Содержание

[Введение 3](#_Toc90945578)

[1. Описание предметной области 5](#_Toc90945579)

[1.1 Общие положения 5](#_Toc90945580)

[1.2 Сведения из теории 7](#_Toc90945581)

[1.3 Постановка задачи 11](#_Toc90945582)

[2 Технологии разработки приложения 12](#_Toc90945583)

[2.1 Алгоритм решения 12](#_Toc90945584)

[2.2 Макет приложения 13](#_Toc90945585)

[2.3 Описание программы 17](#_Toc90945586)

[2.4 Результаты работы программы 20](#_Toc90945587)

[3. Руководство пользователя 23](#_Toc90945588)

[Заключение 27](#_Toc90945589)

[Список использованных источников 28](#_Toc90945590)

# Введение

Автоматизация позволяет частично или полностью освободить человека от исполнения циклических процессов, или процессов, выполняющихся по строго заданному алгоритму. Развитие автоматика получила благодаря промышленно-техническому прогрессу. Даже автоматизация в быту берёт своё начало на промышленных производствах, где стремление к ускорению процесса и, соответственно, увеличение выручки способствовало внедрению новейших на тот момент наукоёмких средств автоматизации. Так же автоматизированные производства позволяли высвобождать большое количество рабочих рук, оптимизируя время и занятость персонала.

Автоматизация в наше время охватывает практически все сферы человеческой деятельности, от поддержания определённых технологических параметров до контроля работы на предприятии с помощью баз данных.

Внедрение автоматизированной системы управления позволяет:

- упростить работу рабочего персонала;

- предотвратить появление человеческой ошибки;

- повысить эффективность производства;

- увеличить прибыль предприятия;

- уменьшить время и точность обработки данных.

Системы управления металлургического завода призваны сократить издержки производства - временные и финансовые, облегчить управление бизнес процессами, включая взаимодействие с клиентами, поставщиками, автоматизировать ведение банковских счетов, заказов, платёжных требований и товаров. Компьютерная программа для металлургического завода должна быть очень гибкой и позволять вести учет в различных разрезах предприятия, как, например: учёт всех товаров на складе, учёт всех клиентов, учёт всех накладных и т. д.

Целью выполнения курсового проекта является развитие навыков реализации сетевых приложений с использованием прикладного интерфейса программирования сокетов; средой разработки прикладных сетевых приложений; получение опыта разработки сетевых программных средств. В ходе выполнения курсового проекта должно формироваться представление об этапах проектирования и создания сетевых программных средств. В ходе достижения цели решаются следующие задачи:

- развитие логического и алгоритмического мышления;

- выработка умения формулировать суждения и выводы, логически последовательно и доказательно их излагать;

- овладение навыками самостоятельной исследовательской работы;

- закрепление использования технологией сетевого программирования на стороне клиента и сервера.

1 Описание предметной области

1.1 Общие положения

Имя собственный мебельный склад, руководитель рано или поздно сталкивается с трудностью учёта деятельности своей компании.

Жёсткие условия конкуренции на рынке поставки металлических изделий предъявляют повышенные требования к управлению предприятием, осуществляющем учёт продукции. Чрезвычайно важно учесть три фактора:

- Формирование быстрого учёта товара;

- Достижение высокого уровня качества обслуживания клиентов;

- Быстрое реагирования на изменения на рынке.

Особенностью системы учёта для данной отрасли, является то, что основным объектом учёта являются заказы клиентов, в соответствии с которыми определяются и накладные, и платёжные требования.

Задачей выполняемой металлургическим заводом является приём заказов и формирование по ним договоров и накладных на печать. Для этого требуется иметь информацию о клиентах, спецификаций, товаров.

Целью курсовой работы является создание базы данных с помощью MS SQL Server и C#. MS SQL Server ˗ инструмент для визуального проектирования баз данных, также для администрирования и разработки. С# ‑ платформа обладает следующими преимуществами:

- переносимость, или кросс-платформенность;

- объектная ориентированность, создана эффективная объектная модель;

- привычный синтаксис С/С++;

- встроенная и прозрачная модель безопасности;

- ориентация на интернет-задачи, сетевые распределенные приложения;

- динамичность, легкость развития и добавления новых возможностей;

- легкость в освоении;

Для автоматизации работы на металлургическом заводе, требуется выводить следующие

запросы:

- Добавление, детали, удаление и редактирование информации.

База данных должна выводить информацию:

- о заказчиках;

- о водителях;

- о топливе;

- о заказах;

- о машинах;

База данных создаётся для решения следующих задач:

- структурировать информацию о заказах;

- структурировать информацию о заказчиках.

Задачами данной курсовой работы являются:

- создание таблиц в базе данных;

- создание связей между таблицами в базе данных;

- создание запросов для в MS SQL Server;

- освоение паттерна Model View Controller

- освоение навыков программирование в среде C#;

- освоение технологии Entity Framework Core;

- освоение системы авторизации и аутентификации;

- реализация доступа клиента к серверу СУБД.

1.2 Сведения из теории

Так как сетевое приложение получает и использует большое количество данных, без систематизации данные не будет представлять какой-либо ценности. Поэтому в настоящее время системы управления базами данных (СУБД) используются для систематизации больших массивов данных, облегчения поиска по базе и внесения в них изменений. В СУБД поддерживаются различные модели данных.

Модель данных — это метод (принцип) логической организации данных, используемый СУБД. Наиболее известными являются иерархическая, сетевая и реляционная модели. Идея базы данных основана на таком привычном и простом способе записи информации - таблица. В самом деле, четкая и несложная структура (строки и столбцы) делает таблицу практически универсальным средством отображения информации любой природы.

C таблицей легко работать: добавлять новые данные; вычеркивать соответствующие строки, если информация уже не нужна; располагать строки в определенном порядке (например, по алфавиту, либо по коду) и т.д. Данные в таблицах могут иметь совершенно различную природу. Поля базы данных могут иметь один таких типов как: символьный (Char); числовые; дата (Date); логический (BOOL) и другие. В СУБД для персональных компьютеров (настольных СУБД) поддерживается преимущественно реляционная модель, которую отличает простота и единообразие представления данных простейшими двумерными таблицами.

Реляционная модель обеспечивает возможность использования в разных СУБД операций обработки данных, имеющих единую основу - алгебру отношений (реляционную алгебру), и универсального языка структурированных запросов SQL (Sequential Query Language). Реляционная база данных представляет собой множество взаимосвязанных двумерных таблиц - реляционных таблиц, называемых также отношениями, в каждой из которых содержатся сведения об одной сущности автоматизируемой предметной области.

Логическую структуру реляционной базы данных образует совокупность реляционных таблиц, между которыми установлены связи.

Реляционная таблица — это таблица, специальным образом организованная в виде файла. При переходе от таблицы к реляционной меняется терминология. Строка называется записью, Каждая запись состоит из полей (элементов столбцов). Для создания таблиц, соответствующих реляционной модели данных, используется процесс, называемый нормализацией данных, Нормализация – это удаление из таблиц повторяющихся данных путем их переноса в новые таблицы, записи которых не содержат повторяющихся значений.

Структура реляционной таблицы определяется составом полей. Каждое поле определяет определённую характеристику сущности. Таблицы состоят из строк и столбцов. Содержимое поля указывается в столбце таблицы, в которых сохраняются свойства объектов, которые отражены в таблице. Столбец содержит записи одного типа. Содержание таблицы заключается в её строках, однотипных по структуре, Каждая строка содержит данные о конкретном экземпляре сущности и называется записью.

После того, как выбран набор таблиц, необходимый для создания базы данных, определены поля таблиц, следует выбрать уникальные поля. Уникальные поля - это такие поля, в которых значения не могут повторяться, Поле, по значению которого можно идентифицировать запись в таблице, называется первичным ключом.

B нормализованной реляционной базе данных связь двух таблиц характеризуется отношениями записей типа один-к-одному, один-ко-многим, многие-к-многим. Отношение один-к-одному предполагает, что каждой записи одной таблицы соответствует одна запись в другой. Отношение один-ко-многим предполагает, что каждой записи первой таблицы соответствует много записей другой таблицы, но каждой записи второй таблицы соответствует только одна запись в первой. Соответственно, связь многие-к-многим - это связь многих записей одной таблицы ко многим записям другой.

Для двух таблиц, находящихся в отношении один-ко-многим, устанавливается связь по уникальному ключу таблицы, представляющей в отношении сторону «один», - главной таблицы в связи. Во второй таблице, представляющей в отношении сторону «многие» и называемой подчинённой, этот ключ связи может быть либо частью уникального ключа, либо не входить в состав ключа. В подчинённой таблице ключ связи называется ещё внешним ключом.

Запросы используются для выборки из базы данных информации, необходимой для пользователя. Под выборкой будем понимать динамическую таблицу с записями данных, которые удовлетворяют соответствующим условиям. Запросы обеспечивают быстрый и эффективный доступ к данным, хранящимся в таблице. Благодаря запросам можно не только выполнить сортировку или вычислить выражение, но и свести данные из связанных таблиц. Запросы не хранятся базой данных, информация храниться лишь в таблицах. Но полученная в результате запроса таблица может использоваться в качестве источника данных в формах отчётах и других запросах. Можно ввести поправки в запрос, которые отобразятся в соответствующей таблице. Целью данной базы данных является более быстрая и качественная обработка данных из таблиц.

База данных мебельного склада включает в себя четыре таблицы: товары, клиенты, спецификации,договора. Таблицы связаны между собой, что обеспечивает целостность данных. База данных предназначена для сотрудников мебельного склада для контроля за заказами, товарами и платежами клиентов.

В таблице «User» содержится информация о личных данных клиентов. В таблице описаны следующие поля:

- Идентификатор клиента (ключевое поле) - целочисленный тип данных;

- ФИО клиента - текстовый тип данных (переменная длина текстового поля 40 символов);

В таблице «Product» содержится информация о товарах. В таблице описаны следующие поля:

- Идентификатор товара (ключевое поле) - целочисленный тип данных;

- Наименование товара - текстовый тип данных (переменная длина текстового поля 20 символов);

- Цена товара - тип данных с плавающей точкой;

- Количество в наличии - целочисленный тип данных;

- Материал - текстовый тип данных (переменная длина текстового поля 20 символов);

В таблице «Specification» содержится информация о спецификации. В таблице описаны следующие поля:

- Идентификатор спецификации (ключевое поле) - целочисленный тип данных;

- Идентификатор товара - целочисленный тип данных;

- Идентификатор клиента - целочисленный тип данных;

- Дата доставки – простой тип;

В таблице «Договор» содержится информация о договоре. В таблице описаны следующие поля:

- Идентификатор договора - целочисленный тип данных;

- Идентификатор спецификации (ключевое поле) - целочисленный тип данных;

- Идентификатор клиента (ключевое поле) - целочисленный тип данных;

- Поставщик - текстовый тип данных (переменная длина текстового поля 20 символов);

- Количество - тип данных с плавающей точкой;

- Итоговая сумма - тип данных с плавающей точкой;

Реляционная модель топливного гаража, описанного выше, представлена на рисунке 1.1

Diagram

Description automatically generated

Рисунок 1.1 – Реляционная модель топливного гаража

Реляционная база данных считается эффективной, если все её таблицы находятся как минимум в 3 нормальной форме. После проверки таблиц на соответствие требованиям 1НФ, 2НФ, 3НФ, можно сделать вывод, что все таблицы удовлетворяют требованиям. В них каждая запись есть отдельное независимое утверждение.

1.3 Постановка задачи

База данных хранит данные о товарах (изделиях), которые могут быть поставлены, заказчиках (покупателях, клиентах), спецификации, а также договора поставки.

При отгрузке товара (изделий) выписывается товаротранспортная накладная.

Оплата товаров производится через банк с оформлением платёжного требования.

2 Технологии разработки приложения

2.1 Алгоритм решения

Алгоритм решения поставленных задач прост и для всех подзадач основан на одном и том же принципе:

- выбрать пункт из главного меню;

- нажать на необходимую кнопку (“Добавить”, “Удалить”, “Редактировать” или “Обновить данные”);

- при выборе первых трёх функций вводятся необходимые данные;

- нажать на кнопку “Добавить”, “Удалить” или “Редактировать”;

- вывести результаты.

Например, алгоритм работы формы “Клиенты”:

- нажать кнопку “Обновить данные”, чтобы заполнить таблицу на данной форме.

Добавление нового клиента:

- нажать на кнопку “Добавить…”;

- ввести необходимые данные;

- нажать кнопку “Добавить” для возвращения на основную клиентскую форму с уже введёнными данными.

Редактирование клиента:

- нажать на кнопку “Редактировать..,”;

- ввести необходимые данные;

- нажать кнопку “Редактировать” для возвращения на основную клиентскую форму с уже введёнными данными.

Удаление клиента:

- нажать на кнопку “Удалить…”;

- ввести идентификатор клиента;

- нажать кнопку “Удалить” для возвращения на основную клиентскую форму с уже удалённым клиентом.

2.2 Макет приложения

Веб-приложение на C# включает три слоя UI (User interface), Infrastructure, Domain, состоит из пользовательского интерфейса, инфраструктуры, моделей.

Контроллеры представлены на рисунке 2.1

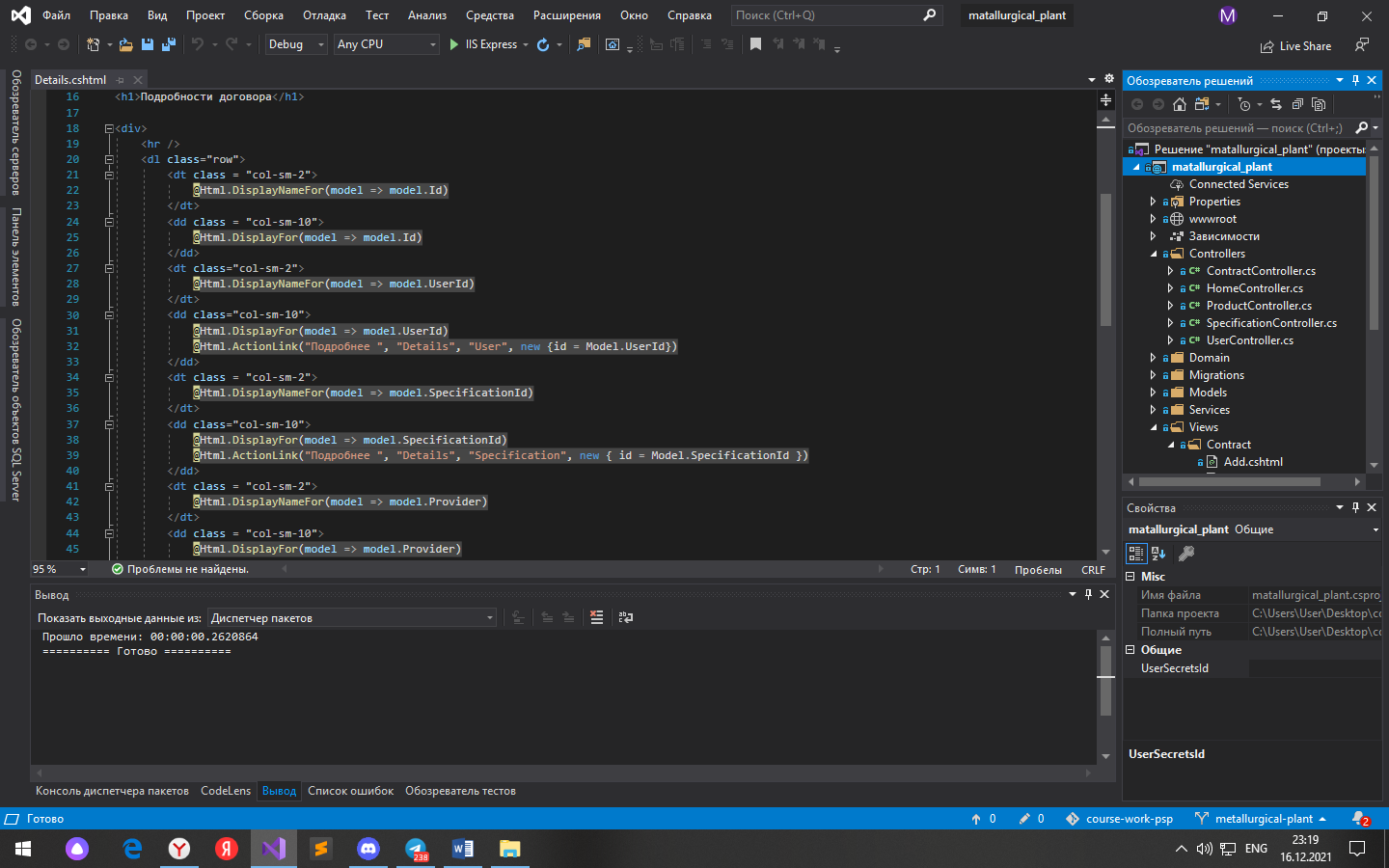


Рисунок 2.1 Контроллеры металлургического завода

Модели представлены на рисунке 2.2

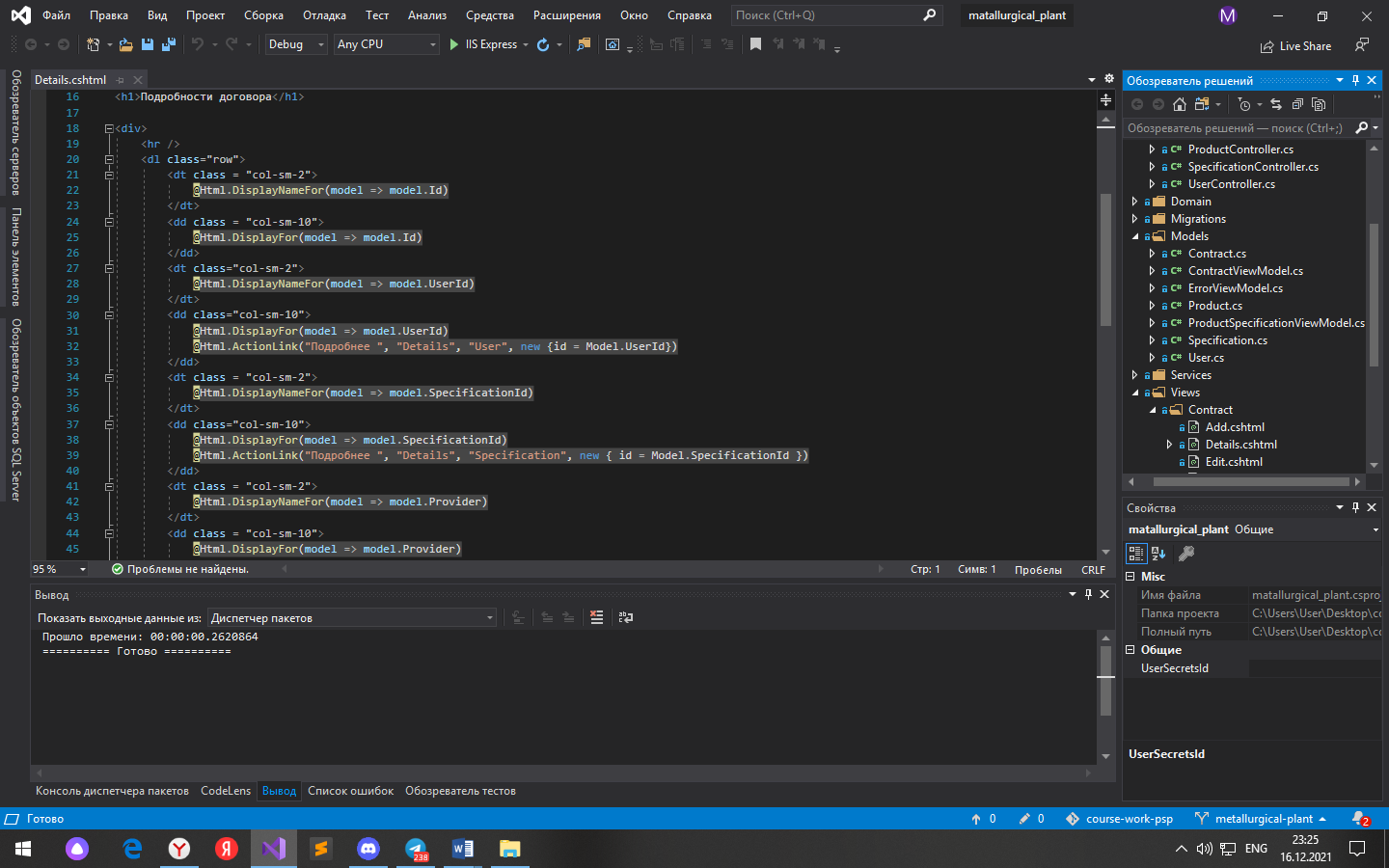


Рисунок 2.2 Модели металлургического завода

Сервисы представлены на рисунке 2.3

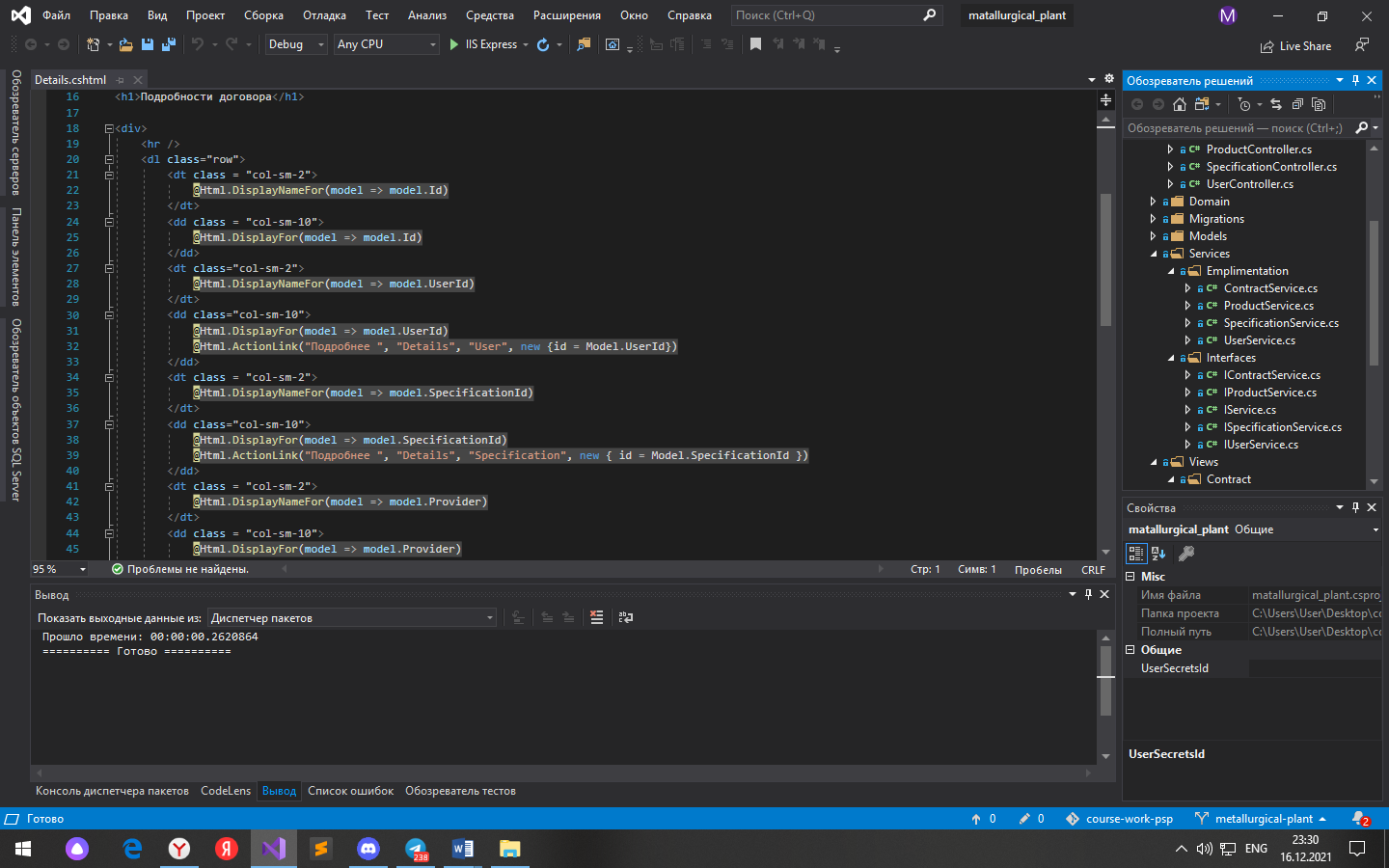


Рисунок 2.3 Сервисы металлургического завода

Интерфейсы представлены на рисунке 2.4

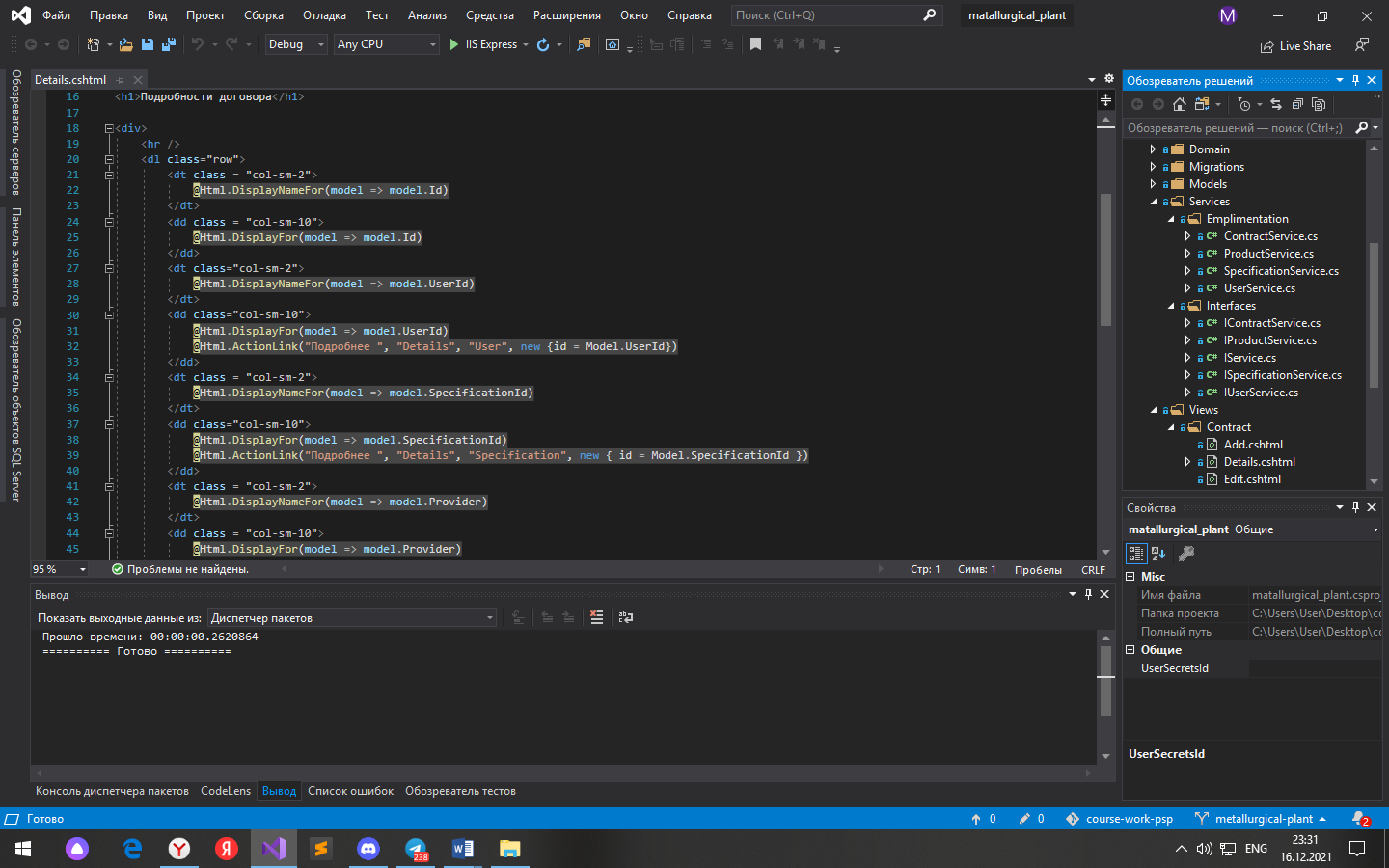


Рисунок 2.4 Интерфейсы металлургического завода

Представления представлены на рисунке 2.5

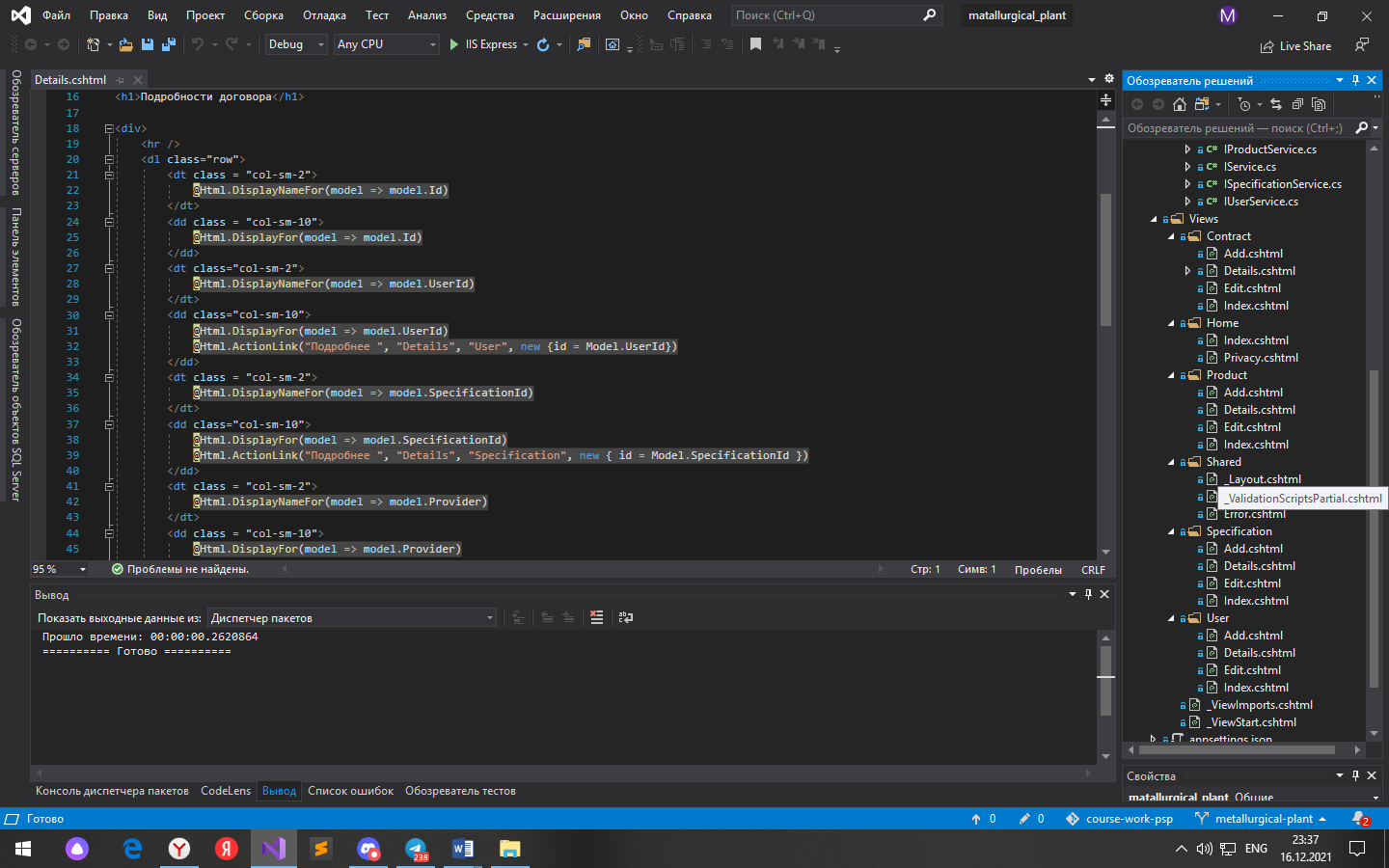


Рисунок 2.5 Представления металлургического завода

Здесь каждая форма отвечает за свою функцию:

- BAC\_Add.java — форма добавления банковского счёта клиента;

- BAC\_Delete.java — форма удаления банковского счёта клиента;

- BAC\_Edit.java — форма редактирования банковского счёта клиента;

- BAS\_Add.java — форма добавления банковского счёта поставщика;

- BAS\_Delete.java — форма удаления банковского счёта поставщика;

- BAS\_Edit.java — форма редактирования банковского счёта поставщика;

- Bank\_accounts.java — форма выбора категории банковских счетов;

- Bank\_acoounts\_clients.java — форма для просмотра банковских счетов клиентов и перехода на формы BAC\_Add.java, BAC\_Delete.java, BAC\_Edit.java

- Bank\_accounts\_suppliers.java — форма для просмотра банковских счетов поставщиков и перехода на формы BAS\_Add.java, BAS\_Delete.java, BAS\_Edit.java

- Banks.java — форма для просмотра всех банков в базе и перехода на формы BanksAdd.java, BanksDelete.java, BanksEdit.java;

- BanksAdd.java — форма добавления банка;

- BanksDelete.java — форма удаления банка;

- BanksEdit.java — форма редактирования банка;

- Clients.java — форма для просмотра всех клиентов в базе и перехода на формы ClientsAdd.java, ClientsDelete.java, ClientsEdit.java;

- ClientsAdd.java — форма добавления клиента;

- ClientsDelete.java — форма удаления клиента;

- ClientsEdit.java — форма редактирования клиента;

- Invoices.java — форма перехода на форму InvoicesView.java;

- InvoicesView.java — форма для просмотра всех автотранспортных накладных по заказам;

- Menu.java — собственно, главное меню программы;

- Orders.java — форма перехода на форму OrdersView.java и другие формы: OrdersAdd.java, OrdersDelete.java, OrdersEdit.java;

- OrdersAdd.java — форма добавления заказа;

- OrdersDelete.java — форма удаления заказа;

- OrdersEdit.java — форма редактирования заказа;

- OrdersView.java — форма для просмотра всех заказов по номерам;

- PayRequirements.java — форма перехода на форму PayRequirementsView.java и другие формы: PayRequirementsAdd.java, PayRequirementsDelete.java;

- PayRequirementsAdd.java — форма добавления платёжного требования;

- PayRequirementsDelete.java — форма удаления платёжного требования;

- PayRequirementsView.java — форма для просмотра всех платёжных требования по номерам заказа;

- Products.java — форма для просмотра всех товаров в базе и перехода на формы ProductsAdd.java, ProductsDelete.java, ProductsEdit.java;

- ProductsAdd.java — форма добавления товара;

- ProductsDelete.java — форма удаления товара;

- ProductsEdit.java — форма редактирования товара;

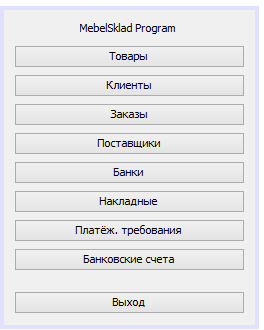
- Suppliers.java — форма для просмотра всех поставщиков в базе и перехода на формы SuppliersAdd.java, SuppliersDelete.java, SuppliersEdit.java;

- SuppliersAdd.java — форма добавления поставщика;

- SuppliersDelete.java — форма удаления поставщика;

- SuppliersEdit.java — форма редактирования поставщика.

Внешний вид программы на примере главного меню:



2.3 Описание программы

В программе используются 36 классов-форм. Все они (не считая главного меню) подключаются к базе данных, получают оттуда информацию, а также добавляют, изменяют или удаляют данные, находящиеся в базе данных. Почти все они однотипны, поэтому рассмотрим для примера класс “Products”.

Для формы Products применяются следующие библиотеки:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

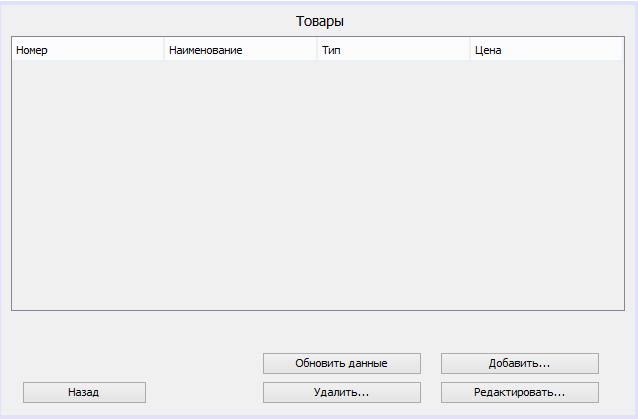
import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.table.DefaultTableModel;

В программе используется технология Swing для создания форм формата JFrame, в которой можно реализовать различные функции при помощи кнопок, таблиц, диалоговых окон и прочего.

Форма Products:



Форма описывается следующей командой:

public class Products extends javax.swing.JFrame

В форме присутствуют кнопки, текстовые поля (в производных формах Добавить, Удалить, Редактировать), таблица метки (label) и другие, которые описываются как:

private javax.swing.JButton jButton1; //Кнопка “Назад”

private javax.swing.JButton jButton2; //Кнопка “Добавить…”

private javax.swing.JButton jButton3; //Кнопка “Редактировать…”

private javax.swing.JButton jButton4; //Кнопка “Удалить…”

private javax.swing.JButton jButton5; //Кнопка “Обновить данные”

private javax.swing.JLabel jLabel1;

private javax.swing.JTable jTable1;

В данной программе используется библиотека JDBC Java DataBase Connectivity.

JDBC - платформенно-независимый стандарт взаимодействия Java-приложений с различными СУБД. JDBC реализован в виде пакета java.sqi, входящего в состав Java SE (Standart Edition). JDBC позволяет устанавливать соединение с базой данных согласно специально описанному URL. Драйверы могут загружаться во время работы программы динамически. Для соединения с базой данных необходимо использовать класс, реализующий интерфейс java.sql.Connection. После соединения с базой данных, можно создавать объекты класса Statement для выполнения запросов к базе данных.

Интерфейс java.sql.ResultSet позволяет извлекать информацию из базы данных.

Соединение с базой данных в форме Products реализуется через кнопку “Обновить данные” (после нажатия которой заполняется таблица на этой же форме) и выглядит следующим образом:

Connection conn=null;

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver").newInstance();

conn = DriverManager.getConnection("jdbc:sqlserver://KIRDIM\\ MSSQLSERVER13:1433;databaseName=Mebelmagaz", "sa", "Freddy");

if(conn!=null)

{

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

model.setRowCount(0);

System.out.println("Database Successfully connected");

Statement stmt = conn.createStatement();

ResultSet resultquery = stmt.executeQuery("SELECT \* FROM [Product]");

int columns = resultquery.getMetaData().getColumnCount();

while (resultquery.next())

{

Object[] row = new Object[columns];

for (int i = 1; i <= columns; i++)

{

row[i-1] = resultquery.getObject(i);

}

((DefaultTableModel) jTable1.getModel()).insertRow(resultquery.getRow() - 1, row);

}

resultquery.close();

stmt.close();

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

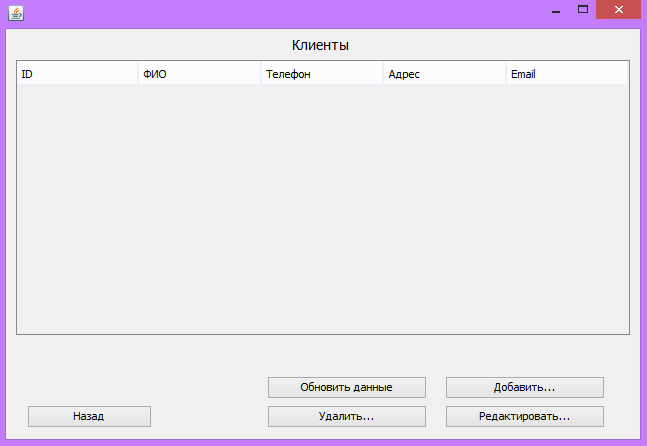
}

Листинг 1 – Код подключения к базе данных

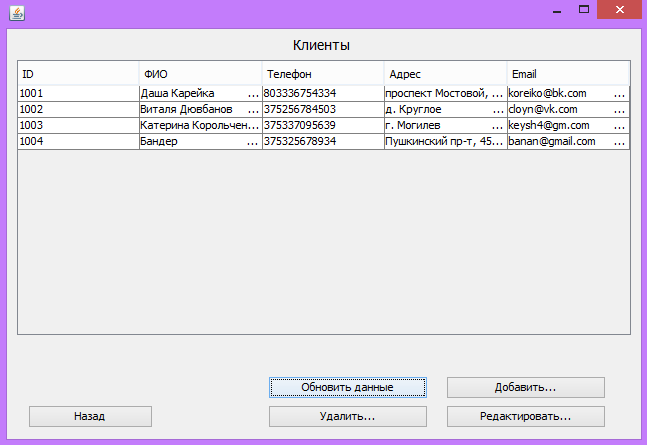
2.4 Результаты работы программы

Чтобы показать результат работы программы, используем несколько основных форм: “Клиенты” и “Заказы”.

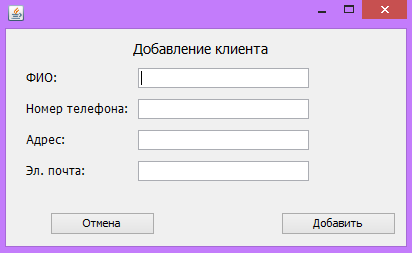
Форма “Клиенты” выглядит следующим образом:



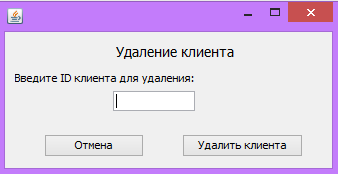
При нажатии кнопки “Обновить данные” таблица заполняется данными из таблицы Client в БД MS SQL Server:



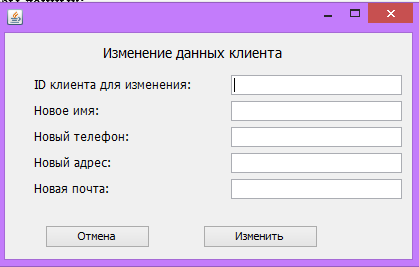
При нажатии кнопки “Добавить…” нас переносит на форму добавления данных, где после ввода всех необходимых данных нужно нажать кнопку “Добавить”, чтобы записать нового клиента в базу данных:



При нажатии кнопки “Удалить…” нас переносит на форму удаления, где после ввода идентификатора клиента необходимо нажать кнопку “Удалить”, чтобы удалить клиента из базы данных:

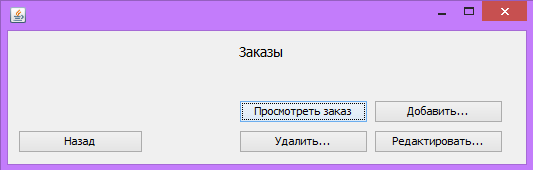


При нажатии кнопки “Редактировать…” нас переносит на форму редактирования, где после ввода идентификатора клиента и всех необходимых данных для изменения нужно нажать кнопку “Изменить”, чтобы изменить данные клиента из базы данных:

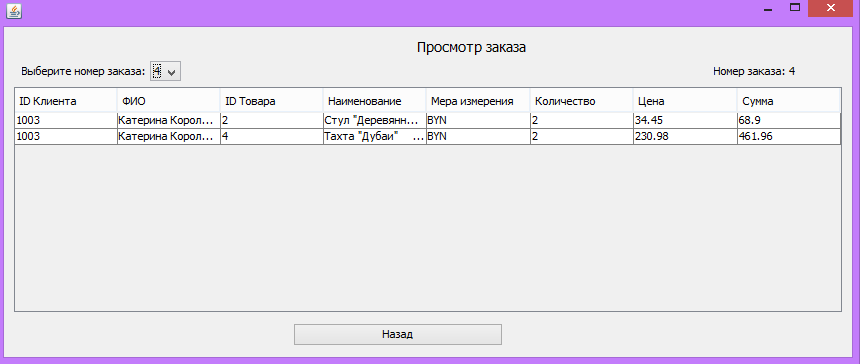


При нажатии кнопки “Назад” нас возвращает в главное меню.

Теперь рассмотрим форму “Заказы”. В ней имеются такие же функции добавления, редактирования и удаления, но есть ещё и кнопка “Просмотреть заказ”.



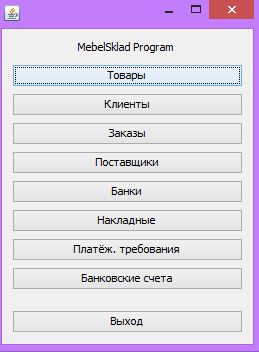
При нажатии на кнопку “Просмотреть заказ” можно при помощи выпадающего списка выбрать необходимый номер заказа и просмотреть всё, что находится в этом заказе:



Все остальные формы работают по такому же одинаковому принципу.

3. Руководство пользователя

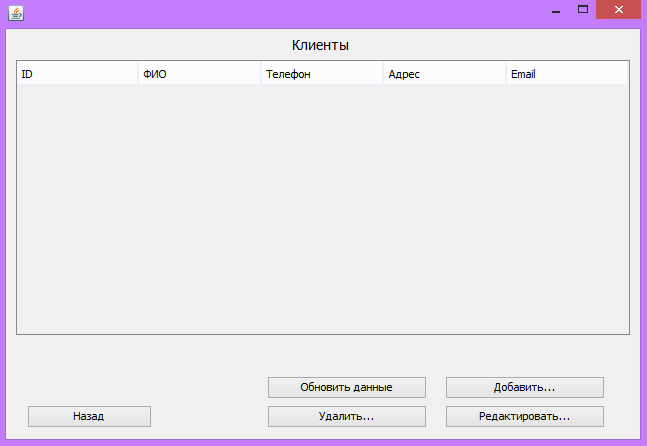
Впервые открыв программу (предварительно подготовив базу данных) пользователь попадает в главное меню программы:



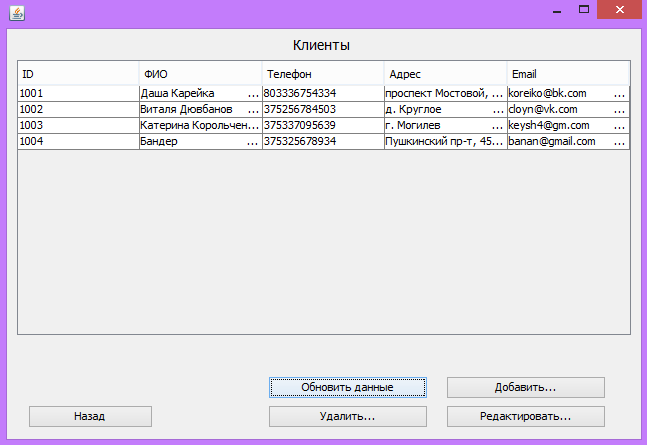
Каждая кнопка ведёт к отдельной одноимённой форме, где можно не только посмотреть уже существующие данные, но и изменить их, удалить или добавить новые.

В формах Товары, Клиенты, Поставщики, Банковские счета и Платёжные требования внешний вид форм, расположение кнопок и функционал одинаковый, поэтому на примере формы Клиенты рассмотрим, как взаимодействовать с программой.

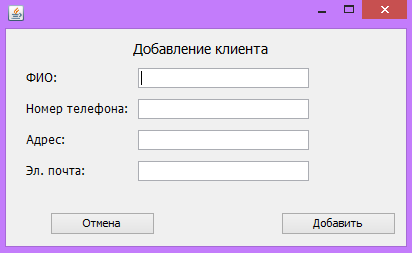
Форма “Клиенты” выглядит следующим образом:



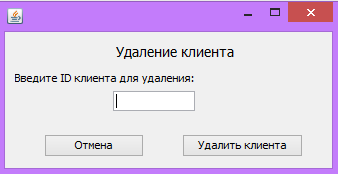
При нажатии кнопки “Обновить данные” таблица заполняется данными из базы данных.



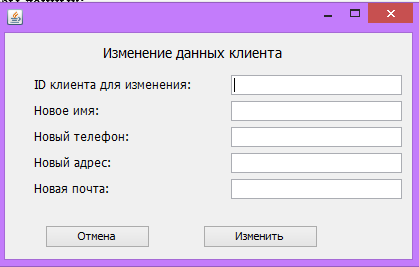
При нажатии кнопки “Добавить…” нас переносит на форму добавления данных, где после ввода всех необходимых данных нужно нажать кнопку “Добавить”, чтобы записать нового клиента в базу данных:



При нажатии кнопки “Удалить…” нас переносит на форму удаления, где после ввода идентификатора клиента необходимо нажать кнопку “Удалить”, чтобы удалить клиента из базы данных:

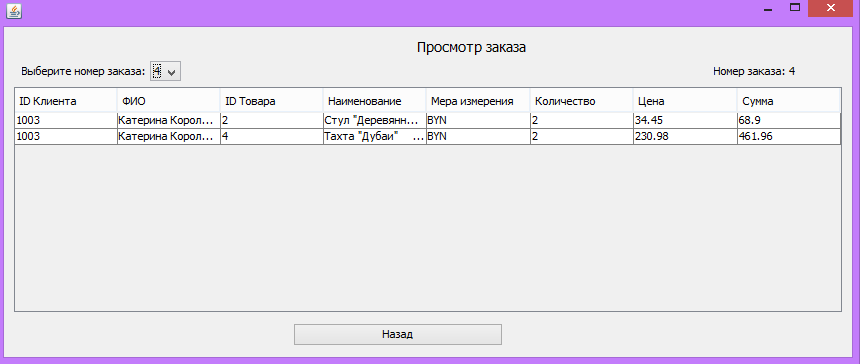


При нажатии кнопки “Редактировать…” нас переносит на форму редактирования, где после ввода идентификатора клиента и всех необходимых данных для изменения нужно нажать кнопку “Изменить”, чтобы изменить данные клиента из базы данных:



При нажатии кнопки “Назад” нас возвращает в главное меню.

Форма “Заказы” отличается от остальных форм дополнительной кнопкой “Просмотреть заказ”. Тут при нажатии на кнопку “Просмотреть заказ” можно при помощи выпадающего списка выбрать необходимый номер заказа и просмотреть всё, что находится в этом заказе:



В форме “Накладные” кнопка “Просмотреть накладную” работает по такому же принципу, как и “Просмотреть заказ” в форме “Заказы”.

Заключение

В соответствии с заданием курсового проекта было разработано и описано сетевое приложение по предметной области «Мебельный склад» с помощью программного обеспечения MSSQL Server и платформы Apache Netbeans IDE на языке программирования Java.

В процессе выполнения курсового проекта развили навыки реализации сетевых приложений c использованием прикладного интерфейса программирования сокетов; среды разработки прикладных сетевых приложений, получили опыт разработки сетевых программных средств.

В ходе написания курсового проекта были решены следующие задачи:

- развитие логического и алгоритмического мышления;

- выработка умения формулировать суждения и выводы, логически, последовательно и доказательно их излагать;

- овладение навыками самостоятельной исследовательской работы;

- закрепление использования технологией сетевого программирования на стороне клиента и сервера.

Результатом курсового проекта стало сетевое приложение с использованием графического интерфейса. После запуска приложения пользователь может просмотреть информацию, находящуюся в базе данных Мебельного склада, а также изменить её, удалить и добавить новую информацию.

# Список использованных источников

# Приложения