Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(МИВлГУ)

Факультет ИТР

Кафедра ИС

КУРСОВАЯ

РАБОТА

по Объектно-ориентированное программирование

(наименование дисциплины)

Тема Разработка структуры классов «Оптовая торговля»

Руководитель

Еремеев С. В. .

(оценка) (фамилия, инициалы)

Члены комиссии

(подпись) (дата)

Студент ИСз-121

(подпись) (Ф.И.О.) (группа)

Яшина А. Л.

(подпись) (Ф.И.О.) (фамилия, инициалы)

(подпись) (дата)

Муром 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc136897007)

[1. Анализ технического задания 6](#_Toc136897008)

[2. Разработка структуры классов 10](#_Toc136897009)

[3. Разработка алгоритмов 12](#_Toc136897010)

[4. Инструкция пользователя 18](#_Toc136897011)

[5. Тестирование системы 19](#_Toc136897012)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 20](#_Toc136897013)

### ВВЕДЕНИЕ

Переход России к рыночной экономике вызывает необходимость кардинальный изменений в деятельности всех отраслей народного хозяйства страны. Особую значимость приобретает формирование адекватной рыночной модели хозяйственного механизма сферы обращения и услуг, поскольку именно торговля, являясь конечным звеном экономической активности субъектов рынка, обеспечивает эффективное удовлетворение запросов и нужд потребителей.

Решение этой ответственной задачи требует от торговли не только усилий, связанной с реализацией производственной стоимости, но и выполнение сложнейшего комплекса мероприятий, обеспечивающих продвижение товаров от производителей к потребителям.

Торговля – обширнейшая область предпринимательской деятельности и сфера приложения труда – получила в последние годы новые импульсы развития, существенно расширив «поле и правила игры» в экономике переходного периода. Среди основных направлений коммерческо-посреднической деятельности на первый план выступает работа по проведению оптовой торговли. Эта работа – основная форма выражения сущности коммерческо-посреднической деятельности, способствующая активному регулированию процессов перемещения и накопления продукции в пространстве и во времени. Оптовая торговля, являясь важнейшим звеном свободного взаимодействия между участниками акта купли-продажи, реализует этот акт в крупных объемах и большими партиями товаров.

Понятие оптовой торговли и ее сущности на протяжении всего периода развития экономической теории является предметом влияния и изучения. Мнения по этому поводу у представителей науки и практиков не слишком расходятся, по крайней мере, по ключевым позициям.

На современном этапе российской экономики оптовая торговля имеет важное значение. Именно оптовая торговля является активным проводником товаров отечественного производства и достаточно большой массы товаров зарубежной продукции на внутренний рынок. Именно оптовая торговля заставляет наших производителей поднимать качество отечественных товаров для завоевания прочного места как на внутреннем, так и на внешнем рынках.

Таким образом, важность этой области деятельности предприятия не требует особых доказательств, поскольку роль и значимость оптовой торговли понятна любому предпринимателю [1].

Задачей данной курсовой работы является разработка информационной системы для организации оптовой торговли. Цель разработки информационной системы – оптимизация работы организаций по ведению оптовой торговли.

В ходе выполнения курсовой работы, будут решены следующие задачи:

- анализ технического задания и сбор требований к программе;

- проектирование программы;

- программная реализация;

- тестирование программы.

### 1. Анализ технического задания

В курсовой работе необходимо разработать информационную систему для организации ведения оптовой торговли.

Выполнять определенные действия по ведению оптовой торговли можно и на бумаге, но данный труд очень трудоемкий, работа в современных условиях становится все требовательней и требовательней. Для конкурентоспособности фирме нужно иметь информационную систему, которая позволит избавится от многих рутинных задач. Разрабатываемая ИС должна обеспечивать оформление приходных и расходных накладных, а также обеспечивать функциональность:

- управление сотрудниками;

- управление контрагентами;

- управление складом;

- управление товарами;

- управление категориями товаров;

- производить учет товара на складе;

- выводить информацию о заполнении склада.

Когда возникает необходимость создания информационной системы или составления программы для решения какой-либо частной задачи, встает вопрос о выборе для этой цели наиболее подходящего языка программирования. Во многих случаях такой выбор диктуется простыми факторами – доступностью того или иного транслятора и умением составлять программы на данном языке. Если же в распоряжении пользователя имеется достаточно большой выбор языков программирования, то следует учитывать следующие обстоятельства:

- назначение разрабатываемой программы – нужна ли она временно или будет использоваться постоянно, планируется ли передавать ее другим организациям, будут ли разрабатываться ее новые версии;

- ожидаемый размер программы – можно ли будет ее создавать как единое целое или придется разбивать на отдельные взаимодействующие модули, требуется ли минимизировать размер памяти, занимаемой программой во время работы;

- требуемая скорость работы программы, соотношение ее диалоговых и вычислительных компонентов;

- возможность и целесообразность использования имеющихся стандартных библиотек подпрограмм, процедур, функций;

- предусматривается ли возможность переноса программы на другие типы машин;

- основные типы данных, с которыми придется иметь дело, необходимость поддержки работы с действительными числами, строками, списками и другими типами структур;

- необходимость сопряжения разрабатываемой программы с другими пакетами или программами, в том числе составленными на других языках программирования;

- характер и уровень использования аппаратных средств – дисплея, клавиатуры и др., необходимость в специальном программировании некоторых функций для работы с внешними устройствами.

С точки зрения этих критериев возможности языков могут весьма сильно различаться, поэтому правильный выбор инструментального языка программирования является непростой задачей.

Для программной реализации выбираем язык C# и среду программирования Visual Studio 2022, версию Net Framework 4.8. Версию Visual Studio 2022 выбираем Community Edition, данная версии является бесплатной и свободно может скачиваться.

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ и некоторых других языков, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов)[2].

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server)[3].

Visual Studio Community Edition, представляет собой бесплатный аналог Visual Studio Professional и обладает практически тем же функционалом, что и последняя [4].

В качестве СУБД выбираем Firebird 4.0. Firebird (FirebirdSQL) — свободная кроссплатформенная реляционная система управления базами данных, работающая на macOS, Linux, Microsoft Windows и некоторых Unix-платформах[5].

### 2. Разработка структуры классов

При рассмотрении исходных данных создается структура иерархии классов информационной системы «Оптовая торговля». Для максимального соответствия заданию, создается 14 классов:

* Program.cs;
* Main.cs;
* Category.cs;
* CategoryEdit.cs;
* Contractor.cs;
* ContractorEdit.cs;
* Employee.cs;
* EmployeeEdit.cs;
* Invoices.cs;
* InvoicesEdit.cs;
* Product.cs;
* ProductEdit.cs;
* Warehouse.cs;
* WarehouseEdit.cs.

Класс Program.cs является точкой входа в приложение. Он запускает главную форму Main.cs, а также в нем реализован метод AddUpdateAppSettings для сохранения настроек подключения, с помощью этого метода доступ к строке подключения возможен из любой точки программы.

Класс Main.cs - главная форма приложения, с помощью этой формы возможен доступ к справочникам и также к журналу накладных. При загрузке формы задается строка подключения к базе данных и проверяется подключение. К кнопкам назначены события запуска других форм (справочников и журнала накладных).

Справочники реализованы классами:

* Category.cs;
* CategoryEdit.cs;
* Contractor.cs;
* ContractorEdit.cs;
* Employee.cs;
* EmployeeEdit.cs;
* Product.cs;
* ProductEdit.cs;
* Warehouse.cs;
* WarehouseEdit.cs.

В информационной системе 5 справочников: категории, контрагенты, сотрудники, товары, места на складе. В реализации справочника участвуют 2 класса:

* Название справочника.cs
* Название справочникаEdit.cs

При запуске справочника отображается форма со списком его элементов, а с постфиксом Edit форма для редактирования справочника (добавление, изменение, удаление элементов)

В каждом справочнике реализован поиск (при этом выполняется отдельный SQL запрос к базе данных), а также есть механизм сортировки.

Работа с накладными реализована 2 классами: Invoices.cs, InvoicesEdit.cs;

Для просмотра списка накладных, а также для поиска и сортировки используется класс Invoices.cs. В информационной системе реализована функция просмотра накладных, при этом имеется возможно просмотреть уже созданные накладные, для этого используется класс InvoicesEdit.

### 3. Разработка алгоритмов

Программа разработана на языке С# в среде Microsoft Visual Studio 2022. Тип проекта – Приложение Windows Forms (.NET Framework). При этом указывается платформа .NET Framework 4.8.

Работа программы реализована в диалоговом режиме с пользователем, т.е. взаимодействие пользователя с программой реализовано через графический интерфейс.

В информационной системе реализовано выполнение SQL запросов к базе данных (выборка, добавление, изменение, удаление), а также алгоритмы вычисления количества товара (приход, расход) и выбора места на складе для хранения товара.

Механизм выполнения SQL запросов реализован следующим образом (на примере справочника "Категории":

* SELECT

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

command = "SELECT ID AS \"Номер\", NAME AS \"Название категории\" FROM CATEGORY;";

categoryCommand = new FbCommand(command, fbCon);

categoryCommand.CommandType = CommandType.Text;

dr = categoryCommand.ExecuteReader();

dt = new DataTable();

dt.Load(dr);

dataGridView1.DataSource = dt;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Focus();

dataGridView1.CurrentCell = dataGridView1.Rows[0].Cells[1];

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

* UPDATE

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

categoryTransaction = fbCon.BeginTransaction();

command = "UPDATE CATEGORY SET CATEGORY.NAME = @name WHERE CATEGORY.ID = @current\_id;";

categoryCommand = new FbCommand(command, fbCon, categoryTransaction);

categoryCommand.Parameters.AddWithValue("@name", cef.getName());

categoryCommand.Parameters.AddWithValue("@current\_id", cef.getID());

categoryCommand.CommandType = CommandType.Text;

categoryCommand.ExecuteNonQuery();

categoryTransaction.Commit();

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

* INSERT

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

categoryTransaction = fbCon.BeginTransaction();

command = "INSERT INTO CATEGORY (NAME) VALUES (@name);";

categoryCommand = new FbCommand(command, fbCon, categoryTransaction);

categoryCommand.Parameters.AddWithValue("@name", cef.getName());

categoryCommand.CommandType = CommandType.Text;

categoryCommand.ExecuteNonQuery();

categoryTransaction.Commit();

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

* DELETE

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

categoryTransaction = fbCon.BeginTransaction();

command = "DELETE FROM CATEGORY WHERE CATEGORY.ID = @current\_record;";

categoryCommand = new FbCommand(command, fbCon, categoryTransaction);

categoryCommand.Parameters.AddWithValue("@current\_record", Convert.ToInt32(dataGridView1[0, dataGridView1.CurrentRow.Index].Value));

categoryCommand.CommandType = CommandType.Text;

categoryCommand.ExecuteNonQuery();

categoryTransaction.Commit();

}

catch (Exception)

{

MessageBox.Show("Запись удалить нельзя, она используется");

}

finally

{

fbCon.Close();

}

Вычисление учет товара на складе осуществляется следующим алгоритмом:

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

invoiceTransaction = fbCon.BeginTransaction();

command = "INSERT INTO INVOICE\_HEADER (ID\_CONTRACTOR, ID\_EMPLOYEE, ID\_INVOICE\_TYPE, INVOICE\_DATE) VALUES (@contractor, @employee, @invoice\_type, @invoice\_date);";

invoiceCommand = new FbCommand(command, fbCon, invoiceTransaction);

invoiceCommand.Parameters.AddWithValue("@contractor", ief.getContractor());

invoiceCommand.Parameters.AddWithValue("@employee", ief.getEmployee());

invoiceCommand.Parameters.AddWithValue("@invoice\_type", ief.getTypeInvoice());

invoiceCommand.Parameters.AddWithValue("@invoice\_date", DateTime.Now);

invoiceCommand.CommandType = CommandType.Text;

invoiceCommand.ExecuteNonQuery();

invoiceTransaction.Commit();

command = "select gen\_id(GEN\_INVOICE\_HEADER\_ID, 0) from rdb$database;";

invoiceCommand = new FbCommand(command, fbCon);

invoiceCommand.CommandType = CommandType.Text;

dr = invoiceCommand.ExecuteReader();

dr.Read();

int last\_id = Convert.ToInt32(dr.GetValue(0));

DataTable dt = ief.getDataTableProducts();

foreach (DataRow row in dt.Rows)

{

string id\_product = row["ID товара"].ToString();

string amount = row["Количество"].ToString();

string price = row["Цена"].ToString();

string place = row["Место на складе"].ToString();

invoiceTransaction = fbCon.BeginTransaction();

command = "INSERT INTO INVOICE\_TABLE\_PART (ID\_INVOICE\_HEADER, ID\_PRODUCT, AMOUNT, TOTAL\_PRICE, ID\_WAREHOUSE) VALUES (@id, @product, @amount, @price, @place);";

invoiceCommand = new FbCommand(command, fbCon, invoiceTransaction);

invoiceCommand.Parameters.Add("@id", FbDbType.Integer).Value = last\_id;

invoiceCommand.Parameters.Add("@product", FbDbType.Integer).Value = Convert.ToInt32(id\_product);

invoiceCommand.Parameters.Add("@amount", FbDbType.Integer).Value = Convert.ToInt32(amount);

invoiceCommand.Parameters.Add("@price", FbDbType.Decimal).Value = Convert.ToDecimal(price);

invoiceCommand.Parameters.Add("@place", FbDbType.Integer).Value = Convert.ToInt32(place);

invoiceCommand.CommandType = CommandType.Text;

invoiceCommand.ExecuteNonQuery();

invoiceTransaction.Commit();

command = "SELECT PRODUCT.AMOUNT FROM PRODUCT WHERE PRODUCT.ID = @current\_id;";

invoiceCommand = new FbCommand(command, fbCon);

invoiceCommand.Parameters.AddWithValue("@current\_id", Convert.ToInt32(id\_product));

invoiceCommand.CommandType = CommandType.Text;

dr = invoiceCommand.ExecuteReader();

dr.Read();

int amount\_product = Convert.ToInt32(dr.GetValue(0));

if (ief.getTypeInvoice() == "1")

{

amount\_product += Convert.ToInt32(amount);

}

else

{

amount\_product -= Convert.ToInt32(amount);

}

if(amount\_product > 0)

{

invoiceTransaction = fbCon.BeginTransaction();

command = "UPDATE PRODUCT SET PRODUCT.AMOUNT = @amount WHERE PRODUCT.ID = @current\_id;";

invoiceCommand = new FbCommand(command, fbCon, invoiceTransaction);

invoiceCommand.Parameters.AddWithValue("@amount", amount\_product);

invoiceCommand.Parameters.AddWithValue("@current\_id", Convert.ToInt32(id\_product));

invoiceCommand.CommandType = CommandType.Text;

invoiceCommand.ExecuteNonQuery();

invoiceTransaction.Commit();

}

else

{

MessageBox.Show("На складе недостаточно товара");

}

}

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

Вычисление местоположения товара на складе осуществляется алгоритмом:

if(isExpenditureInvoice)

{

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

command = "SELECT WAREHOUSE.ID AS \"Номер\", " +

"(SELECT PRODUCT.ID FROM PRODUCT WHERE PRODUCT.ID = INVOICE\_TABLE\_PART.ID\_PRODUCT) AS \"ID\", " +

"(SELECT PRODUCT.PRICE FROM PRODUCT WHERE PRODUCT.ID = INVOICE\_TABLE\_PART.ID\_PRODUCT) AS \"Цена\", " +

"(SELECT PRODUCT.NAME FROM PRODUCT WHERE PRODUCT.ID = INVOICE\_TABLE\_PART.ID\_PRODUCT) AS \"Товар\", " +

"WAREHOUSE.LINE AS \"Ряд\", " +

"WAREHOUSE.SHELF AS \"Полка\", " +

"WAREHOUSE.PLACE AS \"Место\" " +

"FROM INVOICE\_TABLE\_PART, WAREHOUSE WHERE WAREHOUSE.ID = INVOICE\_TABLE\_PART.ID\_WAREHOUSE;";

werehouseCommand = new FbCommand(command, fbCon);

werehouseCommand.CommandType = CommandType.Text;

dr = werehouseCommand.ExecuteReader();

dt = new DataTable();

dt.Load(dr);

dataGridView1.DataSource = dt;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[1].Visible = false;

dataGridView1.Columns[2].Visible = false;

dataGridView1.Focus();

dataGridView1.CurrentCell = dataGridView1.Rows[0].Cells[3];

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

}

else

{

try

{

fbCon = new FbConnection(ConfigurationManager.AppSettings["ConnectionString"]);

fbCon.Open();

command = "SELECT WAREHOUSE.ID AS \"Номер\", " +

"(SELECT PRODUCT.NAME FROM PRODUCT WHERE PRODUCT.ID = INVOICE\_TABLE\_PART.ID\_PRODUCT) AS \"Товар\", " +

"WAREHOUSE.LINE AS \"Ряд\", " +

"WAREHOUSE.SHELF AS \"Полка\", " +

"WAREHOUSE.PLACE AS \"Место\" " +

"FROM INVOICE\_TABLE\_PART RIGHT JOIN WAREHOUSE ON WAREHOUSE.ID = INVOICE\_TABLE\_PART.ID\_WAREHOUSE;";

werehouseCommand = new FbCommand(command, fbCon);

werehouseCommand.CommandType = CommandType.Text;

dr = werehouseCommand.ExecuteReader();

dt = new DataTable();

dt.Load(dr);

if (isAddInvoice)

{

RemoveNullColumnFromDataTable(dt);

}

dataGridView1.DataSource = dt;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Focus();

dataGridView1.CurrentCell = dataGridView1.Rows[0].Cells[1];

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

}

В данном алгоритме есть условие расходная или приходная накладная, при этом используются флаги, реализованные следующими методами:

//Флаг добавления накладной

public void setAddInvoiceFlag()

{

isAddInvoice = true;

}

//Флаг расходной накладной

public void setExpenditureInvoice()

{

isExpenditureInvoice = true;

}

### 4. Инструкция по программному продукту

4.1 Инструкция пользователя

Для начала работы с программой, необходимо запустить исполняемый файл приложения Wholesale.exe. После запуска приложения, будет отображено главное окно приложения.

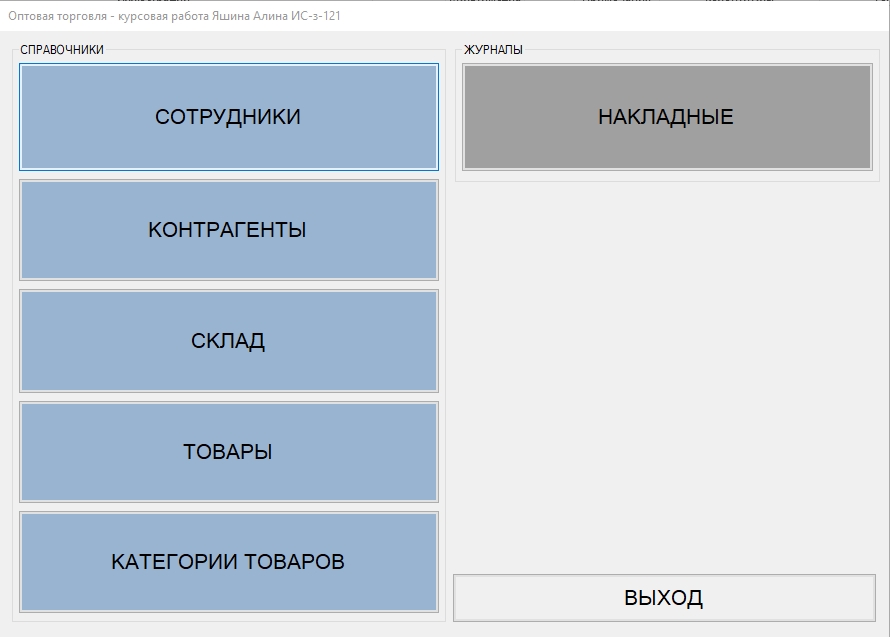


Рисунок 1 - Главное окно приложения

Пользователь может управлять справочниками Сотрудники, Контрагенты, Склад, Товары, Категории товаров, а также журналом накладных.

В качестве примера рассмотрим работу со справочником Сотрудники.

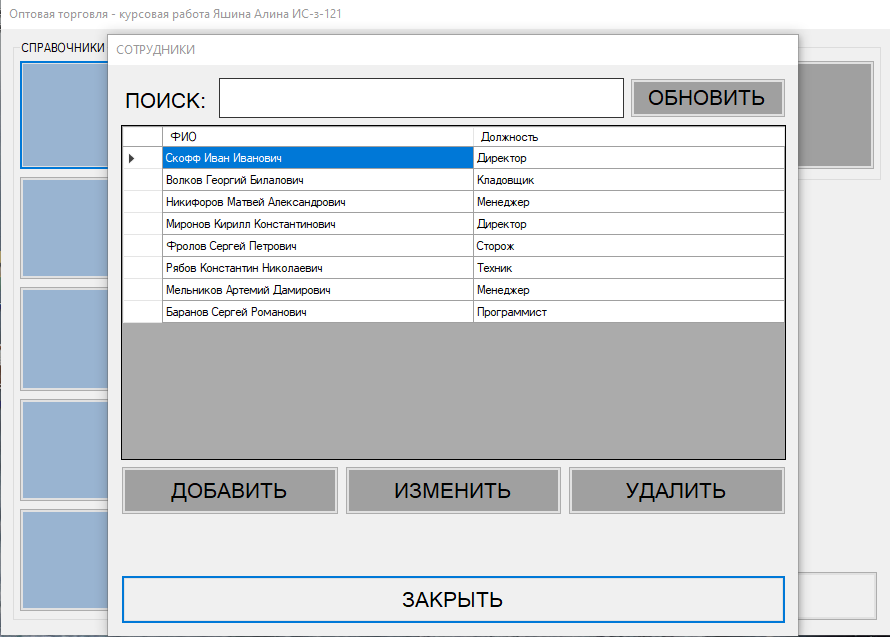


Рисунок 2 - Список сотрудников

Пользователю доступны функции добавления, изменения и удаления записей, а также поиск по справочнику.

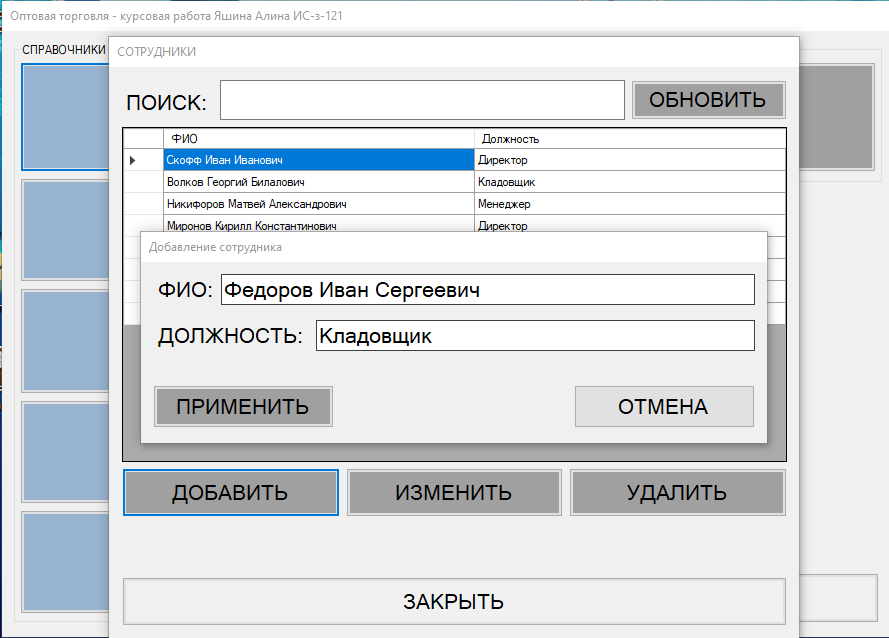


Рисунок 3 - Добавление записи в справочник Сотрудники

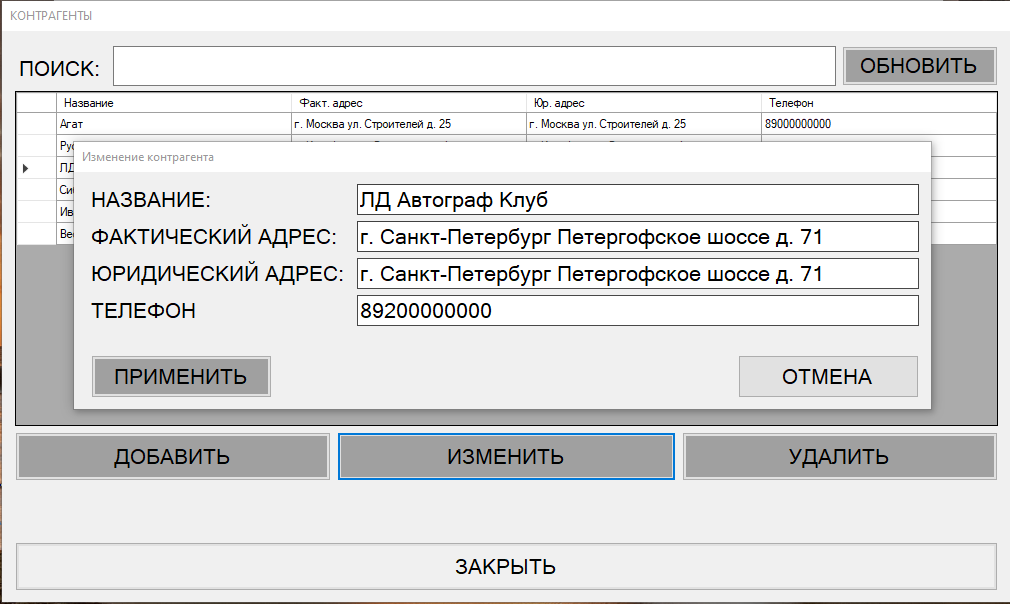


Рисунок 4 - Изменение записи в справочнике Сотрудники

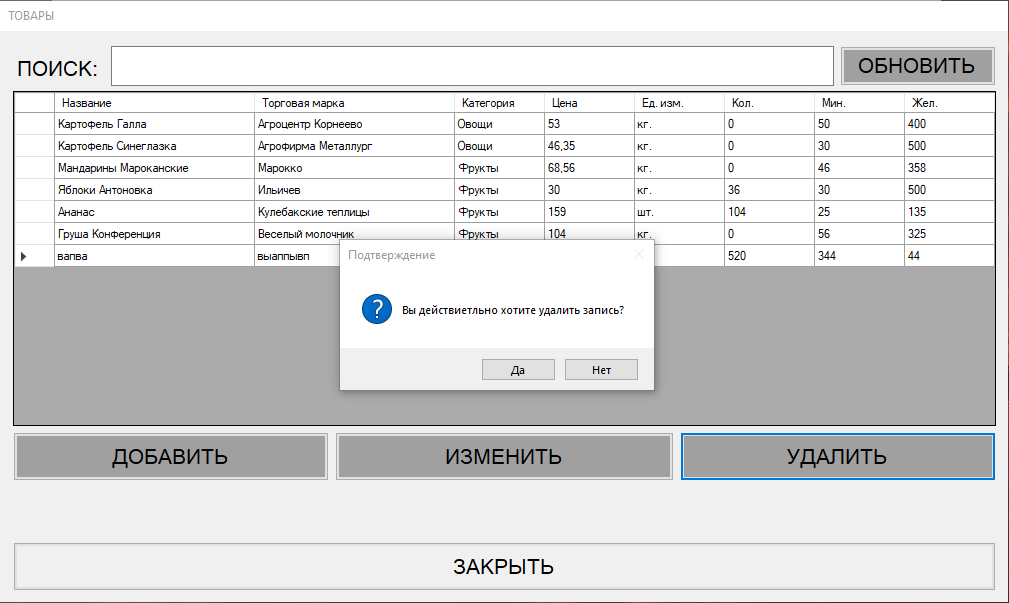


Рисунок 5 - Удаление записи в справочнике Сотрудники

Пользователю доступен журнал Накладные. В системе присутствуют 2 типа накладных - приходные и расходные. В систему можно добавить накладную и посмотреть уже сформированные.

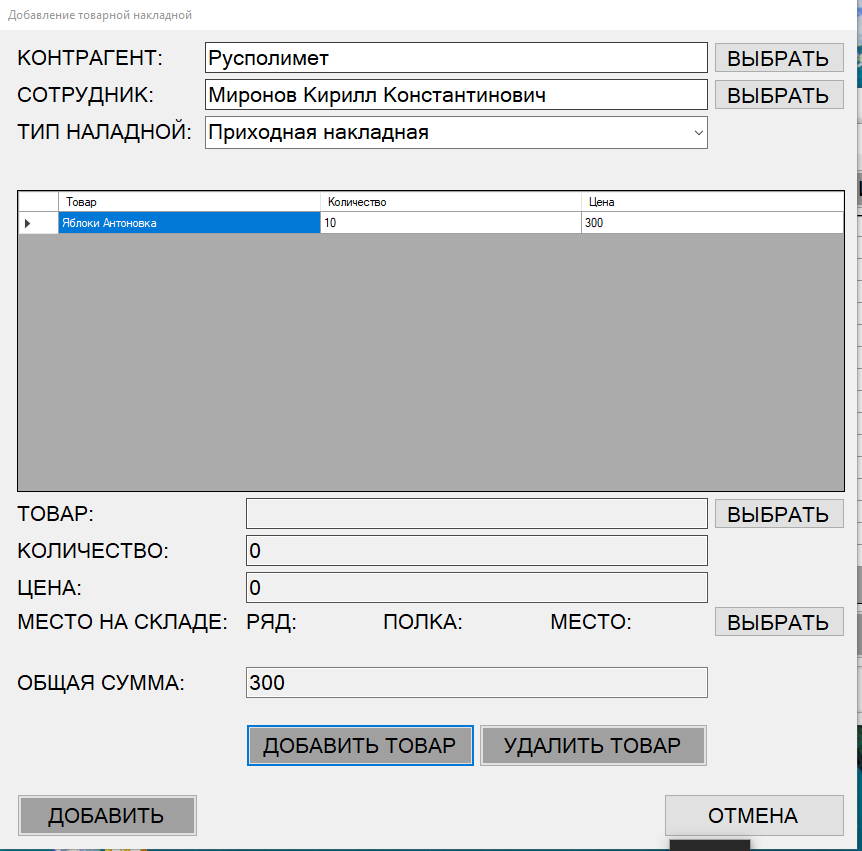


Рисунок 6 - Добавление накладной

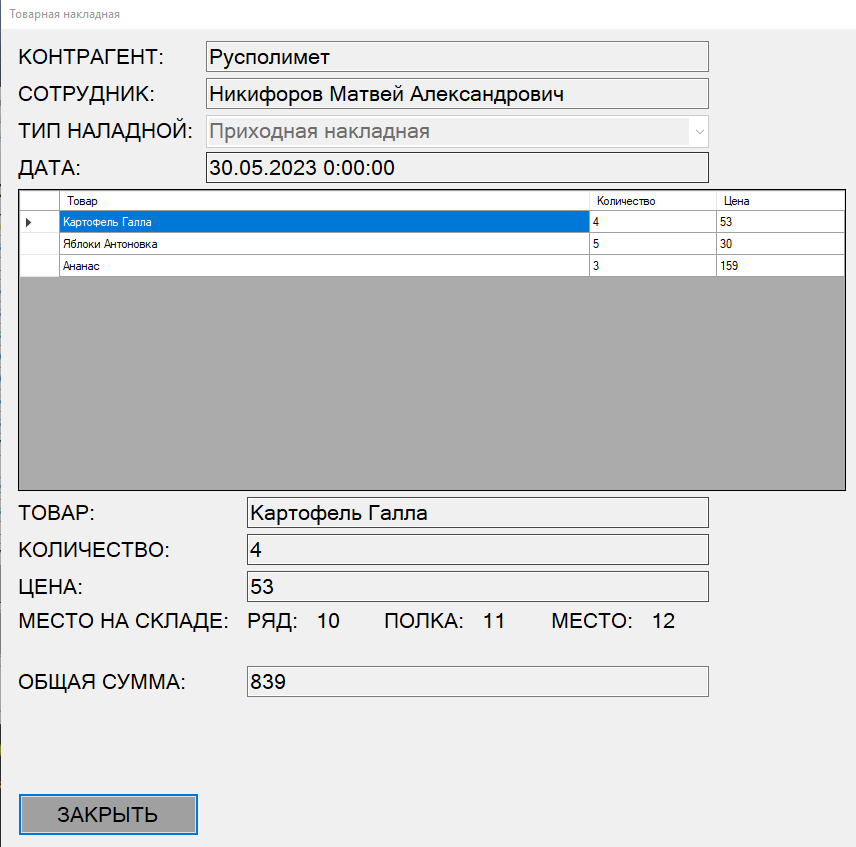


Рисунок 7 - Просмотр сформированной накладной

4.2 Инструкция администратора

Информационная система состоит из клиентского приложения и базы данных. База данных может быть размещена как на ПК, где непосредственно используется программа, так и на отдельном сервере в сети. Для функционирования клиентских приложений, необходимо наличие следующих продуктов, установленных на ПК пользователей:

- операционная система Microsoft Windows 7/8/10;

- .NET Framework 4.8

Для обеспечения связи клиентских рабочих мест с сервером БД, необходима локальная сеть с поддержкой протокола TCP/IP.

Для развертывания серверной стороны ИС, необходимо:

- операционная система Microsoft Windows 7/8/10;

- Firebird SQL Server версии 4.0.

При первоначальном развертывании системы, на серверной стороне необходимо подключить и настроить базу данных со следующими параметрами:

- Имя базы данных: WHOLESALE.FDB;

- Пользователь: SYSDBA;

- Пароль: masterkey.

- Кодовая страница базы данных: UTF-8.

Работа программы возможна без настройки серверной стороны, с локальной базой данных расположенной в виде файла в каталоге программы. Для этого необходимо наличие библиотеки Embedded Firebird Server (fbembed.dll). Данная библиотека должна располагаться в одном каталоге с исполняемым файлом программы.

**5. Тестирование системы**

Под тестированием программного обеспечения понимается процесс выявления ошибок в программном обеспечении и проверка соответствия реализованного функционала требованиям технического задания.

В рамках данной курсовой работы произведено функциональное тестирование разработанного программного продукта по приведенной ниже методике.

1. Произведен запуск программы. Результат: отображено главное окно приложения;

2. Произведено заполнение справочников;

3. Сформирована приходная накладная;

4. Проверено количество поступившего товара на склад

5. Сформирована расходная накладная;

6. Проверено количество отгруженного товара со склада

7. Проверены позиции на складе поступившего товара на склад

8. На тестовых значениях, при помощи программы проверены разработанные алгоритмы.

В ходе тестирования по приведенной выше методике, не выявлено отклонений в работе программы от ожидаемого результата. На основании полученных результатов сделан вывод, что разработанный программный продукт соответствует требованиям технического задания и пригоден к эксплуатации.

**ЗАКЛЮЧЕНИЕ**

В данной курсовой работе, в соответствии с заданием, реализована информационная система для организации оптовой торговли.

В ходе выполнения курсовой работы последовательно были решены следующие задачи:

- анализ исходных данных и формирование требований к программе;

- разработка алгоритмов;

- программная реализация алгоритмов;

- тестирование программного продукта;

- разработка документации.

Разработанная программа обеспечивает формирование приходных и расходных накладных, а также обеспечивает управление:

- сотрудниками;

- контрагентами;

- товарами;

- складом;

- категориями товаров;

Данная программа может быть полезна тем, кто занимается оптовой торговлей. Цель работы достигнута.

### СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Организация торговой деятельности: краткий курс лекций для студентов

3,4 курсов направления подготовки 38.03.06 «Торговое дело» / Сост.: С.А.

Поповская // ФГБОУ ВО «Саратовский ГАУ». – Саратов, 2016. – 59 с.

2. C Sharp // URL: https://ru.wikipedia.org/wiki/C\_Sharp (дата обращения 05.06.2023);

3. Microsoft Visual Studio // URL: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio (дата обращения 05.06.2023);

4. Давыдов В. Г. Программирование и основы алгоритмизации: Учеб. пособие.- М.: Высш. Шк., 2003.-447 с.:ил.

5. Firebird // URL: https://ru.wikipedia.org/wiki/Firebird (дата обращения 05.06.2023);