

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных Технологий

«Допущен к защите»
Заведующий кафедрой _____

(Ф.И.О., ученая степень, звание)

_____ « _____ » 20__ г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка программного обеспечения для охранного мониторинга.

Специальность _____

Выполнил (а) Смолин М.И. БВТУ-10
(Фамилия и инициалы) группа

Научный руководитель Ахметова Майра Ахметовна
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:

Брессева З.Д. ст. преподаватель
(Фамилия и инициалы, ученая степень, звание)
Брессева « 05 » июня 20 14 г.
(подпись)

по безопасности жизнедеятельности:

Шайдарбекова М.К., К.Х.Н., доцент
(Фамилия и инициалы, ученая степень, звание)
Шайдарбекова « 9 » 06 20 14 г.
(подпись)

по применению вычислительной техники:

(Фамилия и инициалы, ученая степень, звание)

_____ « _____ » 20__ г.
(подпись)

(Фамилия и инициалы, ученая степень, звание)

_____ « _____ » 20__ г.
(подпись)

Нормоконтролер: Рахимова З.М. ст. преподаватель
(Фамилия и инициалы, ученая степень, звание)

Рах _____ « _____ » 20__ г.
(подпись)

Рецензент: Авалбаев Е.А. К.Ф.М.Н.
(Фамилия и инициалы, ученая степень, звание)

Авалбаев _____ « _____ » 20__ г.
(подпись)

Алматы 2014 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет 9230 и ДС
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ

на выполнение дипломного проекта

Студент Смолин Павел Игоревич
(фамилия, имя, отчество)

Тема проекта Разработка программного обеспечения для охранного мониторинга

утверждена приказом ректора № ___ от «___» сентября 20___ г.

Срок сдачи законченной работы «___» _____ 20___ г.

Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта

Разработка программного обеспечения для наблюдения и обработки сигналов от объектов мониторинга

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:

Разработать базу данных для хранения всей информации по охраняемым объектам
Разработать программу обработки поступающих сигналов по RS-232 и запись их в базу данных. Разработать приложение для эмуляции работы тревожных сигналов, а так же создание, обновление, удаление картотеки объектов.
Произвести технико-экономические обоснования. Выполнить анализ условий труда, рассмотреть вопросы загрузки, загрузки

Перечень графического материала (с точным указанием обязательных чертежей)

- Структура программного продукта - 1.4
- Задание «Модель формы» - 3.1
- Планшеты конспекта «Свойства объема» - 3.3
- Процесс и редактирование картоны объема 3.6
- Формы «Окто кривой» - 3.4
- Основной алгоритм работы сервера - 4.1

Рекомендуемая основная литература

1. Виллерс Р. Программирование баз данных на SQL. Базовый курс / Р. Виллерс. - М.: Вильямс, 2008.
2. Горелов, А. Экономические работы с СУБД / А. Горелов, С. Макашаринел. - СПб.: Питер, 2005-2006.
3. Малин С. С++ для профессионалов / С. Малин. - М.: Вильямс, 2006.
4. Бьерк С. Язык программирования С++ / С. Бьерк. - М.: Вильямс, 2005.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
Демо-школа	Зряшева З.Д.	30.05 - 05.06.14	Зряшева
Будущ. жизнь	Майфардкова Ж.К.	12.05 - 5.06.14	Майфард

АҢДАТПА

Осы шығарылым жұмысында берілген тапсырыстар бойынша бағдарламалық кешен зерттелген, олар күзет насындарына хабарлманы қабылдауына және талқылауға мүмкіндік береді

Қосымшалардың әртүрлі шарттарға бейімділігін қамтамасыз ету үшін бұл бағдарламалардың әртүрлі компьютерлерде бірдей жұмыс істеуіне мүмкіндік беретін технологияларға таңдау жасалған. Сонымен бірге бағдарламамен ыңғайлы жұмыс істеуді қамтамасыз ететін алгоритмдер құрылған.. Ғылыми-зерттеу жұмысына шыққан шығындар есептелді. Өміртіршілік қауіпсіздігі мәселелері, жұмыс шарттары қарастырылды, жұмыс орындарының жарықтандырылуына және ауа баптауларына есептеулер жүргізілді.

АННОТАЦИЯ

В данной выпускной работе согласно заданию была осуществлена разработка программного комплекса, позволяющего принимать и обрабатывать поступающие сообщения от объектов охраны.

Для обеспечения адаптивности приложения к различным условиям осуществлён выбор технологий, позволяющих программе одинаково функционировать на различных компьютерах, а также разработаны алгоритмы, которые обеспечивают удобную систему работы с программой и наглядность ее алгоритма. Произведен расчет затрат на научно-исследовательскую работу. Рассмотрены вопросы безопасности жизнедеятельности, условий труда, выполнен расчёт кондиционирования рабочего помещения.

Annotation

In this final year project was developed a system that allows to process received and sent messages from secured objects. In order to keep the level of adaptivity of the application to different conditions there is a choice of which technology to use, meanwhile keeping the same level of functionality. In addition to that algorithms were developed that are ensuring comfortable use of it and clear vision of the work flow. Research process expenditures were estimated. Life safety, work place conditions, and air conditioning of the work place was calculated and researched.

Содержание

Введение.....	7
1 Постановка задачи и выбор средства разработки.....	8
1.1 Анализ предметной области.....	8
1.2 Требования к разработке.....	8
1.3 Выбор средства разработки.....	9
1.4 Структура программного продукта.....	10
2 Проектирование базы данных.....	12
2.1 Основы теории баз данных.....	12
2.2 Особенности проектирования баз данных.....	17
2.3 Установка MySQL.....	21
2.4 Среда разработки и администрирования MySQL.....	22
2.5 Создание базы.....	26
2.6 Создание таблиц.....	28
3 Разработка программного продукта.....	31
3.1 Создание интерфейса программы.....	31
3.2 Создание модуля для доступа к базе данных.....	38
3.3 Создание потока отслеживания тревог.....	40
3.4 Обработка команд.....	42
3.5 Разработка программы обработки пакетов сигнала.....	49
3.6 Создание классов для доступа и работы с базой данных.....	53
3.7 Разработка модуля для работы с RS-232.....	56
3.8 Разработка основного алгоритма работы сервера.....	59
4 Технико-экономическое обоснование.....	63
4.1 Определение объема и трудоемкости разработки программного обеспечения.....	63
4.2 Расчет затрат на разработку информационных технологий.....	68
4.3 Вывод по расчету затрат на разработку информационных технологий.....	74
5 Безопасность жизнедеятельности.....	75
5.1 Анализ условий труда.....	75
5.2 Защитное заземление, зануление.....	71
5.3 Расчет системы кондиционирования помещения.....	76
Заключение.....	85
Список используемой литературы.....	86

Введение

Актуальность задачи обеспечения сохранности материальных ценностей на объектах не ставится под сомнения.

Для решения задачи построения системы безопасности следует обозначить основные этапы. Для этого необходимо определить:

- от чего защищать (угрозы);
- как и какими методами (средства).

Защита современного здания (помещения) - задача, решаемая, с помощью современных средств охранной сигнализации с целью предотвращения противоправных посягательств на материальные ценности расположенные в охраняемом помещении.

Важную и действенную роль в решении этой задачи играет оборудование помещений автоматическими установками охранной сигнализации.

Система охранной сигнализации функционирует, как правило, в нерабочее время. При этом окна, двери и заграждения находятся в закрытом состоянии.

Чтобы предотвратить попытку несанкционированного доступа в охраняемые помещения, и в следствии как можно скорее задержать преступника, необходимо сократить время обнаружения попытки несанкционированного доступа и передачи сообщения на ПЦН, для чего успешно применяются средства автоматики.

На современном этапе широкое распространение получила централизованная охрана объектов. Принцип организации такой охраны заключается в следующем: установленные на объектах средства сигнализации на период охраны подключаются через канал связи (как правило, сети GSM, радиоканалы) к ПЦН (пульт централизованного наблюдения).

При проникновении на один из охраняемых объектов посторонних лиц, срабатывают средства охранной сигнализации, установленные на объекте, и на ПЦН поступает сигнал тревоги.

Целью дипломного проекта является разработка программного обеспечения системы охранной сигнализации в.

Для достижения поставленной цели необходимо решить следующие задачи:

- разработка базы данных охранной системы;
- разработка серверного программного обеспечения;
- разработка программного обеспечения для операторов пульта централизованного наблюдения.

1 Постановка задачи и выбор средства разработки

В дипломном проекте в соответствии с заданием автоматизируется деятельность ПЦН, осуществляющих наблюдение охраняемых объектов.

Предметом области автоматизации являются некоторые функции деятельности операторов ПЦН. Основное направление деятельности компании по мониторингу - осуществление охранного мониторинга объектов. Основными объектами являются : магазины, ломбарды, офисы, квартиры и другие объекты.

1.1 Анализ предметной области

Все операции по осуществлению мониторинга выполняют операторы ПЦН, с помощью не сетевого программного обеспечения, т. е. для внесения изменения в карточку объекта, необходимо вручную произвести изменения на каждом компьютере. С ростом числа объектов, увеличивается общее время выполнения операций по внесению изменений в карточку объекта. Так же все события от объектов приходится выполнять на каждом компьютере.

Для реализации выше указанных проблем можно сформулировать следующие задачи :

- проектирование и создание общей базы данных. Сущность задачи состоит в создании таблиц базы данных и разработке структуры данных;
- разработка единого серверного программного обеспечения. Сущность задачи состоит в разработке программы-сервера для обработки поступающих сигналов с объектов;
- разработка программного обеспечения для контроля поступающих сигналов. Сущность задачи состоит в написании сетевой версии программы для операторов ПЦН.

В дипломном проекте предлагается полностью перейти на сетевую версию работы базы данных и всего программного обеспечения ПЦН.

1.2 Требования к разработке

При разработке программного обеспечения были поставлены следующие требования:

- возможность функционирования в рамках внутренней локальной сети предприятия;
- регистрация всей информации связанной с объектами;
- хранение данных о состоянии объекта;
- возможность выдачи информации на экран монитора;
- обеспечение высокой надежности хранения информации;

- возможность расчета технологических данных заказа.

Представленные выше требования к программному обеспечению, могут быть реализованы при помощи выбора средств разработки, отвечающих данным требованиям.

1.3 Выбор средства разработки

Для обеспечения высокой надежности хранения данных была выбрана система управления баз данных - MySQL. MySQL - очень быстрая, надежная система управления реляционными базами данных (СУРБД). База данных позволяет эффективно хранить, искать, сортировать и получать данные. Сервер MySQL управляет доступом к данным, позволяя работать с ними одновременно нескольким пользователям, обеспечивает быстрый доступ к данным и гарантирует предоставление доступа только имеющим на это право пользователям. Следовательно, MySQL является многопользовательским, многопоточковым сервером. Он применяет SQL (Structured Query Language - язык структурированных запросов), используемый по всему миру стандартный язык запросов в базы данных. MySQL появился на рынке в 1996 г., но его разработка началась еще в 1979 г.

В настоящее время пакет MySQL доступен как программное обеспечение с открытым исходным кодом, но в случае необходимости можно получить и коммерческие лицензии.. Она обеспечивает доступ к обширным ресурсам, ведущую в отрасли производительность и масштабируемость корпоративного класса, высочайший уровень безопасности, высочайший уровень доступности и высочайший уровень надежности . MySQL написана на языках C и C++ и работает под управлением различных операционных систем. Есть примеры применения MySQL для поддержания баз данных состоящих из 60000 таблиц, насчитывающих более 5 млрд. строк.

Выбор операционной системы для сервера базы данных и сервера обработки сигналов пал на Ubuntu, и MySQL как нельзя больше подходит для этой ОС. Ubuntu - операционная система, основанная на Debian GNU/Linux. Основным разработчиком и спонсором является компания Canonical. В настоящее время проект активно развивается и поддерживается свободным сообществом.

По утверждениям Canonical, операционную систему Ubuntu использует примерно 20 миллионов пользователей, что делает его самым популярным дистрибутивом Linux для десктопов. Он является 4-м в списке самых популярных дистрибутивов Linux для серверов и его популярность быстро растёт.

Для написания серверной части отвечающей за обработку сигналов приходящих от объектов охраны, была выбрана

кроссплатформенный инструментарий разработки ПО на языке программирования C++ - QT. А в качестве среды разработки была выбрана свободная среда разработки программного обеспечения QDevelop. Цель QDevelop состоит в том, чтобы обеспечивать кроссплатформенное программирование в наиболее используемых средах, в GNU/Linux, Windows NT и Mac OS X, используя один и тот же IDE. Каждая из этих сред уже использует свою собственную, зачастую более высокопроизводительную IDE, примерами являются Visual Studio в Windows и KDevelop в Linux. Их неудобство заключается ограничением на одну среду.

QDevelop не представляет собой облегчённую или имитирующую версию KDevelop. Эта IDE, использующая Qt4, абсолютно независима от KDevelop. Менее функциональный, но более лёгкий в использовании QDevelop способен работать на разных платформах. Другая отличительная черта QDevelop от KDevelop — совершенно разный исходный код.

В особенности QDevelop входит поддержка Qt Designer для создания графического интерфейса, что превращает связку из QDevelop и Qt Designer в среду визуальной разработки и тем самым соответствует концепту быстрой разработки приложений (RAD).

Так же для разработки приложения для операторов была выбрана среда разработки Embarcadero Delphi, ранее Borland Delphi и CodeGear Delphi, — интегрированная среда разработки ПО для Microsoft Windows, Mac OS, iOS и Android на языке Delphi (ранее носившем название Object Pascal), созданная первоначально фирмой Borland и на данный момент принадлежащая и разрабатываемая Embarcadero Technologies. Embarcadero Delphi является частью пакета Embarcadero RAD Studio. Среда предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows, Mac OS X, а также IOS и Android. Благодаря уникальной совокупности простоты языка и генерации машинного кода, позволяет непосредственно, и, при желании, достаточно низкоуровнево взаимодействовать с операционной системой, а также с библиотеками, написанными на C/C++. Созданные программы не зависимы от стороннего ПО, как-то Microsoft .NET Framework, или Java Virtual Machine. Выделение и освобождение памяти контролируется в основном пользовательским кодом, что, с одной стороны, ужесточает требования к качеству кода, а с другой - делает возможным создание сложных приложений, с высокими требованиями к отзывчивости (работа в реальном времени).

1.4 Структура программного продукта

Программный продукт состоит из 3 основных частей. Первая часть это база данных на основе MySQL, которая хранит всю информацию для

работы программного продукта. ReadBuf – консольная программа, для удобства запущенная на том же компьютере что и база данных MySQL. ReadBuf – выполняет обработку и запись в базу данных поступающих сигналов от оборудования, которое стоит на охраняемых объектах. Приложение оператора или администратора – выполняет функцию заполнения базы данных карточками охраняемых объектов, их редактированием и удалением, просмотра отслеживания событий охраняемых объектов, обработка тревожных событий. Схема структуры изображена на рисунке 1.4.1.

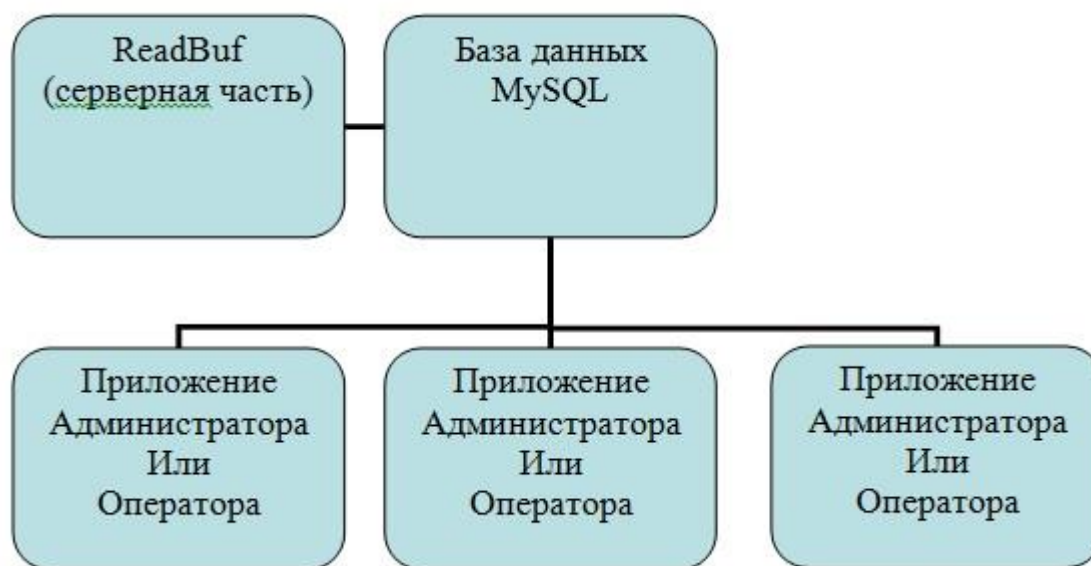


Рисунок 1.4.1 Структура программного продукта

2 Проектирование базы данных

2.1 Основы теории баз данных

С развитием экономики возрастает объем взаимосвязанных данных, необходимых для решения коммерческих и административных задач. Взаимосвязанные данные называют информационной системой. Такая система в первую очередь призвана облегчить труд человека, но для этого она должна как можно лучше соответствовать очень сложной модели реального мира.

Ядром информационной системы являются хранимые в ней данные. На любом предприятии данные различных отделов, как правило, пересекаются, то есть используются в нескольких подразделениях или вообще являются общими. Например, для целей управления часто нужна информация по всему предприятию. Хранившиеся в информационной системе данные должны быть легко доступны в том виде, в каком они нужны для конкретной производственной деятельности предприятия. При этом не имеет существенного значения способ хранения данных. Сегодня на предприятии мы можем встретить систему обработки данных традиционного типа, в которой служащий вручную помещает данные в скоросшиватель, и рядом с ней – современную систему с применением самой быстродействующей ЭВМ, сложнейшего оборудования и программного обеспечения. Несмотря на поразительную несхожесть, обе эти системы обязаны предоставлять достоверную информацию в определенное время, определенному лицу, в определенном месте и с ограниченными затратами.

Построение информационных систем основывается на понятиях теории баз данных.

Предметной областью называется часть реальной системы, представляющая интерес для данного исследования.

При проектировании автоматизированных информационных систем предметная область отображается моделями данных нескольких уровней. Число используемых уровней зависит от сложности системы, но в любом случае включает логический и физический уровни. Предметная область может относиться к любому типу организации (например, банк, университет, малое предприятие или завод).

Необходимо различать полную предметную область (предприятие) и организационную единицу этой предметной области. Организационная единица в свою очередь может представлять свою предметную область (отделы).

Информация, необходимая для описания предметной области, зависит от реальной модели и может включать сведения о персонале,

заработной плате, товарах, накладных, счетах, отчетах по сбыту, то есть сведения о людях, местах, предметах, событиях и понятиях. Объектом называется элемент информационной системы, информацию о котором мы сохраняем. В реляционной теории баз данных объект называется сущностью.

Объект может быть реальным (например, человек, какой-либо предмет или населенный пункт) и абстрактным (например, событие, счет покупателя или изучаемый студентами курс). Так, для складского учета примерами объектов могут служить поставщик, товар, поручение и т. д. Каждый объект обладает набором определенных свойств, которые запоминаются в информационной системе. При обработке данных часто приходится иметь дело с совокупностью однородных объектов, например служащие, и записывать информацию об одних и тех же свойствах для каждого из них.

Классом объектов называют совокупность объектов, обладающих одинаковым набором свойств. Таким образом, для объектов одного класса набор свойств будет одинаков.

Объекты и их свойства являются понятиями реального мира. Для информационного пространства употребляется понятие атрибута объекта.

Атрибут - это информационное отображение свойств объекта. Каждый объект характеризуется рядом основных атрибутов. Например, сотрудник предприятия имеет такие атрибуты, как фамилию, имя, отчество, адрес и возможно идентификационный номер. Каждый атрибут в модели должен иметь уникальное имя - идентификатор. Атрибут при реализации информационной модели на каком-либо носителе информации часто называют элементом данных, полем данных или просто полем.

Таблица - это некоторая регулярная структура, состоящая из конечного набора однотипных записей.

Каждая запись одной таблицы состоит из конечного числа полей, причем конкретное поле каждой записи одной таблицы может содержать данные только одного типа.

Значение данных представляет собой действительные данные, содержащиеся в каждом элементе данных. В зависимости от того, как элементы данных описывают объект, их значения могут быть количественными, качественными или описательными.

Информацию о некоторой предметной области можно представить с помощью нескольких объектов, каждый из которых описывается несколькими элементами данных. Принимаемые элементами данных значения называются данными. Единичный набор принимаемых элементами данных значений называется экземпляром объекта. Объекты связываются между собой определенным образом. Соответствующая модель объектов с составляющими их элементами данных и взаимосвязями называется концептуальной моделью. Концептуальная модель дает общее

представление о потоке данных в предметной области.

Некоторые элементы данных обладают важным для построения информационной модели свойством. Если известно значение, которое принимает такой элемент данных объекта, мы можем идентифицировать значения, которые принимают другие элементы данных этого же объекта.

Ключевым элементом данных называется такой элемент, по которому можно определить значения других элементов данных.

Однозначно идентифицировать объект могут два и более элемента данных. В этом случае их называют «кандидатами» в ключевые элементы данных. Вопрос о том, какой из кандидатов использовать для доступа к объекту, решается разработчиком системы. Выбирать ключевые элементы данных следует тщательно, поскольку правильный выбор способствует созданию достоверной концептуальной модели данных.

Первичный ключ - это атрибут (или группа атрибутов), которые единственным образом идентифицируют каждую строку в таблице.

Понятие первичного ключа является исключительно важным в связи с понятием целостности баз данных.

Альтернативный ключ - это атрибут (или группа атрибутов), несовпадающий с первичным ключом и уникально идентифицирующий экземпляр объекта. Например, для объекта «служащий», который имеет атрибуты «идентификатор», «фамилия», «имя», «отчество», группа атрибутов «фамилия», «имя», «отчество» может являться альтернативным ключом по отношению к атрибуту «идентификатор».

Запись данных - это совокупность значений связанных элементов данных. Записи хранятся на некотором носителе, в качестве которого может выступать человеческий мозг, лист бумаги, память ЭВМ, внешнее запоминающее устройство и т.д.

В современных базах данных допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (например, суммы в денежных единицах), а также данных специального формата (дата, время, временной интервал и т.д.). В любом случае при выборе типа данных необходимо учитывать возможности системы управления базами данных (СУБД), с помощью которой реализуется физическая модель информационной системы.

Доменом называется набор значений элементов данных одного типа, отвечающий поставленным условиям.

В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных, который «забраковывает» недопустимые значения. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена. Понятие домена может также характеризоваться как потенциальное множество допустимых значений

одного типа.

Представление - это сохраняемый в базе данных именованный запрос на выборку данных (из одной или нескольких таблиц).

Результатом выполнения любого запроса на выборку данных является таблица, и поэтому концептуально можно относиться к любому представлению как к таблице.

Связь - это функциональная зависимость между сущностями.

Если между некоторыми сущностями существует связь, то факты из одной сущности ссылаются или некоторым образом связаны с фактами из другой сущности.

Поддержание непротиворечивости функциональных зависимостей между сущностями называется ссылочной целостностью. Поскольку связи содержатся «внутри» реляционной модели, реализация ссылочной целостности может выполняться как приложением, так и самой системой управления базами данных (СУБД) с помощью механизмов декларативной ссылочной целостности и триггеров.

Связи могут быть представлены пятью основными характеристиками:

- родительская сущность;
- дочерняя (зависимая) сущность;
- мощность связи;
- допустимость пустых значений.

Связь называется идентифицирующей, если экземпляр дочерней сущности идентифицируется (однозначно определяется) через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой.

Связь называется не идентифицирующей, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав не ключевых атрибутов дочерней сущности.

Мощность связи представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству экземпляров дочерней сущности. Для любой связи, кроме неспецифической, эта связь записывается как 1:n.

Хранимые процедуры - это приложение (программа), объединяющее запросы и процедурную логику (операторы присваивания, логического ветвления и т.д.) и хранящиеся в базе данных.

Хранимые процедуры позволяют содержать вместе с базой данных достаточно сложные программы, выполняющие большой объем работы без передачи данных по сети и взаимодействия с клиентом. Как правило,

программы, записываемые в хранимых процедурах, связаны с обработкой данных. Тем самым база данных может представлять собой функционально самостоятельный уровень приложения, который может взаимодействовать с другими уровнями для получения запросов или обновления данных.

Правила позволяют вызывать выполнение заданных действий при изменении или добавлении данных в базу данных и тем самым контролировать истинность помещаемых в нее данных.

Обычно действие - это вызов определенной процедуры или функции. Правила могут ассоциироваться с полем или записью и, соответственно, срабатывать при изменении данных в конкретном поле или записи таблицы. Нельзя использовать правила при удалении данных.

В отличие от ограничений, которые являются лишь средством контроля относительно простых условий корректности ввода данных, правила позволяют проверять и поддерживать сколь угодно сложные отношения между элементами данных в базе данных.

Триггеры - это предварительно определенное действие или последовательность действий, автоматически осуществляемых при выполнении операций обновления, добавления или удаления данных .

Триггер является мощным инструментом контроля над изменением данных в базе данных, а также помогает программисту автоматизировать операции, которые должны выполняться в этом случае. Триггер выполняется после проверки правил обновления данных. Ни пользователь, ни приложение не могут активизировать триггер, он выполняется автоматически, когда пользователь или приложение выполняют с базой данных определенные действия. Триггер включает в себя следующие компоненты:

- ограничения, для реализации которых собственно и создается триггер;

- событие, которое будет характеризовать возникновение ситуации, требующей проверки ограничений. События чаще всего связаны с изменением состояния баз данных (например, добавление записи в какую-либо таблицу), но могут учитываться и дополнительные условия (например, добавление записи только с отрицательным значением);

- Предусмотренное действие выполняется за счет выполнения процедуры или последовательности процедур, с помощью которых реализуется логика, требуемая для реализации ограничений;

- использование триггеров при проектировании баз данных позволяет получить при разработке приложения следующие преимущества;

- триггеры всегда выполняются при совершении соответствующих действий. Разработчик продумывает использование триггеров при проектировании базы данных и может больше не вспоминать о них при разработке приложения для доступа к данным. Если для работы с этой же

базой данных вы решите создать новое приложение, триггеры и там будут обрабатывать заданные ограничения.

- при необходимости триггеры можно изменять централизованно непосредственно в базе данных. Пользовательские программы, использующие данные из этой базы данных, не требуют модернизации;

- Система обработки данных, использующая триггеры, обладает лучшей переносимостью в архитектуру клиент-сервер за счет меньшего объема требуемых модификаций;

- ссылочная целостность - это обеспечение соответствия значения внешнего ключа экземпляра дочерней сущности значениям первичного ключа в родительской сущности;

Ссылочная целостность может контролироваться при всех операциях, изменяющих данные.

Для каждой связи на логическом уровне могут быть заданы требования по обработке операций добавления, обновления или удаления данных для родительской и дочерней сущности. Могут использоваться следующие варианты обработки этих событий:

- отсутствие проверки;

- проверка допустимости;

- запрет операции;

- каскадное выполнение операции обновления или удаления данных сразу в нескольких связанных таблицах;

- установка пустого (NULL) значения или заданного значения по умолчанию.

Нормализация отношений - это процесс построения оптимальной структуры таблиц и связей в реляционной базе данных.

В процессе нормализации элементы данных группируются в таблицы, представляющие объекты и их взаимосвязи. Теория нормализации основана на том, что определенный набор таблиц обладает лучшими свойствами при включении, модификации и удалении данных, чем все остальные наборы таблиц, с помощью которых могут быть представлены те же самые данные.

2.2 Особенности проектирования баз данных

Основной формой организации хранения данных в информационных системах являются базы данных. При проектировании системы обработки данных именно данные играют важнейшую роль.

Система автоматизированной обработки данных основывается на использовании определенной модели данных или информационной модели. Модель данных отражает взаимосвязи между объектами.

Процесс создания информационной модели начинается с определения концептуальных требований ряда пользователей.

Концептуальные требования могут определяться и для некоторых задач (приложений), которые в ближайшее время реализовывать не планируется. Это может несколько повысить трудоемкость работы, однако поможет наиболее полно учесть все нюансы функциональности, требуемой для разрабатываемой системы, и снизит вероятность ее переделки в дальнейшем. Требования отдельных пользователей интегрируются в едином «обобщенном представлении». Последнее называют концептуальной моделью.

Концептуальная модель представляет объекты и их взаимосвязи без указания способов их физического хранения.

Таким образом, концептуальная модель является, по существу, моделью предметной области. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели основано на анализе решаемых на этом предприятии задач по обработке данных. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных.

Концептуальная модель транслируется затем в модель данных, совместимую с выбранной СУБД. Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели. Версия концептуальной модели, которая может быть обеспечена конкретной СУБД, называется логической моделью.

Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среде хранения.

Логическая модель данных может быть реляционной, иерархической или сетевой. Пользователям выделяются подмножества этой логической модели, называемые внешними моделями, отражающие их представления о предметной области. Внешняя модель соответствует представлениям, которые пользователи получают на основе логической модели. В то время как концептуальные требования отражают представления, которые пользователи первоначально желали иметь и которые легли в основу разработки концептуальной модели. Логическая модель отображается в физическую память, такую, как диск, лента или какой-либо другой носитель информации.

Физическая модель, определяющая размещение данных, методы доступа и технику индексирования, называется внутренней моделью системы.

Внешние модели никак не связаны с типом физической памяти, в

которой будут храниться данные, и с методами доступа к этим данным. Это положение отражает первый уровень независимости данных. С другой стороны, если концептуальная модель способна учитывать расширение требований к системе в будущем, то вносимые в нее изменения не должны оказывать влияния на существующие внешние модели. Это – второй уровень независимости данных. Построение логической модели обусловлено требованиями используемой СУБД.

Все актуальные требования предметной области и адекватные им «скрытые» требования на стадии проектирования должны найти свое отражение в концептуальной модели. Конечно, нельзя предусмотреть все возможные варианты использования и изменения базы данных. Но в большинстве предметных областей такие основные данные, как объекты и их взаимосвязи, относительно стабильны. Меняются только информационные требования, то есть способы использования данных для получения информации.

Степень независимости данных определяется тщательностью проектирования базы данных. Всесторонний анализ объектов предметной области и их взаимосвязей минимизирует влияние изменения требований к данным в одной программе на другие программы. В этом и состоит всеобъемлющая независимость данных.

Иерархическая и сетевая модели данных стали применяться в системах управления базами данных в начале 60-х годов. В начале 70-х годов была предложена реляционная модель данных. Эти три модели различаются в основном способами представления взаимосвязей между объектами.

Иерархическая модель данных строится по принципу иерархии типов объектов, то есть один тип объекта является главным, а остальные, находящиеся на низших уровнях иерархии, – подчиненными. Между главным и подчиненными объектами устанавливается взаимосвязь «один ко многим». Иными словами, для данного главного типа объекта существует несколько подчиненных типов объектов. В то же время для каждого экземпляра главного объекта может быть несколько экземпляров подчиненных типов объектов.

Узлы и ветви образуют иерархическую древовидную структуру. Узел является совокупностью атрибутов, описывающих объект. Наивысший в иерархии узел называется корневым (это главный тип объекта). Корневой узел находится на первом уровне. Зависимые узлы (подчиненные типы объектов) находятся на втором, третьем и др. уровнях.

В сетевой модели данных понятия главного и подчиненного объектов несколько расширены. Любой объект может быть и главным и подчиненным (в сетевой модели главный объект обозначается термином «владелец набора», а подчиненный – термином «член набора»). Один и тот же объект может одновременно выступать и в роли владельца и в роли

члена набора. Это означает, что каждый объект может участвовать в любом числе взаимосвязей.

В реляционной модели данных объекты и взаимосвязи между ними представляются с помощью таблиц. Взаимосвязи также рассматриваются в качестве объектов. Каждая таблица представляет один объект и состоит из строк и столбцов. В реляционной базе данных каждая таблица должна иметь первичный ключ (ключевой элемент) – поле или комбинацию полей, которые единственным образом идентифицируют каждую строку в таблице. Благодаря своей простоте и естественности представления реляционная модель получила наибольшее распространение в СУБД для персональных компьютеров.

Все тонкости построения информационной модели преследуют одну-единственную цель – получить хорошую базу данных.

Существует очень простое понятие базы данных как большого по объему хранилища, в которое организация помещает все используемые ею данные и из которого различные пользователи могут их получать, используя различные приложения. Такая единая база данных представляется идеальным вариантом, хотя на практике это решение труднодостижимо. Поэтому чаще всего под базой данных понимают любой набор хранящихся в компьютере взаимосвязанных данных.

В основу проектирования базы данных должны быть положены представления конечных пользователей конкретной организации – концептуальные требования к системе.

При рассмотрении требований конечных пользователей необходимо принимать во внимание следующее:

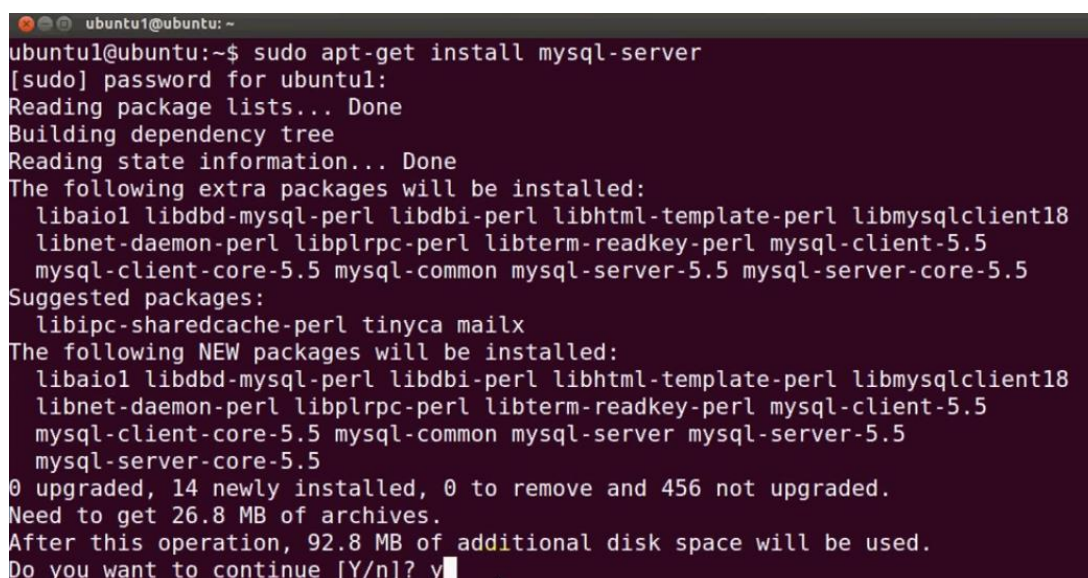
- база данных должна удовлетворять актуальным информационным потребностям организации. Получаемая информация должна по структуре и содержанию соответствовать решаемым задачам;
- база данных должна обеспечивать получение требуемых данных за приемлемое время, то есть отвечать заданным требованиям производительности;
- база данных должна удовлетворять выявленным и вновь возникающим требованиям конечных пользователей;
- база данных должна легко расширяться при реорганизации и расширении предметной области;
- база данных должна легко изменяться при изменении программной и аппаратной среды;
- багруженные в базу данных корректные данные должны оставаться корректными;
- данные до включения в базу данных должны проверяться на корректность;
- доступ к данным, размещаемым в базе данных, должны иметь только лица с соответствующими полномочиями;

При разработке логической модели базы данных прежде всего необходимо решить, какая модель данных наиболее подходит для отображения конкретной концептуальной модели предметной области. Коммерческие системы управления базами данных поддерживают одну из известных моделей данных или некоторую их комбинацию. Большинство СУБД для персональных компьютеров поддерживают реляционную модель данных.

2.3 Установка MySQL

MySQL - это быстрый, многопоточный, многопользовательский и устойчивый сервер SQL базы данных. Он предназначен как для ответственных сильнозагруженных производственных систем, так и для встраивания в массовое программное обеспечение.

Для установки MySQL необходимо запустить следующую команду из терминала: `sudo apt-get install mysql-server` (Рисунок 2.1)



```
ubuntu1@ubuntu: ~  
ubuntu1@ubuntu:~$ sudo apt-get install mysql-server  
[sudo] password for ubuntu1:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  libaiol libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18  
  libnet-daemon-perl libplrpc-perl libterm-readkey-perl mysql-client-5.5  
  mysql-client-core-5.5 mysql-common mysql-server-5.5 mysql-server-core-5.5  
Suggested packages:  
  libipc-sharedcache-perl tinyca mailx  
The following NEW packages will be installed:  
  libaiol libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18  
  libnet-daemon-perl libplrpc-perl libterm-readkey-perl mysql-client-5.5  
  mysql-client-core-5.5 mysql-common mysql-server mysql-server-5.5  
  mysql-server-core-5.5  
0 upgraded, 14 newly installed, 0 to remove and 456 not upgraded.  
Need to get 26.8 MB of archives.  
After this operation, 92.8 MB of additional disk space will be used.  
Do you want to continue [Y/n]? y
```

Рисунок 2.1 Установка MySQL

Начиная с Ubuntu 12.04, MySQL 5.5 устанавливается по умолчанию. Несмотря на 100% совместимость с MySQL 5.1, при необходимости установить версию 5.1 (например в качестве зависимой базы к другим MySQL 5.1 серверам), вы можете заменить устанавливаемый пакет на `mysql-server-5.1`.

В процессе установки у вас запросят пароль для пользователя `root` под MySQL.

Для того чтобы установить консольный клиент MySQL выполните команду: `sudo apt-get install mysql-client`.

Как только установка завершится, сервер MySQL запустится

автоматически. Вы можете использовать следующую команду в терминале для проверки, что сервер MySQL запущен: `sudo netstat -tap | grep mysql`

Когда вы запустите эту команду, вы сможете увидеть что-то похожее на следующую строку: `2556/mysql`

Если сервер не работает, вы можете набрать следующую команду для его запуска: `sudo service mysql restart`

2.4 Среда разработки и администрирования MySQL

С развитием систем баз данных процедуры инсталляции и использования MySQL становятся все проще. Судя по всему, именно простота работы с MySQL стала основной причиной широкой ее популярности среди пользователей. Однако и полностью без управления MySQL работать также не может. Администратор должен хотя бы иногда проверять согласованность и эффективность ее работы и знать, что делать при возникновении проблем. Для этого и была выбран среда разработки и администрирования dbForge Studio.

dbForge Studio является универсальным инструментом для работы с MySQL сервером, который позволяет разработчикам MySQL и администраторам баз данных MySQL создавать и выполнять запросы, разрабатывать и отлаживать процедуры и функции, а также автоматизировать управление объектами баз данных MySQL в удобном пользовательском интерфейсе.

Это приложение для работы с MySQL дополнительно предоставляет инструменты для сравнения, синхронизации, создания резервных копий баз данных по графику, а также для анализа и создания отчетов по данным таблиц MySQL.

Перечень основных возможностей dbForge Studio for MySQL:

1. интеллектуальная разработка SQL кода - Разумное дополнение кода, красивое форматирование и шаблоны кода помогают сделать написание SQL более эффективным и комфортным. Навигация по коду, получение основной информации об объектах и проверка синтаксиса производится моментально(рисунок 2.2).

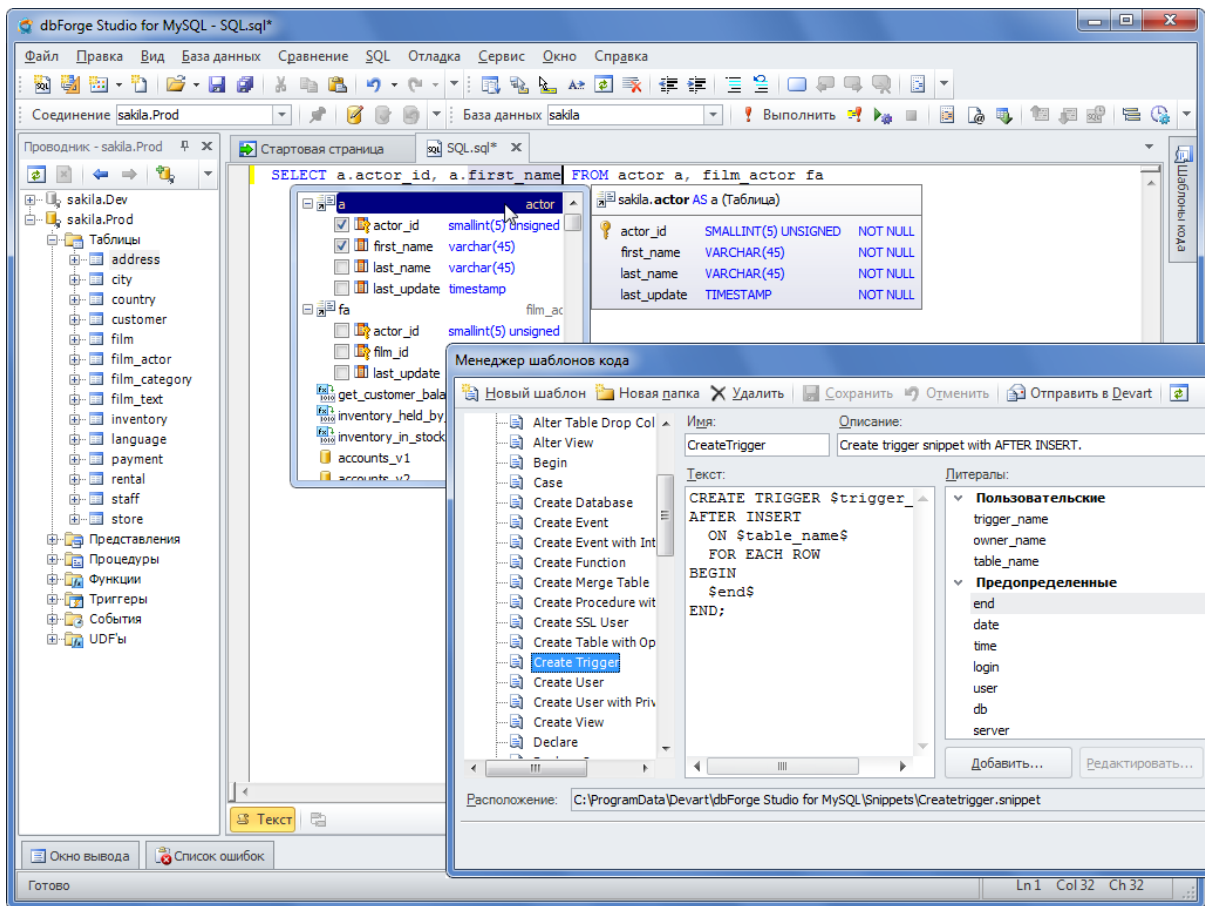


Рисунок 2.2 - Интеллектуальная разработка SQL кода

2. визуальный дизайнер запросов - Визуальное создание запросов на диаграмме, используя удобный редактор выражений. Вы можете создать запросы любой сложности за считанные мгновения. Приложение автоматически соединяет таблицы и позволяет работать с выражениями INSERT, UPDATE, DELETE (Рисунок 2.3);

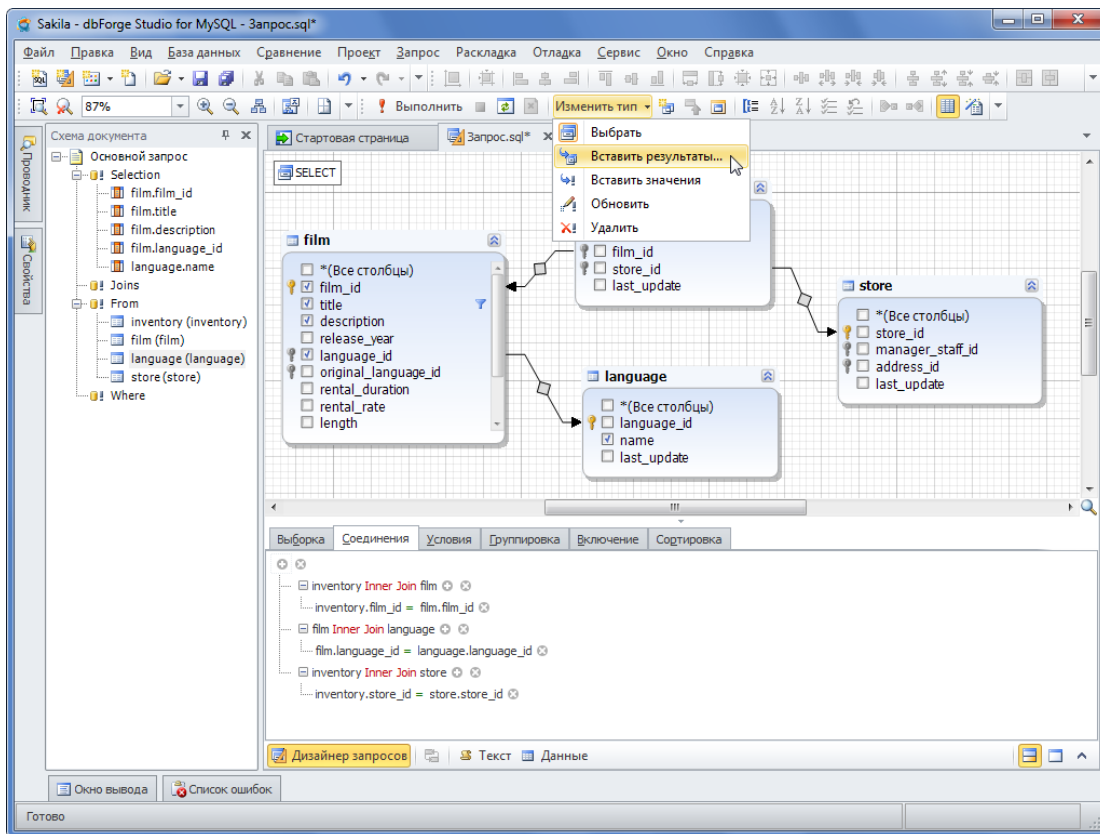


Рисунок 2.3 - Визуальный дизайнер запросов

3.Дизайнер баз данных – Используйте диаграмму базы данных для создания или реверсивного проектирования, переделки, анализа и печати баз данных MySQL, а также для: просмотра связей по внешним ключам, отображения объектов БД со свойствами, выполнение хранимых процедур (рисунок 2.4);

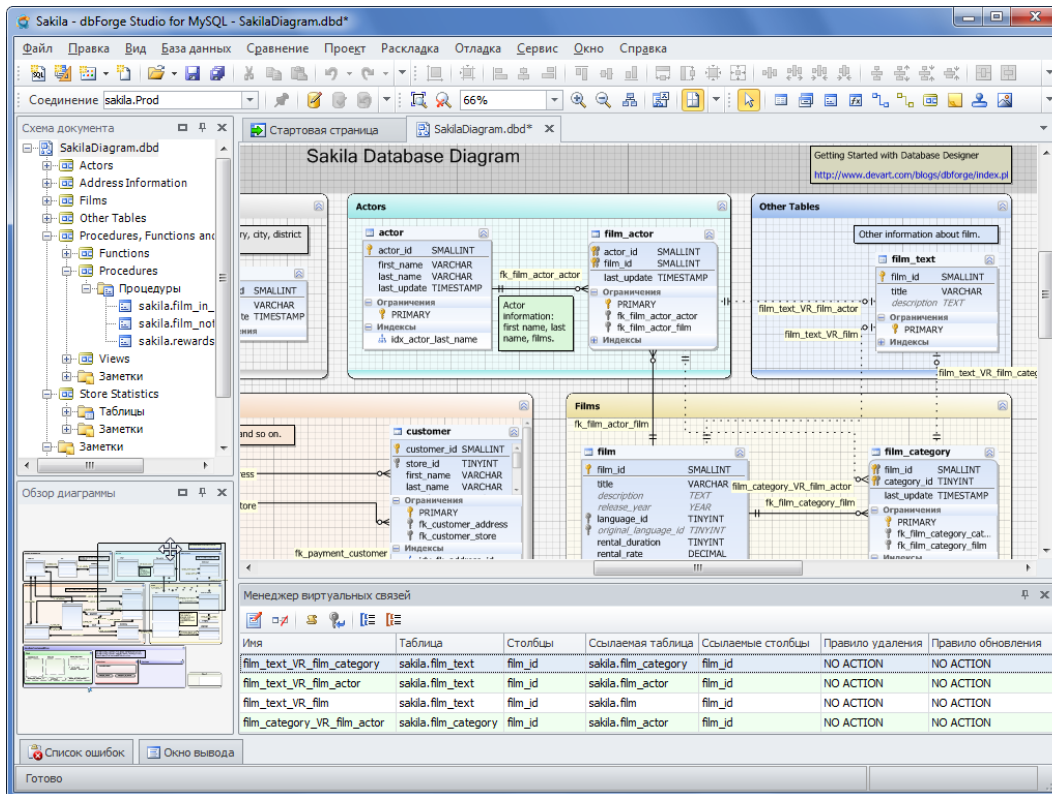


Рисунок 2.4 - Дизайнер баз данных

4. импорт/экспорт данных - добавляйте данные из внешних источников в базы MySQL с помощью наших инструментов для экспорта и импорта. Поддержка 10 популярных форматов и использование шаблонов позволяет автоматизировать весь процесс экспорта и импорта данных с помощью командной строки(Рисунок 2.5);

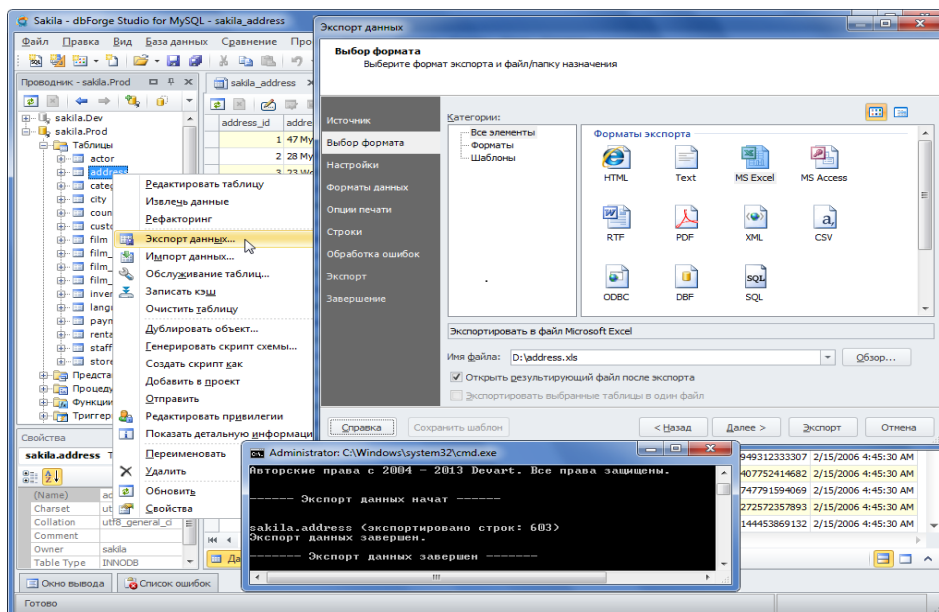


Рисунок 2.5 - Импорт/экспорт данных

5. администрирование БД - инструменты для администрирования и управления базами данных MySQL включают средства для:

- управления ролями и привилегиями пользователей;
- контроля сервисов MySQL;
- управления переменными сервера;
- обслуживания таблиц;
- управления сессиями (рисунок 2.6);

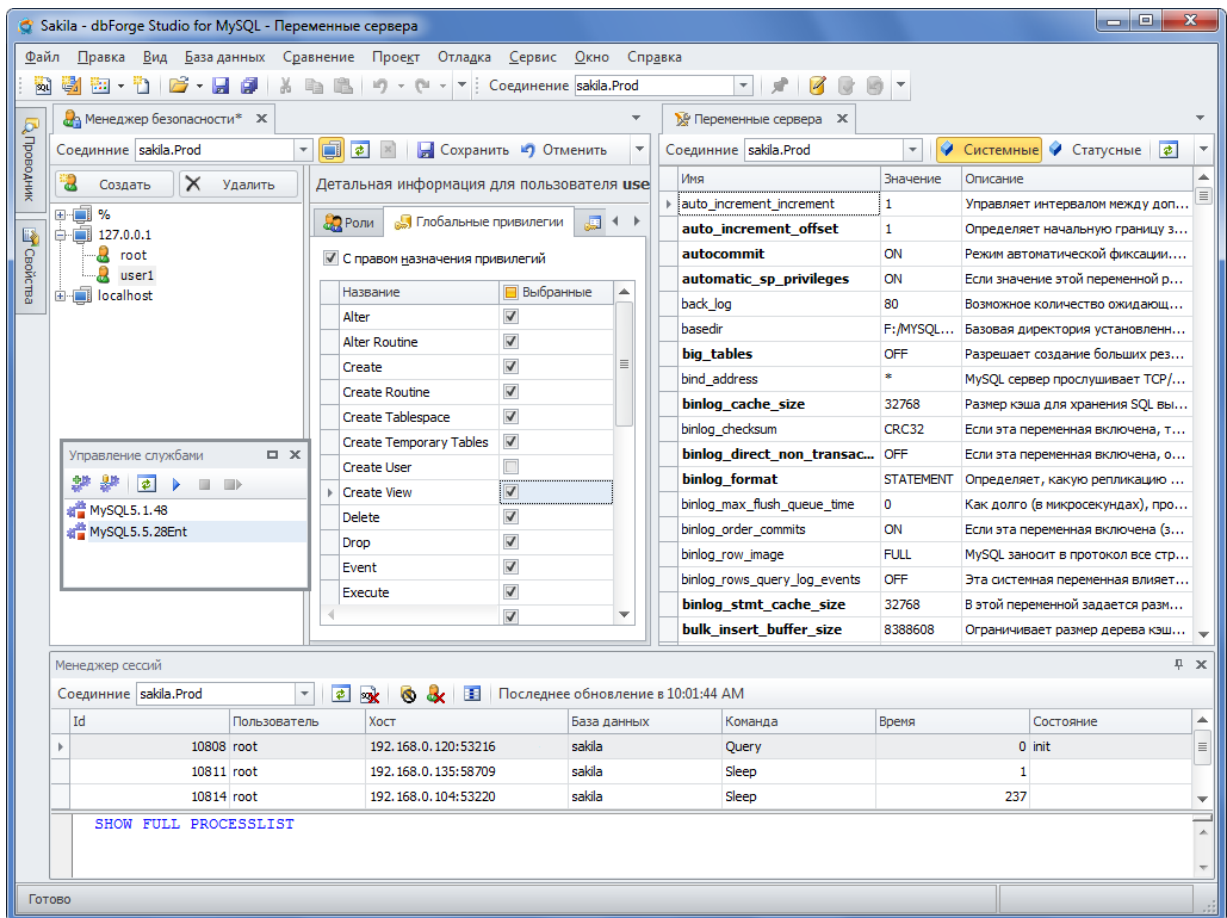


Рисунок 2.6 - Администрирование БД

2.5 Создание базы

Одним из вариантов создания базы данных осуществляется в программе Devart dbForge Studio for MySQL. Она является одним из программных средств администрирования СУБД MySQL. В этой программе предусмотрен целый ряд функциональных средств управления сервером, в которых применяется относительно простой в использовании графический интерфейс пользователя. Главным инструментом Devart

dbForge Studio for MySQL является обозреватель объектов, который позволяет пользователю просматривать, извлекать, и полностью управлять объектами сервера.

После запуска программы Devart dbForge Studio for MySQL открывается диалоговое окно установления соединения (рисунок 2.6).

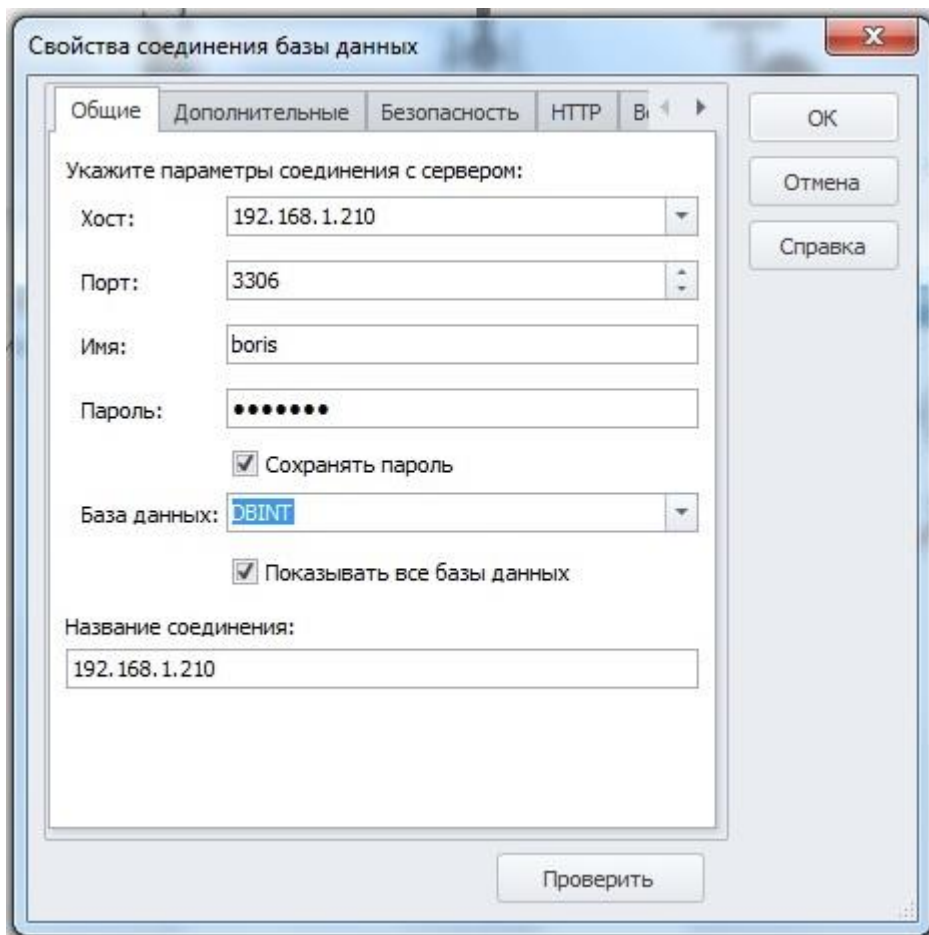


Рисунок 2.6 – Окно установления соединения

После установления соединения с сервером, новая база данных создается при помощи диалогового окна «Создать базу данных...» (рисунок 2.7). Здесь указывается имя новой базы, а также дополнительные настройки, связанные с кодировкой.

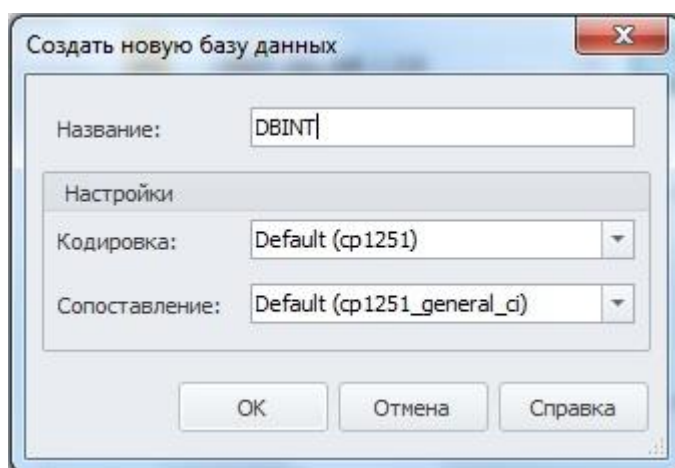


Рисунок 2.7 – Окно создания базы данных

Использование визуальной среды разработки упрощает работу с сервером баз данных. Для того, чтобы создать базу данных вручную необходимо выполнить следующий запрос: `CREATE DATABASE DBINT`

2.6 Создание таблиц

После создания базы данных необходимо создать таблицы, в которых будут храниться все используемые данные для работы программы. Так как принципы создания каждой из таблиц остаются неизменными, то для понимания того, каким образом создавались все таблицы, достаточно описания создания одной, например таблицы карточки объектов (Cards). Для её создания необходимо в обозревателе объектов в списке таблице выбрать пункт «Новая таблица» (рисунок 2.8) и в появившемся конструкторе таблиц (рисунок 2.9) начать создание столбцов таблицы, указывая имя столбца, тип, спецификацию идентификаторы и т.д. █

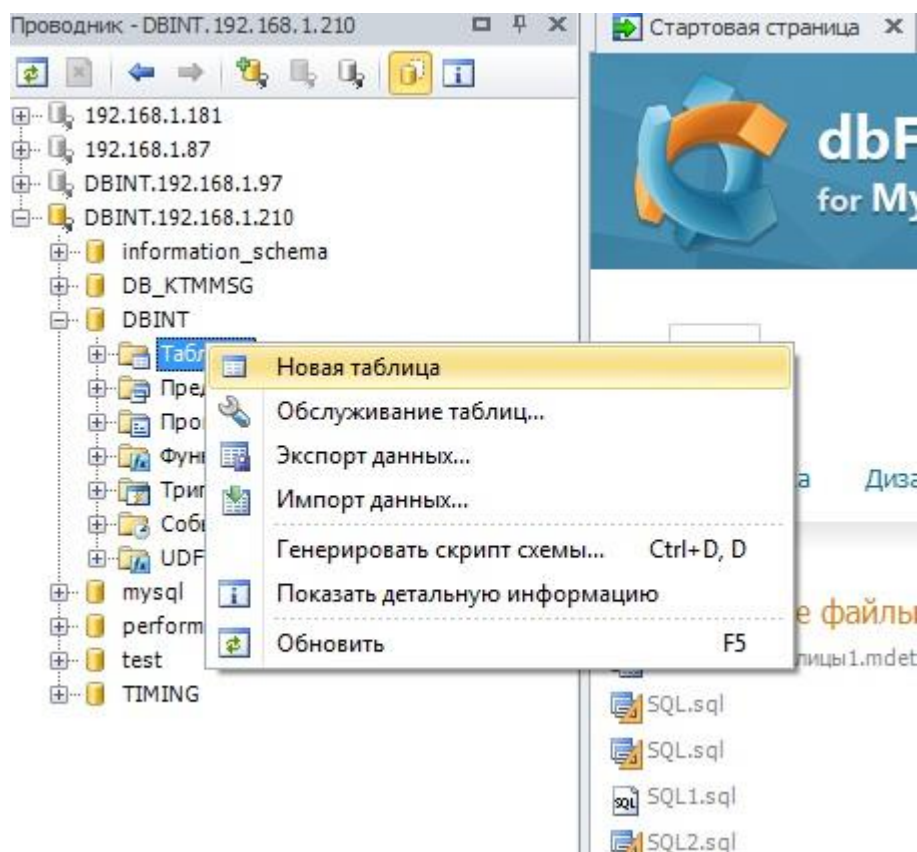


Рисунок 2.8 – Создание таблицы

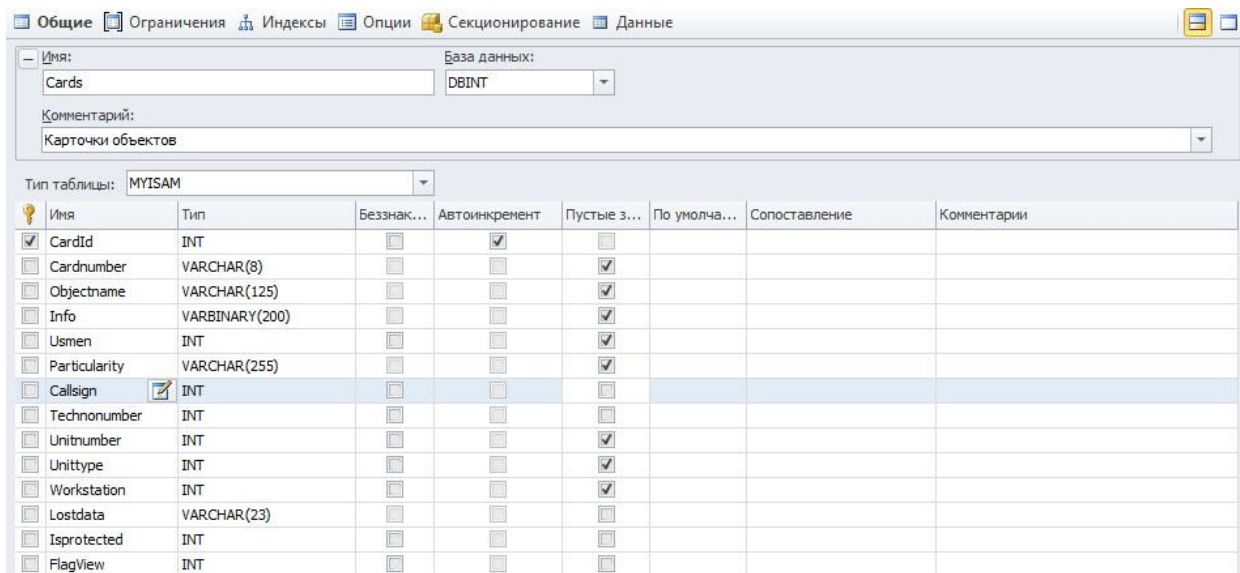


Рисунок 2.9– Конструктор таблиц

Создать таблицу можно и используя SQL запрос, но его

составление может занять гораздо большее время нежели в Devart dbForge Studio for MySQL. Вот пример составления той же таблицы :

```
CREATE TABLE DBINT.Cards (  
  CardId int NOT NULL AUTO_INCREMENT,  
  Cardnumber varchar(8) DEFAULT NULL,  
  Objectname varchar(125) DEFAULT NULL,  
  Info varbinary(200) DEFAULT NULL,  
  Usmen int DEFAULT NULL,  
  Particularity varchar(255) DEFAULT NULL,  
  Callsign int NOT NULL,  
  Technonumber int NOT NULL,  
  Unitnumber int DEFAULT NULL,  
  Unittype int DEFAULT NULL,  
  Workstation int DEFAULT NULL,  
  Lostdata varchar(23) NOT NULL,  
  Isprotected int NOT NULL,  
  FlagView int NOT NULL,  
  PRIMARY KEY (CardId)  
)  
ENGINE = MYISAM  
COMMENT = 'Карточки объектов'  
ROW_FORMAT = fixed;
```

В конструкторе таблиц также указывается столбец с первичным ключом. В данном случае это столбец с именем «CardId». Первичный ключ наиболее удобен для тех или иных практических целей, например для создания внешних ключей в других отношениях либо для создания кластерного индекса. Поэтому в качестве первичного ключа как правило выбирают тот, который имеет наименьший размер (физического хранения) и/или включает наименьшее количество атрибутов.

После создания всех необходимых таблиц базы данных на основе первичных ключей будут выставляться связи между таблицами , чтобы связать строки одних таблиц со строками других.

3 Разработка программного продукта

3.1 Создание интерфейса программы

Создание интерфейса программы начинается после создания проекта с «заготовкой» формы в среде Delphi 7. В первую очередь, перед созданием каких-либо элементов взаимодействия, необходимо задать различные свойства формы (рисунок 3.1.1), такие как: название, иконка, размеры, цвет фона окна и т.д.

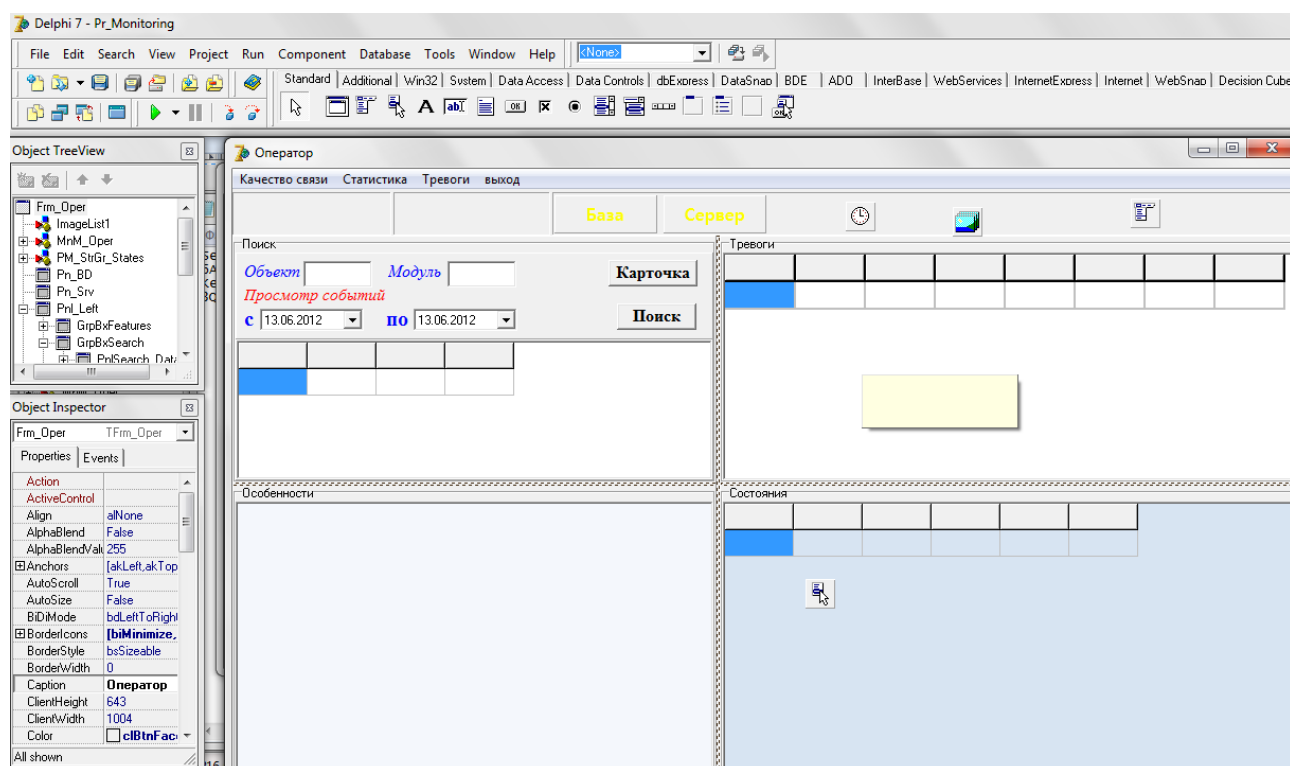


Рисунок 3.1 – Задание свойств формы

Решив, какие именно элементы меню должны присутствовать в программе, можно приступать к его формированию. Для начала в форму приложения необходимо добавить элемент «MainMenu», который располагается на вкладке панели инструментов. После его добавления при помощи визуального редактора создаются основные элементы меню (Рисунок 3.2) и выпадающие списки.

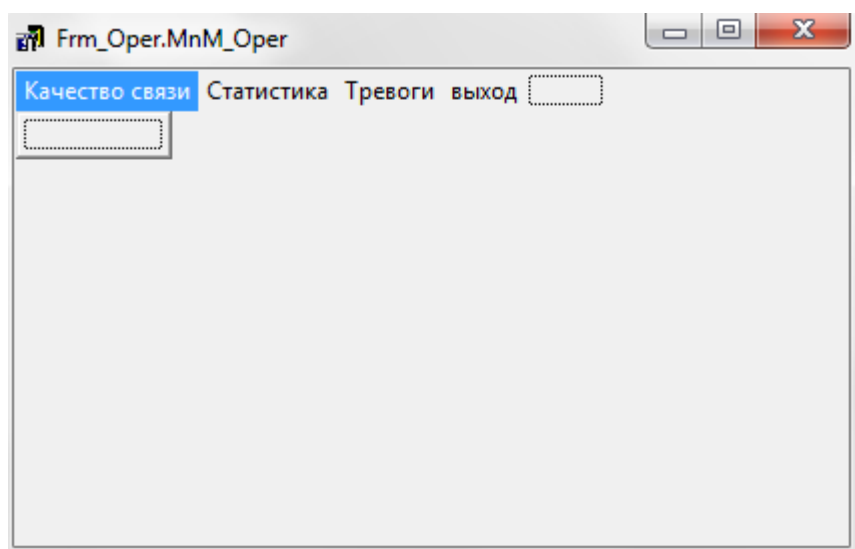


Рисунок 3.2 – Визуальный редактор меню

Дальнейшее проектирование интерфейса заключается в создании интерактивных элементов содержащих кнопки, текстовые поля, таблицы. Для достижения определенного расположения элементов в форме используют элемент «Panel», который позволяет группировать коллекции элементов управления.

Для выборки событий по дате в программе используется стандартный элемент управления «DateTimePicker», который позволяет пользователю выбрать дату и отобразить их в выбранном формате. А для отображения результатов поиска используется элемент «StringGrid», а все остальные элементы поиска заключены в элемент «GroupBox» (рисунок 3.3)

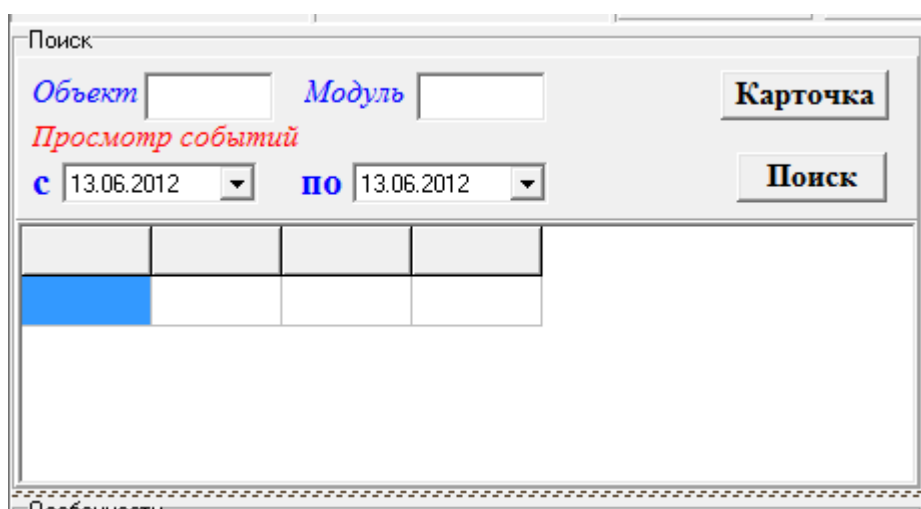


Рисунок 3.3 – Панель поиска событий объекта

Для отображения списка тревог, используется «StringGrid», который

так же заключен для удобства отображения в «GroupBox»(рисунок 3.4)



Рисунок 3.4 –Список тревог

Тревожные события по отключению электричества, разрядению аккумулятора по аналогии с таблицей тревог отображаются в «StringGrid», заключенный в «GroupBox» (рисунок 3.5).

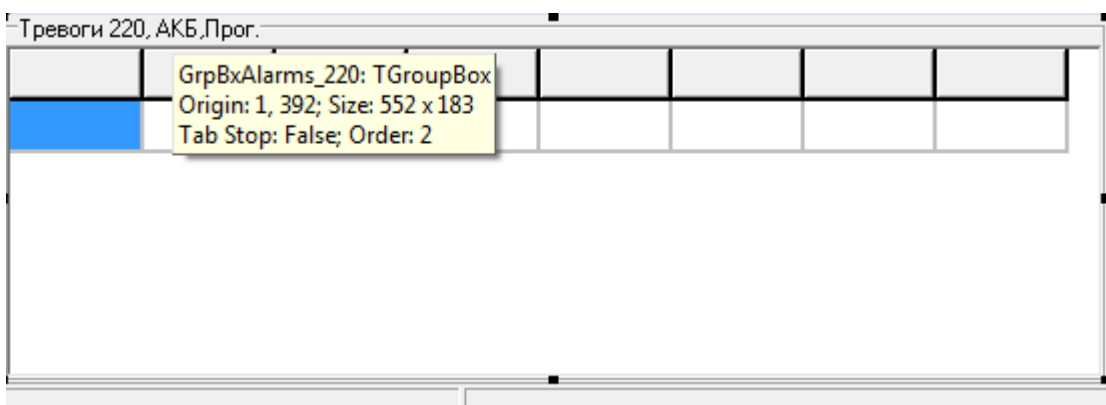


Рисунок 3.5 – Отображение списка тревог по отключению электричества

В еще одном «GroupBox» расположена таблица «StringGrid» в которую поступают все остальные события по объектам мониторинга (рисунок 3.6).

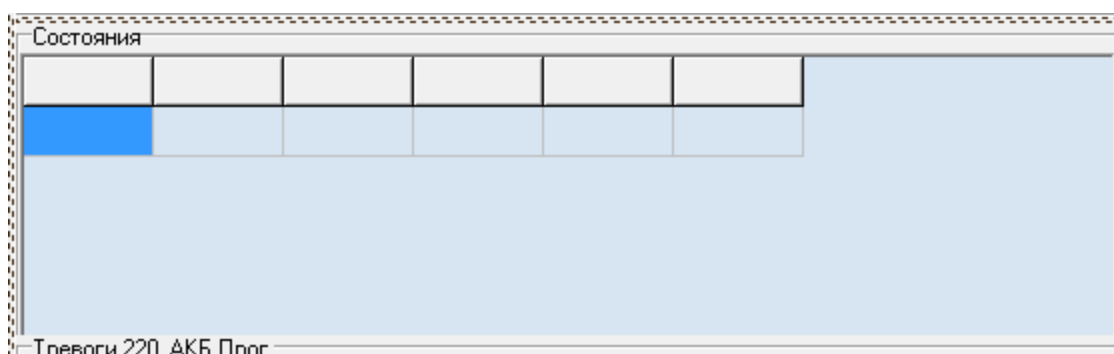


Рисунок 3.6 – Таблица отображение все событий

При выборе строки любой из таблиц, главные особенности объекта выводятся в специальное поле «Мето», которое в свою очередь заключено в «GroupBox» (рисунок 3.7)



Рисунок 3.7 – Вывод особенностей объекта

Для просмотра и редактирования полной информации по объекту необходимо нажать клавишу «Карточка» или двойным кликом мыши в таблице событий. После чего откроется новое окно с всей информацией по объекту(рисунок 3.8). Номер объекта, номер группы реагирования и номер договора выводятся в элемент «Edit», что позволяет при необходимости сразу внести изменения. Другие информационные данные : наименования объекта, адрес объекта, телефоны объекта, особенности объекта выводятся в элемент «Мето», что так же позволяет ,при необходимости, вносить изменения. Другие параметры объекта имеют фиксированные значения,

поэтому они выводятся в элементы «ComboBox». А все значения в «ComboBox» загружаются из соответствующих таблиц базы данных. Так же для блокирования тревожных событий от объектов используется «CheckBox»(3.9)

Рисунок 3.8 – Просмотр и редактирование карточки объекта

Блокирован

Рисунок 3.9 – Блокирование и разблокирование тревог объекта

Последняя информация о состоянии наличия электроэнергии на объекте и состоянии аккумулятора отображаются в элементе «Panel», которые средствами программы подкрашиваются в зеленый и красный цвет, при наличии или отсутствии электроэнергии на объекте соответственно.

Для завершения работы над карточкой объекта расположены 4 кнопки «Button»: «Сохранить»-для сохранения внесенных изменений в базу данных, «Добавить» - для создания новой карточки объекта, «Удалить» - для удаления текущей карточки объекта, «Отмена» - для закрытия карточки

объекта без внесения изменений в базу данных(рисунок 3.10).

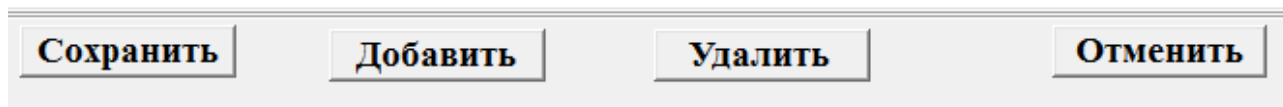


Рисунок 3.10 – Кнопки выбора действия

Для отображения и выбора соответствующего действия над тревогами, нужно нажать двойным кликом мыши по таблице тревог в главном окне, после чего откроется «Окно тревоги»(рисунок 3.3.11). Вся необходимая информация о тревоге отображается в стандартных элементах «Мемо» и «Label», но редактирования запрещено. На форме присутствуют три клавиши выбора действия над тревогой «Убрать с экрана» - для закрытия формы «Окно тревоги», «Подтвердить» - для подтверждения тревоги и «Отработать» - для отработки тревоги. Выбор причины отработки тревоги отображается в элементе «ListBox», значения которой загружаются из соответствующей таблицы базы данных.

Рисунок 3.11 – Форма «Окно тревоги»

Для просмотра и редактирования разделов объекта, необходима нажать клавишу «Разделы» в окне «Карточка объекта». После чего открывается «Окно Разделов»(рисунок 3.12). В верхней части формы расположены элементы «Мето», в которые в качестве напоминания заполняются информацией об объекте : наименования карточки, адрес, телефоны и номер договора соответственно. В низу формы расположены такие же как и в форме «Окно карточки» кнопки «Добавить»- для добавления нового раздела, « Удалить» - для удаления раздела, «Сохранить» - для сохранения внесенных изменений, «Отмена» - для закрытия формы разделов без сохранения внесенных изменений и «Зоны» - для открытия формы «Окно зон». Так как количество разделов может быть разным или вообще отсутствовать, поэтому элементы для настройки и заполнения информации по разделу создаются программно, т.е. так называемым динамическим методом в центре формы, обособленные в отдельный элемент «Panel». При нажатии клавиши «Зоны», открывается форма «Окно зон». Форма «Окно зон» создано полностью по аналогии с формой разделов и содержит такие же элементы.

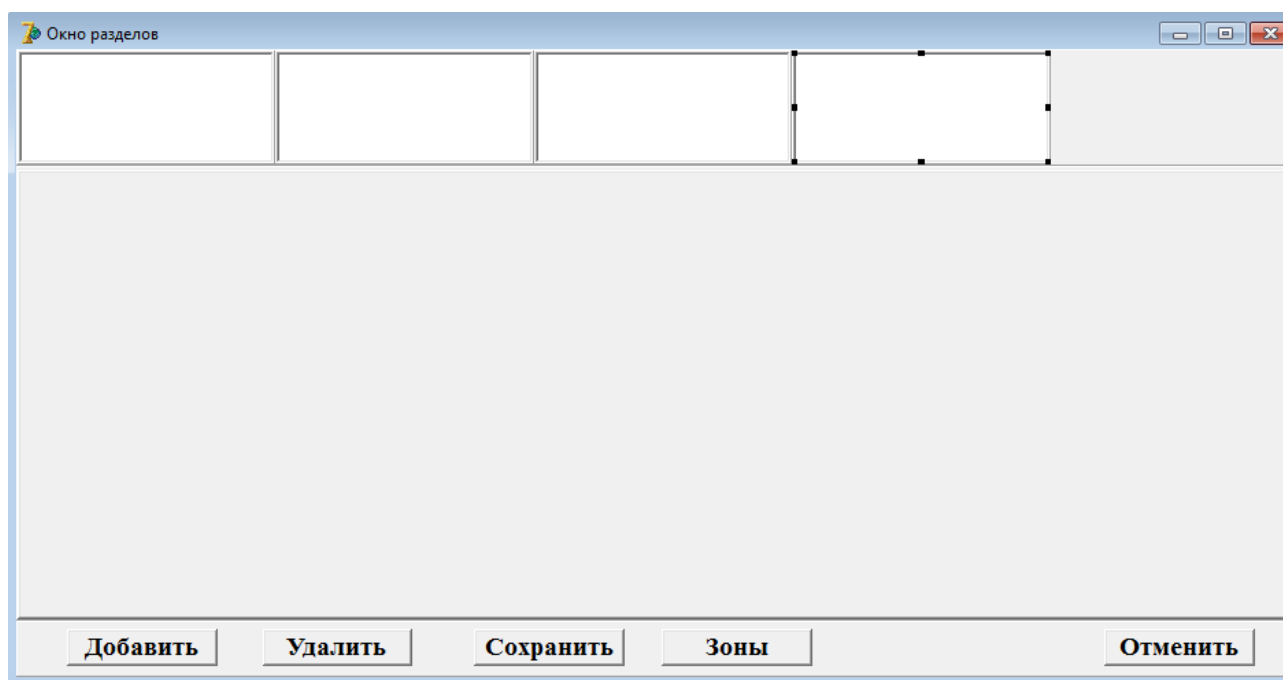


Рисунок 3.12 – Форма «Окно разделов»

Для просмотра, добавления и изменения схемы расположения объекта, необходимо в форме «Карточка объекта» нажать клавишу «Схема», после чего откроется форма «Окно схемы»(рисунок 3.13). Для добавления и изменения файла схемы, необходимо нажать клавишу «...»,

после чего откроется стандартное диалоговое окно, работа которого осуществляется с помощью элемента «OpenDialog». Файл схемы расположения объекта, формата Jpeg, отображается с помощью элемента «Image» в центре формы. Для сохранения пути файла необходимо нажать клавишу «Применить», а для отмены и закрытия формы нажать клавишу «Отменить».

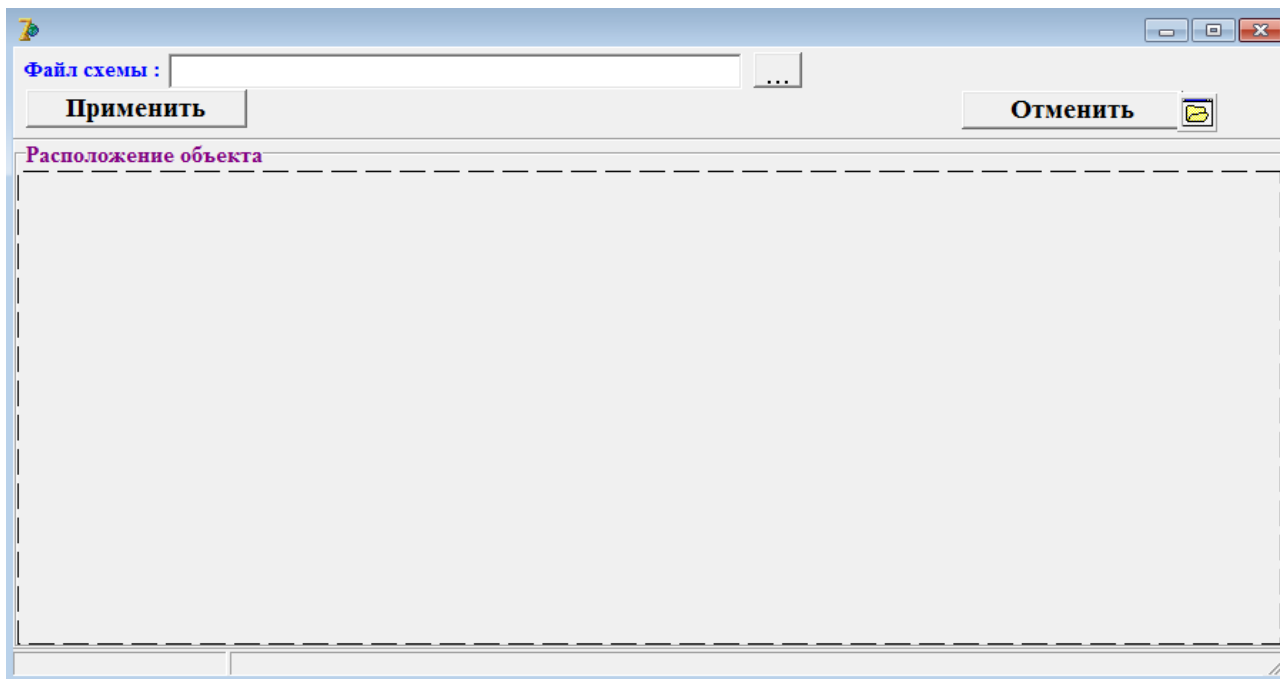


Рисунок 3.13 – Форма «Окно схемы»

3.2 Создание модуля для доступа к базе данных

Что бы получать данные от базы данных , был создан отдельная, не видимая при запуске программы, форма «DataModule». Для подключения к базе данных, используется элемент «ZCnctBD» класса «TZConnection», в котором указан адрес, названия базы данных, пользователь базы данных и пароль к ней. Для выполнения запроса и получения ответа используется элемент «ZQueryRd» класса TZQuery.

Для отслеживания ошибок в работе модуля, была создана процедура логирования ошибок – «L_Error_DM(Str_Log:string)», где «StrLog:string» - входной параметр, текст ошибки формата строки. Вызов процедуры осуществляется в отработке исключительных ситуаций. Процедура создает текстовый файл формата «txt», куда в новой строке записывается текст ошибки. Процедура работает следующим образом :

```
procedure TDataModule1.L_Error_DM(Str_Log:string);  
var  
    F: TextFile;  
    FileName, dt, str: String;
```

```

begin
  FileName := GetCurrentDir + '\Log_Error';
  dt:=DateToStr(Date); FileName := FileName + dt + '.txt';
  AssignFile(F, FileName);
  if FileExists(FileName) then Append(F)
  else Rewrite(F);
  dt:=dt+' '+TimeToStr(Time);
  str := dt + ': ' + Str_Log;
  WriteLn(F, str);
  CloseFile(F);
end;

```

Для выполнения запроса к базе данных без получения ответа, была создана процедура «ReadRcdBD», ее код представлен ниже :

```

procedure TDataModule1.ReadRcdBD(ReadSQL:String);
begin
try
  ZCnctBD.AutoCommit := False;
  ZCnctBD.Connected:=true;
  ZQueryRd.Close;
  ZQueryRd.SQL.Text := ReadSQL;
  ZQueryRd.Open;
  ZCnctBD.Commit;
  ZCnctBD.AutoCommit := True;
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.ReadRcdBD -' + E.Message+'---'+ReadSQL);
  end;
end;

```

Для выполнения и получения ответа от базы данных в необходимом формате были созданы процедуры «GetValIntFromField», «GetValFltFromField», «GetValStrFromField», которые выдают ответы в типе данных integer, float, string соответственно. На примете процедуры «GetValIntFromField», работа с базой данных осуществляется следующим образом :

```

procedure TDataModule1.GetValIntFromField(SqlString, FldName: String; Var
RztValInt: Integer);
begin
try
  ZCnctBD.AutoCommit := False;
  ZCnctBD.Connected:=true;
  ZQueryRd.Close;
  ZQueryRd.SQL.Text := SqlString;
  ZQueryRd.Open;
  ZCnctBD.Commit;
  ZCnctBD.AutoCommit := True;
  RztValInt := ZQueryRd.FieldValues[FldName];
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.GetValIntFromField -' + E.Message+'---'+SqlString);
  end;
end;

```

```
end;
```

Процедуры «GetValFltFromField», «GetValStrFromField», работают аналогичным образом.

При необходимости, для получения информации о наличии или отсутствии определенной записи в базе данных, была создана функция «FunGetRzltNullBD». Результат функция выдает в типе Boolean, т.е. true или false. Работает она следующим образом :

```
function TDataModule1.FunGetRzltNullBD(SqlString : String ) : Boolean;
begin
try
    ZCnctBD.Connected:=true;
    ReadRcdBD(SqlString);
        if (ZQueryRd.IsEmpty) then Result := False
        else Result := True;
except
    on E: Exception do begin
        ZCnctBD.Connected:=false;
        L_Error_DM('DM.FunGetRzltNullBD -' + E.Message+'---'+SqlString);
    end;
end;
end;
```

3.3 Создание потока отслеживания тревог

Для отслеживания тревог, был создан отдельный поток программы, так как отслеживание тревог в общем потоке программы может затруднить или замедлить ее работу. Для удобства чтения программы, в отдельном DataModule-«DM_Alrm_TH».

При создании формы «TDM_Alrm_TH», запускается процедура «DataModuleCreate(Sender: TObject)», в которой осуществляется запуск потока отслеживания тревог и задается приоритет потока :

```
procedure TDM_Alrm_TH.DataModuleCreate(Sender: TObject);
begin
    NewTH_Alrm:=TThread_alr.Create(false);
    NewTH_Alrm.Priority:=tpNormal;
end;
```

Выполнение потока осуществляет стандартная процедура «Execute» :

```
procedure TThread_alr.Execute;
begin
Sleep(2000);
while not Terminated do begin
try
    A_ConZ:=TZConnection.Create(nil);
    A_ConZ.AutoCommit:=true;
    A_ConZ.HostName:=Str_IP;
    A_ConZ.Database := Name_BD;
    A_ConZ.User := User_BD;
    A_ConZ.Password:=PaswordBD;
    A_ConZ.Port:=3306;
    A_ConZ.Protocol:='mysql-5';
```

```

A_ConZ.AutoCommit:=true;
A_QueryRdZ:=TZQuery.Create(nil);
A_QueryRdZ.Connection:=A_ConZ;
A_QueryRdZ.AutoCalcFields:=true;
A_QueryRdZ.ParamCheck:=true;
//***** заполнить массив тревог SetMasAlarme(SQL_ALARME,Mas_Alarme);
try
SetMasAlarme(SQL_ALARME,Mas_Alarme);
SetMasAlarme(SQL_ALARME_220,Mas_Alarme_220);
except
end;
//***** заполнить Табл тревог
Synchronize(Fill_Gr_Alarm);
Synchronize(Fill_Gr_Alarm_220);
finally
SetLength(Mas_Alarme,0); //***** в любом случае освободить массив
SetLength(Mas_Alarme_220,0); //***** в любом случае освободить массив
A_ConZ.Connected:=false;
A_ConZ.Free;
A_QueryRdZ.Free;
end;
Sleep(1000);
end;//while

```

Где **A_ConZ** и **A_QueryRdZ**, динамически созданные, элементы для соединения подключения и выполнения запросов к базе данных, работают по аналогии с элементами в «DataModule1». Процедура «SetMasAlarme» заполняет массив записей тревог :

```

procedure TThread_alr.SetMasAlarme(S_string : String; var MasAlrm : M_Prm);
var i:Integer;
begin
i:=0;
with A_QueryRdZ do begin //***** заполнить массив тревог
Close; SQL.Text := S_string;
Open; First;
while not Eof do begin
inc(i); SetLength(MasAlrm,i);
MasAlrm[i-1].Prm1:=IntToStr(FieldValues['AlarmID']);
MasAlrm[i-1].Prm3:=IntToStr(FieldValues['State']);
MasAlrm[i-1].Prm5:=Copy(FieldValues['ReceiveTime'],0,10);
MasAlrm[i-1].Prm6:=Copy(FieldValues['ReceiveTime'],12,8);
MasAlrm[i-1].Prm11:=IntToStr(FieldValues['RowID']);
MasAlrm[i-1].Prm7:=FieldValues['Cardnumber'];
MasAlrm[i-1].Prm8:=FieldValues['SectionName'];
MasAlrm[i-1].Prm9:=FieldValues['ZoneNumber'];
MasAlrm[i-1].Prm10:=FieldValues['Disce'];
Next;
end; //WHILE
end; //with DM
end;

```

Процедура «Synchronize» выполняет синхронизацию прорисовки таблицы тревог главного потока программы с потоком тревог. Процедура «Fill_Gr_Alarm» заполняет таблицу тревог в главном окне «Оператор» :

```

procedure TThread_alr.Fill_Gr_Alarm;
var i : integer;
begin

```

```

        for i:=0 to (Length(Mas_Alarme)-1) do begin
            Frm_Oper.StrGrdAlarms.Cells[0,i+1]:=Mas_Alarme[i].Prm5;
            Frm_Oper.StrGrdAlarms.Cells[1,i+1]:=Mas_Alarme[i].Prm6;
            Frm_Oper.StrGrdAlarms.Cells[2,i+1]:=Mas_Alarme[i].Prm7;
            Frm_Oper.StrGrdAlarms.Cells[3,i+1]:=Mas_Alarme[i].Prm10;
            Frm_Oper.StrGrdAlarms.Cells[4,i+1]:=Mas_Alarme[i].Prm8;
            Frm_Oper.StrGrdAlarms.Cells[5,i+1]:=Mas_Alarme[i].Prm9;

Frm_Oper.StrGrdAlarms.Cells[6,i+1]:=GetState(StrToInt(Mas_Alarme[i].Prm3));
            Frm_Oper.StrGrdAlarms.Cells[7,i+1]:=Mas_Alarme[i].Prm1;
        end;//for
        For i:=(Length(Mas_Alarme)+1) to Frm_Oper.StrGrdAlarms.RowCount-1
do begin
            Frm_Oper.StrGrdAlarms.Rows[i].Clear;
        end;
        Frm_Oper.StrGrdAlarms.RowCount:=Length(Mas_Alarme)+2;
    end;
end;

```

Функция «GetState» определяет статус тревоги по числовому состоянию тревоги :

```

Function TThread_alr.GetState ( ss :Integer) : String;
begin
    case ss of
        256..1000 : Result := 'отработанная';
        128..255   : Result := 'снятая с охр.';
        4..7       : Result := 'подтвержденная';
        0..3       : Result := 'не подтвержденная';
    end;
end;
end;

```

3.4 Обработка команд

После создания всех элементов интерфейса необходимо запрограммировать действия всех элементов. Для того, чтобы присвоить каждой кнопке действия необходимо в визуальном редакторе меню дважды щелкнуть по соответствующему пункту. Будет открыт редактор кода и курсор будет находиться внутри функции обработчика нажатия клавиши мыши. Например в форме «Окно оператора» для кнопки «Карточка», которая открывает форму «Окно карточки объекта» :

```

procedure TFrm_Oper.BtnSearchCardClick(Sender: TObject);
var str_sql: String;
begin
    str_sql:= selectIdCrd+aps+EdSearch_OB.Text+aps;
    DM.DataModule1.GetRztNotNullBD(str_sql,'CardId',Key_Regim);
    if Key_Regim=True then begin
        DM.DataModule1.GetValIntFromField(str_sql,'CardId',Key_IdCard);
        FrmCrd.GetFullData(Key_IdCard);
        FrmCrd.ShowModal;
    End else begin
        ShowMessage('Карточки с номером '+EdSearch_OB.Text+' не существует');
        EdSearch_OB.Clear;
        EdSearch_OB.SetFocus;
    end;
end;
end;

```


Для того что бы вывести в таблице списка событий, по нажатию кнопки «Поиск», необходимо в обработчик данной кнопки добавить следующий код:

```

procedure TFrm_Oper.BtnSearchClick(Sender: TObject);
var str_sql, Val_Mod:String;
NSrch : integer;
begin
  NSrch:=0;
  Stat.FrmStat.ClearStrGr(StrGrdSearch);
  if EdSearch_OB.Text<>' ' then begin
    str_sql:= ' SELECT NumberMod FROM Modul LEFT JOIN '+
              ' Cards ON Cards.Unitnumber=Modul.Idmod '+
              ' WHERE CardNumber ='''+EdSearch_OB.Text+'''+
              ' AND Cards.Unitnumber=Modul.Idmod ;';

  end
  else begin
    str_sql:='SELECT      NumberMod      FROM      Modul      WHERE      NumberMod=
'+EdSearch_Mod.Text;
  end;
  try
    try
      DM.DataModule1.GetValStrFromField(str_sql,'NumberMod',Val_Mod);
    Except
      ShowMessage('Введенного модуля и объекта не существует');
    end;
  finally
    str_sql:='';
  end;
  str_sql:='SELECT Dat,DiscE FROM KTMMMSG '+
          'LEFT JOIN Enames on Enames.CodeE=KTMMMSG.Msg '+
          ' WHERE Modul = '+Val_Mod+
          ' and ( Dat BETWEEN      '''+FormatDateTime('yyyy-MM-dd',
DtTmPckSearch_Bg.Date) +
          ' 00:00:00.000'+'''' and      '''+FormatDateTime('yyyy-MM-dd',
DtTmPckSearch_End.Date) +
          '23:59:59.999'+'''' ) ORDER BY Dat DESC';
  try
    with DM.DataModule1.ZQueryRd do begin
      Close;SQL.Text:=str_sql;
      Open; First;
      while not Eof do begin
        Inc(NSrch);
        StrGrdSearch.RowCount:=StrGrdSearch.RowCount+1;
        StrGrdSearch.Cells[0,NSrch]:=Copy(FieldValues['Dat'],1,10);
        StrGrdSearch.Cells[1,NSrch]:=Copy(FieldValues['Dat'],11,9);
        StrGrdSearch.Cells[2,NSrch]:=EdSearch_OB.Text;
        StrGrdSearch.Cells[3,NSrch]:=FieldValues['DiscE'];
        Next;
      end;
    end;
  except
  end;
end;
end;

```

Для того что бы по нажатию любой из таблиц формы «Оператора», в элементе «Мето» - особенности объекта, появилась информация, необходимо в обработчик событий «OnClick» всех таблиц, добавить выполнение процедуры «GetInfoCrd», которая, исходя из выбранной

таблицы, выполнить заполнение :

```
procedure TFrm_Oper.GetInfoCrd(StrGr: TStringGrid );
var str_sql, Memo_Info :String;
NmOb,Tm,Sob : integer;
begin
  EdSearch_OB.Clear;
  EdSearch_Mod.Clear;
  MemoFeatures.Clear;
  try
    if StrGr=StrGrdLosses then begin
      NmOb:=1; Tm:=2; Sob:=2;
    end;
    if ((StrGr=StrGrdStates) or (StrGr=StrGrdWork)) then begin
      NmOb:=3; Tm:=1; Sob:= 4;
    end;
    if ((StrGr=StrGrdAlarms) or (StrGr=StrGrdSearch) or
(StrGr=StrGrdAlarms_220)) then begin
      NmOb:=2; Tm:=1; Sob:=3;
    end;
    EdSearch_OB.Text:=StrGr.Cells[NmOb,StrGr.Row];
    Memo_Info:='№ ' +StrGr.Cells[NmOb,StrGr.Row] + '
+StrGr.Cells[Tm,StrGr.Row] +
' ' + StrGr.Cells[Sob,StrGr.Row] + #13#10 ;
    str_sql:= selectIdCrd+aps+StrGr.Cells[NmOb,StrGr.Row]+aps;
    DM.DataModule1.GetValIntFromField(str_sql,'CardId',Key_IdCard);
    with DM.DataModule1.ZQueryRd do begin
      Close; SQL.Text := FullinfoCrd+IntToStr(Key_IdCard); Open;
      Memo_Info:=Memo_Info+FieldValues['Objectname']+'
'+FieldValues['Phones']+#13#10+
      FieldValues['Info']+#13#10+FieldValues['NameGrp']+'-
'+FieldValues['Callnumber'];
      if FieldValues['NameSctr']<>Null Then Memo_Info:=Memo_Info + '
'+FieldValues['NameSctr'];
    end;
    MemoFeatures.Lines.Add(Memo_Info);
  except
    MemoFeatures.Clear;
  end;
end;
```

Для того , что бы по двойному клику всех таблиц, кроме таблиц тревог, открывалась карточка объекта, необходимо в обработчике таблицы «OnDbClick» добавить выполнение процедуры «OpenCrd» :

```
procedure TFrm_Oper.OpenCrd(Numbr : string);
var str_sql: string;
begin
  str_sql:=selectIdCrd+aps+Numbr+aps;
  DM.DataModule1.GetValIntFromField(str_sql,'CardId',Key_IdCard);
  Key_Regim:=True;
  FrmCrd.GetFullData(Key_IdCard);
  FrmCrd.ShowModal;
end;
```

Для того, что бы строки в таблице поиска событий объекта выделялись в два цвета, в обработчике таблицы «OnDrawCell» необходимо добавить, следующий код :

```

procedure TFrm_Oper.StrGrdSearchDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
var
  Buf: array[byte] of char;
begin
  with TStringGrid(Sender) do begin
    if ARow>0 then begin
      Canvas.Font := Font;
      if (ARow mod 2) = 0 then
        Canvas.Brush.Color := clMoneyGreen
      else Canvas.Brush.Color := clAqua;
      Canvas.Font.Color := clWindowText;
      Canvas.FillRect(Rect);
      StrPCopy(Buf, Cells[ACol, ARow]);
      DrawText(Canvas.Handle, Buf, -1, Rect,
        DT_SINGLELINE or DT_VCENTER or DT_NOCLIP or DT_LEFT);
    end;
  end;
end;

```

Для того, что бы по двойному клику таблиц тревог, открывалась форма тревоги, необходимо в обработчик событий «OnDbClick» таблиц добавить следующий код :

```

procedure TFrm_Oper.StrGrdAlarmsDbClick(Sender: TObject);
var
  Alr :TReactionForm;
begin
  try
    if
      TStringGrid(Sender).Cells[TStringGrid(Sender).Col,TStringGrid(Sender).Row]<
      >' then begin
        Alr:=TReactionForm.Create(Application);

Alr.Tag:=StrToInt(TStringGrid(Sender).Cells[7,TStringGrid(Sender).Row]);
        Alr.ShowModal;
      end;
    except
      MessageBox(Frm_Oper.Handle, PChar('Тревога не найдена.'),
        'Ошибка поиска тревоги',
        MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    end;
  end;
end;

```

Для того , что бы строки таблиц тревог, подкрашивались в зависимости от состояния тревоги, в обработчик событий «OnDrawCell» необходимо добавить следующее :

```

procedure TFrm_Oper.StrGrdAlarmsDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
var
  Buf: array[byte] of char;
begin
  with TStringGrid(Sender) do begin
    if ARow>0 then begin
      Canvas.Font := Font;

```

```

        Canvas.Font.Color := clWindowText;
        Canvas.Brush.Color := clWindow;
        if Cells[6,ARow]='не подтвержденная' then Canvas.Brush.Color :=
clRed;
        if Cells[6,ARow]='подтвержденная' then Canvas.Brush.Color :=
RGB(250,128,114);
        if Cells[6,ARow]='снятая с охп.' then Canvas.Brush.Color :=
RGB(0,255,0);
        Canvas.FillRect(Rect);
        StrPCopy(Buf, Cells[ACol, ARow]);
        DrawText(Canvas.Handle, Buf, -1, Rect,
DT_SINGLELINE or DT_VCENTER or DT_NOCLIP or DT_LEFT);
    end; // if
end; //with
end;

```

Перед открытием карточки объекта, выполняется процедура «GetFullData», которая выполняет заполнение всех элементов формы «Карточка объекта» :

```

procedure TFrmCrd.GetFullData(IdCard: Integer);
var str_sql: String;
    Rzt: Integer;
begin
    str_sql := 'SELECT Description FROM ORG';
    with DM.DataModule1.ZQueryRd do begin
        Close; SQL.Text := str_sql; Open; First;
        CmBx_OrgObCrd.Clear;
        while not Eof do begin
            Self.CmBx_OrgObCrd.Items.Add(FieldValues['Description']);
            Application.ProcessMessages;
            Next;
        end;
    str_sql := 'SELECT NameGrp FROM GrpUp';
    with DM.DataModule1.ZQueryRd do begin
        Close; SQL.Text := str_sql; Open; First;
        CmBx_NmGrpCrd.Clear;
        while not Eof do begin
            Self.CmBx_NmGrpCrd.Items.Add(FieldValues['NameGrp']);
            Next;
        end;
    str_sql := 'SELECT NameSctr FROM Sector ';
    Close; SQL.Text := str_sql; Open; First;
    CmBx_NmThnCrd.Clear;
    while not Eof do begin
        Self.CmBx_NmThnCrd.Items.Add(FieldValues['NameSctr']);
        Next;
    end;
    str_sql := 'SELECT NameTpP FROM TypeP';
    Close; SQL.Text := str_sql; Open; First;
    CmBx_TpModCrd.Clear;
    while not Eof do begin
        Self.CmBx_TpModCrd.Items.Add(FieldValues['NameTpP']);
        Next;
    end;
    str_sql := 'SELECT * FROM Cards ' +
        'LEFT JOIN GrpUp ON Cards.Callsign = GrpUp.IdGrp ' +
        'LEFT JOIN Sector ON Cards.Technonumber = Sector.IdSctr '
+
        'LEFT JOIN Modul ON Cards.Unitnumber = Modul.Idmod ' +

```

```

        'LEFT JOIN TypeP ON Cards.Unitttype = TypeP.IdTpP ' +
        'LEFT JOIN ORG ON Cards.Usmen = ORG.IdOrg ' +
        'WHERE ' + {Cards.Callsign = GrpUp.IdGrp AND ' +
        'Cards.Technonumber = Sector.IdSctr AND ' +
        'Cards.Unitnumber = Modul.Idmod AND ' +
        'Cards.Unitttype = TypeP.IdTpP AND ' +
        'Cards.Usmen = ORG.IdOrg AND ' + }
        'Cards.CardId = ' + IntToStr(IdCard);
Close; SQL.Text := str_sql; Open; First;
Self.Ed_NbrOBCrd.Text :=FieldValues['Cardnumber'];
Self.MemNmCrd.Text := FieldValues['Objectname'];
Self.MemAdrsCrd.Text := FieldValues['Info'];
Self.MemTlphCrd.Text := FieldValues['Phones'];
Self.Ed_NumDgCrd.Text := FieldValues['Agreement'];
Self.MemDiskCrd.Text := FieldValues['Particularity'];
Self.Ed_NumGrpCrd.Text := FieldValues['Callnumber'];
Self.CmBx_NmGrpCrd.Text := FieldValues['NameGrp'];
Self.CmBx_NmThnCrd.Text := FieldValues['NameSctr'];
if FieldValues['NumberMod']<>null then Self.CmBx_NbrModCrd.Text
:= FieldValues['NumberMod'];
Self.CmBx_TpModCrd.Text := FieldValues['NameTpP'];
Self.CmBx_OrgObCrd.Text := FieldValues['Description'];
Rzt := FieldValues['Isprotected'];
if Rzt = 0 then Self.ChkBxWorkCrd.Checked := False
else Self.ChkBxWorkCrd.Checked := True;
Rzt := FieldValues['FlagView'];
if Rzt = 0 then Self.ChkBxBlok.Checked := True
else Self.ChkBxBlok.Checked := False;;
end;
end;
end;

```

При нажатии кнопки «Сохранить» выполняется следующий код:

```

procedure TFrmCrd.BtnSaveCrdClick(Sender: TObject);
var str_SQL: String;
begin
GetFullParam;
if Key_Regim = False then begin
str_SQL := 'INSERT INTO `Cards` (`Cardnumber`, `Objectname`, `Info`,
' +
`ShemId`, ' +
`Phones`, `Usmen`, `Agreement`, `Particularity`,
' +
`Callsign`, `Callnumber`, `Technonumber`, `Unitnumber`,
' +
`Unitttype`, `WorkStation`, `Lostdata`, `Isprotected`,
`FlagView`) ' +
'VALUES (''' + Str_Param.Prm1 + ''', ''' + Str_Param.Prm2
+
''', ''' + Str_Param.Prm3 + ''', ''' + Str_Param.Prm4 +
''', ' +
Str_Param.Prm5 + ', ''' + Str_Param.Prm6 + ''', ''' +
Str_Param.Prm7 +
''', ' + Str_Param.Prm8 + ', ' + Str_Param.Prm9 + ', '''
+
Str_Param.Prm10 + ''', ' + Str_Param.Prm11 + ', ' +
Str_Param.Prm12 +
', ' + Str_Param.Prm13 + ', ' + Str_Param.Prm14 + ', '''
+
Str_Param.Prm15 + ''', ' + Str_Param.Prm16 + ', ' +
Str_Param.Prm17 + ');';

```

```

end
else begin
    str_SQL := 'UPDATE `Cards` SET `Cardnumber` = ''' + Str_Param.Prm1 +
    ''', `Objectname` = ''' + Str_Param.Prm2 + ''', `Info` =
    ''' +
        Str_Param.Prm3 + ''', `Phones` = ''' + Str_Param.Prm4 +
        ''', `Agreement` = ''' + Str_Param.Prm6 + ''',
    `Particularity` = ''' +
        Str_Param.Prm7 + ''', `ShemId` = ' + Str_Param.Prm8 +
        ', `Callsign` = ' + Str_Param.Prm9 + ', `Callnumber` =
    ''' +
        Str_Param.Prm10 + ''', `Technonumber` = ' +
    Str_Param.Prm11 +
        ', `Unitnumber` = ' + Str_Param.Prm12 + ', `Unittype` = '
    +
        Str_Param.Prm13 + ', `WorkStation` = ' + Str_Param.Prm14
    +
        ', `Lostdata` = ''' + Str_Param.Prm15 + ''',
    `Isprotected` = ' +
        Str_Param.Prm16 + ', `Usmen` = ' + Str_Param.Prm5 + ',
    `FlagView` = ' +
        Str_Param.Prm17 + ' WHERE `CardId` = ' +
    IntToStr(Key_IdCard) + ';';
    end;
    AddBtnEnabled(True);

    DM.DataModule1.ChangeRcdBD(str_SQL);
    str_SQL := selectIdCrd+aps+ Str_Param.Prm1 +aps;
    DM.DataModule1.GetValIntFromField(str_SQL,'CardId',Key_IdCard);
    if Key_Regim = false then begin
        AddAutoSect();
    end;
    StatusBarCrd.Panels[2].Text := 'Данные сохранены !!!';

end;

```

В обработчик кнопки «Добавить» необходимо добавить следующий код:

```

procedure TFrmCrd.BtnADDCrdClick(Sender: TObject);
begin
    Key_Regim:= false;
    Ed_NbrOBCrd.Enabled:=True;
    Ed_NbrOBCrd.SetFocus;
    FormActivate(Self);
end;

```

Для того, что бы удалить карточку объекта, в обработчик событий при нажатии кнопки «Удалить», добавим следующее:

```

procedure TFrmCrd.BtnDLTCrdClick(Sender: TObject);
var
    result: TModalResult;
    str_SQL: String;
    idcrd : Integer;
begin
    result := MessageDlg('Удалить карточку с номером '+Ed_NbrOBCrd.Text,mtWarning, mbOKCancel, 0);
    if result = mrOK then begin

```

```

try
  str_SQL := selectIdCrd+aps+ Ed_NbrOBCrd.Text +aps;
  DM.DataModule1.GetValIntFromField(str_SQL,'CardId',idcrd);
  str_SQL:='INSERT INTO Cards_LOG (IdCrd, Crdnum_Old, Crdnum_New,
'+
  ' Workstation, Action, DT) VALUES ('+IntToStr(idcrd)+',''' +
  Ed_NbrOBCrd.Text      +      ''','''+Ed_NbrOBCrd.Text      +
  ''','+IntToStr(OperId)+',''+
  ''D'''+', CURRENT_TIMESTAMP());';
  DM.DataModule1.ChangeRcdBD(str_SQL);
  str_SQL := 'DELETE FROM `Cards` WHERE `CardId`='+IntToStr(idcrd);
  DM.DataModule1.ChangeRcdBD(str_SQL);
  ShowMessage('Удалено');
except
  ShowMessage('Ошибка');
end;
end;
if result = mrCancel then ShowMessage('отмена');
end;

```

Для закрытия формы или отмены, в обработчик нажатия кнопки «Отмена», добавим следующий код:

```

procedure TFrmCrd.BtnCnclCrdClick(Sender: TObject);
begin
  Close;
end;

```

Для того, что бы открыть форму «Разделов объекта», необходимо в обработчик кнопки «Разделы» добавить следующий код:

```

procedure TFrmCrd.BtnSectnCrdClick(Sender: TObject);
begin
  Frm_Section.GetFull_SectInfo;
  Frm_Section.ShowModal;
end;

```

Где процедура GetFull_SectInfo выполняет заполнение элементов формы:

```

procedure TFrm_Section.GetFull_SectInfo();
var Page: TTabSheet;
    s_sql: String;
begin
  Key_Card :=Key_IdCard; NbrSct := 0; Flag_New := False;
  ListOut := TList.Create;
  Mem_NmCardSection.Text:=Unit_Card.FrmCrd.MemNmCrd.Text;
  Mem_AdrCardSection.Text := Unit_Card.FrmCrd.MemAdrsCrd.Text;
  Mem_TlphCardSection.Text :=Unit_Card.FrmCrd.MemTlphCrd.Text;
  Mem_DogCardSection.Text      :='Договор      №      '+
Unit_Card.FrmCrd.Ed_NumDgCrd.Text;
  s_sql :=FullinfoSct + IntToStr(Key_Card)+' ORDER BY SectionName';
  DM.DataModule1.GetRztNotNullBD(s_sql, 'IdSection', Flag_New);
  if Flag_New = True then begin
    with DM.DataModule1.ZQueryRd do begin
      Close; SQL.Text := s_sql;
      Open; First;
      while not Eof do begin

```

```

        inc (NbrSct);
        SetLength (Mas_Section, NbrSct);
SetLength (Reg_Delete, NbrSct);
    Reg_Delete [NbrSct-1] := False;
    Mas_Section [NbrSct-1].Prm1 := IntToStr (FieldValues ['IdCard']);
    Mas_Section [NbrSct-1].Prm2 := FieldValues ['SectionName'];
    Mas_Section [NbrSct-1].Prm3 := FieldValues ['InfoSct'];
    Mas_Section [NbrSct-1].Prm4 := FieldValues ['SInfo'];
    Mas_Section [NbrSct-1].Prm5 := IntToStr (FieldValues ['State']);
    Mas_Section [NbrSct-1].Prm6 := FieldValues ['StateDate'];
    Mas_Section [NbrSct-1].Prm7 := FieldValues ['NPassword'];
    Mas_Section [NbrSct-1].Prm8 := FieldValues ['SPassword'];
    Mas_Section [NbrSct-1].Prm9 := IntToStr (FieldValues ['Flags']);
    Mas_Section [NbrSct-1].Prm10 :=
IntToStr (FieldValues ['IdSection']);
    Page := TTabSheet.Create (PgCntrlSection);
    with Page do begin
        PageControl := PgCntrlSection;
        Caption := 'Раздел №' + IntToStr (NbrSct);
        Name := 'TabSh_Sctr' + IntToStr (NbrSct);
    end;
    GetDataSection (Page, NbrSct, Flag_New, Mas_Section); Next;
end;
end;
end;
end;

```

Функцию создания динамических элементов формы, выполняет процедура `GetDataSection`, полный код этой процедуры представлен в приложении А.

Нажатие кнопок «Добавить», «Удалить», «Сохранить», «Отмена» в формах «Окно разделов» и «Окно зон», работает подобным образом, как и в форме «Окно карточки». Создание динамических объектов в форме «Окно зон», работает по аналогии с динамическим созданием объектов в форме «Окно разделов».

После активации формы «Окно тревоги», необходимо в обработчик события «OnActivete» добавить следующий код :

```

procedure TReactionForm.FormActivate (Sender: TObject);
var str_sql, FN: String;
    C_yes : Boolean;
begin
    Al_ID := IntToStr (Self.Tag);
    Fill_Data;
    str_sql := ' SELECT ConfirmTime FROM ALARME WHERE AlarmID = ' +
        Al_ID + ' LIMIT 1';
    FN := 'ConfirmTime';
    DM.DataModule1.GetRztNotNullBD (str_sql, FN, C_yes);
    if C_yes = false then begin
        btnProcess.Enabled := false;
        lbReason.Enabled := False;
    end
    else btnConfirm.Enabled := false;
end;
end;

```

В этой процедуре, с помощью вызова процедуры `Fill_Data`,

осуществляется заполнение элементов формы необходимой информацией по тревоге:

```
procedure TReactionForm.Fill_Data();
var str_sql,ssection :string;
begin
  with DM.DataModule1.ZQueryRd do begin
    Close; SQL.Text :=infoCard+Al_ID;
    Open; First;
    Self.lbObject.Caption:= FieldValues['Name']+' - '+
    Trim(FieldValues['Cardnumber'])+Trim(FieldValues['Objectname']);
    Self.lbSec.Caption:=' Раздел №'+FieldValues['SectionName'];
    Self.lbZoneID.Caption:='Номер зоны: '+FieldValues['ZoneNumber'];
    Self.lbPassword.Caption:='Контрольное слово:
'+FieldValues['NPassword']+
    ' Числовой пароль: ' +FieldValues['SPassword'];
    mInfo.Lines.Text:=Trim((FieldValues['Particularity']));
    mInfo.Lines.Add(Trim(FieldValues['InfoSct']));
    mInfo.Lines.Add(Trim(FieldValues['SInfo']));
    Self.mAddress.Text:=Trim(FieldValues['Info']);
    if FieldValues['NameSctr']<>Null then
Self.lbTechInfo.Caption:=Trim(FieldValues['NameSctr']);
    self.lbState.Caption := 'Группа';
    Self.stCallSign.Caption:= FieldValues['NameGrp']+'-
'+FieldValues['Callnumber'];
    self.lbTime.Caption := 'Время и дата тревоги: ' +
Trim(FieldValues['ReceiveTime']);
    self.mZoneInfo.Text := Trim(FieldValues['ZInfo']);
    case StrToInt(FieldValues['State']) of
      5:
        ssection := 'Снят ' + FieldValues['StateDate'];
      4:
        ssection := 'Поставлен ' + FieldValues['StateDate'];
    else
        ssection := 'Состояние раздела неопределено!';
    end; //end case
    Self.stState.Caption:=ssection;
  end;
end;
```

По нажатии кнопки «Подтвердить» , выполняется следующий код :

```
procedure TReactionForm.btnConfirmClick(Sender: TObject);
var str_sql : string;
begin
  try
    str_sql:= ' UPDATE ALARME SET ConfirmTime = CURRENT_TIMESTAMP(), ' +
    ' UserID = ' + IntToStr(OperId) +
    ' , State = State + 4 ' +
    ' WHERE AlarmID = ' + Al_ID;
    DM.DataModule1.ChangeRcdBD(str_sql);
    btnConfirm.Enabled:=false;
    lbReason.Enabled:=true;
  except
    str_sql := 'Тревога не найдена. Откройте окно подтверждения тревоги
еще раз!';
    MessageBox(Frm_Oper.Handle, PChar(str_sql),
    'Ошибка поиска тревоги',
    MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
  end;
```

```

        Close;
    end;
end;

```

Для того, что бы Отработать тревогу, необходимо нажать кнопку «Отработать», и она содержит следующий код:

```

procedure TReactionForm.btnProcessClick(Sender: TObject);
var str_sq : string;
    idReas : integer;
begin
    if (lbReason.ItemIndex < -1) then begin
        MessageBox(Frm_Oper.Handle, PChar('Выберите вид тревоги'),
            'Ошибка поиска тревоги',
            MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    end
    else begin
        try
            str_sq:= ' SELECT IDReas FROM REASONS WHERE Name = '''+
                lbReason.Items[lbReason.ItemIndex] +''';
            DM.DataModule1.GetValIntFromField(str_sq,'IDReas',idReas);
        except
            MessageBox(Frm_Oper.Handle, PChar('Ошибка при выборе тревоги'),
                'Ошибка ',
                MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
        end;
        try
            str_sq:=' UPDATE ALARME SET ProcessTime = CURRENT_TIMESTAMP(), ' +
                ' Reason = ' + IntToStr(idReas) + ', '+
                ' UserID = ' + IntToStr(OperId) +
                ' , State = State + 256 ' +
                ' WHERE AlarmID = ' + Al_ID;
            DM.DataModule1.ChangeRcdBD(str_sq);
        except
            MessageBox(Frm_Oper.Handle, PChar('Тревога не найдена. Откройте
            окно подтверждения тревоги еще раз'),
                'Ошибка поиска тревоги',
                MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
            Close;
        end;
    end;
    close;
end;

```

Для закрытия формы «Окно тревоги», в обработчик нажатия кнопки «Убрать с экрана», был добавлен код:

```

procedure TReactionForm.btnHideClick(Sender: TObject);
begin
    Close;
end;

```

Полный код программы отслеживания тревожных событий и поиска объектов представлен в приложении А.

3.5 Разработка программы обработки пакетов сигнала

Для расшифровки и записи в базу данных событий от радио-передающих модулей, была разработана программа сервер. В основные задачи этой программы входят:

- подключение к базе данных MySQL;
- считывание информации с порта RS-232;
- формирование пакета данных;
- анализ полученного пакета данных;
- запись пакета в базу данных;
- зормирование тревожных событий.

Как оговаривалось ранее, для разработки приложения была выбрана среда разработки Qdeveloper. Сервер логически разделен на 3 класса:

- Раздел для работы с базой данных(databd.cpp).
- Раздел для формирования пакета(thread.cpp)
- Раздел для анализа пакета(echoserver.cpp)

Во всех разделах создано логирование основных действий, а так же записи ошибок. Ниже представлен код логирования на пример класса «databd.cpp»:

```
void DataBD::SaveToLogBD(QString StrLog)
{
    QTime tm;File flSave; QTextStream stream( &flSave );
    QStringList lines;    QString Fname, line, stm;
    Fname = QDir::currentPath() + "/WorkBD.log";
    tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz");
    line = QDate::currentDate().toString("dd/MM/yy");
    line += ": "; line += stm + " ";
    line += StrLog;
    flSave.setFileName(Fname);    lines << line;
    try{
        if(flSave.exists()) flSave.open(QIODevice::Append);
        else flSave.open(QIODevice::ReadWrite);
        for ( QStringList::Iterator it = lines.begin(); it !=
lines.end(); ++it )
            stream << *it << "\n";
        flSave.close();
        if (stream.status() != QTextStream::Ok){
        }
    }
    catch(...){
        qDebug() << "Error write file";
    }
}
```

3.6 Создание классов для доступа и работы с базой данных

Для того, чтобы получить данные от сервера необходимо установить подключение к SQL-серверу, отправить запрос и получить данные. Для этого был создан класс «QSqlDatabase». А функция подключения к базе

данных выглядит следующим образом:

```
bool DataBD::createConnect()
{
    QString LnInfo;
    QSqlDatabase db;
    try{
        LnInfo = QString::fromLocal8Bit(" Соединение с БД. Процедура
-- DataBD::createConnect() -- ");
        SaveToLogBD(LnInfo);
        db = QSqlDatabase::addDatabase("QMYSQL");
        db.setDatabaseName("DBINT");
        db.setUserName("boris");
        db.setPassword("asol123");
        if (!db.open()){
            return false;
        }
        closed = true;
        return true;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при соединении с
БД. Процедура -- DataBD::createConnect() -- ");
        SaveToLogBD(LnInfo);
        return false;
    }
}
```

Так же для удобства работы с базой данных были созданы специальные функции возвращающие результат запроса в необходимом типе данных. Так как эти функции подобны, рассмотрим на одном примере. Функция возвращает целое значение int:

```
int DataBD::GetValIntFromField(QString SqlString, int NbrFld)
{
    int RztValInt;
    QSqlQuery query;
    QString LnInfo;
    try{
        // LnInfo = QString::fromLocal8Bit(" Получить целое значение
поля БД. Процедура -- DataBD::GetValIntFromField(Prm) -- ");
        // SaveToLogBD(LnInfo);
        query.exec(SqlString);
        query.first();
        RztValInt = query.value(NbrFld).toInt();
        return RztValInt;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при получении
целого значения поля БД. Процедура -- DataBD::GetValIntFromField(Prm) --
");
        SaveToLogBD(LnInfo);
        return false;
    }
}
```

Так же в классе присутствуют функции, которые выполняют анализ вносимых значений в базу данных. Например, функция «GetBufMod»

выполняет проверку на наличие последней записи в таблице уровней сигналов с определенным номером пакета:

```

void DataBD::GetBufMod(struct masParam *Param)
{
    QString s_sql, LnInfo;
    bool R_yes;
    int cntP, Sign, five;
    try{
        LnInfo += QString::fromLocal8Bit(" Буфер сигналов. Процедура
-- DataBD::GetBufMod(Prm) -- ");
//      SaveToLogBD(LnInfo);
        s_sql = "Select * from `KTMSTATSIG` where `Modul` = ";
        s_sql += QString::number(Param->nmMod);
        s_sql += " and Num = "; s_sql += QString::number(Param-
>nmRpt);
        s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
        SaveToLogBD(s_sql);
        R_yes = GetRztNotNullBD(s_sql);
        if(R_yes != true){
            GetSqlInsertBD_Mod(Param);
            SaveToLogBD("R_yes = False GetBufMod ");
        }
        else{
            SaveToLogBD("R_yes = True GetBufMod");
            Sign = GetValIntFromField(s_sql, 4);
            LnInfo = QString::fromLocal8Bit(" Sign=");
            LnInfo += QString::number(Sign);
            SaveToLogBD(LnInfo);
            LnInfo = QString::fromLocal8Bit(" Nsign=");
            LnInfo += QString::number(Param->Sgnl);
            SaveToLogBD(LnInfo);
            five=(Sign*5)/100;
            LnInfo = QString::fromLocal8Bit(" five=");
            LnInfo += QString::number(five);
            SaveToLogBD(LnInfo);
            if ((abs(Sign-Param->Sgnl))<=five){
                cntP = GetValIntFromField(s_sql, 5);
                cntP++; Param->nCnt = cntP;
                GetSqlUpdtBD_Mod(Param);
                LnInfo = QString::fromLocal8Bit(" обновить
GetSqlUpdtBD_Mod ");
                SaveToLogBD(LnInfo);
            }
            else {
                GetSqlInsertBD_Mod(Param);
                LnInfo = QString::fromLocal8Bit(" Втавить
GetSqlInsertBD_Mod ");
                SaveToLogBD(LnInfo);
            }
        }
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в Буфере сигналов.
Процедура -- DataBD::GetBufMod(Prm) -- ");
        SaveToLogBD(LnInfo);
    }
}

```

3.7 Разработка модуля для работы с RS-232

Перед запуском класса «thread.cpp», необходимо задать настройки для работы с RS-232. Эти настройки хранятся в специальной таблице в базе данных. И исходя из этих значений программа выбирает заданные настройки:

```
bool EchoServer::IniComPort(int nport, int baud, int datab, int parity, int stopb)
{
    QString name_port;
    struct termios options;
    try{
// Установка имени порта
        switch(nport){
            case 0: { name_port = "/dev/ttyS0"; break; }
            case 1: { name_port = "/dev/ttyS1"; break; }
            case 2: { name_port = "/dev/ttyS2"; break; }
            case 3: { name_port = "/dev/ttyS3"; break; }
            case 4: { name_port = "/dev/ttyS4"; break; }
            case 5: { name_port = "/dev/ttyS5"; break; }
            case 6: { name_port = "/dev/ttyUSB0"; break;}
        }
        ds_port = ::open(name_port.toUtf8(), O_RDWR | O_NOCTTY | O_NDELAY);
// ds_port = ::open(name_port.toUtf8(), O_RDWR | O_NOCTTY);
        if(ds_port == -1) {
            ::close(ds_port);
            status_port = false;
        }
        else {
            //bzero(&options, sizeof(options));
            threadPak.port = ds_port;
            fcntl(ds_port, F_SETFL, FNDELAY);
            tcgetattr(ds_port, &options);
// Установка заданной скорости
            switch(baud){
                case 0: {
                    cfsetispeed(&options, B110);
                    cfsetospeed(&options, B110);
                    break;
                }
                case 1: {
                    cfsetispeed(&options, B300);
                    cfsetospeed(&options, B300);
                    break;
                }
                case 2: {
                    cfsetispeed(&options, B600);
                    cfsetospeed(&options, B600);
                    break;
                }
                case 3: {
                    cfsetispeed(&options, B1200);
                    cfsetospeed(&options, B1200);
                    break;
                }
                case 4: {
                    cfsetispeed(&options, B2400);
                    cfsetospeed(&options, B2400);
                }
            }
        }
    }
}
```

```

        break;
    }
    case 5: {
        cfsetispeed(&options, B4800);
        cfsetospeed(&options, B4800);
        break;
    }
    case 6: {
        cfsetispeed(&options, B9600);
        cfsetospeed(&options, B9600);
        break;
    }
    case 7: {
        cfsetispeed(&options, B19200);
        cfsetospeed(&options, B19200);
        break;
    }
    case 8: {
        cfsetispeed(&options, B38400);
        cfsetospeed(&options, B38400);
        break;
    }
    case 9: {
        cfsetispeed(&options, B57600);
        cfsetospeed(&options, B57600);
        break;
    }
    case 10: {
        cfsetispeed(&options, B115200);
        cfsetospeed(&options, B115200);
        break;
    }
}
// Установка битов данных
switch(datab){
    case 0: { options.c_cflag |= CS5; break; }
    case 1: { options.c_cflag |= CS6; break; }
    case 2: { options.c_cflag |= CS7; break; }
    case 3: { options.c_cflag |= CS8; break; }
}
// Установка контроля на четность
switch(parity){
    case 0: {
        options.c_cflag &= ~PARENB;
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;
        options.c_cflag |= CS8;
        break;
    }
    case 1: {
        options.c_cflag |= PARENB;
        options.c_cflag &= ~PARODD;
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;
        options.c_cflag |= CS7;
        break;
    }
    case 2: {
        options.c_cflag |= PARENB;
        options.c_cflag |= PARODD;
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;

```

```

        options.c_cflag |= CS7;
        break;
    }
    case 3: {
        options.c_cflag &= ~PARENB;
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;
        options.c_cflag |= CS8;
        break;
    }
}
// Установка стопового бита
switch(stopb){
    case 0: break;
    case 1: break;
    case 2: { options.c_cflag |= CSTOPB; break; }
}
options.c_cflag |= CLOCAL | CREAD;
options.c_cflag &= ~ CRTSCTS;
options.c_iflag = IGNPAR;
options.c_oflag &= ~ OPOST;
options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
options.c_cc[VMIN] = 0;
options.c_cc[VTIME] = 0;
tcflush(ds_port, TCIFLUSH);
tcsetattr(ds_port, TCSANOW, &options);
status_port = true;
}
LnInfo = QString::fromLocal8Bit(" Инициализировали порт.
Процедура -- EchoServer::IniComPort (Param) -- ");
SaveToLogW(LnInfo);
return status_port;
}
catch(...){
    LnInfo = QString::fromLocal8Bit("Ошибка в процедуре --
EchoServer::IniComPort (Param) --");
    SaveToLogW(LnInfo);
    return false;
}
}
}

```

После настройки порта, можно приступать к формированию и преобразованию пакета. Ниже представлена основная функция по работе с RS-232, которая содержит в себе вызов других функций по работе с пакетом:

```

void Thread::run()
{
    QString StrLog, StrOtv, Ln, pak, Ln_test, Tmp, Temp_log;
    unsigned char rsvBf[1024];
    unsigned char rsvPk[1024];
    unsigned char t_byte;
    int countread, FullPak;
    bool flag_CRC;
    try{
        StrLog = QString::fromLocal8Bit(" Процедура -- Thread::run()
-- ");
        SaveToFile(StrLog);
        while(!stopped){

```



```

countread = read(port, rsvBf, 1024);
if(countread < 0) {
    //== Ошибка чтения порта =====
    usleep(2000); //Ожидаем 2 мсек
} //-- End of if(countread < 0) -----
else if(countread==0) {
    //== В буфере порта нет данных =====
    Ln = " <<-- RS ";
    Ln += QString::fromLocal8Bit(" Len=00000 ");
    SaveToFile(Ln);
    usleep(2000); //Ожидаем 2 мсек
} //-- End of if(countread == 0) -----
else{
    for(int i=0; i<countread; i++){
        t_byte = rsvBf[i];
        InpBufQue.enqueue(t_byte);
    }
    Tmp=PrintPakHex(rsvBf, 13);
    FullPak=ParserPaket(rsvPk);
    if(FullPak <= 0){
        usleep(100);
        continue;
    }
    else{
        flag_CRC = GetStringSum(rsvPk);
        flag_CRC = true;
        if(flag_CRC != false){
            StrPakHex =
PrintPakHex(rsvPk, 13);
Ln = QString::fromLocal8Bit("
<<-- RS* ");
Ln += StrPakHex;
SaveToFile(Ln);
emit currentPak(StrPakHex);
        }
    }
    }
    usleep(10000);
}
stopped = false;
StrLog = QString::fromLocal8Bit(" Выход из цикла потока
run() ");
SaveToFile(StrLog);
}
catch(...){
    StrLog = QString::fromLocal8Bit(" Ошибка в процедуре --
Thread::run() -- ");
    SaveToFile(StrLog);
}
}

```

3.8 Разработка основного алгоритма работы сервера по записи событий

В основу алгоритма анализа пакета входят следующие функции:

- понять пакет от модуля объекта или от ретрослятора;
- проверить, есть ли такой модуль в базе данных;
- есть ли такое событие в базе данных;

- проверить, есть ли последнее событие от модуля под таким номером пакета;
 - проверить, новое событие или это повтор существующего;
 - распознавание типа события;
 - запись новых событий;
 - обновление счетчика повторных событий;
 - запись тревожных событий в отдельную таблицу;
- Основная функция содержащая вызов всех типов функций представлена ниже:

```

void EchoServer::RunOproS ()
{
    QString StrOutPak, line, sTmp, Temp, s_sql, LnInfo, Rep;
    unsigned short _len_test_buf;
    unsigned char _test_buf[16];
    // char print_buf[24]; //Буфер для печати
    struct masParam *Prm;
    struct masData *PrmDt;
    bool FlagSend, FlagPak, FlagLastMsg, FlagBuff, FlagMod;
    int cntP;

    try{
        FlagSend = false; FlagPak = false; FlagBuff= false; FlagLastMsg=
false;
        FlagMod = false;
        threadPak.start();
        StrOutPak = threadPak.StrPakHex;
    // ----- Начало обработки принятого пакета -----
        _len_test_buf=0;
        _len_test_buf = GetPak(StrOutPak, _test_buf);
        Prm = new struct masParam;
        GetParamData(_test_buf, Prm);
        PrmDt = new struct masData;
        PrintStrData(_test_buf, PrmDt);
        Rep
        =Prm->vlPak.mid(3,1);
        SaveToLogW (Rep);
        if (Rep == "9"){
            LnInfo = QString::fromLocal8Bit("Rep=9 ");
            LnInfo+=Prm->vlPak;
            Prm->nmMod=(Prm->vlPak.mid(4,1)).toInt(); ;
            Prm->Sgnl= Prm->nmPak;
            LnInfo += QString::fromLocal8Bit("nmMod = ");
            LnInfo += QString::number(Prm->nmMod);
            LnInfo += QString::fromLocal8Bit("nSgn = ");
            LnInfo += QString::number(Prm->Sgnl);
            SaveToLogW (LnInfo);
            taskBD.GetBufMod (Prm);
        }
        else{
            LnInfo = QString::fromLocal8Bit("Rep=A ");
            LnInfo+=Prm->vlPak;
            SaveToLogW (LnInfo);
            FlagMod = SendModul (PrmDt);
            if (FlagMod != false) { //есть такой модуль??
                taskBD.GetBufMod (Prm);
                FlagSend = SendEvents (PrmDt);
                if (FlagSend != false) { // Есть такое событие??
                    line = QString::fromLocal8Bit(" FlagSend = True!!! ");
                    FlagPak = SendNumbrPak (Prm);
                }
            }
        }
    }
}

```

```

        if(FlagPak != false){//Есть такой №pak and Mod??
            FlagBuff = DefDataPak(Prm);
        }//Есть такой №pak and Mod??
        if((FlagPak == false) or ((FlagPak == true) and (FlagBuff ==
true)) ) {//if 1
            Prm->nData = Prm->nUpd;
            taskBD.GetSqlInsertBD_Buffer(Prm);
            FlagLastMsg = SendLastMsg(Prm);
            if(FlagLastMsg != false){// Как последний № Pak??
                s_sql = "SELECT * FROM `KTMMMSG` WHERE `Modul`
= ";
                s_sql += QString::number(Prm->nmMod); s_sql
+="" and `Num`= "; s_sql += QString::number(Prm->nmPak);
                s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
                cntP = taskBD.GetValIntFromField(s_sql,8);
                cntP++; Prm->nCnt = cntP;
                taskBD.GetSqlUpdtBD_Msg(Prm);
            }// Как послед
            else {//else1
                taskBD.GetSqlInsertBD_Msg(Prm);
                line = PrmDt->vlEvnt; line += '*';
                line += QString::number(PrmDt->nmMod);
// Строка принятого события
                StrOutPak = line;
                for (int i = 0; i < SClients.size();
++i){//for
                    GetToClient(SClients.at(i),
StrOutPak);
                }//FOR
            }//else1
        }// if1
    }//есть событие
} //есть модуль
} //rep
delete Prm;
delete PrmDt;
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка в RunOpros");
    SaveToLogW(LnInfo);
}
} //end

```

Полный код программы обработки событий и записи их в базу данных представлен в приложении Б. А основной алгоритм работы на рисунке 4.1.



Рисунок 4.1 – Основной алгоритм работы сервера

4 Технико-экономическое обоснование

В рыночных условиях программное обеспечение выступает преимущественно в виде продукции научно-технических организаций, представляющей собой функционально завершенные и имеющие товарный вид программные средства, поставляемые заказчиком и продаваемые покупателям по рыночным ценам. Все завершенные разработки ПО являются научно-технической продукцией.

Широкое применение вычислительной техники (ВТ) требует постоянного обновления и совершенствования программного обеспечения. Выбор эффективных проектов ПО требует их экономической оценки и расчета экономического эффекта.

Экономический эффект у разработчика выступает в виде роста чистой прибыли (чистого дохода, ЧД) или чистого дисконтированного дохода (ЧДД), научно-технической организации от реализации ПО. Экономический эффект зависит от объема затрат на разработку проекта, уровня цены на разработанный программный продукт и объема продаж.

Стоимостная оценка ПО и определение экономического эффекта у разработчика предполагают составление сметы затрат, которая в денежном выражении включает следующие статьи расходов:

- где $Z_{\text{фот}}$ – общий фонд оплаты труда разработчиков, тенге;
- $Z_{\text{сзи}}$ – отчисления по социальному налогу, тенге;
- M_i – затраты на материалы, тенге;
- $P_{\text{си}}$ – затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;
- $P_{\text{ми}}$ – затраты, связанные с эксплуатацией техники, тенге;
- $P_{\text{зи}}$ – прочие затраты, тенге;
- $P_{\text{ни}}$ – накладные расходы, тенге.

4.1 Определение объема и трудоемкости разработки программного обеспечения

Базой для расчета плановой сметы затрат на разработку ПО является объем программного продукта. Общий объем (V_o) программного продукта определяется исходя из количества и объема функций, реализуемых программой:

$$V_o = \sum_{i=1}^n V_i \quad (4.1)$$

где V_i – объем отдельной функции ПО;

n – общее число функций.

Оценивание объема программного продукта связано с выбором наиболее подходящей единицы измерения размера продукта.

В дипломной работе в качестве единиц измерения объема ПО используется строка исходного кода (LOC). При подсчете LOC следует придерживаться следующих рекомендаций:

- учитывать «строку исходного кода» как одну, если в ней содержится лишь один оператор (если в одной строке содержатся два выполняемых оператора, разделяемых точкой с запятой, то нужно считать две строки, а если один выполняемый оператор разбит на две «физические» строки, то он будет учитываться как один оператор);

- учитывать все имеющиеся выполняемые операторы, поддерживаемые данным продуктом;

- определение данных учитывать лишь один раз;

- не учитывать строки, содержащие комментарии;

- не учитывать отладочный код или другой временный код (пробное ПО, средства тестирования, инструменты разработки и прототипирования и другие инструментальные средства);

- учитывать каждую инициализацию, вызов или включение макроса в качестве части исходного кода;

- не учитывать повторно используемые операторы исходного кода.

Расчет объема программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения (Приложение В), всестороннее техническое обоснование функций ПО и определение объема каждой функции. На стадии технико-экономического обоснования проекта могут быть получены только ориентировочные (прогнозные) оценки на основе имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения действующих нормативов (Приложение В). На основании информации о функциях разрабатываемого ПО по каталогу функций определяется объем функций и общий объем ПО, который уточняется (корректируется) с учетом условий разработки ПО в организации. Уточненный объем ПО (V_y) рассчитывается по формуле:

$$V_y = \sum_{i=1}^n V_{yi} \quad (4.2)$$

где V_{yi} – уточненный объем отдельной функции ПО (LOC).

Используя Приложение В по формуле 4.2 вычислим V_y :

$V_y = 500 + 1100 + 6150 + 5230 + 340 = 13370$ строк исходного кода

По уточненному объему ПО и нормативам затрат труда в расчете на единицу объема определяются нормативная и общая трудоемкость

разработки ПО.

Нормативная трудоемкость разработки ПО (T_n) определяется на основании принятого к расчету объема (V_y) и категории сложности (Приложение Г), которая уточняется с учетом сложности и новизны проекта и степени использования стандартных модулей при разработке.

Нормативная трудоемкость (T_n) служит основой для определения общей трудоемкости (T_o), расчет которой осуществляется различными способами в зависимости от размера проекта.

Общая трудоемкость небольших проектов рассчитывается по формуле:

$$T_o = T_n \times K_c \times K_m \times K_n \quad (4.3)$$

где K_c – коэффициент, учитывающий сложность ПО;

K_m – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

K_n – коэффициент, учитывающий степень новизны ПО.

Коэффициент сложности рассчитывается по формуле:

$$K_c = 1 + \sum_{i=1}^n K_i \quad (4.4)$$

где K_i – коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;

n – количество учитываемых характеристик.

Расчет коэффициента сложности производится на основе данных, представленных в Приложении Д.

Все ПО принято подразделять на три категории сложности (Приложение Д, см.таблицу Д.1) в зависимости от наличия (отсутствия) следующих характеристик:

- высокий уровень языкового интерфейса с пользователем;
- режим работы в реальном времени;
- управление удаленными объектами;
- машинная графика, многомашинные комплексы;
- существенное распараллеливание вычислений;
- нестандартная конфигурация технических средств;
- оптимизационные и особо сложные инженерные и научные

расчеты;

- переносимость ПО.

Из перечисленного списка присутствует - режим работы в реальном времени, поэтому:

$$K_c = 2$$

Поправочный коэффициент, учитывающий степень использования при разработке ПО стандартных модулей (K_m), определяется удельным

весом этих модулей в общем объеме проектируемого продукта (см. таблицу 4.1).

Т а б л и ц а 4.1 – Значения поправочного коэффициента

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО	Значения K_T
1. От 60 % и выше	0,6
2. От 40 % до 60	0,7
3. От 20 % до 40 %	0,8
4. До 20 %	0,9
5. Типовые программы и ПО не используемые для реализации функций разрабатываемого ПО	1,0

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО составляет от 20 % до 40 %, таким образом:

$$K_T = 0,8$$

Поправочный коэффициент, учитывающий новизну разрабатываемого ПО (K_H) определяется на основе данных таблицы 6.2.

Т а б л и ц а 4.2 – Поправочные коэффициенты

Категория новизны	Степень новизны	Использование		Значение K_H
		На основе нового типа ПК	В среде новой ОС	
А	Принципиально новые ПО, не имеющие доступных аналогов	+	+	1,75
		–	+	1,6
		+	–	1,2
		–	–	1,0
Б	ПО, являющиеся развитием определенного параметрического ряда ПО	+	+	1,0
		–	–	0,9
		+	–	0,8
В	ПО, являющиеся развитием определенного параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС	–	–	0,7

Исходя из таблицы 5.2, категория новизны –В, соответственно:
 $K_H = 0,7$

Таким образом, общая трудоемкость составляет:

$$T_o = 399 \times 2 \times 0,8 \times 0,7 = 446,88 \text{ чел./дней}$$

Численность исполнителей и срок разработки ПО. На основе общей трудоемкости определяются плановое число разработчиков ($Ч_p$) и плановые сроки, необходимые для реализации проекта в целом (T_p). При этом могут решаться следующие задачи:

- расчет числа исполнителей при заданных сроках разработки проекта;

- определение сроков разработки проекта при заданной численности исполнителей.

Численность исполнителей проекта ($Ч_p$) рассчитывается по формуле:

$$Ч_p = T_o / (T_p \times \Phi_{эф}) \quad (4.5)$$

где $\Phi_{эф}$ – эффективный фонд времени работы одного работника в течение года (дн.);

T_o - общая трудоемкость разработки проекта (чел./дн.);

T_p - срок разработки проекта (лет).

Срок разработки проекта (T_p) определяется по формуле:

$$T_p = T_o / (Ч_p \times \Phi_{эф}) \quad (4.6)$$

Эффективный фонд времени работы одного работника ($\Phi_{эф}$) рассчитывается по формуле:

$$\Phi_{эф} = D_z - D_n - D_v - D_o, \quad (4.7)$$

где D_z - количество дней в году (365 дней);

D_n - количество праздничных дней в году (17 дней);

D_v - количество выходных дней в году (102 дня);

D_o - количество дней отпуска (24 дня).

$$\Phi_{эф} = 365 - 17 - 102 - 24 = 222 \text{ дня}$$

Так как срок разработки проекта (T_p) составляет 0,5 года, вычислим численность исполнителей проекта:

$$Ч_p = 446,88 / (0,5 \times 222) = 4 \text{ человека}$$

4.2 Расчет затрат на разработку информационных технологий

Количество и заработная плата работников, задействованных в разработке, указано в таблице 4.3

Т а б л и ц а 4.3 -Количество работников и заработная плата

Исполнитель	Количество, человек	Зарботная плата в месяц, тенге
Технический директор	1	200 000
Ведущий инженер	1	150 000
Программист	2	110 000
Итого	4	570 000

Расчет полных затрат на разработку проектного решения в виде информационных технологий (C_{ni}) осуществляется по формуле:

$$C_{ni} = Z_{фот} + Z_{czi} + M_i + P_{ci} + P_{mi} + П_{zi} + P_{ni}, \quad (4.8)$$

где $Z_{фот}$ - общий фонд оплаты труда разработчиков, тенге;

Z_{czi} - отчисления по социальному налогу, тенге;

P_{ci} - затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

P_{mi} - затраты, связанные с эксплуатацией техники, тенге;

$П_{zi}$ - прочие затраты, тенге;

P_{ni} - накладные расходы, тенге.

Размер фонда оплаты труда разработчиков ($Z_{фот}$) рассчитывается по формуле:

$$Z_{фот} = Z_{oi} + Z_{di}, \quad (4.9)$$

где Z_{oi} - основная заработная плата, тенге;

Z_{di} - дополнительная заработная плата, тенге.

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

$$Z_{oi} = \sum_{i=1}^n T_{ci} \times T_{ч} \times \Phi_n \times K \quad (4.10)$$

где n - количество исполнителей, занятых разработкой конкретного ПО;

T_{ci} - часовая тарифная ставка i -го исполнителя (тыс.тенге);

Φ_n - плановый фонд рабочего времени i -го исполнителя (дней);

$T_{\text{ч}}$ - количество часов работы в день (8 часов);

K - коэффициент премирования.

Месячная тарифная ставка каждого исполнителя ($T_{\text{м}}$) определяется путем умножения действующей месячной тарифной ставки 1-го разряда ($T_{\text{м1}}$) на тарифный коэффициент ($T_{\text{к}}$), соответствующий установленному тарифному разряду:

$$T_{\text{м}} = T_{\text{м1}} \times T_{\text{к}}. \quad (4.11)$$

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленную при 40-часовой недельной норме рабочего времени расчетную среднемесячную норму рабочего времени в часах ($\Phi_{\text{р}}$):

$$T_{\text{ч}} = T_{\text{м}} / \Phi_{\text{р}} \quad (4.12)$$

где $T_{\text{ч}}$ - часовая тарифная ставка (тыс.тенге);

$T_{\text{м}}$ - месячная тарифная ставка (тыс.тенге).

Результаты расчета основной заработной платы представлены в таблицы 4.4.

Часовая тарифная ставка для:

Технического директора: $T_{\text{ч}}=200000/24*8= 1042$ тнг./час

Ведущего инженера: $T_{\text{ч}}=150000/24*8= 781$ тнг./час

Для программистов: $T_{\text{ч}}=110000/24*8= 573$ тнг./час

Используя формулу 4.10 вычислим основную заработную плату исполнителей:

$$Z_{oi}=(1042*8*85)+(781*8*98)+(573*8*113)*2==2356848 \text{ тенге}$$

Т а б л и ц а 4.4 - Сводные результаты затрат основной заработной платы

Наименование содержания работ	Исполнитель	Трудоёмкость Норма, час	Заработная плата за час работы, тнг./час	Сумма заработной платы, тнг.
1	2	3	4	5

Т а б л и ц а 4.4 - Сводные результаты затрат основной заработной

платы

Окончание таблицы 4.4

Проведение исследований, распределение обязанностей, разработка рабочего проекта	Технический директор	85 *8=680	1042	708560
Закупка оборудования и ПО, разработка плана, реализация проекта, наладка и тестирование системы, проведение обучения конечных пользователей	Ведущий инженер	98 *8=784	781	612304
Разработка рабочего проекта, Наладка и тестирование системы, проведение обучения конечных пользователей	2 Программиста	(113*8)*2=1808	573	1035984
Итого				2356848

Дополнительная заработная плата составляет 10% от основной заработной платы и рассчитывается по формуле:

$$Z_{oi} = Z_{oi} \times H_{\delta} / 100 \quad (4.13)$$

где H_{δ} - коэффициент дополнительной заработной платы разработчиков.

$$Z_{oi} = 2356848 \times 0.1 = 235685 \text{ тенге}$$

$$Z_{\text{ФОТ}} = Z_{oi} + Z_{oi} = 2356858 + 235685 = 2592533 \text{ тенге}$$

Социальный налог составляет 11% (ст. 358 п. 1 НК РК) от дохода работника, и рассчитывается по формуле:

$$Z_{\text{сзи}} = (Z_{\text{ФОТ}} - \text{ПО}) \times 11\% \quad (4.14)$$

где $ПО$ - пенсионные отчисления, которые составляют 10% от ФОТ и социальным налогом не облагаются:

$$ПО = ФОТ \times 10\% \quad (4.15)$$

$$ПО = 2592533 \times 0.1 = 259253 \text{ тенге}$$

$$З_{сзи} = (2592533 - 259253) \times 0.11 = 256661 \text{ тенге}$$

Расходы по статье «Спецоборудование» (P_{ci}) включают затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного ПО, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию:

$$P_{ci} = \sum_{i=1}^n Ц_{ci} \quad (4.16)$$

где $Ц_{ci}$ - стоимость конкретного специального оборудования (тыс.тенге);

n - количество применяемого специального оборудования.

$Ц_{ci} = 8970$ тенге - Контрольно-передающая панель.

$P_{ci} = 8970$ тенге.

Расходы по статье «Машинное время» (P_{mi}) включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (H_{mv}) машинного времени в зависимости от характера решаемых задач и типа ПК (Приложение Д):

$$P_{mi} = Ц_{mi} \times (V_{oi}/100) \times H_{mv} \quad (4.17)$$

где $Ц_{mi}$ - цена одного машино-часа (тыс.тенге);

V_{oi} - общий объем ПО (строк исходного кода);

H_{mv} - норматив расхода машинного времени на отладку 100 строк исходного кода (машино-часов).

$$Ц_{mi} = W \times S \times K_{им} \quad (4.18)$$

где W - установленная мощность приборов, потребляющих электроэнергию, кВт(1,5);

S - стоимость киловатт-часа электроэнергии (24,32 тг/кВт·ч);

$K_{им}$ - коэффициент использования мощности (0,8);

$$Ц_{mi} = 0,7 \times 24,32 \times 0,8 = 29,2 \text{ тенге}$$

$$P_{mi}=29,2 \times (13370/100) \times 12=46848 \text{ тенге}$$

Затраты по статье «Накладные расходы» (P_{ni}), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (P_{ni}), относятся на конкретное ПО по нормативу (H_{pn}) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации:

$$P_{ni} = Z_{oi} \times H_{pn}/100\% \quad (4.19)$$

где P_{ni} -накладные расходы на конкретную ПО (тыс.тенге);

H_{pn} -норматив накладных расходов в целом по организации в (%), в дипломном проекте - 70% .

$$P_{ni}=2356858 \times 0,7=1649801 \text{ тенге}$$

Величина затрат на материалы на основании исходных данных определяется по формуле:

$$M_i = (Z_{och} \times H_{mz})/100\% \quad (4.20)$$

где H_{mz} -норма расхода материалов от основной заработной платы (5%).

$$M_i=2356858 \times 0,05=117843 \text{ тенге.}$$

Расходы по статье «Прочие затраты» (P_{zi}) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате:

$$P_{zi} = Z_{oi} \times H_{nz}/100\% \quad (4.21)$$

где H_{nz} - норматив прочих затрат в целом по организации в (%), в дипломном проекте 20% .

$$P_{zi}=2356858 \times 0,2=471372 \text{ тенге}$$

Результаты выполненных расчётов представлены в таблице 4.5 и на рисунке 4.1.

Т а б л и ц а 5.5 – Затраты на разработку программного обеспечения для охранного мониторинга

Затраты на разработку	Условное обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	$Z_{ФОТ}$	2356858	48
Социальный налог	$Z_{сзи}$	256661	5,2
Прочие затраты	$П_{зи}$	471372	9,6
Материалы	M_i	117843	2,4
Спецоборудование	$P_{си}$	8970	0,9
Машинное время	$P_{ми}$	46848	0,9
Накладные расходы	$P_{ни}$	1649801	33
Итого:		4908353	100

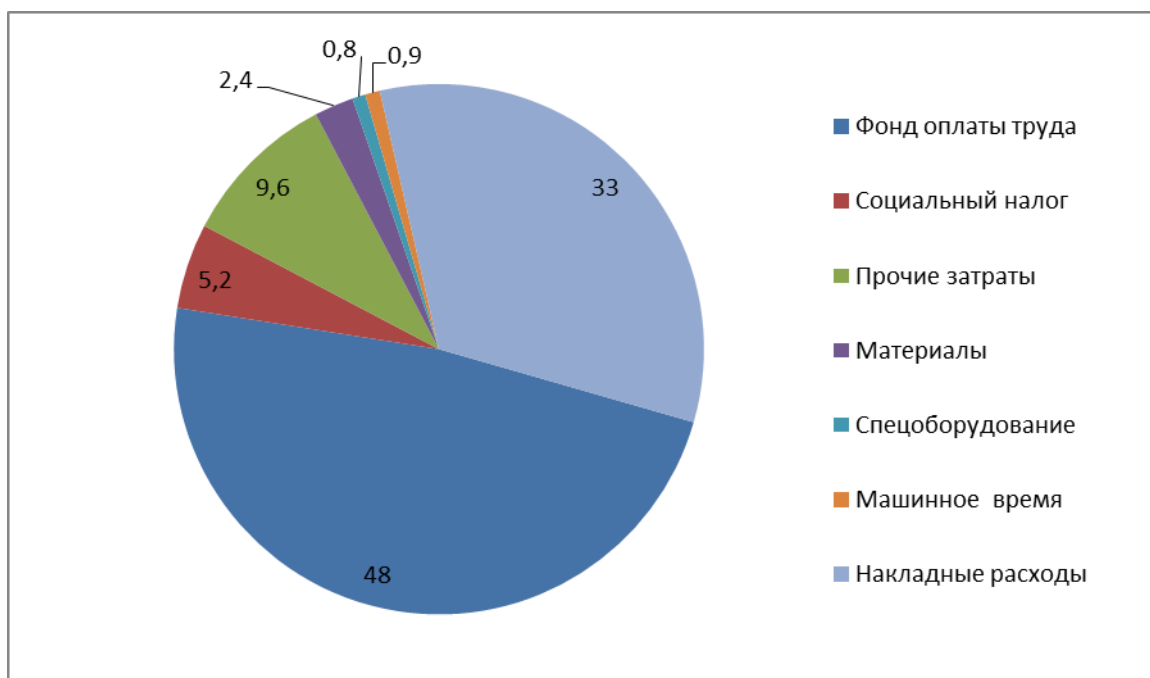


Рисунок 4.1 – Структура затрат

4.3 Вывод по расчету затрат на разработку информационных технологий

Для разработки программного обеспечения для охранного мониторинга необходима сумма в размере 4908353 тенге. Большую часть затрат занимает фонд оплаты труда - 2356858 тенге, что составляет 48%. Так же большую часть затрат составляет накладные расходы – 1649801 тенге, что составляет 33%.

5 Безопасность жизнедеятельности

5.1 Анализ условий труда

При разработке программ большое значение имеет организация условий труда, обеспечивающих нормальную работу рабочего персонала в течение всего рабочего дня, так как в процессе труда человек подвергается воздействию целого ряда санитарно-гигиенических факторов, которые могут вызвать нежелательные последствия, например чрезмерное повышение или понижение температуры тела, повышение давления.

В соответствии с требованиями ГОСТ12.1.005-88 ССБТ нормируются оптимальные и допустимые условия микроклимата (температура воздуха, его влажность, а также скорость в рабочей зоне).

Т а б л и ц а 5 . 1 – Допустимые и оптимальные параметры микроклимата

Период года	Теплый	Холодный
Температура t,С		
допустимая	17-23	28
оптимальная	18-20	20-22
Скорость воздуха w,м/с		
допустимая	0.3	0,4
оптимальная	0.2	0.3
Влажность воздуха %		
допустимая	75	75
оптимальная	40-60	40-60

Свет, освещение относится к одному из основных внешних факторов, постоянно воздействующих на человека в процессе труда. Положительное влияние освещения на производительность труда и его качество не вызывает сомнения. Так, солнечное освещение увеличивает производительность труда в среднем на 10%, а искусственное на 13%, при этом возможность брака снижается на 20-25% [12].

Комната представляет собой помещение длиной 10 м, шириной 5 м и высотой 3 м, имеются 2 слуховых окна длиной 2,5 м и высотой 2 м. В помещении работают до 3 человек. Согласно ГОСТ 12.1.005-88 ССБТ "Воздух рабочей зоны, общие санитарно-гигиенические требования", работа людей в помещении относится к работе средней тяжести. Категории работ по энергозатратам приведены в таблице таблица 5.2.

Т а б л и ц а 5.2 – Категории работ по энергозатратам организма

Работа, Категория	Энергозатраты организма, Дж/с (ккал/час)	Характеристика работы
Физическая средней тяжести	172 – 232	Связанная с ходьбой, выполняемая стоя или сидя, но не требующая перемещения тяжестей

Рассчитаем защитное зануление и кондиционирование помещения, которые удовлетворяют нормам труда.

5.2 Защитное заземление, зануление

Заземлением (рисунок 5.1) называется соединение с землей нетоковедущих металлических частей электрооборудования через металлические детали, закладываемые в землю и называемые заземлителями, и детали, прокладываемые между заземлителями и корпусами электрооборудования, называемые заземляющими проводниками. Проводники и заземлители обычно делаются из низкоуглеродистой стали, называемой в просторечии железом. Где на рисунке 7.1: $-Z_c, Z_b$ - полные сопротивления проводов относительно земли, I_k – ток короткого замыкания, F – разрядник.

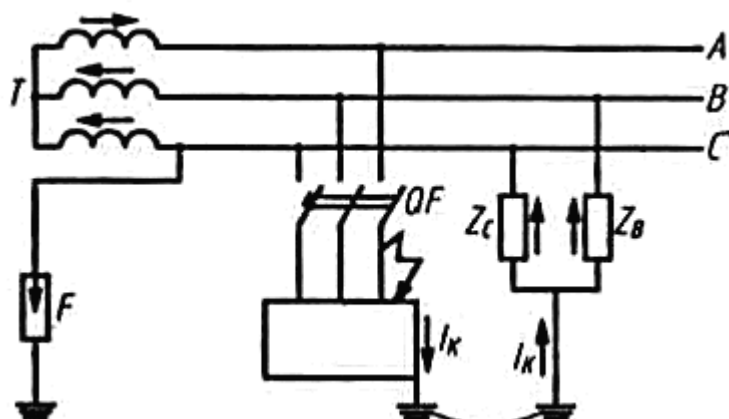


Рисунок 5.1 – Схема заземления в сети с изолированной нейтралью при наличии короткого замыкания

Заземлители в виде штырей, вбиваемых в землю, называются электродами, и могут быть одиночными или групповыми. Заземлитель имеет характеристики, обусловленные стеканием по нему тока в землю. К характеристикам заземлителя относятся:

- напряжение на заземлителе;
- изменение потенциалов точек в земле вокруг заземлителя в зависимости от их расстояния от заземлителя в зоне растекания тока — вид потенциальной кривой;
- вид линий равного потенциала — эквипотенциальных линий на поверхности земли;
- сопротивление заземляющего устройства;
- напряжения прикосновения и шага [11].

На рисунке 5.2 показана схема простого заземлителя в виде стержня или трубы, забиваемых в землю и вид потенциальных кривых и эквипотенциальных линий, условными обозначениями обозначены: 1 - заземляющий проводник, 2 - заземлитель, 3 - эквипотенциальные линии, 0φ - ось величин потенциала, $0x$ - ось расстояний до заземлителя, $\varphi(x)$ - потенциальная кривая, I_3 - ток в заземлителе, $\varphi_3 = U_3$ - напряжение на заземлителе.

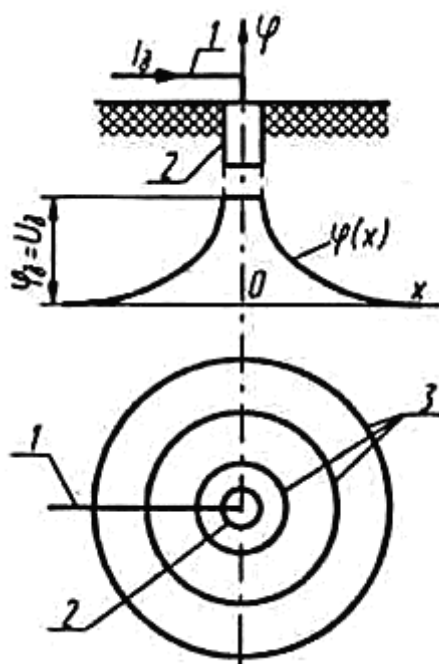


Рисунок 5.2 – Распределение потенциалов у поверхности земли в зоне растекания одиночного заземлителя.

При расстоянии менее 40 м между одиночными заземлителями в групповом заземлителе их зоны растекания накладываются друг на друга, и получается одна зона растекания группового заземлителя, которой соответствует своя потенциальная кривая.

Зануление (рисунок 5.3) предусматривает глухое заземление нейтрали источника или трансформатора трехфазного тока, одного вывода источника однофазного тока, наличие нулевого провода и его повторного

заземления [11].

Заземление нейтрали источника тока имеет целью понизить напряжение на корпусах оборудования и на нулевом проводе, с которым эти корпуса соединены, до безопасного значения при замыкании фазного проводника на землю, при этом создается путь для тока $I_{\phi-3}$. На рисунке обозначены: N – нулевой проводник, $I_{\phi-3}$ – ток замыкания на землю, I_k – ток короткого замыкания, $R_{з\text{м}}$ – сопротивление заземления нулевого провода, $R_{з\text{м пов}}$ – тоже повторное, $R_{з\text{ам}}$ – сопротивление замыкания фазы на землю.

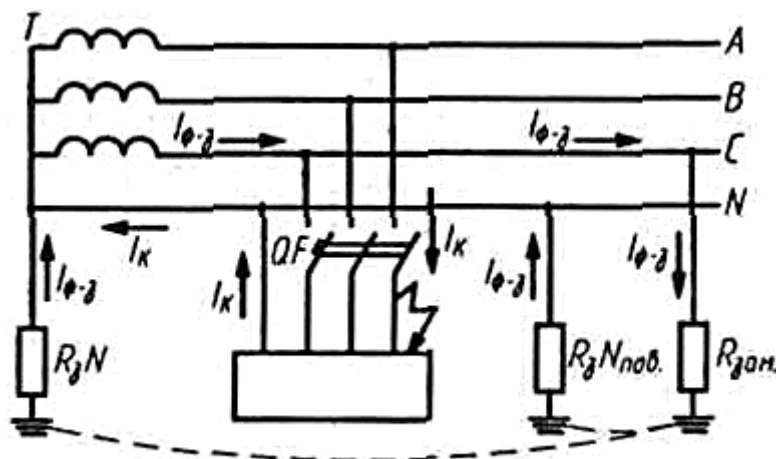


Рисунок 5.3 – Схема зануления при наличии короткого замыкания фазы A на корпус и замыкания фазы C на землю.

Нулевой защитный проводник предназначен для увеличения тока короткого замыкания I_k с целью воздействия этого тока на защиту. Увеличение I_k происходит за счет уменьшения сопротивления току при наличии нулевого провода по сравнению с тем, если бы ток шел через землю.

Повторное заземление нулевого провода предназначено для снижения напряжения на корпусах оборудования при замыкании фазы на корпус как при исправном, так и при оборванном нулевом проводе.

Зануление в электроустановках до 1000 В применяется в 4-проводных сетях с глухо-заземленной нейтралью трансформатора или генератора, в сетях с заземленным выводом источника однофазного тока, в сетях с заземленной средней точкой источника постоянного тока. Зануление выполняется в тех же случаях, что и защитное заземление [11].

Предельные величины сопротивлений заземляющих устройств в системе зануления приведены в таблице 5.3.

Заземление нейтрали источника тока имеет целью понизить напряжение на корпусах оборудования и на нулевом проводе, с которым эти корпуса соединены.

Т а б л и ц а 5.3 – Предельные величины сопротивлений заземляющих устройств

Напряжение сети, В		Сопротивление, Ом			
линейное 3-фазного тока	однофаз ного тока	Заземляюще го устройства нейтрал трансформато ра или генератора	Заземлите лярасполо женного у нейтрал и	общее всех повторных заземлени й нулевого провода	каждого повторног о заземлени я нулевого провода
660	380	2	15	5	15
380	220	4	30	10	30
220	127	8	60	20	60

В качестве нулевых защитных проводников используются нулевые рабочие проводники, за исключением проводников с передвижным электроприемникам. В цепи нулевых защитных проводников не должно быть аппаратов, разъединяющих эти проводники, в том числе предохранителей.

Проверка зануления на соответствие требованиям ПУЭ производится во время монтажа, при сдаче после монтажа и при эксплуатации [10].

Проверяют следующие параметры:

- сопротивление заземлений нейтрали и повторных;
- отношение тока однофазного КЗ на корпус и номинального тока плавкой вставки предохранителя или тока вставки автомата на контролируемом участке сети, причем это отношение должно быть не менее 3, а для автоматов только с электромагнитными расцепителями на номинальный ток до 100А кратность должна быть не менее 1,4 и для автоматов на ток более 100А - 1,25.

5.3 Расчет системы кондиционирования помещения

Обеспечение воздушного комфорта производственных помещений зависит и от системы кондиционирования воздуха. Система кондиционирования воздуха, помимо выполнения задач вентиляции и отопления, позволяет создать благоприятных микроклимат в летний, жаркий период года, благодаря использованию в своем составе фреоновой холодильной установки.

Технологические системы кондиционирования предназначены для обеспечения параметров воздуха, в максимальной степени отвечающих требованиям определенного производственного или технологического процесса.

Задача кондиционирования воздуха состоит в поддержании таких параметров воздушной среды, при которых каждый человек благодаря своей индивидуальной системе автоматической терморегуляции организма чувствовал бы себя комфортно, т.е. не замечал влияния этой среды.

Приведем расчет системы кондиционирования. Кондиционирование обеспечивает наилучшее состояние микроклимата в помещении и обеспечивает условия работы точной и чувствительной аппаратуры, расчет системы кондиционирования должен выполняться в соответствии со СНиП РК 4.02-42-2006 «Отопление, вентиляция и кондиционирование воздуха».

$$n = L/V_p \quad (5.1)$$

где V_p - объем помещения, м³

L - количество воздуха, поступающего в помещение, м³/час

n - под кратностью принимают отношение объема воздуха, подаваемого в помещение или извлекаемого из него системой общеобменной вентиляции в течение 1 часа, к объему воздуха в помещении.

$$Q = Q_{об} + Q_{осв} + Q_{п} + Q_{р} - Q_{отд} \quad (5.2)$$

где $Q_{об}$ - тепло, выделяемое оборудованием.

В помещении находятся: 3 персональных компьютера по 500 Вт (вместе с мониторами)

$$Q_{об} = P_{об} \cdot \eta_1 \cdot \eta_2, \quad (5.3)$$

где $P_{об}$ - мощность, потребляемая всем оборудованием, находящимся в помещении;

η_1 - коэффициент использования мощности = 0,7;

η_2 - коэффициент, характеризующий долю энергии, перешедшей в тепло = 0,6.

$$Q_{об} = 3 \cdot 500 \cdot 0,7 \cdot 0,6 = 630 \text{ (Вт)}$$

$Q_{осв}$ – тепло, выделяемое системой искусственного освещения

$$Q_{осв} = N_{осв} \quad (5.4)$$

где $N_{осв}$ - суммарная мощность источников освещения, принимается равной 50 – 100 Вт/м².

$$Q_{\text{осв}} = 75 \cdot 5 \cdot 10 = 3750 \text{ (Вт)}.$$

$Q_{\text{п}}$ - тепло, выделяемое работающим персоналом.

Тепловыделения человека зависят от тяжести работы, температуры окружающего воздуха и скорости движения воздуха. В расчете используется явное тепло, т.е. тепло, воздействующее на изменение температуры воздуха в помещении. Для умственной работы количество явного тепла, выделяемое одним человеком, составляет 165 Вт при 10оС и 5 Вт при 35оС. Для нормальных условий (20оС) явные тепловыделения одного человека составляют около 55 Вт. В рассчитываемом помещении находится 3 человека. Тогда суммарное тепловыделение от людей будет:

$$Q_{\text{п}} = 3 \cdot 55 = 165 \quad (5.5)$$

$Q_{\text{р}}$ – тепло, вносимое в помещение солнцем (солнечной радиацией), определяется по формуле:

$$Q_{\text{р}} = (q^I F^I + q^{II} F^{II}) \beta \quad (5.6)$$

Где q^I, q^{II} - тепловые потоки от прямой и рассеянной солнечной радиации, Вт/м²;

F^I, F^{II} - площади светового проема, облучаемые и необлучаемые прямой солнечной радиацией, м²;

β - коэффициент теплопропускания (см. таблицу 5.4).

Т а б л и ц а 6.4 – Коэффициенты теплопропускания солнцезащитных устройств β

Солнцезащитные устройства, внутренние	β
Шторы из светлой ткани	0,4

При отсутствии наружных затеняющих козырьков, ребер и т.д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение $F^I = F_0; F^{II} = 0$, ($F_0 = n \cdot H_0 \cdot B_0$, где n – число окон, H_0 – высота, B_0 – ширина).

$$Q_{\text{р}} = 2 \cdot 2,5 \cdot 2 \cdot 0,4 \cdot 145 = 580 \text{ Вт}.$$

$Q_{\text{отд}}$ -теплоотдача естественным путем через конструкции помещения, приблизительно равна количеству тепла, вносимого в помещение солнечной радиацией через окна, то есть:

$Q_{\text{отд}}^{\text{T}} = 0$ Вт для теплого периода;

$Q_{\text{отд}}^{\text{x}} = Q_{\text{p}} = 580$ Вт для холодного периода;

Для теплого периода избыточное тепло:

$$Q^{\text{T}} = 630 + 3750 + 165 + 580 - 0 = 5125 \text{ Вт}$$

Для холодного периода избыточное тепло:

$$Q^{\text{x}} = 630 + 3750 + 165 + 580 - 580 = 4545 \text{ Вт}$$

Если теплонапряженность воздуха:

$$Q_{\text{н}}^{\text{T}} > 23 \text{ Вт/м}^3 \text{ то } \Delta t = 8^{\circ}\text{C}$$

Так как $Q_{\text{н}}^{\text{T}} > 23 \text{ Вт/м}^3$ то $\Delta t = 8^{\circ}\text{C}$.

Теплонапряженность воздуха для холодного периода:

$$Q_{\text{н}}^{\text{x}} = \frac{Q^{\text{x}}}{V_{\text{p}}} = \frac{4545}{10 \cdot 5 \cdot 3} = 30,3 \text{ Вт/м}^3$$

Так как $Q_{\text{н}}^{\text{x}} > 23 \text{ Вт/м}^3$ то $\Delta t = 8^{\circ}\text{C}$.

Количество воздуха, удаляемого (подаваемого) из помещения в теплый период рассчитываем по формуле (5.2):

$$L^{\text{T}} = \frac{3,6 \cdot 5125}{1,2 \cdot 8} = 1921,88 \text{ м}^3/\text{ч}$$

Количество воздуха, удаляемого (подаваемого) из помещения в холодный период:

$$L^{\text{x}} = \frac{3,6 \cdot 4545}{1,2 \cdot 8} = 1704,38 \text{ м}^3/\text{ч}$$

Требуемая кратность воздухообмена для теплого периода:

$$n^{\text{T}} = \frac{1921,88}{150} = 12,81 \text{ ч}^{-1}$$

Требуемая кратность воздухообмена для холодного периода:

$$n^x = \frac{1704,38}{150} = 11,36 \quad \text{ч}^{-1}$$

Объем помещения приходящийся на одного работающего:

$$V_1 = \frac{V_p}{K} = \frac{150}{3} = 50 \text{ м}^3$$

Проверяем соответствие количества подаваемого воздуха санитарными нормами в теплый период:

$$V = \frac{L^T}{K} = \frac{1921,88}{3} = 640,6 \text{ м}^3/\text{ч} \cdot \text{чел}$$

Это значение соответствует СНиП РК 4.02-42-2006.

Для удаления необходимого количества воздуха (1705 19066 м³/час) оборудуем помещение полупромышленным кондиционером фирмы LG. Технические характеристики кондиционера представлены в таблице 6.5.

Т а б л и ц а 6.5 – Основные технические характеристики кондиционера LG P05LN

Основные характеристики	Значение
Электропитание, В	380 В±10 % 50 Гц
Производительность охлаждения, кВт	13,5
Производительность нагрева, кВт	14,1
Номинальный расход воздуха, м ³ /ч	2100
Режим работы	Автоматическая регулировка

Полупромышленные прецизионные кондиционеры представляют собой разновидность шкафных кондиционеров. Они оборудованы различными типами систем микропроцессорного управления и способны поддерживать в помещении не только точные параметры по температуре, но и по влажности.

Прецизионные кондиционеры обладают следующими отличительными характеристиками:

- точность контроля и поддержания температуры (±1%) и влажности (±2%);
- надежность работы при непрерывной эксплуатации;
- возможность работы в широком диапазоне температур наружного воздуха (до минус 35);
- полная совместимость с системами диспетчерского контроля и системами управления микроклиматом здания.

Полупромышленные прецизионные кондиционеры используются в промышленных помещениях, автозалах, серверных, при обслуживании технических узлов связи и других помещениях. С целью создания благоприятных условий для жизнедеятельности человека. Кондиционер обеспечивает:

- охлаждение воздуха;
- автоматическое поддержание заданной температуры;
- очистку воздуха от пыли;
- вентиляцию;
- воздухообмен с окружающей средой.

Для создания микроклимата на рабочем месте устанавливаем 1 кондиционер LG P05LN, т.к. он обеспечивает необходимое поступление воздуха в помещение комнаты.

Заключение

Результатом проделанной работы является программный продукт, состоящий из программы оператора ПЦМ, осуществляющего ряд функций для обеспечения ввода и обработки поступающих сигналов, а также базы данных объектов, в которой осуществляется хранение всей информации по объектам и их состояниям. База данных также содержит таблицы с дополнительной информацией необходимой для расчета основных технологических данных.

Основным преимуществом разработанного программного продукта является система обработки различных сигналов, которая была создана на основе анализа существующей схемы прохождения информации на ПЦМ.

Помимо этого, при помощи программного продукта может быть осуществлен контроль над состоянием прохождения сигналов, благодаря выбору статуса на каждой из стадий работы с программой.

Использование программы позволит компании осуществить автоматизированное сопровождение и обработку информации, а благодаря технологии «клиент-сервер» сберечь время для получения необходимой информации.

Для сопровождения программного продукта была разработана инструкция для пользователя программы, позволяющая в короткие сроки освоить общие приемы работы.

Таким образом, перевод информации на компьютерную сетевую основу позволяет не только уменьшить стоимость хранения информации, но и хранить, структурировать и извлекать информацию оптимальным для пользователя образом.

Список используемой литературы

1. Виейра Р. Программирование баз данных MySQL . Базовый курс / Р. Виейра.- М.: Вильямс, 2008.
2. Сайт: <http://valera.asf.ru/cpp/book/>
3. Сайт: <http://ru.wikipedia.org/wiki/Qt>
4. Виейра Р. Программирование баз данных Microsoft SQL Server 2005. Базовый курс / Р. Виейра. -М.: Вильямс, 2008.– 833 с.
5. Горев, А. Эффективная работа с СУБД / А. Горев, С. Макашарипов, Р. Ахьян. – СПб.: Питер, 2005.– 704 с.
6. Малик С. С++ для профессионалов / С. Малик. – М.: Вильямс, 2006.– 560 с.
7. Стенли Б. Липпман, Барбара Э. Му, Жози Лажойе. Язык программирования С++ (Си). Вводный курс. – М.: «Вильямс», 2006 – 896 с.
8. Бьярн С. Язык программирования С++. 3-е издание. Специальное издание./ Бьярн С.-М.: Вильямс, 2009.
9. З.Д. Еркешева, Г.Ш. Боканова. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В070400 – Вычислительная техника и программное обеспечение. – Алматы: АУЭС, 2013
10. Куатова Д.Я. Экономика предприятия.-А.: «Экономика», 2011.
11. Хакимжанов Т.Е. Безопасность жизнедеятельности. Расчет аспирационных систем. – Алматинский институт энергетики и связи, 2007. – 32с.

Приложение А

```
unit Unit_Oper;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, DM, ComCtrls, ImgList, ToolWin, Menus, StdCtrls, ExtCtrls,
  Registry,
  Unit_Card, ScktComp, Grids, Buttons, ShellApi, Unit_Strgr, MPlayer,
  MMSystem,
  U_Const ;

type
  Rec_Data = record
    Prm1, Prm2, Prm3, Prm4: String;
    Prm5, Prm6, Prm7, Prm8: String;
    Prm9, Prm10, Prm11, Prm12: String;
  end;
  MParam = Rec_Data;
  TFrm_Oper = class(TForm)
    StatusBarInfo: TStatusBar;
    MnM_Oper: TMainMenu;
    Pnl_Left: TPanel;
    Pnl_Right: TPanel;
    Splitter1: TSplitter;
    GrpBxSearch: TGroupBox;
    Splitter2: TSplitter;
    Splitter3: TSplitter;
    GrpBxFeatures: TGroupBox;
    GrpBxAlarms: TGroupBox;
    Splitter4: TSplitter;
    Splitter5: TSplitter;
    GrpBxStates: TGroupBox;
    PnlSearch_Data: TPanel;
    PnlSearch_Info: TPanel;
    LbSearch_OB: TLabel;
    EdSearch_OB: TEdit;
    EdSearch_Mod: TEdit;
    LbSearch_Mod: TLabel;
    LbSearch_Titl: TLabel;
    LbSearch_Bg: TLabel;
    DtTmPckSearch_Bg: TDateTimePicker;
    DtTmPckSearch_End: TDateTimePicker;
    LbSearch_End: TLabel;
    BtnSearch: TButton;
    BtnSearchCard: TButton;
    MemoFeatures: TMemo;
    StrGrdSearch: TStringGrid;
    StrGrdAlarms: TStringGrid;
    StrGrdStates: TStringGrid;
    StBar_TmSt: TStatusBar;
    BitBtn1: TBitBtn;
    PM_StrGr_States: TPopupMenu;
    Vnimanie: TMenuItem;
    Clear_Strgr: TMenuItem;
    StrGrdAlarms_220: TStringGrid;
    GrpBxAlarms_220: TGroupBox;
    procedure MnMOper_CnctClick(Sender: TObject);
  end;
end;
```

Продолжение приложения А

```
procedure TlBtnOper_CnctClick(Sender: TObject);
procedure MnmOper_ExitClick(Sender: TObject);
procedure TlBtnOper_ExitClick(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure CreatGrid();
procedure FormActivate(Sender: TObject);
procedure BtnSearchCardClick(Sender: TObject);
procedure GetDataSGStats(Ms: MParam);
procedure GetDataSGReglam(Ms: MParam);
procedure StatusBarInfoDrawPanel(StatusBar: TStatusBar;
  Panel: TStatusPanel; const Rect: TRect);
procedure StrGrdStatesDrawCell(Sender: TObject; ACol, ARow:
Integer;
  Rect: TRect; State: TGridDrawState);
procedure OpenCrd(Numbr : string);
procedure StrGrdAlarmsDblClick(Sender: TObject);
procedure StrGrdStatesDblClick(Sender: TObject);
procedure StrGrdSearchDblClick(Sender: TObject);
procedure StrGrdWorkDblClick(Sender: TObject);
procedure StrGrdLossesDblClick(Sender: TObject);
procedure GetInfoCrd(StrGr: TStringGrid );
procedure StrGrdStatesClick(Sender: TObject);
procedure EdSearch_OBKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure EdSearch_OBClick(Sender: TObject);
procedure EdSearch_ModClick(Sender: TObject);
procedure StrGrdAlarmsClick(Sender: TObject);
procedure StrGrdWorkClick(Sender: TObject);
procedure StrGrdLossesClick(Sender: TObject);
procedure BtnSearchClick(Sender: TObject);
procedure MnmOper_StatClick(Sender: TObject);
procedure StrGrdSearchClick(Sender: TObject);
procedure StrGrdAlarmsDrawCell(Sender: TObject; ACol, ARow:
Integer;
  Rect: TRect; State: TGridDrawState);
procedure StrGrdSearchDrawCell(Sender: TObject; ACol, ARow:
Integer;
  Rect: TRect; State: TGridDrawState);
procedure StrGrdLossesDrawCell(Sender: TObject; ACol, ARow:
Integer;
  Rect: TRect; State: TGridDrawState);
procedure FormCreate(Sender: TObject);
procedure StBar_TmStDrawPanel(StatusBar: TStatusBar;
  Panel: TStatusPanel; const Rect: TRect);
procedure MnmOper_closeClick(Sender: TObject);
procedure MnmOper_All_Alm_UpClick(Sender: TObject);
procedure MnmOper_Open_AlmClick(Sender: TObject);
procedure VnimanieClick(Sender: TObject);
procedure Clear_StrgrClick(Sender: TObject);
procedure Tm_AllocTimer(Sender: TObject);
procedure BitBtn1Click(Sender: TObject); // открытие формы
тревоги

private
  { Private declarations }
public
  { Public declarations }
end;
```

Продолжение приложения А

```
var
  Frm_Oper: TFrm_Oper;  str_dt, OperStr,s: String;
  OperId,  NWrk, NAlrm, NSt, NLS: Integer;
  Mas_SData: MParam;
  NbrFocusRow, NbrFocusCol : Integer;
  RGB_Alrm,RGB_State,RGB_Schm, RGB_Loss, RGB_Wrk :TColor;

implementation

uses Unit_Iden, DB, StrUtils, Stat, Reaction, EKC;

{$R *.dfm}

procedure TFrm_Oper.MnmOper_CnctClick(Sender: TObject);
var
  i: integer;
begin
  i := ShellExecute(handle, nil,'Level_S', nil, nil, SW_SHOW);
  if i <= 32 then
    MessageBox(Frm_Oper.Handle,'Level_S',
      'Не найден файл',
      MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);

end;

procedure TFrm_Oper.TlBtnOper_CnctClick(Sender: TObject);
begin
  MnmOper_Cnct.Click;
end;

procedure TFrm_Oper.MnmOper_ExitClick(Sender: TObject);
begin
  Self.Hide;
  Frm_Iden.Show;
end;

procedure TFrm_Oper.TlBtnOper_ExitClick(Sender: TObject);
begin
  // MnmOper_Exit.Click;
end;

procedure TFrm_Oper.FormShow(Sender: TObject);
var
  RegIni: TRegIniFile;
  Left_Frm, Top_Frm, Width_Frm, Height_Frm: String;
begin
  RegIni := TRegIniFile.Create('Software\\IntelM');
  try
    RegIni.RootKey := HKEY_LOCAL_MACHINE;
    RegIni.OpenKey('Software', true);
    RegIni.OpenKey('IntelM', true);
    Left_Frm := RegIni.ReadString('SizeW', 'Lf', ' ');
    Top_Frm := RegIni.ReadString('SizeW', 'Tp', ' ');
    Width_Frm := RegIni.ReadString('SizeW', 'Wd', ' ');
```

Продолжение приложения А

```
Height_Frm := RegIni.ReadString('SizeW', 'Hg', ' ');
  finally

  RegIni.Free;
  end;
  Left := StrToInt(Left_Frm);
  Top := StrToInt(Top_Frm);
  Width := StrToInt(Width_Frm);
  Height := StrToInt(Height_Frm);
end;
procedure TFrm_Oper.FormClose(Sender: TObject; var Action:
TCloseAction);
var RegIni: TRegIniFile;
begin
  RegIni := TRegIniFile.Create('Software\\IntelM');
  RegIni.RootKey := HKEY_LOCAL_MACHINE;
  RegIni.OpenKey('Software', true);
  RegIni.OpenKey('IntelM', true);
  RegIni.WriteString('SizeW', 'Lf', IntToStr(Left));
  RegIni.WriteString('SizeW', 'Tp', IntToStr(Top));
  RegIni.WriteString('SizeW', 'Wd', IntToStr(Width));
  RegIni.WriteString('SizeW', 'Hg', IntToStr(Height));
  RegIni.Free;
end;

procedure TFrm_Oper.CreatGrid();
begin
  // ----- Поиск -----
  StrGrdSearch.Cells[0,0] := 'Дата';
  StrGrdSearch.Cells[1,0] := 'Время';
  StrGrdSearch.Cells[2,0] := 'Объект';
  StrGrdSearch.Cells[3,0] := 'Событие';
  StrGrdSearch.ColWidths[0] := 67;
  StrGrdSearch.ColWidths[1] := 55;
  StrGrdSearch.ColWidths[2] := 50;
  StrGrdSearch.ColWidths[3] := 300;
  StrGrdSearch.RowCount := 2;
  // ----- Регламент -----
  StrGrdWork.Cells[0,0] := 'Дата';
  StrGrdWork.Cells[1,0] := 'Время';
  StrGrdWork.Cells[2,0] := 'Модуль';
  StrGrdWork.Cells[3,0] := 'Объект';
  StrGrdWork.Cells[4,0] := 'Событие';
  StrGrdWork.ColWidths[0] := 70;
  StrGrdWork.ColWidths[1] := 70;
  StrGrdWork.ColWidths[2] := 90;
  StrGrdWork.ColWidths[3] := 90;
  StrGrdWork.ColWidths[4] := 350;
  StrGrdWork.RowCount := 2;
  NWrk := 0;
  // ----- Тревоги -----
  StrGrdAlarms.Cells[0,0] := 'Дата';
  StrGrdAlarms.Cells[1,0] := 'Время';
  StrGrdAlarms.Cells[2,0] := 'Объект';
  StrGrdAlarms.Cells[3,0] := 'Событие';
  StrGrdAlarms.Cells[4,0] := 'Раздел';
  StrGrdAlarms.Cells[5,0] := 'Зона';
  StrGrdAlarms.Cells[6,0] := 'ЕКЦ';
  StrGrdAlarms.ColWidths[0] := 75;
```


Продолжение приложения А

```
StrGrdAlarms.ColWidths[1] := 65;
StrGrdAlarms.ColWidths[2] := 65;
StrGrdAlarms.ColWidths[3] := 350;
StrGrdAlarms.ColWidths[4] := 50;
StrGrdAlarms.ColWidths[5] := 40;
StrGrdAlarms.ColWidths[6] := 90;
StrGrdAlarms.ColWidths[7] := -1;
StrGrdAlarms.RowCount := 2;
NAlrm := 0;
// ----- Тревоги 220-----
StrGrdAlarms_220.Cells[0,0] := 'Дата';
StrGrdAlarms_220.Cells[1,0] := 'Время';
StrGrdAlarms_220.Cells[2,0] := 'Объект';
StrGrdAlarms_220.Cells[3,0] := 'Событие';
StrGrdAlarms_220.Cells[4,0] := 'Раздел';
StrGrdAlarms_220.Cells[5,0] := 'Зона';
StrGrdAlarms_220.Cells[6,0] := 'ЕКЦ';
StrGrdAlarms_220.ColWidths[0] := 75;
StrGrdAlarms_220.ColWidths[1] := 65;
StrGrdAlarms_220.ColWidths[2] := 65;
StrGrdAlarms_220.ColWidths[3] := 350;
StrGrdAlarms_220.ColWidths[4] := 50;
StrGrdAlarms_220.ColWidths[5] := 40;
StrGrdAlarms_220.ColWidths[6] := 90;
StrGrdAlarms_220.ColWidths[7] := -1;
StrGrdAlarms_220.RowCount := 2;

// ----- СОСТОЯНИЯ -----
StrGrdStates.Cells[0,0] := 'Дата';
StrGrdStates.Cells[1,0] := 'Время';
StrGrdStates.Cells[2,0] := 'Модуль';
StrGrdStates.Cells[3,0] := 'Объект';
StrGrdStates.Cells[4,0] := 'Событие';
StrGrdStates.ColWidths[0] := 70;
StrGrdStates.ColWidths[1] := 70;
StrGrdStates.ColWidths[2] := 90;
StrGrdStates.ColWidths[3] := 90;
StrGrdStates.ColWidths[4] := 420;
StrGrdStates.ColWidths[5] := -1;
StrGrdStates.DoubleBuffered:=true;
//StrGrdStates.RowCount := 2;

// ----- Потери -----
StrGrdLosses.Cells[0,0] := 'Модуль';
StrGrdLosses.Cells[1,0] := 'Объект';
StrGrdLosses.Cells[2,0] := 'Последнее время';
StrGrdLosses.ColWidths[0] := 90;
StrGrdLosses.ColWidths[1] := 90;
StrGrdLosses.ColWidths[2] := 150;
// StrGrdLosses.RowCount := 6;
NLS := 0;
end;
procedure TFrm_Oper.FormActivate(Sender: TObject);
begin
DtTmPckSearch_Bg.Date:=Now;
DtTmPckSearch_End.Date:=Now;
StatusBarInfo.Panels[0].Text := DateToStr(Date);
StatusBarInfo.Panels[4].Text := 'Оператор ' + OperStr;
StatusBarInfo.Refresh;
StBar_TmSt.Refresh;
```

```

    NbrFocusRow:=0;
    NbrFocusCol:=0;
    StatusBarInfo.Panels[2].Text:=IntToStr(AllocMemCount);
    StatusBarInfo.Panels[4].Text:=IntToStr(AllocMemSize);
StatusBarInfo.Refresh;
    EdSearch_OB.Clear;
    EdSearch_OB.SetFocus;
end;

procedure TFrm_Oper.BtnSearchCardClick(Sender: TObject);
var str_sql: String;

begin
    str_sql:= selectIdCrd+aps+EdSearch_OB.Text+aps;
    DM.DataModule1.GetRztNotNullBD(str_sql,'CardId',Key_Regim);
    if Key_Regim=True then begin
        DM.DataModule1.GetValIntFromField(str_sql,'CardId',Key_IdCard);
        FrmCrd.GetFullData(Key_IdCard);
        FrmCrd.ShowModal;
    end
    else begin
        ShowMessage('Карточки с номером '+EdSearch_OB.Text+' не
существует');
        EdSearch_OB.Clear;
        EdSearch_OB.SetFocus;
    end;
end;

end;

procedure TFrm_Oper.GetDataSGStats(Ms: MParam);
var k:integer;
begin
    k:=StrGrdStates.RowCount;
    StrGrdStates.RowCount :=StrGrdStates.RowCount +1;
    while k>=1 do begin
        MoveRow(StrGrdStates,k-1,k);
        k:=k-1;
    end;
    StrGrdStates.Cells[0,1] := Ms.Prm1;
    StrGrdStates.Cells[1,1] := Ms.Prm8;
    StrGrdStates.Cells[2,1] := Ms.Prm2;
    StrGrdStates.Cells[3,1] := Ms.Prm3;
    StrGrdStates.Cells[4,1] := Ms.Prm4;
    StrGrdStates.Cells[5,1] := Ms.Prm10;
end;

procedure TFrm_Oper.GetDataSGReglam(Ms: MParam);
begin
    inc(NWrk);
    StrGrdWork.RowCount := NWrk + 1;
    StrGrdWork.Cells[0,NSt] := Ms.Prm1;
    StrGrdWork.Cells[1,NSt] := Ms.Prm8;
    StrGrdWork.Cells[2,NSt] := Ms.Prm2;
end;

procedure TFrm_Oper.StatusBarInfoDrawPanel(StatusBar: TStatusBar;
Panel: TStatusBarPanel; const Rect: TRect);
begin
    if DM.DataModule1.ZCnctBD.Connected=true then begin
        if Panel = StatusBarInfo.Panels[1] then begin
            with StatusBarInfo.Canvas do begin

```

Продолжение приложения А

```
    case Unit_Iden.stat_bar of
      0: Brush.Color := clRed;
      1: Brush.Color := clGreen;
    end;
    FillRect(Rect);
    TextOut(Rect.Left, Rect.Top, 'БАЗА');
  end;
end;
end
else begin
  if Panel = StatusBarInfo.Panels[1] then begin
    with StatusBarInfo.Canvas do begin
      case Unit_Iden.stat_bar of
        0: Brush.Color := clGreen;
        1: Brush.Color := clRed;
      end;
      FillRect(Rect);
      TextOut(Rect.Left, Rect.Top, 'БАЗА');
    end;
  end;
end;
end;
end;

procedure TFrm_Oper.StrGrdStatesDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);

begin
  Unit_Strgr.StrGrd_CdTpE_DrawCell(Sender, ACol, ARow, Rect, State);

end;

procedure TFrm_Oper.OpenCrd(Numbr : string);
var str_sql: string;
begin
  str_sql:=selectIdCrd+aps+Numbr+aps;
  DM.DataModule1.GetValIntFromField(str_sql, 'CardId', Key_IdCard);
  Key_Regim:=True;
  FrmCrd.GetFullData(Key_IdCard);
  FrmCrd.ShowModal;
end;

procedure TFrm_Oper.StrGrdAlarmsDblClick(Sender: TObject);
var
  Alr :TReactionForm;
begin
  try
    if
      TStringGrid(Sender).Cells[TStringGrid(Sender).Col, TStringGrid(Sender)
        .Row]<>' ' then begin
        Alr:=TReactionForm.Create(Application);
        Alr.Tag:=StrToInt(TStringGrid(Sender).Cells[7, TStringGrid(Sender).Row
          ]);
        Alr.ShowModal;
      end;
    except
      MessageBox(Frm_Oper.Handle, PChar('Тревога не найдена.'),
        'Ошибка поиска тревоги',
        MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    end;
  end;
end;
```

Продолжение приложения А

```
end;
end;

procedure TFrm_Oper.StrGrdStatesDbClick(Sender: TObject);
begin
  OpenCrd(StrGrdStates.Cells[3,StrGrdStates.Row]);
end;

procedure TFrm_Oper.StrGrdSearchDbClick(Sender: TObject);
begin
  OpenCrd(StrGrdSearch.Cells[2,StrGrdSearch.Row]);
end;

procedure TFrm_Oper.StrGrdWorkDbClick(Sender: TObject);
begin
  OpenCrd(StrGrdSearch.Cells[3,StrGrdWork.Row]);
end;

procedure TFrm_Oper.StrGrdLossesDbClick(Sender: TObject);
begin
  OpenCrd(StrGrdLosses.Cells[1,StrGrdLosses.Row]);
end;

procedure TFrm_Oper.GetInfoCrd(StrGr: TStringGrid );
var str_sql, Memo_Info :String;
NmOb,Tm,Sob : integer;
begin
  EdSearch_OB.Clear;
  EdSearch_Mod.Clear;
  MemoFeatures.Clear;
  try
    if StrGr=StrGrdLosses then begin
      NmOb:=1; Tm:=2; Sob:=2;
    end;
    if ((StrGr=StrGrdStates) or (StrGr=StrGrdWork)) then begin
      NmOb:=3; Tm:=1; Sob:= 4;
    end;
    if ((StrGr=StrGrdAlarms) or (StrGr=StrGrdSearch) or
(StrGr=StrGrdAlarms_220)) then begin
      NmOb:=2; Tm:=1; Sob:=3;
    end;
    EdSearch_OB.Text:=StrGr.Cells[NmOb,StrGr.Row];
    Memo_Info:='№ ' +StrGr.Cells[NmOb,StrGr.Row] + '
+StrGr.Cells[Tm,StrGr.Row] +
' ' + StrGr.Cells[Sob,StrGr.Row] + #13#10 ;
    str_sql:= selectIdCrd+aps+StrGr.Cells[NmOb,StrGr.Row]+aps;
    DM.DataModule1.GetValIntFromField(str_sql,'CardId',Key_IdCard);
    with DM.DataModule1.ZQueryRd do begin
      Close; SQL.Text := FullInfoCrd+IntToStr(Key_IdCard); Open;
      Memo_Info:=Memo_Info+FieldValues['Objectname']+
'+FieldValues['Phones']+#13#10+
      FieldValues['Info']+#13#10+FieldValues['NameGrp']+'-
'+FieldValues['Callnumber'];
      if FieldValues['NameSctr']<>Null Then
Memo_Info:=Memo_Info + ' '+FieldValues['NameSctr'];
    end;
    MemoFeatures.Lines.Add(Memo_Info);
  except
```

Продолжение приложения А

```
    end;
end;

procedure TFrm_Oper.StrGrdStatesClick(Sender: TObject);
begin
  GetInfoCrd(StrGrdStates);
end;

procedure TFrm_Oper.EdSearch_OBKeyDown(Sender: TObject; var Key:
Word;
  Shift: TShiftState);
begin
  if KEY = VK_RETURN then  BtnSearchCardClick(BtnSearchCard);
end;

procedure TFrm_Oper.EdSearch_OBClick(Sender: TObject);
begin
  EdSearch_Mod.Clear;
end;

procedure TFrm_Oper.EdSearch_ModClick(Sender: TObject);
begin
  EdSearch_OB.Clear;
end;

procedure TFrm_Oper.StrGrdAlarmsClick(Sender: TObject);
begin
  if
    (TStringGrid(Sender).Cells[TStringGrid(Sender).Col,TStringGrid(Sender)
    ].Row]<>'') then begin
    if
      (TStringGrid(Sender).Cells[TStringGrid(Sender).Col,TStringGrid(Sender)
      ].Row]<> 'Объект') then begin
      if (TStringGrid(Sender).Row>0) then
        GetInfoCrd(TStringGrid(Sender));
      end;
    end;
  end;
  NbrFocusRow:= TStringGrid(Sender).Row;
  NbrFocusCol:= TStringGrid(Sender).Col;

end;

procedure TFrm_Oper.StrGrdWorkClick(Sender: TObject);
begin
  GetInfoCrd(StrGrdWork);
end;

procedure TFrm_Oper.StrGrdLossesClick(Sender: TObject);
begin
  GetInfoCrd(StrGrdLosses);
end;

procedure TFrm_Oper.BtnSearchClick(Sender: TObject);
var str_sql, Val_Mod:String;
  NSrch : integer;
begin
  NSrch:=0;
  Stat.FrmStat.ClearStrGr(StrGrdSearch);
  if EdSearch_OB.Text<>' ' then begin
```

Продолжение приложения А

```
        ' Cards ON Cards.Unitnumber=Modul.Idmod '+'
        ' WHERE CardNumber ='''+EdSearch_OB.Text+'''+
        ' AND Cards.Unitnumber=Modul.Idmod ;';
    end
    else begin
        str_sql:='SELECT NumberMod FROM Modul WHERE NumberMod=
'+EdSearch_Mod.Text;
    end;
    try
        try
            DM.DataModule1.GetValStrFromField(str_sql,'NumberMod',Val_Mod);
        except
            ShowMessage('Введенного модуля и объекта не существует');
        end;
    finally
        str_sql:='';
    end;
    str_sql:='SELECT Dat,DiscE FROM KTMMMSG '+'
        'LEFT JOIN Enames on Enames.Code=KTMMMSG.Msg '+'
        ' WHERE Modul = '+Val_Mod+
        ' and ( Dat BETWEEN '''+FormatDateTime('yyyy-MM-dd',
DtTmPckSearch_Bg.Date) +
        ' 00:00:00.000'+'' and '''+ FormatDateTime('yyyy-MM-dd',
DtTmPckSearch_End.Date) +
        ' 23:59:59.999'+'' ) ORDER BY Dat DESC';
    try
        with DM.DataModule1.ZQueryRd do begin
            Close;SQL.Text:=str_sql;
            Open; First;
            while not Eof do begin
                Inc(NSrch);
                StrGrdSearch.RowCount:=StrGrdSearch.RowCount+1;
            end;
        end;
        StrGrdSearch.Cells[0,NSrch]:=Copy(FieldValues['Dat'],1,10);
        StrGrdSearch.Cells[1,NSrch]:=Copy(FieldValues['Dat'],11,9);
        StrGrdSearch.Cells[2,NSrch]:=EdSearch_OB.Text;
        StrGrdSearch.Cells[3,NSrch]:=FieldValues['DiscE'];
        Next;
    end;
end;
except
end;
end;

procedure TFrm_Oper.MnmOper_StatClick(Sender: TObject);
begin
    FrmStat.Show;
end;

procedure TFrm_Oper.StrGrdSearchClick(Sender: TObject);
begin
    GetInfoCrd(StrGrdSearch);
end;

procedure TFrm_Oper.StrGrdAlarmsDrawCell(Sender: TObject; ACol,
    ARow: Integer; Rect: TRect; State: TGridDrawState);
var
```

Продолжение приложения А

```
begin
  with TStringGrid(Sender) do begin
    if ARow>0 then begin
      Canvas.Font := Font;
      Canvas.Font.Color := clWindowText;
      Canvas.Brush.Color := clWindow;
      if Cells[6,ARow]='не подтвержденная' then Canvas.Brush.Color
:= clRed;
      if Cells[6,ARow]='подтвержденная' then Canvas.Brush.Color :=
RGB(250,128,114);
      if Cells[6,ARow]='передано в ЕКС' then Canvas.Brush.Color :=
RGB(147,112,219);
      if Cells[6,ARow]='подтвержденная сервером ЕКС' then
Canvas.Brush.Color := RGB(147,112,219);
      if Cells[6,ARow]='подтвержденная оператором ЕКС' then
Canvas.Brush.Color := RGB(238,238,0);
      if Cells[6,ARow]='отрабонанная операторм ЕКС' then
Canvas.Brush.Color := RGB(107,142,35);
      if Cells[6,ARow]='снятая с охр.' then Canvas.Brush.Color :=
RGB(0,255,0);
      Canvas.FillRect(Rect);
      StrPCopy(Buf, Cells[ACol, ARow]);
      DrawText(Canvas.Handle, Buf, -1, Rect,
DT_SINGLELINE or DT_VCENTER or DT_NOCLIP or DT_LEFT);
    end; // if
  end; //with
end;
```

```
procedure TFrm_Oper.StrGrdSearchDrawCell(Sender: TObject; ACol,
ARow: Integer; Rect: TRect; State: TGridDrawState);
var
  Buf: array[byte] of char;
begin
  with TStringGrid(Sender) do begin
    if ARow>0 then begin
      Canvas.Font := Font;
      if (ARow mod 2) = 0 then
        Canvas.Brush.Color := clMoneyGreen
      else Canvas.Brush.Color := clAqua;
      Canvas.Font.Color := clWindowText;
      Canvas.FillRect(Rect);
      StrPCopy(Buf, Cells[ACol, ARow]);
      DrawText(Canvas.Handle, Buf, -1, Rect,
DT_SINGLELINE or DT_VCENTER or DT_NOCLIP or DT_LEFT);
    end;
  end;
end;
```

```
procedure TFrm_Oper.StrGrdLossesDrawCell(Sender: TObject; ACol,
ARow: Integer; Rect: TRect; State: TGridDrawState);
var Buf: array[byte] of char;
begin
```

Продолжение приложения А

```
with TStringGrid(Sender) do begin
    if (ARow mod 2) = 0 then
        Canvas.Brush.Color := RGB_Loss
    else Canvas.Brush.Color := RGB_Schm;
    Canvas.FillRect(Rect);
    StrPCopy(Buf, Cells[ACol, ARow]);
    DrawText(Canvas.Handle, Buf, -1, Rect,
        DT_SINGLELINE or DT_VCENTER or DT_NOCLIP or DT_LEFT);
    end;
end;

end;

procedure TFrm_Oper.FormCreate(Sender: TObject);
begin

    EKC.DataModule2.Read_Losses(StrGrdLosses);
    RGB_Loss:= RGB(255,228,196);
    RGB_State:= RGB(141,238,238);
    RGB_Schm:= RGB(191,239,255);
    RGB_Wrk:= RGB(224,238,238);
    Frm_Oper.DoubleBuffered:=true;
    StrGrdAlarms.DoubleBuffered:=true;
    MemoFeatures.DoubleBuffered:=true;
    StrGrdSearch.DoubleBuffered:=true;
    StrGrdWork.DoubleBuffered:=true;
    StrGrdStates.DoubleBuffered:=true;
    StrGrdLosses.DoubleBuffered:=true;
    StrGrdAlarms_220.DoubleBuffered:=true;
    Pn_BD.Color:=clGreen;
    Pn_Srv.Color:=clGreen;
    CreatGrid();
    StrGrdStates.RowCount := 2;
    StrGrdAlarms.RowCount:=300;
    StrGrdAlarms_220.RowCount:=300;

end;

procedure TFrm_Oper.StBar_TmStDrawPanel(StatusBar: TStatusBar;
    Panel: TStatusPanel; const Rect: TRect);
begin
    if Panel = StBar_TmSt.Panels[1] then begin
        with StBar_TmSt.Canvas do begin
            Brush.Color:=clMoneyGreen;
            Font.Color:=clBlack;
            Font.Size:=18;
            FillRect(Rect);
            TextOut(Rect.Left, Rect.Top,Copy(str_dt,10,8));
        end;
    end;
end;

procedure TFrm_Oper.MnmOper_closeClick(Sender: TObject);
begin
    Self.Hide;
end;
```


Продолжение приложения А

```
Frm_Iden.Show;
end;

procedure TFrm_Oper.MnmOper_All_Alrm_UpClick(Sender: TObject);
var str_sql : string;
begin
try
str_sql:= ' UPDATE ALARME SET ConfirmTime = CURRENT_TIMESTAMP(),
' +
          ' UserID = ' + IntToStr(OperId) +
          ' , State = State + 4 ' +
          ' WHERE State < 256 and ConfirmTime is null';
DM.DataModule1.ChangeRcdBD(str_sql);
except
end;
end;

procedure TFrm_Oper.MnmOper_Open_AlrmClick(Sender: TObject);
var
i: integer;
begin
i := ShellExecute(handle, nil, 'Alarm_Info', nil, nil, SW_SHOW);
if i <= 32 then
MessageBox(Frm_Oper.Handle, 'Alarm_Info',
'Не найден файл',
MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
end;

procedure TFrm_Oper.VnimanieClick(Sender: TObject);
var r,c :integer;
begin
r:= StrGrdStates.Row;
c:=StrGrdStates.Col;
DeleteRow(StrGrdStates,StrGrdStates.Row);
SetGridFocus(StrGrdStates,r,c);
end;

procedure TFrm_Oper.Clear_StrgrClick(Sender: TObject);
begin
Clear_All_StrGr(StrGrdStates);
end;

end.
unit Reaction;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
Dialogs, StdCtrls, Buttons, ExtCtrls, EKC, DM, Unit_Oper;

type
TReactionForm = class(TForm)
lbTechInfo: TLabel;
PanelObject: TPanel;
lbObject: TLabel;
lbState: TLabel;
end;
```

Продолжение приложения А

```
lbSec: TLabel;
  mInfo: TMemo;
  stCallSign: TStaticText;
  stState: TStaticText;
  PanelResponsible: TPanel;
  Label1: TLabel;
  lbResponsible: TLabel;
  Label2: TLabel;
  mAddress: TMemo;
  gbAlarm: TGroupBox;
  lbAlarm: TLabel;
  lbTime: TLabel;
  lbZoneID: TLabel;
  mZoneInfo: TMemo;
  btnHide: TBitBtn;
  gbReason: TGroupBox;
  lbReason: TListBox;
  btnProcess: TBitBtn;
  btnSend: TBitBtn;
  btnConfirm: TBitBtn;
  procedure btnHideClick(Sender: TObject);
  procedure Fill_Data();
  procedure FormCreate(Sender: TObject);
  procedure FormActivate(Sender: TObject);
  procedure btnConfirmClick(Sender: TObject);
  procedure lbReasonClick(Sender: TObject);
  procedure FillVidAlarm();
  procedure btnProcessClick(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  ReactionForm: TReactionForm;
  Al_ID : string;
const
  infoCard = ' SELECT * FROM Cards Left Join GrpUp ON
Cards.Callsign = ' +
              ' GrpUp.IdGrp LEFT JOIN Sector s ON '+'
              ' s.IdSctr=Cards.Technonumber '+'
              ' Left Join ORG ON ORG.IdOrg=Cards.Usmen ' +
              ' Left Join Sections on Cards.CardId =
Sections.IdCard '+'
              ' left join Zones on Sections.IdSection =
Zones.SectionID '+'
              ' left join ALARME on ALARME.ZoneID = Zones.ZoneID '
+
              ' WHERE ALARME.AlarmID = ';

implementation

uses DB, Grids;

{$R *.dfm}

procedure TReactionForm.btnHideClick(Sender: TObject);
```

Продолжение приложения А

```
begin
Close;
end;

procedure TReactionForm.Fill_Data();
var str_sql,ssection :string;
begin
  with DM.DataModule1.ZQueryRd do begin
    Close; SQL.Text :=infoCard+Al_ID;
    Open; First;
    Self.lbObject.Caption:= FieldValues['Name']+' - '+'
Trim(FieldValues['Cardnumber'])+Trim(FieldValues['Objectname']);
    Self.lbSec.Caption:=' Раздел №'+FieldValues['SectionName'];
    Self.lbZoneID.Caption:='Номер зоны:
'+FieldValues['ZoneNumber'];
    Self.lbPassword.Caption:='Контрольное слово:
'+FieldValues['NPassword']+
  ' Числовой пароль: ' +FieldValues['SPassword'];
    mInfo.Lines.Text:=Trim((FieldValues['Particularity']));
    mInfo.Lines.Add(Trim(FieldValues['InfoSct']));
    mInfo.Lines.Add(Trim(FieldValues['SInfo']));
    Self.mAddress.Text:=Trim(FieldValues['Info']);
    if FieldValues['NameSctr']<>Null then
Self.lbTechInfo.Caption:=Trim(FieldValues['NameSctr']);
    self.lbState.Caption := 'Группа';
    Self.stCallSign.Caption:= FieldValues['NameGrp']+'-
'+FieldValues['Callnumber'];
    self.lbTime.Caption := 'Время и дата тревоги: ' +
Trim(FieldValues['ReceiveTime']);
    self.mZoneInfo.Text := Trim(FieldValues['ZInfo']);
    case StrToInt(FieldValues['State']) of
      5:
        ssection := 'Снят ' + FieldValues['StateDate'];
      4:
        ssection := 'Поставлен ' + FieldValues['StateDate'];
    else
        ssection := 'Состояние раздела неопределено!';
    end; //end case
    Self.stState.Caption:=ssection;
  end;
end;

procedure TReactionForm.FormCreate(Sender: TObject);
begin
//Self.Tag:=StrToInt(Frm_Oper.StrGrdAlarms.Cells[7,Frm_Oper.StrGrdAla
rms.Row]);
FormStyle:=fsStayOnTop;
FillVidAlarm;
end;

procedure TReactionForm.FormActivate(Sender: TObject);
var str_sql, FN: String;
    C_yes :Boolean;
begin
//Al_ID:=Frm_Oper.StrGrdAlarms.Cells[7,Frm_Oper.StrGrdAlarms.Row];
  Al_ID:=IntToStr(Self.Tag);
  Fill_Data;
  str_sql:=' SELECT ConfirmTime FROM ALARME WHERE AlarmID = ' +
```

Продолжение приложения А

```
Al_ID+' LIMIT 1';
  FN:='ConfirmTime';
DM.DataModule1.GetRztNotNullBD(str_sql, FN, C_yes);
  if C_yes=false then begin
    btnProcess.Enabled:=false;
    lbReason.Enabled:=False;
  end
  else btnConfirm.Enabled:=false;

end;

procedure TReactionForm.btnConfirmClick(Sender: TObject);
var str_sql : string;
begin
  try
    str_sql:= ' UPDATE ALARME SET ConfirmTime = CURRENT_TIMESTAMP(),
' +
            ' UserID = ' + IntToStr(OperId) +
            ' , State = State + 4 ' +
            ' WHERE AlarmID = ' + Al_ID;
    DM.DataModule1.ChangeRcdBD(str_sql);
    btnConfirm.Enabled:=false;
    lbReason.Enabled:=true;
  except
    str_sql := 'Тревога не найдена. Откройте окно подтверждения
тревоги еще раз!';
    MessageBox(Frm_Oper.Handle, PChar(str_sql),
      'Ошибка поиска тревоги',
      MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    Close;
  end;
end;

procedure TReactionForm.lbReasonClick(Sender: TObject);
begin
  try
    if (lbReason.ItemIndex < -1 ) then btnProcess.Enabled:=false
    else btnProcess.Enabled:=true;

  except
  end;
end;

procedure TReactionForm.FillVidAlarm();
var str_sql :string;
begin
  str_sql:=' SELECT Name FROM REASONS WHERE 1';
  with DM.DataModule1 do begin
    ZQueryRd.Close; ZQueryRd.SQL.Text := str_sql;
    ZQueryRd.Open; ZQueryRd.First;
    lbReason.Clear;
    While not ZQueryRd.Eof do begin
      lbReason.Items.Add(ZQueryRd.FieldValues['Name']);
      ZQueryRd.Next;
    end;
  end;
end;

procedure TReactionForm.btnProcessClick(Sender: TObject);
```

Продолжение приложения А

```
var str_sq : string;
    idReas : integer;
begin
if (lbReason.ItemIndex < -1) then begin
    MessageBox(Frm_Oper.Handle, PChar('Выберите вид тревоги'),
        'Ошибка поиска тревоги',
        MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    end
else begin
    try
        str_sq:= ' SELECT IDReas FROM REASONS WHERE Name = '''+
            lbReason.Items[lbReason.ItemIndex] +'''';
        DM.DataModule1.GetValIntFromField(str_sq, 'IDReas', idReas);
    except
        MessageBox(Frm_Oper.Handle, PChar('Ошибка при выборе
тревоги'),
            'Ошибка ',
            MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
    end;
    try
        str_sq:=' UPDATE ALARME SET ProcessTime = CURRENT_TIMESTAMP(),
' +
            ' Reason = ' + IntToStr(idReas) + ', '+
            ' UserID = ' + IntToStr(OperId) +
            ' , State = State + 256 ' +
            ' WHERE AlarmID = ' + Al_ID;
        DM.DataModule1.ChangeRcdBD(str_sq);
    except
        MessageBox(Frm_Oper.Handle, PChar('Тревога не найдена.
Откройте окно подтверждения тревоги еще раз'),
            'Ошибка поиска тревоги',
            MB_SYSTEMMODAL or MB_ICONERROR or MB_OK);
        Close;
    end;
end;
close;
end;

procedure TReactionForm.FormClose(Sender: TObject;
    var Action: TCloseAction);
begin
Action := caFree;
end;

end.

unit DM;

interface

uses
    Windows, SysUtils, Classes, DB, Registry, ZAbstractRODataset,
    ZAbstractDataset,
    ZDataset, ZConnection, ScktComp, Dialogs;

type
    TDataModule1 = class(TDataModule)
        ZCnctBD: TZConnection;
        ZQueryRd: TZQuery;
    end;
```

Продолжение приложения А

```
ZQueryWrt: TZQuery;
  ZQuerySenior: TZQuery;
  ZQueryTp: TZQuery;
  procedure DataModuleCreate(Sender: TObject);
procedure ChangeRcdBD(StrChng:String);
  procedure ReadRcdBD(ReadSQL:String);
  procedure GetValIntFromField (SqlString, FldName: String; Var
RztValInt: Integer);
  procedure GetValFltFromField(SqlString, FldName: String; Var
RztValFlt: Extended);
  procedure GetValStrFromField(SqlString, FldName: String; Var
RztValStr: String);
  procedure GetRztNotNullBD(SqlString, FldName: String; Var Result:
Boolean);
  procedure L_Error_DM(Str_Log:string);
  //True, если результат запроса не Null
  function FunGetRzltNullBD (SqlString:String) : Boolean;

private
  { Private declarations }
public
  { Public declarations }
end;

var
  DataModule1: TDataModule1;
  Str_IP, Str_IP_EKC: String;
  los : String;
implementation

uses Unit_Oper;

{$R *.dfm}

procedure TDataModule1.DataModuleCreate(Sender: TObject);
var Name_BD, User_BD, PaswordBD: String;
  RegIni: TRegIniFile;
  loss : String;
begin
  RegIni := TRegIniFile.Create('Software\\IntelM');
  try
    RegIni.RootKey := HKEY_LOCAL_MACHINE;
    RegIni.OpenKey('Software', true);
    RegIni.OpenKey('IntelM', true);
    Name_BD := RegIni.ReadString('Login', 'NBD', ' ');
    User_BD := RegIni.ReadString('Login', 'UsrBD', ' ');
    PaswordBD := RegIni.ReadString('Login', 'Pwd', ' ');
    Str_IP := RegIni.ReadString('Login', 'PS', ' ');
    Str_IP_EKC := RegIni.ReadString('Login', 'EKC', ' ');
    ZCnctBD.HostName := Str_IP;
    ZCnctBD.Database := Name_BD;
    ZCnctBD.User := User_BD;
    ZCnctBD.Password := PaswordBD;
    Los := RegIni.ReadString('Option', 'Losses', ' ');
  finally
    RegIni.Free;
  end;
end;
```

Продолжение приложения А

```
end;

procedure TDataModule1.ChangeRcdBD(StrChng:String);
begin
try
ZCnctBD.AutoCommit := False;
  ZCnctBD.Connected:=true;
  ZQueryWrt.Close;
  ZQueryWrt.SQL.Text := StrChng;
  ZQueryWrt.ExecSQL;
  ZCnctBD.Commit;
  ZCnctBD.AutoCommit := True;
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.ChangeRcdBD -' + E.Message+'---'+StrChng);
  end;
end;
end;

procedure TDataModule1.ReadRcdBD(ReadSQL:String);
begin
try
  ZCnctBD.AutoCommit := False;
  ZCnctBD.Connected:=true;
  ZQueryRd.Close;
  ZQueryRd.SQL.Text := ReadSQL;
  ZQueryRd.Open;
  ZCnctBD.Commit;
  ZCnctBD.AutoCommit := True;
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.ReadRcdBD -' + E.Message+'---'+ReadSQL);
  end;
end;
end;

procedure TDataModule1.GetValIntFromField(SqlString, FldName: String;
Var RztValInt: Integer);
begin
try
  ZCnctBD.AutoCommit := False;
  ZCnctBD.Connected:=true;
  ZQueryRd.Close;
  ZQueryRd.SQL.Text := SqlString;
  ZQueryRd.Open;
  ZCnctBD.Commit;
  ZCnctBD.AutoCommit := True;
  RztValInt := ZQueryRd.FieldValues[FldName];
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.GetValIntFromField -' + E.Message+'---'+SqlString);
  end;
end;
end;
end;
```

Продолжение приложения А

```
procedure TDataModule1.GetValFltFromField(SqlString, FldName: String;
Var RztValFlt: Extended);
begin
try
    ZCnctBD.AutoCommit := False;
    ZCnctBD.Connected:=true;
ZQueryRd.Close;
    ZQueryRd.SQL.Text := SqlString;
    ZQueryRd.Open;
    ZCnctBD.Commit;
    ZCnctBD.AutoCommit := True;
    RztValFlt := ZQueryRd.FieldValues[FldName];
except
    on E: Exception do begin
        ZCnctBD.Connected:=false;
        L_Error_DM('DM.GetValFltFromField -' + E.Message+'---
'+SqlString);
        end;
end;
end;

procedure TDataModule1.GetValStrFromField(SqlString, FldName: String;
Var RztValStr: String);
begin
try
    ZCnctBD.AutoCommit := False;
    ZCnctBD.Connected:=true;
ZQueryRd.Close;
    ZQueryRd.SQL.Text := SqlString;
    ZQueryRd.Open;
    ZCnctBD.Commit;
    ZCnctBD.AutoCommit := True;
    RztValStr := ZQueryRd.FieldValues[FldName];
except
    on E: Exception do begin
        ZCnctBD.Connected:=false;
        L_Error_DM('DM.GetValStrFromField -' + E.Message+'---
'+SqlString);
        end;
end;
end;

procedure TDataModule1.GetRztNotNullBD(SqlString, FldName: String;
Var Result: Boolean);
begin
try
    ZCnctBD.Connected:=true;
    ReadRcdBD(SqlString);
    if (ZQueryRd.FieldByName(FldName).IsNull = True) then Result :=
False
    else Result := True;
except
    on E: Exception do begin
        ZCnctBD.Connected:=false;
        L_Error_DM('DM.GetRztNotNullBD -' + E.Message+'---
'+SqlString);
        end;
end;
end;
end;
```


Продолжение приложения А

```
procedure TDataModule1.L_Error_DM(Str_Log:string);
var
  F: TextFile;
  FileName, dt, str: String;
begin
  FileName := GetCurrentDir + '\Log_Error';
  dt:=DateToStr(Date); FileName := FileName + dt + '.txt';
  AssignFile(F, FileName);
  if FileExists(FileName) then Append(F)
  else Rewrite(F);
  dt:=dt+' '+TimeToStr(Time);
  str := dt + ': ' + Str_Log;
  WriteLn(F, str);
  CloseFile(F);
end;

function TDataModule1.FunGetRzltNullBD(SqlString : String ) :
Boolean;
begin
try
  ZCnctBD.Connected:=true;
  ReadRcdBD(SqlString);
  //if (ZQueryRd.FieldByName(FldName).IsNull = True) then Result :=
False
  if (ZQueryRd.IsEmpty) then Result := False
  else Result := True;
except
  on E: Exception do begin
    ZCnctBD.Connected:=false;
    L_Error_DM('DM.FunGetRzltNullBD -' + E.Message+'---
'+SqlString);
  end;
end;
end;
end.
```

Приложение Б

```
#DEFINE _POSIX_SOURCE 1 /* POSIX-совместимый код */
#include <QtGui>
#include <QtNetwork>
#include "echoserver.h"
#include "stdlib.h"
#include <qlabel.h>
#include <qdir.h>
#include <qstring.h>
#include <qfile.h>
#include <stdio.h> // Стандартные объявления ввода / вывода
#include <unistd.h> // Объявления стандартных функций UNIX
#include <fcntl.h> // Объявления управления файлами
#include <errno.h> // Объявления кодов ошибок
#include <termios.h> // Объявления управления POSIX-терминалом
#include <QtCore>
#include <QTextStream>
#include <QtDebug>
#include <QVector>
#include <QIODevice>
#include <QColor>
#include <QToolBar>
#include <QGroupBox>
#include <QTime>
#include <QDateTime>
#include <QFileDialog>
#include <QKeySequence>
#include <QAction>

//
EchoServer::EchoServer(QObject *parent)
    : QTcpServer(parent) {}
EchoServer::~EchoServer(){
    QString Ln;
    while (!Sclients.isEmpty()){
        client = Sclients.takeFirst();
        Ln = QString::fromLocal8Bit("-- Сервер отключается!!!--");
    };

    SaveToLogW(Ln);
    client->flush();
    client->close();
    client->deleteLater();
}
//
bool EchoServer::start(){
    QString Ln;
    server = new QTcpServer(this);
    taskCls = new QAction("Stop", this);
    taskCls-> setShortcut(QKeySequence(Qt::ALT + Qt::Key_X));
    server->listen(QHostAddress::Any, 777);
    connect(server, SIGNAL(newConnection()), this,
    SLOT(slotConnect()));
    connect(server, SIGNAL(readReady()), this, SLOT(GetMessage()));
    connect(&threadPak, SIGNAL(currentPak(const QString&)), this,
    SLOT(RunOpros()));
    connect(taskCls, SIGNAL(triggered()), this,
    SLOT(closeEvent()));
    bytes = 0;
    QTimer *ptimer = new QTimer(this);
```

Продолжение приложения В

```
connect(ptimer, SIGNAL(timeout()), SLOT(sendToControl()));
ptimer->start(900);
cntLnPak = -1; cntLnSgn = -1; cntLnDt = -1;
bSendCmd = false; status_port = false;
bool con = taskBD.createConnect();
if(!con){
    Ln = QString::fromLocal8Bit(" Ошибка подключения ");
}
else{
    Ln = QString::fromLocal8Bit(" База данных подключена ");
}
SaveToLogW(Ln);
onInstParam();
RunOpros();
return true;
}
//
void EchoServer::slotConnect()
{
    QString str, LnInfo;
    try{
        client = server->nextPendingConnection();
        connect(client, SIGNAL(disconnected()), client,
SLOT(removeConnection()));
        connect(client,
SIGNAL(readyReady()), this, SLOT(GetMessage()));
        LnInfo = QString::fromLocal8Bit(" Новое соединение от ");
        str = client->peerAddress().toString(); LnInfo += str;
        SaveToLogW(LnInfo);
        SClients.append(client);
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка соединения с
сервером. Процедура -- EchoServer::slotConnect() -- ");
        SaveToLogW(LnInfo);
    }
}
//
void EchoServer::removeConnection()
{
    QString str, LnInfo;
    try{
        LnInfo = QString::fromLocal8Bit(" Клиент ");
        str = client->peerAddress().toString(); LnInfo += str;
        str = QString::fromLocal8Bit(" отключен. "); LnInfo +=
str;

        SaveToLogW(LnInfo);
        SClients.removeAt(SClients.indexOf(client));
        client->deleteLater();
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка отключения от
сервера. Процедура -- EchoServer::removeConnection() -- ");
        SaveToLogW(LnInfo);
    }
}
//
void EchoServer::GetMessage()
```

Продолжение приложения В

```
QByteArray data;
QString LnInfo;
try{
    if(0 < client->bytesAvailable()) {
        bytes += client->bytesAvailable();
        data = client->readAll();
        SaveToLogW(data);
    }
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка сообщения.
Процедура -- EchoServer::GetMessage() -- ");
    SaveToLogW(LnInfo);
}
}
/** Проверка связи с сервером **/
void EchoServer::sendToControl()
{
    QString LnInfo;
    try{
        LnInfo = "S";
        for (int i = 0; i < SClients.size(); ++i){
            GetToClient(SClients.at(i), LnInfo);
        }
        LnInfo = QString::fromLocal8Bit(" Я на связи. Процедура -
- EchoServer::sendToControl() -- ");
        // SaveToLogW(LnInfo);
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Обрыв связи с
сервером!!! Процедура -- EchoServer::sendToControl() -- ");
        SaveToLogW(LnInfo);
    }
}
}
/*Создание лог файла приложения*/
void EchoServer::SaveToLogW(QString StrLog)
{
    QTime tm;
    QFile flSave;
    QTextStream stream( &flSave );
    QStringList lines;
    QString Fname, line, stm;
    Fname = QDir::currentPath() + "/ReadBuf.log";
    tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz "); //
добавил пробел
    line = QDate::currentDate().toString("dd.MM.yy");
    line += ": "; line += stm; line += StrLog;
    flSave.setFileName(Fname);
    lines << line;
    try{
        if(flSave.exists()) flSave.open(QIODevice::Append);
        else flSave.open(QIODevice::ReadWrite);
        for ( QStringList::Iterator it = lines.begin(); it !=
lines.end(); ++it )
            stream << *it << "\n";
        flSave.close();
        if (stream.status() != QTextStream::Ok){
            qDebug() << "Error write file";
        }
    }
}
```

Продолжение приложения В

```
}
    catch(...){
        qDebug() << "Error write file";
    }
}
//
void EchoServer::GetToClient(QTcpSocket *pSocket, QString str)
{
    QString LnInfo, line, stm;
    QTime tm;
    try{
        LnInfo = QString::fromLocal8Bit(" Передача клиенту.
Процедура -- EchoServer::GetToClient() -- ");
//        SaveToLogW(LnInfo);
        tm = tm.currentTime(); stm = tm.toString("hh:mm:ss");
        line = QDate::currentDate().toString("dd.MM.yy");
        line += " "; line += stm; line += " --- "; line += str;
        pSocket->write(line.toStdString().c_str(), line.count());
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка передачи
клиенту. Процедура -- EchoServer::GetToClient() -- ");
        SaveToLogW(LnInfo);
    }
}
//
void EchoServer::onInstParam()
{
    QString Ln, s_sql;
    s_sql = "SELECT * FROM `PORTS`";
    ItmNmPrt = taskBD.GetValIntFromField(s_sql, 1);
    ItmSpdPrt = taskBD.GetValIntFromField(s_sql, 2);
    ItmBtPrt = taskBD.GetValIntFromField(s_sql, 3);
    ItmChtPrt = taskBD.GetValIntFromField(s_sql, 4);
    ItmStpBtPrt = taskBD.GetValIntFromField(s_sql, 5);
    status_port = IniComPort(ItmNmPrt, ItmSpdPrt, ItmBtPrt,
ItmChtPrt, ItmStpBtPrt);
//    status_port = IniComPort(4, 7, 3, 0, 0);
    if(status_port != true) {
        Ln = QString::fromLocal8Bit(" Не удалось открыть порт =
");
        Ln += QString::number(ItmNmPrt);
    }
    else {
        Ln = QString::fromLocal8Bit(" Открыт для записи / чтения
порт = ");
        Ln += QString::number(ItmNmPrt);
        emit getIDPort(threadPak.port = ds_port);
    }
    SaveToLogW(Ln);
}
//
bool EchoServer::IniComPort(int nport, int baud, int datab, int
parity, int stopb)
{
    QString name_port;
    struct termios options;
    try{
// Установка имени порта
        switch(nport){
```

Продолжение приложения В

```
case 0: { name_port = "/dev/ttyS0"; break;      }
        case 1: { name_port = "/dev/ttyS1"; break;      }
        case 2: { name_port = "/dev/ttyS2"; break;      }
        case 3: { name_port = "/dev/ttyS3"; break;      }
        case 4: { name_port = "/dev/ttyS4"; break;      }
        case 5: { name_port = "/dev/ttyS5"; break;      }
        case 6: { name_port = "/dev/ttyUSB0"; break;    }
    }
    ds_port = ::open(name_port.toUtf8(), O_RDWR | O_NOCTTY |
O_NDELAY);
    // ds_port = ::open(name_port.toUtf8(), O_RDWR | O_NOCTTY);
    if(ds_port == -1) {
        ::close(ds_port);
        status_port = false;
    }
    else {
        //bzero(&options, sizeof(options));
        threadPak.port = ds_port;
        fcntl(ds_port, F_SETFL, FNDELAY);
        tcgetattr(ds_port, &options);
    // Установка заданной скорости
        switch(baud){
            case 0: {
                cfsetispeed(&options, B110);
                cfsetospeed(&options, B110);
                break;
            }
            case 1: {
                cfsetispeed(&options, B300);
                cfsetospeed(&options, B300);
                break;
            }
            case 2: {
                cfsetispeed(&options, B600);
                cfsetospeed(&options, B600);
                break;
            }
            case 3: {
                cfsetispeed(&options, B1200);
                cfsetospeed(&options, B1200);
                break;
            }
            case 4: {
                cfsetispeed(&options, B2400);
                cfsetospeed(&options, B2400);
                break;
            }
            case 5: {
                cfsetispeed(&options, B4800);
                cfsetospeed(&options, B4800);
                break;
            }
            case 6: {
                cfsetispeed(&options, B9600);
                cfsetospeed(&options, B9600);
                break;
            }
            case 7: {
                cfsetispeed(&options,
B19200);
                cfsetospeed(&options, B19200);
            }
        }
    }
}
```

Продолжение приложения В

```
break;
}
case 8: {
    cfsetispeed(&options, B38400);
    cfsetospeed(&options, B38400);
    break;
}
case 9: {
    cfsetispeed(&options, B57600);
    cfsetospeed(&options, B57600);
    break;
}
case 10: {
    cfsetispeed(&options, B115200);
    cfsetospeed(&options, B115200);
    break;
}
}
// Установка битов данных
switch(datab) {
case 0: { options.c_cflag |= CS5; break; }
case 1: { options.c_cflag |= CS6; break; }
case 2: { options.c_cflag |= CS7; break; }
case 3: { options.c_cflag |= CS8; break; }
}
// Установка контроля на четность
switch(parity) {
case 0: {
    options.c_cflag &= ~PARENB;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;
    break;
}
case 1: {
    options.c_cflag |= PARENB;
    options.c_cflag &= ~PARODD;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS7;
    break;
}
case 2: {
    options.c_cflag |= PARENB;
    options.c_cflag |= PARODD;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS7;
    break;
}
case 3: {
    options.c_cflag &= ~PARENB;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;
    break;
}
}
// Установка стопового бита
switch(stopb) {
```

Продолжение приложения В

```
        case 0: break;
        case 1: break;
        case 2: { options.c_cflag |= CSTOPB; break; }
    }
    options.c_cflag |= CLOCAL | CREAD;
    options.c_cflag &= ~ CRTSCTS;
    options.c_iflag = IGNPAR;
    options.c_oflag &= ~ OPOST;
    options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
    options.c_cc[VMIN] = 0;
    options.c_cc[VTIME] = 0;
    tcflush(ds_port, TCIFLUSH);
    tcsetattr(ds_port, TCSANOW, &options);
    status_port = true;
}
LnInfo = QString::fromLocal8Bit(" Инициализировали порт.
Процедура -- EchoServer::IniComPort(Param) -- ");
SaveToLogW(LnInfo);
return status_port;
}
catch(...){
LnInfo = QString::fromLocal8Bit("Ошибка в процедуре --
EchoServer::IniComPort(Param) --");
SaveToLogW(LnInfo);
return false;
}
}
/*Запуск потока приема / передачи */
void EchoServer::RunOpros()
{
    QString StrOutPak, line, sTmp, Temp, s_sql, LnInfo, Rep;
    unsigned short _len_test_buf;
    unsigned char _test_buf[16];
    // char print_buf[24]; //Буфер для печати
    struct masParam *Prm;
    struct masData *PrmDt;
    bool FlagSend, FlagPak, FlagLastMsg, FlagBuff, FlagMod;
    int cntP;
    try{
        FlagSend = false; FlagPak = false; FlagBuff= false;
        FlagLastMsg= false;
        FlagMod = false;
        threadPak.start();
        StrOutPak = threadPak.StrPakHex;
        // ----- Начало обработки принятого пакета -----
        _len_test_buf=0;
        _len_test_buf = GetPak(StrOutPak, _test_buf);
        Prm = new struct masParam;
        GetParamData(_test_buf, Prm);
        PrmDt = new struct masData;
        PrintStrData(_test_buf, PrmDt);
        Rep =Prm->v1Pak.mid(3,1);
        SaveToLogW(Rep);
        if (Rep == "9"){
            LnInfo = QString::fromLocal8Bit("Rep=9 ");
            LnInfo+=Prm->v1Pak; Prm->nmMod=(Prm-
            >v1Pak.mid(4,1)).toInt(); Prm->Sgnl= Prm->nmPak; LnInfo +=
            QString::fromLocal8Bit("nmMod = ");

```


Продолжение приложения В

```

LnInfo += QString::number(Prm->nmMod);
LnInfo += QString::fromLocal8Bit("nSgn = ");
LnInfo += QString::number(Prm->Sgn1);
SaveToLogW(LnInfo);
taskBD.GetBufMod(Prm);
}
else{
LnInfo = QString::fromLocal8Bit("Rep=A ");
LnInfo+=Prm->v1Pak;
SaveToLogW(LnInfo);
FlagMod = SendModul(PrmDt);
if (FlagMod != false) { //есть такой модуль??
taskBD.GetBufMod(Prm);
FlagSend = SendEvents(PrmDt);
if(FlagSend != false){ // Есть такое событие??
line = QString::fromLocal8Bit(" FlagSend = True!!! ");
FlagPak = SendNumbrPak(Prm);
if(FlagPak != false){ //Есть такой №pak and Mod??
FlagBuff = DefDataPak(Prm);
} //Есть такой №pak and Mod??
if((FlagPak == false) or ((FlagPak == true) and (FlagBuff
== true)) ) { //if 1
Prm->nData = Prm->nUpd;
taskBD.GetSqlInsertBD_Buffer(Prm);
FlagLastMsg = SendLastMsg(Prm);
if(FlagLastMsg != false){ // Как последний № Pak??
s_sql = "SELECT * FROM `KTMMMSG` WHERE `Modul`
= ";
s_sql += QString::number(Prm->nmMod); s_sql
+=", and `Num`= "; s_sql += QString::number(Prm->nmPak);
s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
cntP = taskBD.GetValIntFromField(s_sql,8);
cntP++; Prm->nCnt = cntP;
taskBD.GetSqlUpdtBD_Msg(Prm);
} // Как послед
else { //else1
taskBD.GetSqlInsertBD_Msg(Prm);
line = PrmDt->v1Evnt; line += '*';
line += QString::number(PrmDt->nmMod);
// Строка принятого события
StrOutPak = line;
for (int i = 0; i < SClients.size();
+i){ //for
GetToClient(SClients.at(i), StrOutPak);
} //FOR
} //else1
} // if1
} //есть событие
} //есть модуль
} //rep
delete Prm;
delete PrmDt;
}
catch(...){
LnInfo = QString::fromLocal8Bit(" Ошибка в RunOpros");
SaveToLogW(LnInfo);
}
} //end// ----- Конец обработки принятого пакета -----

```

```

//Завершение работы
void EchoServer::closeEvent()
{
    QString LnInfo, line, stm;
    QTime tm;
    threadPak.stop();
//    threadPak.wait();
    tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz");
    line = QDate::currentDate().toString("dd.MM.yy");
    line += ": "; line += stm; line += ": ";
    LnInfo = QString::fromLocal8Bit("Завершение работы сервера ");
    line += LnInfo;
    SaveToLogW(line);
    close();
}
/*Формирование пакета */
int EchoServer::GetPak(QString StrInpt, unsigned char *sndBf)
{
    unsigned char buf[16];
    const char *s_wrk;
    QChar tend;
    unsigned short count, k, len;
    unsigned int t_byte;
    QString str_cmd, s_work;
    try{
        str_cmd = StrInpt; count = 0; k = 0;
        while(k < str_cmd.length()){
            tend = str_cmd[k];
            if(tend != ' '){
                s_work = str_cmd.mid(k,2);
                s_wrk = s_work.toLocal8Bit().data();
                sscanf((const char*)s_wrk, "%02X", &t_byte);
                buf[count]=t_byte;
                count++; k = k+2;
            }
            else k++;
        }
        for(int i=0; i<count; i++) sndBf[i] = buf[i];
        len = count;
        return len;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в GetPak");
        SaveToLogW(LnInfo);
        return 0;
    }
}

/*Формирование кода */
QString EchoServer::GetECod(QString StrInpt)
{
    QChar tend;
    unsigned short k;
    QString str_cmd, s_work, sw;
    str_cmd = StrInpt + ' '; s_work = '0'; k = 0;
    while(k < str_cmd.length()-1){
        tend = str_cmd[k];
        if(tend != ' ') {
            sw = str_cmd.mid(k,2);

```

Продолжение приложения В

```
        if (sw.contains(' ') != false) {
            sw = sw.mid(0,1); sw = '0' + sw;
        }
        s_work += sw; k = k + 2;
    }
    else k++;
}
str_cmd = s_work.mid(1,8);
return str_cmd;
}
/*Функция строки данных*/
void EchoServer::PrintStrData(unsigned char *buf, struct masData
*Prd)
{
    QString paket, LnInfo;
    int NbrOB;
    char print_buf[24]; //Буфер для печати
    union{
        unsigned int ob;
        unsigned char c[2];
    } out_Nbr;
    try{
//      LnInfo += QString::fromLocal8Bit(" Структура строки
данных. Процедура -- EchoServer::PrintStrData(Prm) -- ");
//      SaveToLogW(LnInfo);
// ----- Номер модуля -----
        out_Nbr.c[0] = buf[4];
        out_Nbr.c[1] = buf[3];
        NbrOB = out_Nbr.ob;
        Prd->nmMod = NbrOB;
// ----- Событие -----
        ::sprintf(print_buf, "%X", buf[5]);
        paket = QString::fromAscii(print_buf);
        for(int i=6; i<9; i++){
            ::sprintf(print_buf, "%X", buf[i]);
            paket += " " + QString::fromAscii(print_buf);
        }
        LnInfo = GetECod(paket);
        Prd->vlEvnt = LnInfo;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в структуре
строки данных. Процедура -- EchoServer::PrintStrData(Prm) -- ");
        SaveToLogW(LnInfo);
    }
}
/*Функция заполнения структуры данных*/
void EchoServer::GetParamData(unsigned char *buf, struct masParam
*Pr)
{
    int Nbr;
    QTime tm;
    QString paket, stm, nmVld, Ln, LnInfo;
    char print_buf[24]; //Буфер для печати
    union{
        unsigned int sgn;
        unsigned char c[1];
    } out_Data;
    union{
        unsigned int md;
```

Продолжение приложения В

```
        unsigned char c[2];
    } out_Modl;
    try{
//      LnInfo += QString::fromLocal8Bit(" Структура данных.
Процедура -- EchoServer::GetParamData(Prm) -- ");
//      SaveToLogW(LnInfo);
// ----- Номер пакета -----
        out_Data.c[0] = buf[2];
        Nbr = out_Data.sgn;
        Pr->nmPak = Nbr;
// ----- Номер модуля -----
        out_Modl.c[0] = buf[4]; //было buf[3]
        out_Modl.c[1] = buf[3]; //было buf[4]
        Nbr = out_Modl.md;
        Pr->nmMod = Nbr;
// ----- Номер ретранслятора -----
        out_Data.c[0] = buf[12];
        Nbr = out_Data.sgn;
        Pr->nmRpt = Nbr;
// ----- Уровень сигнала -----
        out_Data.c[0] = buf[11];
        Nbr = out_Data.sgn;
        Pr->Sgnl = Nbr;
// ----- Событие -----
        ::sprintf(print_buf, "%X", buf[5]);
        paket = QString::fromAscii(print_buf);
        for(int i=6; i<9; i++){
            ::sprintf(print_buf, "%X", buf[i]);
            paket += " " + QString::fromAscii(print_buf);
        }
        Pr->vlEvnt = paket;
// ----- Весь пакет -----
        ::sprintf(print_buf, "%X", buf[0]);
        paket = QString::fromAscii(print_buf);
        for(int i=1; i<13; i++){
            ::sprintf(print_buf, "%X", buf[i]);
            paket += " " + QString::fromAscii(print_buf);
        }
        Pr->vlPak = paket;
//      SaveToLogW(paket);
// ----- Счетчик повторов -----
        Pr->nCnt = 1;
// ----- Дата и время прихода -----
        tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz");
        //nmVld = QDate::currentDate().toString("dd/MM/yy");
        nmVld = QDate::currentDate().toString("yyyy-MM-dd");

        nmVld += " "; nmVld += stm;
        Pr->nData = nmVld;
//      SaveToLogW(nmVld);
// ----- Дата и время обновления -----
        Pr->nUpd = nmVld;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в структуре
данных. Процедура -- EchoServer::GetParamData(Prm) -- ");
        SaveToLogW(LnInfo);
    }
} /*Заполнение и контроль за пересылкой события*/
```

Продолжение приложения В

```
bool EchoServer::SendEvents(struct masData *Pr)
{
    QString s_sql, LnInfo;
    bool R_yes;
    try{
//      LnInfo = QString::fromLocal8Bit(" Пересылка события.
Процедура -- EchoServer::SendEvents(Prm) -- ");
//      SaveToLogW(LnInfo);
        s_sql = "Select * from ENames where CodE = '";
        s_sql += Pr->vlEvnt; s_sql += "'";
        R_yes = taskBD.GetRztNotNullBD(s_sql);
        return R_yes;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в пересылке
события. Процедура -- EchoServer::SendEvents(Prm) -- ");
        SaveToLogW(LnInfo);
        return false;
    }
}
/*Сравнение дат поступления пакета*/
bool EchoServer::DefDataPak(struct masParam *Pr)
{
    QString str, tm1,tm2, s_sql, LnInfo;
//unsigned long VrOld, VrUpd, VrDef; закоментировал
double VrOld, VrUpd, VrDef; // добавил
    bool R_yes;
    int cntP;
    try{
//      LnInfo = QString::fromLocal8Bit(" Сравнение дат.
Процедура -- EchoServer::DefDataPak(Prm) -- ");
//      SaveToLogW(LnInfo);
//      SaveToLogW(Pr->nData);
//      /*str = Pr->nData.mid(18,4);
SaveToLogW(str);
VrOld = str.toDouble();*/
        s_sql = "Select * from MsgBuffer where NbrPak = ";
            s_sql += " and NbMod = ";
        s_sql += QString::number(Pr->nmMod);
        str = taskBD.GetValStrFromField(s_sql,2);
        tm1 = str.mid(11,2);//LnInfo= QString::fromLocal8Bit("old
tm1= ");LnInfo +=tm1;SaveToLogW(LnInfo);
        tm2 = str.mid(14,2);//LnInfo= QString::fromLocal8Bit("old
tm2= ");LnInfo +=tm2;SaveToLogW(LnInfo);
        str = str.mid(17,5);//LnInfo= QString::fromLocal8Bit("old
second= ");LnInfo +=str; SaveToLogW(LnInfo);
        VrOld = (tm1.toDouble())*3600 + (tm2.toDouble())*60 +
str.toDouble();
//      LnInfo = QString::fromLocal8Bit(" Старое время VrOld =
"); LnInfo += QString::number(VrOld);
//      SaveToLogW(LnInfo);
        str = Pr->nUpd;
        tm1 = str.mid(11,2);//LnInfo= QString::fromLocal8Bit("Up
tm1= ");LnInfo +=tm1;SaveToLogW(LnInfo);
        tm2 = str.mid(14,2);//LnInfo= QString::fromLocal8Bit("Up
tm2= ");LnInfo +=tm2;SaveToLogW(LnInfo);
        str = str.mid(17,5);//LnInfo= QString::fromLocal8Bit("Up
second= ");LnInfo +=str; SaveToLogW(LnInfo);
        VrUpd = (tm1.toDouble())*3600 + (tm2.toDouble())*60 +
```

Продолжение приложения В

```

str.toDouble();
//      LnInfo = QString::fromLocal8Bit(" Новое время VrUpd = ");
LnInfo += QString::number(VrUpd);
//      SaveToLogW(LnInfo);
//      /*if(VrUpd < VrOld){
//          VrOld = 60-VrOld;
//          VrDef = VrUpd + VrOld;
//      }
//      else {
//          VrDef = VrUpd - VrOld;
//      }*/
VrDef = (abs(VrUpd - VrOld));
//      LnInfo = QString::fromLocal8Bit(" Разница VrDef = ");
LnInfo += QString::number(VrDef);
//      SaveToLogW(LnInfo);
//      if (VrDef == 0) {
//          R_yes = true;
//      }
//      else R_yes = false;
//      if (VrDef > 9) {
//          s_sql = "DELETE FROM MsgBuffer where NbrPak = ";
//          s_sql += QString::number(Pr->nmPak);
//          s_sql += " and NbMod = ";
//          s_sql += QString::number(Pr->nmMod); s_sql += ";";
//          SaveToLogW(s_sql);
//          taskBD.ChangeRcdBD(s_sql);
//          R_yes = true;
//      }
//      LnInfo = QString::fromLocal8Bit(" УДАЛИТЬ ИЗ БУФ<9
"); SaveToLogW(LnInfo);
//      }
//      else {
//          R_yes = false;
//          /*s_sql = "Select * from MsgBuffer where NbrPak =
";
//          s_sql += QString::number(Pr->nmPak);
//          s_sql += " and NbMod = ";
//          s_sql += QString::number(Pr->nmMod);*/
//          cntP = taskBD.GetValIntFromField(s_sql, 4);//
изменил на 4
//          LnInfo = taskBD.GetValStrFromField(s_sql, 2);//
изменил на 2
//          Pr->nData = LnInfo;
//          cntP++; Pr->nCnt = cntP;
//          taskBD.GetSqlUpdtBD_Buffer(Pr);
//      }
//      LnInfo = QString::fromLocal8Bit(" ОБНОВЛЕНИЕ БУФЕРА
"); SaveToLogW(LnInfo);
//      }
//      return R_yes;
//      }
//      catch(...){
//          LnInfo = QString::fromLocal8Bit(" Ошибка в сравнении дат.
Процедура -- EchoServer::DefDataPak(Prm) -- ");
//          SaveToLogW(LnInfo);
//          return false;
//      }
//      }
//      /*Заполнение и контроль за пересылкой номера пакета*/
bool EchoServer::SendNumbrPak(struct masParam *Pr)
{      QString s_sql, LnInfo;

```

Продолжение приложения В

```
bool R_yes;
try{
    LnInfo = QString::fromLocal8Bit(" Пересылка номера.
Процедура -- EchoServer::SendNumbrPak(Prm) -- ");
//    SaveToLogW(LnInfo);
    R_yes = false;
    s_sql = "Select * from MsgBuffer where NbrPak = ";
    s_sql += QString::number(Pr->nmPak);
    s_sql += " and NbMod = ";
    s_sql += QString::number(Pr->nmMod); s_sql += ";";
//    SaveToLogW(s_sql);
    R_yes = taskBD.GetRztNotNullBD(s_sql);
    if (R_yes !=false){
//        LnInfo = QString::fromLocal8Bit(" есть такой
пакет"); SaveToLogW(LnInfo);
    }
    else {
//        LnInfo = QString::fromLocal8Bit(" нет такого
пакета"); SaveToLogW(LnInfo);
    }
    return R_yes;
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка в пересылке
номера. Процедура -- EchoServer::SendNumbrPak(Prm) -- ");
    SaveToLogW(LnInfo);
    return false;
}
}
/*Контроль на последнее сообщение*/
bool EchoServer::SendLastMsg(struct masParam *Pr)
{
    QString s_sql, LnInfo;
    bool Flg,R_last;
    int Npkg;
    try{
        LnInfo = QString::fromLocal8Bit(" Процедура --
EchoServer::SendLastMsg(Prm) -- ");
//        SaveToLogW(LnInfo);
        R_last = false; Flg = false;
        s_sql = "SELECT * FROM `KTMSG` WHERE `Modul`= ";
        s_sql += QString::number(Pr->nmMod);
//        s_sql += " and `Num` = ";
//        s_sql += QString::number(Pr->nmPak);
        s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
//        SaveToLogW(s_sql);
        Flg = taskBD.GetRztNotNullBD(s_sql);
//        SaveToLogW(s_sql);
        if(Flg != false){
            Npkg = taskBD.GetValIntFromField(s_sql,3);

            if (Npkg !=Pr->nmPak){
                R_last = false;
            }
            else R_last=true;
        }
    }
    else R_last = false;
    return R_last;
}
catch(...){
```

Продолжение приложения В

```
        LnInfo = QString::fromLocal8Bit(" Ошибка Процедура --
EchoServer::SendLastMsg(Prm) -- ");
        SaveToLogW(LnInfo);
        return false;
    }
}
//
/*Проверка Modul*/
bool EchoServer::SendModul(struct masData *Pr)
{
    QString s_sql, LnInfo;
    bool R_yes;
    try{
//        LnInfo = QString::fromLocal8Bit(" Пересылка события.
Процедура -- EchoServer::SendModul(Prm) -- ");
//        SaveToLogW(LnInfo);
        s_sql = "Select * from Modul where NumberMod = ";
        s_sql += QString::number(Pr->nmMod); s_sql += " ";
        R_yes = taskBD.GetRztNotNullBD(s_sql);
        return R_yes;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в пересылке
события. Процедура -- EchoServer::SendModul(Prm) -- ");
        SaveToLogW(LnInfo);
        return false;
    }
}
//

#include "databd.h"
#include "stdlib.h"
#include <qdir.h>
#include <stdio.h> // Стандартные объявления ввода / вывода
#include <unistd.h> // Объявления стандартных функций UNIX
#include <fcntl.h> // Объявления управления файлами
#include <errno.h> // Объявления кодов ошибок
#include <QtCore>
#include <QTextStream>
#include <QtDebug>
#include <QIODevice>
#include <QTime>
#include <QDateTime>
#include <QFileDialog>
#include <qstring.h>
#include "fileStru.h"
#include "echoserver.h"
//
DataBD::DataBD()
{
    closed = false;
}
/*Создание лог файла */
void DataBD::SaveToLogBD(QString StrLog)
{
    QTime tm;
    QFile flSave; QTextStream stream( &flSave );
```


Продолжение приложения В

```
QStringList lines;
QString FName, line, stm;
FName = QDir::currentPath() + "/WorkBD.log";
tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz");
line = QDate::currentDate().toString("dd/MM/yy");
line += ": "; line += stm + " "; line += StrLog;
flSave.setFileName(FName);
lines << line;
try{
    if(flSave.exists()) flSave.open(QIODevice::Append);
    else flSave.open(QIODevice::ReadWrite);
    for ( QStringList::Iterator it = lines.begin(); it !=
lines.end(); ++it )
        stream << *it << "\n";
    flSave.close();
    if (stream.status() != QTextStream::Ok){
    }
}
catch(...){
    qDebug() << "Error write file";
}
}
// Соединение с БД
bool DataBD::createConnect()
{
    QString LnInfo;
    QSqlDatabase db;
    try{
        LnInfo = QString::fromLocal8Bit(" Соединение с БД. Процедура
-- DataBD::createConnect() -- ");
        SaveToLogBD(LnInfo);
        db = QSqlDatabase::addDatabase("QMYSQL");
        db.setDatabaseName("DBINT");
        db.setUserName("boris");
        db.setPassword("asol123");
        if (!db.open()){
            return false;
        }
        closed = true;
        return true;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при соединении с
БД. Процедура -- DataBD::createConnect() -- ");
        SaveToLogBD(LnInfo);
        return false;
    }
}
// Выполнение команды в БД
void DataBD::ChangeRcdBD(QString StrChng)
{
    QSqlQuery query;
    QString LnInfo;
    try{
        // LnInfo = QString::fromLocal8Bit(" Выполнение команды в БД.
Процедура -- DataBD::ChangeRcdBD(Prm) -- ");
        // SaveToLogBD(LnInfo);
        query.exec(StrChng);    }
    catch(...){
```

Продолжение приложения В

```
        LnInfo = QString::fromLocal8Bit(" Ошибка при выполнении
команды в БД. Процедура -- DataBD::ChangeRcdBD(Prm) -- ");
        SaveToLogBD(LnInfo);
    }
}
// Получить целое значение поля БД
int DataBD::GetValIntFromField(QString SqlString, int NbrFld)
{
    int RztValInt;
    QSqlQuery query;
    QString LnInfo;
    try{
//        LnInfo = QString::fromLocal8Bit(" Получить целое значение
поля БД. Процедура -- DataBD::GetValIntFromField(Prm) -- ");
//        SaveToLogBD(LnInfo);
        query.exec(SqlString);
        query.first();
        RztValInt = query.value(NbrFld).toInt();
        return RztValInt;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при получении
целого значения поля БД. Процедура -- DataBD::GetValIntFromField(Prm) --
");
        SaveToLogBD(LnInfo);
        return false;
    }
}
// Получить вещественное значение поля БД
float DataBD::GetValFloatFromField(QString SqlString, int NbrFld)
{
    float RztValFloat;
    QSqlQuery query;
    QString LnInfo;
    try{
        LnInfo = QString::fromLocal8Bit(" Получить вещественное
значение поля БД. Процедура -- DataBD::GetValFloatFromField(Prm) -- ");
//        SaveToLogBD(LnInfo);
        query.exec(SqlString);
        query.first();
        RztValFloat = query.value(NbrFld).toDouble();
        return RztValFloat;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при получении
вещественного значения поля БД. Процедура --
DataBD::GetValFloatFromField(Prm) -- ");
        SaveToLogBD(LnInfo);
        return false;
    }
}
// Получить строковое значение поля БД
QString DataBD::GetValStrFromField(QString SqlString, int NbrFld)
{
    QString RztValString;
    QSqlQuery query;
    QString LnInfo;
    try{
```

Продолжение приложения В

```
//          SaveToLogBD(LnInfo);
           query.exec(SqlString);
           query.first();
           RztValString = query.value(NbrFld).toString();
           return RztValString;
       }
       catch(...){
           LnInfo = QString::fromLocal8Bit(" Ошибка при получении
строкового значения поля БД. Процедура -- DataBD::GetValStrFromField(Prm) -
- ");

           SaveToLogBD(LnInfo);
           RztValString = QString::fromLocal8Bit(" Ошибка ");
           return RztValString;
       }
   }
// Проверка наличия записи в БД
bool DataBD::GetRztNotNullBD(QString SqlString)
{
    bool Rezult;
    QSqlQuery query;
    QString LnInfo;
    try{
//          LnInfo = QString::fromLocal8Bit(" Проверка наличия записи в
БД. Процедура -- DataBD::GetRztNotNullBD(Prm) -- ");
//          SaveToLogBD(LnInfo);
           query.exec(SqlString);
           if(query.size()>0) Rezult = true;
           else Rezult = false;
           return Rezult;
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка при проверке
наличия записи в БД. Процедура -- DataBD::GetRztNotNullBD(Prm) -- ");
        SaveToLogBD(LnInfo);
        return false;
    }
}
// Обновление счетчика таблицы MsgBuffer по номеру пакета
void DataBD::GetSqlUpdtBD_Buffer(struct masParam *Param)
{
    QString LnSql, ns;
    try{
//          LnSql = QString::fromLocal8Bit(" Обновление счетчика таблицы
MsgBuffer. Процедура -- DataBD::GetSqlUpdtBD_Buffer(Prm) -- ");
//          SaveToLogBD(LnSql);
           LnSql = "UPDATE `MsgBuffer`"; LnSql += " SET `Cnt` = ";
           LnSql += QString::number(Param->nCnt); LnSql += ", `Upd` =
'"; LnSql += Param->nUpd;
           LnSql += "' WHERE `NbrPak` = ";
           LnSql += QString::number(Param->nmPak);
           LnSql += " and `NbMod` = ";
           LnSql += QString::number(Param->nmMod); ns = ";"; LnSql
+= ns;

           ChangeRcdBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при обновлении
счетчика таблицы MsgBuffer. Процедура -- DataBD::GetSqlUpdtBD_Buffer(Prm) -
- ");
    }
}
```

Продолжение приложения В

```
    }
}
// Вставка новой записи в таблицу MsgBuffer
void DataBD::GetSqlInsertBD_Buffer(struct masParam *Param)
{
    QString LnSql;
    try{
//        LnSql = QString::fromLocal8Bit(" Вставка новой записи в
таблицу MsgBuffer. Процедура -- DataBD::GetSqlInsertBD_Buffer(Prm) -- ");
//        SaveToLogBD(LnSql);
        LnSql = "INSERT INTO `MsgBuffer`(`NbMod`, `NbrPak`, `Dat`,
`Upd`, `Cnt`) VALUES (";
        LnSql += QString::number(Param->nmMod); LnSql += ", ";
        LnSql += QString::number(Param->nmPak); LnSql += ", ";
LnSql += Param->nData;
        LnSql += "' , '"; LnSql += Param->nUpd; LnSql += "' , ";
        LnSql += " 1 )";
        ChangeRcdBD(LnSql);
//        SaveToLogBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при вставке новой
записи в таблицу MsgBuffer. Процедура -- DataBD::GetSqlInsertBD_Buffer(Prm)
-- ");
        SaveToLogBD(LnSql);
    }
}
/*Заполнение и контроль за буфером пакетов*/
// Очистка таблицы MsgBuffer
void DataBD::GetSqlDeleteBD_Buffer()
{
    QString LnSql;
    try{
//        LnSql = QString::fromLocal8Bit(" Очистка таблицы MsgBuffer.
Процедура -- DataBD::GetSqlDeleteBD_Buffer() -- ");
//        SaveToLogBD(LnSql);
        LnSql = "DELETE FROM `MsgBuffer`";
        ChangeRcdBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при очистке таблицы
MsgBuffer. Процедура -- DataBD::GetSqlDeleteBD_Buffer() -- ");
        SaveToLogBD(LnSql);
    }
}
// Обновление счетчика таблицы KTMMMSG по номеру пакета
void DataBD::GetSqlUpdtBD_Msg(struct masParam *Param)
{
    QString LnSql, ns;
    try{
        LnSql = QString::fromLocal8Bit(" Обновление счетчика таблицы
KTMMMSG. Процедура -- DataBD::GetSqlUpdtBD_Msg(Prm) -- ");
//        SaveToLogBD(LnSql);
        LnSql = "UPDATE `KTMMMSG`"; LnSql += " SET `Cnt` = ";
        LnSql += QString::number(Param->nCnt); LnSql += ", `Upd` =
";
        LnSql += Param->nUpd; LnSql += "' WHERE `Num` = ";
        LnSql += QString::number(Param->nmPak);
```

Продолжение приложения В

```
        ns = ";";          LnSql += ns;
        ChangeRcdBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при обновлении
счетчика таблицы KTMMMSG. Процедура -- DataBD::GetSqlUpdtBD_Msg(Prm) -- ");
        SaveToLogBD(LnSql);
    }
}
// Вставка новой записи в таблицу KTMMMSG
void DataBD::GetSqlInsertBD_Msg(struct masParam *Param)
{
    QString LnSql;
    try{
        LnSql = QString::fromLocal8Bit(" Вставка новой записи в
таблицу KTMMMSG. Процедура -- DataBD::GetSqlInsertBD_Msg(Prm) -- ");
        LnSql = "INSERT INTO `KTMMMSG`(`Modul`, `Rep`, `Num`, `Msg`,
`Dat`, `Upd`, `Ademco`, `Cnt`) VALUES (";
        LnSql += QString::number(Param->nmMod); LnSql += ", "; LnSql
+= QString::number(Param->nmRpt);
        LnSql += ", "; LnSql += QString::number(Param->nmPak);
        ////////////////старая
строка//////////////////////////
//          LnSql += ", "; LnSql += Param->vlEvnt; LnSql += ", ";
LnSql += Param->nData;
        ////////////////новая строка//////////////////////////
        LnSql += ", "; LnSql += GECod(Param->vlEvnt); LnSql += ",
"; LnSql += Param->nData;
        LnSql += ", "; LnSql += Param->nUpd; LnSql += ", ";
LnSql += Param->vlPak;
        LnSql += ", "; LnSql += " 1 );";
        ChangeRcdBD(LnSql);
        SaveToLogBD(LnSql); //добавил
        LnSql = "SELECT LAST_INSERT_ID( )";
        Param->ID = GetValIntFromField(LnSql,0);
        LnSql = QString::fromLocal8Bit("last_id=");
        LnSql+=QString::number(Param->ID);
//          SaveToLogBD(LnSql);
        GetTpMsg(Param);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при вставке новой
записи в таблицу KTMMMSG. Процедура -- DataBD::GetSqlInsertBD_Msg(Prm) --
");
        SaveToLogBD(LnSql);
    }
}
// Обновление счетчика таблицы KTMSTATSIG по номеру модуля
void DataBD::GetSqlUpdtBD_Mod(struct masParam *Param)
{
    QString LnSql ,s_sql;
    int last_id;
    try{
        LnSql = QString::fromLocal8Bit(" Обновление счетчика таблицы
KTMSTATSIG. Процедура -- DataBD::GetSqlUpdtBD_Mod(Prm) -- ");
//          SaveToLogBD(LnSql);
        s_sql = "Select ID from `KTMSTATSIG` where `Modul` = ";
        s_sql += QString::number(Param->nmMod);
```

Продолжение приложения В

```
s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
last_id = GetValIntFromField(s_sql, 0);
LnSql = "UPDATE `KTMSTATSIG`"; LnSql += " SET `Cnt` = ";
LnSql += QString::number(Param->nCnt); LnSql += ", `Upd` =
";
LnSql += Param->nUpd; LnSql += "' WHERE ID = "; LnSql +=
QString::number(last_id);
LnSql += " ";
SaveToLogBD(LnSql);
ChangeRcdBD(LnSql);
}
catch(...){
LnSql = QString::fromLocal8Bit(" Ошибка при обновлении
счетчика таблицы KTMSTATSIG. Процедура -- DataBD::GetSqlUpdtBD_Mod(Prm) --
");
SaveToLogBD(LnSql);
}
}
// Вставка новой записи в таблицу KTMSTATSIG
void DataBD::GetSqlInsertBD_Mod(struct masParam *Param)
{
QString LnSql;
try{
LnSql = QString::fromLocal8Bit(" Вставка новой записи в
таблицу KTMSTATSIG. Процедура -- DataBD::GetSqlInsertBD_Mod(Prm) -- ");
// SaveToLogBD(LnSql);
LnSql = "INSERT INTO `KTMSTATSIG`(`Modul`, `Num`, `Rep`,
`Signal`, `Cnt`, `Dat`, `Upd`) VALUES (";
LnSql += QString::number(Param->nmMod); LnSql += ", "; LnSql
+= QString::number(Param->nmRpt);
LnSql += ", "; LnSql += (Param->vlPak.mid(4,1));
LnSql += ", "; LnSql += QString::number(Param->Sgnl);
LnSql += ", "; LnSql += QString::number(Param->nCnt); LnSql
+= ", "; LnSql += Param->nData;
LnSql += "' , '"; LnSql += Param->nUpd; LnSql += "' )";
ChangeRcdBD(LnSql);
}
catch(...){
LnSql = QString::fromLocal8Bit(" Ошибка при вставке новой
записи в таблицу KTMSTATSIG. Процедура -- DataBD::GetSqlInsertBD_Mod(Prm) -
- ");
SaveToLogBD(LnSql);
}
}
/*Заполнение и контроль за сигналом*/
void DataBD::GetBufMod(struct masParam *Param)
{
QString s_sql, LnInfo;
bool R_yes;
int cntP, Sign, five;
try{
LnInfo += QString::fromLocal8Bit(" Буфер сигналов. Процедура --
DataBD::GetBufMod(Prm) -- ");
// SaveToLogBD(LnInfo);
s_sql = "Select * from `KTMSTATSIG` where `Modul` = ";
s_sql += QString::number(Param->nmMod);
s_sql += " and Num = "; s_sql += QString::number(Param->nmRpt);
s_sql += " ORDER BY `Upd` DESC LIMIT 1 ";
SaveToLogBD(s_sql); R_yes = GetRztNotNullBD(s_sql);
```

Продолжение приложения В

```
if(R_yes != true){
    GetSqlInsertBD_Mod(Param);
    SaveToLogBD("R_yes = False GetBufMod ");
}
else{
    SaveToLogBD("R_yes = True GetBufMod");
    Sign = GetValIntFromField(s_sql, 4);
    LnInfo = QString::fromLocal8Bit(" Sign=");
    LnInfo += QString::number(Sign);
    SaveToLogBD(LnInfo);
    LnInfo = QString::fromLocal8Bit(" Nsign=");
    LnInfo += QString::number(Param->Sgnl);
    SaveToLogBD(LnInfo);
    five=(Sign*5)/100;
    LnInfo = QString::fromLocal8Bit(" five=");
    LnInfo += QString::number(five);
    SaveToLogBD(LnInfo);
    if ((abs(Sign-Param->Sgnl))<=five){
        cntP = GetValIntFromField(s_sql, 5);
        cntP++; Param->nCnt = cntP;
        GetSqlUpdtBD_Mod(Param);
        LnInfo = QString::fromLocal8Bit(" обновить GetSqlUpdtBD_Mod
");
        SaveToLogBD(LnInfo);
    }
    else {
        GetSqlInsertBD_Mod(Param);
        LnInfo = QString::fromLocal8Bit(" Втавить
GetSqlInsertBD_Mod ");
        SaveToLogBD(LnInfo);
    }
}
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка в Буфере сигналов.
Процедура -- DataBD::GetBufMod(Prm) -- ");
    SaveToLogBD(LnInfo);
}
}
/*ДОБАВИЛ */
QString DataBD::GECod(QString StrInpt)
{
    QChar tend;
    unsigned short k;
    QString str_cmd, s_work, sw;
    str_cmd = StrInpt + ' '; s_work = '0'; k = 0;
    while(k < str_cmd.length()-1){
        tend = str_cmd[k];
        if(tend != ' ') {
            sw = str_cmd.mid(k,2);
            if (sw.contains(' ') != false) {
                sw = sw.mid(0,1); sw = '0' + sw;
            }
            s_work += sw; k = k + 2;
        }
        else k++;
    }
    str_cmd = s_work.mid(1,8);
    return str_cmd;
}
```

Продолжение приложения В

```

}
//
/*Разпознавание типа сообщения*/
void DataBD::GetTpMsg(struct masParam *Param)
{
    QString s_sql, LnInfo;
    struct masEvent*Evnt;
    bool R_yes;
    try{
        LnInfo = QString::fromLocal8Bit(" Разпознавание типа
сообщения. Процедура -- DataBD::GetTpMsg(Prm) -- ");
//        SaveToLogBD(LnInfo);
        Evnt = new struct masEvent;
        s_sql = " SELECT CardId, FlagView FROM Cards left join Modul
";
        s_sql += " ON Modul.Idmod = Cards.Unitnumber Where NumberMod
= ";
        s_sql +=QString::number(Param->nmMod);
        s_sql += " and Modul.Idmod = Cards.Unitnumber";
//        SaveToLogBD(s_sql);
        R_yes =GetRztNotNullBD(s_sql);
        if (R_yes!=false){// Есть карточка?
            LnInfo = QString::fromLocal8Bit(" GetTpMsg
R_yes=true ");
//            SaveToLogBD(LnInfo);
            Evnt->Flg = GetValIntFromField(s_sql,1);
            if (Evnt->Flg!=0) { // Заблокирована?
                LnInfo = QString::fromLocal8Bit("GetTpMsg
Flg=true ");
//                SaveToLogBD(LnInfo);
                Evnt->idCrd = GetValIntFromField(s_sql,0);
                s_sql = "SELECT `CodTpE` FROM `Enames` WHERE Code = '";
                s_sql += GECod(Param->vlEvnt); s_sql += "'";
                Evnt->CdTpE = GetValIntFromField(s_sql,0);
                LnInfo = "Code = "; LnInfo += Param->vlEvnt;
                LnInfo += " CodTp ="; LnInfo+=   QString::number(Evnt-
>CdTpE);
                s_sql = " SELECT Rajon, Zone FROM Enames WHERE Code =
'";
                s_sql +=GECod(Param->vlEvnt); s_sql += "'";
                Evnt->NbZn = GetValIntFromField(s_sql,1);
                Evnt->NrSct = GetValIntFromField(s_sql,0);
//                SaveToLogBD(LnInfo);
                Evnt->idSection = GetIDSection(Evnt);
                if (Evnt->idSection!=0) {
//                    LnInfo =QString::fromLocal8Bit("
IDSection=yes "); SaveToLogBD(LnInfo);
                    switch (Evnt->CdTpE)
                    {
//                        case 1 : {
//                            LnInfo
=QString::fromLocal8Bit(" выбрано 1 "); SaveToLogBD(LnInfo);
                            if (Evnt->NbZn!=0){
                                Evnt->NbZn = Evnt-
>NbZn+10;
                                AlrmMsg(Param,Evnt);
                            }
                        }
                    }
                }
            }
        }
    }
}

```


Продолжение приложения В

```
AlrmMsg_NotZone (Param, Evnt);
    }
    break;
}
case 2 : {
    LnInfo
// =QString::fromLocal8Bit(" выбрано 2 "); SaveToLogBD(LnInfo);
    if (Evnt->NbZn!=0){
        Evnt->NbZn = Evnt-
>NbZn+10;
        TechMsg (Param, Evnt);
    }
    else {
        // обработка тех
события без зоны
        TechMsg_NotZone (Param, Evnt);
    }
    break;
}
case 3 :{
    LnInfo
// =QString::fromLocal8Bit(" выбрано 3 "); SaveToLogBD(LnInfo); break;
}
case 4 : {
    LnInfo
// =QString::fromLocal8Bit(" выбрано 4 "); SaveToLogBD(LnInfo);
    DisArm_Arm (Param, Evnt);
break;
}
case 5 : {
    LnInfo
// =QString::fromLocal8Bit(" выбрано 5 "); SaveToLogBD(LnInfo);
    DisArm_Arm (Param, Evnt);
    Send_ALARM_1R (Evnt-
>idSection);
    break;
}
}
}
else LnInfo = QString::fromLocal8Bit(" R_yes=false");
delete Evnt;
}

catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка в Разпознавании
типа сообщения. Процедура -- DataBD::GetTpMsg (Prm) -- ");
    SaveToLogBD(LnInfo);
    delete Evnt;
}

}
//
/*Тревожное Событие*/
void DataBD::AlrmMsg(struct masParam *Param, struct masEvent*Evnt)
{
    QString s_sql, LnInfo, Code;
```

Продолжение приложения В

```
bool R_Alrm;
try{
    LnInfo = QString::fromLocal8Bit(" Тревожное событие.
Процедура -- DataBD::AlrmMsg -- ");
    // SaveToLogBD(LnInfo);
    Evnt->idZone = GetIDZone(Evnt,0);
    if (Evnt->idZone!=0) {// есть зона?
        LnInfo = QString::fromLocal8Bit(" AlrmMsg --зона
есть ");
        // SaveToLogBD(LnInfo);
        Evnt->FlgZn = GetIDZone(Evnt,1);
        LnInfo = QString::fromLocal8Bit(" idZone= ");
        LnInfo +=QString::number(Evnt->idZone);
        // SaveToLogBD(LnInfo);
        Code = GECod(Param->vlEvnt).mid(0,5);
        LnInfo = QString::fromLocal8Bit(" Тревожное событие.
Code = ");
        LnInfo += Code;
        // SaveToLogBD(LnInfo);
        if ((Evnt->FlgZn!=1) and (Code!="01120")){//
блокирована?//если 0 то не блок
        //добавить или обновить ALARME
        //по состоянию зоны
        LnInfo = QString::fromLocal8Bit(" AlrmMsg --
зона не блокирована ");
        // SaveToLogBD(LnInfo);
        s_sql = " SELECT AlarmID FROM ALARME WHERE
ZoneID = ";
        s_sql += QString::number(Evnt->idZone);
        s_sql += " and State < 128 ";
        R_Alrm = GetRztNotNullBD(s_sql);
        if (R_Alrm!=false){
            //update
            Evnt->idAlrm =
            GetValIntFromField(s_sql,0);
            ALARME WHERE AlarmID = ";
            Evnt->idAlrm);
            GetValIntFromField(s_sql,0);
            s_sql = " SELECT ReceiveCount FROM
            s_sql += QString::number(Evnt-
            Evnt->cnt =
            Evnt->cnt++;
            GetSqlUpdt_Alarme(Param,Evnt);
        }
        else {
            //insert
            GetSqlInsert_Alarme(Param,Evnt);
        }
        //обновить состояние зоны
        GetSqlUpdt_ZonStat(Evnt);
    }
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка в Тревожное
событие. Процедура -- DataBD::AlrmMsg -- ");
    SaveToLogBD(LnInfo);
}
}
```

Продолжение приложения В

```
//
/*Вставка новой записи в таблицу ALARME*/
void DataBD::GetSqlInsert_Alarme(struct masParam *Param, struct
masEvent*Evnt)
{
    QString LnSql;
    try{
        LnSql = QString::fromLocal8Bit(" Вставка новой записи в
таблицу Alarme. Процедура -- DataBD::GetSqlInsert_Alarme(Evnt) -- ");
//        SaveToLogBD(LnSql);
        LnSql = " INSERT INTO `ALARME`(`ZoneID`, `State`,
`ReceiveTime`,`";
        LnSql += " `LastReceiveTime`, `ReceiveCount`, `RowID`) VALUES
( ";
        LnSql += QString::number(Evnt->idZone); LnSql += " , 1 , '";
        LnSql += Param->nData; LnSql += " ' , '";
        LnSql += Param->nData; LnSql += " ' , 1 , ";
        LnSql += QString::number(Param->ID); LnSql += " )";
        ChangeRcdBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при вставке новой
записи в таблицу Alarme. Процедура -- DataBD::GetSqlInsert_Alarme(Evnt) --
");
        SaveToLogBD(LnSql);
    }
}
//
/*Обновление таблицы ALARME по idZone*/
void DataBD::GetSqlUpdt_Alarme(struct masParam *Param, struct
masEvent*Evnt)
{
    QString LnSql;
    try{
        LnSql = QString::fromLocal8Bit(" Обновление  таблицы ALARME.
Процедура -- DataBD::GetSqlUpdt_Alarme() -- ");
//        SaveToLogBD(LnSql);
        LnSql = " UPDATE `ALARME` SET `LastReceiveTime`= '";
        LnSql += Param->nData; LnSql += " ' , ReceiveCount = ";
        LnSql += QString::number(Evnt->cnt);
        LnSql += " WHERE AlarmID = "; LnSql +=
QString::number(Evnt->idAlrm);
//        SaveToLogBD(LnSql);
        ChangeRcdBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при обновлении
таблицы ALARME. Процедура -- DataBD::GetSqlUpdt_Alarme() -- ");
        SaveToLogBD(LnSql);
    }
}
//
/*Техническое событие*/
void DataBD::TechMsg(struct masParam *Param, struct masEvent*Evnt)
{
    QString s_sql, LnInfo;
    bool R_yes;
    int Stat;
    try{
```

Продолжение приложения В

```

LnInfo = QString::fromLocal8Bit(" Техническое событие.
Процедура -- DataBD::TechMsg -- ");
//
SaveToLogBD(LnInfo);
if (Evt->NbZn!=0) {
    //восстановление
    Evt->idZone= GetIDZone(Evt,0);
    if (Evt->idZone!=0) { // есть зона?
        LnInfo = QString::fromLocal8Bit(" TechMsg --
зона есть ");
//
SaveToLogBD(LnInfo);
Evt->FlgZn =GetIDZone(Evt,1);
LnInfo = QString::fromLocal8Bit(" idZone=
");
LnInfo +=QString::number(Evt->idZone);
SaveToLogBD(LnInfo);
if (Evt->FlgZn!=1) { // блокирована?
    //есть такая зона в ALARME
    LnInfo = QString::fromLocal8Bit("
TechMsg --зона не блокирована ");
//
SaveToLogBD(LnInfo);
s_sql = " SELECT AlarmID FROM ALARME
WHERE ZoneID = ";
s_sql += QString::number(Evt->
idZone);
s_sql += " and State < 128 ";
R_yes = GetRztNotNullBD(s_sql);
if (R_yes!=false) {
    //зона есть в ALARME надо
обновить RecoverTime
Evt->idAlrm =
GetValIntFromField(s_sql,0);
s_sql = " UPDATE `ALARME`
SET RecoverTime = ";
s_sql += "'";
if ((Evt->NbZn == 105) or (Evt->NbZn == 107)) {
    Stat =
LnInfo =
SaveToLogBD(LnInfo);
if (Stat<128) {
    s_sql += " ,
State = State +128";
}
}
s_sql += " WHERE AlarmID = ";
SaveToLogBD(s_sql);
ChangeRcdBD(s_sql);
}
}
//обновить состояние зоны
GetSqlUpdt_ZonStat(Evt);
}
}
else{
// другое техническое

```

Продолжение приложения В

```
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка в Техническое
событие. Процедура -- DataBD::TechMsg -- ");
        SaveToLogBD(LnInfo);
    }
}
//
/* ALARME notZone*/
void DataBD::AlrmMsg_NotZone(struct masParam *Param, struct masEvent*Evt)
{
    QString s_sql, LnSql, msg,msg2,tr1,CODD;
    bool R_220, R_AKB;
    try{
        LnSql = QString::fromLocal8Bit(" Тревожное событие, не зона.
Процедура -- DataBD::AlrmMsg_NotZone(Evt) -- ");
        SaveToLogBD(LnSql);
        CODD = GECod(Param->vlEvt);
        msg = "00017851";
        if (CODD == msg) {
            // вход в прог
            LnSql = QString::fromLocal8Bit(" вход в прог ");
            SaveToLogBD(LnSql);
            Evt->NbZn = 105;
            Evt->NrSct = 0;
            Evt->idSection = GetIDSection(Evt);
        }
        else {
            s_sql = " SELECT `CodE` FROM `Enames` WHERE `CodTpE`
=1 AND `CodE` ";
            s_sql += " IN ( '0000000A', '0000009A', '0000029A',
'00009002', '00017021', ";
            s_sql += " '00017331' ) AND `CodE` = '";
            s_sql+=CODD;
            s_sql += "' LIMIT 1 ";
            SaveToLogBD(s_sql);
            R_AKB = GetRztNotNullBD(s_sql);
            if (R_AKB!=false) {
                //AKB
                LnSql = QString::fromLocal8Bit(" AKB ");
                SaveToLogBD(LnSql);
                Evt->NbZn = 107;
                Evt->NrSct = 0;
                Evt->idSection = GetIDSection(Evt);
            }
            else {
                //не AKB и не прог
                s_sql = " SELECT CodE FROM `Enames` WHERE
`CodE` IN ( ";
                s_sql += " '0000000B', '0000009B',
'000000D2', '0000029B', ";
                s_sql += " '000002D2', '00017011',
'00017321', '1319AAAA', ";
                s_sql += " '029BAAAA',
'00009003','012C0100','012C0101' ) AND `CodE` = '";
                s_sql += CODD; s_sql += "' LIMIT 1 ";
                R_220 = GetRztNotNullBD(s_sql);
            }
        }
    }
}
```

Продолжение приложения В

```
if (R_220!=false) {
    //пропало 220
    LnSql = QString::fromLocal8Bit("
пропало 220 ");
    SaveToLogBD(LnSql);
    Evnt->NbZn = 106;
    Evnt->NrSct = 0;
    Evnt->idSection = GetIDSection(Evnt);
}
else {
    // зона на modul
    LnSql = QString::fromLocal8Bit(" зона
на modul ");
    SaveToLogBD(LnSql);
    tr1="000002B8"; //SaveToLogBD(tr1);
    if (CODD == "000002B8") {
        // TP.KH 1
        LnSql = QString::fromLocal8Bit("Evnt-
>NbZn = 2; ");
        SaveToLogBD(LnSql);
        Evnt->NbZn = 2;
    }
    else{
        msg="000002B9";
        if (CODD == msg){
            // TP.KH 2
            LnSql =
QString::fromLocal8Bit("Evnt->NbZn = 3; ");
            SaveToLogBD(LnSql);
            Evnt->NbZn = 3;
        }
        else{
            msg="00009004";
            if (CODD == msg){
                //Пож шлейф
                LnSql =
QString::fromLocal8Bit("Evnt->NbZn = 4; ");
                SaveToLogBD(LnSql);
                Evnt->NbZn = 4;
            }
            else {
                msg="000002B7";
                if (CODD == msg){
                    // Обрыв
                    LnSql =
Панели
QString::fromLocal8Bit("Evnt->NbZn = 8; ");
                    SaveToLogBD(LnSql);
                    Evnt->NbZn =
8;
                }
                else{
                    //Может
                    LnSql =
принуждение?
QString::fromLocal8Bit("Может принуждение?");
                    SaveToLogBD(LnSql);
```


Продолжение приложения В

```

s_sql = " SELECT `CodE` FROM `Enames` WHERE `CodTpE`
=2 AND `CodE` ";
s_sql += " IN ( '0000000D', '000000DA', '000002DA',
'0000A002', '00017731', ";
s_sql += " '00017481', '00017421', '3318AAAA',
'020AAAAA') AND `CodE` = '";
s_sql+=GECod(Param->vlEvnt);
s_sql += "' LIMIT 1 ";
SaveToLogBD(s_sql);
R_AKB = GetRztNotNullBD(s_sql);
if (R_AKB!=false) {
//AKB
LnSql = QString::fromLocal8Bit(" AKB ");
SaveToLogBD(LnSql);
Evt->NbZn = 107;
Evt->NrSct = 0;
Evt->idSection = GetIDSection(Evt);
}
else {
//не AKB и не прог
s_sql = " SELECT CodE FROM `Enames` WHERE
`CodE` IN ( ";
s_sql += " '0000000E', '000000D3',
'000000DB', '000002D3', ";
s_sql += " '000002DB', '0000A003',
'00017411', '00017721', ";
s_sql += " '3319AAAA',
'02DBAAAA','012D0100','012D0101') AND `CodE` = '";
s_sql += GECod(Param->vlEvnt); s_sql += "'
LIMIT 1 ";
R_220 = GetRztNotNullBD(s_sql);
if (R_220!=false) {
//возстановилось 220
LnSql = QString::fromLocal8Bit("
пропало 220 ");
SaveToLogBD(LnSql);
Evt->NbZn = 106;
Evt->NrSct = 0;
Evt->idSection = GetIDSection(Evt);
}
else {
// может на перед?
LnSql = QString::fromLocal8Bit(" зона
на modul ");
SaveToLogBD(LnSql);
if (Ecod == "0000023F"){
LnSql = QString::fromLocal8Bit(" Вст
тр1 ");
SaveToLogBD(LnSql);
Evt->NbZn = 2;
}
else {
if (Ecod == "0000024F"){
LnSql = QString::fromLocal8Bit(" Вст
тр2 ");
SaveToLogBD(LnSql);
Evt->NbZn = 3;
}
else {

```


Продолжение приложения В

```
void DataBD::DisArm_Arm(struct masParam *Param, struct masEvent*Evnt)
{
    QString LnSql, Ecod, sstm;
    try{
        LnSql = QString::fromLocal8Bit("Событие взятие или снятие .
Процедура -- DataBD::DisArm_Arm() -- ");
        SaveToLogBD(LnSql);
        LnSql = QString::fromLocal8Bit(" NbZn=");
        LnSql += QString::number(Evnt->NbZn);
        LnSql += QString::fromLocal8Bit(" NrSct=");
        LnSql += QString::number(Evnt->NrSct);
        LnSql += QString::fromLocal8Bit(" Crdid=");
        LnSql += QString::number(Evnt->idCrd);
        LnSql += QString::fromLocal8Bit(" idSection=");
        LnSql += QString::number(Evnt->idSection);
        LnSql += QString::fromLocal8Bit(" vlEvnt=");
        LnSql += GECod(Param->vlEvnt);
        // if (Evnt->CdTpE == 4){// постанoвка
        //     LnSql = QString::fromLocal8Bit("взятиетtt");
        //     SaveToLogBD(LnSql);
        //     if ((Evnt->NrSct!=0) and (Evnt->NbZn==0)){
        //         LnSql = QString::fromLocal8Bit("обновить разделы
Evnt->NrSct!=0) and (Evnt->NbZn=0 ");
        //         LnSql = " Update Sections SET State =";
        //         LnSql +=QString::number(Evnt->CdTpE);
        //         LnSql += " , StateDate = '";
        //         LnSql += Param->nData;
        //         LnSql += "' WHERE IdSection = ";
        //         LnSql +=QString::number(Evnt->idSection);
        //         SaveToLogBD(LnSql);
        //         ChangeRcdBD(LnSql);
        //     }
        //     else{//else 1
        //         if ((Evnt->NrSct == 0) and (Evnt->NbZn==0)){// if 0
        //             Ecod = GECod(Param->vlEvnt);
        //             sstm = Ecod.mid(0,5);
        //             LnSql = QString::fromLocal8Bit("sstm=");
        //             LnSql += sstm;SaveToLogBD(LnSql);
        //             if (Evnt->CdTpE == 4){// постанoвка
        //                 LnSql =
        //                 QString::fromLocal8Bit("взятиетtt");
        //                 SaveToLogBD(LnSql);
        //                 LnSql =
        //                 QString::fromLocal8Bit("обновить разделы Evnt->NrSct=0 ");
        //                 SaveToLogBD(LnSql);
        //                 if ((Ecod == "0000026F") or (sstm ==
        //                 "00002")) {
        //                     GetSqlUpdate_Sections(Evnt-
        //                     >CdTpE, Param->nData, Evnt->idCrd);
        //                 }
        //                 if (Ecod == "0000F00A"){
        //                     LnSql =
        //                     GetSqlUpdate_Sections(Evnt-
        //                     >CdTpE, Param->nData, Evnt->idCrd);
        //                 }
        //             }
        //         }
        //     }
    }
}
```

Продолжение приложения В

```
SaveToLogBD(LnSql);
if ((Ecod == "0000027F") or (sstm ==
"00001")) {
    GetSqlUpdate_Sections(Evnt-
>CdTpE, Param->nData, Evnt->idCrd);
    }
    } // снятие
    } // end if 0 0
    } // end else 1
}
catch(...){
    LnSql = QString::fromLocal8Bit(" Ошибка при Событие взятие
или снятие . Процедура -- DataBD::DisArm_Arm() -- ");
    SaveToLogBD(LnSql);
}
}
//
//Получить idSection
int DataBD::GetIDSection(struct masEvent*Evnt)
{
    int idSct;
    QString LnInfo;
    bool R_nn;
    try{
        LnInfo = QString::fromLocal8Bit(" Получить Получить
idSection . Процедура -- DataBD::GetIDSection() -- ");
//
        SaveToLogBD(LnInfo);
        LnInfo = " SELECT IdSection From Sections WHERE IdCard = ";
        LnInfo += QString::number(Evnt->idCrd);
        LnInfo += " and SectionName = "; LnInfo +=
QString::number(Evnt->NrSct);
//
        SaveToLogBD(LnInfo);
        R_nn = GetRztNotNullBD(LnInfo);
        if (R_nn!=false){
            idSct = GetValIntFromField(LnInfo,0);
            return idSct;
        }
        else{
            return 0;
        }
    }
    catch(...){
        LnInfo = QString::fromLocal8Bit(" Ошибка Получить Получить
idSection . Процедура -- DataBD::GetIDSection() -- ");
        SaveToLogBD(LnInfo);
        return false;
    }
}
//
/* Получить idZone или Flags */
int DataBD::GetIDZone(struct masEvent*Evnt, int NbrFld)
{
    int FldVls;
    QString LnInfo;
    bool R_nn;
    try{
        LnInfo = QString::fromLocal8Bit(" Получить idZone .
Процедура -- DataBD::GetIDZone() -- ");
        SaveToLogBD(LnInfo);
```

Продолжение приложения В

```
LnInfo = " SELECT Zones.ZoneID, Zones.Flags FROM Zones left
join ";
LnInfo += " Sections on Sections.IdSection=Zones.SectionID";
LnInfo += " Left join Cards on Cards.CardId=Sections.IdCard
WHERE ";
LnInfo += " Sections.IdSection=Zones.SectionID and ";
LnInfo += " Cards.CardId=Sections.IdCard and ";
LnInfo += " Zones.SectionID = ";
LnInfo += QString::number(Evnt->idSection);
LnInfo += " and Zones.ZoneNumber = "; LnInfo +=
QString::number(Evnt->NbZn);
SaveToLogBD(LnInfo);
R_nn = GetRztNotNullBD(LnInfo);
if (R_nn!=false){
    FldVls = GetValIntFromField(LnInfo,NbrFld);
    return FldVls;
}
else{
    return 0;
}
}
catch(...){
    LnInfo = QString::fromLocal8Bit(" Ошибка при получении
idZone . Процедура -- DataBD::GetIDZone() -- ");
    SaveToLogBD(LnInfo);
    return false;
}
}
//
// Обновление Sections при R=0 и Z=0
void DataBD::GetSqlUpdate_Sections(int CdTpE,QString Date, int idCrd)
{
    QString LnSql;
    try{
        LnSql = QString::fromLocal8Bit("Обновление Sections при R=0
и Z=0. Процедура -- DataBD::GetSqlInsertBD_Buffer() -- ");
        SaveToLogBD(LnSql);
        LnSql = " Update Sections SET State =";
        LnSql +=QString::number(CdTpE);
        LnSql += " , StateDate = '";
        LnSql += Date;
        LnSql += "' WHERE IdCard = ";
        LnSql +=QString::number(idCrd);
        ChangeRcdBD(LnSql);
        SaveToLogBD(LnSql);
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка Обновление Sections
при R=0 и Z=0. Процедура -- DataBD::GetSqlInsertBD_Buffer() -- ");
        SaveToLogBD(LnSql);
    }
}
// Проверка нет ли зоны в тревоге
void DataBD::Send_ALARM_1R(int IdSection)
{
    QString LnSql;
    bool R_yes;
    try{
```

Продолжение приложения В

```
LnSql = QString::fromLocal8Bit("Проверка нет ли зоны в
тревоге . Процедура -- DataBD::Send_ALARM_1R() -- ");
SaveToLogBD(LnSql);
LnSql = " SELECT `AlarmID` FROM `ALARME` ";
LnSql+= " LEFT JOIN Zones ON Zones.ZoneID = ALARME.ZoneID ";
LnSql+= " LEFT JOIN Sections ON Sections.IdSection =
Zones.SectionID ";
LnSql+= " WHERE Zones.ZoneID = ALARME.ZoneID ";
LnSql+= " and Sections.IdSection = Zones.SectionID and
Sections.IdSection = ";
LnSql+= QString::number(IdSection);
LnSql+= " AND (Zones.ZoneNumber BETWEEN 10 AND 90)";
LnSql+= " AND ALARME.State <256";
SaveToLogBD(LnSql);
R_yes = GetRztNotNullBD(LnSql);
if (R_yes!=false){
LnSql = QString::fromLocal8Bit("при снятии есть зона
в тревоге");
SaveToLogBD(LnSql);
LnSql = " UPDATE `ALARME` ";
LnSql+= " LEFT JOIN Zones ON Zones.ZoneID =
ALARME.ZoneID ";
LnSql+= " LEFT JOIN Sections ON Sections.IdSection =
Zones.SectionID ";
LnSql+= " SET ALARME.State = ALARME.State + 128 ";
LnSql+= " WHERE Zones.ZoneID = ALARME.ZoneID ";
LnSql+= " and Sections.IdSection = Zones.SectionID
and Sections.IdSection = ";
LnSql+= QString::number(IdSection);
LnSql+= " AND (Zones.ZoneNumber BETWEEN 10 AND
90)";
LnSql+= " AND ALARME.State <256";
ChangeRcdBD(LnSql);
SaveToLogBD(LnSql);
}
}
catch(...){
LnSql = QString::fromLocal8Bit(" Ошибка при Проверке нет ли
зоны в тревоге . Процедура -- DataBD::Send_ALARM_1R() -- ");
SaveToLogBD(LnSql);
}
}
//
/*Обновить состояние зоны*/
void DataBD::GetSqlUpdt_ZonStat(struct masEvent*Evnt)
{
QString LnSql;
try{
LnSql = QString::fromLocal8Bit(" Обновление EventType Zone.
Процедура -- DataBD::GetSqlUpdt_ZonStat() -- ");
SaveToLogBD(LnSql);
LnSql = " UPDATE `Zones` SET `EventType`= ' ";
LnSql += QString::number(Evnt->CdTpE);
LnSql += "' WHERE `ZoneID`= ";
LnSql += QString::number(Evnt->idZone);
SaveToLogBD(LnSql);
ChangeRcdBD(LnSql);
}
catch(...){
```

Продолжение приложения В

```
LnSql = QString::fromLocal8Bit(" Ошибка при обновлении
EventType Zone. Процедура -- DataBD::GetSqlUpdt_ZonStat() -- ");
SaveToLogBD(LnSql);
}
}
//
/*Проверить есть ли 98 зона*/
void DataBD::Get_98Zone(struct masEvent*Evnt)
{
    QString LnSql;
    bool R_98;
    try{
        LnSql = QString::fromLocal8Bit(" Проверка наличия зоны 98 -
- DataBD::Get_98Zone() -- ");
        SaveToLogBD(LnSql);
        LnSql = " SELECT Z.`ZoneID` ";/*
                " FROM Zones Z " +
LEFT JOIN Sections S ON S.`IdSection` = Z.`SectionID`
LEFT JOIN Cards C ON C.`CardId` = S.`IdCard`
WHERE S.`IdSection` = Z.`SectionID`
AND C.`CardId` = S.`IdCard`
AND Z.`ZoneNumber` =98
AND C.`CardId` =8965
AND S.`SectionName` =1;*/
    }
    catch(...){
        LnSql = QString::fromLocal8Bit(" Ошибка при Проверки наличия
зоны 98 -- DataBD::Get_98Zone() -- ");
        SaveToLogBD(LnSql);
    }
}

#include <QtCore>
#include <QTextStream>
#include <QIODevice>
#include <QtDebug>
#include <QVector>
#include <QTime>
#include <QDateTime>
#include <iostream>
#include <qdir.h>
#include <qfile.h>
#include <time.h>
#include <queue>

#include "echoserver.h"
#include "thread.h"

using namespace std;
//
Thread::Thread( )
{
    stopped = false;
}
//
void Thread::run()
{
    QString StrLog, StrOtvvet, Ln, pak, Ln_test, Tmp, Temp_log;
```


Продолжение приложения В

```
//
void Thread::stop()
{
    QString StrLog;
    stopped = true;
    StrLog = QString::fromLocal8Bit(" Остановлен поток передачи ");
    SaveToFile(StrLog);
}
/*Создание лог файла */
void Thread::SaveToFile(QString StrLog)
{
    QTime tm;
    QFile flSave;
    QTextStream stream( &flSave );
    QStringList lines;
    QString Fname, line, stm;
    Fname = QDir::currentPath() + "/Potok.log";
    tm = tm.currentTime(); stm = tm.toString("hh:mm:ss.zzz ");
    line = QDate::currentDate().toString("dd/MM/yy");
    line += ": "; line += stm; line += StrLog;
    flSave.setFileName(Fname);
    lines << line;
    try{
        if(flSave.exists()) flSave.open(QIODevice::Append);
        else flSave.open(QIODevice::ReadWrite);
        for ( QStringList::Iterator it = lines.begin(); it !=
lines.end(); ++it )
            stream << *it << "\n";
        flSave.close();
        if (stream.status() != QTextStream::Ok){
            qDebug() << "Error write file";
        }
    }
    catch(...){
        qDebug() << "Error write file";
    }
}
/*Функция распечатки пакета 16*/
QString Thread::PrintPakHex(unsigned char *buf, unsigned short len)
{
    QString paket, Tmpplog;
    char print_buf[24]; //Буфер для печати
    //char Tmp_buf[24]; // dobaliv
    ::sprintf(print_buf, "%X", buf[0]);
    paket = QString::fromAscii(print_buf);
    // SaveToFile(paket); //для отладки
    for(int i=1; i<len; i++){
        ::sprintf(print_buf, "%X", buf[i]);
        paket += " " + QString::fromAscii(print_buf);
        //SaveToFile(paket); //для отладки
    }
    // ::sprintf(Tmp_buf, "%X", buf[1]); //добавил
    // TmppLog = QString::fromAscii(Tmp_buf); // dobavil
    // SaveToFile(TmppLog); //для отладки добавил
    }
    SaveToFile(paket);
    return paket;
}
/*Процедура буфера модема */
int Thread::ParserPaket(unsigned char *rBf2)
```


Продолжение приложения В

```

{
    QString Ln, pak, Tmpp;
    int i=0;
    int psBgn, rlen, ret, k;
    unsigned short len_InpBuf;
    unsigned short len_OutBuf;
    unsigned char t_byte;
    //char printt[24];
    //char Tmp_char[24]; //dobavil
    QQueue <unsigned char> ::iterator iQ;
    try {
        //Tmpp = QString::fromLocal8Bit(" Вход в ParserPaket");
        //SaveToFile(Tmpp);
        ret=0;
        if(!InpBufQue.isEmpty()){
            len_InpBuf=InpBufQue.size();
            while(i<len_InpBuf){
                t_byte = InpBufQue.value(i);
                if(t_byte == 0x34){
                    //Tmpp = QString::fromLocal8Bit(" posle ==0x34
ParserPaket");
                    //SaveToFile(Tmpp);
                    psBgn = i;
                    rBf2[0]=t_byte;
                    i++;
                    len_OutBuf=13;
                    rlen=len_InpBuf-psBgn;
                    if(rlen>=len_OutBuf){
                        //Tmpp = QString::number(i);
                        //Tmpp+=QString::fromLocal8Bit("
=i");
                        //SaveToFile(Tmpp);
                        rBf2[1]=InpBufQue.value(i); //dobavil
                        for(int j=0; j<len_OutBuf; j++){
                            i++;
                            rBf2[j+2]=InpBufQue.value(i);
                        }
                        //Tmpp="rbf2-"; //dobavil
                        //for (int ww=0; ww<len_OutBuf;
ww++){ //dobavil
                        //      ::sprintf(Tmp_char, "%X",
rBf2[ww]); //dobavil
                        //      Tmpp += " " +
QString::fromAscii(Tmp_char); //dobavil
                        //} //dobavil
                        //SaveToFile(Tmpp); //dobavil
                        //Tmpp = QString::fromLocal8Bit("
posle k=0");
                        //      SaveToFile(Tmpp);
                        for(k=0; k<(len_OutBuf+psBgn);
k++){
                            if(!InpBufQue.isEmpty()) {
                                InpBufQue.dequeue();
                                Tmpp =
QString::number(k);
                                //      Tmpp +=
QString::fromLocal8Bit(" len_OutBuf+psBgn ParserPaket");
                                //      SaveToFile(Tmpp); }

```

Продолжение приложения В

```
        }
        ret=len_OutBuf; break;
    }
    else {
        ret=0;
        break;
    }
}
else i++;
}
}
return ret;
}
catch (...) {
    Ln = " <<-- RS ";
    Ln += QString::fromLocal8Bit(" Ошибка в процедуре
ParserPaket (Param)");
    SaveToFile(Ln);
    return ret = -1;
}
}
/*Расчет контрольной суммы */
bool Thread::GetStringSum(unsigned char *rBf2)
{
    QString Ln, sB, sP;
    int sum,/* def,*/ len;
    //unsigned char bf[16];
    char buf[16];
    bool FlgCRC;
    try{
//        Ln = QString::fromLocal8Bit(" Vxod v GetStringSum ");
//        SaveToFile(Ln);
        sum = 0;/* bf[0] = 0x4B; */FlgCRC = false;
        for(int i = 0; i < 9; i++) sum += (int)rBf2[i];//ispravil
1 na 0
//Ln= QString::fromLocal8Bit(" sum= ");
//Ln += QString::number(sum,16);
//SaveToFile(Ln);
//def = sum - (int)bf[0]; //zakomentiroval
sB = QString::number(sum,16); //ispravil def na sum
sB = sB.toUpper();
::sprintf(buf, "%X", rBf2[9]); sP =
QString::fromAscii(buf);
//Ln = QString::fromLocal8Bit(" sp= ");
//Ln +=sP;
//SaveToFile(Ln);
len = sB.length(); //SaveToFile(QString::number(len));
sB = sB.mid(len-2,2);
// SaveToFile(sB); // ispravil 1 na 2
len = sP.length();
//SaveToFile(QString::number(len));
sP = sP.mid(len-2,2);
//SaveToFile(sP);// ispravil 1 na 2
if (sB == sP) FlgCRC = true;
return FlgCRC;
//Ln = QString::fromLocal8Bit(" Vxod v GetStringSum ");
//SaveToFile(Ln);
}
}
```

Продолжение приложения В

```
catch (...) {
    Ln = " <<-- RS Error ";
    Ln += QString::fromLocal8Bit(" Ошибка в процедуре
GetStringSum(Param) ");
    SaveToFile(Ln);
    return false;
}
//
```