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

Актуальность работы. Использование новых информационных технологий в разных сферах деятельности человека привело к необходимости упрощения и ускорения работы с товарами, их покупки, продажи и доставки. Такое программное обеспечение востребовано на рынке информационных технологий, поэтому постоянно изменяется, включая в себя новые возможности.

В настоящее время распространены веб-приложения по доставки различного рода товаров. Существуют многочисленные приложения, которые пользуются спросом на доставку.

Предметной областью является перевозки топлива.

Целью курсовой работы является создание веб-приложения и 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.

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.

Современные веб-приложения используют системы аутентификации, авторизации, идентификации.

Идентификация — процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно определяющий этого субъекта в информационной системе.

Аутентификация — процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

1.3 Постановка задачи

Название приложения – топливный гараж. Предоставляет услуги автомобильных перевозок топлива. Пользователи приложения делятся на три роли: администратор, водитель, заказчик. Администратор ведет учёт за заказами, имеет возможность редактировать заказы, водителей, заказчиков. Администратор может добавлять топливо, машины, водителей. Водитель отслеживает свои заказы. Заказчик может добавлять новые заказы и просматривать информацию о своих существующих заказах.

Входные данные для администратора:

- Топливо

- Машины

- Водители

- Пользователи

Выходные данные для администратора:

- Отчёт о продажах в Excel

- Таблица обо всех заказах

- Таблица водителей

- Таблица с топливом

- Таблица с машинами

Входные данные для заказчика:

- Заказ

Выходные данные для водителя:

- Таблица с заказами

Выходные данные для водителя:

- Таблица с заказами

Выбор среды и языка программирования. В качестве языка программирования был выбран C#. Для разработки веб-приложения была выбрана среда Visual Studio 2019 Community. Visual Studio Community - бесплатная полнофункциональная расширяемая среда IDE для создания современных приложений Android, iOS и Windows, а также веб-приложений и облачных служб. C# использует объектно-ориентированный подход к программированию во всем. Это означает, что надо описывать абстрактные конструкции на основе предметной области, а потом реализовывать между ними взаимодействие. Данный подход пользуется большой популярностью, потому что позволяет не держать в голове всю информацию, а работать по принципу черного ящика.

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 навигационного меню;

– нажать на необходимую ссылку (“Добавить”, “Удалить”, “ Изменить” или “Подробнее”);

– при выборе пунктов “Добавить”, “Изменить”, необходимо ввести нужные данные;

– нажать на кнопку “Save” или “Create”, либо нажать на ссылку “Изменить” или “Назад”;

– вывести результаты.

Например, алгоритм работы веб-страницы “Заказы”:

Добавление нового заказа:

– нажать на ссылку “Оформить новый заказ”;

– ввести необходимые данные;

– нажать кнопку “Create” для добавления топлива и возвращения на веб-страницу “Заказы” уже с обновленными данными.

Редактирование заказа:

– нажать на ссылку “Изменить”;

– изменить необходимые данные;

– нажать кнопку “Save” для возвращения на веб-страницу “Заказы” уже с измененными записями;

Удаление Заказа:

– нажать на ссылку “Удалить” для обновления веб-страницы “Заказы”;

Просмотр подробной информации о заказе:

– нажать на ссылку “Подробнее” открывается подробная информация заказа;

– нажать на кнопку “Save” для изменения необходимой информации;

– нажать на ссылку “Назад” для возвращения на веб-страницу “Заказы”;

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

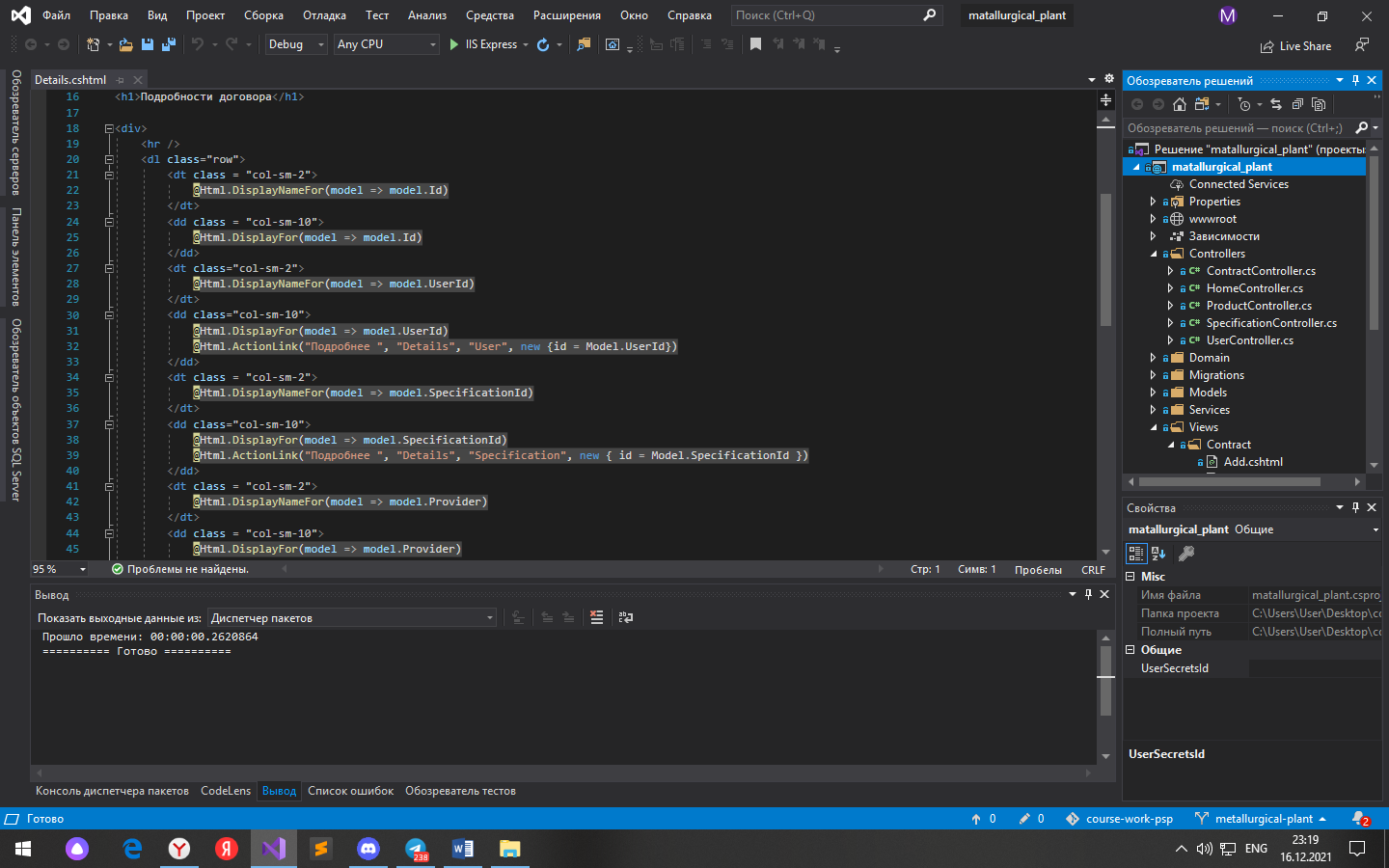


Рисунок 2.1 Контроллеры металлургического завода

Модели представлены на рисунке 2.2

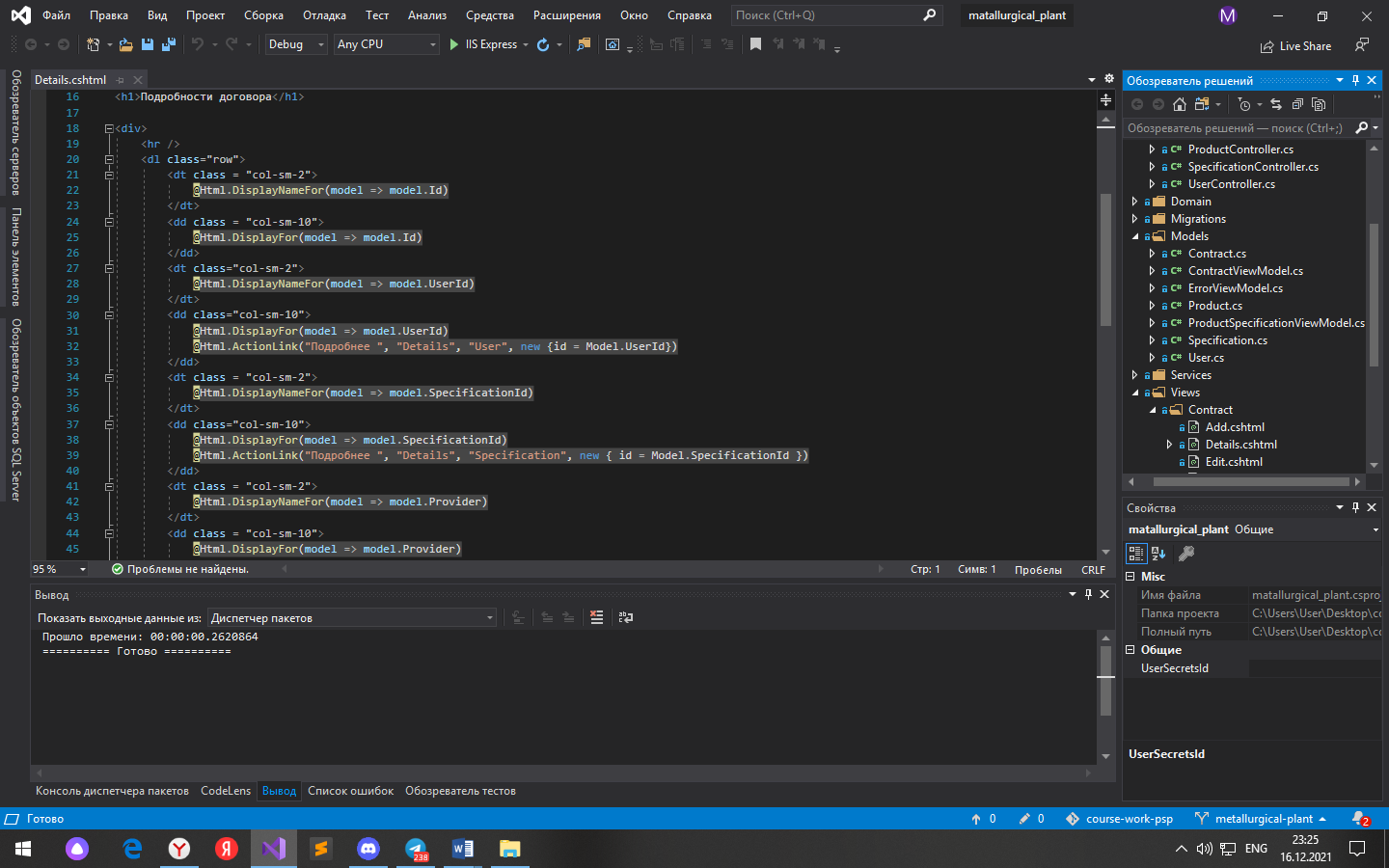


Рисунок 2.2 Модели металлургического завода

Сервисы представлены на рисунке 2.3

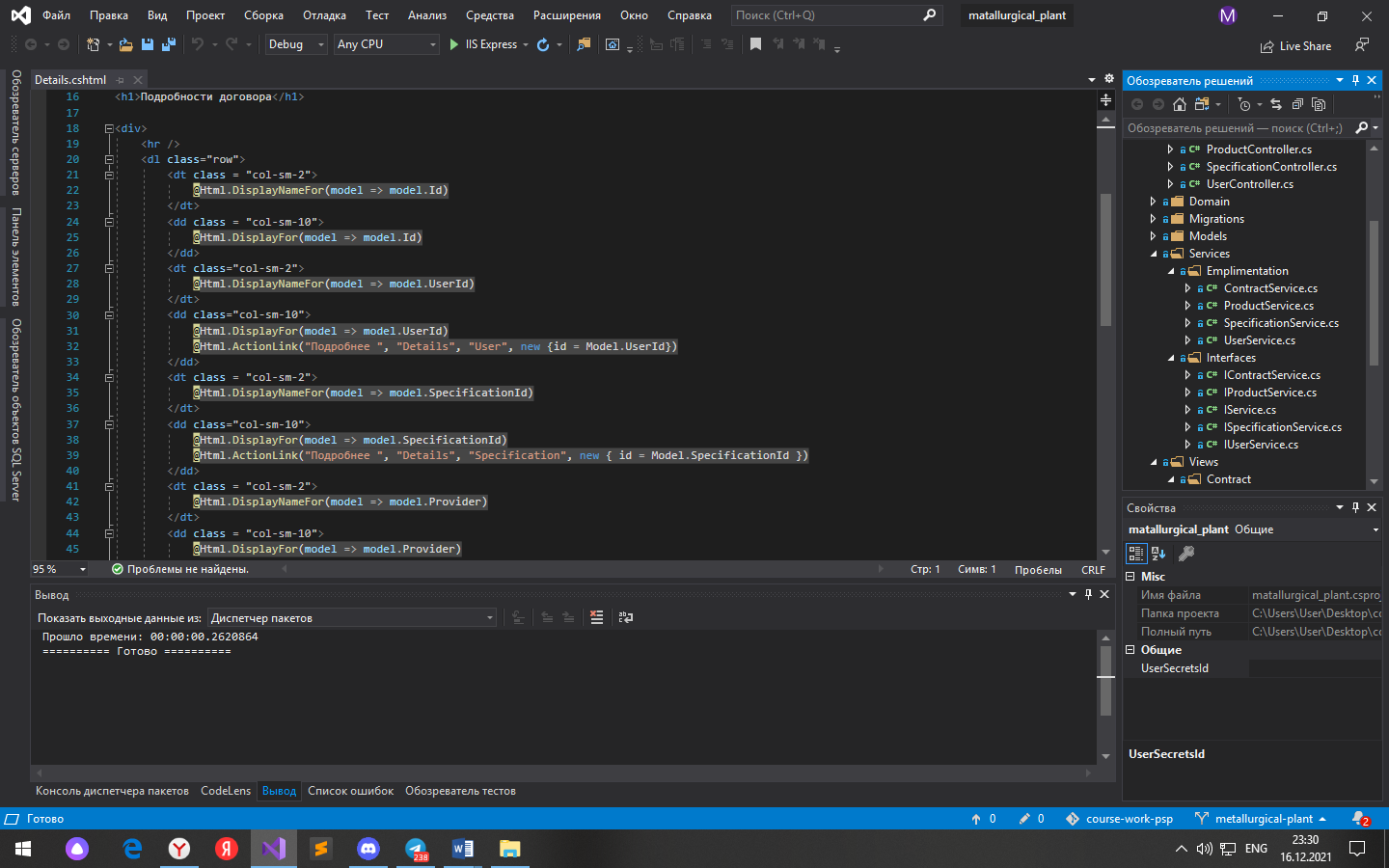


Рисунок 2.3 Сервисы металлургического завода

Интерфейсы представлены на рисунке 2.4

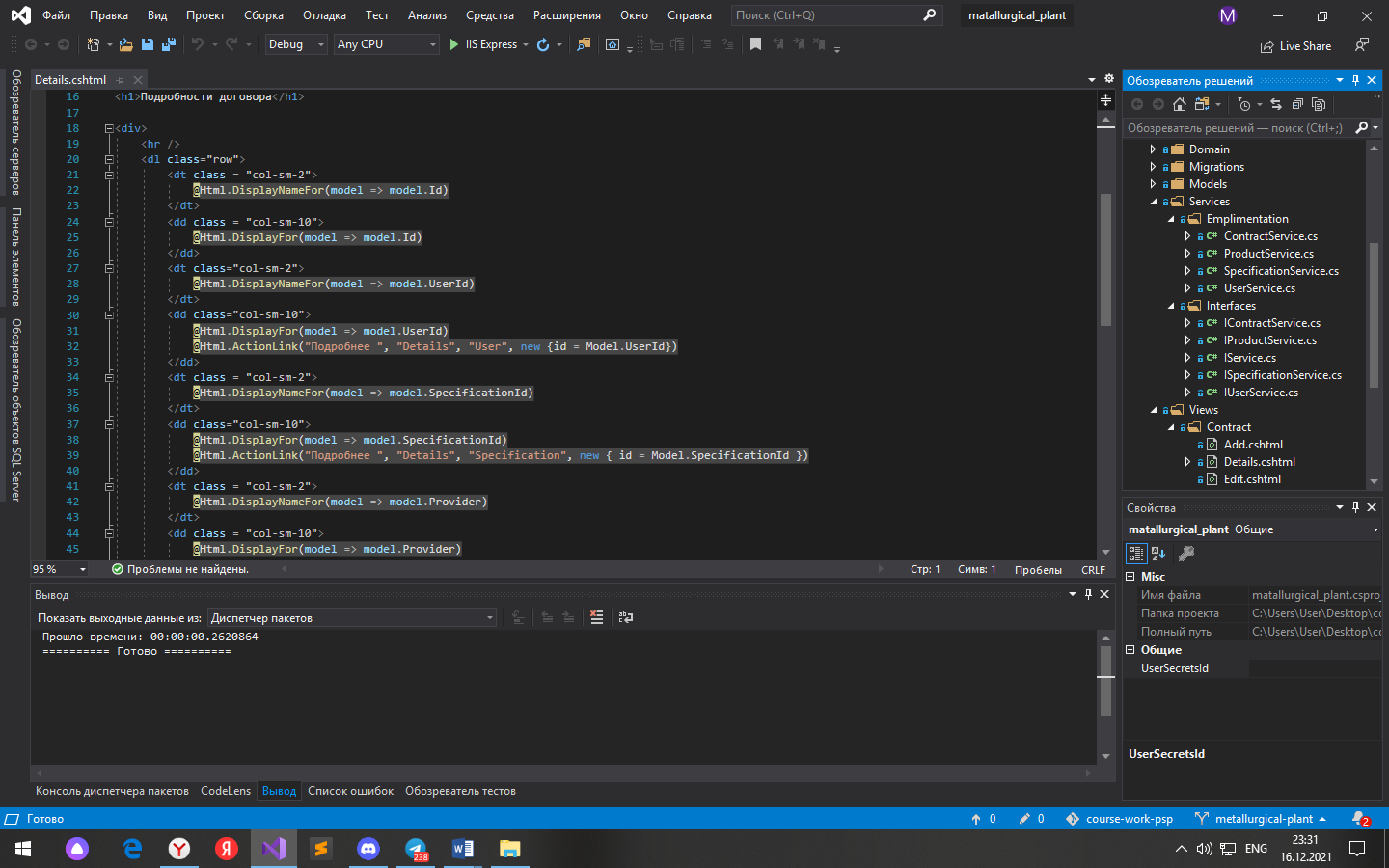


Рисунок 2.4 Интерфейсы металлургического завода

Представления представлены на рисунке 2.5

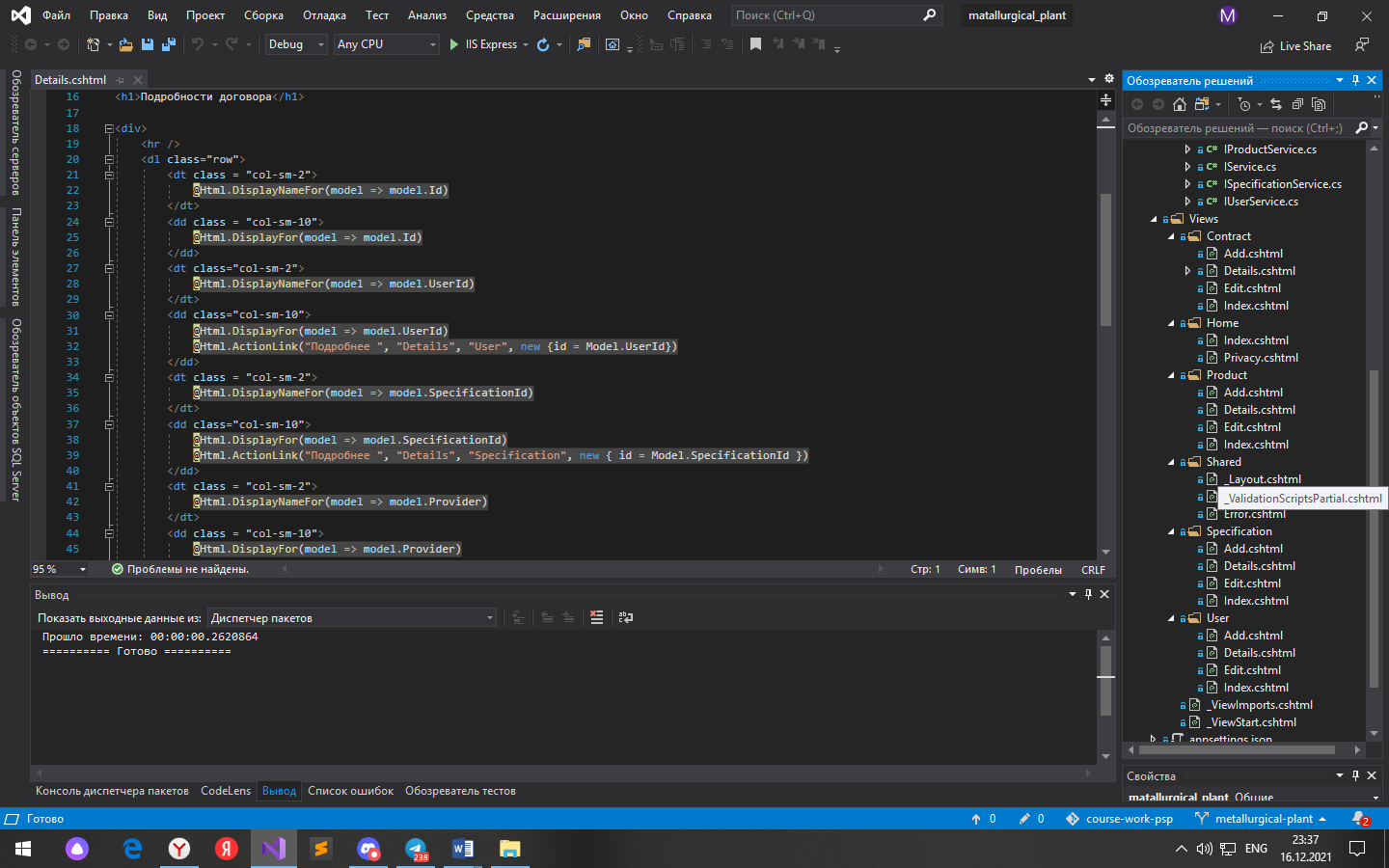


Рисунок 2.5 Представления металлургического завода

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;

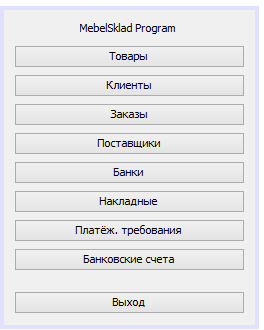
- Suppliers.java — форма для просмотра всех поставщиков в базе и перехода на формы SuppliersAdd.java, SuppliersDelete.java, SuppliersEdit.java;

- 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;

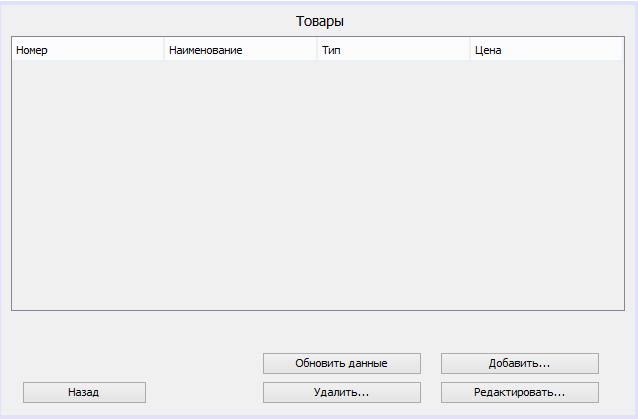
import java.util.logging.Level;

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();

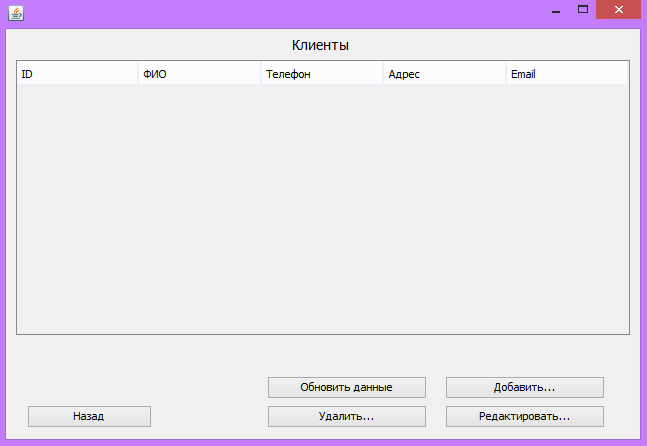
}

Листинг 1 – Код подключения к базе данных

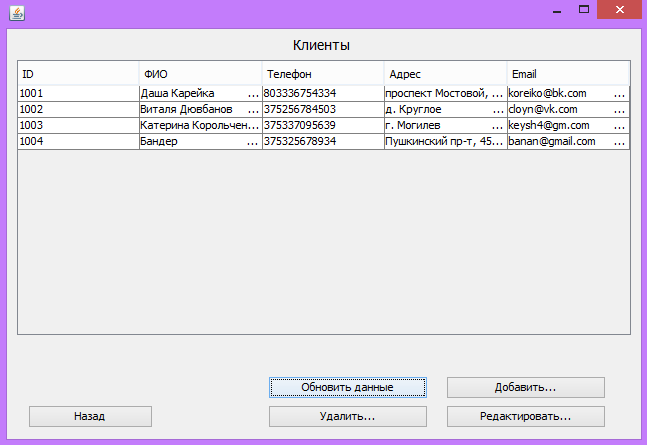
2.4 Результаты работы программы

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

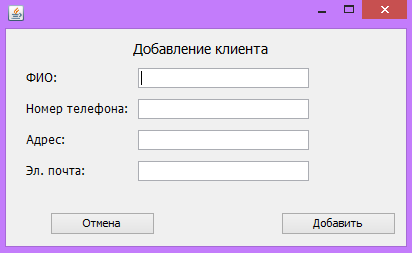
Форма “Клиенты” выглядит следующим образом:



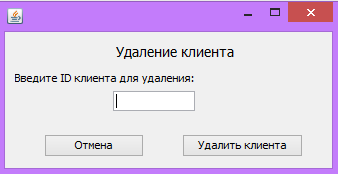
При нажатии кнопки “Обновить данные” таблица заполняется данными из таблицы Client в БД MS SQL Server:



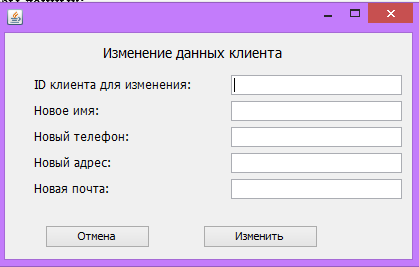
При нажатии кнопки “Добавить…” нас переносит на форму добавления данных, где после ввода всех необходимых данных нужно нажать кнопку “Добавить”, чтобы записать нового клиента в базу данных:



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

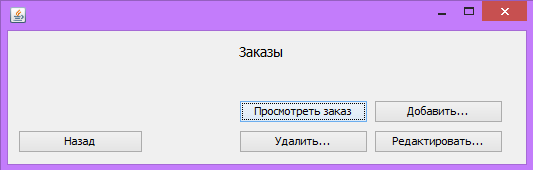


При нажатии кнопки “Редактировать…” нас переносит на форму редактирования, где после ввода идентификатора клиента и всех необходимых данных для изменения нужно нажать кнопку “Изменить”, чтобы изменить данные клиента из базы данных:

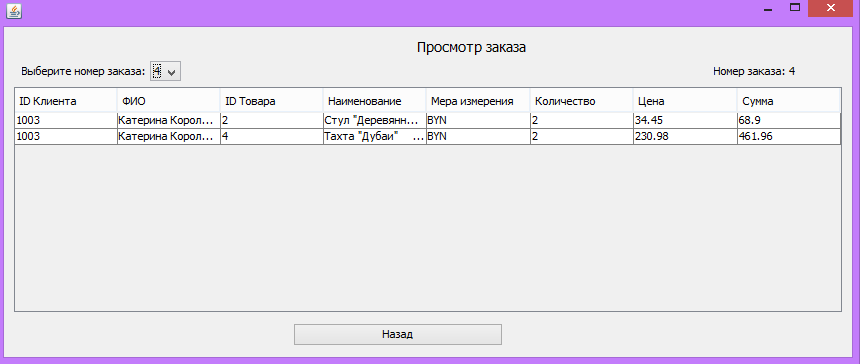


При нажатии кнопки “Назад” нас возвращает в главное меню.

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



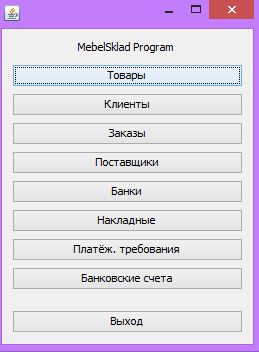
При нажатии на кнопку “Просмотреть заказ” можно при помощи выпадающего списка выбрать необходимый номер заказа и просмотреть всё, что находится в этом заказе:



Все остальные формы работают по такому же одинаковому принципу.

3. Руководство пользователя

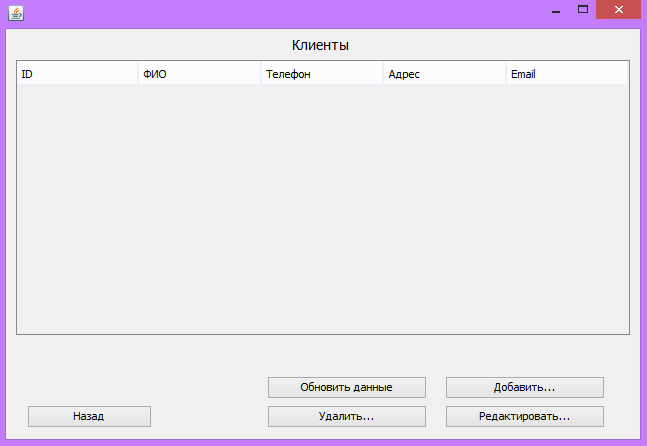
Впервые открыв программу (предварительно подготовив базу данных) пользователь попадает в главное меню программы:



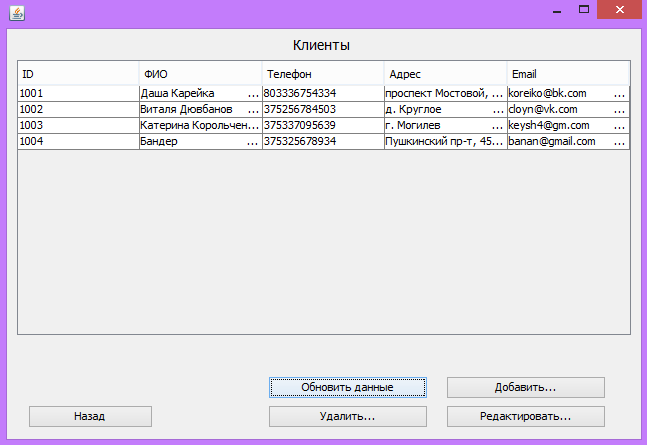
Каждая кнопка ведёт к отдельной одноимённой форме, где можно не только посмотреть уже существующие данные, но и изменить их, удалить или добавить новые.

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.

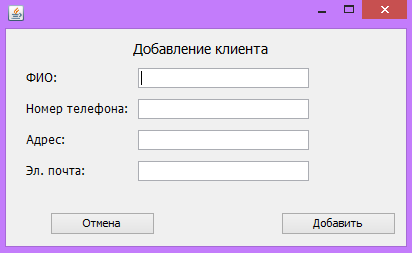
Форма “Клиенты” выглядит следующим образом:



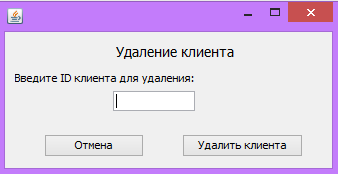
При нажатии кнопки “Обновить данные” таблица заполняется данными из базы данных.



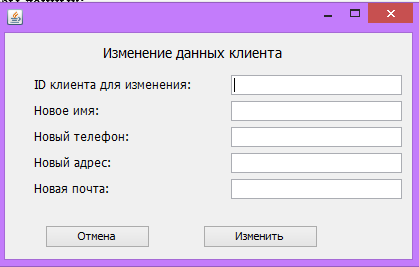
При нажатии кнопки “Добавить…” нас переносит на форму добавления данных, где после ввода всех необходимых данных нужно нажать кнопку “Добавить”, чтобы записать нового клиента в базу данных:



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

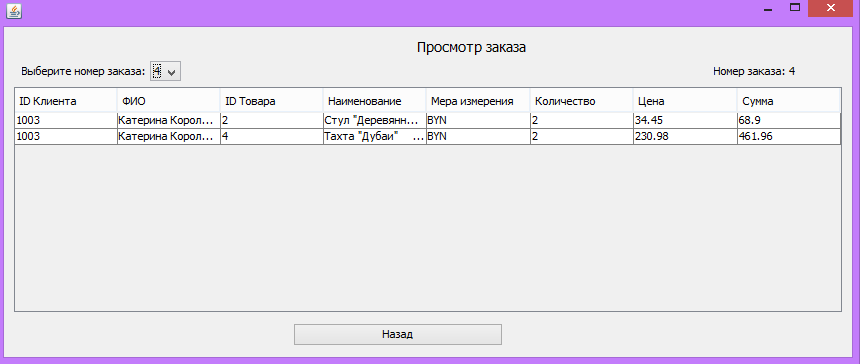


При нажатии кнопки “Редактировать…” нас переносит на форму редактирования, где после ввода идентификатора клиента и всех необходимых данных для изменения нужно нажать кнопку “Изменить”, чтобы изменить данные клиента из базы данных:



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 ходе работы над курсовым проектом я отточил навыки работы с клиентской и серверной частью, а именно работой с MVC паттерном в языке программирования C#, javascript, css, технологией Entity Framework Core.

В ходе выполнения курсового проекта были закреплены понятия объектно-ориентированного и программирования, а также работой с пользовательским интерфейсом.

Интерфейс созданного приложения удобен, прост и интуитивно понятен для всех пользователей.

Серверная часть приложения была написана на языке C# с использованием MVC.

Список использованных источников

1. Справочный материал MSDN. Microsoft [Электронный ресурс] — 2021. —Режим доступа: <https://msdn.microsoft.com/library>.
2. Справочный материал w3schools.com [Электронный ресурс] — 2021. — Режим доступа: [https://www.w3schools.com](https://www.w3schools.com/).
3. Сайт о программировании Habrahabr.ru [Электронный ресурс] — 2021 — Режим доступа: [http://habrahabr.ru](http://habrahabr.ru/).
4. Сайт о программировании Stackoverflow.com [Электронный ресурс] — 2021. — Режим доступа: [https://stackoverflow.com](https://stackoverflow.com/).
5. Сайт о программировании Metanit.com [Электронный ресурс] — 2021. — Режим доступа: <https://metanit.com>.

# Приложение 1