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 Управление данными

(наименование дисциплины)

Тема Разработка программы "Автомастерская" с базой данных

Руководитель

Ковалев Ю.А.

(оценка) (фамилия, инициалы)

Члены комиссии

(подпись) (дата)

Студент ИСз-121

(подпись) (Ф.И.О.) (группа)

Догадин И.А.

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

(подпись) (дата)

Муром 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc135751522)

[1. Анализ технического задания 6](#_Toc135751523)

[1.1 Описание приложения 6](#_Toc135751524)

[1.2 Главные цели приложения 6](#_Toc135751525)

[1.3 Функции приложения 7](#_Toc135751526)

[1.4 Выбор технологий 8](#_Toc135751527)

[1.5 Анализ предметной области 9](#_Toc135751528)

[2. Проектирование и реализация базы данных 11](#_Toc135751529)

[2.1 Проектирование сущностей базы данных 11](#_Toc135751530)

[2.2 Реализация сущностей, атрибутов и связей 14](#_Toc135751531)

[2.3 Создание ER-диаграммы 22](#_Toc135751532)

[3. Разработка приложения 23](#_Toc135751533)

[4. Тестирование приложения 40](#_Toc135751534)

[ЗАКЛЮЧЕНИЕ 44](#_Toc135751535)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46](#_Toc135751536)

[ПРИЛОЖЕНИЕ А SQL-КОД ДЛЯ ГЕНЕРАЦИИ БАЗЫ ДАННЫХ 47](#_Toc135751537)

### ВВЕДЕНИЕ

В настоящее время, в связи с возросшими потоками информации, которую необходимо обработать для принятия верного решения, разрабатываются специальные программные продукты, позволяющие представить информацию в более наглядной для оператора форме. К подобным программным продуктам относятся базы данных.

База данных – это совокупность записей различного типа, содержащая перекрестные ссылки. Базы данных не помогают пользователю в принятии решения, они только предоставляют всю полноту информации необходимую для принятия этого решения, а вся ответственность за результат применения этого решения возлагается на оператора.

Широкое применение базы данных нашли в областях формирования различного рода документации, где необходимо не только создать бумажную копию некоторого документа, но и сохранить ее для возможного обращения в будущем. Типовая информационная система ориентирована на сбор и обработку информации и включает в себя базу данных и средства работы с ней.

Внедрение информационных технологий на любой фирме обуславливает не только эффективность процессов, но также облегчает всю работу большим количеством данных.

В данном курсовом проекте необходимо разработать информационную систему для организации оптовой торговли. Разработанный программный продукт может быть полезен для применения в фирмах, которые занимаются оптовой торговлей.

1. Анализ технического задания

1.1 Описание автоматизируемой области

В данном курсовом проекте необходимо разработать информационную систему для организации оптовой торговли.

Ниже описан процесс работы фирмы, которая занимается оптовой торговлей.

При поступлении товара оператор (работник слада) принимает его на склад. Оформляя товар формируется приходная накладная, в которой указывается:

- контрагент (фирма поставщик);

- сотрудник склада;

В накладной указывается список товаров, которые поступили на склад. У каждого товара указывается:

- название;

- количество;

- место на складе (где товар будет хранится).

В приходной накладной указывается общая сумма за товар, поступивший на склад.

При отгрузке товара со склада, работник склада формирует расходную накладную. В расходной накладной указывается:

- фирма которой отгружается товар;

- сотрудник склада.

В табличной части расходной накладной указывается список товаром отгруженных со склада, а также общая сумма по накладной.

1.2 Формирование требований к системе

Исходя из приведенного выше анализа предметной области, сформулированы следующие основные требования к разрабатываемой базе данных:

- регистрация сотрудников фирмы, которые будут заниматься оформлением приходных и расходных накладных;

- регистрация контрагентов с которыми будет работать фирма;

- организовать управление складом, в базе данных должны быть записи о местах хранения товаров;

- в базе данных нужно хранить записи о товарах, с которыми работает фирма;

- товары в базе данных нужно разделить на группы.

- централизованное хранение информации на сервере базы данных для обеспечения возможности ввода и данных различными пользователями (работниками склада, менеджерами и т. д.).

Система должна удовлетворять следующим требованиям надежности работы:

- система должна корректно обрабатывать ошибки ввода данных, допущенные оператором;

- при возникновении ошибок, система должна сообщать пользователю описание ошибки и рекомендации по ее устранению (если возможно);

- система должна корректно обрабатывать ситуации недоступности сервера базы данных по локальной сети, и восстанавливать полную работоспособность при восстановлении доступности базы данных.

2. Проектирование базы данных

Для проектирования и реализации информационной системы в курсовом проекте использовалось программное обеспечение IBExpert. Данное программное обеспечение является бесплатным и имеет графический интерфейс, что для проектирования базы данных очень важно. В IBExpert можно сгенерировать ER-диаграмму, которая будет графически отражать базу данных.

Процесс построения информационной модели состоит из следующих шагов:

- определение сущностей;

- определение зависимостей между сущностями;

- задание первичных и альтернативных ключей;

- определение атрибутов сущностей;

- приведение модели к требуемому уровню нормальной формы.

На основании анализа предметной области, в модели базы данных выделены следующие основные сущности:

- сотрудник;

- контрагент;

- склад;

- товар;

- категория.

- приходная накладная;

- расходная накладная;

При проектировании базы данных сущности связываются один ко многим.

Следующим этапом является проектирование физической модели базы данных.

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Для реализации поставленной задачи, используется СУБД Firebird. Данная СУБД является реляционной, следовательно, физическая модель данных будет построена по следующим правилам:

- каждой сущности соответствует таблица физической модели;

- каждому атрибуту сущностей соответствует атрибут таблицы физической модели.

- реляционные отношения преобразуются в ограничения целостности данных.

Таблица 1 Структура таблицы CATEGORY (Категория)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| NAME | VARCHAR |  |  |

Таблица 2 Структура таблицы CONTRACTOR (Контрагент)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| NAME | VARCHAR |  |  |
| ACTUAL\_ADDRESS | VARCHAR |  |  |
| LEGAL\_ADDRESS | VARCHAR |  |  |
| PHONE | VARCHAR |  |  |

Таблица 3 Структура таблицы EMPLOYEE (Сотрудник)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| FIO | VARCHAR |  |  |
| POST | VARCHAR |  |  |

Таблица 4 Структура таблицы WAREHOUSE (Склад)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| LINE | INTEGER |  |  |
| SHELF | INTEGER |  |  |
| PLACE | INTEGER |  |  |

Таблица 5 Структура таблицы PRODUCT (Товар)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| NAME | VARCHAR |  |  |
| TRADEMARK | VARCHAR |  |  |
| ID\_CATEGORY | INTEGER |  | + |
| PRICE | DECIMAL |  |  |
| UNIT | VARCHAR |  |  |
| AMOUNT | INTEGER |  |  |
| MIN\_STOCK | INTEGER |  |  |
| WANT\_STOCK | INTEGER |  |  |
| DESCRIPTION | VARCHAR |  |  |

Таблица 6 Структура таблицы INVOICE\_TYPE (Тип накладной)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| NAME | VARCHAR |  |  |

Таблица 7 Структура таблицы INVOICE\_HEADER (Шапка накладной)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| ID\_CONTRACTOR | INTEGER |  | + |
| ID\_EMPLOYEE | INTEGER |  | + |
| ID\_INVOICE\_TYPE | INTEGER |  | + |
| INVOICE\_DATE | DATE |  |  |

Таблица 8 Структура таблицы INVOICE\_TABLE\_PART (Табличная часть накладной)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Первичный ключ | Внешний ключ |
| ID | INTEGER | + |  |
| ID\_INVOICE\_HEADER | INTEGER |  | + |
| ID\_PRODUCT | INTEGER |  | + |
| AMOUNT | INTEGER |  |  |
| TOTAL\_PRICE | DECIMAL |  |  |
| ID\_WAREHOUSE | INTEGER |  | + |

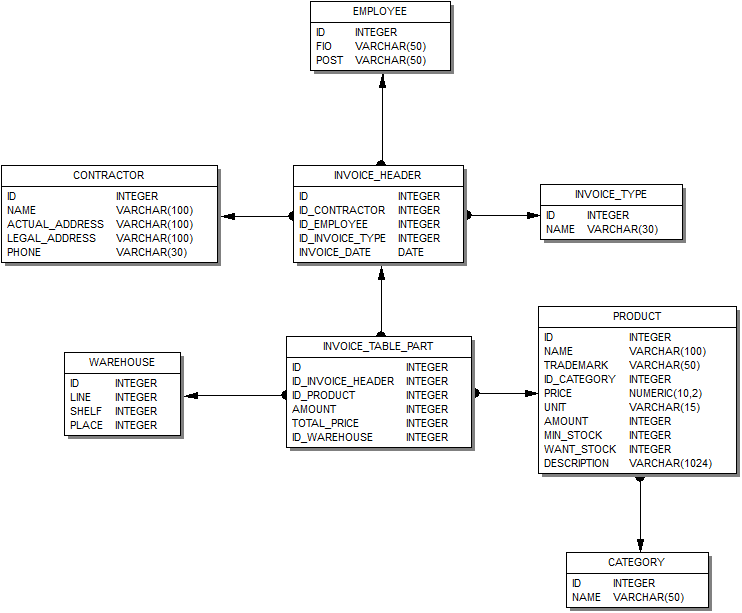


Рисунок 1 - ER-диаграмма

Первичные ключи в базе данных генерируются внутренними средствами (автоинкрементном). Для таблицы INVOICE\_HEADER используется генератор и триггер (перед вставкой записи в базу данных), для того чтобы получить идентификатор последней вставленной записи.

3. Разработка базы данных

База данных реализована на СУБД Firebird версии 4.0. Для управления СУБД и реализации объектов базы данных использовано средство администрирования IBExpert версии 2022.

Таблицы базы данных созданы в соответствии с разработанной ранее физической моделью данных. Ниже приведены SQL скрипты для создания таблиц и связей между ними.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Generated by IBExpert 2022.3.4.1 31.05.2023 18:47:59 \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SET SQL DIALECT 3;

SET NAMES UTF8;

CREATE DATABASE 'C:\Users\admin\source\repos\Wholesale\db\WHOLESALE.FDB'

USER 'SYSDBA' PASSWORD 'masterkey'

PAGE\_SIZE 16384

DEFAULT CHARACTER SET UTF8 COLLATION UTF8;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Generators \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE GENERATOR GEN\_INVOICE\_HEADER\_ID START WITH 1 INCREMENT BY 1;

SET GENERATOR GEN\_INVOICE\_HEADER\_ID TO 127;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Tables \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE CATEGORY (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

NAME VARCHAR(50)

);

CREATE TABLE CONTRACTOR (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

NAME VARCHAR(100),

ACTUAL\_ADDRESS VARCHAR(100),

LEGAL\_ADDRESS VARCHAR(100),

PHONE VARCHAR(30)

);

CREATE TABLE EMPLOYEE (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

FIO VARCHAR(50),

POST VARCHAR(50)

);

CREATE TABLE INVOICE\_HEADER (

ID INTEGER NOT NULL,

ID\_CONTRACTOR INTEGER,

ID\_EMPLOYEE INTEGER,

ID\_INVOICE\_TYPE INTEGER,

INVOICE\_DATE DATE

);

CREATE TABLE INVOICE\_TABLE\_PART (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

ID\_INVOICE\_HEADER INTEGER,

ID\_PRODUCT INTEGER,

AMOUNT INTEGER,

TOTAL\_PRICE DECIMAL(10,2),

ID\_WAREHOUSE INTEGER

);

CREATE TABLE INVOICE\_TYPE (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

NAME VARCHAR(30)

);

CREATE TABLE PRODUCT (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

NAME VARCHAR(100),

TRADEMARK VARCHAR(50),

ID\_CATEGORY INTEGER,

PRICE DECIMAL(10,2),

UNIT VARCHAR(15),

AMOUNT INTEGER,

MIN\_STOCK INTEGER,

WANT\_STOCK INTEGER,

DESCRIPTION VARCHAR(1024)

);

CREATE TABLE WAREHOUSE (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY,

LINE INTEGER,

SHELF INTEGER,

PLACE INTEGER

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Autoincrement generators \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ALTER TABLE CATEGORY ALTER ID RESTART WITH 3;

ALTER TABLE CONTRACTOR ALTER ID RESTART WITH 11;

ALTER TABLE EMPLOYEE ALTER ID RESTART WITH 19;

ALTER TABLE INVOICE\_TABLE\_PART ALTER ID RESTART WITH 28;

ALTER TABLE INVOICE\_TYPE ALTER ID RESTART WITH 2;

ALTER TABLE PRODUCT ALTER ID RESTART WITH 32;

ALTER TABLE WAREHOUSE ALTER ID RESTART WITH 20;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Primary keys \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ALTER TABLE CATEGORY ADD CONSTRAINT PK\_CATEGORY PRIMARY KEY (ID);

ALTER TABLE CONTRACTOR ADD CONSTRAINT PK\_CONTRACTOR PRIMARY KEY (ID);

ALTER TABLE EMPLOYEE ADD CONSTRAINT PK\_EMPLOYEE PRIMARY KEY (ID);

ALTER TABLE INVOICE\_HEADER ADD CONSTRAINT PK\_INVOICE\_HEADER PRIMARY KEY (ID);

ALTER TABLE INVOICE\_TABLE\_PART ADD CONSTRAINT PK\_INVOICE\_TABLE\_PART PRIMARY KEY (ID);

ALTER TABLE INVOICE\_TYPE ADD CONSTRAINT PK\_INVOICE\_TYPE PRIMARY KEY (ID);

ALTER TABLE PRODUCT ADD CONSTRAINT PK\_PRODUCT PRIMARY KEY (ID);

ALTER TABLE WAREHOUSE ADD CONSTRAINT PK\_WAREHOUSE PRIMARY KEY (ID);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Foreign keys \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ALTER TABLE INVOICE\_HEADER ADD CONSTRAINT FK\_INVOICE\_HEADER\_1 FOREIGN KEY (ID\_CONTRACTOR) REFERENCES CONTRACTOR (ID);

ALTER TABLE INVOICE\_HEADER ADD CONSTRAINT FK\_INVOICE\_HEADER\_2 FOREIGN KEY (ID\_EMPLOYEE) REFERENCES EMPLOYEE (ID);

ALTER TABLE INVOICE\_HEADER ADD CONSTRAINT FK\_INVOICE\_HEADER\_3 FOREIGN KEY (ID\_INVOICE\_TYPE) REFERENCES INVOICE\_TYPE (ID);

ALTER TABLE INVOICE\_TABLE\_PART ADD CONSTRAINT FK\_INVOICE\_TABLE\_PART\_1 FOREIGN KEY (ID\_INVOICE\_HEADER) REFERENCES INVOICE\_HEADER (ID);

ALTER TABLE INVOICE\_TABLE\_PART ADD CONSTRAINT FK\_INVOICE\_TABLE\_PART\_2 FOREIGN KEY (ID\_PRODUCT) REFERENCES PRODUCT (ID);

ALTER TABLE INVOICE\_TABLE\_PART ADD CONSTRAINT FK\_INVOICE\_TABLE\_PART\_3 FOREIGN KEY (ID\_WAREHOUSE) REFERENCES WAREHOUSE (ID);

ALTER TABLE PRODUCT ADD CONSTRAINT FK\_PRODUCT\_2 FOREIGN KEY (ID\_CATEGORY) REFERENCES CATEGORY (ID);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Triggers \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SET TERM ^ ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Triggers for tables \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Trigger: INVOICE\_HEADER\_BI \*/

CREATE TRIGGER INVOICE\_HEADER\_BI FOR INVOICE\_HEADER

ACTIVE BEFORE INSERT POSITION 0

as

begin

if (new.id is null) then

new.id = gen\_id(gen\_invoice\_header\_id,1);

end

^

SET TERM ; ^

4. Разработка клиентского приложения

4.1 Реализация работы с данными

Программа для работы с базой данных реализована на платформе .NET Framework версии 4.8. Среда разработки – Microsoft Visual Studio 2022, язык программирования – C#. Для разработки проекта использована Embedded версия Firebird 4.0, которая подключена к проекту и не требует установки полноценной версии СУБД Firebird.

Работа с базой данных из проекта реализована с помощью FirebirdSql.Data.FirebirdClient, с помощью которого можно делать запросы непосредственно из проекта.

4.2 Реализация логики приложения и пользовательского интерфейса

Клиентское программное обеспечение построено на платформе .NET Framework 4.8. Проект реализован как приложение WindowsForms. Ниже приведен перечень программных модулей проекта.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

Таблица 9 Структура проекта пользовательского приложения

|  |  |
| --- | --- |
| Имя файла исходного кода | Описание |
| Category.cs | Форма управления справочником "Категория" |
| CategoryEdit.cs | Диалоговое окно формы "Категория" |
| Contractor.cs | Форма управления справочником "Контрагент" |
| ContractorEdit.cs | Диалоговое окно формы "Контрагент" |
| Employee.cs | Форма управления справочником "Сотрудник" |
| EmployeeEdit.cs | Диалоговое окно формы "Сотрудник" |
| Invoices.cs | Форма управления журналом "Накладная" |
| InvoicesEdit.cs | Диалоговое окно формы "Накладная" |
| Main.cs | Главная форма |
| Product.cs | Форма управления справочником "Товар" |
| ProductEdit.cs | Диалоговое окно формы "Товар" |
| Program.cs | Точка входа в приложение |
| Warehouse.cs | Форма управления справочником "Склад" |
| WarehouseEdit.cs | Диалоговое окно формы "Склад" |

Подключение к базе данных происходит при загрузке главной формы приложения и все параметры подключения указаны в строке подключения

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

{

fb\_cons = new FbConnectionStringBuilder();

fb\_cons.Charset = "UTF8";

fb\_cons.ClientLibrary = ".\\fbclient.dll";

fb\_cons.UserID = "SYSDBA";

fb\_cons.Password = "masterkey";

fb\_cons.Dialect = 3;

fb\_cons.Database = "..\\..\\db\\WHOLESALE.FDB";

fb\_cons.ServerType = FbServerType.Embedded;

//ЕСЛИ РАЗВОРАЧИВАТЬ НА ПОЛНОЦЕННОМ СЕРВЕРЕ

//fb\_cons.ServerType = 0;

//fb\_cons.DataSource = "localhost";

//fb\_cons.Port = 3050;

Program.AddUpdateAppSettings("ConnectionString", fb\_cons.ToString());

try

{

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

fbCon.Open();

fbCon.Close();

}

catch (Exception x)

{

MessageBox.Show(x.Message);

}

finally

{

fbCon.Close();

}

}

Для тестирования подключения происходит подключение к базе и отключения. Настройки подключения сохраняются в ConnectionString, чтобы потом можно было обратиться к ним из любой точки приложения.

Работа пользователя с приложением построена в диалоговом режиме. Для выполнения команд пользователя, отображается соответствующее модальное диалоговое окно, в котором реализован определенный функционал. Логика работы программы после закрытия модального окна может быть различной, в зависимости от результата выполнения операции в модальном окне. Для этого производится анализ возвращаемого результата диалогового окна.

5. Тестирование программного продукта

Тестирование разработанного программного продукта выполняется с целью проверки соответствия реализованного функционала требованиям технического задания, а также проверки надежности работы и отсутствия ошибок в реализации функционала. Ниже приведена методика тестирования программного продукта.

Таблица 10 Методика тестирования программного продукта

|  |  |
| --- | --- |
| Выполняемая функция | Требуемый результат |
| 1. Запустить файл Wholesale.exe | Отобразится главное окно приложения |
| 2. На главном окне нажать на кнопку СОТРУДНИКИ | Отобразится форма справочника сотрудники |
| 3. На форме СОТРУДНИКИ нажать на кнопку ДОБАВИТЬ | Отобразится модальное окно формы СОТРУДНИКИ. |
| 4. Заполнить данные о сотруднике и нажать кнопку ДОБАВИТЬ | Запись добавиться в базу данных |
| 5. На форме сотрудники нажать на кнопку ИЗМЕНИТЬ | Появится модальное окно формы СОТРУДНИКИ |
| 6. Изменить данные о сотруднике и нажать кнопку применить | Измененная запись добавится в базу данных |
| 7. На форме сотрудники нажать на кнопку УДАЛИТЬ | Выделенная запись удалится из базы данных |
| 8. Повторить шаги 2 - 7 для справочника КОНТРАГЕНТЫ |  |
| 9. Повторить шаги 2 - 7 для справочника СКЛАД |  |
| 10. Повторить шаги 2 - 7 для справочника ТОВАРЫ |  |
| 11. Повторить шаги 2 - 7 для справочника КАТЕГОРИИ ТОВАРОВ |  |
| 12. Нажать на кнопку ТОВАРНЫЕ НАКЛАДНЫЕ | Отобразиться форма с журналом накладных |
| 13. Нажать на кнопку добавить | Отобразиться форма добавления накладной |
| 14. Заполнить накладную данными и нажать на кнопку ДОБАВИТЬ | Запись о накладной добавиться с базу данных |
| 15. Выделить накладную и нажать на кнопку ПРОСМОТР | Отобразиться форма с данными о накладной |

Результаты, полученные в ходе тестирования, позволяют сделать заключение в том, что реализация программного продукта соответствует требованиям технического задания.

6. Руководство по программному продукту

6.1 Руководство пользователя

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

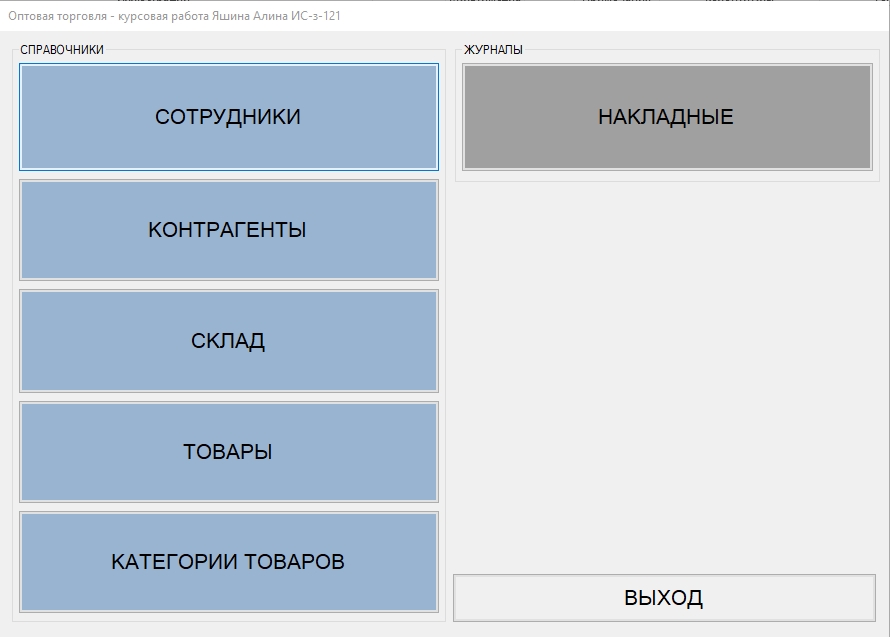


Рисунок 2 - Главное окно приложения

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

В качестве примера рассмотрим работу со справочником Сотрудники.

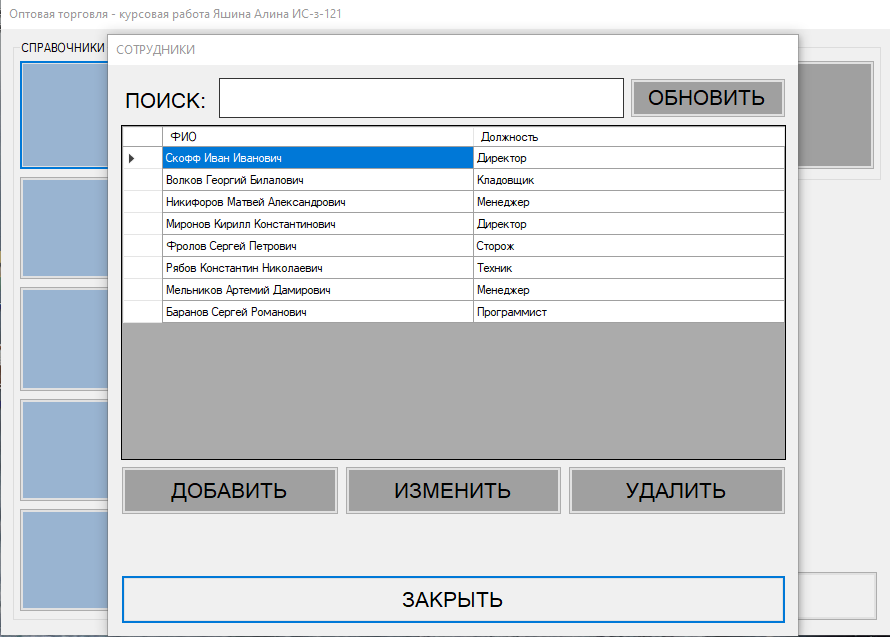


Рисунок 3 - Список сотрудников

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

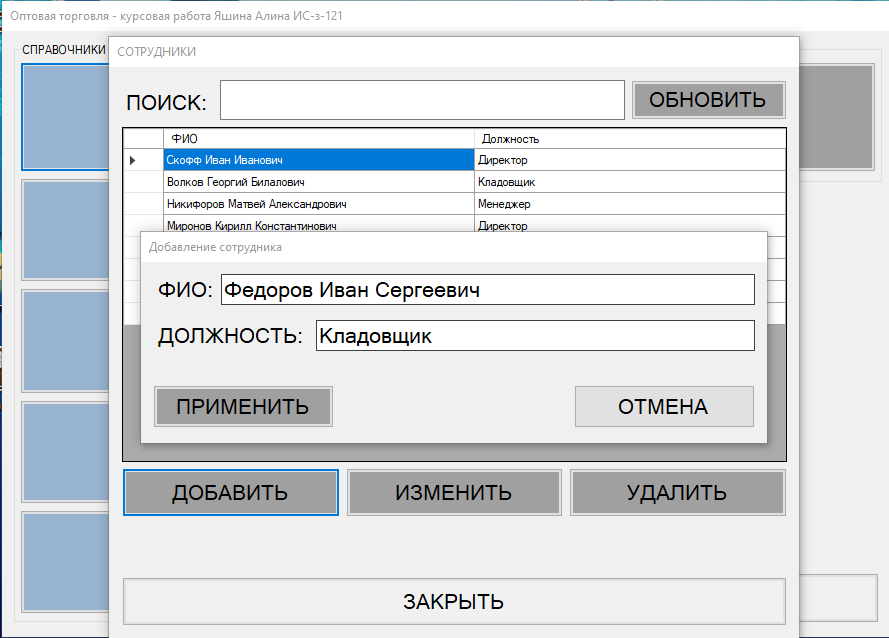


Рисунок 4 - Добавление записи в справочник Сотрудники

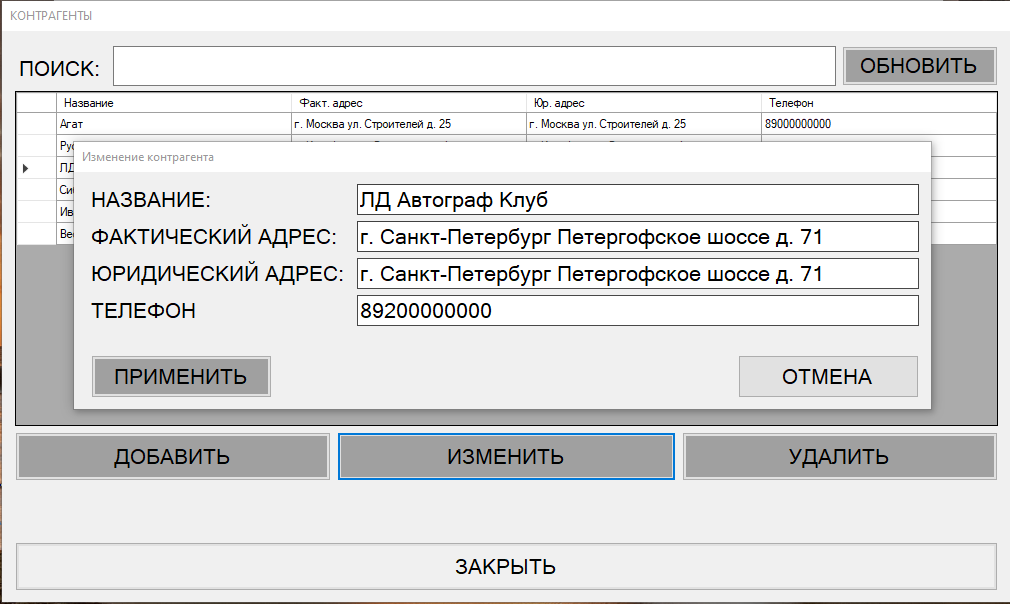


Рисунок 5 - Изменение записи в справочнике Сотрудники

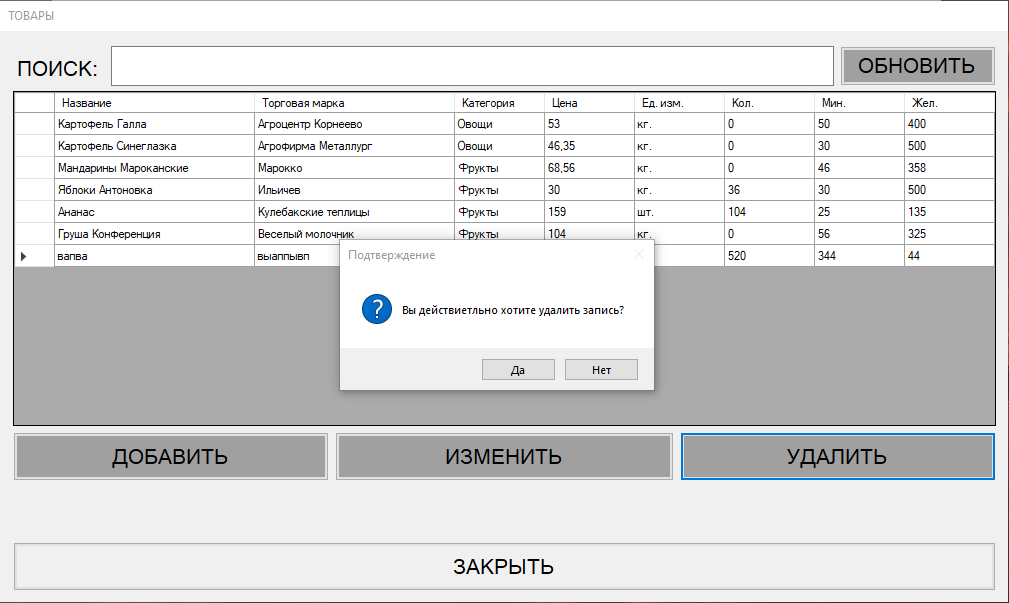


Рисунок 6 - Удаление записи в справочнике Сотрудники

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

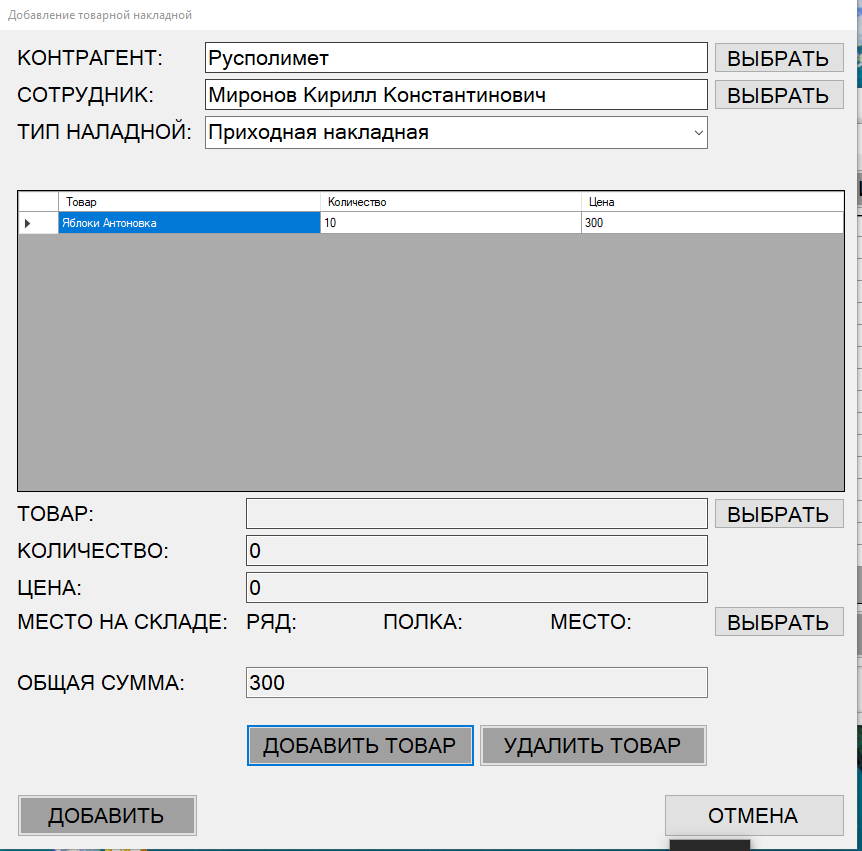


Рисунок 7 - Добавление накладной

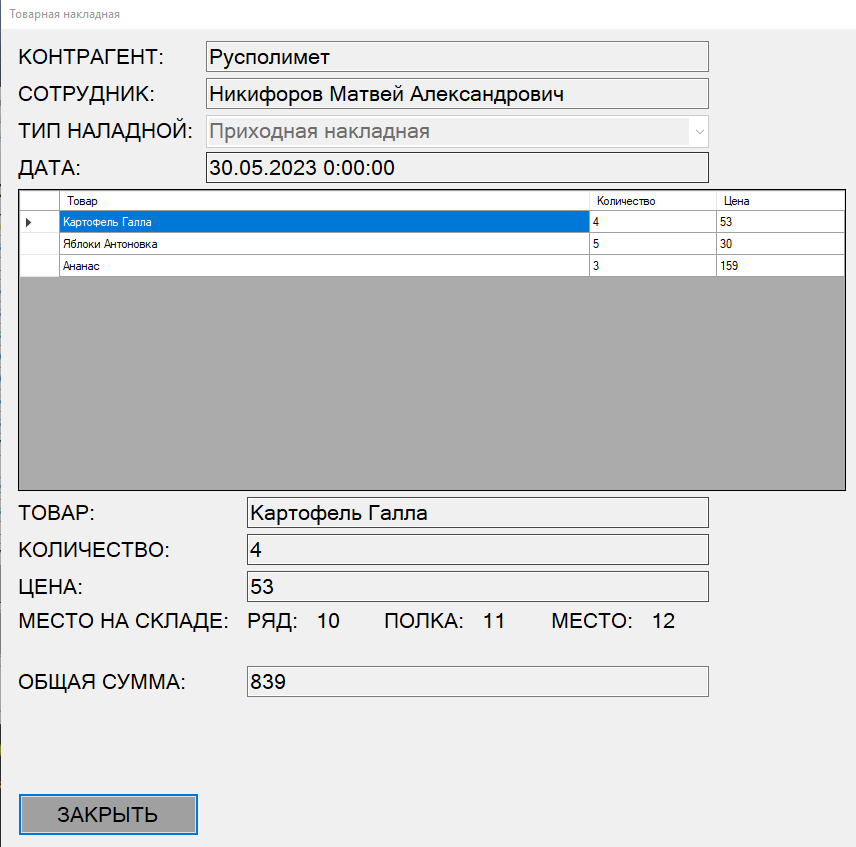


Рисунок 8 - Просмотр сформированной накладной

6.2 Руководство 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.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте в соответствии с заданием разработана информационная система для организации оптовой торговли.

В ходе работы над курсовым проектом, были выполнены следующие задачи:

- анализ технического задания;

- анализ предметной области;

- проектирование базы данных;

- реализация базы данных;

- реализация клиентского программного обеспечения.

Для проектирования базы данных было использовано ПО IBExpert. Реализация базы данных осуществлена с помощью языке SQL-запросов. Разработанная программа позволяет выполнять следующие задачи:

- вести учет сотрудников, контрагентов, склада, категорий товаров, товаров;

- формировать приходные и расходные накладные;

- в информационной системе реализован удобный поиск по базе данных.

Разработанный программный продукт может быть полезен для применения в различных фирмах занимающимися оптовой торговлей.