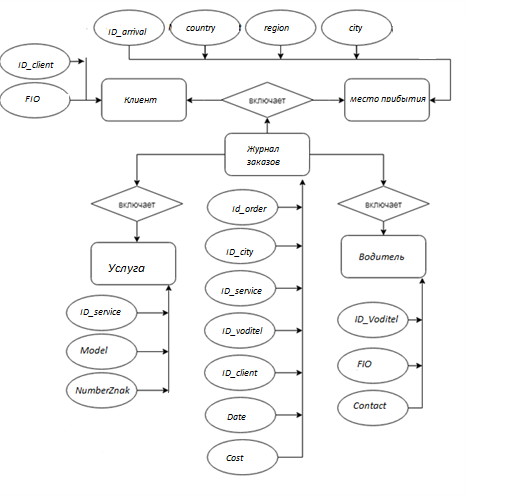


Рисунок 1 – ER-диаграмма базы данных

## 1.2 Построение логической модели

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

На таблице приведенной ниже логическая модель:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица | Поле | Описание | Тип данных (длина) | Ограничения | Значение по умолч. | Допускается  NULL |
| Журнал | ID\_order | Идентификатор заказа | int | PK, identity | Нет | Нет |
| ID\_city | Идентификатор города | int | FK, Ref city(id) | Нет | Нет |
| ID\_service | Идентификатор услуги | int | FK, Ref service(id) | Нет | Нет |
| ID\_Voditel | Идентификатор водителя | int | FK, Ref voditel(id) | Нет | Нет |
| Стоимость | Стоимость | float | >0 | Нет | Нет |
| Date | Дата выполнения | date | - | Текущая дата | Нет |
| Место прибытия | ID\_arrival | Идентификатор мест назначения | int | PK, identity | Нет | Нет |
| country | Старана | varchar(50) | До 50 символов | Нет | Нет |
| region | Название региона | varchar(50) | До 50 символов | Нет | Нет |
| City | Город прибытия | varchar(50) | До 50 символов | нет | нет |
| Услуга | ID\_service | Идентификатор услуги | int | PK, identity | Нет | Нет |
| Model | Модель | varchar(50) | До 50 символов | Нет | Нет |
| NumberZnak | Номерной знак | int | >999 | Нет | Да |
| Водитель | ID\_Voditel | Идентификатор водителя | int | PK, identity | Нет | Нет |
| FIO | ФИО | varchar(50) | До 50 символов | Нет | Нет |
| Contact | Контактные данные | varchar(50) | До 50 символов | Нет | Нет |
| Клиент | ID\_client | Идентификатор клиента | int | PK, identity | Нет | Нет |
| FIO | ФИО клиента | varchar(50) | До 50 символов | Нет | Нет |
| Contact | Контактные данные | int | >0 | Нет | Нет |

Перечень представлений, хранимых процедур, функций пользователя и триггеров:

JournalFromCostRange – процедура поиска записей в журнале по диапазону стоимости:

CREATE PROCEDURE [dbo].[RecordFromCostRange]

@StartCostRange float,

@EndCostRange float

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM journal J

WHERE J.cost >= @StartCostRange AND J.cost <= @EndCostRange

END

GO

InsertJournal – процедура добавления новой записи в журнал, возвращающая идентификатор новой записи:

ALTER PROCEDURE InsertJournal

@date,

@cost,

@id\_voditel,

@id\_city,

@id\_order,

@id int OUTPUT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Journal VALUES (@user\_id, @id\_order, @id\_city, @id\_voditel, @cost, @date GETDATE());

SET @id = @@IDENTITY;

END

JournalFromDateToDate - процедура поиска записей в журнале за определенный промежуток времени:

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[RecordFromDateToDate]

@FirstDate date,

@SecondDate date

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM journal J

WHERE J.Date >= @FirstDate AND J.Date <= @SecondDate

END

GO

showAllJournalInformation – представление всей информации из всех таблиц:

VIEW showAllJournalInformation

SELECT dbo.driver.surname, dbo.driver.contact, dbo.arrival.country, dbo.arrival.region, dbo.arrival.city, dbo.client.fullName, dbo.client.contact, dbo.service.name,

dbo.service.model, dbo.service.NumberZnak, dbo.journal.quantity, dbo.journal.Date, dbo.journal.cost

FROM dbo.arrival

INNER JOIN dbo.journal ON dbo.arrival.idCity = dbo.journal.cityId

showAllDriverSaleInformation – представление всей информации о водителе и общей суммы продажи в журнале:

VIEW showAllDriverSaleInformation

SELECT dbo.journal.journalId, dbo.journal.cost,

dbo.driver.fullName, dbo.driver.contact

FROM dbo.driver INNER JOIN dbo.journal

ON dbo.driver.driverId = dbo.journal.driverId

whenDeleteOrders - триггер запрета удаления товара, если о нем есть записи в журнале:

GO

CREATE TRIGGER whenDeleteOrders

ON [dbo].[service]

AFTER DELETE

AS

BEGIN

IF (SELECT COUNT(\*)

FROM deleted, Journal

Where deleted.servicetId = Journal.servicetId) > 0

BEGIN

ROLLBACK TRANSACTION

END

END

whenDeleteDrivers - триггер запрета удаления водителя, если о нем есть записи в журнале:

GO

CREATE TRIGGER whenDeleteOrders

ON [dbo].[driver]

AFTER DELETE

AS

BEGIN

IF (SELECT COUNT(\*)

FROM deleted, Journal

Where deleted.driverId = Journal.drivertId) > 0

BEGIN

ROLLBACK TRANSACTION

END

END

` CostCantBeZero – триггер, сумма продажи не может быть меньше 1:

GO

CREATE TRIGGER CostCantBeZero

ON [dbo].[journal]

AFTER INSERT, UPDATE

AS IF UPDATE(cost)

BEGIN

IF (SELECT cost

FROM inserted) < 1

BEGIN

ROLLBACK TRANSACTION

END

END

allCost - скалярная функция для вычисления общей суммы в журнале:

CREATE FUNCTION allCost ()

RETURNS float

AS

BEGIN

DECLARE @fullNumber float

SELECT @fullNumber = SUM(J.cost)

FROM journal J

RETURN @fullNumber

END GO

## 1.3 Построение физической модели

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Физическая модель обеспечивается SQL SERVER MANAGEMENT STUDIO по умолчанию: один вид файлов хранит саму базу данных, а другой вид файлов хранит все транзакции базы данных, количество и размеры данных файлов установлены по умолчанию.

Данные об файловой структуры базы данных отображены на рисунке 1.1 и рисунке 1.2.

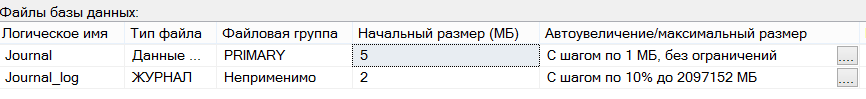
+

Рисунок 1.1 – Данные об файловой структуре

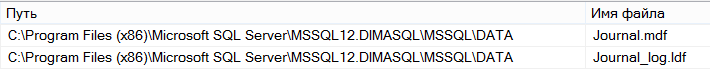


Рисунок 1.2 – Путь хранения базы данных

# Разработка приложения для работы с базой данных

## 2.1 Назначение

Разрабатываемое приложение будет служить для работы с базой данных «Автобаза – информационная система учета работы транспорта и водителей».

Приложение должно содержать функционал по добавлению, изменению, удалению информации об водителях, автомобилях, дате выполнения, городе, журнале ведения учёта заказов. Приложение должно максимально упростить взаимодействие пользователя и базы данных.

Предметная область представлена следующей информацией:

* водители;
* автомобили;
* города назначения;
* предоставляемой услуге;
* клиенте;
* журнал учёта.

## 2.2 Структура приложения

Приложение представляет собой автоматизированную информационную систему учета работы транспорта и водителей.

Приложение было разработано в виде приложения Windows Forms в среде Microsoft Visual Studio 2017.

В приложении для подключения и работы с базой данных используются встроенные инструменты MS Visual Studio:

* DataSet;
* DataView;
* DataGridView;
* SqlConnection;
* SqlCommand.

## 2.3 Описание приложения

Для хранения таблиц «Cities», «Clients», «Drivers», «Services», «Journal», а также для хранения представлений при необходимости был использован объект DataSet (рис. 2.1).

Для получения таблиц для DataSet был использован класс DataAdapter.

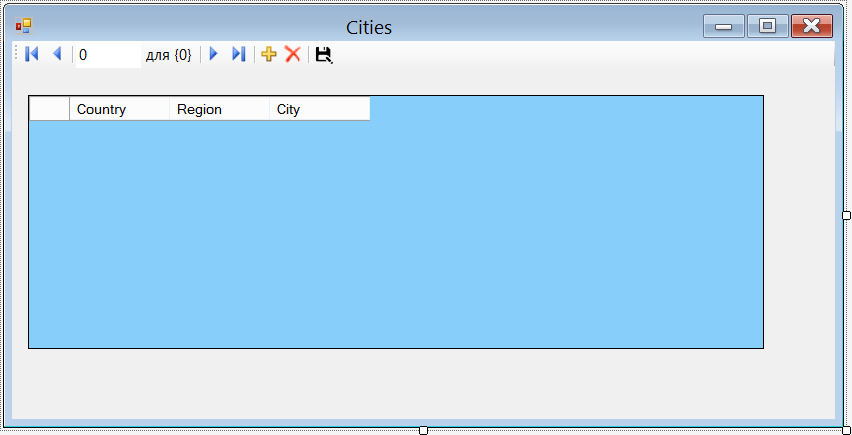


Рисунок 2.1 – использование DataSet

Для отдельного хранения каждой таблицы был использован класс DataTable, который содержит такие таблицы, как «Cities», «Clients», «Drivers», «Services», «Journal».

Код использования DataTable для таблицы «Journal»:

private void FormJournal\_Load(object sender, EventArgs e)

{

this.journalTableAdapter.Fill(this.journalDataSet.journal);

}

Для фильтрации данных в таблицах используется класс DataView. Он содержит свойство RowFilter, при присваивании значении которому, таблица, для которого был создан DataView будет отфильтрован согласно строке фильтрации.

Пример использования фильтрации для представления «Search»:

private void button2\_Click(object sender, EventArgs e)

{

double start, end;

bool s = double.TryParse(textBox1.Text, out start);

if (!s) start = 1;

s = double.TryParse(textBox2.Text, out end);

if (!s) end = 100;

this.journalTableAdapter.FillByCost(this.journalDataSet.journal, start, end);

}

Для отображения содержимого базы данных в приложении используется компонент DataGridView. **Этот компонент применяется ко всем таблицам приложения «**Cities», «Clients», «Drivers», «Services», «Journal», а также представлениям.

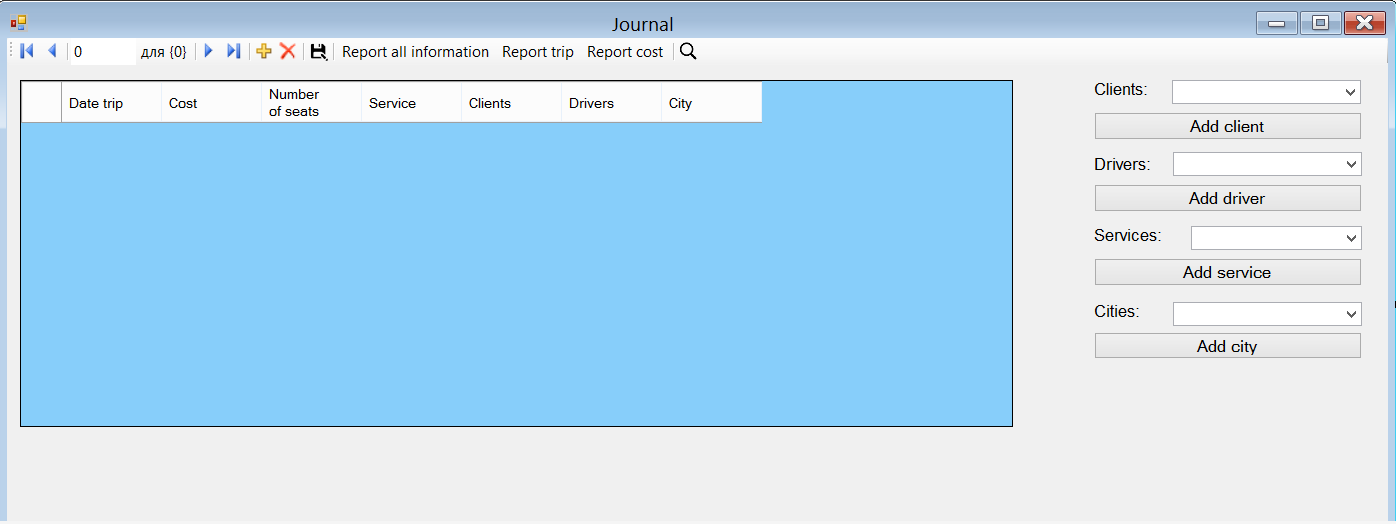


Рисунок 2.2 – применение DataGridView к таблице «Journal»

Для работы с полями, содержащими дату и время использовался компонент dateTimePicker. Был применен для поля DateTime в формах для работы с таблицами «Search report».

private void button1\_Click(object sender, EventArgs e)

{

string start = "" + dateTimePicker1.Value;

string end = "" + dateTimePicker2.Value;

this.journalTableAdapter.FillByDate(this.journalDataSet.journal, start.Substring(0, 10) , end.Substring(0, 10));

}

Для оформления отчета и создания справки в приложении использовался компонент ReportViewer. С помощью этого компонента были созданы формы «ReportDateForm», «ReportForm», «ReportOneForm». Для отображения отчета на этих компонентах были созданы элементы «Report1», «Report2», «Report3». Также был офрмлен отчёт с параметром, в качестве параметра выступает дата, клиент, стоимость.

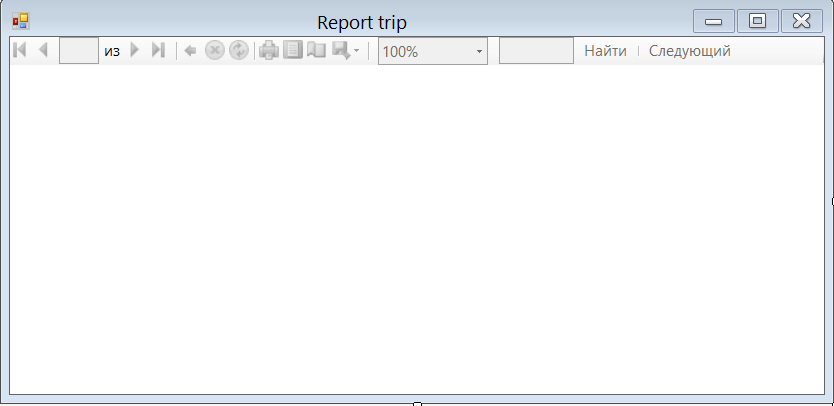


Рисунок 2.3 – форма отчета о заказанной услуге

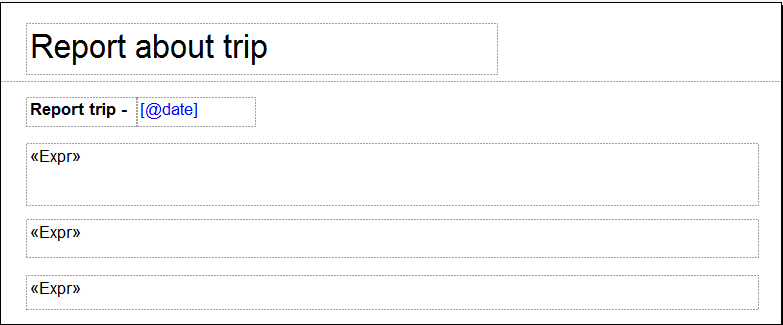


Рисунок 2.4 – эскиз отчета

## 2.4 Тестирование приложения

Тестирование созданного программного продукта будем производить на следующей конфигурации компьютера: Intel® Core™ i7-4720@ 2.60 GHz, 8.00 Гб RAM, ОС Windows 8.1. Для того, чтобы приложение запустилось на пк должно быть установлено:

* MS Visual Studio 2017;
* SQL Server 2014;
* SQL Management Studio 2014.

В процессе создания программы были устранены серьезные ошибки, носящие важное влияние на работу приложения. В конечном продукте явных ошибок обнаружено не было.

## 2.5 Руководство пользователя

Программа написана на языке C# в среде MS Visual Studio 2017. Функционал реализован в соответствии с требованиями к проекту.

После запуска приложения пользователь попадает на главное окно прилоложения (рис. 2.5).

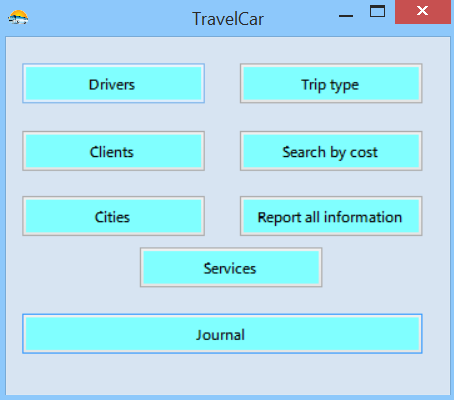


Рисунок 2.5 – главное окно

С главного окна пользователь может перейти в любую форму, при нажатии на кнопку “Drivers” пользователю открывается окно таблицы водителей (рис. 2.6).

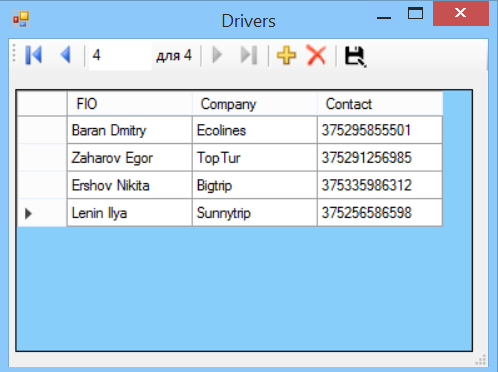


Рисунок 2.6 – окно таблицы водителей

Чтобы перейти на окно клинетов, то нужно нажать на кнопку “Clients” на главном окне (рис. 2.7).

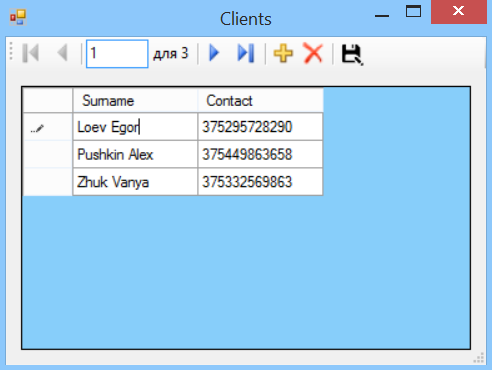


Рисунок 2.7 – окно таблицы клиентов

Аналогично, чтобы перейти на окна таблиц “Cities”, “Service”, “Trip Type”, “Journal”, то нужно кликнуть по соответстующим кнопкам на главном окне приложения.

После нажатия на кнопку “Journal” открывается окно журнала учёта водителей (рис. 2.8).

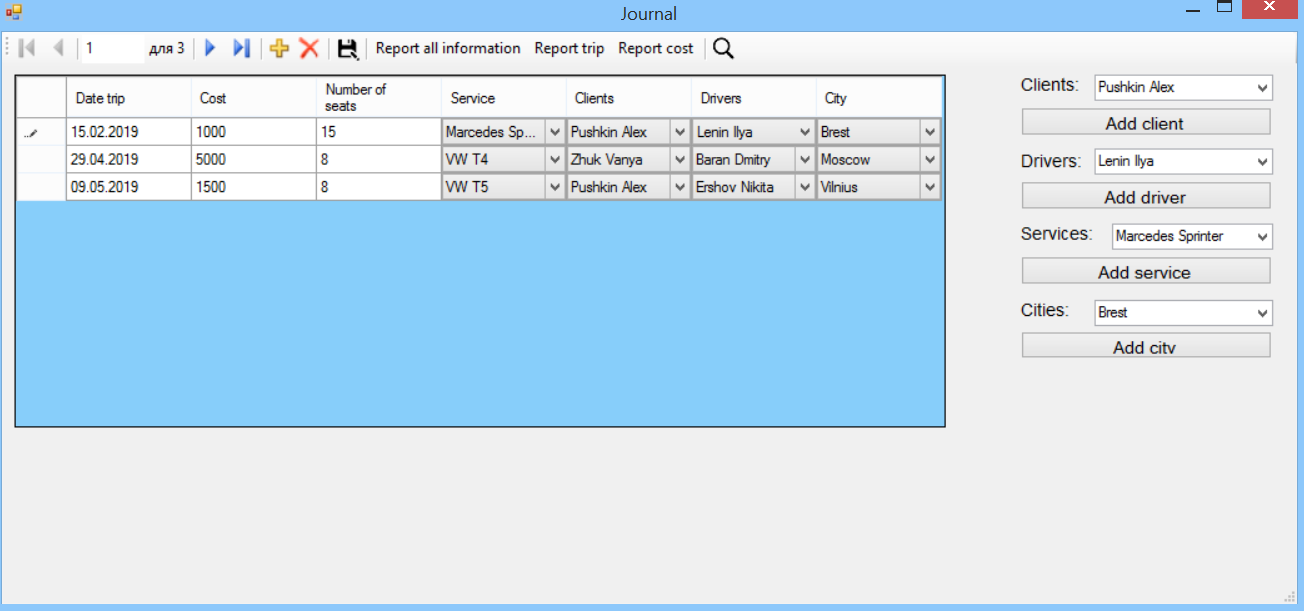


Рисунок 2.8 – окно журнала учёта водителей

Вся основная работа пользователя происходит именно в этом окне. Здесь пользователь может добавлять, редактировать и удалять записи в журнале. Также можно добавить нового клиента, водителя, услугу, город. Есть возможность с этого окна открыть в модальном окне окна клиентов, водителей, услуг и городов. После внесения изменений в другие таблицы, изменения сразу же подтягиваются в окне журнала.

При внесении изменений в таблицы и выходе, появляется окно с подтверждением выполненных действий (рис. 2.9).

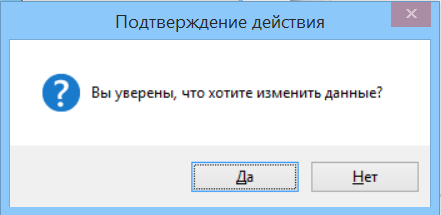


Рисунок 2.9 – окно подтверждения действий

Пользователь, находясь в окне журнала учёта водителей, может открыть отчёты “Report all information”, “Report trip”, “Report cost” и окно поиска “Search” нажатием на соответстующие кнопки сверху окна (рис. 2.10).

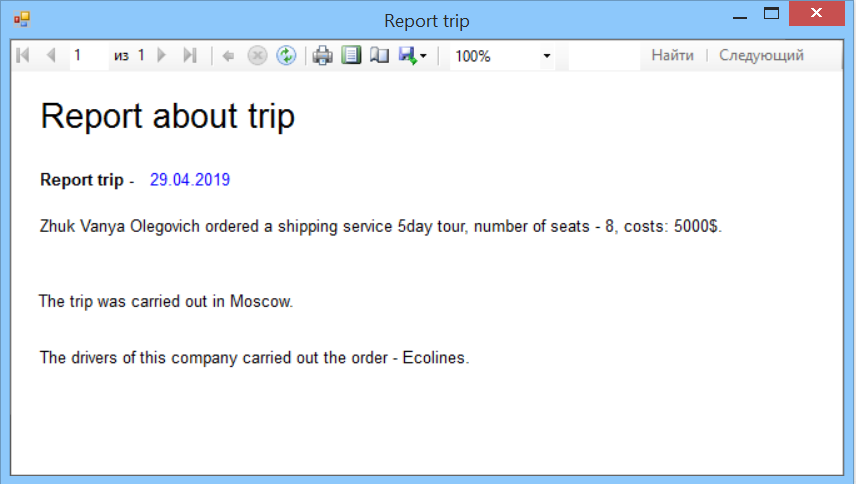


Рисунок 2.10 – окно отчёта с параметром

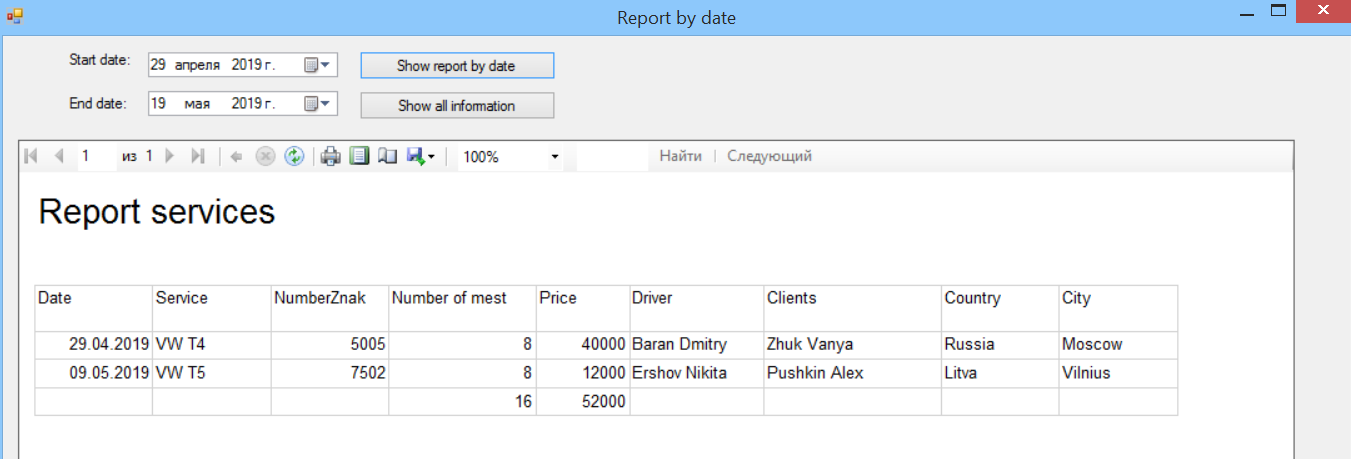


Рисунок 2.11 – окно отчёта

Нажав на главной форме на кнопку “Search by cost” открывается окно поиска по двум параметрам:

* по цене;
* по периоду времени.

Выбрав период даты, можно увидеть журнал в пределах выбранных дат (рис. 2.12)

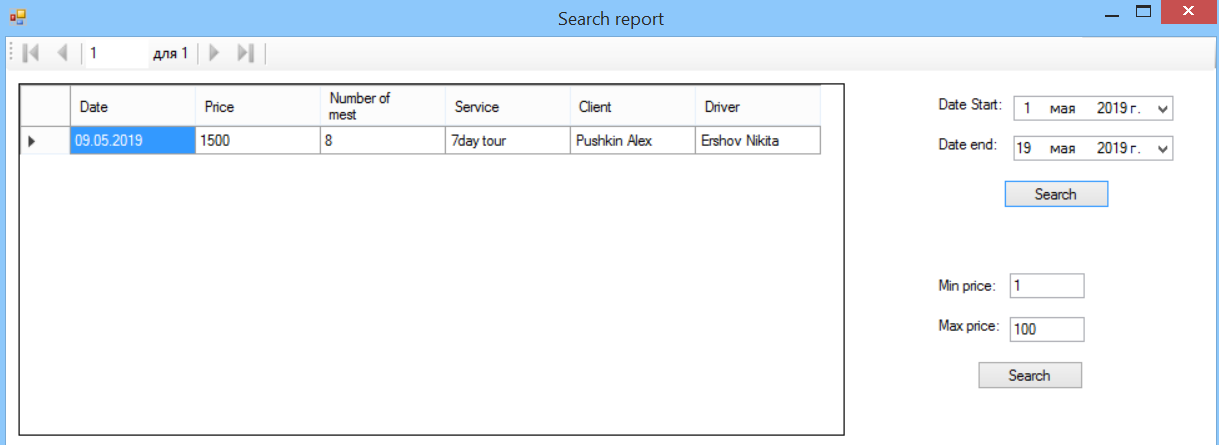


Рисунок 2.12 – окно журнала за выбранный период

Выбрав минимальную и максимульную стоимость услуги, можно увидеть журнал в пределах выбранных значений (рис. 2.13)

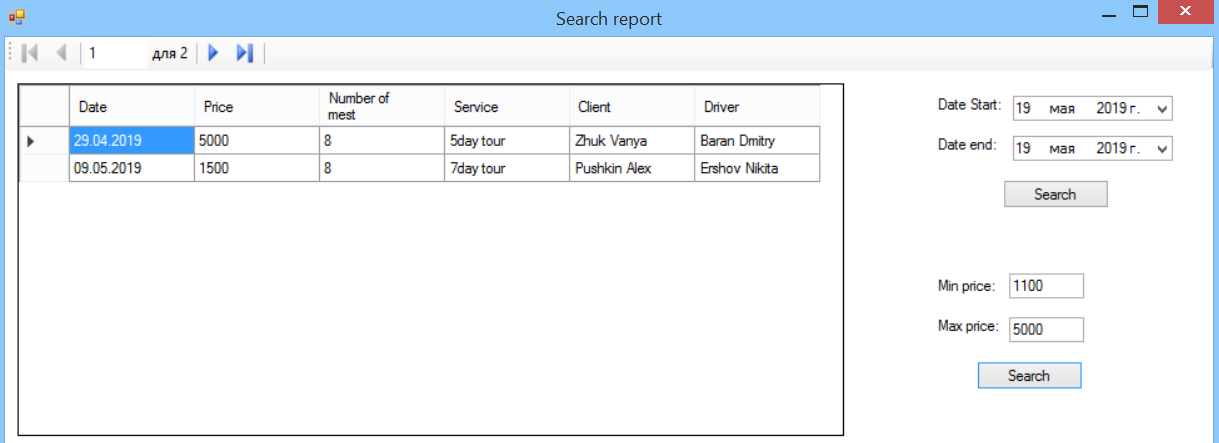


Рисунок 2.13 – окно журнала в предалах ввёдных значений стоимости

Для закрытия приложения нужно нажать на крестик в правом верхнем углу главного окна. После закрытия приложения все изменённые данные будут сохранены в базу данных.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была спроектирована база данных и разработано приложение для работы с этой базой данных. Исходя из цели, было разработано приложение отображения информации о водителе, услуге предоставляемой, клиентах, городах прибытия и для редактирования журнала учёта водителей.

Разработанное приложение можно использовать на производствах, логистических и туристических фирмах. Благодаря простоте использования это приложение значительно упрощает и ускоряет работу по ведению учёта водителей.

Приложение имеет пользовательский интерфейс, разработанный посредством использования средств, предоставляемых Microsoft Visual Studio. В связи с этим, данных продукт способен работать на любых машинах, поддерживающих операционную систему Windows и не обладающих сверхмощной производительностью.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Понимание SQL» — Мартин Грубер. – Яз. Русск. – Москва, 1993.
2. «Изучаем SQL» — Алан Бьюли. – Яз. Русск.
3. Wikipedia [Электронный ресурс]. – Wikipedia, 20.03.2019. –https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0\_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85, - Загл. С экрана. – Яз. Русск.
4. Sql-language [Электронный ресурс]. -sql-language, 17.05.2018 – **Язык запросов SQL** https://sql-language.ru/query-select.html, свободный, - Загл. С экрана. – Яз. Русск.
5. Metanit [Электронный ресурс]. – Metanit.com, 19.05.2019 - <https://metanit.com/sharp/general.php>, - Загл. С экрана. – Яз. Русск.

# ПРИЛОЖЕНИЕ

Листинг программы:

Главная форма:

namespace journal

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

FormJournal form = new FormJournal();

form.ShowDialog();

}

private void button2\_Click(object sender, EventArgs e)

{

FormDriver form = new FormDriver();

form.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

FormClients form = new FormClients();

form.ShowDialog();

}

private void button4\_Click(object sender, EventArgs e)

{

FormCity form = new FormCity();

form.ShowDialog();

}

private void button5\_Click(object sender, EventArgs e)

{

FormService form = new FormService();

form.Owner = this;

form.ShowDialog();

}

private void button6\_Click(object sender, EventArgs e)

{

FormTripType form = new FormTripType();

form.ShowDialog();

}

private void button7\_Click(object sender, EventArgs e)

{

SearchForm form = new SearchForm();

form.Owner = this;

form.ShowDialog();

}

private void button8\_Click(object sender, EventArgs e)

{

ReportForm form = new ReportForm();

form.Owner = this;

form.ShowDialog();

}

private void button9\_Click(object sender, EventArgs e)

{

}

}

}

Форма городов:

namespace journal

{

public partial class FormCity : Form

{

private bool itemChanged = false;

public FormCity()

{

InitializeComponent();

}

private void FormAddresses\_Load(object sender, EventArgs e)

{

this.addressTableAdapter.Fill(this.journalDataSet.address);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

save();

}

private bool save()

{

try

{

if (common.saveChanges(itemChanged))

{

Validate();

addressBindingSource.EndEdit();

addressTableAdapter.Update(journalDataSet);

return true;

}

else

{

journalDataSet.RejectChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

itemChanged = false;

}

return false;

}

private void FormAddresses\_FormClosing(object sender, FormClosingEventArgs e)

{

FormJournal main = this.Owner as FormJournal;

if (save() && main !=null)

{

main.addressTableAdapter.Fill(main.journalDataSet.address);

}

}

private void FormAddresses\_Shown(object sender, EventArgs e)

{

itemChanged = false;

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

itemChanged = true;

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

itemChanged = true;

}

}

}

Форма клиентов:

namespace journal

{

public partial class FormClients : Form

{

private bool itemChanged = false;

public FormClients()

{

InitializeComponent();

}

private void FormCustomers\_Load(object sender, EventArgs e)

{

this.customerTableAdapter.Fill(this.journalDataSet.customer);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

save();

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

itemChanged = true;

}

private void FormCustomers\_FormClosing(object sender, FormClosingEventArgs e)

{

FormJournal main = this.Owner as FormJournal;

if (save() && main != null)

{

main.customerTableAdapter.Fill(main.journalDataSet.customer);

}

}

private void FormCustomers\_Shown(object sender, EventArgs e)

{

itemChanged = false;

}

private bool save()

{ try

{

if (common.saveChanges(itemChanged))

{

Validate();

customerBindingSource.EndEdit();

customerTableAdapter.Update(journalDataSet);

return true;

}

else

{

journalDataSet.RejectChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

itemChanged = false;

}

return false;

}

}

}

Форма водителей:

namespace journal

{

public partial class FormDriver : Form

{

private bool itemChanged = false;

public FormDriver()

{

InitializeComponent();

}

private void FormProdusers\_Load(object sender, EventArgs e)

{

this.produserTableAdapter1.Fill(this.journalDataSet.produser);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

save();

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

itemChanged = true;

}

private void FormProdusers\_FormClosing(object sender, FormClosingEventArgs e)

{

FormJournal main = this.Owner as FormJournal;

if (save() && main != null)

{

main.produserTableAdapter.Fill(main.journalDataSet.produser);

}

}

private void FormProdusers\_Shown(object sender, EventArgs e)

{

itemChanged = false;

}

private bool save()

{

try

{

if (common.saveChanges(itemChanged))

{

Validate();

produserBindingSource.EndEdit();

produserTableAdapter1.Update(journalDataSet);

return true;

}

else

{

journalDataSet.RejectChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

itemChanged = false;

}

return false;

}

}

}

Форма журнала:

namespace journal

{

public partial class FormJournal : Form

{

private bool itemChanged = false;

public FormJournal()

{

InitializeComponent();

}

private void FormJournal\_Load(object sender, EventArgs e)

{

this.addressTableAdapter.Fill(this.journalDataSet.address);

this.goodsTableAdapter.Fill(this.journalDataSet.goods);

this.produserTableAdapter.Fill(this.journalDataSet.produser);

this.customerTableAdapter.Fill(this.journalDataSet.customer);

this.journalTableAdapter.Fill(this.journalDataSet.journal);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

save();

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

validate();

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void save()

{

try

{

if (common.saveChanges(itemChanged))

{

Validate();

journalBindingSource.EndEdit();

journalTableAdapter.Update(journalDataSet);

}

else

{

journalDataSet.RejectChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

itemChanged = false;

}

}

private void FormJournal\_FormClosing(object sender, FormClosingEventArgs e)

{

save();

}

private void FormJournal\_Shown(object sender, EventArgs e)

{

itemChanged = false;

}

private void button2\_Click(object sender, EventArgs e)

{

FormDriver cat = new FormDriver();

cat.Owner = this;

cat.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

FormService cat = new FormService();

cat.Owner = this;

cat.ShowDialog();

}

private void button1\_Click(object sender, EventArgs e)

{

FormClients cat = new FormClients();

cat.Owner = this;

cat.ShowDialog();

}

private void toolStripButton1\_Click\_1(object sender, EventArgs e)

{

save();

}

private void validate()

{

for (int index = 0; index < journalDataGridView.CurrentRow.Cells.Count; index++)

{

DataGridViewCellCollection cells = journalDataGridView.CurrentRow.Cells;

if (cells[index].Value == DBNull.Value)

{

switch (index)

{

case 3: cells[3].Value = comboBox3.SelectedValue; break;

case 4: cells[4].Value = comboBox1.SelectedValue; break;

case 5: cells[5].Value = comboBox2.SelectedValue; break;

case 6: cells[6].Value = 2; break;

case 0: cells[0].Value = DateTime.Now.Date; break;

case 1: cells[1].Value = 1; break;

case 2: cells[2].Value = 1; break;

}

}

}

}

public void journalDataGridView\_CellLeave(object sender, DataGridViewCellEventArgs e)

{

validate();

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

ReportForm form = new ReportForm();

form.Owner = this;

form.ShowDialog();

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

ReportOneForm form = new ReportOneForm();

form.Owner = this;

form.ShowDialog();

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

ReportDateForm form = new ReportDateForm();

form.Owner = this;

form.ShowDialog();

}

private void toolStripButton5\_Click(object sender, EventArgs e)

{

SearchForm form = new SearchForm();

form.Owner = this;

form.ShowDialog();

}

}

}

Форма поиска:

namespace journal

{

public partial class SearchForm : Form

{

public SearchForm()

{

InitializeComponent();

}

private void journalBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

this.Validate();

this.journalBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.journalDataSet);

}

private void SearchForm\_Load(object sender, EventArgs e)

{

this.produserTableAdapter.Fill(this.journalDataSet.produser);

this.customerTableAdapter.Fill(this.journalDataSet.customer);

this.goodsTableAdapter.Fill(this.journalDataSet.goods);

this.journalTableAdapter.Fill(this.journalDataSet.journal);

}

private void button1\_Click(object sender, EventArgs e)

{

string start = "" + dateTimePicker1.Value;

string end = "" + dateTimePicker2.Value;

this.journalTableAdapter.FillByDate(this.journalDataSet.journal, start.Substring(0, 10) , end.Substring(0, 10));

}

private void button2\_Click(object sender, EventArgs e)

{

double start, end;

bool s = double.TryParse(textBox1.Text, out start);

if (!s) start = 1;

s = double.TryParse(textBox2.Text, out end);

if (!s) end = 100;

this.journalTableAdapter.FillByCost(this.journalDataSet.journal, start, end);

}

}

Форма услуг:

namespace journal

{

public partial class FormService : Form

{

private bool itemChanged = false;

public FormService()

{

InitializeComponent();

}

private void FormProducts\_Load(object sender, EventArgs e)

{

this.categoryTableAdapter.Fill(this.journalDataSet.category);

this.goodsTableAdapter.Fill(this.journalDataSet.goods);

}

private void FormProducts\_Shown(object sender, EventArgs e)

{

itemChanged = false;

}

private bool save()

{

try

{

if (common.saveChanges(itemChanged))

{

Validate();

goodsBindingSource.EndEdit();

goodsTableAdapter.Update(journalDataSet);

return true;

}

else

{

journalDataSet.RejectChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

itemChanged = false;

}

return false;

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

save();

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

itemChanged = true;

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

validate();

}

private void validate()

{

for (int index = 0; index < dataGridView1.CurrentRow.Cells.Count; index++)

{

DataGridViewCellCollection cells = dataGridView1.CurrentRow.Cells;

if (cells[index].Value == DBNull.Value)

{

switch (index)

{

case 3: cells[3].Value = 2; break;

case 0: cells[0].Value = ""; break;

case 1: cells[1].Value = ""; break;

case 2: cells[2].Value = 1; break;

}

}

}

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

itemChanged = true;

}

private void FormProducts\_FormClosing(object sender, FormClosingEventArgs e)

{

FormJournal main = this.Owner as FormJournal;

if (save() && main != null)

{

main.goodsTableAdapter.Fill(main.journalDataSet.goods);

}

}

Формы отчётов:

namespace journal

{

public partial class ReportDateForm : Form

{

public ReportDateForm()

{

InitializeComponent();

}

private void ReportDateForm\_Load(object sender, EventArgs e)

{

this.selectAllTableAdapter.Fill(this.JournalDataSet.selectAll);

this.reportViewer1.RefreshReport();

}

}

}

namespace journal

{

public partial class ReportForm : Form

{

public ReportForm()

{

InitializeComponent();

}

private void ReportForm\_Load(object sender, EventArgs e)

{

this.selectAllTableAdapter.Fill(this.JournalDataSet.selectAll);

this.reportViewer1.RefreshReport();

}

}

}