**Компьютерная академия «ШАГ»**

**Черниговский филиал**

**Кафедра Разработки программного обеспечения**

**ДИПЛОМНЫЙ ПРОЕКТ**

**“Система CRM для хранения и управления данными по клиентам, заказам для малого предприятия”**

**Студента группы ПС/14**

**Приступа Е.С.**

**Рево М.А.**

**Научный руководитель:**

**Потапенко О.В.**

**Чернигов 2018**

Аннотация

Тема работы: Система CRM для хранения и управления данными по клиентам, заказам для малого предприятия.

Объектом разработки была программа для хранения, создания клиентов и их заказов. Объект разработки представляет собой программу связанную с локальной базой данных. При разработке использовались технологии HTML, CSS, PHP, MySQL, JavaScript (JQuery) .

Цель работы:Система CRM для хранения и управления данными по клиентам, заказам для малого предприятия, с возможностью показа этих данных по дате, названию клиента, названию заказа, возможность добавления материалов и работ в конкретный заказ.

В ходе выполнения разработки программы были получены:

- программа-клиент, которая обеспечивает связь между пользователем/администратором и информацией базы данных;

- база данных, которая хранит данные;

Содержание

[Введение](#_gjdgxs) 4

[Техническое задание](#_30j0zll) 5

[1 ВЫБОР ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТА](#_1fob9te) 7

[2 РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ](#_3znysh7) 8

[3 РАЗРАБОТКА АЛГОРИТМОВ](#_2et92p0) 10

[4 РАЗРАБОТКА БАЗЫ ДАННЫХ](#_tyjcwt) 11

[5 РАЗРАБОТКА ИНТЕРФЕЙСА](#_3dy6vkm) 14

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](#_1t3h5sf) 18

[Выводы](#_4d34og8) 26

[Список использованной литературы](#_2s8eyo1) 27

[Приложения](#_17dp8vu) 28

Введение

Использование систем для хранения и управления данными все чаще находят применение в самых различных областях жизни и трудовой деятельности человека. В связи с этим большую актуальность приобретает освоение принципов построения и эффективного применения соответствующих технологий и программных продуктов: систем управления базами данных, систем автоматизации проектирования, средств администрирования и защиты баз данных и других.

Использование баз данных и информационных систем становится неотъемлемой составляющей бизнес деятельности современного человека и функционирования многих преуспевающих организаций.

От правильного выбора инструментальных средств создания информационных систем, определения подходящей модели данных, обоснования рациональной схемы построения базы данных, организации запросов к хранимым данным и множества других моментов зависит эффективность функционирования разрабатываемых систем.

Поэтому самая главная цель данной дипломной работы - это решение конкретной задачи по созданию такой системы, где бы пользователь мог хранить данные о своих клиентах, создавать для этих клиентов непосредственно некоторый заказ, наполнять его материалами и работами, рассчитывать стоимость заказа исходя из затрат на приобретение материалов, трудозатрат на изготовление, затрат на доставку, вносить данные об оплатах клиентом, вносить данные о процессе производства, установки или доставки заказчику его заказа.

Техническое задание

Краткое описание: программа предназначена для использования в малом предприятии , так называемом ФОП - физическое лицо предприниматель.

Рабочая среда: одиночный ПК.

База данных: локальная база данных.

Данные для хранения: списки клиентов, списки заказов, списки материалов, списки поставщиков.

Пользователи: администратор, ФОП, ведущий бизнес.

Информация, поступающая в базу:

название клиента и его данные, а именно: Фамилия Имя Отчество, телефон, адрес нахождения;

название заказа и его данные, а именно: Название, Дата получения заказа, Дата окончания исполнения заказа, Цена заказа, Состояние, Комплектация, Оплата заказчиком сумм;

название заказчика и его данные, а именно: Название, Доп. характеристики, Телефон, Адрес;

название материала и его данные, а именно: Название, Доп. характеристики,

Единица измерения, Цена за Единицу, Поставщик;

поставщик Материалов и его данные, а именно: Название, Доп. характеристики, Телефон, Адрес.

Для работы программы нужен ПК.

Системные требования:

- Windows 7;

- файловая система NTFS;

- Браузер Хром или Опера;

- быстродействие процессора 1.0 ГГц и выше;

- процессор Intel Core I3 или выше.

Назначение: программа предназначена для использования учителями, родителями с целью занесения и контроля успеваемости, посещаемости учеников.

Программа должна обеспечивать возможность выполнения следующих функций:

•занесение данных о контактах, клиентах, заказах, поставщиках, материалах в базу данных;

•внесение изменений в записи базы;

•удаление записей;

•хранение данных;

•просмотр данных;

•добавление, удаление материалов в базу материалов;

•добавление, удаление материалов в заказ;

1 ВЫБОР ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТА

Проект разрабатывался согласно парадигмы программирования, в которой основными концепциями являются понятия объектов и классов (объектно-ориентированное программирование).

При разработке проекта были использованы такие технологии как:

- HTML - *HyperText Markup Language* — “язык гипертекстовой разметки”

-CSS - *Cascading Style Sheets-* каскадные таблицы стилей, позволяющие описывать внешний вид документа, написанного с использованием HTML;

- PHP - скриптовый общего назначения, интенсивно применяемый для разработки веб-приложений;

- JQuery - библиотека JavaScript ,предназначенная для взаимодействия JavaScript и HTML и позволяющая управлять отображение в браузере нужных данных в удобном виде;

-MySQL - реляционная система управления базами данных;

2 РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ

В базе данных будет храниться следующая информация данные:

* о контактах - они еще не клиенты нашего предприятия, но могут ими стать,
* о клиентах,
* о поставщиках,
* о материалах,
* об оплатах поставщиками за выполнение заказа,
* логин/пароль;

Пользователями данной СУБД являются администратор выполняющий настройку программы, ФЛП-пользователь, выполняющий внесение, редактирование, удаление данных. При этом разные категории пользователей имеют разные права доступа: если пользователь-администратор может добавлять, удалять, изменять записи в базе данных, а ФЛП-пользователь может выполнять только занесение данных, касающихся бизнес сегмента.

3 РАЗРАБОТКА АЛГОРИТМОВ

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий.

Алгоритм работы ФЛП-пользователь:

1. заходит в систему через ввод логина и пароля.
2. в таблице выбирает заказ из списка из списка заказов.
3. во вкладке-заказе может добавлять материалы, оплаты, вносить изменения в цену и т.д.
4. во вкладке-контакты может добавить контакт, найти нужный контакт, перейти на вкладку изменения данных конкретного контакта .
5. во вкладке-клиенты может добавить клиента, найти нужного клиента, перейти на вкладку изменения данных конкретного клиента.
6. во вкладке-поставщики может добавить поставщика, найти нужного поставщика, перейти на вкладку изменения данных конкретного поставщика.
7. во вкладке-материалы может добавить материал, найти нужный материал, перейти на вкладку изменения данных конкретного материала.
8. во вкладке-отчеты ...дописать Катерина

Администратор - это лица, которые непосредственно настраивают программу. Также настраивает права доступа к базе данных. ...дописать Катерина

В нашей системе есть такие компоненты: база данных и сама программа. При старте программы идет запрашивание «имени» и «пароля», после ввода информации и попытке войти в программу происходит соединение с базой данных при помощи которого (соединения) идет обмен данными между этими компонентами. При распознании «администратора» доступны все элементы управления, при распознании «пользователя» доступны не все элементы управления.

После изменения данных происходит обновление базы данных.

...дописать Катерина

4 РАЗРАБОТКА БАЗЫ ДАННЫХ

В связи с тем, что программа работает с базой данных MySQL, для базы данных была разработана реляционная база данных и нормализованная согласно первых трех нормальных форм. В базе данных создали 16 таблиц (рис 4.1):

* users – таблица пользователей системы;
* clients –таблица клиентов ;
* contatctsclients – таблица контактов;
* materials– таблица материалов;
* materialsToOrder– таблица материалов, используемых в заказах;
* menu – таблица для вывода меню;
* orders – таблица заказов;
* payments – таблица платежей ;
* suppliers - таблица поставщиков;
* task – таблица дополнительных действий;

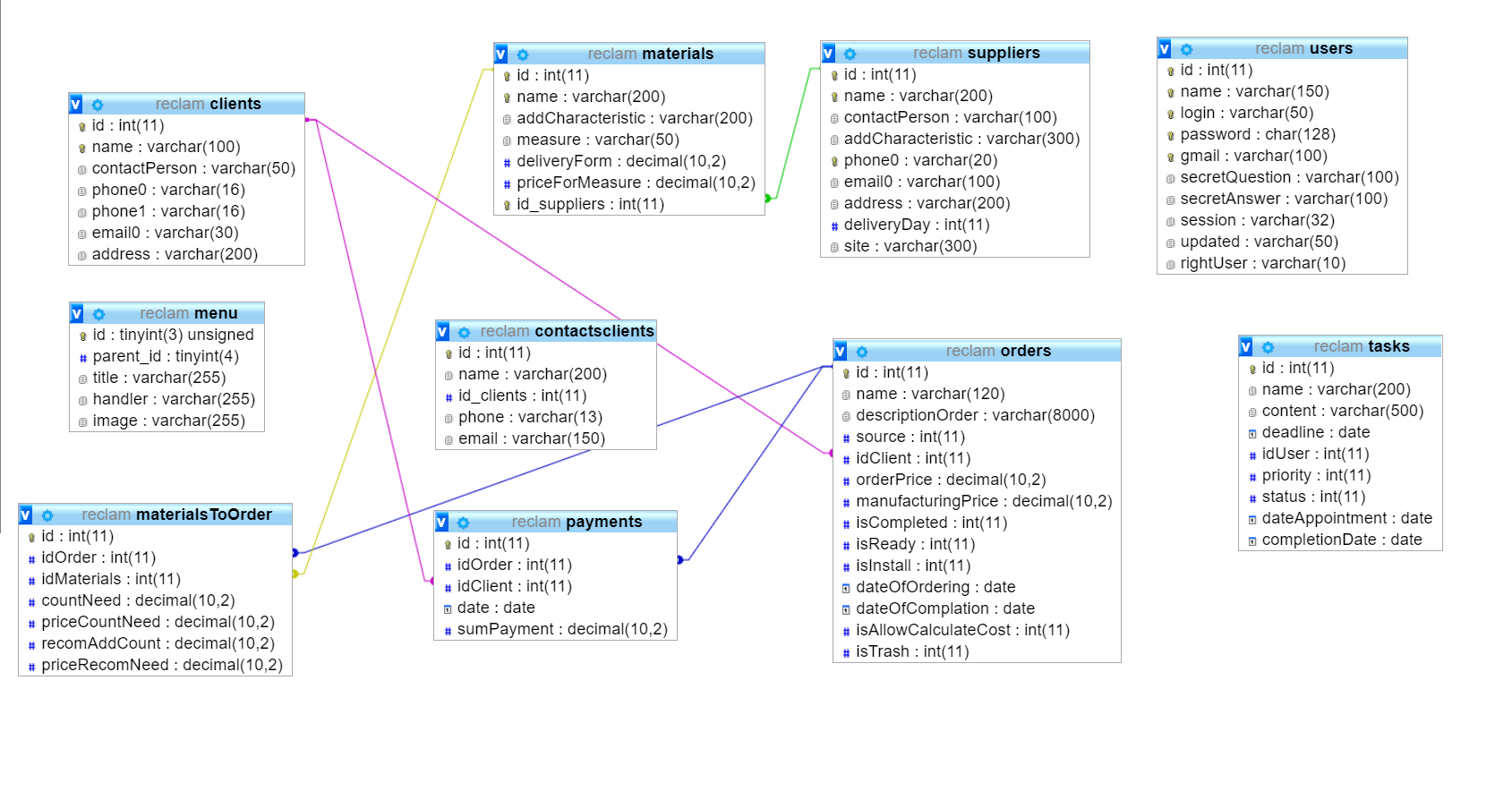


Рисунок 4.1 – Схема связей таблиц базы данных.

5 РАЗРАБОТКА ИНТЕРФЕЙСА

Интерфейс — совокупность возможностей, способов и методов одновременного взаимодействия двух систем, устройств или программ для обмена информацией между ними, определённая их характеристиками, характеристиками соединения, сигналов обмена и т.п.

Самое главное в интерфейсе не насыщенность цветовой гаммы, а удобство использования. Иногда при большом количестве настроек программы, пользователь начинает в ней путаться из-за чего отстраняется от использования данного продукта. Интерфейс нашего продукта настолько прост, что даже самый неподготовленный пользователь сможет с ним разобраться без специальной подготовки. Интерфейс разрабатывался так, чтобы пользователи не смогли навредить базе данных. В нашей системе использовались основные окна. А именно:

* окно авторизации (рис. 5.1) – с помощью его только авторизованный пользователь сможет войти в программу;
* основное окно из которого доступны все действия (рис. 5.2) состоит из нескольких зон :

1. зона “меню” (рис.5.4) - то есть зона выбора “чем будем управлять” - контакты, или клиенты, или поставщики, или материалы. По желанию клиенту можно произвести добавку дополнительных блоков в это меню.
2. зона “поиска и добавления” (рис. 5.5) - можно найти сущность по введенным в поля запросам, можно создать новую сущность через кнопку добавить.
3. зона “дополнительной информации, для показа успешности - неуспешности выполняемых действий пользователя- находится над зоной “поиска и добавления”.
4. зона “основного контента” - показ сущностей в форме таблицы - находится ниже “зоны поиска” прямо по центру экрана

- окна администратора, как такового, не предусмотрено, но при входе в систему, если ввести логин и пароль, то система распознает администратора и даст ему дополнительные права, при этом добавит в меню слева еще дополнительно поле “add module” (рис. 5.3) – в котором можно настраивать программу, вносить различные изменения в базу данных и так далее;

Катерина - заменить

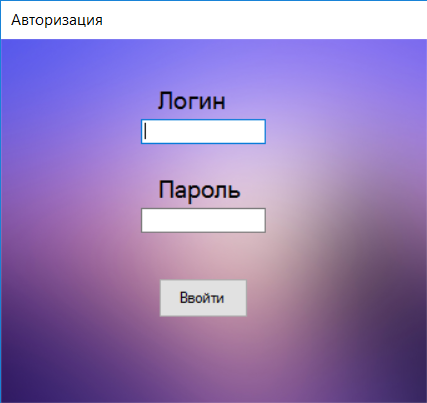


Рисунок 5.1 – Окно входа в систему.



Рисунок 5.2 - Основное окно

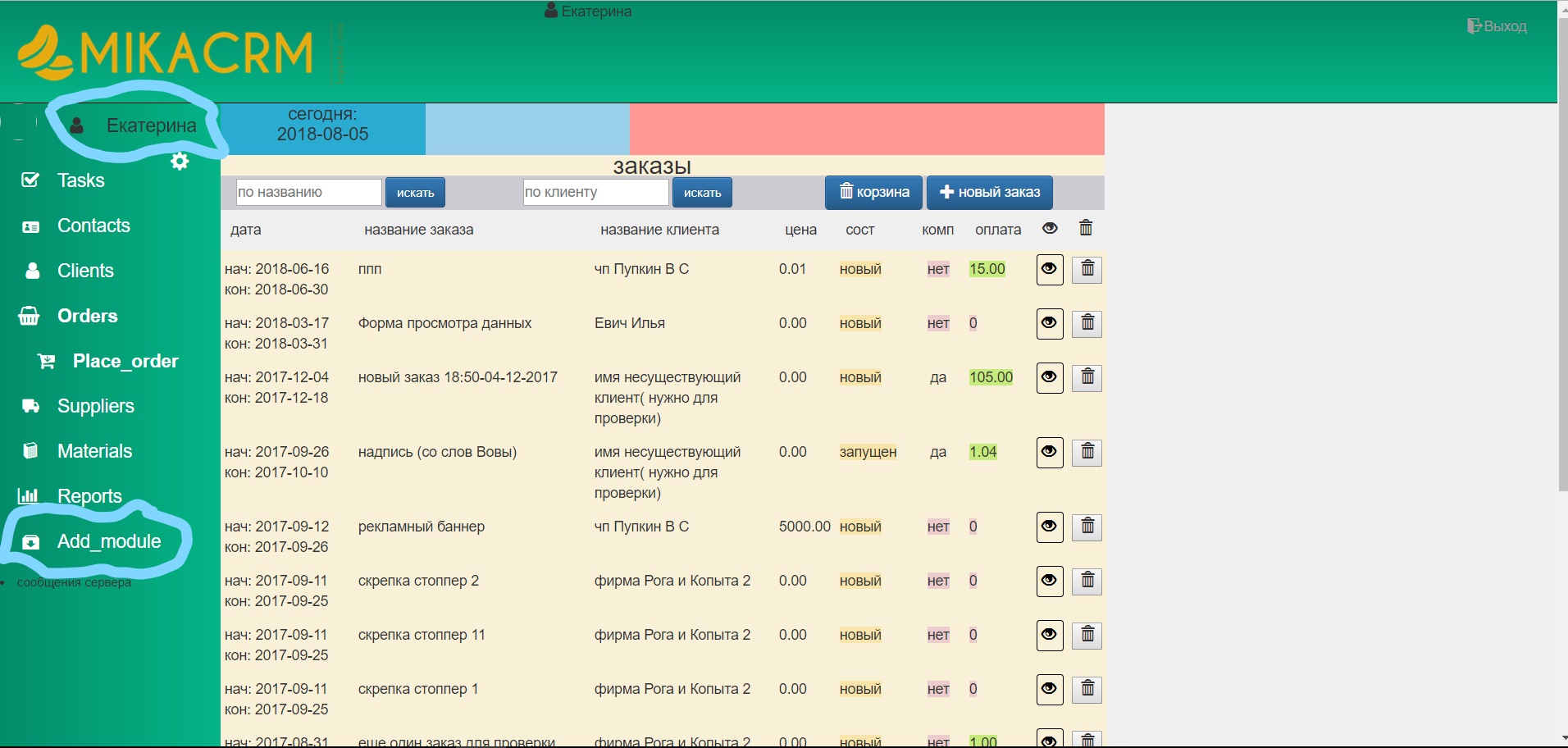


Рисунок 5.3 - форма распознала администратора и добавила дополнительный пункт меню для его работы.

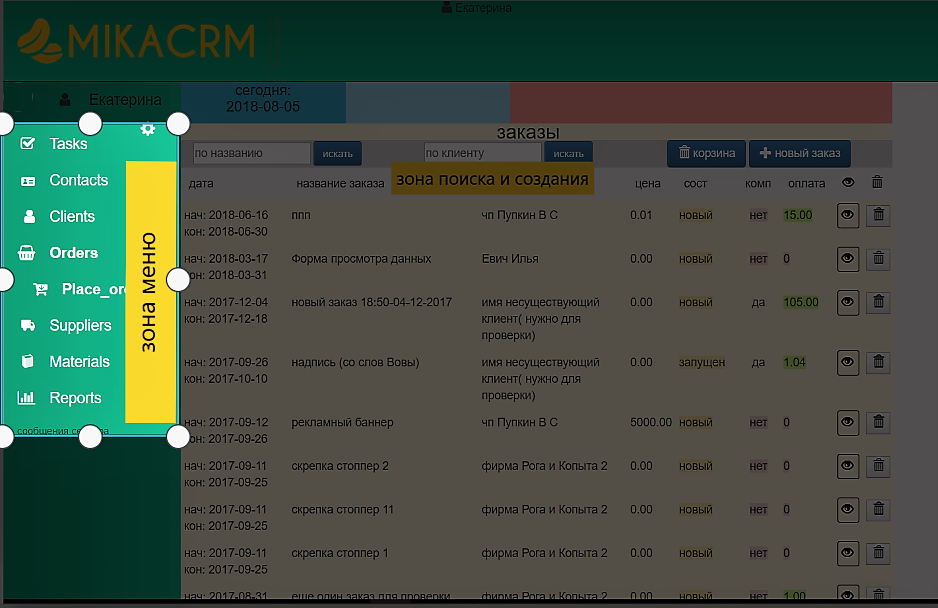


Рисунок 5.4 зона “меню”

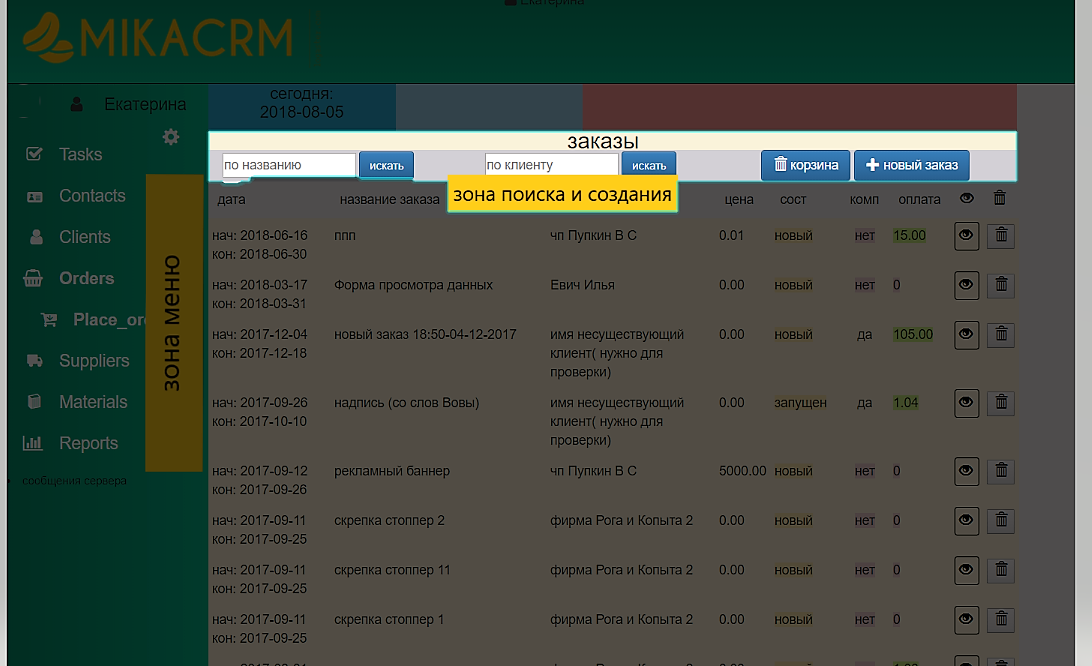


Рисунок 5.5 Зона “поиска и добавления новой сущности”

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для начала работы с программой необходимо запустить браузер. В адресной строке нужно ввести адрес домена, где расположен этот сайт. Производитель программы может помочь зарегистрировать доменное имя на доступном хостинге. При этом отобразится окно входа в систему (рис. 6.1):

Доработать Катерина...

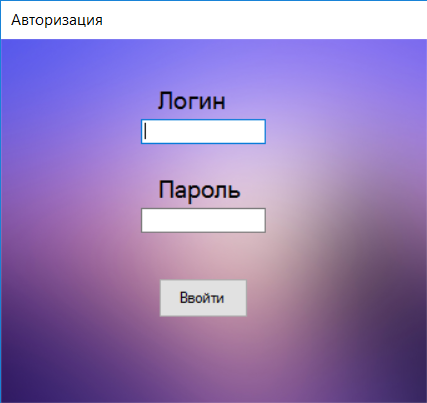


Рисунок 6.1 – Окно входа в систему.

Для входа в систему, необходимо ввести имя и соответствующий пароль. После верификации откроется рабочее окно где можно сразу приступить к работе согласно данного “руководсву пользователя”

1. Работа пользователя.

1.1. Выбор сущности с которой будем работать.

1.1.1. Путем нажатия на название в меню контент с которым предстоит работа;

1.1.2. в главной зоне загрузится контент согласно выбранного пункта меню;

1.1.3. в зоне поиска можно выбрать сущность через ввод в поле поиска названия или дополнительной характеристики сущности и работать с найденным согласно заданным данным;

1.2. Создание заказа

1.2.1. проведем полное создание нового заказчика и добавим ему заказ.

Для этого на главном меню кликнем пункт “клиент”. При этом отобразятся все клиенты, что есть в нашей базе данных. Нажмем в поле поиска “+новый клиент”. Подгрузится форма создания и добавления нового клиента в нашу базу данных. Надо заполнить форму пройдя по всем полям и следуя подсказкам. После заполнения форму нажать кнопку отправить. (рис. 6.2).

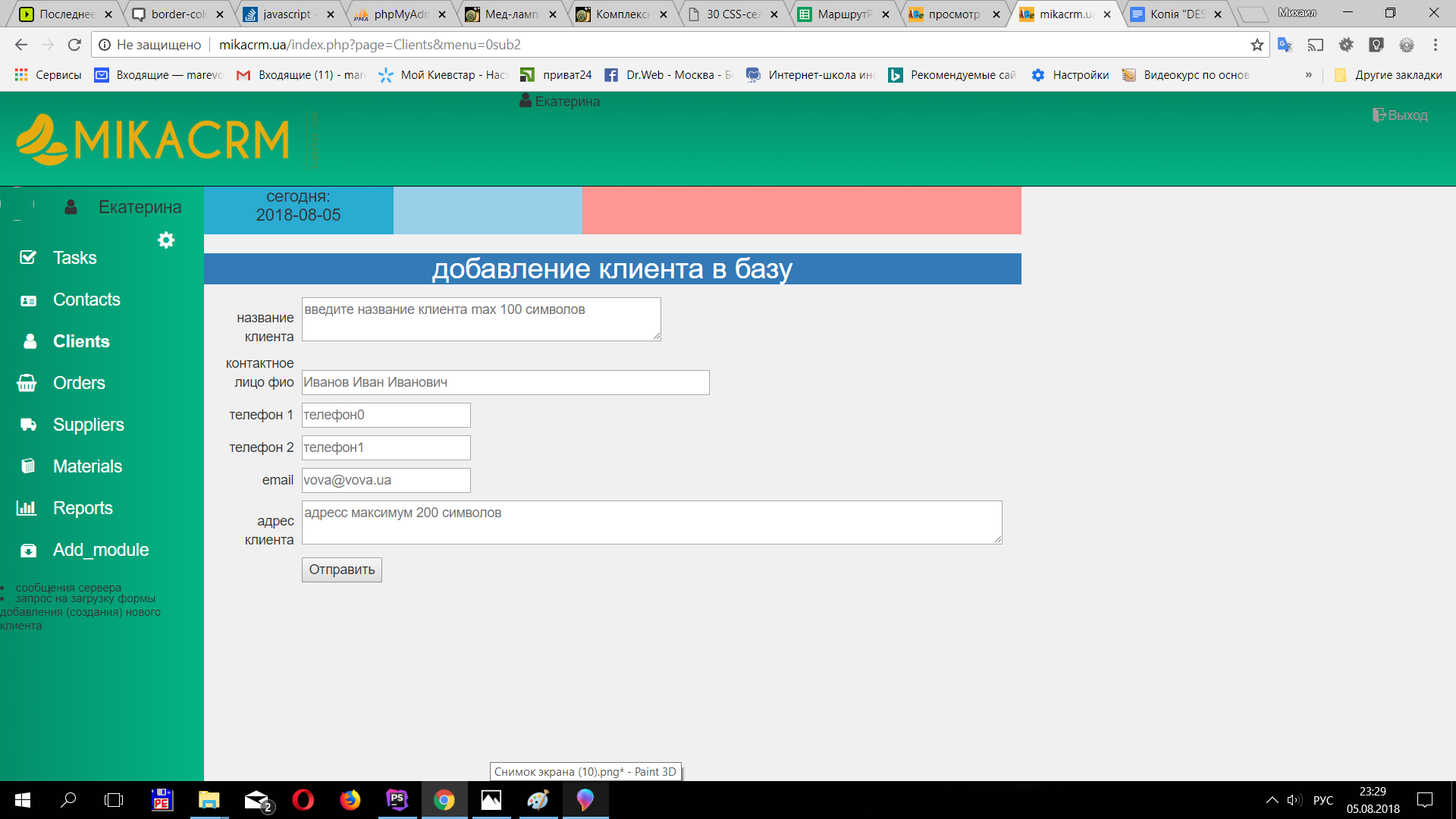


Рисунок 6.2 - Создание нового клинта.

Если при заполнении полей значения не проходят валидацию, при схождении с поля, выскакивает подсказка о не правильных значениях (рис. 6.3)

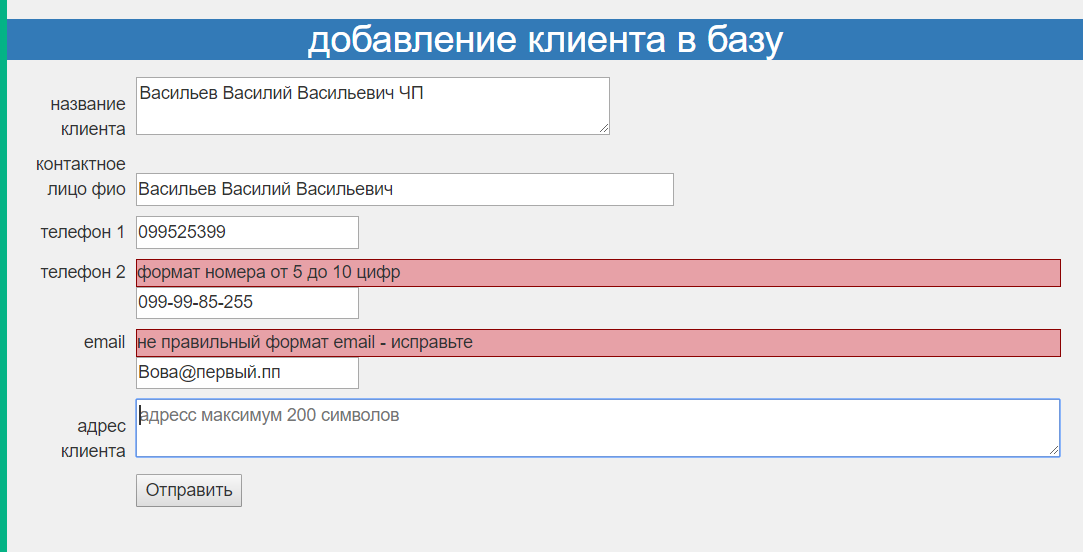


Рисунок 6.3 -Неправильно заполненые поля нуждаются в исправлении.

1.2.2. После заполнения полей и нажатии кнопки “Отправить” в “строке подсказки”, которая находится над “строкой поиска” мы получим подсказку о том, что все получилось, то есть появится и исчезнет подсказка “клиент успешно добавлен” (рис. 6.4).

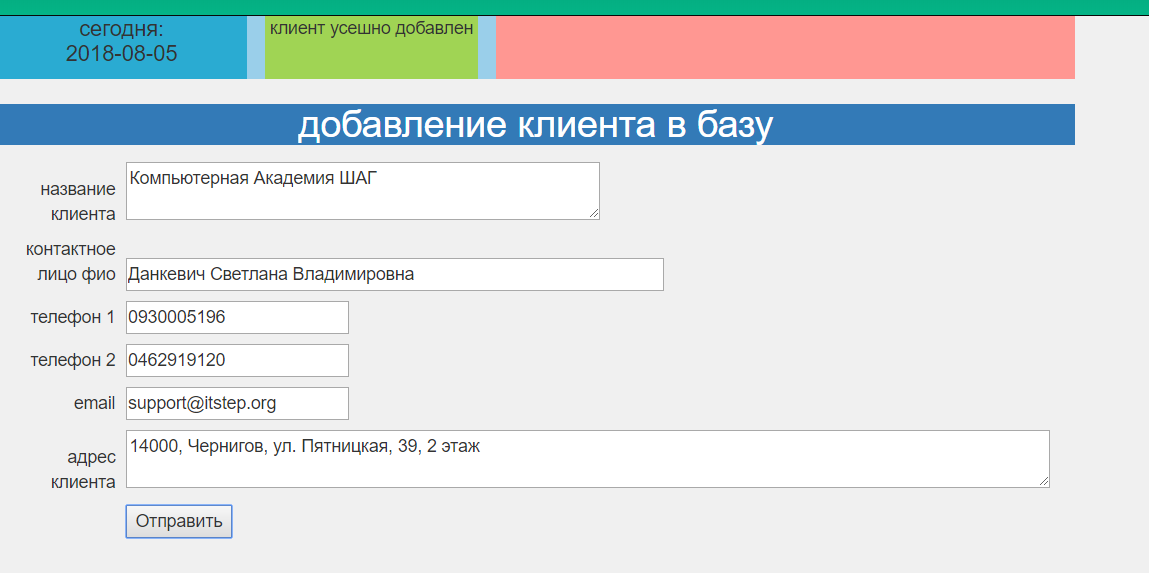


Рисунок 6.4 - Подсказка об успешности операции добавления в базу нового клиента.

1.2.3. Затем можно перейти в меню “клиенты” для правки данных клиента или его полного удаления (Если нет заказов у данного клиента, то его можно удалить, в противном случае удалить его просто так не получиться), или перейти в меню “заказы” для добавления заказа для этого клиента ( или любого другого, находящегося в нашей базе. Перейдем в меню заказы и создадим заказ. Все будет так же как и при создании клиента.

1.2.4. Заполним поле “Название” заказа, поле “Описание заказа”.

1.2.5. В выпадающем меню можно выбрать нашего клиента - ШАГ, или можно через поле поиска найти этого клиента, чтобы выбрать. (рис. 6.5)

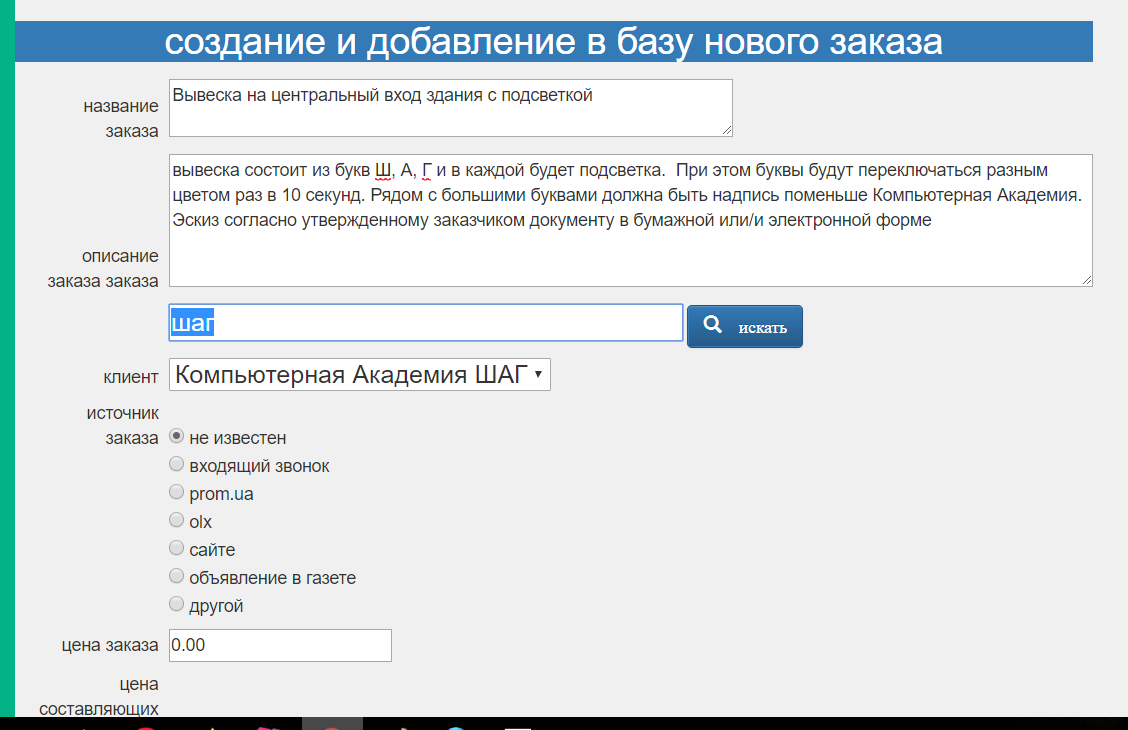


Рисунок 6.5 - Пример выбора/поиска и выбора заказчика.

1.2.6. продолжаем заполнение полей “Источник заказа”, “Цена заказа”. Это поле можно оставить пока нулевым, с тем, чтобы заполнить его после добавления материалов и работ ( добавление материалов будет рассмотрено ниже).

1.2.7. состояние заказа, естественно при создании заказа - не укомплектовано ( стоит по умолчанию). Как только мы поставим, что заказ укомплектован, то лишимся возможности добавлять материалы;

1.2.8. дата взятия заказа ( возможно дата начала работ ) и дата закрытия заказа (возможно дата окончания работ и проплат по этому заказу) установлены с разницей в 2 недели, но можно и сдвинуть, выбрав другую дату;

1.2.9. жмем кнопку отправить и получим ответ от сервера, что все успешно ( естественно, если нет ошибок валидации, и название заказа не повторяет другой заказ) ;

1.2.10. теперь, перейдем в меню “заказы” (рис. 6.6) и увидим там наш заказ прямо сверху. Начнем работать с данным заказом;

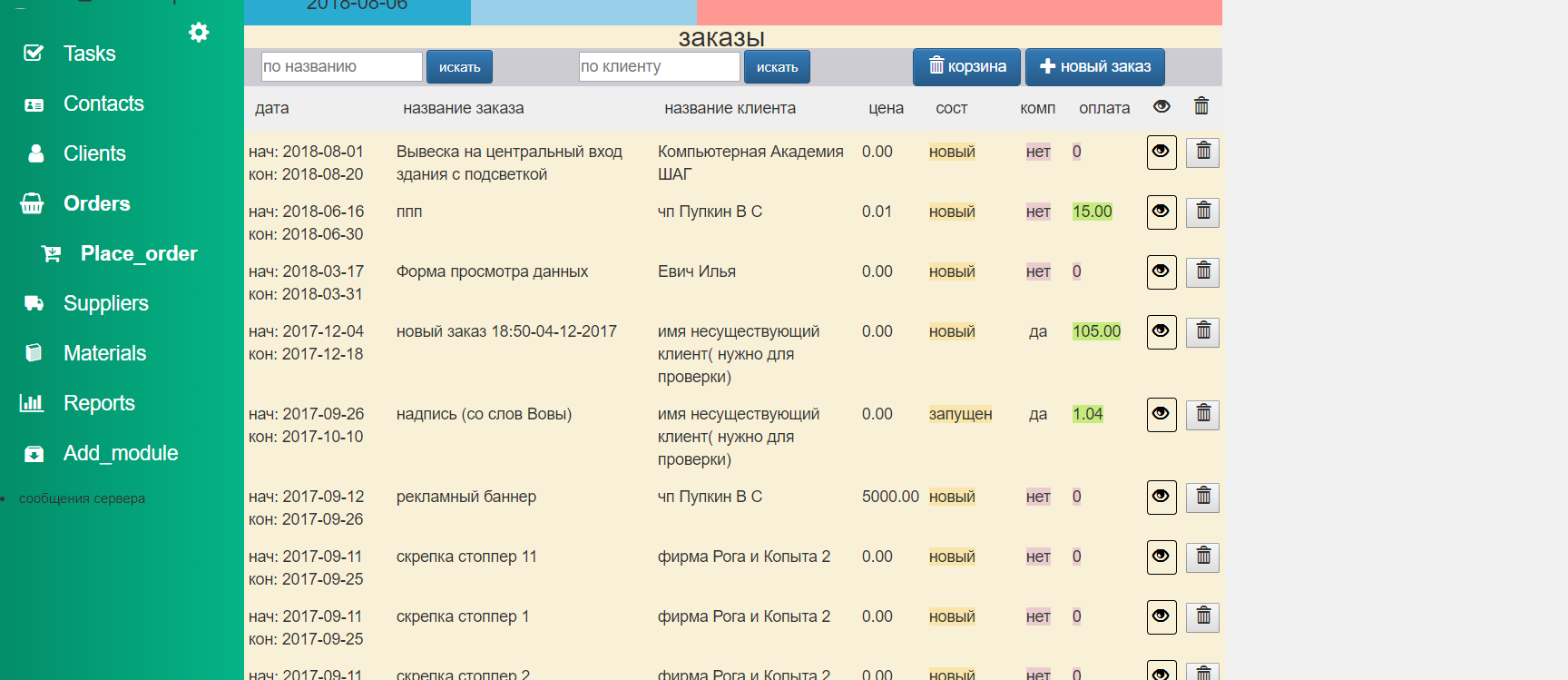


Рисунок 6.6 - Просмотр меню “заказы” с нашим вновь созданным заказом.

1.3. Работа с заказом.

1.3.1. Кратко о вкладке “Заказы”.

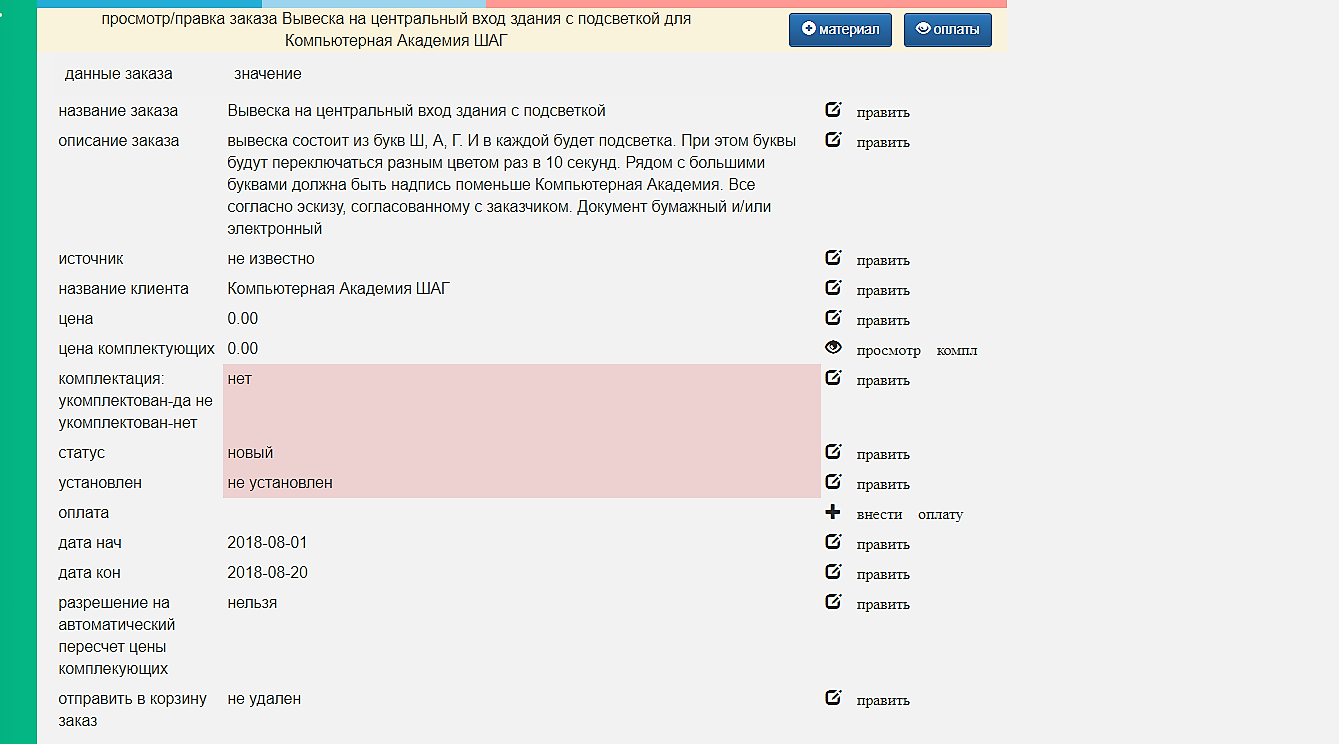
На вкладке “Заказы” мы видим список заказов. В зоне поиска можно найти заказ по клиенту или по названию заказа. Справа, напротив каждого заказа, есть 2 кнопки “Просмотр” и “Удалить”. Можно перейти к просмотру конкретного заказа или удалить его. Удаленный заказ можно найти через кнопку “Корзина”( находится в зоне поиска, рядом с кнопкой “Новый заказ”. Если вы удалите заказ, он исчезнет из вкладки заказы, но его можно найти в корзине и восстановить. Тогда он снова появится. И можно будет снова работать с ним… Давайте перейдем в “Просмотр заказа” и поглядим что можно делать с заказом.(рис.6.7.)

Рисунок 6.7 - Просмотр одного нашего нового “Заказа”.

1.3.2. Кратко о “Заказ”.

При отображении одного заказа мы можем видеть все данные касающиеся данного заказа. Если кликнуть справа “править”, появиться меню правки того раздела, справа от которого мы сделали клик. (рис. 6.8) ( Двойной клик по этому разделу непосредственно тоже вызовет меню правки. Выбор за пользователем, как вызывать меню правки конкретного поля.).

1.3.3. Правка поля ( раздела заказа). После вызова меню правки можно внести изменения и нажав кнопку “изменить” сохранить новые изменения. Можно нажать “передумал” и изменения не сохранятся.

1.3.4. Особенности некоторых пунктов.

1.3.4.1. Пункт “Укомплектован”. Пока наш заказ не укомплектован, можно по клику При изменении в пункте укомплектован на “укомплектован” вы не сможете добавлять новые материалы к заказу, а, значит, пункт меню “Просмотр Комплектующих” будет доступен только для просмотра и изменения количества уже добавленных материалов. Это служит некоторой подсказкой-предостережением для пользователя, чтобы он обратил дополнительное внимание на добавку и изменение количества материалов.

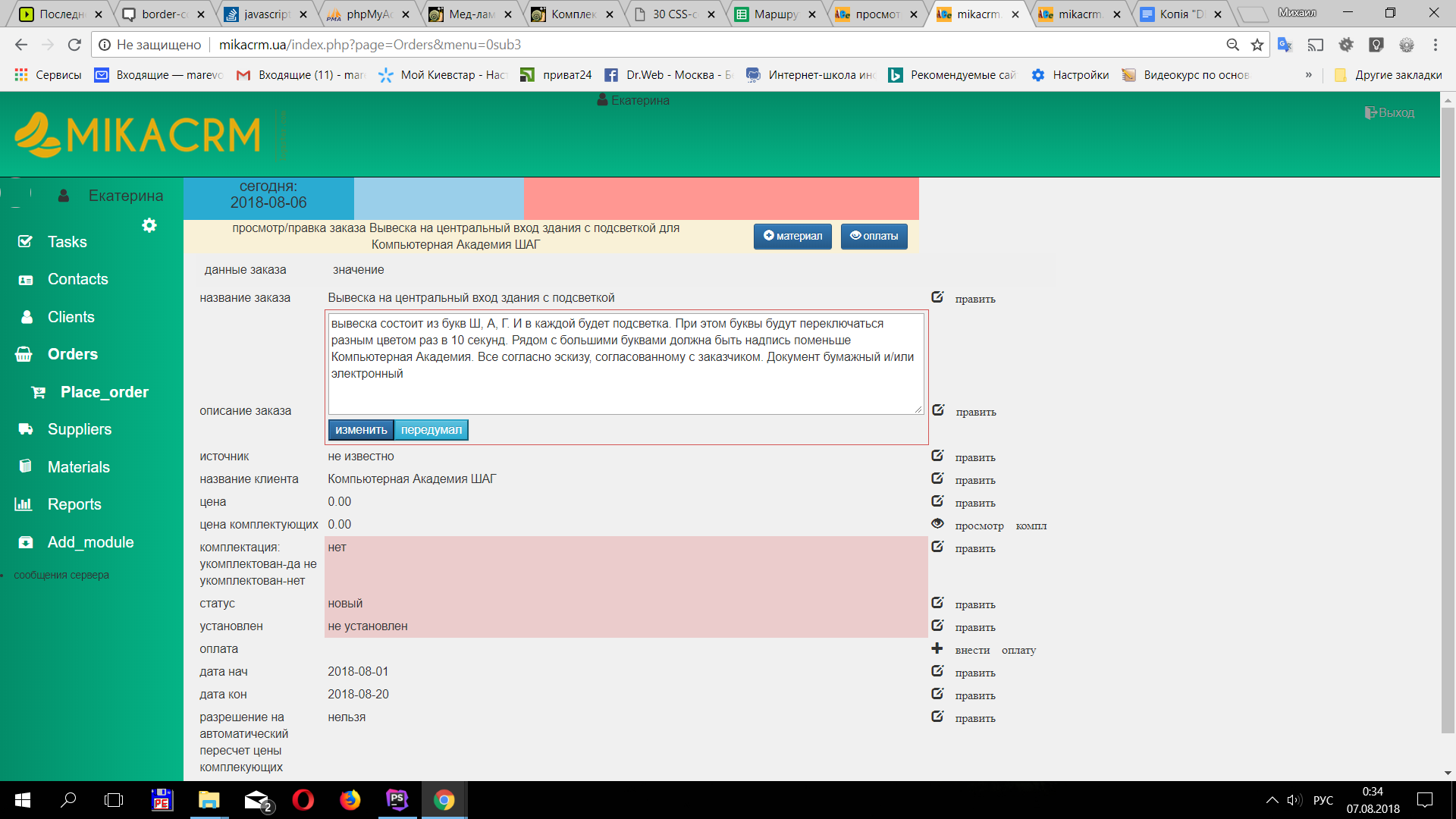


Рисунок 6.8 - Вызов меню правки нашего нового “Заказа”.

1.3.4.2. Пункт “Просмотр компл”.

При вызове этого пункта будет выведено модальное окно в котором отобразятся данные о материалах добавленных в заказ.(рис.6.9). Как видим, пока материалов нет. Давайте рассмотрим добавление материала.

1.3.4.2. Добавление материала в заказ.

Нажимаем кнопку “+материал” ( находится сверху слева) и нам будут выведены материалы, добавленные в базу материалов. С ними мы можем работать. (Если какого-то материала нет, то его сначала надо внести в базу. Это осуществляется по той же схеме, что и внесение клиента, заказа.)

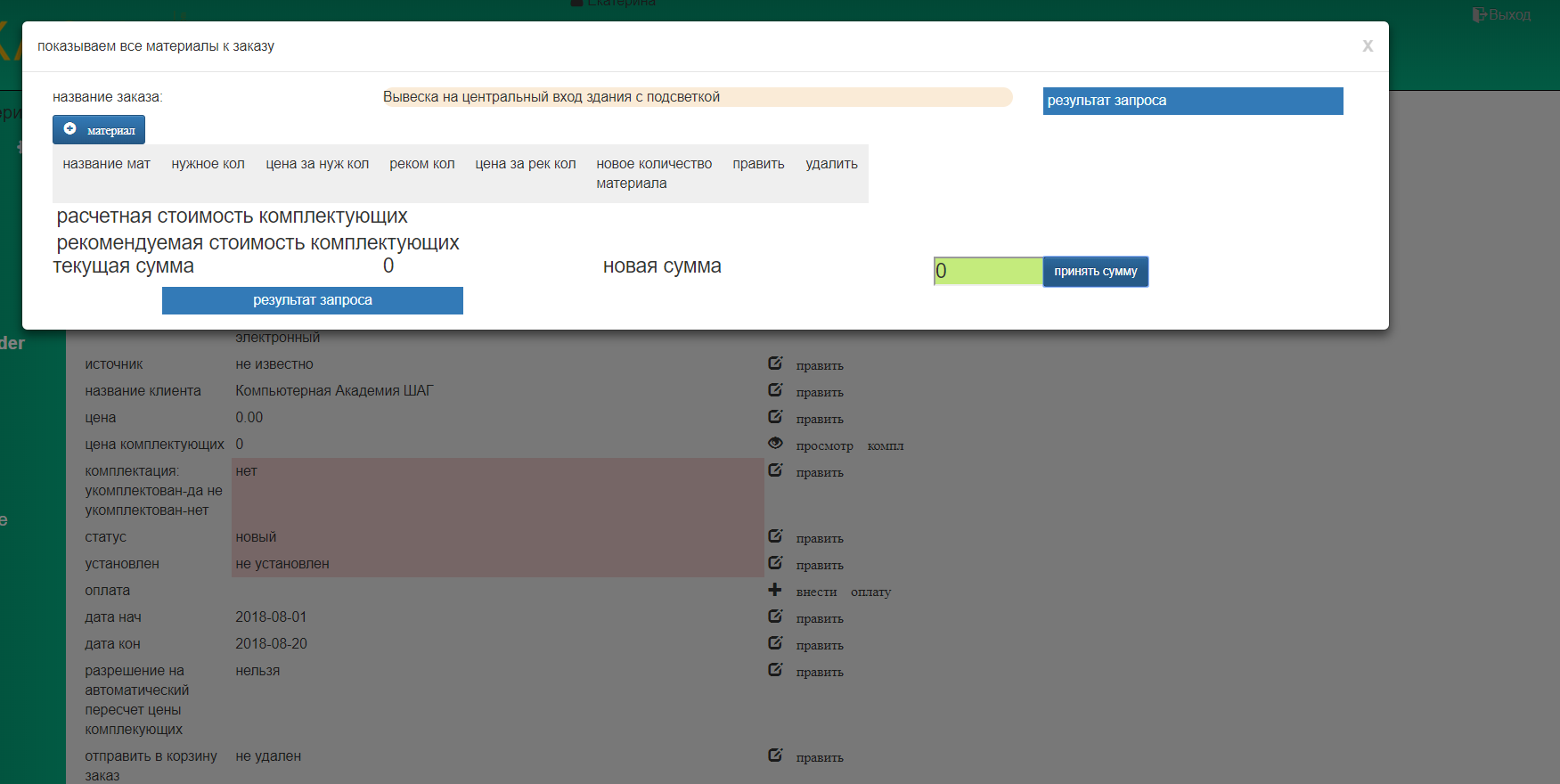


Рисунок 6.9 Показ материалов добавленных в заказ.

Если сделать двойной клик по строке с материалом, то выплывет строчка с данными этого материала и полем количества. (рис.6.10)

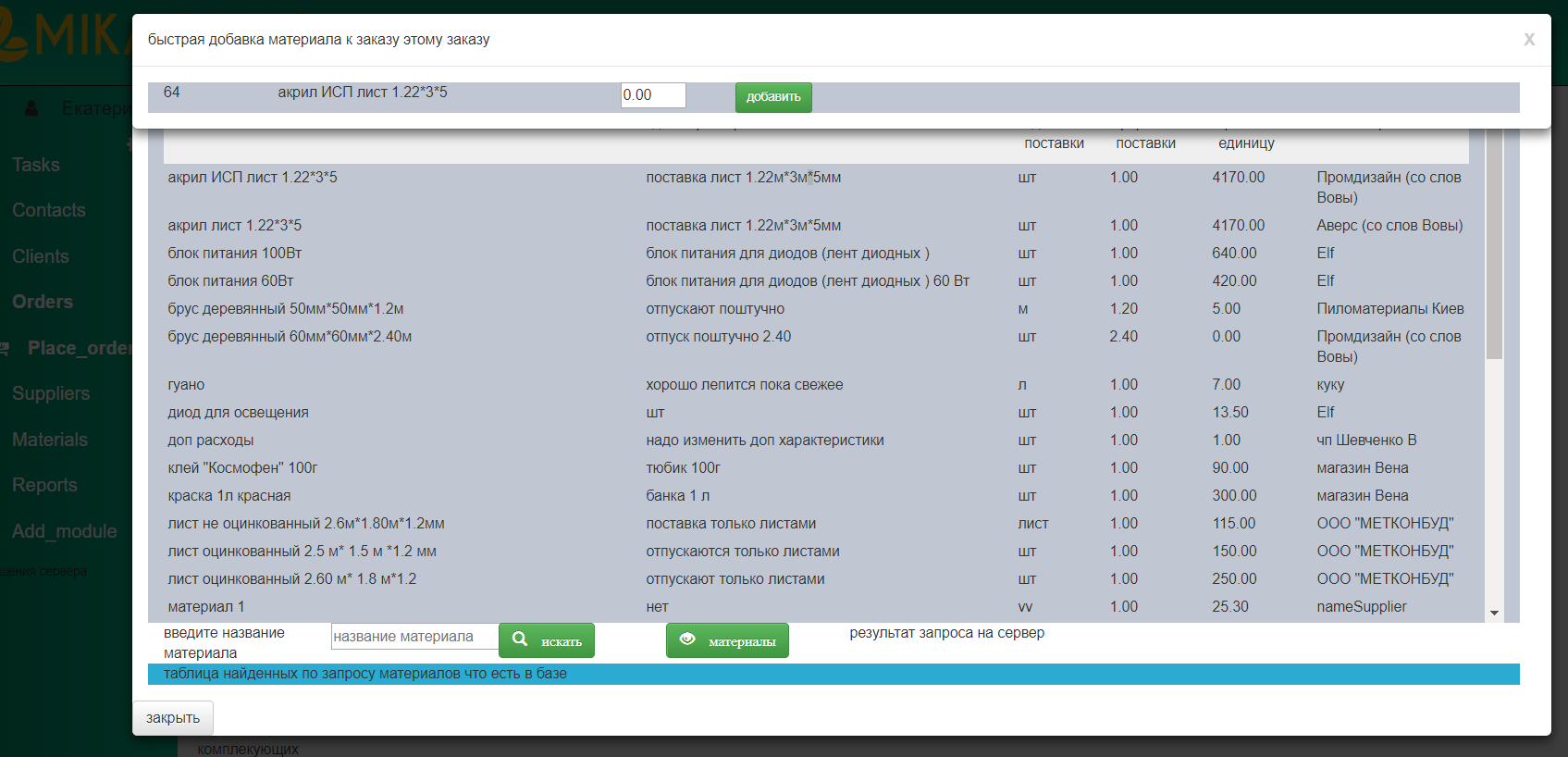


Рисунок 6.10. Быстрая добавка материала через двойной клик по строке с материалом.

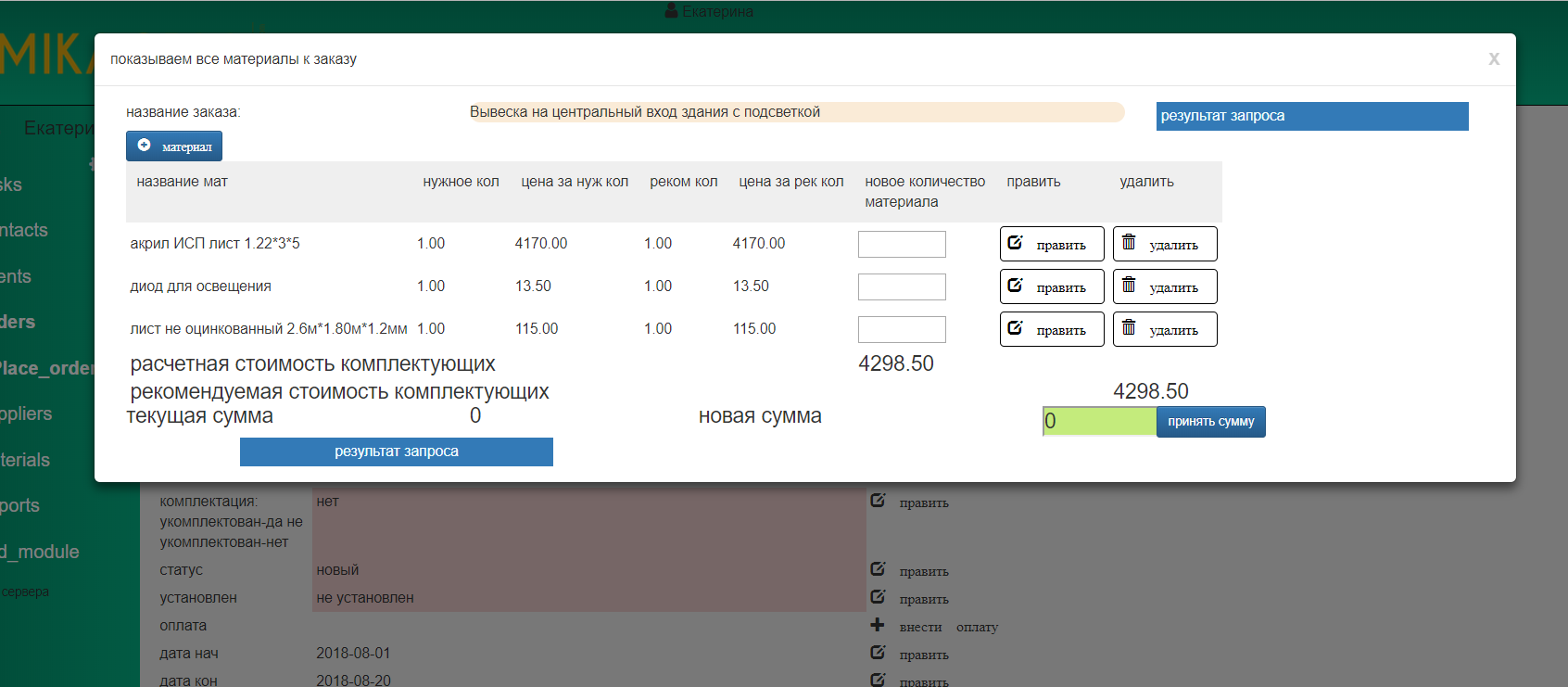
Выберем количество и кликнем рядом по “добавить”. Вылезет подсказка “успешно” и мы сможем по кнопке “просмотр материалов” увидеть все материалы, что добавлены в заказ. Давайте еще добавим парочку материалов, чтобы было с чем работать в следующей фазе, и вызовем “просмотр материалов” (рис.6.11) . Теперь можно перейти к работе с материалами в заказе.

Рис. 6.11. Просмотр материалов, добавленных в заказ.

1.3.4.2. Работа с материалами в заказе.

В окошке показаны все материалы, что есть в этом заказе, поля для изменения количества материалов и удаления материала из заказа. Так же есть графы “Расчетная стоимость комплектующих показывает сколько стоим то количество материала, что вы добавили. “Рекомендуемая стоимость” показывает, сколько реально нужно будет заплатить за эти материалы. К примеру нужны диоды для освещения. Они продаются поштучно. А если нужно какое-то количество листового материала, то придется закупать его минимальное количество, что может отпустить поставщик. Это нужно для понимания, что если надо третья часть листа для нашего заказа, то придется все равно купить лист. “Текущая сумма” за материалы вводится самим пользователем на основе вышеописанного анализа. Давайте для пробы добавим 20 диодов и удалим лист не оцинкованный, чтобы заменить его на лист оцинкованный.

Для этого в поле поиска в нижней части окна введем название того, что ищем, кликнем рядом “искать” и получим результат поиска. Дальше - как и раньше , то есть двойной клик по строке и так далее… ( рис. 6.12)

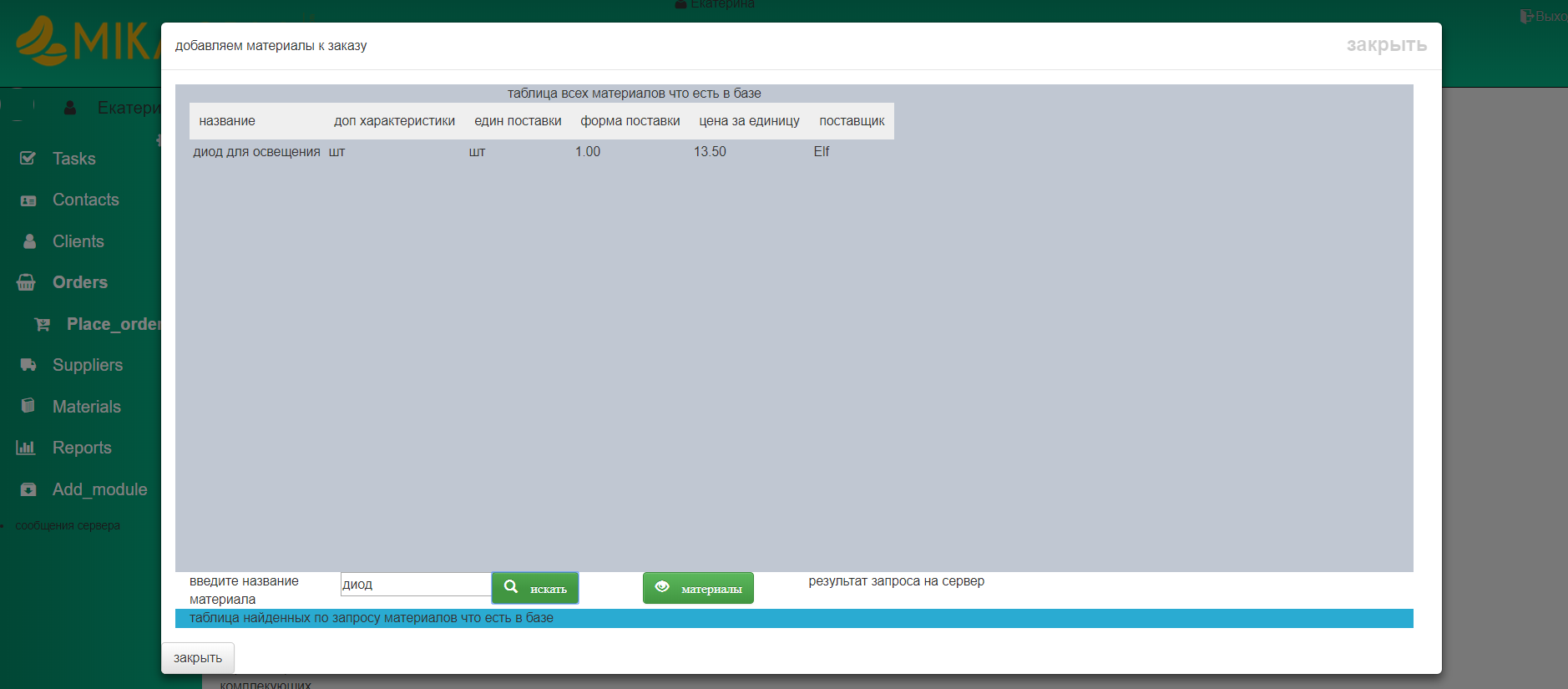


рис. 6.12. Поиск материала по названию и вывод для добавления в материалы заказа.

Так же поступим с оцинкованным листом, но предварительно из уже добавленных материалов к заказу, удалим не оционкованный лист. Просмотрим результат (рис. 6.13).

Таким образом мы можем добавлять и удалять материалы к заказу, менять их количество, и просматривать и изменять текущую сумму потраченых на материалы для конкретного заказа.

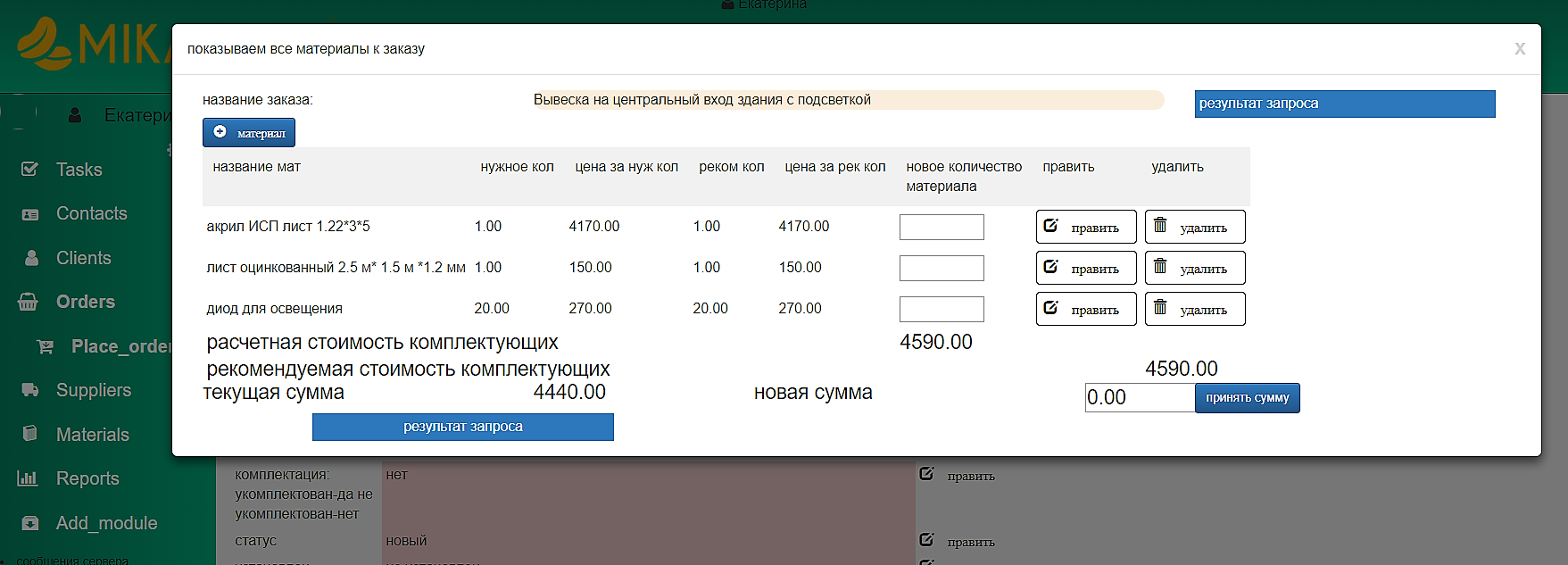


рис. 6.13. Результат удаления и добавления материалов в заказ.

Выводы

В данной курсовой работе была рассмотрена разработка программы для помощи учителям, родителям контроля посещаемости, успеваемости учеников начальной школы. В процессе разработки данной системы неоднократно встречались различные трудности, в первую очередь, связанные с освоением новой технологии Windows Forms, MySQL и Wcf-службы. Но благодаря тщательному изучению данной технологии все-таки удалось справиться со всеми трудностями.

Благодаря данной курсовой работе, были изучены принципы организации баз данных, приобретены огромные навыки по проектированию баз данных, а также по разработке и реализации систем управления базами данных с использованием Windows Forms и MySQL, а также развертывания службы Windows Communication Foundation (WCF).

Список использованной литературы

1. лекции КА «Шаг»
2. Статьи с сайта <https://msdn.microsoft.com/>
3. Статьи с сайта <https://ru.wikipedia.org/>
4. Статьи с сайта https://metanit.com/

Приложения

Приложение 1 Код wcf-службы.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Linq;

using System.ServiceProcess;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Net;

using System.Net.Mail;

using System.Windows.Forms;

namespace Wf\_Entity

{

partial class Service2 : ServiceBase

{

internal void TestStartupAndStop(string[] args)

{

this.OnStart(args);

this.OnStop();

}

private System.ComponentModel.IContainer components;

public Service2(string[] args)

{

InitializeComponent();

string eventSourceName = "MySource";

string logName = "MyNewLog";

if (args.Count() > 0)

{ eventSourceName = args[0]; }

if (args.Count() > 1)

{ logName = args[1]; }

eventLog1 = new System.Diagnostics.EventLog();

if (!System.Diagnostics.EventLog.SourceExists(eventSourceName))

{

System.Diagnostics.EventLog.CreateEventSource(eventSourceName, logName);

}

eventLog1.Source = eventSourceName;

eventLog1.Log = logName;

}

protected override void OnStart(string[] args)

{

// TODO: Добавьте код для запуска службы.

eventLog1.WriteEntry("In OnStart");

SendEmailAsync();//.GetAwaiter();

//Console.Read();

}

protected override void OnStop()

{

// TODO: Добавьте код, выполняющий подготовку к остановке службы.

eventLog1.WriteEntry("In onStop.");

}

private static void SendEmailAsync()

{

MailAddress // отправитель - устанавливаем адрес и отображаемое в письме имя

from = new MailAddress("katy-sp@mail.ru", "Administrator");

MailAddress to;

MailMessage m = new MailMessage();

using (team\_project\_3Entities2 connection = new team\_project\_3Entities2())

{

var parents = connection.parents.Select(p => new

{

Name = p.name\_parent,

Email = p.email\_parent

});

foreach (var k in parents)

{

// кому отправляем

try

{

to = new MailAddress(k.Email);

}

catch

{

continue;

}

name\_parent = k.Name;

email\_parent = k.Email;

// создаем объект сообщения

m = new MailMessage(from, to);

// тема письма

m.Subject = "Тест";

m.Body = "Уважаемый " + name\_parent;

var students = connection.parents.Where(p => p.name\_parent == name\_parent).FirstOrDefault().students;

foreach (var p in students)

{

if (p.progress == 1)

{

m.Body += "!\n Оценки вашего ребёнка " + p.name\_student + " :\n";

var name\_subject = from score in p.scores

join subject in connection.subjects on score.id\_subject equals subject.id\_subject

select new

{

Name = subject.name\_subject,

Data = score.date,

student\_scores = score.score

};

foreach (var n in name\_subject)

{

student\_name\_subject = n.Name;

m.Body += n.Data + ": " + student\_name\_subject + " - " + n.student\_scores + "\n";

}

// текст письма

id\_student = p.id\_student;

student\_name = p.name\_student;

}

if (p.attendance == 1)

{

try

{

var name\_subject = from be\_absentt in connection.be\_absent

join timetables in connection.timetable on be\_absentt.id\_time\_table equals timetables.id\_time\_table

join subject in connection.subjects on timetables.id\_subject equals subject.id\_subject

join number\_lesonn in connection.number\_lesson on timetables.lesson\_num equals number\_lesonn.lesson\_num

where be\_absentt.id\_student==p.id\_student

select new

{

Name = subject.name\_subject,

Data = be\_absentt.data,

Lesson\_start = number\_lesonn.time\_start\_lesson,

Lesson\_end = number\_lesonn.time\_end\_lesson

};

foreach (var n in name\_subject)

{

student\_name\_subject = n.Name;

m.Body += n.Data + " На " + student\_name\_subject + " - " + n.Lesson\_start + " - " + n.Lesson\_end + " Ваш ребенок отсутствовал" + "\n";

}

}

catch

{

continue;

}

}

}

// адрес smtp-сервера и порт, с которого будем отправлять письмо

SmtpClient smtp = new SmtpClient("smtp.mail.ru");

smtp.Port = 25;

smtp.Credentials = new NetworkCredential(from.Address, "2306ka");

smtp.EnableSsl = t

try

{

smtp.Send(m);

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

}

Приложение 2. Заполнение временной таблицы класса именами учеников и id учеников.

private void InserStudentIdNameToTableClassScoreAbsent()//загрузка в таблицу class\_score\_absent id\_student, name\_student

{

using (var con = new team\_project\_3Entities())

{//сделаем выборку сначала всех студентов выбранного класса

var query\_student =

from st in con.students

where st.id\_class == id\_class

orderby st.name\_student

select new { st.id\_student, st.name\_student };

//добавим в эту таблицу class\_score\_absent студентов и их имена

foreach (var it in query\_student)

{

var st\_sc\_ab = new class\_score\_absent

{

id\_student = it.id\_student,

name\_student = it.name\_student,

};

con.class\_score\_absent.Add(st\_sc\_ab);

}

con.SaveChanges();

//сделаем запрос прогулов из таблицы ( у нас есть дата, класс,надо еще выбрать урок- и, соответственно, id\_time\_table

}

}

Приложение 3. Вставка оценок учеников во временную таблицу

private void InsertScoreToTableClassScoreAbsent()//загрузка в таблицу всех оценок

{

using (var con = new team\_project\_3Entities())

{//считали все оценки за этот день для этого предмета для этого класса

var query\_score =

from score in con.scores

join stud in con.class\_score\_absent

on score.id\_student equals stud.id\_student

where score.id\_subject == id\_subject &&

score.date\_score == date\_score\_absent

select new { idSt = score.id\_student, sc = score.score, scT = score.score\_type };

dataGridView1.DataSource = query\_score.ToList();

foreach (var sc in query\_score)

{

foreach (var stud in con.class\_score\_absent)

{

if (stud.id\_student == sc.idSt)//выбрали студента

{

if (sc.scT == 1)

{

stud.score\_lesson = sc.sc;

}

if (sc.scT == 2)

{

stud.score\_samost = sc.sc;

}

if (sc.scT == 3)

{

stud.score\_kontr = sc.sc;

}

try

{

con.SaveChanges();

}

catch (Exception ex)

{

MessageBox.Show(ex.Data + "\n" + ex.Message + "\n" + ex.Source);

}

}

}

}

}

}

Приложение 4. Вставка во временную таблицу пропусков учеников

private void InsertAbsentToTableClassScoreAbsent()

{

if (id\_class == -1 || id\_subject == -1 || id\_time\_table == -1)

{

MessageBox.Show("сначала выберите класс и предмет");

return;

}

using (var con = new team\_project\_3Entities())

{

var query\_absent =

from be\_a in con.be\_absent

join stud in con.class\_score\_absent

on be\_a.id\_student equals stud.id\_student

where be\_a.id\_time\_table == id\_time\_table &&

be\_a.date\_absent == date\_score\_absent

select new { idSt = be\_a.id\_student, typeAbsent = be\_a.type\_absent };

// dataGridView1.DataSource = query\_absent.ToList(); //контроль по проверке прогулов в грид 1

foreach (var abs in query\_absent)

{

foreach (var stud in con.class\_score\_absent)

{

if (stud.id\_student == abs.idSt)

{

stud.type\_absent = abs.typeAbsent;

try

{

con.SaveChanges();

}

catch (Exception ex)

{

MessageBox.Show(ex.Data + "\n" + ex.Message + "\n" + ex.Source);

}

}

}

}

}

}