Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

филиал «Минский радиотехнический колледж»

ПРОГРАММНОЕ СРЕДСТВО АВТОМАТИЗАЦИИ РАБОЧЕГО МЕСТА МЕНЕДЖЕРА АТЕЛЬЕ ПО РЕМОНТУ БЫТОВОЙ ТЕХНИКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Базы данных и системы управления базами данных»

КП 32492.013102.081 ПЗ

Руководители / Виничук О.Н./

Разработчик /Лепетило В.В./

МИНСК 2016

Содержание

[Введение 2](#_Toc469625182)

[1 Постановка задачи 3](#_Toc469625183)

[1.1 Описание предметной области 3](#_Toc469625184)

[1.2 Цели и задачи проектирования 7](#_Toc469625185)

[2 Проектирование задачи 8](#_Toc469625186)

[2.1 Проектирование схем базы данных 8](#_Toc469625187)

[2.2 Алгоритм реализации задачи 12](#_Toc469625188)

[2.3 Описание сред реализации задачи 13](#_Toc469625189)

[3 Программная реализация 17](#_Toc469625190)

[3.1 Физическая структура базы данных 17](#_Toc469625191)

[3.2 Структура приложения 23](#_Toc469625192)

[4 Тестирование 27](#_Toc469625193)

[5 Описание применения 29](#_Toc469625194)

[Заключение 34](#_Toc469625195)

[Список использованных источников 35](#_Toc469625196)

[Приложение А Текст программы 36](#_Toc469625197)

[Приложение Б Диаграмма вариантов использования 53](#_Toc469625198)

[Приложение В Диаграмма деятельности 54](#_Toc469625199)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

*КП 32492.013102.081 ПЗ*

Разраб.

Лепетило В.В. 27М

Провер.

Виничук О.Н.

Реценз.

Н. Контр.

Утверд.

ПРОГРАММНОЕ СРЕДСТВО АВТОМАТИЗАЦИИ РАБОЧЕГО МЕСТА МЕНЕДЖЕРА АТЕЛЬЕ ПО РЕМОНТУ БЫТОВОЙ ТЕХНИКИ

Пояснительная записка

Лит.

Листов

*55*

МРК

# Введение

Тема курсового проекта – “Автоматизация рабочего места менеджера ателье по ремонту бытовой техники”.

В настоящее время невозможно представить работу практически любых видов деятельности без автоматизации. Она необходима для ускорения сложных операций, в том числе расчетов. Автоматизация исключает человеческий фактор, позволяет контролировать выполнение трудовых функций всего предприятия.

Проводить автоматизацию процессов следует для того, чтобы уберечь свою фирмы от лишних расходов. Некоторые руководители небольших фирм предпочитают справлять силами и умениями своих сотрудников. Но приобрести определенное программное обеспечение доступно многим, расходы на него быстро окупаются. Фирма в дальнейшем получает только прибыль.

Автоматизация особенно необходима на тех предприятиях, где численность сотрудников превышает семь человек. Руководителям невозможно контролировать большое количество работников без применения современных информационных систем.

Целью данного курсового проекта является разработка программного средства автоматизация рабочего места менеджера ателье по ремонту бытовой техники. Для достижения данной цели были поставлены следующие задачи:

- спроектировать базы данных;

- обеспечить связи между первичными и внешними ключами;

- осуществление добавления, редактирования, удаления записей из таблиц;

- возможность осуществлять поиск информации по заказам, клиентам, а также сотрудникам.

Объектом курсового проекта является автоматизация рабочего места по ремонту бытовой техники.

Предмет исследования – ремонт бытовой техники.

Процесс оформления заказов включает в себя множество повторяющихся действий, поэтому с помощью автоматизированной программы позволит существенно быстро увеличить работу пользователя. Оформление заказов, заполнение сведений о клиентах, сотрудниках, а также выполнять поиск по определённой информации, будут осуществляться без трудностей.

# 1 Постановка задачи

Необходимо разработать программное средство “Автоматизация рабочего места менеджера ателье по ремонту бытовой техники” с использованием языка запросов SQL, технологии Windows Forms на языке программирования C#. Программное средство должно выполнять следующие функции:

* добавление, редактирование, удаление информации о заказах, сотрудниках, клиентах;
* поиск информации в базе данных о сотрудниках, клиентах, заказах.

Интерфейс программного средства должен обладать следующими функциями:

* отображение записей, хранящихся в базе данных, в удобной для пользователя форме;
* меню, содержащее в себе основные возможности работы с программным средством.

Необходимо спроектировать базу данных которая обеспечит автоматизацию рабочего места ателье по ремонту бытовой техники.

1.1 Описание предметной области

Сервисный центр по ремонту гарантийной бытовой техники является предприятием, которое выполняет разноплановую починку техники по гарантии заказчика.

В настоящее время для сервисного обслуживания оборудования активно используются специализированные сайты. Данная технология очень удобна, так как позволяет любому человеку воспользоваться услугами сервисного центра. Но в то же время всегда существует опасность «взлома» сайта, что может сделать его недоступным для нуждающихся пользователей, а также может произойти кража конфиденциальной информации, что может повлечь за собой негативные последствия в виде разглашения данной информации или её использования в недобросовестных целях.

Для обеспечения оперативности обработки информации о наличии продукции, заказах, клиентах и оплате необходима автоматизированная система, основанная на базе данных. Её использование сократит время, требуемое на обслуживание клиентов.

Программа будет доступна только для менеджера сервисного центра, позволяя ему вести записи о поступающих заказах, о сроках их выполнения, о поступающей техники, её производителе.

Входная информация — информация, которую получает устройство. Выходная информация — информация, которая получается после обработки устройством.

Входными данными программного средства являются:

* критерии поиска;
* информация, вводимая менеджером (информация о клиентах, заказах, сотрудниках, категориях товаров и фирмах-производителей);
* системная информация (дата, время).

Выходная информация:

* результаты поиска базы данных;
* записи, хранящиеся в базы данных, отображаемые в программе.

Постоянными данными являются:

* структуры базы данных;
* справочная информация.

Для определения преимуществ и недостатков разрабатываемого программного средства были выбраны 3 его аналога:

- “Учёт заказов”;

- “Менеджер СТО”;

- “Учёт заказов и клиентов в режиме нон-стоп”.

Программа “Учёт заказов” – программный комплекс для автоматизации работы организаций, чья деятельность связана с доставкой товаров по заказам или реализации услуг по телефону. Система ориентирована на модель доставки по телефону, с использованием возможностей цифровой АТС, хотя возможна и работа без телефонного модуля, например при заявках с сайта, или непосредственно при личном визите клиента. В состав комплекса входят следующие модули: модуль работы с заказами, модуль складского учета, справочник продуктов/услуг, настройки системы, подсистема репликации данных между удаленными точками.

Главное окно программного средства изображено на рисунке 1.1.

Достоинства:

* многопользовательский режим работы с разделением прав доступа;
* статистика по заказам;
* возможность работы на нескольких удаленных точках с репликацией данных между точками.

Недостатки:

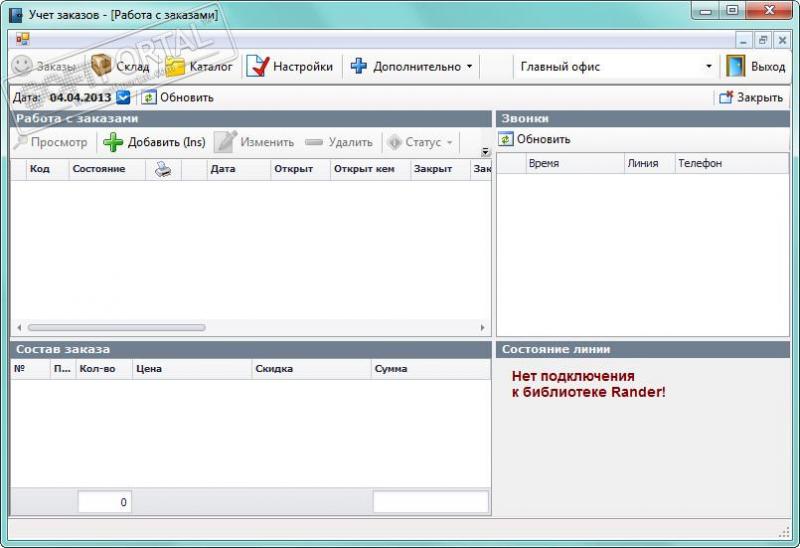
* программа является Demo версией, что в последствии несёт множество ошибок и неисправностей в приложении;
* программа является платной.

Рисунок 1.1 – Главное окно программы “Учёт заказов”

Программа “Менеджер СТО” – это программа для учёта заказов и клиентов автосервисного центра.

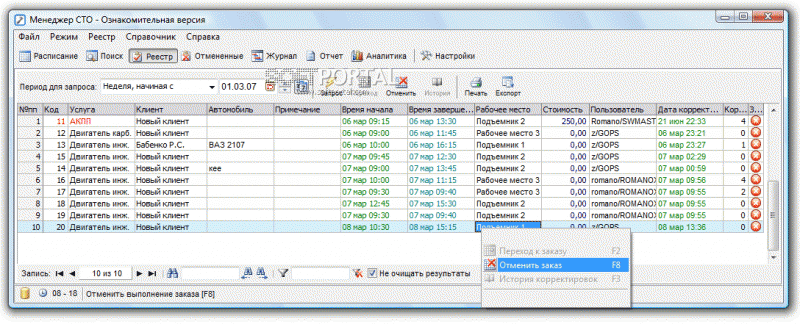
Главное окно программы изображено на рисунке 1.2.

Рисунок 1.2 – Главное окно программы «Менеджер СТО»

Достоинства:

* заказы можно передать по сети;
* присутствует показатель активности;
* надёжная защита.

Главным недостатком рассматриваемого аналога является его коммерческая версия, за которую необходимо платить.

Программа “Учёт заказов и клиентов в режиме нон-стоп” – это программа позволяет быстро оформлять заказы и сведения о клиентах, и сохранять их в базу данных.

Главное окно программы изображено на рисунке 1.3.

Достоинства:

* учёт в онлайн режиме;
* готовые шаблоны, таблицы, формы документов;
* возможность оформления документов за 1-2 минуты;
* возможность учёта каждого этапа обработки заказа.

Недостатки:

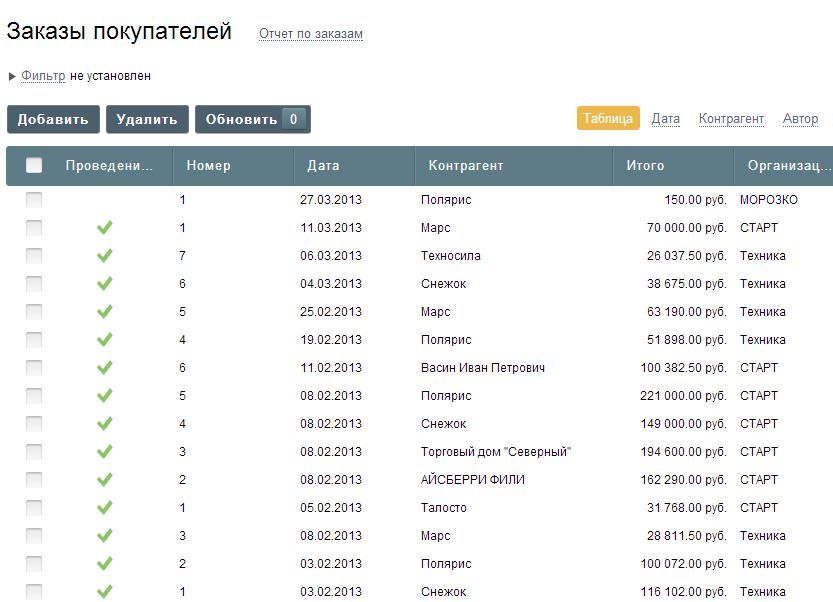
* присутствие ошибок в программе;
* программа является платной.

Рисунок 1.3 – Главное окно программы «Учёт заказов и клиентов в режиме нон-стоп»

В процессе рассмотрения аналогов разрабатываемого программного средства были выделены следующие преимущества:

* работа с программой без использования интернета, что позволит повысить безопасность хранящихся данных;
* простота интерфейса программы, которая позволит пользователю быстро освоить все доступные функции;
* удобная справочная система, которая поможет начинающему пользователю быстро адаптироваться к программе.

Главным недостатком программы является невозможность её использования из любой точки мира без предварительной установки.

1.2 Цели и задачи проектирования

Целью данного курсового проекта является разработка программного средства автоматизация рабочего места менеджера ателье по ремонту бытовой техники. Программное средство должно осуществлять:

* выдачу информации о заказах, клиентах, сотрудниках;
* добавление заказов;
* редактирование информации;
* поиск по клиентам, сотрудникам и заказам.

Программа должна иметь простой и понятный интерфейс. Интерфейс программного средства будет содержать:

* элементы управления;
* окно, в котором будет отображаться информация, хранящаяся в базе данных;
* комбинированные цвета, благоприятные для восприятия пользователем.

Необходимо разработать две диаграммы:

* диаграмма деятельности;
* диаграмма вариантов использования.

Задачи проектирования:

- создать базу данных которая обеспечит автоматизацию рабочего места ателье по ремонту бытовой техники;

- обеспечить атрибуты первичными и внешними ключами для связи между таблицами;

- создать запросы на добавление, удаление, изменения и поиск записей в таблицах базы данных.

Программное средство должно содержать справку, с помощью которой будет происходить быстрое освоение и правильное использование элементов управления и самой программы.

2 Проектирование задачи

2.1 Проектирование схем базы данных

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

- корректность схемы БД, то есть база должна быть гомоморфным образом моделируемой предметной области, где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных;

- обеспечение ограничений (на объёмы внешней и оперативной памяти и другие ресурсы вычислительной системы);

- эффективность функционирования (соблюдение ограничений на время реакции системы на запрос и обновление данных);

- защита данных (от аппаратных и программных сбоев и несанкционированного доступа);

- простота и удобство эксплуатации;

- гибкость, то есть возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Логическая структура БД всегда является некоторой реализацией модели данных СУБД для конкретной предметной области.

Логическая модель представлена двенадцатью отношениями, диаграмма «Сущность-связь» представлена на рисунке 2.1:

* фирма;
* категория
* товары;
* заказы;
* гарантии;
* клиенты;
* вид ремонта;
* сотрудники;
* детали;
* пол;
* должности;
* модели.

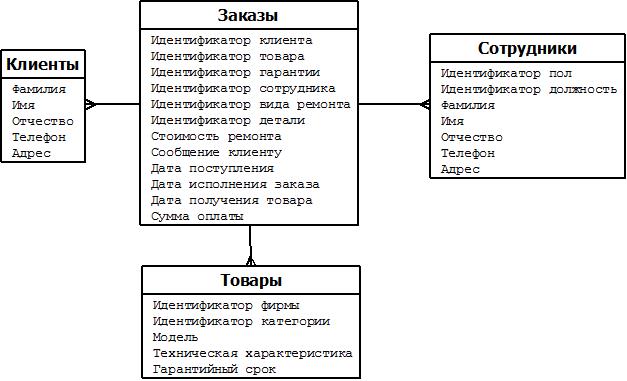


Рисунок 2.1 – Диаграмма «Сущность-связь»

Каждое отношение включает в себя множество атрибутов.

Атрибут - это поименованный столбец отношения.

В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение обычно имеет вид двумерной табли­цы, в которой строки соответствуют отдельным записям, а столбцы — атрибутам. При этом атрибуты могут располагаться в любом порядке — независимо от их переупорядочивания отношение будет оставаться одним и тем же, а потому иметь тот же смысл. Схема базы данных представлена на рисунке 2.2.

Отношение “Фирма” включает следующие атрибуты:

- идентификатор фирмы;

- название фирмы.

Отношение “Категории” включает следующие атрибуты:

- идентификатор категории;

- название категории.

Отношение “Товары” включает следующие атрибуты:

- идентификатор товаров;

- идентификатор фирмы;

- идентификатор категории;

- название модели;

- технические характеристики;

- гарантийный срок.

Отношение “Заказы” включает следующие атрибуты:

- идентификатор Заказа;

- идентификатор клиента;

- идентификатор товара;

- идентификатор гарантия;

- идентификатор исполнения заказа;

- идентификатор вид ремонта;

- идентификатор сотрудник;

- идентификатор деталь;

- стоимость ремонта;

- сообщение клиенту;

- дата исполнения заказа

- дата получения товара;

- сумма оплаты;

- дата поступления.

Отношение “Гарантия” включает следующие атрибуты:

- идентификатор гарантии;

- дата гарантии.

Отношение “Клиенты” включает следующие атрибуты:

- идентификатор клиента;

- фамилия;

- имя;

- отчество;

- телефон;

- адрес.

Отношение “Детали” включает следующие атрибуты:

- идентификатор детали;

- название детали;

- количество.

Отношение “Сотрудники” включает следующие атрибуты:

- идентификатор сотрудника;

- идентификатор пола;

- идентификатор должность;

- фамилия;

- имя;

- отчество;

- телефон;

- адрес.

Отношение “Виды ремонта” включает следующие атрибуты:

- идентификатор вид ремонта;

- вид ремонта.

Отношение “Должности” включает следующие атрибуты:

- идентификатор должности;

- название должности.

Отношение “Пол” включает следующие атрибуты:

- идентификатор пол;

- название пола.

Отношение “Модели” включает следующие атрибуты:

- идентификатор модель;

- название модели.

Далее будут расставлены связи между отношениями.

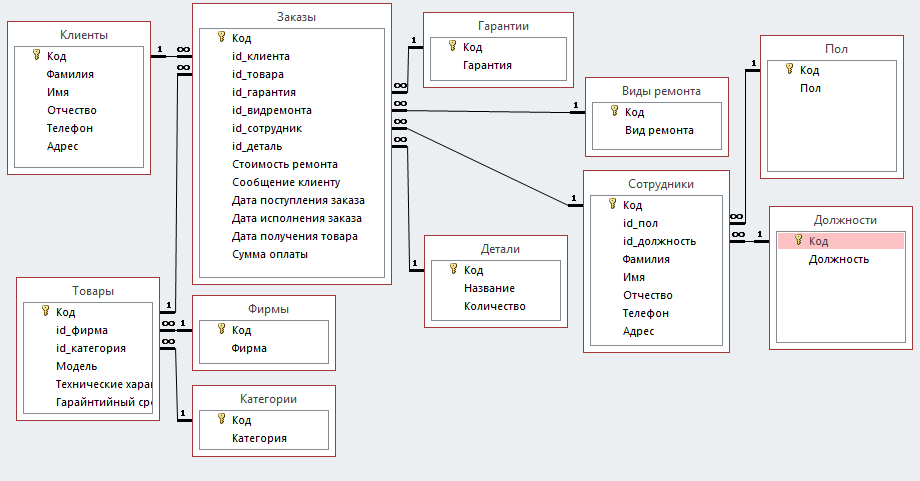


Рисунок 2.2 – Схема базы данных

Связь – это некоторая ассоциация между двумя отношениями. Связи позволяют по одному отношению находить другие отношения, связанные с ними.

Связи между таблицами организуются с помощью различных ключей. Связное отношение хранит ключи двух или более объектных отношений, по этим ключам устанавливаются связи между объектами. Связное отношение кроме связывае­мых ключей может иметь и другие атрибуты, которые будут функционально зависеть от этой связи. Ключи в связных отношениях называются внешними ключами, поскольку они являются первичными ключами других отношений [2].

Существует четыре вида связей (отношений между таблицами):

– один к одному;

– один ко многим;

– многие к одному;

– многие ко многим.

Связь один к одному означает, что каждой записи первой таблицы соответствует только одна, связанная с ней запись второй таблицы и наоборот. Такой тип отношений используется крайне редко, так как фактически все данные могут быть помещены в одной таблице. Он полезен тогда, когда одну громоздкую таблицу целесообразно разделить на две с логически сгруппированными полями.

Связь один ко многим характерна тем, что запись одной таблицы связана с несколькими записями другой таблицы. В то же время запись второй таблицы не может быть связана более чем с одной записью первой таблицы.

Связь многие ко многим или непрямая табличная связь определяет связь одной записи первой таблицы с несколькими записями второй таблицы, в то же время как одна запись второй таблицы может быть связана с несколькими записями первой таблицы. На практике такой сложный вид связи между двумя таблицами реализуется через промежуточную (связующую) таблицу. Она позволяет заменить одну связь вида многие ко многим на две последовательные связи: многие к одному и один ко многим, которые реализуются проще.

## 2.2 Алгоритм реализации задачи

Алгоритм – это система последовательных операций (в соответствии с определенными правилами) для решения какой-нибудь задачи.

Виды алгоритмов:

* линейный алгоритм — набор команд (указаний), выполняемых последовательно во времени друг за другом;
* разветвляющийся алгоритм — алгоритм, содержащий хотя бы одно условие, в результате проверки которого может осуществляться разделение на несколько альтернативных ветвей алгоритма;
* циклический алгоритм — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.

Неотъемлемым свойством алгоритма является его результативность, то есть алгоритмическая инструкция лишь тогда может быть названа алгоритмом, когда при любом сочетании исходных данных она гарантирует, что через конечное число шагов будет обязательно получен результат.

Блок-схема — графическое представление алгоритма. Она состоит из функциональных блоков, которые выполняют различные назначения (ввод/вывод, начало/конец, вызов функции и т.д.).

Структурная блок-схема — графическое изображение алгоритма в виде схемы, связанных между собой с помощью стрелок (линий перехода) блоков — графических символов, каждый из которых соответствует одному шагу алгоритма.

Алгоритм начнёт свою работу после запуска приложения. Будет показано окно авторизации. После успешной авторизации будет предоставлен выбор в виде иконки и названия таблицы, по нажатию которой будет осуществляться переход к таблице с заполненными данными. Записи редактируются по нажатии соответствующей иконки. После просмотра или редактирования таблицы осуществляется переход в главное меню. Далее можно выбрать другую таблицу или выйти из программного средства [3].

## 2.3 Описание сред реализации задачи

Для написания программы была выбрана среда программирования Visual Studio 2015 и язык программирования C#, а для создания базы данных использовалась свободная реляционная система управления базами данных - MySQL.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы [1].

Начиная с версии 5.0, СУБД MySQL практически полностью удовлетворяет стандарту структурированного языка запросов SQL и, следовательно, совместима с другими базами данных, а также стали доступными следующие важные механизмы:

- хранимые процедуры и функции, объединяющие в себе целые последовательности запросов;

- триггеры, то есть хранимые процедуры, привязанные к событию изменения таблицы;

- представления – выборки данных, которые можно представить как полноценные реально существующие таблицы базы данных;

- курсоры, позволяющие цикле просмотреть каждую строку результирующей таблицы запросов;

- информационная схема, то есть переносимый набор представлений системной таблицы, в которой хранится разнообразная внутренняя информация;

- обработчики ошибок;

- множество новых функций.

Основные достоинства СУБД MySQL:

- высокое качество – MySQL характеризуется устойчивой работой;

- наряду с Oracle, MySQL считается одной из самых быстрых СУБД в мире;

- открытый код доступен для просмотра и модернизации, что поз-воляет постоянно улучшать программный продукт;

- СУБД MySQL, разработанная с использованием языков C/C++, протестирована на многих платформах, среди которых Windows, Linux, FreeBSD, Mac OS X, OS/2, Solaris и другие;

- MySQL поддерживает API (Application Programming Interface, программный интерфейс приложения) для С, C++, Eiffel, Java, Perl, PHP, Python, Ruby и Tcl. MySQL можно успешно применять как для построения Web-страниц с использованием Perl, PHP и Java, так и для работы прикладной программы, созданной с использованием Delphi, Builder C++ или платформы .NET;

- СУБД MySQL предоставляет широкий выбор типов таблиц, в том числе и сторонних разработчиков, что позволяет реализовать оптимальную для решаемой задачи производительность и функциональность;

- локализация в MySQL выполнена корректно. У пользователя, как правило, не возникает проблем при обработке русского содержимого БД.

В стандартном дистрибутиве MySQL поставляются клиентские про-граммы (утилиты), взаимодействующие с MySQL-сервером: mysql (кон-сольный клиент для доступа к MySQL-серверу, позволяющий выполнять SQL-запросы), mysqladmin (утилита для выполнения административных функций, таких как создание или удаление баз данных, получение информа-ции о процессах, состоянии сервера и т. п.), mysqldump (утилита для вывода содержимого базы данных MySQL в виде текстового файла с SQL-операторами), mysqlimport (выполняет перенос информации из текстового файла в таблицы баз данных) и mysqlshow (отображает информацию о существующих базах данных, таблицах, полях и индексах).

К утилитам, которые могут функционировать без подключения к сер-веру MySQL, относятся: myisampack (сжимает таблицы типа MylSAM, уменьшая их в размере и делая доступными только для чтения), mysqlcheck (утилита, используемая для описания, проверки, оптимизации и восстановления таблиц), mysqlbinlog (данная утилита используется для чтения содержимого журнала двоичной регистрации при восстановлении данных в нештатных ситуациях) и perror (утилита, которая выводит рас-шифровку кодов системных ошибок и ошибок MySQL).

Среда разработки Visual Studio предоставляет необходимый инструментарий для эффективного и быстрого создания приложений с графическим интерфейсом.

C# — объектно-ориентированный язык программирования. Основные преимущества языка C#:

* простота, он относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java;
* объектно-ориентирован, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы;
* поддержка Visual C# в Visual Studio реализована в виде полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки.

Синтаксис C# делает проще то, что было сложно в С++, и обеспечивает мощные возможности, такие как типы значений Nullable, перечисления, делегаты, лямбда-выражения и прямой доступ к памяти, чего нет в Java. С# поддерживает универсальные методы и типы, обеспечивая более высокий уровень безопасности и производительности, а также итераторы, позволяющие при реализации коллекции классов определять собственное поведение итерации, которое может легко использоваться в клиентском коде.

Как объектно-ориентированный язык C# поддерживает понятия инкапсуляции, наследования и полиморфизма. Все переменные и методы инкапсулируются в определения классов. Класс может наследовать непосредственно из одного родительского класса, но может реализовывать любое число интерфейсов.

# 3 Программная реализация

## 3.1 Физическая структура базы данных

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Этапы физического проектирования баз данных:

- перенос глобальной логической модели данных в среду целевой СУБД;

- проектирование основных отношений;

- разработка способов получения производных данных;

- реализация ограничений предметной области;

- проектирование физического представления базы данных;

- анализ транзакций;

- выбор файловой структуры;

- определение индексов;

- определение требований к дисковой памяти;

- проектирование пользовательских представлений;

-разработка механизмов защиты;

- обоснование необходимости введения контролируемой избыточности;

- текущий контроль и настройка операционной системы.

Физическое проектирование баз данных включает шесть основных этапов. Концептуальное и логическое проектирование охватывает три первых этапа разработки баз данных, а физическое проектирование — этапы с 4 по 9 этапы.

Этап 4. “Реализация ограничений предметной области” - стадии физического проектирования включает разработку основных отношений и реализацию ограничений предметной области с использованием доступных функциональных средств целевой СУБД, На этом этапе должно быть также принято решение по выбору способов получения производных данных, которые включены в модель данных.

Этап 5. “Проектирование физического представления базы данных” - включает выбор файловой организации и индексов для основных отношений. Как правило, СУБД для персональных компьютеров имеют фиксированную структуру внешней памяти, а другие СУБД предоставляют несколько альтернативных вариантов файловой организации для хранения данных. С точки зрения пользователя организация внутренней структуры хранения отношений должна быть совершенно прозрачной — пользователь должен иметь возможность получать доступ к любому отношению и к отдельным его строкам без учета способа хранения данных. Это означает, что СУБД должна обеспечивать полную независимость физического хранения данных от их логической организации. Только в этом случае внесение изменений в физическую организацию базы данных не окажет никакого влияния на работу пользователей. Соответствие между логической моделью данных и физической моделью данных определяется внутренней схемой базы данных. Разработчик должен предоставить подробные физические проекты базы данных с учетом применяемой СУБД и операционной системы. В проекте реализации базы данных в СУБД разработчик должен определить структуры файлов, которые будут использоваться для представления каждого отношения. В проекте реализации базы данных в операционной системе разработчик должен указать расположение отдельных файлов и обеспечить необходимую их защиту.

Этап 6. “Анализ транзакций” - необходимо принять решение о том, как должно быть реализовано каждое пользовательское представление.

Этапе 7. “Выбор файловой структуры” - осуществляется проектирование средств защиты, необходимых для предотвращения несанкционированного доступа к данным, включая управление доступом к основным отношениям.

Этап 8. “Определение индексов” - анализируется также необходимость снижения уровня требований нормализации данных в логической модели, что может способствовать повышению общей производительности системы. Однако эти действия следует предпринимать только в случае реальной необходимости, поскольку введение в базу данных избыточности неизбежно вызовет появление проблем с поддержанием целостности данных.

Этап 9. “Определение требований к дисковой памяти” - описан способ организации текущего контроля операционной системы, позволяющий своевременно обнаруживать и устранять все проблемы производительности, которые могут быть решены на уровне проекта, а также учитывать новые или изменившиеся требования.

Физическая структура базы данных представлена следующими таблицами:

- “categories”;

- “clients”;

- “detali”;

- “dolgnosti”;

- “firms”;

- “garantii”;

- “models”;

- “sex”;

- “sotrudniki”;

- “tovary”;

- “zakazy”;

- “categories”;

- “vidremonta”.

Таблица “categories” хранит категории товаров, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.1.

Таблица 3.1 - Физическая структура таблицы “categories”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_category | int | первичный ключ |
| category | text | название категории |

Таблица “clients” хранит информацию о клиентах, содержит 6 атрибутов. Описание, тип и название атрибутов описано в таблице 3.2.

Таблица 3.2 - Физическая структура таблицы “clients”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_client | int | первичный ключ |
| familiya | text | фамилия |
| imya | text | имя |

Продолжение таблицы 3.2

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| otchestvo | text | отчество |
| telephon | text | телефон |
| addres | text | адрес |

Таблица “detali” хранит информацию о названии детали и сколько их имеется, содержит 3 атрибута. Описание, тип и название атрибутов описано в таблице 3.3.

Таблица 3.3 - Физическая структура таблицы “detali”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_detal | int | первичный ключ |
| nazvanie | text | название детали |
| kolichesstvo | int | количество деталей |

Таблица “dolgnosti” хранит названия должностей, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.4.

Таблица 3.4 - Физическая структура таблицы “dolgnosti”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_dolgnost | int | первичный ключ |
| nazvanie | text | название должности |

Таблица “firms” хранит название фирм, производителей товаров, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.5.

Таблица 3.5 - Физическая структура таблицы “firms”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_firm | int | первичный ключ |
| firma | text | название фирмы |

Таблица “garantii” хранит даты гарантий, содержит 4 атрибута. Описание, тип и название атрибутов описано в таблице 3.6.

Таблица 3.6 - Физическая структура таблицы “garantii”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_garantiya | int | первичный ключ |
| garantiyniy\_srok | date | гарантийный срок |

Продолжение таблицы 3.6

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| nazvanie\_centra | text | название центра |
| addres | text | адрес центра |

Таблица “models” хранит модели бытовой техники, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.7.

Таблица 3.7 - Физическая структура таблицы “models”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | описание |
| id\_model | int | первичный ключ |
| nazvanie | text | название можели |

Таблица “sex” хранит названия пола, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.8.

Таблица 3.8 - Физическая структура таблицы “sex”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_ sex | int | первичный ключ |
| sex | text | название пола |

Таблица “sotrudniki” хранит информацию о сотрудниках, содержит 8 атрибутов. Описание, тип и название атрибутов описано в таблице 3.9.

Таблица 3.9 - Физическая структура таблицы “sotrudniki”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_sotrudnik | int | первичный ключ |
| id\_sex | int | внешний ключ таблицы “sex” |
| id\_dolgnost | int | внешний ключ таблицы “dolgnosti” |
| familiya | text | фамилия сотрудника |
| imya | text | имя сотрудника |
| otchestvo | text | отчество сотрудника |
| telephon | text | телефон сотрудника |
| addres | text | адрес сотрудника |

Таблица “tovary” хранит информацию о товарах, содержит 6 атрибутов. Описание, тип и название атрибутов описано в таблице 3.10.

Таблица 3.10 - Физическая структура таблицы “tovary”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_tovar | int | первичный ключ |
| id\_firm | int | внешний ключ таблицы “firms” |
| id\_category | int | внешний ключ таблицы “categories” |
| id\_model | int | внешний ключ таблицы “models” |
| seriyniy\_nomer | text | серийный номер |
| tehnichiskie\_haracteristici | text | технические характеристики |
| garantiyniy\_srok | date | гарантийный срок |

Таблица “zakazy” хранит заказы клиентов, содержит 14 атрибутов. Описание, тип и название атрибутов описано в таблице 3.11.

Таблица 3.11 - Физическая структура таблицы “zakazy”

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_zakaz | int | первичный ключ |
| id\_client | int | внешний ключ таблицы “clients” |
| id\_tovar | int | внешний ключ таблицы “tovary” |
| id\_garantiya | int | внешний ключ таблицы “garantii” |
| id\_vidremont | int | внешний ключ таблицы “[vidremonta](http://localhost/PHPMyAdmin.by/sql.php?server=1&db=remontbytovoytekhniki&table=vidremonta&pos=0&token=9aab269930c4705d244a3de909ea71e1)” |
| id\_sotrudnik | int | внешний ключ таблицы “[sotrudniki](http://localhost/PHPMyAdmin.by/sql.php?server=1&db=remontbytovoytekhniki&table=sotrudniki&pos=0&token=9aab269930c4705d244a3de909ea71e1)” |
| id\_detal | int | внешний ключ таблицы “[detali](http://localhost/PHPMyAdmin.by/sql.php?server=1&db=remontbytovoytekhniki&table=detali&pos=0&token=9aab269930c4705d244a3de909ea71e1)” |
| stoimost\_remonta | int | стоимость ремонта |

Продолжение таблицы 3.11

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| alert\_client | bool | оповещение клиента |
| data\_postupleniya\_zakaza | date | дата поступления заказа |
| data\_ispolneniya\_zakaza | date | дата исполнения заказа |
| data\_polucheniya\_tovara | date | дата получения товара |

Таблица “vidremonta” хранит информацию о видах ремонта, содержит 2 атрибута. Описание, тип и название атрибутов описано в таблице 3.12.

Таблица 3.12 - Физическая структура таблицы “vidremonta”.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| id\_vidremont | int | первичный ключ |
| nazvanie | int | название ремонта |

В этом пункте было рассмотрено описание способа физической реализации логического проекта базы данных.

3.2 Структура приложения

В разрабатываемом программном средстве существует 7 классов: «Database», «FieldForm», «EditFormTable», «Authorization», «Menu», «DatabaseShow», «EditForm».

Класс «Database» - это класс, с помощью которого осуществляется подключение к базе данных, а также выполнение запросов. В нем содержатся следующие поля:

- string Server – адрес сервера;

- string Username – имя пользователя от базы данных;

- string Password – пароль от базы данных;

- string NameDatabase – название базы данных;

- MySqlConnection mysqlconnection – соединение с базой данных;

- MySqlCommand mysqlcommand – запрос;

- MySqlDataAdapter mysqladapter – объект, заполняющий таблицу данными.

Класс «FieldForm» - данный класс содержит информацию об элементе управления, который будет создаваться динамически. В нём содержаться следующие поля:

- List<string> FieldFullNames – содержит названия полей с запроса;

- string FieldCaption – название поля;

- bool HaveLink – текстовое поле;

- bool Date – дата гарантии;

- bool Check – сообщение клиенту;

- bool CheckInt – поле с типом int;

- EditFormTable TableField – ссылка на другую форму;

- List<int> IDs – содержит идентификаторы записей.

Класс «EditFormTable» - данный класс содержит информацию об элементах управления, которые будет создаваться динамически. В нём содержаться следующие поля:

- string NameForm – название формы;

- string NameTable – название таблицы;

- EditType Type – тип редактирования (добавить\редактировать);

- List<FieldForm> - содержит информацию полей.

Класс «Authorization» - данный класс является окном авторизации. Содержит поле 1 поле - Database database – информация об базе данных.

Класс «Menu» является главным меню, который предоставляет переход к таблицам.

Класс «DatabaseShow» - содержит таблицу, которая отображает полученную информацию в результате запроса, а также позволяет сортировать по критериям.

Класс «EditForm» - форма добавления и редактирования;

В каждом классе реализованы методы.

Методы класса «Database»:

- public bool OpenConnect() - открытие соединения с базой данных;

- public DataTable SelectQuery(string query) – формирование таблицы с данными;

- public void Query(string query) – выполнение запроса;

- public Database(string server, string username, string password, string namedatabase = "remontbytovoytekhniki") – инициализация компонентов формы.

Класс «FieldForm» содержит 1 метод - FieldForm() – конструктор класса.

Класс «EditFormTable» содержит 1 метод - EditFormTable() – конструктор класса.

Методы класса «Authorization»:

- private void pictureBox4\_Click(object sender, EventArgs e) – проверка на ввод правильности данных;

- public Authorization() – инициализация компонентов формы.

Методы класса «Menu»:

- private void Menu\_FormClosed(object sender, FormClosedEventArgs e) – закрытие приложения;

- private void pictureBox1\_Click(object sender, EventArgs e) – открытие формы с таблицой “Категория”;

- public void CreateFormsForTables() – создание объектов формы добавления и редактирования;

- private void pictureBox3\_Click(object sender, EventArgs e) – открытие формы с таблицей “Клиент”;

- private void pictureBox4\_Click(object sender, EventArgs e) – открытие формы с таблицей “Детали”;

- private void pictureBox5\_Click(object sender, EventArgs e) – открытие формы с таблицей “Должности”;

- private void pictureBox9\_Click(object sender, EventArgs e) – открытие формы с таблицей “Сотрудники”;

- private void pictureBox12\_Click(object sender, EventArgs e) – открытие формы с таблицей “Заказы”;

- private void pictureBox8\_Click(object sender, EventArgs e) – открытие формы с таблицей “Модели”;

private void pictureBox11\_Click(object sender, EventArgs e) – открытие формы с таблицей “Виды ремонта”;

- private void pictureBox10\_Click(object sender, EventArgs e) – открытие формы с таблицей “Товары”;

- private void pictureBox7\_Click(object sender, EventArgs e) – открытие формы с таблицей “Гарантии”;

- private void pictureBox6\_Click(object sender, EventArgs e) – открытие формы с таблицей “Фирмы”;

- private void pictureBox13\_Click(object sender, EventArgs e) – открытие справки.

Методы класса «DatabseShow»:

- private void dataGridView1\_CellMouseUp(object sender, CellMouseEventArgs e) – показ контекстного меню;

- private void addRowToolStripMenuItem\_Click(object sender, EventArgs e) – открытие формы добавление записей;

- private void editRowToolStripMenuItem\_Click(object sender, EventArgs e) – открытие формы редактирование записи;

- private void LoadData(string[] names, string query) – загрузка данных в таблицу;

- private int FindIndexRow(int ID) – нахождение индекса выделенной строки;

- private void SetElements(string nametable) – изменение свойств элементов управления на форме;

- private int SetSizeDataGridView() – изменение ширины таблицы;

- private void FillSearchCrit() – изменение заголовка столбцов таблицы;

- private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e) – выбор криетерия поиска;

- private void textBox1\_TextChanged(object sender, EventArgs e) – ввод данных для осуществления поиска.

Методы класса « EditForm»:

- private void CreateControlElements(EditFormTable eft) – создание элементов управления на форме;

- private List<string> GetValuesForComboBox(FieldForm fieldf) – получение значений для выпадающего списка;

- private void FillValues() – заполнение значений элементов управления;

- private string CreateInsertQuery() – формирование запроса на добавление записи в таблицу базы данных;

- private string CreateUpdateQuery() - формирование запроса на изменение записи в таблице базы данных;

- private int GetValueFieldComboBox(ComboBox cb, FieldForm fieldf) – получение идентификатора с другой таблицы;

- private bool EmptyValueInField() – проверка на правильность ввода данных.

Для разработки потребовались следующие компоненты:

- button – элемент управления "кнопка", реагирующий на событие Click;

- textBox – элемент управления, который может использоваться для ввода информации;

- label – возвращает или задает текстовое содержимое;

- pictureBox – используется для отображения рисунка;

- dataTimePicker – отображает время и дату;

- comboBox – предназначен для выбора значения, с раскрывающимся списком, который можно отображать и скрывать нажатием принадлежащей этому элементу управления кнопки со стрелкой;

- checkbox – представляет флажок;

- listBox – содержит список элементов для выбора.

Главным компонентом для отображения данных используется dataGridView.

4 Тестирование

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий две различные цели:

- продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;

- выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Качество программного обеспечения можно определить как совокупную характеристику исследуемого ПО с учётом следующих составляющих:

- надёжность;

- сопровождаемость;

- практичность;

- эффективность;

- мобильность;

- функциональность.

Уровни тестирования:

- модульное тестирование - тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция;

- интеграционное тестирование - тестируются интерфейсы между компонентами, подсистемами или системами;

- системное тестирование - тестируется интегрированная система на её соответствие требованиям;

- альфа-тестирование - имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования [5];

- бета-тестирование - в некоторых случаях выполняется распространение предварительной версии для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Таблица 4.1 – Ход тестирования

|  |  |  |
| --- | --- | --- |
| Ход тестирования | Ожидание | Результат |
| Авторизация | Переход в главное меню | Переход в главное меню |
| Просмотр справочной информации | Отображение справочной информации | Отображение справочной информации |
| Переход к выбранной таблице | Вывод выбранной таблицы | Вывод выбранной таблицы |
| Поиск с введённым значением другого типа | Вывод предупреждения | Вывод предупреждения |
| Удаление записи в таблице | Изменение таблицы в БД, изменение таблицы на форме | Таблица в БД и на форме изменена |
| Вывод контекстного меню по нажатию правой кнопки мыши | Вывод контекстного меню | Вывод контекстного меню |
| Переход на форму редактирования | Переход на форму редактирования | Переход на форму редактирования |
| Динамическое создание элементов управления на форме редактирования | Созданные элементы управления | Созданные элементы управления |
| Добавление новой записи | Изменение таблицы в БД, изменение таблицы на форме | Таблица в БД и на форме изменена |
| Добавление или изменение записи с пустыми полями | Вывод предупреждения | Вывод предупреждения |
| Изменение записи | Изменение таблицы в БД, изменение таблицы на форме | Таблица в БД и на форме изменена |

В ходе тестирования программного автоматизации рабочего места менеджера ателье по ремонту бытовой техники ошибок не найдено.

# 5 Описание применения

Системные требования — это описание примерных характеристик, которым должен соответствовать компьютер для того, чтобы на нём могло использоваться какое-либо определённое программное обеспечение. Эти характеристики могут описывать требования как к аппаратному обеспечению, так и к программному окружению. Обычно такие требования составляются производителем или автором программного обеспечения [4].

Для некоторого ПО различают минимальные и рекомендуемые системные требования:

- минимальные системные требования — это набор условий, необходимых для возможности запуска и работы программного продукта. Однако, наличие минимальных системных требований не отменяет возможность запуска программного обеспечения на компьютерах, которые по характеристикам слабее минимальных;

- рекомендуемые системные требования — набор характеристик, подразумевающих оптимальную работу большей части возможностей продукта. Однако, даже если компьютер и подходит под рекомендуемые системные требования, это не значит высокой производительности программного обеспечения, например, в некоторых играх невозможно играть на максимальных настройках графики;

- максимальные системные требования — набор характеристик которые позволяют запустить программного обеспечения с максимальной производительностью и функциями. Однако встречаются данные требования довольно редко и почти всегда в видеоиграх.

Минимальные системные требования программного средства “Автоматизированного рабочего места менеджера ателье по ремонту бытовой техники”:

- видеокарта на 64 МВ;

- 384 МВ ОЗУ;

- процессор 2,2 GHz;

- операционная система 7,8,10;

- 10 МВ свободного дискового пространства.

Для работы программы необходимо установить программную платформу .NET Framework 4.5.2 и MySQL Community Edition.

Чтобы установить программное средство “АРМ ремонт бытовой техники” необходим диск с установщиком «Setup.exe». После запуска установщика откроется окно “Мастера установки”, рисунок 5.1.

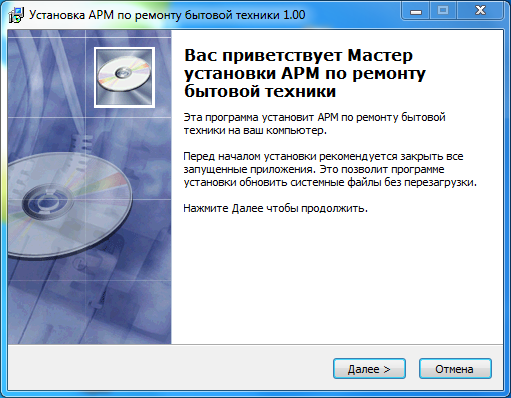


Рисунок 5.1 – Установка программного средства

Для продолжения установки необходимо нажать «Далее». После чего откроется форма, представленная на рисунке рисунок 5.2, где требуется выбрать место установки.

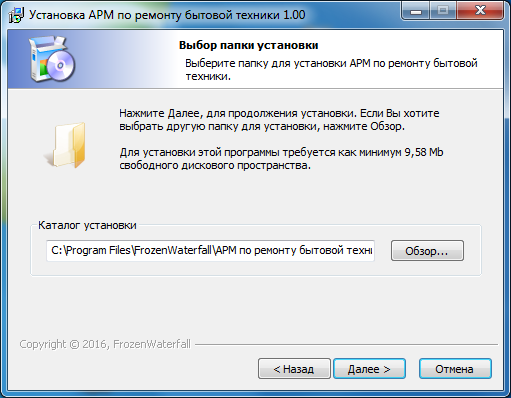
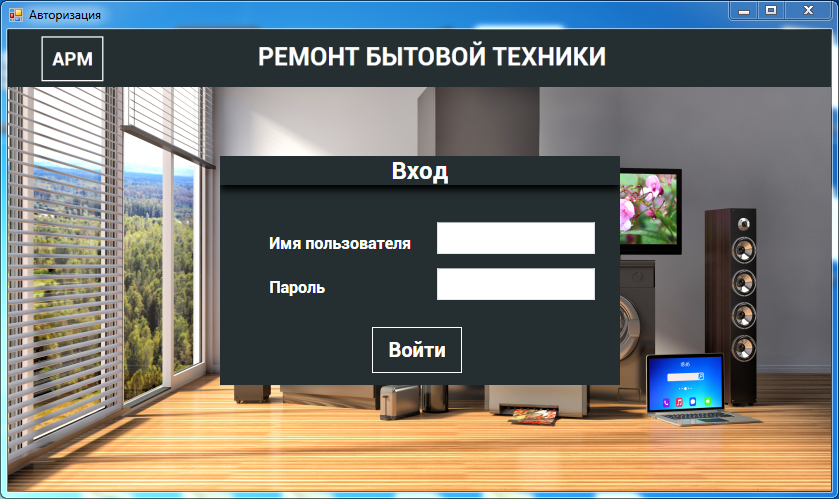


Рисунок 5.2 – Выбор места установки

Далее необходимо следовать пунктам установщика. После установки ярлык программы появится на рабочем столе.

Чтобы программа имела доступ к базе данных её необходимо сперва создать. На диске будет расположен файл “Database.sql”. После открытия этого файла будет открыто приложение MySQL, и необходимо ввести имя пользователя и пароль. При успешной авторизации отобразятся все запросы, находящиеся в файле “Database.sql”, и после их выполнения будет создана база данных с таблицами.

После запуска программного средства откроется окно авторизации, рисунок 5.3. В полях необходимо ввести имя пользователя и пароль, как и при авторизации в программе MySQL. После успешной авторизации откроется главное меню, рисунок 5.4.



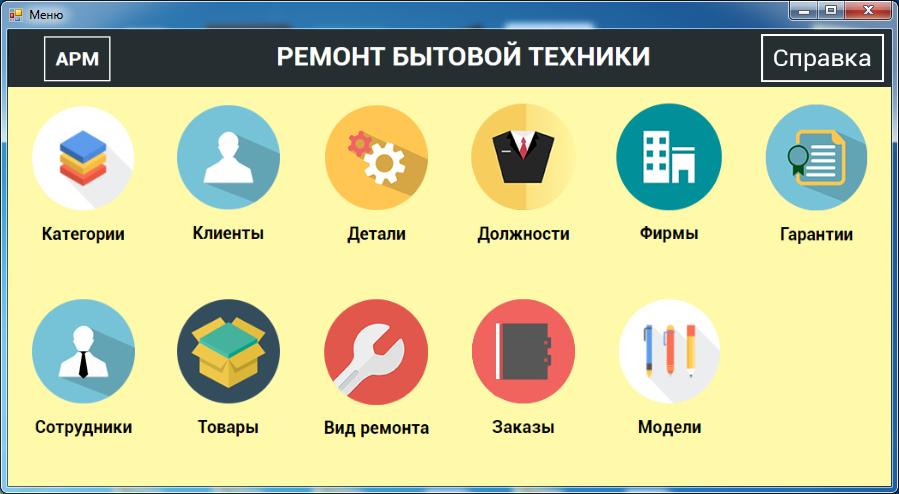
Рисунок 5.3 – Окно авторизации

Рисунок 5.4 – Окно главного меню

В главном меню расположены пункты в виде картинки с названием. При нажатии на соответствующий пункт будет открыто окно с таблицей и заполненными данными. На примере будет выбран пункт “Клиенты”, рисунок 5.5.

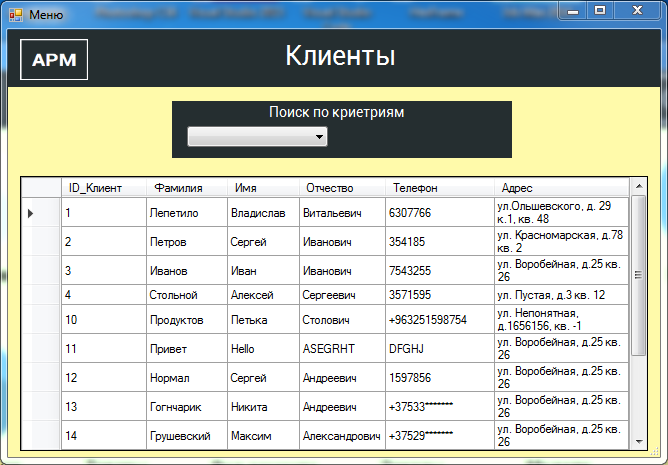


Рисунок 5.5 - Окно “Клиенты”

Чтобы изменять, добавлять или удалять данные необходимо навести на таблицу и нажать на правую кнопку мыши, после чего будет показано контекстное меню.

При нажатии кнопки “Добавить” в контекстном меню откроется окно добавление, рисунок 5.6.

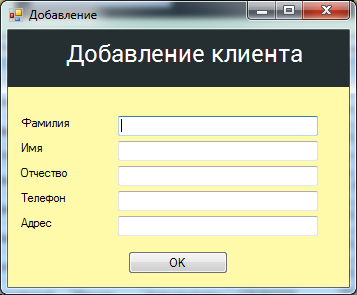


Рисунок 5.6 - Окно добавления записи

После введённых данных в окне добавления необходимо нажать кнопку “ОК” и после чего закрыть текущее окно, добавленная запись будет отображена в таблице.

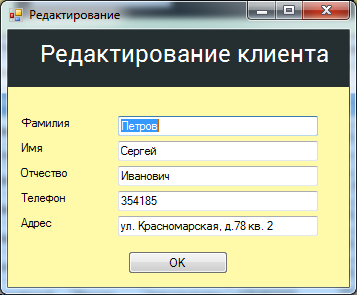
Чтобы редактировать данные необходимо нажать на нужную запись в таблице левой кнопкой мыши, в вызванном контекстном меню выбрать пункт “Редактировать запись” и после чего откроется окно редактирования, рисунок 5.7. В окне появятся все данные из выбранной записи и их можно редактировать. После завершения редактирования необходимо нажать на “ОК”;

Рисунок 5.7 - Окно редактирования записи

Для удаления записи из таблицы необходимо в вызванном контекстном меню выбрать пункт “Удалить запись”.

# Заключение

Целью курсового проекта является разработка автоматизированного рабочего места менеджера ателье по ремонту бытовой техники.

Разработанное программное средство “Автоматизированное рабочее место менеджера ателье по ремонту бытовой техники” обладает следующими функциями:

- хранение информации о фирмах, категориях, моделях, пол, должностях, товаров, гарантиях, сотрудниках, клиентах, видов ремонтов, деталях и заказах в базе данных;

- добавление новых данных - пользователь может добавить любую необходимую информацию в базу данных;

- редактирование данных - пользователь может быстро обновлять и вносить изменения в уже существующие записи базы данных;

- удаление данных - пользователь может производить удаление определённых данных из базы данных;

- наличие интуитивно-понятного интерфейса, для облегчения взаимодействия пользователя с программным средством;

- проверка на правильность ввода данных;

- осуществление поиска в таблицах.

Проведённое тестирование показало, что программное средство работает корректно и готово к использованию.

Во время создания программного средства были изучены и закреплены навыки с языком SQL для создания, модификации и управления данными в СУБД MySQL. Также были закреплены следующие навыки с языком программирования C#:

- создание и использование объектов, которая предоставляет MySQL библиотека;

- динамическое создание элементов управления;

- создание методов для вывода и считывания данных с таблицы.

Цели разработки достигнуты, программное средство соответствует требованиям, которые были описаны в формулировке задачи.

Разработаны диаграммы классов, которая показывает структуру и иерархии классов в программе и вариантов использования, описывающая то, как можно использовать программу.

При разработке программного средства, поставленная задача выполнена полностью.

# Список использованных источников

1 Дейтел, Х.С. С#: перевод с английского / Х.С. Дейтел, П.С. Дейтел. – СПб : БХВ – Петербург, 2011. – 1056 с.

2 Канкурогова, Н.И. C# для чайников: учеб. пособие / Н.И. Канкурогова. – Москва: ООО "Издательский центр "Академия", 2012. – 314 с.

3 Кормен, Т. Алгоритмы. Построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест . – М.: Форум, ИНФРА, 2013. – 469 c.

4 Линн, Б. Изучаем SQL / Б. Линн, М. Моррисон. – М. Эксмо, 2011. – 325 с.

5 Стиллмен.Э., Грин.Дж. Изучаем С# / Э. Стиллмен, Дж.Грин. – 3-е изд. – СПб.: Питер, 2014. – 812 с.

# Приложение А

(обязательное)

Текст программы

public class Database

{

public string Server { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string NameDatabase { get; set; }

public MySqlConnection mysqlconnection;

public MySqlCommand mysqlcommand;

public MySqlDataAdapter mysqladapter;

public string NetConnectionString

{

get

{

return string.Format("Server={0};Uid={1};Pwd={2};Database={3};charset=utf8", Server, Username, Password, NameDatabase);

}

}

public bool OpenConnect()

{

mysqlconnection = new MySqlConnection(NetConnectionString);

try

{

mysqlconnection.Open();

return true;

}

catch

{

MessageBox.Show("Неправильно введены логин или пароль!", "Ошибка!");

return false;

}

}

public DataTable SelectQuery(string query)

{

DataTable dataTable = new DataTable();

mysqlcommand = new MySqlCommand(query, mysqlconnection);

mysqladapter = new MySqlDataAdapter(mysqlcommand);

mysqladapter.Fill(dataTable);

return dataTable;

}

public void Query(string query)

{

try

{

OpenConnect();

mysqlcommand = new MySqlCommand(query, mysqlconnection);

mysqlcommand.ExecuteNonQuery();

}

catch(MySqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK);

}

}

public Database(string server, string username, string password, string namedatabase = "remontbytovoytekhniki")

{

this.Server = server;

this.Username = username;

this.Password = password;

this.NameDatabase = namedatabase;

}

}

public enum EditType

{

Add,

Edit

}

public class FieldForm

{

public List<string> FieldFullNames { get; set; }

public string FieldCaption { get; set; }

public bool HaveLink { get; set; }

public bool Date { get; set; }

public bool Check { get; set; }

public bool CheckInt { get; set; }

public EditFormTable TableField { get; set; }

public List<int> IDs { get; set; }

public FieldForm()

{

FieldFullNames = new List<string>();

}

}

public class EditFormTable

{

public string NameForm { get; set; } //Название формы

public string NameTable { get; set; } //Название таблицы

public EditType Type { get; set; }

public List<FieldForm> Fields { get; set; }

public EditFormTable()

{

Fields = new List<FieldForm>();

}

}

public partial class Authorization : Form

{

public Authorization()

{

InitializeComponent();

}

public Database database;

private void pictureBox4\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || textBox2.Text == "")

{

MessageBox.Show("Заполните поля!");

return;

}

database = new Database("localhost", textBox1.Text, textBox2.Text);

if (database.OpenConnect())

(new Menu(this)).Show();

}

}

public partial class Menu : Form

{

public Menu(Authorization auth)

{

InitializeComponent();

this.auth = auth;

auth.Hide();

CreateFormsForTables();

}

private Authorization auth;

///ДЛЯ ДОБАВЛЕНИЯ И РЕДАКТИРОВАНИЯ///

EditFormTable eftZakazy;

EditFormTable eftClients;

EditFormTable eftVidremonta;

EditFormTable eftDetali;

EditFormTable eftTovary;

EditFormTable eftGarantii;

EditFormTable eftSotrudniki;

EditFormTable eftSex;

EditFormTable eftDolgnosti;

EditFormTable eftModels;

EditFormTable eftCategories;

EditFormTable eftFirms;

////////////////////////////////////

private void Menu\_FormClosed(object sender, FormClosedEventArgs e)

{

auth.Close();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Категория", "Название"};

new DatabaseShow("Категории", auth.database, eftCategories, names, "SELECT \* FROM categories").ShowDialog();

}

public void CreateFormsForTables()

{

eftZakazy = new EditFormTable() { NameForm = "заказа", NameTable = "zakazy" };

eftClients = new EditFormTable() { NameForm = "клиента", NameTable = "clients" };

eftVidremonta = new EditFormTable() { NameForm = "вида ремонта", NameTable = "vidremonta" };

eftDetali = new EditFormTable() { NameForm = "детали", NameTable = "detali" };

eftTovary = new EditFormTable() { NameForm = "товара", NameTable = "tovary" };

eftGarantii = new EditFormTable() { NameForm = "гарантии", NameTable = "garantii" };

eftSotrudniki = new EditFormTable() { NameForm = "сотрудника", NameTable = "sotrudniki" };

eftSex = new EditFormTable() { NameForm = "пола", NameTable = "sex" };

eftDolgnosti = new EditFormTable() { NameForm = "должности", NameTable = "dolgnosti" };

eftModels = new EditFormTable() { NameForm = "модели", NameTable = "models" };

eftCategories = new EditFormTable() { NameForm = "категории", NameTable = "categories" };

eftFirms = new EditFormTable() { NameForm = "фирмы", NameTable = "firms" };

////////////////ЗАКАЗЫ/////////////////

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.id\_client", "clients.familiya", "clients.imya" }, FieldCaption = "Клиент", HaveLink = true, IDs = new List<int>(), TableField = eftClients });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "tovary.id\_tovar", "tovary.seriyniy\_nomer" }, FieldCaption = "Товары", HaveLink = true, IDs = new List<int>(), TableField = eftTovary });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "garantii.id\_garantiya", "garantii.garantiyniy\_srok" }, FieldCaption = "Гарантия", HaveLink = true, IDs = new List<int>(), TableField = eftGarantii });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "vidremonta.id\_vidremont", "vidremonta.nazvanie" }, FieldCaption = "Вид ремонта", HaveLink = true, IDs = new List<int>(), TableField = eftVidremonta });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.id\_sotrudnik", "sotrudniki.familiya", "sotrudniki.imya" }, FieldCaption = "Сотрудник", HaveLink = true, IDs = new List<int>(), TableField = eftSotrudniki });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "detali.id\_detal", "detali.nazvanie", "detali.kolichesstvo" }, FieldCaption = "Деталь", HaveLink = true, IDs = new List<int>(), TableField = eftDetali });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "zakazy.stoimost\_remonta" }, FieldCaption = "Стоимость ремонта", CheckInt = true });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "zakazy.alert\_client" }, FieldCaption = "Сообщить клиенту", Check = true });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "zakazy.data\_postupleniya\_zakaza" }, FieldCaption = "Дата поступления заказа", Date = true });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "zakazy.data\_ispolneniya\_zakaza" }, FieldCaption = "Дата исполнения заказа", Date = true });

eftZakazy.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "zakazy.data\_polucheniya\_tovara" }, FieldCaption = "Дата получения товара", Date = true });

///////////////КЛИЕНТЫ////////////////

eftClients.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.familiya" }, FieldCaption = "Фамилия" });

eftClients.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.imya" }, FieldCaption = "Имя" });

eftClients.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.otchestvo" }, FieldCaption = "Отчество" });

eftClients.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.telephon" }, FieldCaption = "Телефон" });

eftClients.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "clients.addres" }, FieldCaption = "Адрес" });

//////////////ВИД РЕМОНТА/////////////

eftVidremonta.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "vidremonta.nazvanie" }, FieldCaption = "Вид ремонта" });

////////////////ДЕТАЛИ////////////////

eftDetali.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "detali.nazvanie" }, FieldCaption = "Название" });

eftDetali.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "detali.kolichesstvo" }, FieldCaption = "Количество", CheckInt = true });

////////////////ТОВАРЫ////////////////

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "firms.id\_firm", "firms.firma" }, FieldCaption = "Фирма", HaveLink = true, IDs = new List<int>(), TableField = eftFirms });

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "categories.id\_category", "categories.category" }, FieldCaption = "Категория", HaveLink = true, IDs = new List<int>(), TableField = eftCategories });

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "models.id\_model", "models.nazvanie" }, FieldCaption = "Модель", HaveLink = true, IDs = new List<int>(), TableField = eftModels });

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "tovary.seriyniy\_nomer" }, FieldCaption = "Серийный номер" });

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "tovary.tehnichiskie\_haracteristici" }, FieldCaption = "Технические характеристики" });

eftTovary.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "tovary.garantiyniy\_srok" }, FieldCaption = "Гарантийный срок", Date = true });

////////////////ГАРАНТИИ//////////////

eftGarantii.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "garantii.garantiyniy\_srok" }, FieldCaption = "Гарантийный срок", Date = true });

eftGarantii.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "garantii.nazvanie\_centra" }, FieldCaption = "Название центра" });

eftGarantii.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "garantii.addres" }, FieldCaption = "Адрес" });

////////////////Сотрудники//////////////

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sex.id\_sex", "sex.sex" }, FieldCaption = "Пол", HaveLink = true, IDs = new List<int>(), TableField = eftSex });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "dolgnosti.id\_dolgnost", "dolgnosti.nazvanie" }, FieldCaption = "Должность", HaveLink = true, IDs = new List<int>(), TableField = eftDolgnosti });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.familiya" }, FieldCaption = "Фамилия" });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.imya" }, FieldCaption = "Имя" });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.otchestvo" }, FieldCaption = "Отчество" });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.telephon" }, FieldCaption = "Телефон" });

eftSotrudniki.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sotrudniki.addres" }, FieldCaption = "Адрес" });

///////////////////ПОЛ//////////////////

eftSex.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "sex.sex" }, FieldCaption = "Пол" });

////////////////Должности///////////////

eftDolgnosti.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "dolgnosti.nazvanie" }, FieldCaption = "Должность" });

////////////////Модели///////////////

eftModels.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "models.nazvanie" }, FieldCaption = "Модель" });

////////////////Категории///////////////

eftCategories.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "categories.category" }, FieldCaption = "Категория" });

////////////////Фирмы///////////////

eftFirms.Fields.Add(new FieldForm() { FieldFullNames = new List<string>() { "firms.firma" }, FieldCaption = "Фирма" });

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Клиент", "Фамилия", "Имя", "Отчество", "Телефон", "Адрес" };

new DatabaseShow("Клиенты", auth.database, eftClients, names, "SELECT \* FROM clients").ShowDialog();

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Детали", "Название", "Количество"};

new DatabaseShow("Детали", auth.database, eftDetali, names, "SELECT \* FROM detali").ShowDialog();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Должности", "Название"};

new DatabaseShow("Должности", auth.database, eftDolgnosti, names, "SELECT \* FROM dolgnosti").ShowDialog();

}

private void pictureBox9\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Сотрудник", "Пол", "Должность", "Фамилия", "Имя", "Отчество", "Телефон", "Адрес" };

new DatabaseShow("Сотрудники", auth.database, eftSotrudniki, names, "SELECT sotrudniki.id\_sotrudnik, sex.sex, dolgnosti.nazvanie, sotrudniki.familiya, sotrudniki.imya, sotrudniki.otchestvo, sotrudniki.telephon, sotrudniki.addres FROM dolgnosti INNER JOIN(sex INNER JOIN sotrudniki ON sex.id\_sex = sotrudniki.id\_sex) ON dolgnosti.id\_dolgnost = sotrudniki.id\_dolgnost; "/\*"SELECT \* FROM sotrudniki"\*/).ShowDialog();

}

private void pictureBox12\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Заказа", "(Клиент) Фамилия", "(Клиент) Имя", "(Клиент) Отчество", "(Клиент) Телефон", "(Клиент) Адрес", "(Товар) Фирма", "(Товар) Категория", "(Товар) Модель", "(Товар) Серийный номер", "(Товар) Технические характеристики", "(Товар) Гарантийный срок", "(Гарантия) Гарантийный срок", "(Гарантия) Название центра", "(Гарантия) Адрес", "Вид ремонта", "(Сотрудник) Пол", "(Сотрудник) Должность", "(Сотрудник) Фамилия", "(Сотрудник) Имя", "(Сотрудник) Отчество", "(Сотрудник) Телефон", "(Сотрудник) Адрес", "(Деталь) Название", "(Деталь) Количество", "Стоимость ремонта", "Сообщение клиенту", "Дата поступления заказа", "Дата исполнения заказа", "Дата получения товара" };

new DatabaseShow("Заказы", auth.database, eftZakazy, names, "SELECT zakazy.id\_zakaz, clients.familiya, clients.imya, clients.otchestvo, clients.telephon, clients.addres, firms.firma, categories.category, models.nazvanie, tovary.seriyniy\_nomer, tovary.tehnichiskie\_haracteristici, tovary.garantiyniy\_srok, garantii.garantiyniy\_srok, garantii.nazvanie\_centra, garantii.addres, vidremonta.nazvanie, sex.sex, dolgnosti.nazvanie, sotrudniki.familiya, sotrudniki.imya, sotrudniki.otchestvo, sotrudniki.telephon, sotrudniki.addres, detali.nazvanie, detali.kolichesstvo, zakazy.stoimost\_remonta, zakazy.alert\_client, zakazy.data\_postupleniya\_zakaza, zakazy.data\_ispolneniya\_zakaza, zakazy.data\_polucheniya\_tovara FROM vidremonta INNER JOIN((models INNER JOIN(firms INNER JOIN(categories INNER JOIN tovary ON categories.id\_category = tovary.id\_category) ON firms.id\_firm = tovary.id\_firm) ON models.id\_model = tovary.id\_model) INNER JOIN((sex INNER JOIN(dolgnosti INNER JOIN sotrudniki ON dolgnosti.id\_dolgnost = sotrudniki.id\_dolgnost) ON sex.id\_sex = sotrudniki.id\_sex) INNER JOIN(garantii INNER JOIN(detali INNER JOIN(clients INNER JOIN zakazy ON clients.id\_client = zakazy.id\_client) ON detali.id\_detal = zakazy.id\_detal) ON garantii.id\_garantiya = zakazy.id\_garantiya) ON sotrudniki.id\_sotrudnik = zakazy.id\_sotrudnik) ON tovary.id\_tovar = zakazy.id\_tovar) ON vidremonta.id\_vidremont = zakazy.id\_vidremont; "/\*"SELECT zakazy.id\_zakaz, clients.familiya, clients.imya, clients.otchestvo, clients.telephon, clients.addres, firms.firma, categories.category, models.nazvanie, tovary.seriyniy\_nomer, tovary.tehnichiskie\_haracteristici, tovary.garantiyniy\_srok, garantii.garantiyniy\_srok, garantii.nazvanie\_centra, garantii.addres, vidremonta.nazvanie, sotrudniki.id\_sex, sotrudniki.id\_dolgnost, sotrudniki.familiya, sotrudniki.imya, sotrudniki.otchestvo, sotrudniki.telephon, sotrudniki.addres, detali.nazvanie, detali.kolichesstvo, zakazy.stoimost\_remonta, zakazy.alert\_client, zakazy.data\_postupleniya\_zakaza, zakazy.data\_ispolneniya\_zakaza, zakazy.data\_polucheniya\_tovara FROM vidremonta INNER JOIN((models INNER JOIN(firms INNER JOIN(categories INNER JOIN tovary ON categories.id\_category = tovary.id\_category) ON firms.id\_firm = tovary.id\_firm) ON models.id\_model = tovary.id\_model) INNER JOIN((sex INNER JOIN(dolgnosti INNER JOIN sotrudniki ON dolgnosti.id\_dolgnost = sotrudniki.id\_dolgnost) ON sex.id\_sex = sotrudniki.id\_sex) INNER JOIN(garantii INNER JOIN(detali INNER JOIN(clients INNER JOIN zakazy ON clients.id\_client = zakazy.id\_client) ON detali.id\_detal = zakazy.id\_detal) ON garantii.id\_garantiya = zakazy.id\_garantiya) ON sotrudniki.id\_sotrudnik = zakazy.id\_sotrudnik) ON tovary.id\_tovar = zakazy.id\_tovar) ON vidremonta.id\_vidremont = zakazy.id\_vidremont; "\*//\*"SELECT \* FROM zakazy"\*/).ShowDialog();

}

private void pictureBox8\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Модель", "Название" };

new DatabaseShow("Модели", auth.database, eftModels, names, "SELECT \* FROM models").ShowDialog();

}

private void pictureBox11\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Ремонт", "Название" };

new DatabaseShow("Виды ремонта", auth.database, eftVidremonta, names, "SELECT \* FROM vidremonta").ShowDialog();

}

private void pictureBox10\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Товар", "Фирма", "Категория", "Модель", "Серийный номер", "Технические характеристики", "Гарантийный срок" };

new DatabaseShow("Товары", auth.database, eftTovary, names, "SELECT tovary.id\_tovar, firms.firma, categories.category, models.nazvanie, tovary.seriyniy\_nomer, tovary.tehnichiskie\_haracteristici, tovary.garantiyniy\_srok FROM models INNER JOIN(categories INNER JOIN(firms INNER JOIN tovary ON firms.id\_firm = tovary.id\_firm) ON categories.id\_category = tovary.id\_category) ON models.id\_model = tovary.id\_model;"/\*"SELECT \* FROM tovary"\*/).ShowDialog();

}

private void pictureBox7\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Гарантия", "Гарантийный срок", "Название центра", "Адрес" };

new DatabaseShow("Гарантии", auth.database, eftGarantii, names, "SELECT \* FROM garantii").ShowDialog();

}

private void pictureBox6\_Click(object sender, EventArgs e)

{

string[] names = { "ID\_Фирма", "Название"};

new DatabaseShow("Фирмы", auth.database, eftFirms, names, "SELECT \* FROM firms").ShowDialog();

}

private void pictureBox13\_Click(object sender, EventArgs e)

{

}

}

public partial class DatabaseShow : Form

{

public DatabaseShow(string nameDataBase, Database db, EditFormTable editTable, string[] names, string query)

{

InitializeComponent();

\_editFormTable = editTable;

\_database = db;

\_query = query;

\_captionsColumns = names;

LoadData(\_captionsColumns, \_query);

SetElements(nameDataBase);

comboBox1.DropDownWidth = 210;

}

EditFormTable \_editFormTable;

Database \_database;

DataTable \_dataTable;

Type \_currentType;

string \_query;

string[] \_captionsColumns;

private void dataGridView1\_CellClick\_1(object sender, DataGridViewCellEventArgs e) //Выделение всей строки

{

dataGridView1.CurrentRow.Selected = true;

}

private void dataGridView1\_CellMouseUp(object sender, DataGridViewCellMouseEventArgs e) //Показ ContexMenuStrip

{

if (e.Button == MouseButtons.Right)

dataGridView1.ContextMenuStrip.Show();

}

private void addRowToolStripMenuItem\_Click(object sender, EventArgs e)

{

EditForm ef = new EditForm(\_editFormTable, \_database);

ef.FormClosing += delegate (object senderr, FormClosingEventArgs ev)

{

LoadData(\_captionsColumns, \_query);

dataGridView1.CurrentCell.Selected = false;

if (ef.lastInsertIndex != -1)

{

int indexrow = FindIndexRow(ef.lastInsertIndex);

if (indexrow == -1)

return;

dataGridView1.Rows[indexrow].Selected = true;

}

};

ef.ShowDialog();

}

private void editRowToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dataGridView1.CurrentRow.Index == -1)

{

MessageBox.Show("Не выбрана запись!");

return;

}

EditForm ef = new EditForm(\_editFormTable, \_database, (int)dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[0].Value);

ef.FormClosing += delegate (object senderr, FormClosingEventArgs ev)

{

LoadData(\_captionsColumns, \_query);

dataGridView1.CurrentCell.Selected = false;

if (ef.ID != -1)

{

int indexrow = FindIndexRow(ef.ID);

if (indexrow == -1)

return;

dataGridView1.Rows[indexrow].Selected = true;

}

if (ef.EditFormTable.Type == EditType.Add)

{

if (ef.lastInsertIndex != -1)

{

int indexrow = FindIndexRow(ef.lastInsertIndex);

if (indexrow == -1)

return;

dataGridView1.Rows[indexrow].Selected = true;

}

}

else

{

if (ef.ID != -1)

{

int indexrow = FindIndexRow(ef.ID);

if (indexrow == -1)

return;

dataGridView1.Rows[indexrow].Selected = true;

}

}

};

ef.ShowDialog();

}

private void LoadData(string[] names, string query)

{

\_dataTable = \_database.SelectQuery(query);

dataGridView1.DataSource = \_dataTable;

try

{

for (int i = 0; i < dataGridView1.ColumnCount; i++)

dataGridView1.Columns[i].HeaderText = names[i];

}

catch { }

dataGridView1.AutoResizeColumns();

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

}

private int FindIndexRow(int ID)

{

foreach (DataGridViewRow row in dataGridView1.Rows)

if ((int)row.Cells[0].Value == ID)

return row.Index;

return -1;

}

private void dataGridView1\_CellEnter(object sender, DataGridViewCellEventArgs e)

{

dataGridView1.CurrentCell.Selected = false;

}

private void dataGridView1\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

//dataGridView1.CurrentRow.Selected = true;

}

private void SetElements(string nametable)

{

label1.Text = nametable;

pictureBoxHeader.Controls.Add(label1);

this.Controls.Remove(label1);

this.WindowState = FormWindowState.Maximized;

pictureBoxHeader.Width = this.Width;

panel1.Location = new Point(this.Width / 2 - panel1.Width/2, panel1.Location.Y);

dateTimePickerSearch.CustomFormat = "yyyy-MMM-dd";

FillSearchCrit();

}

private int SetSizeDataGridView()

{

int size = 0;

for(int i = 0; i < dataGridView1.ColumnCount; i++)

{

size += dataGridView1.Columns[i].Width;

}

return size + 45;

}

private void DatabaseShow\_Resize(object sender, EventArgs e)

{

if (this.Width < 379)

{

this.Width = 379;

}

dataGridView1.Width = this.Width - 40;

dataGridView1.Height = this.Height - 190;

pictureBoxHeader.Width = this.Width;

}

private void pictureBoxHeader\_Resize(object sender, EventArgs e)

{

label1.Location = new Point(pictureBoxHeader.Width /2 - label1.Width/2, label1.Location.Y);

panel1.Location = new Point(this.Width / 2 - panel1.Width / 2, panel1.Location.Y);

}

private void FillSearchCrit()

{

foreach (string val in \_captionsColumns)

comboBox1.Items.Add(val);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == -1)

return;

DataView dv = new DataView(\_dataTable);

dataGridView1.DataSource = dv;

if (\_dataTable.Columns[comboBox1.SelectedIndex].DataType == typeof(int) || \_dataTable.Columns[comboBox1.SelectedIndex].DataType == typeof(string))

{

dateTimePickerSearch.Visible = false;

checkBoxSearch.Visible = false;

textBoxSearch.Visible = true;

}

else if (\_dataTable.Columns[comboBox1.SelectedIndex].DataType == typeof(DateTime))

{

dateTimePickerSearch.Visible = true;

checkBoxSearch.Visible = false;

textBoxSearch.Visible = false;

}

else

{

dateTimePickerSearch.Visible = false;

checkBoxSearch.Visible = true;

textBoxSearch.Visible = false;

}

\_currentType = \_dataTable.Columns[comboBox1.SelectedIndex].DataType;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

DataView dv = new DataView(\_dataTable);

if (\_currentType == typeof(int))

{

if (textBoxSearch.Text.Length == 0)

{

dataGridView1.DataSource = dv;

return;

}

if (!Char.IsDigit(textBoxSearch.Text[textBoxSearch.Text.Length - 1]))

{

MessageBox.Show("Должны вводиться числа!", "Ошибка!", MessageBoxButtons.OK);

textBoxSearch.Text = "";

return;

}

dv.RowFilter = string.Format("{0} = '{1}'", \_dataTable.Columns[comboBox1.SelectedIndex].ColumnName, textBoxSearch.Text);

}

else

{

dv.RowFilter = string.Format("{0} LIKE '%{1}%'", \_dataTable.Columns[comboBox1.SelectedIndex].ColumnName, textBoxSearch.Text);

}

dataGridView1.DataSource = dv;

}

private void dateTimePickerSearch\_ValueChanged(object sender, EventArgs e)

{

DataView dv = new DataView(\_dataTable);

dv.RowFilter = string.Format("{0} = '{1}'", \_dataTable.Columns[comboBox1.SelectedIndex].ColumnName, dateTimePickerSearch.Value.Year+"."+dateTimePickerSearch.Value.Month+"."+dateTimePickerSearch.Value.Day);

dataGridView1.DataSource = dv;

}

private void checkBoxSearch\_CheckedChanged(object sender, EventArgs e)

{

DataView dv = new DataView(\_dataTable);

dv.RowFilter = string.Format("{0} = '{1}'", \_dataTable.Columns[comboBox1.SelectedIndex].ColumnName, checkBoxSearch.Checked);

dataGridView1.DataSource = dv;

}

private void deleteRowToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dataGridView1.CurrentRow == null)

{

MessageBox.Show("Необходимо выбрать строку!", "Ошибка!", MessageBoxButtons.OK);

return;

}

\_database.Query(string.Format("DELETE FROM {0} WHERE {1} = '{2}'", \_editFormTable.NameTable, \_editFormTable.NameTable+"."+\_dataTable.Columns[0].ColumnName, (int)dataGridView1.CurrentRow.Cells[0].Value));

LoadData(\_captionsColumns, \_query);

dataGridView1.CurrentCell.Selected = false;

}

}

public partial class EditForm : Form

{

public EditForm(EditFormTable eft, Database db, int ID)

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen;

EditFormTable = eft;

\_database = db;

this.ID = ID;

this.lastInsertIndex = -1;

CreateControlElements(eft);

FillValues();

label1.Text = "Редактирование " + eft.NameForm;

EditFormTable.Type = EditType.Edit;

this.Text = "Редактирование";

pictureBoxHeader.Controls.Add(label1);

this.Controls.Remove(label1);

pictureBoxHeader.Width = this.Width;

}

public EditForm(EditFormTable eft, Database db)

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen;

EditFormTable = eft;

\_database = db;

this.ID = -1;

this.lastInsertIndex = -1;

CreateControlElements(eft);

label1.Text = "Добавление " + eft.NameForm;

EditFormTable.Type = EditType.Add;

this.Text = "Добавление";

pictureBoxHeader.Controls.Add(label1);

this.Controls.Remove(label1);

pictureBoxHeader.Width = this.Width;

}

public int ID;

public int lastInsertIndex;

private Database \_database;

public EditFormTable EditFormTable;

private List<Control> CreatedControls = new List<Control>();

private void CreateControlElements(EditFormTable eft)

{

int X = 10;

int Y = 30;

int maxwidth = 0;

bool havecombobox = false;

for (int i = 0; i < eft.Fields.Count; i++)

{

Label label = new Label();

label.Text = eft.Fields[i].FieldCaption;

//label.Width = eft.Fields[i].FieldCaption.Length \* 10;

maxwidth = (maxwidth < label.Width? label.Width : maxwidth);

label.Location = new Point(X, Y);

panel1.Controls.Add(label);

Y += 25;

}

Y = 30;

for (int i = 0; i < eft.Fields.Count; i++)

{

if (eft.Fields[i].HaveLink)

{

havecombobox = true;

ComboBox comboBox = new ComboBox();

comboBox.Tag = eft.Fields[i];

comboBox.Name = "comboBox" + i;

comboBox.DropDownStyle = ComboBoxStyle.DropDownList;

comboBox.Width = 200;

comboBox.Location = new Point(maxwidth + X, Y);

UpdateComboBox(comboBox, eft.Fields[i]);

panel1.Controls.Add(comboBox);

CreatedControls.Add(comboBox);

Button buttonf = new Button();

buttonf.Cursor = Cursors.Hand;

//button

buttonf.Text = "Добавить запись";

buttonf.Width = 100;

buttonf.Location = new Point(comboBox.Width + comboBox.Location.X + 5, Y);

buttonf.Click += delegate (object sender, EventArgs arg)

{

EditForm ef = new EditForm((comboBox.Tag as FieldForm).TableField, \_database);

ef.FormClosing += Ef\_FormClosing;

ef.Tag = comboBox;

ef.ShowDialog();

};

panel1.Controls.Add(buttonf);

Button buttong = new Button();

buttong.Cursor = Cursors.Hand;

//button

buttong.Text = "Изменить запись";

buttong.Width = 100;

buttong.Location = new Point(buttonf.Width + buttonf.Location.X + 5, Y);

buttong.Click += delegate (object sender, EventArgs arg)

{

if (comboBox.SelectedIndex == -1)

{

MessageBox.Show("Необходимо выбрать запись!");

return;

}

EditForm ef = new EditForm((comboBox.Tag as FieldForm).TableField, \_database, (comboBox.Tag as FieldForm).IDs[comboBox.SelectedIndex]);

ef.FormClosing += Ef\_FormClosing;

ef.Tag = comboBox;

ef.ShowDialog();

};

panel1.Controls.Add(buttong);

}

else if (eft.Fields[i].Date)

{

DateTimePicker dateTimePicker = new DateTimePicker();

dateTimePicker.Tag = eft.Fields[i];

dateTimePicker.CustomFormat = "yyyy-MMM-dd";

dateTimePicker.Format = DateTimePickerFormat.Custom;

dateTimePicker.Name = "dateTimePicker" + i;

dateTimePicker.Width = 200;

dateTimePicker.Location = new Point(maxwidth + X, Y);

panel1.Controls.Add(dateTimePicker);

CreatedControls.Add(dateTimePicker);

}

else if (eft.Fields[i].Check)

{

CheckBox checkBox = new CheckBox();

checkBox.Tag = eft.Fields[i];

checkBox.Name = "checkBox" + i;

checkBox.Width = 200;

checkBox.Location = new Point(maxwidth + X, Y);

panel1.Controls.Add(checkBox);

CreatedControls.Add(checkBox);

}

else

{

TextBox textBox = new TextBox();

textBox.Tag = eft.Fields[i];

if (eft.Fields[i].CheckInt)

textBox.TextChanged += delegate (object sender, EventArgs arg)

{

if (textBox.Text.Length == 0)

return;

if (!Char.IsDigit(textBox.Text[textBox.Text.Length - 1]))

{

MessageBox.Show("Должны вводиться числа!", "Ошибка!", MessageBoxButtons.OK);

textBox.Text = "";

return;

}

};

textBox.Name = "textBox" + i;

textBox.Width = 200;

textBox.Location = new Point(maxwidth + X, Y);

panel1.Controls.Add(textBox);

CreatedControls.Add(textBox);

}

Y += 25;

}

panel1.Width = maxwidth + 240 + (havecombobox ? 200 : 0);

panel1.Height = Y+40;

//panel1.AutoSize = true;

this.Width = panel1.Width+17;

this.Height = panel1.Height + 100;

Button button = new Button();

button.Cursor = Cursors.Hand;

button.Click += SendQuery;

button.Name = "ButtonOk";

button.Text = "OK";

button.Width = 100;

button.Location = new Point(panel1.Width/2-button.Width/2, Y+10);

panel1.Controls.Add(button);

pictureBoxHeader.Width = panel1.Width;

label1.Location = new Point(pictureBoxHeader.Width / 2 - label1.Width / 2, label1.Location.Y);

}

private void Ef\_FormClosing(object sender, FormClosingEventArgs e)

{

//MessageBox.Show((sender as EditForm).lastInsertIndex.ToString());

if ((sender as EditForm).EditFormTable.Type == EditType.Add)

{

if ((sender as EditForm).lastInsertIndex == -1)

return;

UpdateComboBox(((sender as EditForm).Tag as ComboBox), ((sender as EditForm).Tag as ComboBox).Tag as FieldForm);

((sender as EditForm).Tag as ComboBox).SelectedIndex = (((sender as EditForm).Tag as ComboBox).Tag as FieldForm).IDs.IndexOf((sender as EditForm).lastInsertIndex);

}

else

{

UpdateComboBox(((sender as EditForm).Tag as ComboBox), ((sender as EditForm).Tag as ComboBox).Tag as FieldForm);

((sender as EditForm).Tag as ComboBox).SelectedIndex = (((sender as EditForm).Tag as ComboBox).Tag as FieldForm).IDs.IndexOf((sender as EditForm).ID);

}

}

private List<string> GetValuesForComboBox(FieldForm fieldf)

{

fieldf.IDs.Clear();

List<string> stringsvalues = new List<string>();

DataTable dt;

string query = "SELECT";

for (int i = 0; i < fieldf.FieldFullNames.Count; i++)

query += (i == 0? " ": ", ") + fieldf.FieldFullNames[i];

query += " FROM " + fieldf.FieldFullNames[0].Split('.')[0];

dt = \_database.SelectQuery(query);

bool firstid;

string str = string.Empty;

foreach(DataRow row in dt.Rows)

{

str = "";

firstid = true;

foreach (DataColumn column in dt.Columns)

{

if (firstid)

fieldf.IDs.Add(int.Parse(row[column].ToString()));

else

str += row[column] + " ";

firstid = false;

}

stringsvalues.Add(str);

}

return stringsvalues;

}

private void FillValues()

{

DataTable dt = \_database.SelectQuery(string.Format("SELECT \* FROM {0}", EditFormTable.NameTable));

dt = \_database.SelectQuery(string.Format("SELECT \* FROM {0} WHERE {1} = {2}", EditFormTable.NameTable, dt.Columns[0].ColumnName, ID));

for(int i = 1; i < dt.Columns.Count; i++)

{

if (dt.Columns[i] == null)

continue;

if (CreatedControls[i - 1] is ComboBox)

{

FieldForm ff = (CreatedControls[i - 1] as ComboBox).Tag as FieldForm;

(CreatedControls[i - 1] as ComboBox).SelectedIndex = ff.IDs.IndexOf((int)dt.Rows[0][dt.Columns[i]]);

}

else if (CreatedControls[i - 1] is TextBox)

{

(CreatedControls[i - 1] as TextBox).Text = dt.Rows[0][dt.Columns[i]].ToString();

}

else if (CreatedControls[i - 1] is DateTimePicker)

{

string date = dt.Rows[0][dt.Columns[i]].ToString();

string[] dateval;

date = date.Split(' ')[0];

dateval = date.Split('/');

DateTime dtime = new DateTime(int.Parse(dateval[2]), int.Parse(dateval[0]), int.Parse(dateval[1]));

(CreatedControls[i - 1] as DateTimePicker).Value = dtime; //= dtime;

//(CreatedControls[i - 1] as DateTimePicker).Value = new DateTime(int.Parse(dateval[2]), int.Parse(dateval[1]), int.Parse(dateval[0]));//.Year = (int)dateval[2];

//(CreatedControls[i - 1] as DateTimePicker).SelectedIndex = ff.IDs.IndexOf((int)dt.Rows[0][dt.Columns[i]]);

}

else

{

bool valcheck = (bool)dt.Rows[0][dt.Columns[i]];

(CreatedControls[i - 1] as CheckBox).Checked = valcheck;

//(CreatedControls[i - 1] as CheckBox).Checked = valcheck == 1 ? true : false;

}

}

}

private void UpdateComboBox(ComboBox cb, FieldForm fieldf) //Заполняем ComboBox данными

{

cb.Items.Clear();

List<string> valcombobox = GetValuesForComboBox(fieldf);

for (int q = 0; q < valcombobox.Count; q++)

cb.Items.Add(valcombobox[q]);

}

private void SendQuery(object sender, EventArgs e) //Выполняем запрос

{

if (EmptyValueInField())

return;

if (EditFormTable.Type == EditType.Add)

\_database.Query(CreateInsertQuery());

else

\_database.Query(CreateUpdateQuery());

lastInsertIndex = (int)\_database.mysqlcommand.LastInsertedId;

}

private string CreateInsertQuery() //Создание запроса на добавление запись в таблицу

{

string query = "INSERT INTO " + EditFormTable.NameTable + " (";

DataTable dt = \_database.SelectQuery("SELECT \* FROM " + EditFormTable.NameTable);

bool first = true;

bool cont = true;

foreach (DataColumn dc in dt.Columns)

{

if (cont)

{

cont = false;

continue;

}

if (first)

{

query += dc.ColumnName;

first = false;

}

else

query += ", " + dc.ColumnName;

}

query += ") values(";

first = true;

//CreatedControls

foreach (Control control in CreatedControls)

{

if (control is Label || control is Button)

continue;

if (first)

{

query += " '";

//query += (control is ComboBox) ? " '" + GetValueFieldComboBox((control as ComboBox), ((control as ComboBox).Tag as FieldForm)) + "' " : " '"+(control as TextBox).Text+"' ";

first = false;

}

else

query += ", '";

//query += (control is ComboBox) ? ", '"+GetValueFieldComboBox((control as ComboBox), ((control as ComboBox).Tag as FieldForm))+"' " : ", '" + (control as TextBox).Text + "' ";

//MessageBox.Show(control.Name);

if (control is ComboBox)

{

query += GetValueFieldComboBox((control as ComboBox), (control as ComboBox).Tag as FieldForm) + "' ";

}

else if (control is TextBox)

{

query += (control as TextBox).Text + "' ";

}

else if (control is DateTimePicker)

{

//query += (control as DateTimePicker).Value.Date + "' ";

query += string.Format("{0}.{1}.{2}", (control as DateTimePicker).Value.Year, (control as DateTimePicker).Value.Month, (control as DateTimePicker).Value.Date.Day) + "' ";

}

else

{

query += ((control as CheckBox).Checked ? 1 : 0) + "' ";

}

}

query += ");";

return query;

}

private string CreateUpdateQuery() //Создание запроса на изменении записи в таблице

{

string query = "UPDATE " + EditFormTable.NameTable + " SET ";

DataTable dt = \_database.SelectQuery("SELECT \* FROM " + EditFormTable.NameTable);

bool first = true;

for(int i = 1; i < dt.Columns.Count; i++)

{

if (first)

{

//query += dt.Columns[i].ColumnName + " = '" + ((CreatedControls[i - 1] is ComboBox) ? GetValueFieldComboBox((CreatedControls[i - 1] as ComboBox), (CreatedControls[i - 1] as ComboBox).Tag as FieldForm)+"' ": (CreatedControls[i - 1] as TextBox).Text+"' ");

first = false;

}

else

query += ", ";//query += ", "+dt.Columns[i].ColumnName + " = '" + ((CreatedControls[i - 1] is ComboBox) ? GetValueFieldComboBox((CreatedControls[i - 1] as ComboBox), (CreatedControls[i - 1] as ComboBox).Tag as FieldForm) + "' " : (CreatedControls[i - 1] as TextBox).Text + "' ");

query += dt.Columns[i].ColumnName + " = '";

if (CreatedControls[i - 1] is ComboBox)

{

query += GetValueFieldComboBox((CreatedControls[i - 1] as ComboBox), (CreatedControls[i - 1] as ComboBox).Tag as FieldForm) + "' ";

}

else if (CreatedControls[i - 1] is TextBox)

{

query += (CreatedControls[i - 1] as TextBox).Text + "' ";

}

else if (CreatedControls[i - 1] is DateTimePicker)

{

query += string.Format("{0}.{1}.{2}", (CreatedControls[i - 1] as DateTimePicker).Value.Year, (CreatedControls[i - 1] as DateTimePicker).Value.Month, (CreatedControls[i - 1] as DateTimePicker).Value.Date.Day) + "' ";

}

else

{

query += ((CreatedControls[i - 1] as CheckBox).Checked? 1 : 0) + "' ";

}

}

query += "WHERE " + dt.Columns[0] + " = '" + ID + "'";

return query;

}

private int GetValueFieldComboBox(ComboBox cb, FieldForm fieldf) //Получаем значение ID из другой таблицы

{

return fieldf.IDs[cb.SelectedIndex];

/\*

for (int i = 0; i < fieldf.IDs.Count; i++)

if (cb.SelectedIndex == i)

return fieldf.IDs[i];

return -1;

\*/

}

private void pictureBoxHeader\_Resize(object sender, EventArgs e)

{

label1.Location = new Point(pictureBoxHeader.Width / 2 - label1.Width / 2, label1.Location.Y);

//panel1.Location = new Point(this.Width / 2 - panel1.Width / 2, panel1.Location.Y);

}

private void EditForm\_Resize(object sender, EventArgs e)

{

//pictureBoxHeader.Width = this.Width;

}

private bool EmptyValueInField()

{

foreach(Control c in CreatedControls)

{

if (c is ComboBox)

{

if ((c as ComboBox).SelectedIndex == -1)

{

FieldForm ff = (c as ComboBox).Tag as FieldForm;

MessageBox.Show("В поле '" + ff.FieldCaption + "' отсутствует значение!", "Ошибка", MessageBoxButtons.OK);

return true;

}

}

else if (c is TextBox)

{

if ((c as TextBox).Text == "")

{

FieldForm ff = (c as TextBox).Tag as FieldForm;

MessageBox.Show("В поле '" + ff.FieldCaption + "' отсутствует значение!", "Ошибка", MessageBoxButtons.OK);

return true;

}

}

}

return false;

}

}

# Приложение Б

(обязательное)

Диаграмма вариантов использования

# Приложение В

(обязательное)

Диаграмма деятельности